

maximatecc.

CCAux

2.4.0.0

Thu May 23 2013

# Contents

<b>1</b>	<b>Main Page</b>	<b>1</b>
1.1	Introduction	1
1.2	Changelog	1
1.2.1	Version 2.4.0.0 - XA/XS, XM platforms	1
1.2.2	Version 2.3.0.0 - XA/XS platform	3
1.2.3	Version 2.2.0.0 - XA/XS platform	3
1.2.4	Version 2.1.0.0 - XA/XS platform	4
1.2.5	Version 2.0.0.0 - XA/XS platform	4
1.3	Known Issues	5
<b>2</b>	<b>Namespace Index</b>	<b>6</b>
2.1	Namespace List	6
<b>3</b>	<b>Data Structure Index</b>	<b>7</b>
3.1	Data Structures	7
<b>4</b>	<b>File Index</b>	<b>8</b>
4.1	File List	8
<b>5</b>	<b>Namespace Documentation</b>	<b>9</b>
5.1	CrossControl Namespace Reference	9
5.1.1	Typedef Documentation	29
5.1.1.1	_PowerMgrConf	29
5.1.1.2	_PowerMgrStatus	29
5.1.1.3	ABOUTHANDLE	29
5.1.1.4	ADCHANDLE	29
5.1.1.5	AUXVERSIONHANDLE	30
5.1.1.6	BACKLIGHTHANDLE	30
5.1.1.7	BATTERYHANDLE	30
5.1.1.8	BUZZERHANDLE	30
5.1.1.9	CANSETTINGHANDLE	30
5.1.1.10	CONFIGHANDLE	30
5.1.1.11	DIAGNOSTICHANDLE	30
5.1.1.12	DIGIOHANDLE	30
5.1.1.13	FIRMWAREUPGHANDLE	30
5.1.1.14	FRONTLEDHANDLE	30
5.1.1.15	LIGHTSENSORHANDLE	30
5.1.1.16	POWERHANDLE	30
5.1.1.17	POWERMGRHANDLE	30

5.1.1.18	SMARTHANDLE	30
5.1.1.19	TELEMATICSHANDLE	30
5.1.1.20	TOUCHSCREENCALIBHANDLE	30
5.1.1.21	TOUCHSCREENHANDLE	30
5.1.1.22	VersionType	30
5.1.1.23	VIDEOHANDLE	30
5.1.2	Enumeration Type Documentation	30
5.1.2.1	ButtonPowerTransitionStatus	30
5.1.2.2	CalibrationConfigParam	31
5.1.2.3	CalibrationModeSettings	31
5.1.2.4	CanFrameType	32
5.1.2.5	CCAuxColor	32
5.1.2.6	CCStatus	32
5.1.2.7	ChargingStatus	32
5.1.2.8	DeInterlaceMode	33
5.1.2.9	eErr	33
5.1.2.10	ErrorStatus	34
5.1.2.11	hwErrorStatusCodes	34
5.1.2.12	JidaSensorType	34
5.1.2.13	LightSensorOperationRange	35
5.1.2.14	LightSensorSamplingMode	35
5.1.2.15	OCDStatus	35
5.1.2.16	PowerAction	36
5.1.2.17	PowerMgrConf	36
5.1.2.18	PowerMgrStatus	36
5.1.2.19	PowerSource	36
5.1.2.20	RS4XXPort	37
5.1.2.21	shutDownReasonCodes	37
5.1.2.22	startupReasonCodes	37
5.1.2.23	TouchScreenModeSettings	37
5.1.2.24	TriggerConf	38
5.1.2.25	TSAdvancedSettingsParameter	38
5.1.2.26	UpgradeAction	39
5.1.2.27	VideoChannel	39
5.1.2.28	VideoRotation	39
5.1.2.29	videoStandard	40
5.1.2.30	VoltageEnum	40
5.1.3	Function Documentation	41
5.1.3.1	About_getAddOnHWversion	41
5.1.3.2	About_getAddOnManufacturingDate	41
5.1.3.3	About_getAddOnPCBArt	42
5.1.3.4	About_getAddOnPCBSerial	42
5.1.3.5	About_getDisplayResolution	43
5.1.3.6	About_getFrontPcbRev	43
5.1.3.7	About_getIOExpanderValue	43
5.1.3.8	About_getIsAnybusMounted	44
5.1.3.9	About_getIsBTMounted	44
5.1.3.10	About_getIsDisplayAvailable	45
5.1.3.11	About_getIsGPRSMounted	45
5.1.3.12	About_getIsGPSPMounted	46

5.1.3.13	About_getIsIOExpanderMounted	46
5.1.3.14	About_getIsTouchScreenAvailable	46
5.1.3.15	About_getIsWLANMounted	47
5.1.3.16	About_getMainHWversion	47
5.1.3.17	About_getMainManufacturingDate	48
5.1.3.18	About_getMainPCBArt	48
5.1.3.19	About_getMainPCBSerial	49
5.1.3.20	About_getMainProdArtNr	49
5.1.3.21	About_getMainProdRev	50
5.1.3.22	About_getNrOfCANConnections	50
5.1.3.23	About_getNrOfDigIOConnections	51
5.1.3.24	About_getNrOfETHConnections	51
5.1.3.25	About_getNrOfSerialConnections	52
5.1.3.26	About_getNrOfUSBConnections	52
5.1.3.27	About_getNrOfVideoConnections	53
5.1.3.28	About_getUnitSerial	53
5.1.3.29	About_hasOsBooted	54
5.1.3.30	About_release	54
5.1.3.31	Adc_getVoltage	55
5.1.3.32	Adc_release	55
5.1.3.33	AuxVersion_getCCAuxDrvVersion	56
5.1.3.34	AuxVersion_getCCAuxVersion	57
5.1.3.35	AuxVersion_getFPGAVersion	58
5.1.3.36	AuxVersion_getFrontVersion	58
5.1.3.37	AuxVersion_getOSVersion	59
5.1.3.38	AuxVersion_getSSVersion	60
5.1.3.39	AuxVersion_release	60
5.1.3.40	Backlight_getAutomaticBLFilter	61
5.1.3.41	Backlight_getAutomaticBLParams	61
5.1.3.42	Backlight_getAutomaticBLStatus	62
5.1.3.43	Backlight_getHWStatus	62
5.1.3.44	Backlight_getIntensity	63
5.1.3.45	Backlight_getLedDimming	63
5.1.3.46	Backlight_getStatus	64
5.1.3.47	Backlight_release	64
5.1.3.48	Backlight_setAutomaticBLFilter	65
5.1.3.49	Backlight_setAutomaticBLParams	65
5.1.3.50	Backlight_setIntensity	66
5.1.3.51	Backlight_setLedDimming	66
5.1.3.52	Backlight_startAutomaticBL	67
5.1.3.53	Backlight_stopAutomaticBL	67
5.1.3.54	Battery_getBatteryChargingStatus	67
5.1.3.55	Battery_getBatteryHWversion	68
5.1.3.56	Battery_getBatterySerial	69
5.1.3.57	Battery_getBatterySwVersion	69
5.1.3.58	Battery_getBatteryTemp	70
5.1.3.59	Battery_getBatteryVoltageStatus	71
5.1.3.60	Battery_getHwErrorStatus	71
5.1.3.61	Battery_getMinMaxTemp	72
5.1.3.62	Battery_getPowerSource	73

5.1.3.63	Battery_getTimer	74
5.1.3.64	Battery_isBatteryPresent	74
5.1.3.65	Battery_release	75
5.1.3.66	Buzzer_buzze	75
5.1.3.67	Buzzer_getFrequency	76
5.1.3.68	Buzzer_getTrigger	76
5.1.3.69	Buzzer_getVolume	77
5.1.3.70	Buzzer_release	77
5.1.3.71	Buzzer_setFrequency	78
5.1.3.72	Buzzer_setTrigger	78
5.1.3.73	Buzzer_setVolume	78
5.1.3.74	CanSetting_getBaudrate	79
5.1.3.75	CanSetting_getFrameType	80
5.1.3.76	CanSetting_release	80
5.1.3.77	CanSetting_setBaudrate	80
5.1.3.78	CanSetting_setFrameType	81
5.1.3.79	Config_getCanStartupPowerConfig	81
5.1.3.80	Config_getExtFanStartupPowerConfig	82
5.1.3.81	Config_getExtOnOffSigTrigTime	82
5.1.3.82	Config_getFrontBtnTrigTime	82
5.1.3.83	Config_getHeatingTempLimit	83
5.1.3.84	Config_getLongButtonPressAction	83
5.1.3.85	Config_getOnOffSigAction	84
5.1.3.86	Config_getPowerOnStartup	84
5.1.3.87	Config_getRS485Enabled	84
5.1.3.88	Config_getShortButtonPressAction	85
5.1.3.89	Config_getStartupTriggerConfig	85
5.1.3.90	Config_getStartupVoltageConfig	86
5.1.3.91	Config_getSuspendMaxTime	86
5.1.3.92	Config_getVideoStartupPowerConfig	87
5.1.3.93	Config_release	87
5.1.3.94	Config_setCanStartupPowerConfig	87
5.1.3.95	Config_setExtFanStartupPowerConfig	88
5.1.3.96	Config_setExtOnOffSigTrigTime	88
5.1.3.97	Config_setFrontBtnTrigTime	89
5.1.3.98	Config_setHeatingTempLimit	89
5.1.3.99	Config_setLongButtonPressAction	89
5.1.3.100	Config_setOnOffSigAction	90
5.1.3.101	Config_setPowerOnStartup	90
5.1.3.102	Config_setRS485Enabled	91
5.1.3.103	Config_setShortButtonPressAction	91
5.1.3.104	Config_setStartupTriggerConfig	92
5.1.3.105	Config_setStartupVoltageConfig	92
5.1.3.106	Config_setSuspendMaxTime	92
5.1.3.107	Config_setVideoStartupPowerConfig	93
5.1.3.108	Diagnostic_clearHwErrorStatus	93
5.1.3.109	Diagnostic_getHwErrorStatus	93
5.1.3.110	Diagnostic_getMinMaxTemp	94
5.1.3.111	Diagnostic_getPCBTemp	94
5.1.3.112	Diagnostic_getPMTemp	95

5.1.3.113 Diagnostic_getPowerCycles . . . . .	95
5.1.3.114 Diagnostic_getShutDownReason . . . . .	95
5.1.3.115 Diagnostic_getSSTemp . . . . .	96
5.1.3.116 Diagnostic_getStartupReason . . . . .	96
5.1.3.117 Diagnostic_getTimer . . . . .	97
5.1.3.118 Diagnostic_release . . . . .	97
5.1.3.119 DigIO_getDigIO . . . . .	98
5.1.3.120 DigIO_release . . . . .	98
5.1.3.121 DigIO_setDigIO . . . . .	99
5.1.3.122 FirmwareUpgrade_getUpgradeStatus . . . . .	99
5.1.3.123 FirmwareUpgrade_release . . . . .	100
5.1.3.124 FirmwareUpgrade_shutDown . . . . .	100
5.1.3.125 FirmwareUpgrade_startFpgaUpgrade . . . . .	100
5.1.3.126 FirmwareUpgrade_startFpgaVerification . . . . .	101
5.1.3.127 FirmwareUpgrade_startFrontUpgrade . . . . .	102
5.1.3.128 FirmwareUpgrade_startFrontVerification . . . . .	103
5.1.3.129 FirmwareUpgrade_startSSUpgrade . . . . .	104
5.1.3.130 FirmwareUpgrade_startSSVerification . . . . .	105
5.1.3.131 FrontLED_getColor . . . . .	106
5.1.3.132 FrontLED_getEnabledDuringStartup . . . . .	107
5.1.3.133 FrontLED_getIdleTime . . . . .	107
5.1.3.134 FrontLED_getNrOfPulses . . . . .	107
5.1.3.135 FrontLED_getOffTime . . . . .	108
5.1.3.136 FrontLED_getOnTime . . . . .	108
5.1.3.137 FrontLED_getSignal . . . . .	109
5.1.3.138 FrontLED_getStandardColor . . . . .	109
5.1.3.139 FrontLED_release . . . . .	110
5.1.3.140 FrontLED_setColor . . . . .	110
5.1.3.141 FrontLED_setEnabledDuringStartup . . . . .	111
5.1.3.142 FrontLED_setIdleTime . . . . .	111
5.1.3.143 FrontLED_setNrOfPulses . . . . .	111
5.1.3.144 FrontLED_setOff . . . . .	112
5.1.3.145 FrontLED_setOffTime . . . . .	112
5.1.3.146 FrontLED_setOnTime . . . . .	112
5.1.3.147 FrontLED_setSignal . . . . .	113
5.1.3.148 FrontLED_setStandardColor . . . . .	113
5.1.3.149 GetAbout . . . . .	114
5.1.3.150 GetAdc . . . . .	114
5.1.3.151 GetAuxVersion . . . . .	115
5.1.3.152 GetBacklight . . . . .	116
5.1.3.153 GetBattery . . . . .	116
5.1.3.154 GetBuzzer . . . . .	116
5.1.3.155 GetCanSetting . . . . .	117
5.1.3.156 GetConfig . . . . .	117
5.1.3.157 GetDiagnostic . . . . .	118
5.1.3.158 GetDigIO . . . . .	118
5.1.3.159 GetErrorStringA . . . . .	119
5.1.3.160 GetErrorStringW . . . . .	119
5.1.3.161 GetFirmwareUpgrade . . . . .	119
5.1.3.162 GetFrontLED . . . . .	120

5.1.3.163	GetHwErrorStatusStringA	120
5.1.3.164	GetHwErrorStatusStringW	120
5.1.3.165	GetLightsensor	121
5.1.3.166	GetPower	121
5.1.3.167	GetPowerMgr	121
5.1.3.168	GetSmart	122
5.1.3.169	GetStartupReasonStringA	123
5.1.3.170	GetStartupReasonStringW	123
5.1.3.171	GetTelematics	123
5.1.3.172	GetTouchScreen	124
5.1.3.173	GetTouchScreenCalib	124
5.1.3.174	GetVideo	124
5.1.3.175	Lightsensor_getAverageIlluminance	125
5.1.3.176	Lightsensor_getIlluminance	125
5.1.3.177	Lightsensor_getIlluminance2	126
5.1.3.178	Lightsensor_getOperatingRange	126
5.1.3.179	Lightsensor_release	126
5.1.3.180	Lightsensor_setOperatingRange	127
5.1.3.181	Lightsensor_startAverageCalc	127
5.1.3.182	Lightsensor_stopAverageCalc	128
5.1.3.183	Power_ackPowerRequest	128
5.1.3.184	Power_getBLPowerStatus	129
5.1.3.185	Power_getButtonPowerTransitionStatus	129
5.1.3.186	Power_getCanOCDStatus	130
5.1.3.187	Power_getCanPowerStatus	130
5.1.3.188	Power_getExtFanPowerStatus	131
5.1.3.189	Power_getVideoOCDStatus	131
5.1.3.190	Power_getVideoPowerStatus	132
5.1.3.191	Power_release	132
5.1.3.192	Power_setBLPowerStatus	133
5.1.3.193	Power_setCanPowerStatus	133
5.1.3.194	Power_setExtFanPowerStatus	134
5.1.3.195	Power_setVideoPowerStatus	134
5.1.3.196	PowerMgr_getConfiguration	134
5.1.3.197	PowerMgr_getPowerMgrStatus	135
5.1.3.198	PowerMgr_hasResumed	137
5.1.3.199	PowerMgr_registerControlledSuspendOrShutDown	139
5.1.3.200	PowerMgr_release	140
5.1.3.201	PowerMgr_setAppReadyForSuspendOrShutdown	141
5.1.3.202	Smart_getDeviceSerial	143
5.1.3.203	Smart_getInitialTime	144
5.1.3.204	Smart_getRemainingLifeTime	144
5.1.3.205	Smart_release	145
5.1.3.206	Telematics_getBTPowerStatus	145
5.1.3.207	Telematics_getBTStartUpPowerStatus	146
5.1.3.208	Telematics_getGPRSPowerStatus	147
5.1.3.209	Telematics_getGPRSStartUpPowerStatus	147
5.1.3.210	Telematics_getGPSAntennaStatus	148
5.1.3.211	Telematics_getGPSPowerStatus	149
5.1.3.212	Telematics_getGPSSStartUpPowerStatus	149

5.1.3.213 Telematics_getTelematicsAvailable . . . . .	150
5.1.3.214 Telematics_getWLANPowerStatus . . . . .	150
5.1.3.215 Telematics_getWLANStartUpPowerStatus . . . . .	151
5.1.3.216 Telematics_release . . . . .	152
5.1.3.217 Telematics_setBTPowerStatus . . . . .	152
5.1.3.218 Telematics_setBTStartUpPowerStatus . . . . .	152
5.1.3.219 Telematics_setGPRSPowerStatus . . . . .	153
5.1.3.220 Telematics_setGPRSStartUpPowerStatus . . . . .	153
5.1.3.221 Telematics_setGPSPowerStatus . . . . .	153
5.1.3.222 Telematics_setGPSStartUpPowerStatus . . . . .	154
5.1.3.223 Telematics_setWLANPowerStatus . . . . .	154
5.1.3.224 Telematics_setWLANStartUpPowerStatus . . . . .	155
5.1.3.225 TouchScreen_getAdvancedSetting . . . . .	155
5.1.3.226 TouchScreen_getMode . . . . .	156
5.1.3.227 TouchScreen_getMouseRightClickTime . . . . .	156
5.1.3.228 TouchScreen_release . . . . .	157
5.1.3.229 TouchScreen_setAdvancedSetting . . . . .	157
5.1.3.230 TouchScreen_setMode . . . . .	158
5.1.3.231 TouchScreen_setMouseRightClickTime . . . . .	158
5.1.3.232 TouchScreenCalib_checkCalibrationPointFinished . . . . .	159
5.1.3.233 TouchScreenCalib_getConfigParam . . . . .	159
5.1.3.234 TouchScreenCalib_getMode . . . . .	159
5.1.3.235 TouchScreenCalib_release . . . . .	160
5.1.3.236 TouchScreenCalib_setCalibrationPoint . . . . .	160
5.1.3.237 TouchScreenCalib_setConfigParam . . . . .	160
5.1.3.238 TouchScreenCalib_setMode . . . . .	161
5.1.3.239 Video_activateSnapshot . . . . .	161
5.1.3.240 Video_createBitmap . . . . .	161
5.1.3.241 Video_freeBmpBuffer . . . . .	162
5.1.3.242 Video_getActiveChannel . . . . .	162
5.1.3.243 Video_getColorKeys . . . . .	163
5.1.3.244 Video_getCropping . . . . .	163
5.1.3.245 Video_getDecoderReg . . . . .	164
5.1.3.246 Video_getDeInterlaceMode . . . . .	164
5.1.3.247 Video_getMirroring . . . . .	164
5.1.3.248 Video_getRawImage . . . . .	165
5.1.3.249 Video_getRotation . . . . .	165
5.1.3.250 Video_getScaling . . . . .	166
5.1.3.251 Video_getStatus . . . . .	166
5.1.3.252 Video_getVideoArea . . . . .	166
5.1.3.253 Video_getVideoStandard . . . . .	167
5.1.3.254 Video_init . . . . .	167
5.1.3.255 Video_minimize . . . . .	168
5.1.3.256 Video_release . . . . .	168
5.1.3.257 Video_restore . . . . .	168
5.1.3.258 Video_setActiveChannel . . . . .	168
5.1.3.259 Video_setColorKeys . . . . .	169
5.1.3.260 Video setCropping . . . . .	169
5.1.3.261 Video_setDecoderReg . . . . .	170
5.1.3.262 Video_setDeInterlaceMode . . . . .	170



5.1.3.263	Video_setMirroring	170
5.1.3.264	Video_setRotation	171
5.1.3.265	Video_setScaling	171
5.1.3.266	Video_setVideoArea	171
5.1.3.267	Video_showFrame	172
5.1.3.268	Video_showVideo	172
5.1.3.269	Video_takeSnapshot	173
5.1.3.270	Video_takeSnapshotBmp	173
5.1.3.271	Video_takeSnapshotRaw	174
5.1.4	Variable Documentation	174
5.1.4.1	DigitalIn_1	174
5.1.4.2	DigitalIn_2	174
5.1.4.3	DigitalIn_3	174
5.1.4.4	DigitalIn_4	174
5.1.4.5	Video1Conf	174
5.1.4.6	Video2Conf	175
5.1.4.7	Video3Conf	175
5.1.4.8	Video4Conf	175
<b>6</b>	<b>Data Structure Documentation</b>	<b>176</b>
6.1	BatteryTimerType Struct Reference	176
6.1.1	Field Documentation	176
6.1.1.1	RunTime_0_40	176
6.1.1.2	RunTime_40_60	176
6.1.1.3	RunTime_60_70	176
6.1.1.4	RunTime_70_80	177
6.1.1.5	RunTime_Above80	177
6.1.1.6	RunTime_m20	177
6.1.1.7	RunTime_m20_0	177
6.1.1.8	TotRunTimeBattery	177
6.1.1.9	TotRunTimeMain	177
6.2	BuzzerSetup Struct Reference	177
6.2.1	Field Documentation	177
6.2.1.1	frequency	177
6.2.1.2	volume	178
6.3	FpgaLedTimingType Struct Reference	178
6.3.1	Field Documentation	178
6.3.1.1	idleTime	178
6.3.1.2	ledNbr	178
6.3.1.3	nrOfPulses	178
6.3.1.4	offTime	178
6.3.1.5	onTime	178
6.4	LedColorMixType Struct Reference	179
6.4.1	Field Documentation	179
6.4.1.1	blue	179
6.4.1.2	green	179
6.4.1.3	red	179
6.5	LedTimingType Struct Reference	179
6.5.1	Field Documentation	179
6.5.1.1	idleTime	179

6.5.1.2	nrOfPulses	180
6.5.1.3	offTime	180
6.5.1.4	onTime	180
6.6	received_video Struct Reference	180
6.6.1	Field Documentation	180
6.6.1.1	received_framerate	180
6.6.1.2	received_height	180
6.6.1.3	received_width	180
6.7	TimerType Struct Reference	180
6.7.1	Detailed Description	181
6.7.2	Field Documentation	181
6.7.2.1	Above80RunTime	181
6.7.2.2	RunTime40_60	181
6.7.2.3	RunTime60_70	181
6.7.2.4	RunTime70_80	181
6.7.2.5	TotHeatTime	181
6.7.2.6	TotRunTime	181
6.7.2.7	TotSuspTime	181
6.8	UpgradeStatus Struct Reference	182
6.8.1	Detailed Description	182
6.8.2	Field Documentation	182
6.8.2.1	currentAction	182
6.8.2.2	errorCode	182
6.8.2.3	percent	182
6.9	version_info Struct Reference	182
6.9.1	Field Documentation	183
6.9.1.1	build	183
6.9.1.2	major	183
6.9.1.3	minor	183
6.9.1.4	release	183
6.10	video_dec_command Struct Reference	183
6.10.1	Field Documentation	183
6.10.1.1	decoder_register	183
6.10.1.2	register_value	183
<b>7</b>	<b>File Documentation</b>	<b>184</b>
7.1	IncludeFiles/About.h File Reference	184
7.2	IncludeFiles/Adc.h File Reference	186
7.3	IncludeFiles/AuxVersion.h File Reference	187
7.4	IncludeFiles/Backlight.h File Reference	188
7.5	IncludeFiles/Battery.h File Reference	189
7.6	IncludeFiles/Buzzer.h File Reference	190
7.7	IncludeFiles/CanSetting.h File Reference	191
7.8	IncludeFiles/CCAuxErrors.h File Reference	192
7.9	IncludeFiles/CCAuxTypes.h File Reference	192
7.10	IncludeFiles/CCPlatform.h File Reference	194
7.11	IncludeFiles/Config.h File Reference	194
7.12	IncludeFiles/Diagnostic.h File Reference	197
7.13	IncludeFiles/DiagnosticCodes.h File Reference	198
7.14	IncludeFiles/DigIO.h File Reference	198

---

7.15 IncludeFiles/FirmwareUpgrade.h File Reference . . . . .	199
7.16 IncludeFiles/FrontLED.h File Reference . . . . .	200
7.17 IncludeFiles/Lightsensor.h File Reference . . . . .	201
7.18 IncludeFiles/Power.h File Reference . . . . .	202
7.19 IncludeFiles/PowerMgr.h File Reference . . . . .	203
7.20 IncludeFiles/Releasenotes.dox File Reference . . . . .	204
7.21 IncludeFiles/Smart.h File Reference . . . . .	204
7.22 IncludeFiles/Telematics.h File Reference . . . . .	205
7.23 IncludeFiles/TouchScreen.h File Reference . . . . .	207
7.24 IncludeFiles/TouchScreenCalib.h File Reference . . . . .	208
7.25 IncludeFiles/Video.h File Reference . . . . .	209

# Chapter 1

## Main Page

### 1.1 Introduction

This documentation is generated from the CCAux source code. CCAux ([CrossControl Common Aux control](#)) is an API that gives access to settings, features and many hardware interfaces; backlight, buzzer, diagnostics, frontled, lightsensor and analog video interfaces.

The API is available for multiple platforms and operating systems: Linux on the C-Cpilot XA, XS and XM products in all variations. For the XM platform, Windows XP and 7 is also supported.

The known issues and changelog presented here also cover the following maximatecc applications (which are using the API and are released in conjunction with it):

- CCSettings
- ccvideo
- ccsettingsconsole
- touchcalibrator
- ccauxd

### 1.2 Changelog

#### 1.2.1 Version 2.4.0.0 - XA/XS, XM platforms

- Removed the following functions: Config\_get/set TFT Mode/Scan/Mirror
- Optimized version queries of different firmware components
- Bugfixes for Backlight and Lightsensor
- The factory defaults settings in CCsettings no longer generates errors

- CCSettings and StartupGUI rebranded for maximatecc
- CCSettings now adapts to the number of CAN ports available
- Added the following function blocks: Battery, PowerMgr and Smart from 1.x API
- XM: CCAux2 is now fully supported on the XM platform with the same functionality as in the 1.6.4.0 release.
- XM: CCAux api 1.6.4.0 will be available for backwards compability. It is compatible back to the 1.3.1.0 release.
- XA/XS: Config\_setRS485Enabled now sets MP\_RS422\_MODE GPIO pins to correct state
- XA/XS: Video\_setMirroring implemented
- XA/XS: Playing two video channels simultaneously now works (1/2+3/4)
- XA/XS: Video can be cropped from left/right for channels 3/4
- XA/XS: Various other improvements for video channels 3/4
- XA/XS: Video standard now reported correctly
- XA/XS: ccvideo context menu now appearing consistently
- XA/XS: ccvideo context menu hanging now fixed
- XA/XS: ccvideo blanking now fixed
- XA/XS: ccvideo now handles rotation
- XA/XS: ccsettingsconsole now up to date
- XA/XS: Context menu no longer opened while calibrating
- XA/XS: The PowerOnAtStartup setting ("Always start when power turned on" in CCsettings) was always read as Enabled
- XA/XS: 1V2 is now a supported ADC channel on some instances
- XA/XS: Added TS\_TCHAUTOCAL in TouchScreen class
- ccauxd: Fixed issues that caused crash when shutting the daemon off
- ccauxd: Added support for PowerMgr

### 1.2.2 Version 2.3.0.0 - XA/XS platform

- Functions added: Backlight\_getHWStatus, Config\_getRS485Enabled and Config\_setRS485Enabled
- CCSettings: Led tab improved
- CCSettings: Hide unsupported options in Power tab
- CCSettings: Hide suspend options if unsupported by HW
- CCSettings: Fixed rotation glitches
- Bugfixes
- Known issues:
  - Same as 2.2.0.0 release, plus:
  - ccvideo: Rightclick (long press) menu not appearing consistently

### 1.2.3 Version 2.2.0.0 - XA/XS platform

- Functions added: About\_getIsAnybusMounted, Config\_setTFTMode, Config\_getTFTMode, Video\_showFrame and About\_getIOExpanderValue
- Fixed rotation issues with GUI applications
- Many bugfixes
- Known issues:
  - When automatic backlight is enabled, updating SS or Front uC software is very slow and may fail. Workaround: Make sure automatic backlight is disabled before attempting to do any firmware upgrade.
  - CCSettings - Advanced: After Firmware update, the shutdown button does not work. Workaround: Turn off power to the device.
  - Qt application issue, when QWS application is started from command line (# app -qws)
    - \* When plugging out USB input device, the application will hang. Workaround: # app -qws < /dev/zero
  - Some info/functions are missing from ccsettingsconsole
  - About\_hasOsBooted can return true even when not all drivers have not been loaded (API)
  - Calling Video\_showVideo for ports 3/4 will not return if no camera is attached
  - Cannot show analog video from two ports simultaneously (1/2+3/4), trying to do so leads to crash
  - For ports 3/4, video sometimes scrolls or has wrong size when starting the application first time

- API calls for analog video currently not supported: get/setMirroring, get/setCropping (for ports 3/4), get/setDeInterlaceMode, get/setScaling, get/setColorKeys
- ccvideo: Selecting "Mirror image" does not have an effect

#### 1.2.4 Version 2.1.0.0 - XA/XS platform

- Functions added: Power\_getVideoOCDStatus, Power\_getCanOCDStatus and About\_hasOsBooted
- Touch calibration can be started from CCSettings
- 7" touch calibration now supported
- Many bugfixes
- Known issues:
  - About\_hasOsBooted can return true even when not all drivers have not been loaded
  - Analog video API only supports VIDEO1/2 ports
  - Video control only supports positioning and resizing
  - The factory defaults button in the Advanced tab in CCSettings produces some error messages. These can be ignored

#### 1.2.5 Version 2.0.0.0 - XA/XS platform

- Initial release
- The CCAux API v1.x from the CCpilot XM platform has been rewritten to ensure compability between releases
- Porting to CCpilot XA/XS platform nearly complete. Some new platform specific functions remain to be implemented
- The API gives access to several hardware interfaces, for example backlight, buzzer, diagnostics, frontled, lightsensor and analog video interfaces
- Known issues:
  - Digital input/output does not work correctly
  - CAN settings interface does not work
  - Analog video API only supports VIDEO1/2 ports
  - Video control only supports positioning and resizing
  - SS/Front software update - sometimes crashes before update has begun. When this happens (segmentation fault or Open failed error), restart the unit and try again
  - Font issue in CCSettings causes some text to disappear

- TouchCalibrator cannot be started from within CCSettings. Instead it can be started manually: # TouchCalibrator -qws
- The factory defaults button in the Advanced tab in CCSettings produces some error messages. These can be ignored
- Error messages related to automatic backlight will show the very first time the Display tab in CCsettings is opened. These can be ignored.
- GetHWErrorStatusString functions do not return correct description of error messages

### **1.3 Known Issues**

- XA/XS: API calls for analog video currently not supported: get/setDeInterlaceMode, get/setScaling, get/setColorKeys
- XA/XS: ccvideostream: de-interlacing artifacts with certain output window sizes



## Chapter 2

# Namespace Index

### 2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">CrossControl</a> . . . . .	9
--	---

## Chapter 3

# Data Structure Index

### 3.1 Data Structures

Here are the data structures with brief descriptions:

BatteryTimerType	176
BuzzerSetup	177
FpgaLedTimingType	178
LedColorMixType	179
LedTimingType	179
received_video	180
TimerType	180
UpgradeStatus	182
version_info	182
video_dec_command	183

# Chapter 4

## File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

<a href="#">IncludeFiles/About.h</a>	184
<a href="#">IncludeFiles/Adc.h</a>	186
<a href="#">IncludeFiles/AuxVersion.h</a>	187
<a href="#">IncludeFiles/Backlight.h</a>	188
<a href="#">IncludeFiles/Battery.h</a>	189
<a href="#">IncludeFiles/Buzzer.h</a>	190
<a href="#">IncludeFiles/CanSetting.h</a>	191
<a href="#">IncludeFiles/CCAuxErrors.h</a>	192
<a href="#">IncludeFiles/CCAuxTypes.h</a>	192
<a href="#">IncludeFiles/CCPlatform.h</a>	194
<a href="#">IncludeFiles/Config.h</a>	194
<a href="#">IncludeFiles/Diagnostic.h</a>	197
<a href="#">IncludeFiles/DiagnosticCodes.h</a>	198
<a href="#">IncludeFiles/DigIO.h</a>	198
<a href="#">IncludeFiles/FirmwareUpgrade.h</a>	199
<a href="#">IncludeFiles/FrontLED.h</a>	200
<a href="#">IncludeFiles/Lightsensor.h</a>	201
<a href="#">IncludeFiles/Power.h</a>	202
<a href="#">IncludeFiles/PowerMgr.h</a>	203
<a href="#">IncludeFiles/Smart.h</a>	204
<a href="#">IncludeFiles/Telematics.h</a>	205
<a href="#">IncludeFiles/TouchScreen.h</a>	207
<a href="#">IncludeFiles/TouchScreenCalib.h</a>	208
<a href="#">IncludeFiles/Video.h</a>	209

## Chapter 5

# Namespace Documentation

### 5.1 CrossControl Namespace Reference

#### Data Structures

- struct [BatteryTimerType](#)
- struct [received\\_video](#)
- struct [video\\_dec\\_command](#)
- struct [version\\_info](#)
- struct [BuzzerSetup](#)
- struct [LedTimingType](#)
- struct [FpgaLedTimingType](#)
- struct [LedColorMixType](#)
- struct [TimerType](#)
- struct [UpgradeStatus](#)

#### Typedefs

- typedef void \* [ABOUTHANDLE](#)
- typedef void \* [ADCHANDLE](#)
- typedef void \* [AUXVERSIONHANDLE](#)
- typedef void \* [BACKLIGHTHANDLE](#)
- typedef void \* [BATTERYHANDLE](#)
- typedef void \* [BUZZERHANDLE](#)
- typedef void \* [CANSETTINGHANDLE](#)
- typedef struct [version\\_info](#) [VersionType](#)
- typedef void \* [CONFIGHANDLE](#)
- typedef void \* [DIAGNOSTICHANDLE](#)
- typedef void \* [DIGIOHANDLE](#)
- typedef void \* [FIRMWAREUPGHANDLE](#)
- typedef void \* [FRONTLEDHANDLE](#)

- typedef void \* LIGHTSENSORHANDLE
- typedef void \* POWERHANDLE
- typedef enum  
CrossControl::PowerMgrConf \_PowerMgrConf
- typedef enum  
CrossControl::PowerMgrStatus \_PowerMgrStatus
- typedef void \* POWERMGRHANDLE
- typedef void \* SMARTHANDLE
- typedef void \* TELEMATICSHANDLE
- typedef void \* TOUCHSCREENHANDLE
- typedef void \* TOUCHSCREENCALIBHANDLE
- typedef void \* VIDEOHANDLE

### Enumerations

- enum ChargingStatus {  
ChargingStatus\_NoCharge = 0, ChargingStatus\_Charging = 1, ChargingStatus-  
\_FullyCharged = 2, ChargingStatus\_TempLow = 3,  
ChargingStatus\_TempHigh = 4, ChargingStatus\_Unknown = 5 }
- enum PowerSource { PowerSource\_Battery = 0, PowerSource\_ExternalPower =  
1 }
- enum ErrorStatus {  
ErrorStatus\_NoError = 0, ErrorStatus\_ThermistorTempSensor = 1, ErrorStatus-  
\_SecondaryTempSensor = 2, ErrorStatus\_ChargeFail = 3,  
ErrorStatus\_Overcurrent = 4, ErrorStatus\_Init = 5 }
- enum VoltageEnum {  
VOLTAGE\_24VIN = 0, VOLTAGE\_24V, VOLTAGE\_12V, VOLTAGE\_12-  
VID,  
VOLTAGE\_5V, VOLTAGE\_3V3, VOLTAGE\_VTFT, VOLTAGE\_5VSTB,  
VOLTAGE\_1V9, VOLTAGE\_1V8, VOLTAGE\_1V5, VOLTAGE\_1V2,  
VOLTAGE\_1V05, VOLTAGE\_1V0, VOLTAGE\_0V9, VOLTAGE\_VREF\_I-  
NT,  
VOLTAGE\_24V\_BACKUP, VOLTAGE\_2V5, VOLTAGE\_1V1, VOLTAGE-  
\_1V3\_PER,  
VOLTAGE\_1V3\_VDDA }
- enum LightSensorOperationRange { RangeStandard = 0, RangeExtended = 1 }
- enum LightSensorSamplingMode { SamplingModeStandard = 0, SamplingMode-  
Extended, SamplingModeAuto }
- enum CCStatus { Disabled = 0, Enabled = 1 }
- enum eErr {  
ERR\_SUCCESS = 0, ERR\_OPEN\_FAILED = 1, ERR\_NOT\_SUPPORTED =  
2, ERR\_UNKNOWN\_FEATURE = 3,  
ERR\_DATATYPE\_MISMATCH = 4, ERR\_CODE\_NOT\_EXIST = 5, ERR\_-  
BUFFER\_SIZE = 6, ERR\_IOCTL\_FAILED = 7,  
ERR\_INVALID\_DATA = 8, ERR\_INVALID\_PARAMETER = 9, ERR\_CRE-  
ATE\_THREAD = 10, ERR\_IN\_PROGRESS = 11,  
ERR\_CHECKSUM = 12, ERR\_INIT\_FAILED = 13, ERR\_VERIFY\_FAILED

- = 14, ERR\_DEVICE\_READ\_DATA\_FAILED = 15,
- ERR\_DEVICE\_WRITE\_DATA\_FAILED = 16, ERR\_COMMAND\_FAILED = 17, ERR\_EEPROM = 18, ERR\_JIDA\_TEMP = 19,
- ERR\_AVERAGE\_CALC\_STARTED = 20, ERR\_NOT\_RUNNING = 21, ERR\_I2C\_EXPANDER\_READ\_FAILED = 22, ERR\_I2C\_EXPANDER\_WRITE\_FAILED = 23,
- ERR\_I2C\_EXPANDER\_INIT\_FAILED = 24, ERR\_NEWER\_SS\_VERSION\_REQUIRED = 25, ERR\_NEWER\_FPGA\_VERSION\_REQUIRED = 26, ERR\_NEWER\_FRONT\_VERSION\_REQUIRED = 27,
- ERR\_TELEMATICS\_GPRS\_NOT\_AVAILABLE = 28, ERR\_TELEMATICS\_WLAN\_NOT\_AVAILABLE = 29, ERR\_TELEMATICS\_BT\_NOT\_AVAILABLE = 30, ERR\_TELEMATICS\_GPS\_NOT\_AVAILABLE = 31,
- ERR\_MEM\_ALLOC\_FAIL = 32, ERR\_JOIN\_THREAD = 33 }
- enum DeInterlaceMode { DeInterlace\_Even = 0, DeInterlace\_Odd = 1, DeInterlace\_BOB = 2 }
- enum VideoChannel { Analog\_Channel\_1 = 0, Analog\_Channel\_2 = 1, Analog\_Channel\_3 = 2, Analog\_Channel\_4 = 3 }
- enum videoStandard { STD\_M\_J\_NTSC = 0, STD\_B\_D\_G\_H\_I\_N\_PAL = 1, STD\_M\_PAL = 2, STD\_D\_PAL = 3, STD\_NTSC = 4, STD\_SECAM = 5 }
- enum VideoRotation { RotNone = 0, Rot90, Rot180, Rot270 }
- enum CanFrameType { FrameStandard, FrameExtended, FrameStandardExtended }
- enum TriggerConf { Front\_Button\_Enabled = 1, OnOff\_Signal\_Enabled = 2, Both\_Button\_And\_Signal\_Enabled = 3 }
- enum PowerAction { NoAction = 0, ActionSuspend = 1, ActionShutDown = 2 }
- enum ButtonPowerTransitionStatus { BPTS\_No\_Change = 0, BPTS\_ShutDown = 1, BPTS\_Suspend = 2, BPTS\_Restart = 3, BPTS\_BtnPressed = 4, BPTS\_BtnPressedLong = 5, BPTS\_SignalOff = 6 }
- enum OCDStatus { OCD\_OK = 0, OCD\_OC = 1, OCD\_POWER\_OFF = 2 }
- enum JidaSensorType { TEMP\_CPU = 0, TEMP\_BOX = 1, TEMP\_ENV = 2, TEMP\_BOARD = 3, TEMP\_BACKPLANE = 4, TEMP\_CHIPSETS = 5, TEMP\_VIDEO = 6, TEMP\_OTHER = 7 }
- enum UpgradeAction { UPGRADE\_INIT, UPGRADE\_PREP\_COM, UPGRADE\_READING\_FILE, UPGRADE\_CONVERTING\_FILE, UPGRADE\_FLASHING, UPGRADE\_VERIFYING, UPGRADE\_COMPLETE, UPGRADE\_COMPLETE\_WITH\_ERRORS }
- enum CCAuxColor { RED = 0, GREEN, BLUE, CYAN, MAGENTA, YELLOW, UNDEFINED\_COLOR }
- enum RS4XXPort { RS4XXPort1 = 1, RS4XXPort2, RS4XXPort3, RS4XXPort4 }
- enum startupReasonCodes { startupReasonCodeUndefined = 0x0000, startupReasonCodeButtonPress = 0x0055,

- `startupReasonCodeExtCtrl = 0x00AA, startupReasonCodeMPRestart = 0x00F0, startupReasonCodePowerOnStartup = 0x000F }`
- enum `shutDownReasonCodes` { `shutdownReasonCodeNoError = 0x001F` }
- enum `hwErrorStatusCodes` { `errCodeNoErr = 0` }
- enum `PowerMgrConf` { `Normal = 0, ApplicationControlled = 1, BatterySuspend = 2` }
- enum `PowerMgrStatus` { `NoRequestsPending = 0, SuspendPending = 1, ShutdownPending = 2` }
- enum `TouchScreenModeSettings` { `MOUSE_NEXT_BOOT = 0, TOUCH_NEXT_BOOT = 1, MOUSE_NOW = 2, TOUCH_NOW = 3` }
- enum `TSAdvancedSettingsParameter` {  
`TS_RIGHT_CLICK_TIME = 0, TS_LOW_LEVEL = 1, TS_UNTOUCHLEVEL = 2, TS_DEBOUNCE_TIME = 3,`  
`TS_DEBOUNCE_TIMEOUT_TIME = 4, TS_DOUBLECLICK_MAX_CLICK_TIME = 5, TS_DOUBLE_CLICK_TIME = 6, TS_MAX_RIGHTCLICK_DISTANCE = 7,`  
`TS_USE_DEJITTER = 8, TS_CALIBTATION_WIDTH = 9, TS_CALIBRATION_MEASUREMENTS = 10, TS_RESTORE_DEFAULT_SETTINGS = 11,`  
`TS_TCHAUTOCAL = 12` }
- enum `CalibrationModeSettings` {  
`MODE_UNKNOWN = 0, MODE_NORMAL = 1, MODE_CALIBRATION_5P = 2, MODE_CALIBRATION_9P = 3,`  
`MODE_CALIBRATION_13P = 4` }
- enum `CalibrationConfigParam` {  
`CONFIG_CALIBRATION_WITH = 0, CONFIG_CALIBRATION_MEASUREMENTS = 1, CONFIG_5P_CALIBRATION_POINT_BORDER = 2, CONFIG_13P_CALIBRATION_POINT_BORDER = 3,`  
`CONFIG_13P_CALIBRATION_TRANSITION_MIN = 4, CONFIG_13P_CALIBRATION_TRANSITION_MAX = 5` }

## Functions

- `EXTERN_C CCAUXDLL_API ABOUTHANDLE CCAUXDLL_CALLING_CONV GetAbout (void)`
- `EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV About_release (ABOUTHANDLE)`
- `EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV About_getMainPCBSerial (ABOUTHANDLE, char *buff, int len)`
- `EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV About_getUnitSerial (ABOUTHANDLE, char *buff, int len)`
- `EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV About_getMainPCBArt (ABOUTHANDLE, char *buff, int length)`
- `EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV About_getMainManufacturingDate (ABOUTHANDLE, char *buff, int len)`

- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getMainHWversion](#) (ABOOUTHANDLE, char \*buff, int len)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getMainProdRev](#) (ABOOUTHANDLE, char \*buff, int len)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getMainProdArtNr](#) (ABOOUTHANDLE, char \*buff, int len)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getNrOfETHConnections](#) (ABOOUTHANDLE, unsigned char \*NrOfConnections)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getNrOfCANConnections](#) (ABOOUTHANDLE, unsigned char \*NrOfConnections)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getNrOfVideoConnections](#) (ABOOUTHANDLE, unsigned char \*NrOfConnections)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getNrOfUSBConnections](#) (ABOOUTHANDLE, unsigned char \*NrOfConnections)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getNrOfSerialConnections](#) (ABOOUTHANDLE, unsigned char \*NrOfConnections)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getNrOfDigIOConnections](#) (ABOOUTHANDLE, unsigned char \*NrOfConnections)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getIsDisplayAvailable](#) (ABOOUTHANDLE, bool \*available)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getIsTouchScreenAvailable](#) (ABOOUTHANDLE, bool \*available)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getDisplayResolution](#) (ABOOUTHANDLE, char \*buff, int len)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getAddOnPCBSerial](#) (ABOOUTHANDLE, char \*buff, int len)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getAddOnPCBArt](#) (ABOOUTHANDLE, char \*buff, int length)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getAddOnManufacturingDate](#) (ABOOUTHANDLE, char \*buff, int len)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getAddOnHWversion](#) (ABOOUTHANDLE, char \*buff, int len)



- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getIsWLANMounted](#) (ABOOUTHANDLE, bool \*mounted)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getIsGPSPMounted](#) (ABOOUTHANDLE, bool \*mounted)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getIsGPRSPMounted](#) (ABOOUTHANDLE, bool \*mounted)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getIsBTMounted](#) (ABOOUTHANDLE, bool \*mounted)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getFrontPcbRev](#) (ABOOUTHANDLE, unsigned char \*major, unsigned char \*minor)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getIsIOExpanderMounted](#) (ABOOUTHANDLE, bool \*mounted)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getIOExpanderValue](#) (ABOOUTHANDLE, unsigned short \*value)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_hasOsBooted](#) (ABOOUTHANDLE, bool \*bootComplete)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getIsAnybusMounted](#) (ABOOUTHANDLE, bool \*mounted)
- EXTERN\_C CCAUXDLL\_API  
[ADCHANDLE](#)  
CCAUXDLL\_CALLING\_CONV [GetAdc](#) (void)
- EXTERN\_C CCAUXDLL\_API void  
CCAUXDLL\_CALLING\_CONV [Adc\\_release](#) (ADCHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Adc\\_getVoltage](#) (ADCHANDLE, VoltageEnum selection, double \*value)
- EXTERN\_C CCAUXDLL\_API  
[AUXVERSIONHANDLE](#)  
CCAUXDLL\_CALLING\_CONV [GetAuxVersion](#) (void)
- EXTERN\_C CCAUXDLL\_API void  
CCAUXDLL\_CALLING\_CONV [AuxVersion\\_release](#) (AUXVERSIONHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [AuxVersion\\_getFPGAVersion](#) (AUXVERSIONHANDLE, unsigned char \*major, unsigned char \*minor, unsigned char \*release, unsigned char \*build)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [AuxVersion\\_getSSVersion](#) (AUXVERSIONHANDLE, unsigned char \*major, unsigned char \*minor, unsigned char \*release, unsigned char \*build)

- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [AuxVersion\\_getFrontVersion](#) (AUXVERSIONHANDLE, unsigned char \*major, unsigned char \*minor, unsigned char \*release, unsigned char \*build)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [AuxVersion\\_getCCAuxVersion](#) (AUXVERSIONHANDLE, unsigned char \*major, unsigned char \*minor, unsigned char \*release, unsigned char \*build)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [AuxVersion\\_getOSVersion](#) (AUXVERSIONHANDLE, unsigned char \*major, unsigned char \*minor, unsigned char \*release, unsigned char \*build)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [AuxVersion\\_getCCAuxDrvVersion](#) (AUXVERSIONHANDLE, unsigned char \*major, unsigned char \*minor, unsigned char \*release, unsigned char \*build)
- EXTERN\_C CCAUXDLL\_API  
[BACKLIGHTHANDLE](#)  
CCAUXDLL\_CALLING\_CONV [GetBacklight](#) (void)
- EXTERN\_C CCAUXDLL\_API void  
CCAUXDLL\_CALLING\_CONV [Backlight\\_release](#) (BACKLIGHTHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Backlight\\_getIntensity](#) (BACKLIGHTHANDLE, unsigned char \*intensity)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Backlight\\_setIntensity](#) (BACKLIGHTHANDLE, unsigned char intensity)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Backlight\\_getStatus](#) (BACKLIGHTHANDLE, unsigned char \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Backlight\\_getHWStatus](#) (BACKLIGHTHANDLE, bool \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Backlight\\_startAutomaticBL](#) (BACKLIGHTHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Backlight\\_stopAutomaticBL](#) (BACKLIGHTHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Backlight\\_getAutomaticBLStatus](#) (BACKLIGHTHANDLE, unsigned char \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Backlight\\_setAutomaticBLParams](#) (BACKLIGHTHANDLE, bool bSoftTransitions)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Backlight\\_getAutomaticBLParams](#) (BACKLIGHTHANDLE, bool \*bSoftTransitions, double \*k)

- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Backlight\\_setAutomaticBLFilter](#) (BACKLIGHTHANDLE, unsigned long averageWndSize, unsigned long rejectWndSize, unsigned long rejectDeltaInLux, [LightSensorSamplingMode](#) mode)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Backlight\\_getAutomaticBLFilter](#) (BACKLIGHTHANDLE, unsigned long \*averageWndSize, unsigned long \*rejectWndSize, unsigned long \*rejectDeltaInLux, [LightSensorSamplingMode](#) \*mode)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Backlight\\_getLedDimming](#) (BACKLIGHTHANDLE, [CCStatus](#) \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Backlight\\_setLedDimming](#) (BACKLIGHTHANDLE, [CCStatus](#) status)
- EXTERN\_C CCAUXDLL\_API  
[BATTERYHANDLE](#)  
CCAUXDLL\_CALLING\_CONV [GetBattery](#) (void)
- EXTERN\_C CCAUXDLL\_API void  
CCAUXDLL\_CALLING\_CONV [Battery\\_release](#) ([BATTERYHANDLE](#))
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Battery\\_isBatteryPresent](#) ([BATTERYHANDLE](#), bool \*batteryIsPresent)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Battery\\_getBatteryVoltageStatus](#) ([BATTERYHANDLE](#), unsigned char \*batteryVoltagePercent)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Battery\\_getBatteryChargingStatus](#) ([BATTERYHANDLE](#), [ChargingStatus](#) \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Battery\\_getPowerSource](#) ([BATTERYHANDLE](#), [PowerSource](#) \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Battery\\_getBatteryTemp](#) ([BATTERYHANDLE](#), signed short \*temperature)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Battery\\_getHwErrorStatus](#) ([BATTERYHANDLE](#), [ErrorStatus](#) \*errorCode)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Battery\\_getTimer](#) ([BATTERYHANDLE](#), [Battery-TimerType](#) \*times)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Battery\\_getMinMaxTemp](#) ([BATTERYHANDLE](#), signed short \*minTemp, signed short \*maxTemp)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Battery\\_getBatteryHWversion](#) ([BATTERYHANDLE](#), char \*buff, int len)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Battery\\_getBatterySwVersion](#) ([BATTERYHANDLE](#), unsigned short \*major, unsigned short \*minor, unsigned short \*release, unsigned short \*build)

- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Battery\\_getBatterySerial](#) (BATTERYHANDLE, char \*buff, int len)
- EXTERN\_C CCAUXDLL\_API  
[BUZZERHANDLE](#)  
CCAUXDLL\_CALLING\_CONV [GetBuzzer](#) (void)
- EXTERN\_C CCAUXDLL\_API void  
CCAUXDLL\_CALLING\_CONV [Buzzer\\_release](#) (BUZZERHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Buzzer\\_getFrequency](#) (BUZZERHANDLE, unsigned short \*frequency)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Buzzer\\_getVolume](#) (BUZZERHANDLE, unsigned short \*volume)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Buzzer\\_getTrigger](#) (BUZZERHANDLE, bool \*trigger)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Buzzer\\_setFrequency](#) (BUZZERHANDLE, unsigned short frequency)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Buzzer\\_setVolume](#) (BUZZERHANDLE, unsigned short volume)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Buzzer\\_setTrigger](#) (BUZZERHANDLE, bool trigger)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Buzzer\\_buzze](#) (BUZZERHANDLE, int time, bool blocking)
- EXTERN\_C CCAUXDLL\_API  
[CANSETTINGHANDLE](#)  
CCAUXDLL\_CALLING\_CONV [GetCanSetting](#) (void)
- EXTERN\_C CCAUXDLL\_API void  
CCAUXDLL\_CALLING\_CONV [CanSetting\\_release](#) (CANSETTINGHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [CanSetting\\_getBaudrate](#) (CANSETTINGHANDLE, unsigned char net, unsigned short \*baudrate)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [CanSetting\\_getFrameType](#) (CANSETTINGHANDLE, unsigned char net, [CanFrameType](#) \*frameType)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [CanSetting\\_setBaudrate](#) (CANSETTINGHANDLE, unsigned char net, unsigned short baudrate)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [CanSetting\\_setFrameType](#) (CANSETTINGHANDLE, unsigned char net, [CanFrameType](#) frameType)
- EXTERN\_C CCAUXDLL\_API char  
const \*CCAUXDLL\_CALLING\_CONV [GetErrorStringA](#) (eErr errCode)

- EXTERN\_C CCAUXDLL\_API wchar\_t  
const \*CCAUXDLL\_CALLING\_CONV [GetErrorStringW](#) (eErr errCode)
- EXTERN\_C CCAUXDLL\_API  
[CONFIGHANDLE](#)  
CCAUXDLL\_CALLING\_CONV [GetConfig](#) ()
- EXTERN\_C CCAUXDLL\_API void  
CCAUXDLL\_CALLING\_CONV [Config\\_release](#) ([CONFIGHANDLE](#))
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_getStartupTriggerConfig](#) ([CONFIGHANDLE](#), [TriggerConf](#) \*config)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_getShortButtonPressAction](#) ([CONFIGHANDLE](#), [PowerAction](#) \*action)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_getLongButtonPressAction](#) ([CONFIGHANDLE](#), [PowerAction](#) \*action)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_getOnOffSigAction](#) ([CONFIGHANDLE](#), [PowerAction](#) \*action)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_getFrontBtnTrigTime](#) ([CONFIGHANDLE](#), unsigned short \*triggertime)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_getExtOnOffSigTrigTime](#) ([CONFIGHANDLE](#), unsigned long \*triggertime)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_getSuspendMaxTime](#) ([CONFIGHANDLE](#), unsigned short \*maxTime)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_getCanStartupPowerConfig](#) ([CONFIGHANDLE](#), [CCStatus](#) \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_getVideoStartupPowerConfig](#) ([CONFIGHANDLE](#), unsigned char \*config)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_getExtFanStartupPowerConfig](#) ([CONFIGHANDLE](#), [CCStatus](#) \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_getStartupVoltageConfig](#) ([CONFIGHANDLE](#), double \*voltage)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_getHeatingTempLimit](#) ([CONFIGHANDLE](#), signed short \*temperature)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_getPowerOnStartup](#) ([CONFIGHANDLE](#), [CCStatus](#) \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_setStartupTriggerConfig](#) ([CONFIGHANDLE](#), [TriggerConf](#) conf)

- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV Config\_setShortButtonPressAction (CONFIGHANDLE, PowerAction action)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV Config\_setLongButtonPressAction (CONFIGHANDLE, PowerAction action)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV Config\_setOnOffSigAction (CONFIGHANDLE, PowerAction action)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV Config\_setFrontBtnTrigTime (CONFIGHANDLE, unsigned short triggertime)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV Config\_setExtOnOffSigTrigTime (CONFIGHANDLE, unsigned long triggertime)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV Config\_setSuspendMaxTime (CONFIGHANDLE, unsigned short maxTime)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV Config\_setCanStartupPowerConfig (CONFIGHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV Config\_setVideoStartupPowerConfig (CONFIGHANDLE, unsigned char config)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV Config\_setExtFanStartupPowerConfig (CONFIGHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV Config\_setStartupVoltageConfig (CONFIGHANDLE, double voltage)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV Config\_setHeatingTempLimit (CONFIGHANDLE, signed short temperature)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV Config\_setPowerOnStartup (CONFIGHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV Config\_setRS485Enabled (CONFIGHANDLE, RS4XXPort port, bool enabled)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV Config\_getRS485Enabled (CONFIGHANDLE, RS4XXPort port, bool \*enabled)
- EXTERN\_C CCAUXDLL\_API  
DIAGNOSTICHANDLE  
CCAUXDLL\_CALLING\_CONV GetDiagnostic (void)
- EXTERN\_C CCAUXDLL\_API void  
CCAUXDLL\_CALLING\_CONV Diagnostic\_release (DIAGNOSTICHANDLE)

- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Diagnostic\\_getSSTemp](#) (DIAGNOSTICHANDLE, signed short \*temperature)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Diagnostic\\_getPCBTemp](#) (DIAGNOSTICHANDLE, signed short \*temperature)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Diagnostic\\_getPMTemp](#) (DIAGNOSTICHANDLE, unsigned char index, signed short \*temperature, [JidaSensorType](#) \*jst)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Diagnostic\\_getStartupReason](#) (DIAGNOSTICHANDLE, unsigned short \*reason)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Diagnostic\\_getShutDownReason](#) (DIAGNOSTICHANDLE, unsigned short \*reason)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Diagnostic\\_getHwErrorStatus](#) (DIAGNOSTICHANDLE, unsigned short \*errorCode)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Diagnostic\\_getTimer](#) (DIAGNOSTICHANDLE, [TimerType](#) \*times)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Diagnostic\\_getMinMaxTemp](#) (DIAGNOSTICHANDLE, signed short \*minTemp, signed short \*maxTemp)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Diagnostic\\_getPowerCycles](#) (DIAGNOSTICHANDLE, unsigned short \*powerCycles)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Diagnostic\\_clearHwErrorStatus](#) (DIAGNOSTICHANDLE)
- EXTERN\_C CCAUXDLL\_API char  
const \*CCAUXDLL\_CALLING\_CONV [GetHwErrorStatusStringA](#) (unsigned short errorCode)
- EXTERN\_C CCAUXDLL\_API wchar\_t  
const \*CCAUXDLL\_CALLING\_CONV [GetHwErrorStatusStringW](#) (unsigned short errorCode)
- EXTERN\_C CCAUXDLL\_API char  
const \*CCAUXDLL\_CALLING\_CONV [GetStartupReasonStringA](#) (unsigned short code)
- EXTERN\_C CCAUXDLL\_API wchar\_t  
const \*CCAUXDLL\_CALLING\_CONV [GetStartupReasonStringW](#) (unsigned short code)
- EXTERN\_C CCAUXDLL\_API  
[DIGIOHANDLE](#)  
CCAUXDLL\_CALLING\_CONV [GetDigIO](#) (void)
- EXTERN\_C CCAUXDLL\_API void  
CCAUXDLL\_CALLING\_CONV [DigIO\\_release](#) ([DIGIOHANDLE](#))

- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [DigIO\\_getDigIO](#) (DIGIOHANDLE, unsigned char \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [DigIO\\_setDigIO](#) (DIGIOHANDLE, unsigned char state)
- EXTERN\_C CCAUXDLL\_API  
[FIRMWAREUPGHANDLE](#)  
CCAUXDLL\_CALLING\_CONV [GetFirmwareUpgrade](#) (void)
- EXTERN\_C CCAUXDLL\_API void  
CCAUXDLL\_CALLING\_CONV [FirmwareUpgrade\\_release](#) (FIRMWAREUPGHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [FirmwareUpgrade\\_startFpgaUpgrade](#) (FIRMWAREUPGHANDLE, const char \*filename, bool blocking)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [FirmwareUpgrade\\_startFpgaVerification](#) (FIRMWAREUPGHANDLE, const char \*filename, bool blocking)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [FirmwareUpgrade\\_startSSUpgrade](#) (FIRMWAREUPGHANDLE, const char \*filename, bool blocking)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [FirmwareUpgrade\\_startSSVerification](#) (FIRMWAREUPGHANDLE, const char \*filename, bool blocking)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [FirmwareUpgrade\\_startFrontUpgrade](#) (FIRMWAREUPGHANDLE, const char \*filename, bool blocking)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [FirmwareUpgrade\\_startFrontVerification](#) (FIRMWAREUPGHANDLE, const char \*filename, bool blocking)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [FirmwareUpgrade\\_getUpgradeStatus](#) (FIRMWAREUPGHANDLE, UpgradeStatus \*status, bool blocking)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [FirmwareUpgrade\\_shutDown](#) (FIRMWAREUPGHANDLE)
- EXTERN\_C CCAUXDLL\_API  
[FRONTLEDHANDLE](#)  
CCAUXDLL\_CALLING\_CONV [GetFrontLED](#) (void)
- EXTERN\_C CCAUXDLL\_API void  
CCAUXDLL\_CALLING\_CONV [FrontLED\\_release](#) (FRONTLEDHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [FrontLED\\_getSignal](#) (FRONTLEDHANDLE, double \*frequency, unsigned char \*dutyCycle)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [FrontLED\\_getOnTime](#) (FRONTLEDHANDLE, unsigned char \*onTime)



- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV FrontLED\_getOffTime (FRONTLEDHANDLE, unsigned char \*offTime)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV FrontLED\_getIdleTime (FRONTLEDHANDLE, unsigned char \*idleTime)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV FrontLED\_getNrOfPulses (FRONTLEDHANDLE, unsigned char \*nrOfPulses)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV FrontLED\_getColor (FRONTLEDHANDLE, unsigned char \*red, unsigned char \*green, unsigned char \*blue)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV FrontLED\_getStandardColor (FRONTLEDHANDLE, CCAuxColor \*color)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV FrontLED\_getEnabledDuringStartup (FRONTLEDHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV FrontLED\_setSignal (FRONTLEDHANDLE, double frequency, unsigned char dutyCycle)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV FrontLED\_setOnTime (FRONTLEDHANDLE, unsigned char onTime)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV FrontLED\_setOffTime (FRONTLEDHANDLE, unsigned char offTime)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV FrontLED\_setIdleTime (FRONTLEDHANDLE, unsigned char idleTime)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV FrontLED\_setNrOfPulses (FRONTLEDHANDLE, unsigned char nrOfPulses)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV FrontLED\_setColor (FRONTLEDHANDLE, unsigned char red, unsigned char green, unsigned char blue)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV FrontLED\_setStandardColor (FRONTLEDHANDLE, CCAuxColor color)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV FrontLED\_setOff (FRONTLEDHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV FrontLED\_setEnabledDuringStartup (FRONTLEDHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API  
LIGHTSENSORHANDLE  
CCAUXDLL\_CALLING\_CONV GetLightsensor (void)

- EXTERN\_C CCAUXDLL\_API void  
CCAUXDLL\_CALLING\_CONV [Lightsensor\\_release](#) ([LIGHTSENSORHANDLE](#))
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Lightsensor\\_getIlluminance](#) ([LIGHTSENSORHANDLE](#), unsigned short \*value)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Lightsensor\\_getIlluminance2](#) ([LIGHTSENSORHANDLE](#), unsigned short \*value, unsigned char \*ch0, unsigned char \*ch1)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Lightsensor\\_getAverageIlluminance](#) ([LIGHTSENSORHANDLE](#), unsigned short \*value)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Lightsensor\\_startAverageCalc](#) ([LIGHTSENSORHANDLE](#), unsigned long averageWndSize, unsigned long rejectWndSize, unsigned long rejectDeltaInLux, [LightSensorSamplingMode](#) mode)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Lightsensor\\_stopAverageCalc](#) ([LIGHTSENSORHANDLE](#))
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Lightsensor\\_getOperatingRange](#) ([LIGHTSENSORHANDLE](#), [LightSensorOperationRange](#) \*range)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Lightsensor\\_setOperatingRange](#) ([LIGHTSENSORHANDLE](#), [LightSensorOperationRange](#) range)
- EXTERN\_C CCAUXDLL\_API  
[POWERHANDLE](#)  
CCAUXDLL\_CALLING\_CONV [GetPower](#) (void)
- EXTERN\_C CCAUXDLL\_API void  
CCAUXDLL\_CALLING\_CONV [Power\\_release](#) ([POWERHANDLE](#))
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Power\\_getBLPowerStatus](#) ([POWERHANDLE](#), [CCStatus](#) \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Power\\_getCanPowerStatus](#) ([POWERHANDLE](#), [CCStatus](#) \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Power\\_getVideoPowerStatus](#) ([POWERHANDLE](#), unsigned char \*videoStatus)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Power\\_getExtFanPowerStatus](#) ([POWERHANDLE](#), [CCStatus](#) \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Power\\_getButtonPowerTransitionStatus](#) ([POWERHANDLE](#), [ButtonPowerTransitionStatus](#) \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Power\\_getVideoOCDSStatus](#) ([POWERHANDLE](#), [OCDSStatus](#) \*status)

- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV Power\_getCanOCDStatus (POWERHANDLE, OCDStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV Power\_setBLPowerStatus (POWERHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV Power\_setCanPowerStatus (POWERHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV Power\_setVideoPowerStatus (POWERHANDLE, unsigned char status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV Power\_setExtFanPowerStatus (POWERHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV Power\_ackPowerRequest (POWERHANDLE)
- EXTERN\_C CCAUXDLL\_API  
POWERMGRHANDLE  
CCAUXDLL\_CALLING\_CONV GetPowerMgr (void)
- EXTERN\_C CCAUXDLL\_API void  
CCAUXDLL\_CALLING\_CONV PowerMgr\_release (POWERMGRHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV PowerMgr\_registerControlledSuspendOrShutdown (POWERMGRHANDLE, PowerMgrConf conf)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV PowerMgr\_getConfiguration (POWERMGRHANDLE, PowerMgrConf \*conf)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV PowerMgr\_getPowerMgrStatus (POWERMGRHANDLE, PowerMgrStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV PowerMgr\_setAppReadyForSuspendOrShutdown (POWERMGRHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV PowerMgr\_hasResumed (POWERMGRHANDLE, bool \*resumed)
- EXTERN\_C CCAUXDLL\_API  
SMARTHANDLE  
CCAUXDLL\_CALLING\_CONV GetSmart (void)
- EXTERN\_C CCAUXDLL\_API void  
CCAUXDLL\_CALLING\_CONV Smart\_release (SMARTHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV Smart\_getRemainingLifeTime (SMARTHANDLE, unsigned char \*lifetimepercent)

- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Smart\\_getDeviceSerial](#) (SMARTHANDLE, char \*buff, int len)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Smart\\_getInitialTime](#) (SMARTHANDLE, time\_t \*time)
- EXTERN\_C CCAUXDLL\_API  
[TELEMATICSHANDLE](#)  
CCAUXDLL\_CALLING\_CONV [GetTelematics](#) (void)
- EXTERN\_C CCAUXDLL\_API void  
CCAUXDLL\_CALLING\_CONV [Telematics\\_release](#) (TELEMATICSHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Telematics\\_getTelematicsAvailable](#) (TELEMATICSHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Telematics\\_getGPRSPowerStatus](#) (TELEMATICSHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Telematics\\_getGPRSStartupPowerStatus](#) (TELEMATICSHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Telematics\\_getWLANPowerStatus](#) (TELEMATICSHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Telematics\\_getWLANStartupPowerStatus](#) (TELEMATICSHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Telematics\\_getBTPowerStatus](#) (TELEMATICSHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Telematics\\_getBTStartupPowerStatus](#) (TELEMATICSHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Telematics\\_getGPSPowerStatus](#) (TELEMATICSHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Telematics\\_getGPSStartupPowerStatus](#) (TELEMATICSHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Telematics\\_getGPSAntennaStatus](#) (TELEMATICSHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Telematics\\_setGPRSPowerStatus](#) (TELEMATICSHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Telematics\\_setGPRSStartupPowerStatus](#) (TELEMATICSHANDLE, CCStatus status)

- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV Telematics\_setWLANPowerStatus (TELEMATICSHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV Telematics\_setWLANStartUpPowerStatus (TELEMATICSHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV Telematics\_setBTPowerStatus (TELEMATICSHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV Telematics\_setBTStartUpPowerStatus (TELEMATICSHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV Telematics\_setGPSPowerStatus (TELEMATICSHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV Telematics\_setGPSStartUpPowerStatus (TELEMATICSHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API  
TOUCHSCREENHANDLE  
CCAUXDLL\_CALLING\_CONV GetTouchScreen (void)
- EXTERN\_C CCAUXDLL\_API void  
CCAUXDLL\_CALLING\_CONV TouchScreen\_release (TOUCHSCREENHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV TouchScreen\_getMode (TOUCHSCREENHANDLE, TouchScreenModeSettings \*config)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV TouchScreen\_getMouseRightClickTime (TOUCHSCREENHANDLE, unsigned short \*time)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV TouchScreen\_setMode (TOUCHSCREENHANDLE, TouchScreenModeSettings config)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV TouchScreen\_setMouseRightClickTime (TOUCHSCREENHANDLE, unsigned short time)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV TouchScreen\_setAdvancedSetting (TOUCHSCREENHANDLE, TSAdvancedSettingsParameter param, unsigned short data)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV TouchScreen\_getAdvancedSetting (TOUCHSCREENHANDLE, TSAdvancedSettingsParameter param, unsigned short \*data)
- EXTERN\_C CCAUXDLL\_API  
TOUCHSCREENCALIBHANDLE  
CCAUXDLL\_CALLING\_CONV GetTouchScreenCalib (void)
- EXTERN\_C CCAUXDLL\_API void  
CCAUXDLL\_CALLING\_CONV TouchScreenCalib\_release (TOUCHSCREENCALIBHANDLE)

- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [TouchScreenCalib\\_setMode](#) ([TOUCHSCREENCALIBHANDLE](#), [CalibrationModeSettings](#) mode)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [TouchScreenCalib\\_getMode](#) ([TOUCHSCREENCALIBHANDLE](#), [CalibrationModeSettings](#) \*mode)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [TouchScreenCalib\\_setCalibrationPoint](#) ([TOUCHSCREENCALIBHANDLE](#), unsigned char pointNr)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [TouchScreenCalib\\_checkCalibrationPointFinished](#) ([TOUCHSCREENCALIBHANDLE](#), bool \*finished, unsigned char pointNr)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [TouchScreenCalib\\_getConfigParam](#) ([TOUCHSCREENCALIBHANDLE](#), [CalibrationConfigParam](#) param, unsigned short \*value)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [TouchScreenCalib\\_setConfigParam](#) ([TOUCHSCREENCALIBHANDLE](#), [CalibrationConfigParam](#) param, unsigned short value)
- EXTERN\_C CCAUXDLL\_API  
[VIDEOHANDLE](#)  
CCAUXDLL\_CALLING\_CONV [GetVideo](#) (void)
- EXTERN\_C CCAUXDLL\_API void  
CCAUXDLL\_CALLING\_CONV [Video\\_release](#) ([VIDEOHANDLE](#))
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_init](#) ([VIDEOHANDLE](#), unsigned char deviceNr)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_showVideo](#) ([VIDEOHANDLE](#), bool show)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_setDeInterlaceMode](#) ([VIDEOHANDLE](#), [DeInterlaceMode](#) mode)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_getDeInterlaceMode](#) ([VIDEOHANDLE](#), [DeInterlaceMode](#) \*mode)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_setMirroring](#) ([VIDEOHANDLE](#), [CC-Status](#) mode)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_getMirroring](#) ([VIDEOHANDLE](#), [CC-Status](#) \*mode)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_setRotation](#) ([VIDEOHANDLE](#), [Video-Rotation](#) rotation)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_getRotation](#) ([VIDEOHANDLE](#), [Video-Rotation](#) \*rotation)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_setActiveChannel](#) ([VIDEOHANDLE](#), [VideoChannel](#) channel)

- [EXTERN\\_C CCAUXDLL\\_API eErr](#)  
CCAUXDLL\_CALLING\_CONV [Video\\_getActiveChannel](#) (VIDEOHANDLE, VideoChannel \*channel)
- [EXTERN\\_C CCAUXDLL\\_API eErr](#)  
CCAUXDLL\_CALLING\_CONV [Video\\_setColorKeys](#) (VIDEOHANDLE, unsigned char rKey, unsigned char gKey, unsigned char bKey)
- [EXTERN\\_C CCAUXDLL\\_API eErr](#)  
CCAUXDLL\_CALLING\_CONV [Video\\_getColorKeys](#) (VIDEOHANDLE, unsigned char \*rKey, unsigned char \*gKey, unsigned char \*bKey)
- [EXTERN\\_C CCAUXDLL\\_API eErr](#)  
CCAUXDLL\_CALLING\_CONV [Video\\_setVideoArea](#) (VIDEOHANDLE, unsigned short topLeftX, unsigned short topLeftY, unsigned short bottomRightX, unsigned short bottomRightY)
- [EXTERN\\_C CCAUXDLL\\_API eErr](#)  
CCAUXDLL\_CALLING\_CONV [Video\\_getRawImage](#) (VIDEOHANDLE, unsigned short \*width, unsigned short \*height, float \*frameRate)
- [EXTERN\\_C CCAUXDLL\\_API eErr](#)  
CCAUXDLL\_CALLING\_CONV [Video\\_getVideoArea](#) (VIDEOHANDLE, unsigned short \*topLeftX, unsigned short \*topLeftY, unsigned short \*bottomRightX, unsigned short \*bottomRightY)
- [EXTERN\\_C CCAUXDLL\\_API eErr](#)  
CCAUXDLL\_CALLING\_CONV [Video\\_getVideoStandard](#) (VIDEOHANDLE, videoStandard \*standard)
- [EXTERN\\_C CCAUXDLL\\_API eErr](#)  
CCAUXDLL\_CALLING\_CONV [Video\\_getStatus](#) (VIDEOHANDLE, unsigned char \*status)
- [EXTERN\\_C CCAUXDLL\\_API eErr](#)  
CCAUXDLL\_CALLING\_CONV [Video\\_setScaling](#) (VIDEOHANDLE, float x, float y)
- [EXTERN\\_C CCAUXDLL\\_API eErr](#)  
CCAUXDLL\_CALLING\_CONV [Video\\_getScaling](#) (VIDEOHANDLE, float \*x, float \*y)
- [EXTERN\\_C CCAUXDLL\\_API eErr](#)  
CCAUXDLL\_CALLING\_CONV [Video\\_activateSnapshot](#) (VIDEOHANDLE, bool activate)
- [EXTERN\\_C CCAUXDLL\\_API eErr](#)  
CCAUXDLL\_CALLING\_CONV [Video\\_takeSnapshot](#) (VIDEOHANDLE, const char \*path, bool bInterlaced)
- [EXTERN\\_C CCAUXDLL\\_API eErr](#)  
CCAUXDLL\_CALLING\_CONV [Video\\_takeSnapshotRaw](#) (VIDEOHANDLE, char \*rawImgBuffer, unsigned long rawImgBuffSize, bool bInterlaced)
- [EXTERN\\_C CCAUXDLL\\_API eErr](#)  
CCAUXDLL\_CALLING\_CONV [Video\\_takeSnapshotBmp](#) (VIDEOHANDLE, char \*\*bmpBuffer, unsigned long \*bmpBufSize, bool bInterlaced, bool bNTSCFormat)
- [EXTERN\\_C CCAUXDLL\\_API eErr](#)  
CCAUXDLL\_CALLING\_CONV [Video\\_createBitmap](#) (VIDEOHANDLE, char \*\*bmpBuffer, unsigned long \*bmpBufSize, const char \*rawImgBuffer, unsigned long rawImgBufSize, bool bInterlaced, bool bNTSCFormat)

- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_freeBmpBuffer](#) (VIDEOHANDLE, char \*bmpBuffer)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_minimize](#) (VIDEOHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_restore](#) (VIDEOHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_setDecoderReg](#) (VIDEOHANDLE, unsigned char decoderRegister, unsigned char registerValue)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_getDecoderReg](#) (VIDEOHANDLE, unsigned char decoderRegister, unsigned char \*registerValue)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_setCropping](#) (VIDEOHANDLE, unsigned char top, unsigned char left, unsigned char bottom, unsigned char right)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_getCropping](#) (VIDEOHANDLE, unsigned char \*top, unsigned char \*left, unsigned char \*bottom, unsigned char \*right)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_showFrame](#) (VIDEOHANDLE)

## Variables

- const unsigned char [Video1Conf](#) = (1 << 0)
- const unsigned char [Video2Conf](#) = (1 << 1)
- const unsigned char [Video3Conf](#) = (1 << 2)
- const unsigned char [Video4Conf](#) = (1 << 3)
- const unsigned char [DigitalIn\\_1](#) = (1 << 0)
- const unsigned char [DigitalIn\\_2](#) = (1 << 1)
- const unsigned char [DigitalIn\\_3](#) = (1 << 2)
- const unsigned char [DigitalIn\\_4](#) = (1 << 3)

### 5.1.1 Typedef Documentation

#### 5.1.1.1 typedef enum CrossControl::PowerMgrConf \_PowerMgrConf

Enumeration of the settings that can be used with the PowerMgr system.

#### 5.1.1.2 typedef enum CrossControl::PowerMgrStatus \_PowerMgrStatus

#### 5.1.1.3 typedef void\* ABOUTHANDLE

#### 5.1.1.4 typedef void\* ADCHANDLE



- 5.1.1.5 typedef void\* AUXVERSIONHANDLE
- 5.1.1.6 typedef void\* BACKLIGHTHANDLE
- 5.1.1.7 typedef void\* BATTERYHANDLE
- 5.1.1.8 typedef void\* BUZZERHANDLE
- 5.1.1.9 typedef void\* CANSETTINGHANDLE
- 5.1.1.10 typedef void\* CONFIGHANDLE
- 5.1.1.11 typedef void\* DIAGNOSTICHANDLE
- 5.1.1.12 typedef void\* DIGIOHANDLE
- 5.1.1.13 typedef void\* FIRMWAREUPGHANDLE
- 5.1.1.14 typedef void\* FRONTLEDHANDLE
- 5.1.1.15 typedef void\* LIGHTSENSORHANDLE
- 5.1.1.16 typedef void\* POWERHANDLE
- 5.1.1.17 typedef void\* POWERMGRHANDLE
- 5.1.1.18 typedef void\* SMARTHANDLE
- 5.1.1.19 typedef void\* TELEMATICSHANDLE
- 5.1.1.20 typedef void\* TOUCHSCREENCALIBHANDLE
- 5.1.1.21 typedef void\* TOUCHSCREENHANDLE
- 5.1.1.22 typedef struct version\_info VersionType
- 5.1.1.23 typedef void\* VIDEOHANDLE

## 5.1.2 Enumeration Type Documentation

### 5.1.2.1 enum ButtonPowerTransitionStatus

Current status for front panel button and on/off signal. If any of them generate a suspend or shutdown event, it can also be read, briefly. When the button/signal is released, typically BPTS\_Suspend or BPTS\_ShutDown follows.

Enumerator

*BPTS\_No\_Change* No change

***BPTS\_ShutDown*** A shutdown has been initiated since the front panel button has been pressed longer than the set `FrontBtnShutDownTrigTime`

***BPTS\_Suspend*** Suspend mode has been initiated since the front panel button has been pressed (shortly) and suspend mode is enabled

***BPTS\_Restart*** Not currently in use

***BPTS\_BtnPressed*** The front panel button is currently pressed. It has not been released and it has not yet been held longer than `FrontBtnShutDownTrigTime`.

***BPTS\_BtnPressedLong*** The front panel button is currently pressed. It has not been released and it has been held longer than `FrontBtnShutDownTrigTime`.

***BPTS\_SignalOff*** The external on/off signal is low, but not yet long enough for the `ExtOnOffSigSuspTrigTime`.

#### 5.1.2.2 enum CalibrationConfigParam

Touch screen calibration parameters

Enumerator

***CONFIG\_CALIBRATION\_WITH***

***CONFIG\_CALIBRATION\_MEASUREMENTS*** Accepted error value when calibrating.

***CONFIG\_5P\_CALIBRATION\_POINT\_BORDER*** Number of measurements to accept a calibration point.

***CONFIG\_13P\_CALIBRATION\_POINT\_BORDER*** The number of pixels from the border where the 5 point calibration points should be located.

***CONFIG\_13P\_CALIBRATION\_TRANSITION\_MIN*** The number of pixels from the border where the 13 point calibration points should be located.

***CONFIG\_13P\_CALIBRATION\_TRANSITION\_MAX*** Min defines the transition area in number of pixels, where the two different calibrations are used.

#### 5.1.2.3 enum CalibrationModeSettings

Touch screen calibration modes

Enumerator

***MODE\_UNKNOWN***

***MODE\_NORMAL*** Unknown mode.

***MODE\_CALIBRATION\_5P*** Normal operation mode.

***MODE\_CALIBRATION\_9P*** Calibration with 5 points mode.

***MODE\_CALIBRATION\_13P*** Calibration with 9 points mode.

## 5.1.2.4 enum CanFrameType

Can frame type settings

Enumerator

*FrameStandard*

*FrameExtended*

*FrameStandardExtended*

## 5.1.2.5 enum CCAuxColor

Enumeration of standard colors

Enumerator

*RED*

*GREEN* RGB 0xF, 0x0, 0x0

*BLUE* RGB 0x0, 0xF, 0x0

*CYAN* RGB 0x0, 0x0, 0xF

*MAGENTA* RGB 0x0, 0xF, 0xF

*YELLOW* RGB 0xF, 0x0, 0xF

*UNDEFINED\_COLOR* RGB 0xF, 0xF, 0x0

Returns if color is not a standard color

## 5.1.2.6 enum CCStatus

Enable/disable enumeration

Enumerator

*Disabled*

*Enabled* The setting is disabled or turned off

## 5.1.2.7 enum ChargingStatus

Current charging status of the battery.

Enumerator

*ChargingStatus\_NoCharge* The battery is not being charged. System is running on battery power.

*ChargingStatus\_Charging* The battery is currently being charged

*ChargingStatus\_FullyCharged* The battery is fully charged

***ChargingStatus\_TempLow*** The temperature is too low to allow the battery to be charged

***ChargingStatus\_TempHigh*** The temperature is too high to allow the battery to be charged

***ChargingStatus\_Unknown*** There was an error determining the charging status

#### 5.1.2.8 enum DeInterlaceMode

Enumerator

***DeInterlace\_Even***

***DeInterlace\_Odd*** Use only even rows from the interlaced input stream

***DeInterlace\_BOB*** Use only odd rows from the interlaced input stream

#### 5.1.2.9 enum eErr

Error code enumeration

Enumerator

***ERR\_SUCCESS***

***ERR\_OPEN\_FAILED*** Success

***ERR\_NOT\_SUPPORTED*** Open failed

***ERR\_UNKNOWN\_FEATURE*** Not supported

***ERR\_DATATYPE\_MISMATCH*** Unknown feature

***ERR\_CODE\_NOT\_EXIST*** Datatype mismatch

***ERR\_BUFFER\_SIZE*** Code doesn't exist

***ERR\_IOCTL\_FAILED*** Buffer size error

***ERR\_INVALID\_DATA*** IoCtrl operation failed

***ERR\_INVALID\_PARAMETER*** Invalid data

***ERR\_CREATE\_THREAD*** Invalid parameter

***ERR\_IN\_PROGRESS*** Failed to create thread

***ERR\_CHECKSUM*** Operation in progress

***ERR\_INIT\_FAILED*** Checksum error

***ERR\_VERIFY\_FAILED*** Initialization failed

***ERR\_DEVICE\_READ\_DATA\_FAILED*** Failed to verify

***ERR\_DEVICE\_WRITE\_DATA\_FAILED*** Failed to read from device

***ERR\_COMMAND\_FAILED*** Failed to write to device

***ERR\_EEPROM*** Command failed

***ERR\_JIDA\_TEMP*** Error in EEPROM memory

***ERR\_AVERAGE\_CALC\_STARTED*** Failed to get JIDA temperature

***ERR\_NOT\_RUNNING*** Calculation already started  
***ERR\_I2C\_EXPANDER\_READ\_FAILED*** Thread isn't running  
***ERR\_I2C\_EXPANDER\_WRITE\_FAILED*** I2C read failure  
***ERR\_I2C\_EXPANDER\_INIT\_FAILED*** I2C write failure  
***ERR\_NEWER\_SS\_VERSION\_REQUIRED*** I2C initialization failure  
***ERR\_NEWER\_FPGA\_VERSION\_REQUIRED*** SS version too old  
***ERR\_NEWER\_FRONT\_VERSION\_REQUIRED*** FPGA version too old  
***ERR\_TELEMATICS\_GPRS\_NOT\_AVAILABLE*** FRONT version too old  
***ERR\_TELEMATICS\_WLAN\_NOT\_AVAILABLE*** GPRS module not available  
  
***ERR\_TELEMATICS\_BT\_NOT\_AVAILABLE*** WLAN module not available  
***ERR\_TELEMATICS\_GPS\_NOT\_AVAILABLE*** Bluetooth module not available  
  
***ERR\_MEM\_ALLOC\_FAIL*** GPS module not available  
***ERR\_JOIN\_THREAD*** Failed to allocate memory

#### 5.1.2.10 enum ErrorStatus

Enumerator

***ErrorStatus\_NoError***  
***ErrorStatus\_ThermistorTempSensor***  
***ErrorStatus\_SecondaryTempSensor***  
***ErrorStatus\_ChargeFail***  
***ErrorStatus\_Overcurrent***  
***ErrorStatus\_Init***

#### 5.1.2.11 enum hwErrorStatusCodes

The error codes returned by getHWErrorStatus.

Enumerator

***errCodeNoErr***

#### 5.1.2.12 enum JidaSensorType

Jida temperature sensor types

Enumerator

***TEMP\_CPU***

*TEMP\_BOX*  
*TEMP\_ENV*  
*TEMP\_BOARD*  
*TEMP\_BACKPLANE*  
*TEMP\_CHIPSETS*  
*TEMP\_VIDEO*  
*TEMP\_OTHER*

#### 5.1.2.13 enum LightSensorOperationRange

Light sensor operation ranges.

Enumerator

*RangeStandard*  
*RangeExtended* Light sensor operation range standard

#### 5.1.2.14 enum LightSensorSamplingMode

Light sensor sampling modes.

Enumerator

*SamplingModeStandard*  
*SamplingModeExtended* Standard sampling mode.  
*SamplingModeAuto* Extended sampling mode.  
Auto switch between standard and extended sampling mode depending on saturation.

#### 5.1.2.15 enum OCDStatus

Overcurrent detection status.

Enumerator

*OCD\_OK* Normal operation, no overcurrent condition detected  
*OCD\_OC* Overcurrent has been detected, power has therefore been turned off, but may be functioning again if the problem disappeared after re-test  
*OCD\_POWER\_OFF* Overcurrent has been detected, power has been permanently turned off

## 5.1.2.16 enum PowerAction

Button and on/off signal actions.

Enumerator

- NoAction* No action taken
- ActionSuspend* The system enters suspend mode
- ActionShutDown* The system shuts down

## 5.1.2.17 enum PowerMgrConf

Enumeration of the settings that can be used with the PowerMgr system.

Enumerator

- Normal* Applications will not be able to delay suspend/shutdown requests. This is the normal configuration that is used when the PowerMgr class is not being used. Setting this configuration turns off the feature if it is in use.
- ApplicationControlled* Applications can delay suspend/shutdown requests.
- BatterySuspend* In this mode, the computer will automatically enter suspend mode when the unit starts running on battery power. Applications can delay suspend/shutdown requests. This mode is only applicable if the unit has an external battery. Using this configuration on a computer without an external battery will be the same as using the configuration ApplicationControlled.

## 5.1.2.18 enum PowerMgrStatus

Enumerator

- NoRequestsPending* No suspend or shutdown requests.
- SuspendPending* A suspend request is pending.
- ShutdownPending* A shutdown request is pending.

## 5.1.2.19 enum PowerSource

Current power source of the computer.

Enumerator

- PowerSource\_Battery*
- PowerSource\_ExternalPower*

## 5.1.2.20 enum RS4XXPort

Enumeration of RS4XX ports (1-4)

Enumerator

*RS4XXPort1*

*RS4XXPort2*

*RS4XXPort3*

*RS4XXPort4*

## 5.1.2.21 enum shutDownReasonCodes

The shutdown codes returned by getShutDownReason.

Enumerator

*shutdownReasonCodeNoError*

## 5.1.2.22 enum startupReasonCodes

The restart codes returned by getStartupReason.

Enumerator

*startupReasonCodeUndefined*

*startupReasonCodeButtonPress* Unknown startup reason.

*startupReasonCodeExtCtrl* The system was started by front panel button press

*startupReasonCodeMPRestart* The system was started by the external control signal

*startupReasonCodePowerOnStartup* The system was restarted by OS request

## 5.1.2.23 enum TouchScreenModeSettings

Touch screen USB profile settings

Enumerator

*MOUSE\_NEXT\_BOOT*

*TOUCH\_NEXT\_BOOT* Set the touch USB profile to mouse profile. Active upon the next boot.

*MOUSE\_NOW* Set the touch USB profile to touch profile. Active upon the next boot.

*TOUCH\_NOW* Immediately set the touch USB profile to mouse profile.



### 5.1.2.24 enum TriggerConf

Trigger configuration enumeration. Valid settings for enabling of front button and external on/off signal.

#### Enumerator

***Front\_Button\_Enabled*** Front button is enabled for startup and wake-up

***OnOff\_Signal\_Enabled*** The external on/off signal is enabled for startup and wake-up

***Both\_Button\_And\_Signal\_Enabled*** Both of the above are enabled

### 5.1.2.25 enum TSAdvancedSettingsParameter

Touch screen advanced settings parameters

#### Enumerator

***TS\_RIGHT\_CLICK\_TIME*** Right click time in ms, except for touch profile on XM platform

***TS\_LOW\_LEVEL*** Lowest A/D value required for registering a touch event. Front uc 0.5.3.1 had the default value of 3300, newer versions: 3400.

***TS\_UNTOUCHLEVEL*** A/D value where the screen is considered to be untouched.

***TS\_DEBOUNCE\_TIME*** Debounce time is the time after first detected touch event during which no measurements are being taken. This is used to avoid faulty measurements that frequently happens right after the actual touch event. Front uc 0.5.3.1 had the default value of 3ms, newer versions: 24ms.

***TS\_DEBOUNCE\_TIMEOUT\_TIME*** After debounce, an event will be ignored if after this time there are no valid measurements above *TS\_LOW\_LEVEL*. This time must be larger than *TS\_DEBOUNCE\_TIME*. Front uc 0.5.3.1 had the default value of 12ms, newer versions: 36ms.

***TS\_DOUBLECLICK\_MAX\_CLICK\_TIME*** Parameter used for improving double click accuracy. A touch event this long or shorter is considered to be one of the clicks in a double click.

***TS\_DOUBLE\_CLICK\_TIME*** Parameter used for improving double click accuracy. Time allowed between double clicks. Used for double click improvement.

***TS\_MAX\_RIGHTCLICK\_DISTANCE*** Maximum distance allowed to move pointer and still consider the event a right click.

***TS\_USE\_DEJITTER*** The dejitter function enables smoother pointer movement. Set to non-zero to enable the function or zero to disable it.

***TS\_CALIBTATION\_WIDTH*** Accepted difference in measurement during calibration of a point.

***TS\_CALIBRATION\_MEASUREMENTS*** Number of measurements needed to accept a calibration point.

***TS\_RESTORE\_DEFAULT\_SETTINGS*** Set to non-zero to restore all the above settings to their defaults. This parameter cannot be read and setting it to zero has no effect.

***TS\_TCHAUTOCAL*** Time (in units of 200 ms) until the touch screen is recalibrated when continuously touching the screen at one point. A setting of zero disables the recalibration. Valid for PCAP touch panels only. Device must be restarted for changes to have any effect. The default value is 50 which corresponds to 10 seconds.

#### 5.1.2.26 enum UpgradeAction

Upgrade Action enumeration

Enumerator

***UPGRADE\_INIT***

***UPGRADE\_PREP\_COM*** Initiating, checking for compatibility etc

***UPGRADE\_READING\_FILE*** Preparing communication

***UPGRADE\_CONVERTING\_FILE*** Opening and reading the supplied file

***UPGRADE\_FLASHING*** Converting the mcs format to binary format

***UPGRADE\_VERIFYING*** Flashing the file

***UPGRADE\_COMPLETE*** Verifying the programmed image

***UPGRADE\_COMPLETE\_WITH\_ERRORS*** Upgrade was finished

Upgrade finished prematurely, see errorCode for the reason of failure

#### 5.1.2.27 enum VideoChannel

The available analog video channels

Enumerator

***Analog\_Channel\_1***

***Analog\_Channel\_2***

***Analog\_Channel\_3***

***Analog\_Channel\_4***

#### 5.1.2.28 enum VideoRotation

Enumerator

***RotNone***

***Rot90***

***Rot180***

***Rot270***

## 5.1.2.29 enum videoStandard

Enumerator

***STD\_M\_J\_NTSC***  
***STD\_B\_D\_G\_H\_I\_N\_PAL*** (M,J) NTSC ITU-R BT.601  
***STD\_M\_PAL*** (B, D, G, H, I, N) PAL ITU-R BT.601  
***STD\_PAL*** (M) PAL ITU-R BT.601  
***STD\_NTSC*** PAL-Nc ITU-R BT.601  
***STD\_SECAM*** NTSC 4.43 ITU-R BT.601

## 5.1.2.30 enum VoltageEnum

Voltage type enumeration

Enumerator

***VOLTAGE\_24VIN***  
***VOLTAGE\_24V*** < 24VIN  
***VOLTAGE\_12V*** < 24V  
***VOLTAGE\_12VID*** < 12V  
***VOLTAGE\_5V*** < 12VID  
***VOLTAGE\_3V3*** < 5V  
***VOLTAGE\_VTFT*** < 3.3V  
***VOLTAGE\_5VSTB*** < VTFT  
***VOLTAGE\_1V9*** < 5VSTB  
***VOLTAGE\_1V8*** < 1.9V  
***VOLTAGE\_1V5*** < 1.8V  
***VOLTAGE\_1V2*** < 1.5V  
***VOLTAGE\_1V05*** < 1.2V  
***VOLTAGE\_1V0*** < 1.05V  
***VOLTAGE\_0V9*** < 1.0V  
***VOLTAGE\_VREF\_INT*** < 0.9V  
***VOLTAGE\_24V\_BACKUP*** < SS internal VRef  
***VOLTAGE\_2V5*** < 24V backup capacitor  
***VOLTAGE\_1V1*** < 2.5V  
***VOLTAGE\_1V3\_PER*** < 1.1V  
***VOLTAGE\_1V3\_VDDA*** < 1.3V\_PER  
< 1.3V\_VDDA

### 5.1.3 Function Documentation

#### 5.1.3.1 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV

CrossControl::About\_getAddOnHWversion ( ABOUTHANDLE , char \* *buff*, int *len* )

Get Add on hardware version.

Supported Platform(s): XM, XA, XS

##### Parameters

<i>buff</i>	Text output buffer.
<i>len</i>	Maximum length of the output buffer. If the actual length of the data is greater, an error will be returned.

##### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

##### Example Usage:

```
err = About_getAddOnHWversion (pAbout, buffer, buffer_len);
if (CrossControl::ERR_SUCCESS == err)
    cout << "Add on hardware version: " << buffer << endl;
```

#### 5.1.3.2 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV

CrossControl::About\_getAddOnManufacturingDate ( ABOUTHANDLE , char \* *buff*, int *len* )

Get Add on manufacturing date.

Supported Platform(s): XM, XA, XS

##### Parameters

<i>buff</i>	Text output buffer.
<i>len</i>	Maximum length of the output buffer. If the actual length of the data is greater, an error will be returned.

##### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

##### Example Usage:

```
err = About_getAddOnManufacturingDate (pAbout, buffer, buffer_len);
if (CrossControl::ERR_SUCCESS == err)
    cout << "Add on manufacturing date: " << buffer << endl;
```

## 5.1.3.3 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV

CrossControl::About\_getAddOnPCBArt ( ABOUTHANDLE , char \* *buff*, int *length* )

Get Add on PCB article number.

Supported Platform(s): XM, XA, XS

## Parameters

<i>buff</i>	Text output buffer.
<i>length</i>	Maximum length of the output buffer. If the actual length of the data is greater, an error will be returned.

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

## Example Usage:

```
err = About_getAddOnPCBArt (pAbout, buffer, buffer_len);
if (CrossControl::ERR_SUCCESS == err)
    cout << "Add on PCB article number: " << buffer << endl;
```

## 5.1.3.4 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV

CrossControl::About\_getAddOnPCBSerial ( ABOUTHANDLE , char \* *buff*, int *len* )

Get Add on PCB serial number.

Supported Platform(s): XM, XA, XS

## Parameters

<i>buff</i>	Text output buffer.
<i>len</i>	Maximum length of the output buffer. If the actual length of the data is greater, an error will be returned.

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

## Example Usage:

```
err = About_getAddOnPCBSerial (pAbout, buffer, buffer_len);
if (CrossControl::ERR_SUCCESS == err)
    cout << "Add on PCB serial number: " << buffer << endl;
```

**5.1.3.5 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**  
**CrossControl::About\_getDisplayResolution ( ABOUTHANDLE , char \* *buff*, int *len* )**

Get display resolution.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>buff</i>	Text output buffer.
<i>len</i>	Maximum length of the output buffer. If the actual length of the data is greater, an error will be returned. The display resolution will be returned in the format "1024x768"

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```
err = About_getDisplayResolution (pAbout, buffer, buffer_len);
if (CrossControl::ERR_SUCCESS == err)
    cout << "Display resolution: " << buffer << endl;
```

**5.1.3.6 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**  
**CrossControl::About\_getFrontPcbRev ( ABOUTHANDLE , unsigned char \* *major*, unsigned char \* *minor* )**

Get the front hardware pcb revision in the format major.minor (e.g. 1.1).

Supported Platform(s): XA, XS

**Parameters**

<i>major</i>	The major pcb revision.
<i>minor</i>	The minor pcb revision.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.7 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**  
**CrossControl::About\_getIOExpanderValue ( ABOUTHANDLE , unsigned short \* *value* )**

Get Value for IO Expander

Supported Platform(s): XA, XS

**Parameters**

<i>value</i>	IO Expander value.
--------------	--------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.8 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**

**CrossControl::About\_getIsAnybusMounted ( ABOUTHANDLE , bool \* *mounted* )**

Get Anybus mounting status.

Supported Platform(s): XA, XS

**Parameters**

<i>mounted</i>	Is Anybus mounted?
----------------	--------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```
bool isAnybusMounted;
err = CrossControl::About_getIsAnybusMounted(pAbout, &
    isAnybusMounted);
if (CrossControl::ERR_SUCCESS == err)
    cout << "Anybus mounted: " << (isAnybusMounted ? "YES" : "NO") << endl;
```

**5.1.3.9 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**

**CrossControl::About\_getIsBTMounted ( ABOUTHANDLE , bool \* *mounted* )**

Get BlueTooth module mounting status.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>mounted</i>	Is module mounted?
----------------	--------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```
bool isBTMounted;
err = About_getIsBTMounted (pAbout, &isBTMounted);
if (CrossControl::ERR_SUCCESS == err)
    cout << "BT mounted: " << (isBTMounted ? "YES" : "NO") << endl;
```

#### 5.1.3.10 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV

CrossControl::About\_getIsDisplayAvailable ( ABOUTHANDLE , bool \* *available* )

Get Display module status. (Some product variants does not have a display)

Supported Platform(s): XM, XA, XS

##### Parameters

<i>available</i>	Is display available?
------------------	-----------------------

##### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

##### Example Usage:

```
bool displayAvailable;
err = About_getIsDisplayAvailable (pAbout, &displayAvailable);
if (CrossControl::ERR_SUCCESS == err)
    cout << "Display available: " << (displayAvailable ? "YES" : "NO") << endl;
```

#### 5.1.3.11 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV

CrossControl::About\_getIsGPRSMounted ( ABOUTHANDLE , bool \* *mounted* )

Get GPRS module mounting status.

Supported Platform(s): XM, XA, XS

##### Parameters

<i>mounted</i>	Is module mounted?
----------------	--------------------

##### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

##### Example Usage:

```
bool isGPRSMounted;
err = About_getIsGPRSMounted (pAbout, &isGPRSMounted);
if (CrossControl::ERR_SUCCESS == err)
    cout << "GPRS mounted: " << (isGPRSMounted ? "YES" : "NO") << endl;
```



### 5.1.3.12 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::About\_getIsGPSPMounted ( ABOUTHANDLE , bool \* *mounted* )

Get GPS module mounting status.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>mounted</i>	Is module mounted?
----------------	--------------------

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

#### Example Usage:

```
bool isGPSPMounted;
err = About_getIsGPSPMounted (pAbout, &isGPSPMounted);
if (CrossControl::ERR_SUCCESS == err)
    cout << "GPS mounted: " << (isGPSPMounted ? "YES" : "NO") << endl;
```

### 5.1.3.13 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::About\_getIsIOExpanderMounted ( ABOUTHANDLE , bool \* *mounted* )

Get IO Expander mounting status.

Supported Platform(s): XA, XS

#### Parameters

<i>mounted</i>	Is IO Expander mounted?
----------------	-------------------------

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

#### Example Usage:

```
bool isIOExpanderMounted;
err = CrossControl::About_getIsIOExpanderMounted (pAbout, &
    isIOExpanderMounted);
if (CrossControl::ERR_SUCCESS == err)
    cout << "IOExpander mounted: " << (isIOExpanderMounted ? "YES" : "NO") << endl;
```

### 5.1.3.14 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::About\_getIsTouchScreenAvailable ( ABOUTHANDLE , bool \* *available* )

Get Display TouchScreen status.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>available</i>	Is TouchScreen available?
------------------	---------------------------

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

#### Example Usage:

```
bool touchScreenAvailable;
err = About_getIsTouchScreenAvailable (pAbout, &touchScreenAvailable);
if (CrossControl::ERR_SUCCESS == err)
    cout << "TouchScreen available: " << (touchScreenAvailable ? "YES" : "NO") << endl;
```

#### 5.1.3.15 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::About\_getIsWLANMounted ( ABOUTHANDLE , bool \* *mounted* )

Get WLAN module mounting status.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>mounted</i>	Is module mounted?
----------------	--------------------

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

#### Example Usage:

```
bool isWLANMounted;
err = About_getIsWLANMounted (pAbout, &isWLANMounted);
if (CrossControl::ERR_SUCCESS == err)
    cout << "WLAN mounted: " << (isWLANMounted ? "YES" : "NO") << endl;
```

#### 5.1.3.16 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::About\_getMainHWversion ( ABOUTHANDLE , char \* *buff*, int *len* )

Get main hardware version (PCB revision).

Supported Platform(s): XM, XA, XS

## Parameters

<i>buff</i>	Text output buffer.
<i>len</i>	Maximum length of the output buffer. If the actual length of the data is greater, an error will be returned.

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

## Example Usage:

```
err = About_getMainHWversion (pAbout, buffer, buffer_len);
if (CrossControl::ERR_SUCCESS == err)
    cout << "Main hardware version: " << buffer << endl;
```

#### 5.1.3.17 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::About\_getMainManufacturingDate ( ABOUTHANDLE , char \* *buff*, int *len* )

Get main manufacturing date.

Supported Platform(s): XM, XA, XS

## Parameters

<i>buff</i>	Text output buffer.
<i>len</i>	Maximum length of the output buffer. If the actual length of the data is greater, an error will be returned.

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

## Example Usage:

```
err = About_getMainManufacturingDate (pAbout, buffer, buffer_len);
if (CrossControl::ERR_SUCCESS == err)
    cout << "Manufacturing date: " << buffer << endl;
```

#### 5.1.3.18 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::About\_getMainPCBArt ( ABOUTHANDLE , char \* *buff*, int *length* )

Get main PCB article number.

Supported Platform(s): XM, XA, XS

## Parameters

<i>buff</i>	Text output buffer.
<i>length</i>	Maximum length of the output buffer. If the actual length of the data is greater, an error will be returned.

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

## Example Usage:

```
err = About_getMainPCBArt (pAbout, buffer, buffer_len);
if (CrossControl::ERR_SUCCESS == err)
    cout << "Main PCB article number: " << buffer << endl;
```

#### 5.1.3.19 EXTERN.C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::About\_getMainPCBSerial ( ABOUTHANDLE , char \* buff, int len )

Get main PCB serial number.

Supported Platform(s): XM, XA, XS

## Parameters

<i>buff</i>	Text output buffer.
<i>len</i>	Maximum length of the output buffer. If the actual length of the data is greater, an error will be returned.

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

## Example Usage:

```
err = About_getMainPCBSerial (pAbout, buffer, buffer_len);
if (CrossControl::ERR_SUCCESS == err)
    cout << "Main PCB serial: " << buffer << endl;
```

#### 5.1.3.20 EXTERN.C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::About\_getMainProdArtNr ( ABOUTHANDLE , char \* buff, int len )

Get main product article number.

Supported Platform(s): XM, XA, XS

## Parameters

<i>buff</i>	Text output buffer.
<i>len</i>	Maximum length of the output buffer. If the actual length of the data is greater, an error will be returned.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```
err = About_getMainProdArtNr (pAbout, buffer, buffer_len);
if (CrossControl::ERR_SUCCESS == err)
    cout << "Main product article number: " << buffer << endl;
```

**5.1.3.21 EXTERN.C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**

**CrossControl::About\_getMainProdRev ( ABOUTHANDLE , char \* *buff*, int *len* )**

Get main product revision.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>buff</i>	Text output buffer.
<i>len</i>	Maximum length of the output buffer. If the actual length of the data is greater, an error will be returned.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```
err = About_getMainProdRev (pAbout, buffer, buffer_len);
if (CrossControl::ERR_SUCCESS == err)
    cout << "Main product revision: " << buffer << endl;
```

**5.1.3.22 EXTERN.C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**

**CrossControl::About\_getNrOfCANConnections ( ABOUTHANDLE , unsigned char \* *NrOfConnections* )**

Get number of CAN connections present.

Supported Platform(s): XM, XA, XS

## Parameters

<i>NrOfConnections</i>	Returns the number of connections.
------------------------	------------------------------------

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

## Example Usage:

```
unsigned char nrOfCANConnections;
err = About_getNrOfCANConnections (pAbout, &nrOfCANConnections);
if (CrossControl::ERR_SUCCESS == err)
    cout << "Nr of CAN connections: " << (int)nrOfCANConnections << endl;
```

#### 5.1.3.23 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::About\_getNrOfDigIOConnections ( ABOUTHANDLE , unsigned char \* NrOfConnections )

Get number of digital I/O connections present.

Supported Platform(s): XM, XA, XS

## Parameters

<i>NrOfConnections</i>	Returns the number of input or input/output connections.
------------------------	--

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

## Example Usage:

```
unsigned char nrOfDigIOConnections;
err = About_getNrOfDigIOConnections (pAbout, &nrOfDigIOConnections);
if (CrossControl::ERR_SUCCESS == err)
    cout << "Nr of digital I/O connections: " << (int)nrOfDigIOConnections << endl;
```

#### 5.1.3.24 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::About\_getNrOfETHConnections ( ABOUTHANDLE , unsigned char \* NrOfConnections )

Get number of ethernet connections present.

Supported Platform(s): XM, XA, XS

## Parameters

<i>NrOfConnections</i>	Returns the number of connections.
------------------------	------------------------------------

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

## Example Usage:

```
unsigned char nrOfEthConnections;
err = About_getNrOfETHConnections (pAbout, &nrOfEthConnections);
if (CrossControl::ERR_SUCCESS == err)
    cout << "Nr of ethernet connections: " << (int)nrOfEthConnections << endl;
```

## 5.1.3.25 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV

CrossControl::About\_getNrOfSerialConnections ( ABOUTHANDLE , unsigned char \*  
*NrOfConnections* )

Get number of serial port (RS232) connections present.

Supported Platform(s): XM, XA, XS

## Parameters

<i>NrOfConnections</i>	Returns the number of connections.
------------------------	------------------------------------

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

## Example Usage:

```
unsigned char nrOfSerialConnections;
err = About_getNrOfSerialConnections (pAbout, &nrOfSerialConnections);
if (CrossControl::ERR_SUCCESS == err)
    cout << "Nr of serial connections: " << (int)nrOfSerialConnections << endl;
```

## 5.1.3.26 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV

CrossControl::About\_getNrOfUSBConnections ( ABOUTHANDLE , unsigned char \*  
*NrOfConnections* )

Get number of USB connections present.

Supported Platform(s): XM, XA, XS

## Parameters

<i>NrOf-Connections</i>	Returns the number of connections.
-------------------------	------------------------------------

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

## Example Usage:

```
unsigned char nrOfUSBConnections;
err = About_getNrOfUSBConnections (pAbout, &nrOfUSBConnections);
if (CrossControl::ERR_SUCCESS == err)
    cout << "Nr of USB connections: " << (int)nrOfUSBConnections << endl;
```

## 5.1.3.27 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV

CrossControl::About\_getNrOfVideoConnections ( ABOUTHANDLE , unsigned char \*  
*NrOfConnections* )

Get number of Video connections present.

Supported Platform(s): XM, XA, XS

## Parameters

<i>NrOf-Connections</i>	Returns the number of connections.
-------------------------	------------------------------------

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

## Example Usage:

```
unsigned char nrOfVideoConnections;
err = About_getNrOfVideoConnections (pAbout, &nrOfVideoConnections);
if (CrossControl::ERR_SUCCESS == err)
    cout << "Nr of video connections: " << (int)nrOfVideoConnections << endl;
```

## 5.1.3.28 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV

CrossControl::About\_getUnitSerial ( ABOUTHANDLE , char \* *buff*, int *len* )

Get unit serial number.

Supported Platform(s): XM, XA, XS



## Parameters

<i>buff</i>	Text output buffer.
<i>len</i>	Maximum length of the output buffer. If the actual length of the data is greater, an error will be returned.

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

## Example Usage:

```
err = About_getUnitSerial (pAbout, buffer, buffer_len);
if (CrossControl::ERR_SUCCESS == err)
    cout << "Unit serial: " << buffer << endl;
```

### 5.1.3.29 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::About\_hasOsBooted ( ABOUTHANDLE , bool \* *bootComplete* )

Get the status of the OS boot process. In Linux, drivers may be delay-loaded at start-up. If the application is started early in the boot-process, this function can be used to determine when full functionality can be obtained from the API/drivers.

Supported Platform(s): XA, XS

## Parameters

<i>boot-Complete</i>	Is the OS fully booted?
----------------------	-------------------------

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

## Example Usage:

```
bool isBootComplete;
err = CrossControl::About_hasOsBooted(pAbout, &isBootComplete);
if (CrossControl::ERR_SUCCESS == err)
    cout << "System bootup complete: " << (isBootComplete ? "YES" : "NO") << endl;
```

### 5.1.3.30 EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV CrossControl::About\_release ( ABOUTHANDLE )

Delete the About object.

Supported Platform(s): XM, XA, XS

**Returns**

-

**Example Usage:**

```

ABOUTHANDLE pAbout = ::GetAbout();
assert(pAbout);

list_about_information(pAbout);

About_release(pAbout);

```

**5.1.331 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::Adc\_getVoltage ( ADCHANDLE , VoltageEnum *selection*, double \* *value*  
)**

Read measured voltage.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>selection</i>	The type of voltage to get.
<i>value</i>	Voltage value in Volt.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```

err = Adc_getVoltage(pAdc, selection, &voltage);
if (err == CrossControl::ERR_SUCCESS)
{
    cout << left << setw(7) << description << ": " <<
        fixed << setprecision(2) << voltage << "V" << endl;
}
else if (err == CrossControl::ERR_NOT_SUPPORTED)
{
    /* Don't print anything */
}
else
{
    cout << left << setw(7) << description << ": " <<
        fixed << setprecision(2) << CrossControl::GetErrorStringA(err) << endl;
}

```

**5.1.332 EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV  
CrossControl::Adc\_release ( ADCHANDLE )**

Delete the ADC object.

Supported Platform(s): XM, XA, XS

**Returns**

-

**Example Usage:**

```

ADCHANDLE pAdc = ::GetAdc();
assert(pAdc);

output_voltage(pAdc, "24VIN", CrossControl::VOLTAGE_24VIN);
output_voltage(pAdc, "24V", CrossControl::VOLTAGE_24V);
output_voltage(pAdc, "12V", CrossControl::VOLTAGE_12V);
output_voltage(pAdc, "12VID", CrossControl::VOLTAGE_12VID);
output_voltage(pAdc, "5V", CrossControl::VOLTAGE_5V);
output_voltage(pAdc, "3V3", CrossControl::VOLTAGE_3V3);
output_voltage(pAdc, "VTFT", CrossControl::VOLTAGE_VTFT);
output_voltage(pAdc, "5VSTB", CrossControl::VOLTAGE_5VSTB);
output_voltage(pAdc, "1V9", CrossControl::VOLTAGE_1V9);
output_voltage(pAdc, "1V8", CrossControl::VOLTAGE_1V8);
output_voltage(pAdc, "1V5", CrossControl::VOLTAGE_1V5);
output_voltage(pAdc, "1V2", CrossControl::VOLTAGE_1V2);
output_voltage(pAdc, "1V05", CrossControl::VOLTAGE_1V05);
output_voltage(pAdc, "1V0", CrossControl::VOLTAGE_1V0);
output_voltage(pAdc, "0V9", CrossControl::VOLTAGE_0V9);
output_voltage(pAdc, "VREF_INT", CrossControl::VOLTAGE_VREF_INT);
output_voltage(pAdc, "24V_BACKUP", CrossControl::VOLTAGE_24V_BACKUP);
output_voltage(pAdc, "2V5", CrossControl::VOLTAGE_2V5);
output_voltage(pAdc, "1V1", CrossControl::VOLTAGE_1V1);
output_voltage(pAdc, "1V3_PER", CrossControl::VOLTAGE_1V3_PER);
output_voltage(pAdc, "1V3_VDDA", CrossControl::VOLTAGE_1V3_VDDA);

Adc_release(pAdc);

```

**5.1.3.33 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**

**CrossControl::AuxVersion\_getCCAuxDrvVersion ( AUXVERSIONHANDLE , unsigned char \* *major*, unsigned char \* *minor*, unsigned char \* *release*, unsigned char \* *build* )**

Get the **CrossControl** CCAux CCAuxDrv version. Can be used to check that the correct driver is loaded. The version should be the same as that of AuxVersion\_getCCAuxVersion (Win32).

Supported Platform(s): XM

**Parameters**

<i>major</i>	Major version number
<i>minor</i>	Minor version number
<i>release</i>	Release version number
<i>build</i>	Build version number

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```

err = AuxVersion_getCCAuxDrvVersion(
    pAuxVersion,
    &major,
    &minor,
    &release,
    &build);

cout << setw(column_width) << "CCAux Driver Version: ";
if (CrossControl::ERR_SUCCESS == err)
    cout << (int) major << "." <<
        (int) minor << "." <<
        (int) release << "." <<
        (int) build << endl;
else
    cout << "unknown" << endl;

```

#### 5.1.3.34 EXTERN.C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV

CrossControl::AuxVersion\_getCCAuxVersion ( AUXVERSIONHANDLE , unsigned char \* *major*, unsigned char \* *minor*, unsigned char \* *release*, unsigned char \* *build* )

Get the [CrossControl](#) CCAux API version. CCAux includes: CCAuxService/ccauxd - Windows Service/Linux daemon. CCAux2.dll/libccaux2 - The implementation of this API.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>major</i>	Major version number
<i>minor</i>	Minor version number
<i>release</i>	Release version number
<i>build</i>	Build version number

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

#### Example Usage:

```

err = AuxVersion_getCCAuxVersion(
    pAuxVersion,
    &major,
    &minor,
    &release,
    &build);

cout << setw(column_width) << "CC Aux Version: ";
if (CrossControl::ERR_SUCCESS == err)
    cout <<
        (int) major << "." <<
        (int) minor << "." <<
        (int) release << "." <<
        (int) build << endl;
else
    cout << "unknown" << endl;

```

**5.1.3.35 EXTERN.C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**  
**CrossControl::AuxVersion\_getFPGAVersion ( AUXVERSIONHANDLE , unsigned char \*  
*major*, unsigned char \* *minor*, unsigned char \* *release*, unsigned char \* *build* )**

Get the FPGA software version

Supported Platform(s): XM, XA, XS

#### Parameters

<i>major</i>	Major version number
<i>minor</i>	Minor version number
<i>release</i>	Release version number
<i>build</i>	Build version number

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

#### Example Usage:

```
err = AuxVersion_getFPGAVersion(
    pAuxVersion,
    &major,
    &minor,
    &release,
    &build);

cout << setw(column_width) << "FPGA Version: ";
if (CrossControl::ERR_SUCCESS == err)
    cout << (int) major << "." <<
        (int) minor << "." <<
        (int) release << "." <<
        (int) build << endl;
else
    cout << "unknown" << endl;
```

**5.1.3.36 EXTERN.C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**  
**CrossControl::AuxVersion\_getFrontVersion ( AUXVERSIONHANDLE , unsigned char \*  
*major*, unsigned char \* *minor*, unsigned char \* *release*, unsigned char \* *build* )**

Get the front microcontroller software version

Supported Platform(s): XM, XA, XS

#### Parameters

<i>major</i>	Major version number
<i>minor</i>	Minor version number
<i>release</i>	Release version number
<i>build</i>	Build version number

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```
err = AuxVersion_getFrontVersion(
    pAuxVersion,
    &major,
    &minor,
    &release,
    &build);

cout << setw(column_width) << "Front Micro Controller Version: ";
if (CrossControl::ERR_SUCCESS == err)
    cout << (int) major << "." <<
        (int) minor << "." <<
        (int) release << "." <<
        (int) build << endl;
else
    cout << "unknown" << endl;
```

**5.1.3.37 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**

**CrossControl::AuxVersion\_getOSVersion ( AUXVERSIONHANDLE , unsigned char \*  
major, unsigned char \* minor, unsigned char \* release, unsigned char \* build )**

Get the [CrossControl](#) Operating System version.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>major</i>	Major version number
<i>minor</i>	Minor version number
<i>release</i>	Release version number
<i>build</i>	Build version number

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```
err = AuxVersion_getOSVersion(
    pAuxVersion,
    &major,
    &minor,
    &release,
    &build);

cout << setw(column_width) << "Operating System Version: ";
if (CrossControl::ERR_SUCCESS == err)
    cout << (int) major << "." <<
        (int) minor << "." <<
        (int) release << "." <<
        (int) build << endl;
```

```
else
    cout << "unknown" << endl;
```

**5.1.3.38 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::AuxVersion\_getSSVersion ( AUXVERSIONHANDLE , unsigned char \*  
major, unsigned char \* minor, unsigned char \* release, unsigned char \* build )**

Get the System Supervisor software version

Supported Platform(s): XM, XA, XS

**Parameters**

<i>major</i>	Major version number
<i>minor</i>	Minor version number
<i>release</i>	Release version number
<i>build</i>	Build version number

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```
err = AuxVersion_getSSVersion(
    pAuxVersion,
    &major,
    &minor,
    &release,
    &build);

cout << setw(column_width) << "System Supervisor Version: ";
if (CrossControl::ERR_SUCCESS == err)
    cout << (int) major << "." <<
        (int) minor << "." <<
        (int) release << "." <<
        (int) build << endl;
else
    cout << "unknown" << endl;
```

**5.1.3.39 EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV  
CrossControl::AuxVersion\_release ( AUXVERSIONHANDLE )**

Delete the AuxVersion object.

Supported Platform(s): XM, XA, XS

**Returns**

-

**Example Usage:**

```

AUXVERSIONHANDLE pAuxVersion = ::GetAuxVersion();
assert (pAuxVersion);

output_versions (pAuxVersion);

AuxVersion_release (pAuxVersion);

```

#### 5.1.3.40 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV

**CrossControl::Backlight\_getAutomaticBLFilter** ( BACKLIGHTHANDLE , unsigned long \* *averageWndSize*, unsigned long \* *rejectWndSize*, unsigned long \* *rejectDeltaInLux*, LightSensorSamplingMode \* *mode* )

Get light sensor filter parameters for automatic backlight control.

Supported Platform(s): XM, XA, XS

##### Parameters

<i>average-WndSize</i>	The average window size in nr of samples.
<i>rejectWnd-Size</i>	The reject window size in nr of samples.
<i>rejectDelta-InLux</i>	The reject delta in lux.
<i>mode</i>	The configured sampling mode.

##### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

#### 5.1.3.41 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV

**CrossControl::Backlight\_getAutomaticBLParams** ( BACKLIGHTHANDLE , bool \* *bSoftTransitions*, double \* *k* )

Get parameters for automatic backlight control.

Supported Platform(s): XM, XA, XS

##### Parameters

<i>bSoft-Transitions</i>	Soft transitions used?
<i>k</i>	K value.



**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.42 EXTERN.C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**

**CrossControl::Backlight\_getAutomaticBLStatus** ( BACKLIGHTHANDLE , unsigned char \* *status* )

Get status from automatic backlight control.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>status</i>	1=running, 0=stopped.
---------------	-----------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.43 EXTERN.C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**

**CrossControl::Backlight\_getHWStatus** ( BACKLIGHTHANDLE , bool \* *status* )

Get backlight hardware status.

**Parameters**

<i>status</i>	Backlight controller status. true: All backlight drivers works ok, false: one or more backlight drivers are faulty.
---------------	---

Supported Platform(s): XM, XA, XS

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```
bool backlightStatus = false;
err = Backlight_getHWStatus(pBacklight, &backlightStatus);
if(err == ERR_SUCCESS)
{
    if(backlightStatus)
        printf("Backlight hardware status: OK\n");
    else
        printf("Backlight hardware status: not OK, one or more backlight drivers are faulty\n");
}
else if(err == ERR_NOT_SUPPORTED)
{
```

```

    printf("Backlight_getHWStatus: Not supported!\n");
}
else
{
    printf("Error(%d) in function Backlight_getHWStatus: %s\n", err,
        GetErrorStringA(err));
}

```

#### 5.1.3.44 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV

**CrossControl::Backlight\_getIntensity ( BACKLIGHTHANDLE , unsigned char \*  
intensity )**

Get backlight intensity. Note that there might be hardware limitations, limiting the minimum and/or maximum value to other than (1..255).

Supported Platform(s): XM, XA, XS

##### Parameters

<i>intensity</i>	The current backlight intensity (1..255).
------------------	---

##### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

##### Example Usage:

```

err = Backlight_getIntensity(pBacklight, &value);
if(err == ERR_SUCCESS)
{
    printf("Current backlight intensity (0-255): %d\n", value);
}
else
{
    printf("Error(%d) in function Backlight_getIntensity: %s\n", err,
        GetErrorStringA(err));
}

```

#### 5.1.3.45 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV

**CrossControl::Backlight\_getLedDimming ( BACKLIGHTHANDLE , CCStatus \* status )**

Get the current setting for Led dimming. If enabled, the function automatically dims the LED according to the current backlight setting; Low backlight gives less bright LED. This works with manual backlight setting and automatic backlight, but only if the led is set to pure red, green or blue color. If another color is being used, this functionality must be implemented separately.

Supported Platform(s): XM, XA, XS

##### Parameters

<i>status</i>	Enabled/Disabled
---------------	------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.46 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**

**CrossControl::Backlight\_getStatus ( BACKLIGHTHANDLE , unsigned char \* *status* )**

Get backlight controller status. Deprecated, use Backlight\_getHWStatus instead.

Supported Platform(s): XM

**Parameters**

<i>status</i>	Backlight controller status. Bit 0: status controller 1. Bit 1: status controller 2. Bit 2: status controller 3. Bit 3: status controller 4. 1=normal, 0=fault.
---------------	---

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```
err = Backlight_getStatus(pBacklight, &value);
if(err == ERR_SUCCESS)
{
    printf("Backlight status: \nBL1:%s\nBL2:%s\nBL3:%s\nBL4:%s\n",
        (value & 0x01)? "OK" : "NOT OK or missing",
        (value & 0x02)? "OK" : "NOT OK or missing",
        (value & 0x04)? "OK" : "NOT OK or missing",
        (value & 0x08)? "OK" : "NOT OK or missing");
}
else if(err == ERR_NOT_SUPPORTED)
{
    printf("Backlight_getStatus: Not supported!\n");
}
else
{
    printf("Error(%d) in function Backlight_getStatus: %s\n", err,
        GetErrorStringA(err));
}
```

**5.1.3.47 EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV**

**CrossControl::Backlight\_release ( BACKLIGHTHANDLE )**

Delete the backlight object.

Supported Platform(s): XM, XA, XS

**Returns**

-

**Example Usage:**

```

BACKLIGHTHANDLE pBacklight = ::GetBacklight();
assert(pBacklight);

change_backlight(pBacklight);

Backlight_release(pBacklight);

```

**5.1.3.48 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**  
**CrossControl::Backlight\_setAutomaticBLFilter ( BACKLIGHTHANDLE , unsigned long**  
***averageWndSize*, unsigned long *rejectWndSize*, unsigned long *rejectDeltaInLux*,**  
**LightSensorSamplingMode *mode* )**

Set light sensor filter parameters for automatic backlight control.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>average-WndSize</i>	The average window size in nr of samples.
<i>rejectWnd-Size</i>	The reject window size in nr of samples.
<i>rejectDelta-InLux</i>	The reject delta in lux.
<i>mode</i>	The configured sampling mode.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.49 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**  
**CrossControl::Backlight\_setAutomaticBLParams ( BACKLIGHTHANDLE , bool**  
***bSoftTransitions* )**

Set parameters for automatic backlight control.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>bSoft-Transitions</i>	Use soft transitions?
--------------------------	-----------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.50 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**

**CrossControl::Backlight\_setIntensity ( BACKLIGHTHANDLE , unsigned char *intensity* )**

Set backlight intensity. Note that there might be hardware limitations, limiting the minimum and/or maximum value to other than (1..255).

Supported Platform(s): XM, XA, XS

**Parameters**

<i>intensity</i>	The backlight intensity to set (1..255).
------------------	--

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```
err = Backlight_setIntensity(pBacklight, value);

if(err == ERR_SUCCESS)
{
    printf("Setting backlight intensity: %d\n", value);
}
else
{
    printf("Error(%d) in function Backlight_setIntensity: %s\n", err,
        GetErrorStringA(err));
}
```

**5.1.3.51 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**

**CrossControl::Backlight\_setLedDimming ( BACKLIGHTHANDLE , CCStatus *status* )**

Enable/disable Led dimming. If enabled, the function automatically dims the LED according to the current backlight setting; Low backlight gives less bright LED. This works with manual backlight setting and automatic backlight, but only if the led is set to pure red, green or blue color. If another color is being used, this functionality must be implemented separately.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>status</i>	Enabled/Disabled
---------------	------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.52 EXTERN.C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::Backlight\_startAutomaticBL ( BACKLIGHTHANDLE )**

Start automatic backlight control. Note that reading the light sensor at the same time as running the automatic backlight control is not supported.

Supported Platform(s): XM, XA, XS

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.53 EXTERN.C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::Backlight\_stopAutomaticBL ( BACKLIGHTHANDLE )**

Stop automatic backlight control.

Supported Platform(s): XM, XA, XS

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.54 EXTERN.C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::Battery\_getBatteryChargingStatus ( BATTERYHANDLE , ChargingStatus  
\* *status* )**

Get battery charging status.

Supported Platform(s): XM

**Parameters**

<i>status</i>	the current charging mode of the battery.
---------------	---

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

```

ChargingStatus cs;
error = Battery_getBatteryChargingStatus(pBattery, &cs);
if(error == ERR_NOT_SUPPORTED && !bpresent)
{
    cout << "getBatteryChargingStatus: " << GetErrorStringA(error) << " - battery is not
        present!" << std::endl;
}
else if(error != ERR_SUCCESS)
{
    cout << "getBatteryChargingStatus: " << GetErrorStringA(error) << std::endl;
}
else
{
    switch(cs)
    {
    case ChargingStatus_NoCharge:
        cout << "getBatteryChargingStatus: Battery is not being charged" << std::endl;
        break;
    case ChargingStatus_Charging:
        cout << "getBatteryChargingStatus: Battery is being charged" << std::endl;
        break;
    case ChargingStatus_FullyCharged:
        cout << "getBatteryChargingStatus: Battery is fully charged" << std::endl;
        break;
    case ChargingStatus_TempLow:
        cout << "getBatteryChargingStatus: Temperature is too low to charge the battery" << std::endl;
        break;
    case ChargingStatus_TempHigh:
        cout << "getBatteryChargingStatus: Temperature is too high to charge the battery" << std::endl;
        break;
    case ChargingStatus_Unknown:
        cout << "getBatteryChargingStatus: ChargingStatus_Unknown" << std::endl;
        break;
    default:
        cout << "getBatteryChargingStatus: invalid return value" << std::endl;
        break;
    }
}
}

```

### 5.1.3.55 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV

CrossControl::Battery\_getBatteryHWversion ( BATTERYHANDLE , char \* buff, int len )

Get battery hardware version (PCB revision).

Supported Platform(s): XM

#### Parameters

<i>buff</i>	Text output buffer.
<i>len</i>	Maximum length of the output buffer. If the actual length of the data is greater, an error will be returned.

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

#### Example Usage:

```

char buf[255];
error = Battery_getBatteryHWversion(pBattery, buf, sizeof(buf));
if(error == ERR_NOT_SUPPORTED && !bpresent)

```

```

{
    cout << "getBatteryHWversion: " << GetErrorStringA(error) << " - battery is not present!"
         << std::endl;
}
else if(error != ERR_SUCCESS)
{
    cout << "getBatteryHWversion: " << GetErrorStringA(error) << std::endl;
}
else
{
    cout << "getBatteryHWversion: " << buf << std::endl;
}
}

```

### 5.1.3.56 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV

**CrossControl::Battery\_getBatterySerial ( BATTERYHANDLE , char \* buff, int len )**

Get battery serial number.

Supported Platform(s): XM

#### Parameters

<i>buff</i>	Text output buffer.
<i>len</i>	Maximum length of the output buffer. If the actual length of the data is greater, an error will be returned. The serial number is 10 characters plus terminating zero, in total 11 bytes in size.

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

#### Example Usage:

```

error = Battery_getBatterySerial(pBattery,buf, sizeof(buf));
if(error == ERR_NOT_SUPPORTED && !bpresent)
{
    cout << "getBatterySerial: " << GetErrorStringA(error) << " - battery is not present!" <
    < std::endl;
}
else if(error != ERR_SUCCESS)
{
    cout << "getBatterySerial: " << GetErrorStringA(error) << std::endl;
}
else
{
    cout << "getBatterySerial: " << buf << std::endl;
}
}

```

### 5.1.3.57 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV

**CrossControl::Battery\_getBatterySwVersion ( BATTERYHANDLE , unsigned short \* major, unsigned short \* minor, unsigned short \* release, unsigned short \* build )**

Get the battery software version

Supported Platform(s): XM



**Parameters**

<i>major</i>	Major version number
<i>minor</i>	Minor version number
<i>release</i>	Release version number
<i>build</i>	Build version number

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```

unsigned short major;
unsigned short minor;
unsigned short release;
unsigned short build;
error = Battery_getBatterySwVersion(pBattery, &major, &minor, &release, &build
);
if(error == ERR_NOT_SUPPORTED && !bpresent)
{
    cout << "getBatterySwVersion: " << GetErrorStringA(error) << " - battery is not present!
        " << std::endl;
}
else if(error != ERR_SUCCESS)
{
    cout << "getBatterySwVersion: " << GetErrorStringA(error) << std::endl;
}
else
{
    cout << "getBatterySwVersion: v" << major << "." << minor << "." << release << "." << build <<
        std::endl;
}

```

**5.1.3.58 EXTERN.C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**

**CrossControl::Battery\_getBatteryTemp ( BATTERYHANDLE , signed short \*  
temperature )**

Get battery temperature.

Supported Platform(s): XM

**Parameters**

<i>temperature</i>	PCB Temperature in degrees Celsius.
--------------------	-------------------------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```

short temp;
error = Battery_getBatteryTemp(pBattery, &temp);

```

```

if(error == ERR_NOT_SUPPORTED && !bpresent)
{
    cout << "getBatteryTemp: " << GetErrorStringA(error) << " - battery is not present!" <<
        std::endl;
}
else if(error != ERR_SUCCESS)
{
    cout << "getBatteryTemp: " << GetErrorStringA(error) << std::endl;
}
else
{
    cout << "getBatteryTemp: " << temp << " deg C" << std::endl;
}
}

```

#### 5.1.3.59 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV

CrossControl::Battery\_getBatteryVoltageStatus ( BATTERYHANDLE , unsigned char \*  
batteryVoltagePercent )

Get battery voltage status.

Supported Platform(s): XM

##### Parameters

<i>battery- Voltage- Percent</i>	the current voltage level of the battery, in percent [0..100].
--	--

##### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

##### Example Usage:

```

unsigned char s;
error = Battery_getBatteryVoltageStatus(pBattery, &s);
if(error == ERR_NOT_SUPPORTED && !bpresent)
{
    cout << "getBatteryVoltageStatus: " << GetErrorStringA(error) << " - battery is not
        present!" << std::endl;
}
else if(error != ERR_SUCCESS)
{
    cout << "getBatteryVoltageStatus: " << GetErrorStringA(error) << std::endl;
}
else
{
    cout << "getBatteryVoltageStatus: " << (int)s << " %" << std::endl;
}
}

```

#### 5.1.3.60 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV

CrossControl::Battery\_getHwErrorStatus ( BATTERYHANDLE , ErrorStatus \*  
errorCode )

Get hardware error code. If hardware errors are found or other problems are discovered by the battery pack, they are reported here.

Supported Platform(s): XM

#### Parameters

<i>errorCode</i>	Error code. Zero means no error.
------------------	----------------------------------

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

#### Example Usage:

```

ErrorStatus es;
error = Battery_getHwErrorStatus(pBattery, &es);

if(error == ERR_NOT_SUPPORTED && !bpresent)
{
    cout << "getHwErrorStatus: " << GetErrorStringA(error) << " - battery is not present!" <<
        < std::endl;
}
else if(error != ERR_SUCCESS)
{
    cout << "getHwErrorStatus: " << GetErrorStringA(error) << std::endl;
}
else
{
    switch(es)
    {
        case ErrorStatus_NoError:
            cout << "getHwErrorStatus: " << "Battery reports no HW errors" << std::endl;
            break;
        case ErrorStatus_ThermistorTempSensor:
            cout << "getHwErrorStatus: " << "Battery error! The thermistor temp sensor is not working" <<
                std::endl;
            break;
        case ErrorStatus_SecondaryTempSensor:
            cout << "getHwErrorStatus: " << "Battery error! The secondary temp sensor is not working" <<
                std::endl;
            break;
        case ErrorStatus_ChargeFail:
            cout << "getHwErrorStatus: " << "Battery error! Charging failed" << std::endl;
            break;
        case ErrorStatus_Overcurrent:
            cout << "getHwErrorStatus: " << "Battery error! Overcurrent detected" << std::endl;
            break;
        case ErrorStatus_Init:
            cout << "getHwErrorStatus: " << "Battery error! Battery not initiated" << std::endl;
            break;
        default:
            cout << "getHwErrorStatus: " << "invalid return value" << std::endl;
            break;
    }
}

```

**5.1.3.61 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Battery\_getMinMaxTemp ( BATTERYHANDLE , signed short \* *minTemp*,  
 signed short \* *maxTemp* )**

Get temperature interval of the battery.

Supported Platform(s): XM

**Parameters**

<i>minTemp</i>	Minimum measured temperature.
<i>maxTemp</i>	Maximum measured temperature.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```
short max;
error = Battery_getMinMaxTemp(pBattery, &temp, &max);
if(error == ERR_NOT_SUPPORTED && !bpresent)
{
    cout << "getMinMaxTemp: " << GetErrorStringA(error) << " - battery is not present!" <<
        std::endl;
}
else if(error != ERR_SUCCESS)
{
    cout << "getMinMaxTemp: " << GetErrorStringA(error) << std::endl;
}
else
{
    cout << "getMinMaxTemp: MinTemp:" << temp << ", MaxTemp: " << max << std::endl;
}
}
```

**5.1.3.62 EXTERN.C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**

**CrossControl::Battery\_getPowerSource ( BATTERYHANDLE , PowerSource \* status )**

Get the currently used power source.

Supported Platform(s): XM

**Parameters**

<i>status</i>	the current power source, external power or battery.
---------------	--

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```
PowerSource ps;
error = Battery_getPowerSource(pBattery, &ps);
if(error == ERR_NOT_SUPPORTED && !bpresent)
{
    cout << "getPowerSource: " << GetErrorStringA(error) << " - battery is not present!" <<
        std::endl;
}
else if(error != ERR_SUCCESS)
{
    cout << "getPowerSource: " << GetErrorStringA(error) << std::endl;
}
}
```

```

else
{
    if (ps == PowerSource_Battery)
        cout << "getPowerSource: Power source: Battery" << std::endl;
    else
        cout << "getPowerSource: Power source: External Power" << std::endl;
}

```

### 5.1.3.63 EXTERN.C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV

**CrossControl::Battery\_getTimer ( BATTERYHANDLE , BatteryTimerType \* times )**

Get battery diagnostic timer.

Supported Platform(s): XM

#### Parameters

<i>times</i>	Get a struct with the current diagnostic times.
--------------	---

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

```

BatteryTimerType times;
memset(&times, 0, sizeof(times));
error = Battery_getTimer(pBattery, &times);
if(error == ERR_NOT_SUPPORTED && !bpresent)
{
    cout << "getTimer: " << GetErrorStringA(error) << " - battery is not present!" <<
        std::endl;
}
else if(error != ERR_SUCCESS)
{
    cout << "getTimer: " << GetErrorStringA(error) << std::endl;
}
else
{
    cout << "getTimer: " << std::endl;
    cout << "Total run time on main power=" << times.TotRunTimeMain*60 << " min(s)" << std::endl
        << "Total run time on battery power=" << times.TotRunTimeBattery*60 << " min(s)" << std::endl
        << "Total run time below -20C=" << times.RunTime_m20 << " min(s)" << std::endl
        << "Total run time -20-0C=" << times.RunTime_m20_0 << " min(s)" << std::endl
        << "Total run time 0-40C=" << times.RunTime_0_40 << " min(s)" << std::endl
        << "Total run time 40-60C=" << times.RunTime_40_60 << " min(s)" << std::endl
        << "Total run time 60-70C=" << times.RunTime_60_70 << " min(s)" << std::endl
        << "Total run time 70-80C=" << times.RunTime_70_80 << " min(s)" << std::endl
        << "Total run time above 80C=" << times.RunTime_Above80 << " min(s)" << std::endl;
}

```

### 5.1.3.64 EXTERN.C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV

**CrossControl::Battery\_isBatteryPresent ( BATTERYHANDLE , bool \* batteryIsPresent )**

Is an external battery connected?

Supported Platform(s): XM

## Parameters

<i>batteryIsPresent</i>	true if a battery is connected, otherwise false.
-------------------------	--

## Returns

-

## Example Usage:

```

error = Battery_isBatteryPresent(pBattery, &bpresent);
if(error != ERR_SUCCESS)
{
    cout << "isBatteryPresent: " << GetErrorStringA(error) << std::endl;
}
else
{
    if(bpresent)
    {
        cout << "Battery is present. Testing functionality... " << std::endl;
    }
    else
    {
        cout << "Battery is NOT present." << std::endl;
    }
}

```

### 5.1.3.65 EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV CrossControl::Battery\_release ( BATTERYHANDLE )

Delete the Battery object

Supported Platform(s): XM.

## Returns

-

## Example Usage:

```

BATTERYHANDLE pBattery = ::GetBattery();
assert(pBattery);

readBatteryInfo(pBattery);

Battery_release(pBattery);

```

### 5.1.3.66 EXTERN\_C CCAUXDLL\_API eErrr CCAUXDLL\_CALLING\_CONV CrossControl::Buzzer\_buaze ( BUZZERHANDLE , int time, bool blocking )

Buzzes for a specified time.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>time</i>	Time (ms) to buzz.
<i>blocking</i>	Blocking or non-blocking function.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```
err = Buzzer_setFrequency(pBuzzer, freq);
if(err != ERR_SUCCESS)
{
    cout << "Error(" << err << ") in function setFrequency: " << GetErrorStringA(err) << endl;
}
else
{
    err = Buzzer_buzze(pBuzzer, duration, true);
    if(err != ERR_SUCCESS)
    {
        cout << "Error(" << err << ") in function buzze: " << GetErrorStringA(err) << endl;
    }
}
```

**5.1.3.67 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**

**CrossControl::Buzzer\_getFrequency ( BUZZERHANDLE , unsigned short \* *frequency* )**

Get buzzer frequency.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>frequency</i>	Current frequency (700-10000 Hz).
------------------	-----------------------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.68 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**

**CrossControl::Buzzer\_getTrigger ( BUZZERHANDLE , bool \* *trigger* )**

Get buzzer trigger. The Buzzer is enabled when the trigger is enabled.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>trigger</i>	Current trigger status.
----------------	-------------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.69 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**

**CrossControl::Buzzer\_getVolume ( BUZZERHANDLE , unsigned short \* volume )**

Get buzzer volume.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>volume</i>	Current volume (0-51).
---------------	------------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```
err = Buzzer_getVolume( pBuzzer, &vol);
if(err == ERR_SUCCESS)
{
    cout << "Buzzer volume was: " << vol << endl;
}
else
{
    cout << "Error(" << err << ") in function getVolume: " << GetErrorStringA(err) << endl;
    vol = 40;
}
```

**5.1.3.70 EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV**

**CrossControl::Buzzer\_release ( BUZZERHANDLE )**

Delete the Buzzer object.

Supported Platform(s): XM, XA, XS

**Returns**

-

**Example Usage:**

```
BUZZERHANDLE pBuzzer = ::GetBuzzer();
assert(pBuzzer);

play_beeps(pBuzzer);

Buzzer_release(pBuzzer);
```



### 5.1.3.71 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::Buzzer\_setFrequency ( BUZZERHANDLE , unsigned short *frequency* )

Set buzzer frequency.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>frequency</i>	Frequency to set (700-10000 Hz).
------------------	----------------------------------

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

#### Example Usage:

```
err = Buzzer_setFrequency(pBuzzer, freq);
if(err != ERR_SUCCESS)
{
    cout << "Error(" << err << ") in function setFrequency: " << GetErrorStringA(err) <<
        endl;
}
else
{
    err = Buzzer_buzze(pBuzzer, duration, true);
    if(err != ERR_SUCCESS)
    {
        cout << "Error(" << err << ") in function buzze: " << GetErrorStringA(err) << endl;
    }
}
```

### 5.1.3.72 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::Buzzer\_setTrigger ( BUZZERHANDLE , bool *trigger* )

Set buzzer trigger. The Buzzer is enabled when the trigger is enabled.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>trigger</i>	Status to set.
----------------	----------------

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

### 5.1.3.73 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::Buzzer\_setVolume ( BUZZERHANDLE , unsigned short *volume* )

Set buzzer volume.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>volume</i>	Volume to set (0-51).
---------------	-----------------------

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

#### Example Usage:

```
err = Buzzer_setVolume( pBuzzer, 20);
if(err == ERR_SUCCESS)
{
    cout << "Buzzer volume set to 20" << endl;
}
else
{
    cout << "Error(" << err << ") in function setVolume: " << GetErrorStringA(err) << endl;
}
```

#### 5.1.3.74 EXTERN.C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV

CrossControl::CanSetting\_getBaudrate ( CANSETTINGHANDLE , unsigned char *net*, unsigned short \* *baudrate* )

Get Baud rate

Supported Platform(s): XM, XA, XS

#### Parameters

<i>net</i>	CAN net (1-4) to get settings for.
<i>baudrate</i>	CAN baud rate (kbit/s).

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

#### Example Usage:

```
err = CanSetting_getBaudrate(pCanSetting, net, &baudrates[net-1]);
if(err != ERR_SUCCESS)
{
    cout << "Error(" << err << ") in function getBaudrate: " <<
    GetErrorStringA(err) << endl;
    break;
}
```

**5.1.3.75** EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::CanSetting\_getFrameType ( CANSETTINGHANDLE , unsigned char *net*,  
 CanFrameType \* *frameType* )

Get frame type

Supported Platform(s): XM, XA, XS

#### Parameters

<i>net</i>	CAN net (1-4) to get settings for.
<i>frameType</i>	CAN frame type

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

#### Example Usage:

```
err = CanSetting_getFrameType(pCanSetting, net, &frametypes[net-1]);
if(err != ERR_SUCCESS)
{
    cout << "Error(" << err << ") in function getFrameType: " <<
    GetErrorStringA(err) << endl;
    break;
}
```

**5.1.3.76** EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV  
 CrossControl::CanSetting\_release ( CANSETTINGHANDLE )

Delete the CanSetting object.

Supported Platform(s): XM, XA, XS

#### Returns

-

#### Example Usage:

```
CANSETTINGHANDLE pCanSetting = ::GetCanSetting();
assert(pCanSetting);

read_cansettings(pCanSetting);

CanSetting_release(pCanSetting);
```

**5.1.3.77** EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::CanSetting\_setBaudrate ( CANSETTINGHANDLE , unsigned char *net*,  
 unsigned short *baudrate* )

Set Baud rate. The changes will take effect after a restart.

Supported Platform(s): XM, XA, XS

## Parameters

<i>net</i>	CAN net (1-4).
<i>baudrate</i>	CAN baud rate (kbit/s). The driver will calculate the best supported baud rate if it does not support the given baud rate. The maximum baud rate is 1000 kbit/s.

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

## 5.1.3.78 EXTERN.C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV

CrossControl::CanSetting\_setFrameType ( CANSETTINGHANDLE , unsigned char *net*, CanFrameType *frameType* )

Set frame type. The changes will take effect after a restart.

Supported Platform(s): XM, XA, XS

## Parameters

<i>net</i>	CAN net (1-4).
<i>frameType</i>	CAN frameType

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

## 5.1.3.79 EXTERN.C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV

CrossControl::Config\_getCanStartupPowerConfig ( CONFIGHANDLE , CCStatus \* *status* )

Get Can power at startup configuration. The status of Can power at startup and at resume from suspended mode. At resume from suspend, this setting overrides the setting of the setCanPowerStatus function.

Supported Platform(s): XM, XA, XS

## Parameters

<i>status</i>	Enabled/Disabled
---------------	------------------

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.80 EXTERN.C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Config\_getExtFanStartupPowerConfig ( CONFIGHANDLE , CCStatus \*  
*status* )

Get External fan power at startup configuration. The status at startup and at resume from suspended mode. At resume from suspend, this setting overrides the setting of the setExtFanPowerStatus function.

Supported Platform(s): XM

#### Parameters

<i>status</i>	Enabled/Disabled
---------------	------------------

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.81 EXTERN.C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Config\_getExtOnOffSigTrigTime ( CONFIGHANDLE , unsigned long \*  
*triggertime* )

Get external on/off signal trigger time.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>triggertime</i>	Time in seconds that the external signal has to be low for the unit to enter suspend mode or shut down (trigger an action). This time can be set from one second up to several years, if needed.
--------------------	--

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.82 EXTERN.C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Config\_getFrontBtnTrigTime ( CONFIGHANDLE , unsigned short \*  
*triggertime* )

Get front button trigger time for long press.

Supported Platform(s): XM, XA, XS

## Parameters

<i>triggertime</i>	Time in milliseconds that the button has to be pressed for the press to count as a long button press. A button press twice this time will generate a hard shut down. If this time is set under 4000ms, the hard shut down minimum time of 8s is used instead.
--------------------	---

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

## 5.1.3.83 EXTERN.C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV

CrossControl::Config\_getHeatingTempLimit ( CONFIGHANDLE , signed short \*  
*temperature* )

Get the current limit for heating. When temperature is below this limit, the system is internally heated until the temperature rises above the limit. The default and minimum value is -25 degrees Celsius. The maximum value is +5 degrees Celsius.

Supported Platform(s): XM, XA, XS

## Parameters

<i>temperature</i>	The current heating limit, in degrees Celsius (-25 to +5)
--------------------	---

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

## 5.1.3.84 EXTERN.C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV

CrossControl::Config\_getLongButtonPressAction ( CONFIGHANDLE , PowerAction \*  
*action* )

Get long button press action. Gets the configured action for a long button press: No-Action, ActionSuspend or ActionShutDown. A long button press is determined by the FrontBtnTrigTime.

Supported Platform(s): XM, XA, XS

## Parameters

<i>action</i>	The configured action.
---------------	------------------------

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.85 EXTERN.C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Config\_getOnOffSigAction ( CONFIGHANDLE , PowerAction \* *action* )

Get On/Off signal action. Gets the configured action for an On/Off signal event: No-Action, ActionSuspend or ActionShutDown. An On/Off signal event is determined by the ExtOnOffSigTrigTime.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>action</i>	The configured action.
---------------	------------------------

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.86 EXTERN.C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Config\_getPowerOnStartup ( CONFIGHANDLE , CCStatus \* *status* )

Get power on start-up behavior. If enabled, the unit always starts when power is turned on, disregarding the setting for StartupTriggerConfig at that time. The StartupTriggerConfig still applies if the unit is shut down or suspended, without removing the power supply.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>status</i>	Enabled/Disabled
---------------	------------------

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.87 EXTERN.C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Config\_getRS485Enabled ( CONFIGHANDLE , RS4XXPort *port*, bool \* *enabled* )

Get RS485 mode configuration for RS4XX port.

Supported Platform(s): XA, XS

#### Parameters

<i>port</i>	RS4XX port (RS4XXPort1-4)
<i>enabled</i>	Is the RS485 port enabled (true/false)

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.88 EXTERN.C CCAUXDLL.API eErr CCAUXDLL\_CALLING\_CONV**

**CrossControl::Config\_getShortButtonPressAction** ( CONFIGHANDLE , PowerAction \* *action* )

Get short button press action. Gets the configured action for a short button press: No-Action, ActionSuspend or ActionShutDown.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>action</i>	The configured action.
---------------	------------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.89 EXTERN.C CCAUXDLL.API eErr CCAUXDLL\_CALLING\_CONV**

**CrossControl::Config\_getStartupTriggerConfig** ( CONFIGHANDLE , TriggerConf \* *config* )

Get Start-up trigger configuration. Is the front button and/or the external on/off signal enabled as triggers for startup and wake up from suspended mode?

Supported Platform(s): XM, XA, XS

**Parameters**

<i>config</i>	One of: Front_Button_Enabled, OnOff_Signal_Enabled or Both_Button_And_Signal_Enabled.
---------------	---

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```
err = Config_getStartupTriggerConfig(pConfig, &trig);
if(err == ERR_SUCCESS)
{
    cout << "Start-up trigger is set to: ";
    switch(trig)
    {
        case Front_Button_Enabled: cout << "Front button only" << endl; break;
    }
}
```



```

    case OnOff_Signal_Enabled: cout << "On/Off signal only" << endl; break;
    case Both_Button_And_Signal_Enabled: cout << "Front button or On/off
    signal" << endl; break;
    default: cout << "Error - Undefined StartupTrigger" << endl; break;
  }
}
else
{
  cout << "Error(" << err << ") in function getStartupTriggerConfig: " <<
  GetErrorStringA(err) << endl;
}

```

#### 5.1.3.90 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV

CrossControl::Config\_getStartupVoltageConfig ( CONFIGHANDLE , double \* *voltage* )

Get the voltage threshold required for startup. The external voltage must be stable above this value for the unit to start up. The default and minimum value is 9V. It could be set to a higher value for a 24V system.

Supported Platform(s): XM

##### Parameters

<i>voltage</i>	The current voltage setting. (9V .. 28V)
----------------	--

##### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

#### 5.1.3.91 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV

CrossControl::Config\_getSuspendMaxTime ( CONFIGHANDLE , unsigned short \* *maxTime* )

Get suspend mode maximum time.

Supported Platform(s): XM

##### Parameters

<i>maxTime</i>	Maximum suspend time in minutes. After this time in suspended mode, the unit will shut down to save power. A value of 0 means that the automatic shut down function is not used.
----------------	--

##### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.92** EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Config\_getVideoStartupPowerConfig ( CONFIGHANDLE , unsigned char  
 \* *config* )

Get Video power at startup configuration. The status of Video power at startup and at resume from suspended mode. At resume from suspend, this setting overrides the setting of the setVideoPowerStatus function.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>config</i>	Bitwise representation of the four video channels. See the VideoXConf defines. if the bit is 1, the power is enabled, else disabled.
---------------	--

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.93** EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV  
 CrossControl::Config\_release ( CONFIGHANDLE )

Delete the Config object.

Supported Platform(s): XM, XA, XS

#### Returns

-

#### Example Usage:

```
CONFIGHANDLE pConfig = ::GetConfig();
assert (pConfig);

conf_example (pConfig);

Config_release (pConfig);
```

**5.1.3.94** EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Config\_setCanStartupPowerConfig ( CONFIGHANDLE , CCStatus *status*  
 )

Set Can power at startup configuration. The status of Can power at startup and at resume from suspended mode. At resume from suspend, this setting overrides the setting of the setCanPowerStatus function.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>status</i>	Enabled/Disabled
---------------	------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.95 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**  
**CrossControl::Config\_setExtFanStartupPowerConfig ( CONFIGHANDLE , CCStatus**  
***status* )**

Set External fan power at startup configuration. The status at startup and at resume from suspended mode. At resume from suspend, this setting overrides the setting of the setExtFanPowerStatus function.

Supported Platform(s): XM

**Parameters**

<i>status</i>	Enabled/Disabled
---------------	------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.96 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**  
**CrossControl::Config\_setExtOnOffSigTrigTime ( CONFIGHANDLE , unsigned long**  
***triggertime* )**

Set external on/off signal trigger time.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>triggertime</i>	Time in seconds that the external signal has to be low for the unit to enter suspend mode or shut down (trigger an action). This time can be set from one second up to several years, if needed.
--------------------	--

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.97 **EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**  
**CrossControl::Config\_setFrontBtnTrigTime ( CONFIGHANDLE , unsigned short**  
*triggertime )*

Set front button trigger time for long press.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>triggertime</i>	Time in milliseconds that the button has to be pressed for the press to count as a long button press. A button press twice this time will generate a hard shut down. If this time is set under 4000ms, the hard shut down minimum time of 8s is used instead.
--------------------	---

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.98 **EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**  
**CrossControl::Config\_setHeatingTempLimit ( CONFIGHANDLE , signed short**  
*temperature )*

Set the current limit for heating. When temperature is below this limit, the system is internally heated until the temperature rises above the limit. The default and minimum value is -25 degrees Celsius. The maximum value is +5 degrees Celsius.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>temperature</i>	The heating limit, in degrees Celsius (-25 to +5)
--------------------	---

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.99 **EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**  
**CrossControl::Config\_setLongButtonPressAction ( CONFIGHANDLE , PowerAction**  
*action )*

Set long button press action. Sets the configured action for a long button press: No-Action, ActionSuspend or ActionShutDown. A long button press is determined by the FrontBtnTrigTime.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>action</i>	The action to set.
---------------	--------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.100 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**

**CrossControl::Config\_setOnOffSigAction ( CONFIGHANDLE , PowerAction *action* )**

Set On/Off signal action. Sets the configured action for an On/Off signal event: No-Action, ActionSuspend or ActionShutDown. An On/Off signal event is determined by the ExtOnOffSigTrigTime.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>action</i>	The action to set.
---------------	--------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.101 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**

**CrossControl::Config\_setPowerOnStartup ( CONFIGHANDLE , CCStatus *status* )**

Set power on start-up behavior. If enabled, the unit always starts when power is turned on, disregarding the setting for StartupTriggerConfig at that time. The StartupTriggerConfig still applies if the unit is shut down or suspended, without removing the power supply.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>status</i>	Enabled/Disabled
---------------	------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.102 `EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV  
CrossControl::Config_setRS485Enabled ( CONFIGHANDLE , RS4XXPort port, bool  
enabled )`

Set RS485 mode enabled or disabled for RS4XX port.

Supported Platform(s): XA, XS

#### Parameters

<i>port</i>	RS4XX port (RS4XXPort1-4)
<i>enabled</i>	RS485 enabled (true/false)

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.103 `EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV  
CrossControl::Config_setShortButtonPressAction ( CONFIGHANDLE , PowerAction  
action )`

Set short button press action. Sets the configured action for a short button press: No-Action, ActionSuspend or ActionShutDown.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>action</i>	The action to set.
---------------	--------------------

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

#### Example Usage:

```
err = Config_setShortButtonPressAction(pConfig,
    ActionSuspend);
if(err == ERR_SUCCESS)
{
    cout << "ShortButtonPressAction set to Suspend!" << endl;
}
else
{
    cout << "Error(" << err << ") in function setShortButtonPressAction: " <<
        GetErrorStringA(err) << endl;
}
```

5.1.3.104 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Config\_setStartupTriggerConfig ( CONFIGHANDLE , TriggerConf *conf* )

Set Start-up trigger configuration. Should the front button and/or the external on/off signal be enabled as triggers for startup and wake up from suspended mode?

Supported Platform(s): XM, XA, XS

#### Parameters

<i>conf</i>	Must be one of: Front_Button_Enabled, OnOff_Signal_Enabled or Both_Button_And_Signal_Enabled.
-------------	---

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.105 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Config\_setStartupVoltageConfig ( CONFIGHANDLE , double *voltage* )

Set the voltage threshold required for startup. The external voltage must be stable above this value for the unit to start up. The default and minimum value is 9V. It could be set to a higher value for a 24V system.

Supported Platform(s): XM

#### Parameters

<i>voltage</i>	The voltage to set (9V .. 28V).
----------------	---------------------------------

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.106 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Config\_setSuspendMaxTime ( CONFIGHANDLE , unsigned short *maxTime* )

Set suspend mode maximum time.

Supported Platform(s): XM

#### Parameters

<i>maxTime</i>	Maximum suspend time in minutes. After this time in suspended mode, the unit will shut down to save power. A value of 0 means that this function is not used.
----------------	---

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.107** EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Config\_setVideoStartupPowerConfig ( CONFIGHANDLE , unsigned char *config* )

Set Video power at startup configuration. The status of Video power at startup and at resume from suspended mode. At resume from suspend, this setting overrides the setting of the setVideoPowerStatus function.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>config</i>	Bitwise representation of the four video channels. See the VideoXConf defines. if the bit is 1, the power is enabled, else disabled.
---------------	--

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.108** EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Diagnostic\_clearHwErrorStatus ( DIAGNOSTICHANDLE )

Clear the HW error status (this function is used by the [CrossControl](#) service/daemon to log any hardware errors)

Supported Platform(s): XM, XA, XS

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.109** EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Diagnostic\_getHwErrorStatus ( DIAGNOSTICHANDLE , unsigned short \* *errorCode* )

Get hardware error code. If hardware errors are found or other problems are discovered by the SS, they are reported here. See [DiagnosticCodes.h](#) for error codes.

Supported Platform(s): XM, XA, XS



## Parameters

<i>errorCode</i>	Error code. Zero means no error.
------------------	----------------------------------

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.110 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**  
**CrossControl::Diagnostic\_getMinMaxTemp ( DIAGNOSTICHANDLE , signed short \*  
*minTemp*, signed short \* *maxTemp* )**

Get diagnostic temperature interval of the unit.

Supported Platform(s): XM, XA, XS

## Parameters

<i>minTemp</i>	Minimum measured PCB temperature.
<i>maxTemp</i>	Maximum measured PCB temperature.

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

## Example Usage:

```
err = Diagnostic_getMinMaxTemp(pDiagnostic, &sValue, &sValue2);
printString(err, "Minimum temp", sValue, "deg C");
printString(err, "Maximum temp", sValue2, "deg C");
```

**5.1.3.111 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**  
**CrossControl::Diagnostic\_getPCBTemp ( DIAGNOSTICHANDLE , signed short \*  
*temperature* )**

Get PCB temperature.

Supported Platform(s): XM, XA, XS

## Parameters

<i>temperature</i>	PCB Temperature in degrees Celsius.
--------------------	-------------------------------------

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.112 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Diagnostic\_getPMTemp ( DIAGNOSTICHANDLE , unsigned char *index*,  
 signed short \* *temperature*, JidaSensorType \* *jst* )

Get Processor Module temperature. This temperature is read from the Kontron JIDA API. This API also has a number of other functions, please see the JIDA documentation for how to use them separately.

#### Parameters

<i>index</i>	Zero-based index of the temperature sensor. Different boards may have different number of sensors. The CCPilot XM currently has 2 sensors, board and cpu. An error is returned if the index is not supported.
--------------	---

Supported Platform(s): XM

#### Parameters

<i>temperature</i>	Temperature in degrees Celsius.
<i>jst</i>	The type of sensor that is being read.

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.113 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Diagnostic\_getPowerCycles ( DIAGNOSTICHANDLE , unsigned short \*  
*powerCycles* )

Get number of power cycles.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>powerCycles</i>	Total number of power cycles.
--------------------	-------------------------------

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.114 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Diagnostic\_getShutDownReason ( DIAGNOSTICHANDLE , unsigned  
 short \* *reason* )

Get shutdown reason.

Supported Platform(s): XM

#### Parameters

<i>reason</i>	See <a href="#">DiagnosticCodes.h</a> for shutdown codes.
---------------	---

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.115** EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Diagnostic\_getSSTemp ( DIAGNOSTICHANDLE , signed short \*  
*temperature* )

Get System Supervisor temperature.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>temperature</i>	System Supervisor temperature in degrees Celsius.
--------------------	---

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

```
err = Diagnostic_getSSTemp(pDiagnostic, &sValue);
printString(err, "Main board (SS) temp", sValue, "deg C");
```

**5.1.3.116** EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Diagnostic\_getStartupReason ( DIAGNOSTICHANDLE , unsigned short  
 \* *reason* )

Get startup reason.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>reason</i>	See <a href="#">DiagnosticCodes.h</a> for startup codes.
---------------	--

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.117 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**

**CrossControl::Diagnostic\_getTimer ( DIAGNOSTICHANDLE , TimerType \* times )**

Get diagnostic timer.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>times</i>	Get a struct with the current diagnostic times.
--------------	---

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```
err = Diagnostic_getTimer(pDiagnostic, &tt);
printStringTime(err, "Total run time", tt.TotRunTime);
printStringTime(err, "Total suspend time", tt.TotSuspTime);
printStringTime(err, "Total heat time", tt.TotHeatTime);
printStringTime(err, "Total run time 40-60 deg C", tt.RunTime40_60);
printStringTime(err, "Total run time 60-70 deg C", tt.RunTime60_70);
printStringTime(err, "Total run time 70-80 deg C", tt.RunTime70_80);
printStringTime(err, "Total run time above 80 deg C", tt.Above80RunTime);
```

**5.1.3.118 EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV**

**CrossControl::Diagnostic\_release ( DIAGNOSTICHANDLE )**

Delete the Diagnostic object.

Supported Platform(s): XM, XA, XS

**Returns**

-

**Example Usage:**

```
DIAGNOSTICHANDLE pDiagnostic = ::GetDiagnostic();
assert(pDiagnostic);

diagnostic_example(pDiagnostic);

Diagnostic_release(pDiagnostic);
```

### 5.1.3.119 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::DigIO\_getDigIO ( DIGIOHANDLE , unsigned char \* status )

Get Digital inputs.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>status</i>	Status of the four digital input pins. Bit0: Digital input 1. Bit1: Digital input 2. Bit2: Digital input 3. Bit3: Digital input 4. Bit 4..7 are always zero.
---------------	--

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

#### Example Usage:

```
err = DigIO_getDigIO (pDigIO, &inputs);
if (CrossControl::ERR_SUCCESS == err)
{
    cout << "Digital In 1: " <<
        ((inputs & CrossControl::DigitalIn_1) ? "High" : "Low") << endl;
    cout << "Digital In 2: " <<
        ((inputs & CrossControl::DigitalIn_2) ? "High" : "Low") << endl;
    cout << "Digital In 3: " <<
        ((inputs & CrossControl::DigitalIn_3) ? "High" : "Low") << endl;
    cout << "Digital In 4: " <<
        ((inputs & CrossControl::DigitalIn_4) ? "High" : "Low") << endl;
}
else
{
    cout << "Unable to read digital input status." << endl;
}
```

### 5.1.3.120 EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV CrossControl::DigIO\_release ( DIGIOHANDLE )

Delete the DigIO object.

Supported Platform(s): XM, XA, XS

#### Returns

-

#### Example Usage:

```
DIGIOHANDLE pDigIO = ::GetDigIO();
assert (pDigIO);

list_digital_inputs (pDigIO);

DigIO_release (pDigIO);
```

**5.1.3.121** EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::DigIO\_setDigIO ( DIGIOHANDLE , unsigned char *state* )

Set Digital outputs.

Supported Platform(s): XA, XS

**Parameters**

<i>state</i>	State of the four digital output pins. Bit0: Digital output 1. Bit1: Digital output 2. Bit2: Digital output 3. Bit3: Digital output 4. Bit 4..7 not used.
--------------	---

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```
err = DigIO_setDigIO (pDigIO, inputs);
if (CrossControl::ERR_SUCCESS == err)
{
  cout << "Digital out set to the status read." << endl;
}
else
{
  cout << "Unable to set digital output status." << endl;
}
```

**5.1.3.122** EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::FirmwareUpgrade\_getUpgradeStatus ( FIRMWAREUPGHANDLE ,  
 UpgradeStatus \* *status*, bool *blocking* )

Gets the status of an upgrade operation. The upgrade status is common for all upgrade and verification methods.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>status</i>	The current status of the upgrade operation.
<i>blocking</i>	Whether or not the function should wait until a new status event has been reported. If blocking is set to false, the function will return immediately with the current status.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.123 EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV  
CrossControl::FirmwareUpgrade.release ( FIRMWAREUPGHANDLE )

Delete the FirmwareUpgrade object.

Supported Platform(s): XM, XA, XS

#### Returns

-

Example Usage:

```
FirmwareUpgrade_release(pFirmwareUpgrade);
```

5.1.3.124 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::FirmwareUpgrade.shutdown ( FIRMWAREUPGHANDLE )

Shut down the operating system.

Supported Platform(s): XM, XA, XS

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.125 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::FirmwareUpgrade.startFpgaUpgrade ( FIRMWAREUPGHANDLE , const  
char \* filename, bool blocking )

Start an upgrade of the FPGA. After a FPGA upgrade, the system should be shut down. Full functionality of the system cannot be guaranteed until a fresh startup has been performed.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>filename</i>	Path and filename to the .mcs file to program.
<i>blocking</i>	Whether or not the function should wait until completion. If blocking is set to false, the function will return immediately. One must then call getUpgradeStatus to get the status of the upgrade operation. If blocking is set to true, the function will return when the operation is complete. This might take a few minutes.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```

cout << "Upgrading FPGA" << endl;

for(int i=0;i<max_retries;i++)
{
    // Reinitialize upgrade handle
    FirmwareUpgrade_release(pFirmwareUpgrade);
    pFirmwareUpgrade = GetFirmwareUpgrade();
    assert(pFirmwareUpgrade != NULL);

    err = FirmwareUpgrade_startFpgaUpgrade(pFirmwareUpgrade, path.c_str(),
        true);
    if (CrossControl::ERR_SUCCESS == err) {
        cout << "Upgrade Ok" << endl;
        break;
    }
    else if(CrossControl::ERR_VERIFY_FAILED == err) {
        // Reinitialize upgrade handle
        FirmwareUpgrade_release(pFirmwareUpgrade);
        pFirmwareUpgrade = GetFirmwareUpgrade();
        assert(pFirmwareUpgrade != NULL);

        err = FirmwareUpgrade_startFpgaVerification(pFirmwareUpgrade,
            path.c_str(), true);

        if (CrossControl::ERR_SUCCESS == err) {
            cout << "Upgrade Ok" << endl;
            break;
        }
    }
    else
    {
        cout << "Error " << err << " in function startFpgaUpgrade: " <<
            GetErrorStringA(err) << std::endl;
    }
}
}

```

**5.1.3.126 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::FirmwareUpgrade\_startFpgaVerification ( FIRMWAREUPGHANDLE ,  
 const char \* filename, bool blocking )**

Start a verification of the FPGA. Verifies the FPGA against the file to program. This could be useful if verification during programming fails.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>filename</i>	Path and filename to the .mcs file to verify against.
<i>blocking</i>	Whether or not the function should wait until completion. If blocking is set to false, the function will return immediately. One must then call getUpgradeStatus to get the status of the operation. If blocking is set to true, the function will return when the operation is complete. This might take a few minutes.



**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```

cout << "Upgrading FPGA" << endl;

for(int i=0;i<max_retries;i++)
{
    // Reinitialize upgrade handle
    FirmwareUpgrade_release(pFirmwareUpgrade);
    pFirmwareUpgrade = GetFirmwareUpgrade();
    assert(pFirmwareUpgrade != NULL);

    err = FirmwareUpgrade_startFpgaUpgrade(pFirmwareUpgrade, path.c_str(),
        true);
    if (CrossControl::ERR_SUCCESS == err) {
        cout << "Upgrade Ok" << endl;
        break;
    }
    else if (CrossControl::ERR_VERIFY_FAILED == err) {
        // Reinitialize upgrade handle
        FirmwareUpgrade_release(pFirmwareUpgrade);
        pFirmwareUpgrade = GetFirmwareUpgrade();
        assert(pFirmwareUpgrade != NULL);

        err = FirmwareUpgrade_startFpgaVerification(pFirmwareUpgrade,
            path.c_str(), true);

        if (CrossControl::ERR_SUCCESS == err) {
            cout << "Upgrade Ok" << endl;
            break;
        }
    }
    else
    {
        cout << "Error " << err << " in function startFpgaUpgrade: " <<
            GetErrorStringA(err) << std::endl;
    }
}

```

**5.1.3.127 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**

**CrossControl::FirmwareUpgrade\_startFrontUpgrade ( FIRMWAREUPGHANDLE ,  
const char \* filename, bool blocking )**

Start an upgrade of the front microprocessor. After a front upgrade, the system should be shut down. The front will not work until a fresh startup has been performed.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>filename</i>	Path and filename to the .hex file to program.
<i>blocking</i>	Whether or not the function should wait until completion. If blocking is set to false, the function will return immediately. One must then call fpgaUpgradeStatus to get the status of the upgrade operation. If blocking is set to true, the function will return when the operation is complete. This might take a few minutes.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```

cout << "Upgrading front" << endl;

for(int i=0;i<max_retries;i++)
{
    // Reinitialize upgrade handle
    FirmwareUpgrade_release(pFirmwareUpgrade);
    pFirmwareUpgrade = GetFirmwareUpgrade();
    assert(pFirmwareUpgrade != NULL);

    err = FirmwareUpgrade_startFrontUpgrade(pFirmwareUpgrade, path.c_str()
        , true);
    if (CrossControl::ERR_SUCCESS == err) {
        cout << "Upgrade Ok" << endl;
        break;
    }
    else if(CrossControl::ERR_VERIFY_FAILED == err) {
        // Reinitialize upgrade handle
        FirmwareUpgrade_release(pFirmwareUpgrade);
        pFirmwareUpgrade = GetFirmwareUpgrade();
        assert(pFirmwareUpgrade != NULL);

        err = FirmwareUpgrade_startFrontVerification(pFirmwareUpgrade,
            path.c_str(), true);

        if (CrossControl::ERR_SUCCESS == err) {
            cout << "Upgrade Ok" << endl;
            break;
        }
    }
    else
    {
        cout << "Error " << err << " in function startFrontUpgrade: " <<
            GetErrorStringA(err) << std::endl;
    }
}

```

**5.1.3.128 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**

**CrossControl::FirmwareUpgrade\_startFrontVerification ( FIRMWAREUPGHANDLE ,  
const char \* filename, bool blocking )**

Start a verification of the front microprocessor. Verifies the front microprocessor against the file to program. This could be useful if verification during programming fails.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>filename</i>	Path and filename to the .hex file to verify against.
<i>blocking</i>	Whether or not the function should wait until completion. If blocking is set to false, the function will return immediately. One must then call getUpgradeStatus to get the status of the operation. If blocking is set to true, the function will return when the operation is complete. This might take a few minutes.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```

cout << "Upgrading front" << endl;

for(int i=0;i<max_retries;i++)
{
    // Reinitialize upgrade handle
    FirmwareUpgrade_release(pFirmwareUpgrade);
    pFirmwareUpgrade = GetFirmwareUpgrade();
    assert(pFirmwareUpgrade != NULL);

    err = FirmwareUpgrade_startFrontUpgrade(pFirmwareUpgrade, path.c_str()
    , true);
    if (CrossControl::ERR_SUCCESS == err) {
        cout << "Upgrade Ok" << endl;
        break;
    }
    else if(CrossControl::ERR_VERIFY_FAILED == err) {
        // Reinitialize upgrade handle
        FirmwareUpgrade_release(pFirmwareUpgrade);
        pFirmwareUpgrade = GetFirmwareUpgrade();
        assert(pFirmwareUpgrade != NULL);

        err = FirmwareUpgrade_startFrontVerification(pFirmwareUpgrade,
        path.c_str(), true);

        if (CrossControl::ERR_SUCCESS == err) {
            cout << "Upgrade Ok" << endl;
            break;
        }
    }
    else
    {
        cout << "Error " << err << " in function startFrontUpgrade: " <<
        GetErrorStringA(err) << std::endl;
    }
}

```

**5.1.3.129 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**

**CrossControl::FirmwareUpgrade\_startSSUpgrade ( FIRMWAREUPGHANDLE , const char \* filename, bool blocking )**

Start an upgrade of the System Supervisor microprocessor (SS). After an SS upgrade, the system must be shut down. The SS handles functions for shutting down of the computer. In order to shut down after an upgrade, shut down the OS and then toggle the power. The backlight will still be on after the OS has shut down.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>filename</i>	Path and filename to the .hex file to program.
<i>blocking</i>	Whether or not the function should wait until completion. If blocking is set to false, the function will return immediately. One must then call fpgaUpgradeStatus to get the status of the upgrade operation. If blocking is set to true, the function will return when the operation is complete. This might take a few minutes.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```

cout << "Upgrading SS" << endl;

for(int i=0;i<max_retries;i++)
{
    // Reinitialize upgrade handle
    FirmwareUpgrade_release(pFirmwareUpgrade);
    pFirmwareUpgrade = GetFirmwareUpgrade();
    assert(pFirmwareUpgrade != NULL);

    err = FirmwareUpgrade_startSSUpgrade(pFirmwareUpgrade, path.c_str(), true
    );
    if (CrossControl::ERR_SUCCESS == err) {
        cout << "Upgrade Ok" << endl;
        break;
    }
    else if(CrossControl::ERR_VERIFY_FAILED == err) {
        // Reinitialize upgrade handle
        FirmwareUpgrade_release(pFirmwareUpgrade);
        pFirmwareUpgrade = GetFirmwareUpgrade();
        assert(pFirmwareUpgrade != NULL);

        err = FirmwareUpgrade_startSSVerification(pFirmwareUpgrade, path.
        c_str(), true);

        if (CrossControl::ERR_SUCCESS == err) {
            cout << "Upgrade Ok" << endl;
            break;
        }
    }
    else
    {
        cout << "Error " << err << " in function startSSUpgrade: " <<
        GetErrorStringA(err) << std::endl;
    }
}
}

```

**5.1.3.130 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::FirmwareUpgrade\_startSSVerification ( FIRMWAREUPGHANDLE ,  
 const char \* filename, bool blocking )**

Start a verification of the System Supervisor microprocessor (SS). Verifies the SS against the file to program. This could be useful if verification during programming fails.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>filename</i>	Path and filename to the .hex file to verify against.
<i>blocking</i>	Whether or not the function should wait until completion. If blocking is set to false, the function will return immediately. One must then call getUpgradeStatus to get the status of the operation. If blocking is set to true, the function will return when the operation is complete. This might take a few minutes.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```

cout << "Upgrading SS" << endl;

for(int i=0;i<max_retries;i++)
{
    // Reinitialize upgrade handle
    FirmwareUpgrade_release(pFirmwareUpgrade);
    pFirmwareUpgrade = GetFirmwareUpgrade();
    assert(pFirmwareUpgrade != NULL);

    err = FirmwareUpgrade_startSSUpgrade(pFirmwareUpgrade, path.c_str(), true
);
    if (CrossControl::ERR_SUCCESS == err) {
        cout << "Upgrade Ok" << endl;
        break;
    }
    else if (CrossControl::ERR_VERIFY_FAILED == err) {
        // Reinitialize upgrade handle
        FirmwareUpgrade_release(pFirmwareUpgrade);
        pFirmwareUpgrade = GetFirmwareUpgrade();
        assert(pFirmwareUpgrade != NULL);

        err = FirmwareUpgrade_startSSVerification(pFirmwareUpgrade, path.
c_str(), true);

        if (CrossControl::ERR_SUCCESS == err) {
            cout << "Upgrade Ok" << endl;
            break;
        }
    }
    else
    {
        cout << "Error " << err << " in function startSSUpgrade: " <<
GetErrorStringA(err) << std::endl;
    }
}

```

**5.1.3.131 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::FrontLED\_getColor ( FRONTLEDHANDLE , unsigned char \* *red*,  
unsigned char \* *green*, unsigned char \* *blue* )**

Get front LED color mix.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>red</i>	Red color intensity 0-0x0F.
<i>green</i>	Green color intensity 0-0x0F.
<i>blue</i>	Blue color intensity 0-0x0F.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

## Example Usage:

```
err = FrontLED_getColor(pFrontLED, &red, &green, &blue);
if(err != ERR_SUCCESS)
{
    cout << "Error(" << err << ") in function getColor: " << GetErrorStringA(err) << endl;
}
```

**5.1.3.132 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::FrontLED\_getEnabledDuringStartup ( FRONTLEDHANDLE , CCStatus  
\* *status* )**

Is the front LED enabled during startup? If enabled, the LED will blink yellow to indicate startup progress. It will turn green once the OS has started.

Supported Platform(s): XM, XA, XS

## Parameters

<i>status</i>	LED Enabled or Disabled during startup.
---------------	---

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.133 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::FrontLED\_getIdleTime ( FRONTLEDHANDLE , unsigned char \*  
*idleTime* )**

Get front LED idle time.

Supported Platform(s): XM, XA, XS

## Parameters

<i>idleTime</i>	Time in 100ms increments.
-----------------	---------------------------

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.134 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::FrontLED\_getNrOfPulses ( FRONTLEDHANDLE , unsigned char \*  
*nrOfPulses* )**

Get number of pulses during a blink sequence.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>nrOfPulses</i>	Number of pulses.
-------------------	-------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.135 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::FrontLED\_getOffTime ( FRONTLEDHANDLE , unsigned char \* *offTime* )

Get front LED off time.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>offTime</i>	Time in 10ms increments.
----------------	--------------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.136 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::FrontLED\_getOnTime ( FRONTLEDHANDLE , unsigned char \* *onTime* )

Get front LED on time.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>onTime</i>	Time in 10ms increments. 0 = off
---------------	----------------------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.137 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**  
**CrossControl::FrontLED\_getSignal ( FRONTLEDHANDLE , double \* *frequency*, unsigned char \* *dutyCycle* )**

Get front LED signal. Note, the values may vary from previously set values with setSignal. This is due to precision-loss in approximations.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>frequency</i>	LED blink frequency (0.2-50 Hz).
<i>dutyCycle</i>	LED on duty cycle (0-100%).

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```
err = FrontLED_getSignal(pFrontLED, &freq, &dutyCycle);
if(err != ERR_SUCCESS)
{
    cout << "Error(" << err << ") in function getSignal: " << GetErrorStringA(err) << endl;
}
```

**5.1.3.138 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**  
**CrossControl::FrontLED\_getStandardColor ( FRONTLEDHANDLE , CCAuxColor \* *color* )**

Get front LED color from a set of standard colors. If the color is not one of the predefined colors, UNDEFINED\_COLOR will be returned.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>color</i>	Color from CCAuxColor enum.
--------------	-----------------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.



### 5.1.3.139 EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV CrossControl::FrontLED\_release ( FRONTLEDHANDLE )

Delete the FrontLED object.

Supported Platform(s): XM, XA, XS

#### Returns

-

Example Usage:

```
FRONTLEDHANDLE pFrontLED = ::GetFrontLED();
assert(pFrontLED);

led_example(pFrontLED);

FrontLED_release(pFrontLED);
```

### 5.1.3.140 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::FrontLED\_setColor ( FRONTLEDHANDLE , unsigned char *red*, unsigned char *green*, unsigned char *blue* )

Set front LED color mix.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>red</i>	Red color intensity 0-0x0F.
<i>green</i>	Green color intensity 0-0x0F.
<i>blue</i>	Blue color intensity 0-0x0F.

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

```
err = FrontLED_setColor(pFrontLED, red, green, blue);
if(err != ERR_SUCCESS)
{
    cout << "Error(" << err << ") in function setColor: " << GetErrorStringA(err) << endl;
}
```

5.1.3.141 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::FrontLED\_setEnabledDuringStartup ( FRONTLEDHANDLE , CCStatus  
*status* )

Should the front LED be enabled during startup? If enabled, the LED will blink yellow to indicate startup progress. It will turn green once the OS has started.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>status</i>	Enable or Disable the LED during startup.
---------------	---

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.142 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::FrontLED\_setIdleTime ( FRONTLEDHANDLE , unsigned char *idleTime* )

Get front LED idle time.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>idleTime</i>	Time in 100ms.
-----------------	----------------

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.143 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::FrontLED\_setNrOfPulses ( FRONTLEDHANDLE , unsigned char  
*nrOfPulses* )

Set front LED number of pulses during a blink sequence.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>nrOfPulses</i>	Number of pulses.
-------------------	-------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.144 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::FrontLED\_setOff ( FRONTLEDHANDLE )**

Set front LED off.

Supported Platform(s): XM, XA, XS

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.145 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::FrontLED\_setOffTime ( FRONTLEDHANDLE , unsigned char *offTime* )**

Set front LED off time.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>offTime</i>	Time in 10ms increments.
----------------	--------------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```
err = FrontLED_setOffTime(pFrontLED, 25);
if(err != ERR_SUCCESS)
{
    cout << "Error(" << err << ") in function setOfftime: " << GetErrorStringA(err) << endl;
}
```

**5.1.3.146 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::FrontLED\_setOnTime ( FRONTLEDHANDLE , unsigned char *onTime* )**

Set front LED on time.

Supported Platform(s): XM, XA, XS

## Parameters

<i>onTime</i>	Time in 10ms increments. 0 = off
---------------	----------------------------------

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

## Example Usage:

```
err = FrontLED_setOnTime(pFrontLED, 25);
if(err != ERR_SUCCESS)
{
    cout << "Error(" << err << ") in function setOnTime: " << GetErrorStringA(err) << endl;
}
```

**5.1.3.147 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**  
**CrossControl::FrontLED\_setSignal ( FRONTLEDHANDLE , double *frequency*,  
 unsigned char *dutyCycle* )**

Set front LED signal.

Supported Platform(s): XM, XA, XS

## Parameters

<i>frequency</i>	LED blink frequency (0.2-50 Hz).
<i>dutyCycle</i>	LED on duty cycle (0-100%).

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

## Example Usage:

```
err = FrontLED_setSignal(pFrontLED, freq, dutycycle);
if(err != ERR_SUCCESS)
{
    cout << "Error(" << err << ") in function setSignal: " << GetErrorStringA(err) << endl;
}
```

**5.1.3.148 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**  
**CrossControl::FrontLED\_setStandardColor ( FRONTLEDHANDLE , CCAuxColor *color*  
 )**

Set one of the front LED standard colors.

Supported Platform(s): XM, XA, XS

## Parameters

<i>color</i>	Color from CCAuxColor enum.
--------------	-----------------------------

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

## Example Usage:

```
err = FrontLED_setStandardColor(pFrontLED, RED);
if(err != ERR_SUCCESS)
{
    cout << "Error(" << err << ") in function setStandardColor: " <<
        GetErrorStringA(err) << endl;
}
```

#### 5.1.3.149 EXTERN\_C CCAUXDLL\_API ABOUTHANDLE CCAUXDLL\_CALLING\_CONV CrossControl::GetAbout ( void )

Factory function that creates instances of the About object.

Supported Platform(s): XM, XA, XS

## Returns

ABOUTHANDLE to an allocated About object. The returned handle needs to be deallocated using the [About\\_release\(ABOUTHANDLE\)](#) method when it's no longer needed.

Returns NULL if it fails to allocate memory.

## Example Usage:

```
ABOUTHANDLE pAbout = ::GetAbout();
assert(pAbout);

list_about_information(pAbout);

About_release(pAbout);
```

#### 5.1.3.150 EXTERN\_C CCAUXDLL\_API ADCHANDLE CCAUXDLL\_CALLING\_CONV CrossControl::GetAdc ( void )

Factory function that creates instances of the Adc object.

Supported Platform(s): XM, XA, XS

**Returns**

ADCHANDLE to an allocated Adc object. The returned handle needs to be deallocated using the [Adc\\_release\(ADCHANDLE\)](#) method when it's no longer needed.

Returns NULL if it fails to allocate memory.

**Example Usage:**

```
ADCHANDLE pAdc = ::GetAdc();
assert (pAdc);

output_voltage (pAdc, "24VIN", CrossControl::VOLTAGE_24VIN);
output_voltage (pAdc, "24V", CrossControl::VOLTAGE_24V);
output_voltage (pAdc, "12V", CrossControl::VOLTAGE_12V);
output_voltage (pAdc, "12VID", CrossControl::VOLTAGE_12VID);
output_voltage (pAdc, "5V", CrossControl::VOLTAGE_5V);
output_voltage (pAdc, "3V3", CrossControl::VOLTAGE_3V3);
output_voltage (pAdc, "VTFT", CrossControl::VOLTAGE_VTFT);
output_voltage (pAdc, "5VSTB", CrossControl::VOLTAGE_5VSTB);
output_voltage (pAdc, "1V9", CrossControl::VOLTAGE_1V9);
output_voltage (pAdc, "1V8", CrossControl::VOLTAGE_1V8);
output_voltage (pAdc, "1V5", CrossControl::VOLTAGE_1V5);
output_voltage (pAdc, "1V2", CrossControl::VOLTAGE_1V2);
output_voltage (pAdc, "1V05", CrossControl::VOLTAGE_1V05);
output_voltage (pAdc, "1V0", CrossControl::VOLTAGE_1V0);
output_voltage (pAdc, "0V9", CrossControl::VOLTAGE_0V9);
output_voltage (pAdc, "VREF_INT", CrossControl::VOLTAGE_VREF_INT);
output_voltage (pAdc, "24V_BACKUP", CrossControl::VOLTAGE_24V_BACKUP);
output_voltage (pAdc, "2V5", CrossControl::VOLTAGE_2V5);
output_voltage (pAdc, "1V1", CrossControl::VOLTAGE_1V1);
output_voltage (pAdc, "1V3_PER", CrossControl::VOLTAGE_1V3_PER);
output_voltage (pAdc, "1V3_VDDA", CrossControl::VOLTAGE_1V3_VDDA);

Adc_release (pAdc);
```

### 5.1.3.151 EXTERN\_C CCAUXDLL\_API AUXVERSIONHANDLE CCAUXDLL\_CALLING\_CONV CrossControl::GetAuxVersion ( void )

Factory function that creates instances of the AuxVersion object.

Supported Platform(s): XM, XA, XS

**Returns**

AUXVERSIONHANDLE to an allocated AuxVersion object. The returned handle needs to be deallocated using the [AuxVersion\\_release\(AUXVERSIONHANDLE\)](#) method when it's no longer needed. Returns NULL if it fails to allocate memory.

**Example Usage:**

```
AUXVERSIONHANDLE pAuxVersion = ::GetAuxVersion();
assert (pAuxVersion);

output_versions (pAuxVersion);

AuxVersion_release (pAuxVersion);
```

**5.1.3.152 EXTERN\_C CCAUXDLL\_API BACKLIGHTHANDLE  
CCAUXDLL\_CALLING\_CONV CrossControl::GetBacklight ( void )**

Factory function that creates instances of the Backlight object.

Supported Platform(s): XM, XA, XS

**Returns**

BACKLIGHTHANDLE to an allocated Backlight object. The returned handle needs to be deallocated using the [Backlight\\_release\(BACKLIGHTHANDLE\)](#) method when it's no longer needed. Returns NULL if it fails to allocate memory.

Example Usage:

```
BACKLIGHTHANDLE pBacklight = ::GetBacklight();  
assert (pBacklight);  
  
change_backlight (pBacklight);  
  
Backlight_release (pBacklight);
```

**5.1.3.153 EXTERN\_C CCAUXDLL\_API BATTERYHANDLE CCAUXDLL\_CALLING\_CONV  
CrossControl::GetBattery ( void )**

Factory function that creates instances of the Battery object.

Supported Platform(s): XM

**Returns**

BATTERYHANDLE to an allocated battery object. The returned handle needs to be deallocated using the [Battery\\_release\(BATTERYHANDLE\)](#) method when it's no longer needed. Returns NULL if it fails to allocate memory.

Example Usage:

```
BATTERYHANDLE pBattery = ::GetBattery();  
assert (pBattery);  
  
readBatteryInfo (pBattery);  
  
Battery_release (pBattery);
```

**5.1.3.154 EXTERN\_C CCAUXDLL\_API BUZZERHANDLE CCAUXDLL\_CALLING\_CONV  
CrossControl::GetBuzzer ( void )**

Factory function that creates instances of the Buzzer object.

Supported Platform(s): XM, XA, XS

**Returns**

BUZZERHANDLE to an allocated Buzzer object. The returned handle needs to be deallocated using the [Buzzer\\_release\(BUZZERHANDLE\)](#) method when it's no longer needed. Returns NULL if it fails to allocate memory.

**Example Usage:**

```
BUZZERHANDLE pBuzzer = ::GetBuzzer();
assert(pBuzzer);

play_beeps(pBuzzer);

Buzzer_release(pBuzzer);
```

### 5.1.3.155 EXTERN\_C CCAUXDLL\_API CANSETTINGHANDLE CCAUXDLL\_CALLING\_CONV CrossControl::GetCanSetting ( void )

Factory function that creates instances of the CanSetting object.

Supported Platform(s): XM, XA, XS

**Returns**

CANSETTINGHANDLE to an allocated CanSetting object. The returned handle needs to be deallocated using the [CanSetting\\_release\(CANSETTINGHANDLE\)](#) method when it's no longer needed. Returns NULL if it fails to allocate memory.

**Example Usage:**

```
CANSETTINGHANDLE pCanSetting = ::GetCanSetting();
assert(pCanSetting);

read_cansettings(pCanSetting);

CanSetting_release(pCanSetting);
```

### 5.1.3.156 EXTERN\_C CCAUXDLL\_API CONFIGHANDLE CCAUXDLL\_CALLING\_CONV CrossControl::GetConfig ( )

Video channel 4 config

Factory function that creates instances of the Config object.

Supported Platform(s): XM, XA, XS

**Returns**

CONFIGHANDLE to an allocated Config object. The returned handle needs to be deallocated using the [Config\\_release\(CONFIGHANDLE\)](#) method when it's no longer needed. Returns NULL if it fails to allocate memory.

**Example Usage:**



```
CONFIGHANDLE pConfig = ::GetConfig();
assert(pConfig);

conf_example(pConfig);

Config_release(pConfig);
```

### 5.1.3.157 EXTERN\_C CCAUXDLL\_API DIAGNOSTICHANDLE CCAUXDLL\_CALLING\_CONV CrossControl::GetDiagnostic ( void )

Factory function that creates instances of the Diagnostic object.

Supported Platform(s): XM, XA, XS

#### Returns

DIAGNOSTICHANDLE to an allocated Diagnostic object. The returned handle needs to be deallocated using the [Diagnostic\\_release\(DIAGNOSTICHANDLE\)](#) method when it's no longer needed. Returns NULL if it fails to allocate memory.

#### Example Usage:

```
DIAGNOSTICHANDLE pDiagnostic = ::GetDiagnostic();
assert(pDiagnostic);

diagnostic_example(pDiagnostic);

Diagnostic_release(pDiagnostic);
```

### 5.1.3.158 EXTERN\_C CCAUXDLL\_API DIGIOHANDLE CCAUXDLL\_CALLING\_CONV CrossControl::GetDigIO ( void )

Factory function that creates instances of the DigIO object.

Supported Platform(s): XM, XA, XS

#### Returns

DIGIOHANDLE to an allocated DigIO object. The returned handle needs to be deallocated using the [DigIO\\_release\(DIGIOHANDLE\)](#) method when it's no longer needed. Returns NULL if it fails to allocate memory.

#### Example Usage:

```
DIGIOHANDLE pDigIO = ::GetDigIO();
assert(pDigIO);

list_digital_inputs(pDigIO);

DigIO_release(pDigIO);
```

5.1.3.159 **EXTERN\_C CCAUXDLL\_API char const\*** CCAUXDLL\_CALLING\_CONV  
**CrossControl::GetErrorStringA ( eErr *errCode* )**

Get a string description of an error code.

**Parameters**

<i>errCode</i>	An error code for which to get a string description.
----------------	--

**Returns**

String description of an error code.

5.1.3.160 **EXTERN\_C CCAUXDLL\_API wchar\_t const\*** CCAUXDLL\_CALLING\_CONV  
**CrossControl::GetErrorStringW ( eErr *errCode* )**

Get a string description of an error code.

**Parameters**

<i>errCode</i>	An error code for which to get a string description.
----------------	--

**Returns**

String description of an error code.

5.1.3.161 **EXTERN\_C CCAUXDLL\_API FIRMWAREUPGHANDLE**  
**CCAUXDLL\_CALLING\_CONV CrossControl::GetFirmwareUpgrade ( void )**

Factory function that creates instances of the Adc object.

Supported Platform(s): XM, XA, XS

**Returns**

FIRMWAREUPGHANDLE to an allocated FirmwareUpgrade object. The returned handle needs to be deallocated using the [FirmwareUpgrade\\_release\(FIRMWAREUPGHANDLE\)](#) method when it's no longer needed. Returns NULL if it fails to allocate memory.

**Example Usage:**

```
FIRMWAREUPGHANDLE pFirmwareUpgrade = GetFirmwareUpgrade();
assert(pFirmwareUpgrade != NULL);
```

**5.1.3.162 EXTERN\_C CCAUXDLL\_API FRONTLEDHANDLE CCAUXDLL\_CALLING\_CONV  
CrossControl::GetFrontLED ( void )**

Factory function that creates instances of the FrontLED object.

Supported Platform(s): XM, XA, XS

**Returns**

FRONTLEDHANDLE to an allocated FrontLED object. The returned handle needs to be deallocated using the [FrontLED\\_release\(FRONTLEDHANDLE\)](#) method when it's no longer needed. Returns NULL if it fails to allocate memory.

Example Usage:

```
FRONTLEDHANDLE pFrontLED = ::GetFrontLED();
assert (pFrontLED);

led_example (pFrontLED);

FrontLED_release (pFrontLED);
```

**5.1.3.163 EXTERN\_C CCAUXDLL\_API char const\* CCAUXDLL\_CALLING\_CONV  
CrossControl::GetHwErrorStatusStringA ( unsigned short *errCode* )**

Get a string description of an error code returned from getHwErrorStatus.

**Parameters**

<i>errCode</i>	An error code for which to get a string description.
----------------	--

**Returns**

String description of an error code.

**5.1.3.164 EXTERN\_C CCAUXDLL\_API wchar\_t const\* CCAUXDLL\_CALLING\_CONV  
CrossControl::GetHwErrorStatusStringW ( unsigned short *errCode* )**

Get a string description of an error code returned from getHwErrorStatus.

**Parameters**

<i>errCode</i>	An error code for which to get a string description.
----------------	--

**Returns**

String description of an error code.

**5.1.3.165 EXTERN\_C CCAUXDLL\_API LIGHTSENSORHANDLE  
CCAUXDLL\_CALLING\_CONV CrossControl::GetLightsensor ( void )**

Factory function that creates instances of the Lightsensor object.

Supported Platform(s): XM, XA, XS

**Returns**

LIGHTSENSORHANDLE to an allocated Lightsensor object. The returned handle needs to be deallocated using the [Lightsensor\\_release\(LIGHTSENSORHANDLE\)](#) method when it's no longer needed. Returns NULL if it fails to allocate memory.

**Example Usage:**

```
LIGHTSENSORHANDLE pLightSensor = ::GetLightsensor();  
assert (pLightSensor);  
  
ls_example (pLightSensor);  
  
Lightsensor_release (pLightSensor);
```

**5.1.3.166 EXTERN\_C CCAUXDLL\_API POWERHANDLE CCAUXDLL\_CALLING\_CONV  
CrossControl::GetPower ( void )**

Factory function that creates instances of the Power object.

Supported Platform(s): XM, XA, XS

**Returns**

POWERHANDLE to an allocated Power object. The returned handle needs to be deallocated using the [Power\\_release\(POWERHANDLE\)](#) method when it's no longer needed. Returns NULL if it fails to allocate memory.

**Example Usage:**

```
POWERHANDLE pPower = ::GetPower();  
assert (pPower);  
  
power_example (pPower);  
  
Power_release (pPower);
```

**5.1.3.167 EXTERN\_C CCAUXDLL\_API POWERMGRHANDLE  
CCAUXDLL\_CALLING\_CONV CrossControl::GetPowerMgr ( void )**

Factory function that creates instances of the PowerMgr object.

Supported Platform(s): XM, XA, XS

**Returns**

POWERMGRHANDLE to an allocated PowerMgr structure. The returned handle needs to be deallocated using the PowerMgr::Release() method when it's no longer needed. Returns NULL if it fails to allocate memory.

**Example Usage:**

```

CrossControl::eErr err;
POWERMGRHANDLE pPowerMgr = ::GetPowerMgr();
BATTERYHANDLE pBattery = ::GetBattery();

assert(pPowerMgr);
assert(pBattery);

// Register a separate exit handler for the case where OS is initiating the shutdown. The Application
// must handle this case itself.
atexit(fnExit);

bool bBatt = false;
Battery_isBatteryPresent(pBattery, &bBatt);
if(bBatt) // Ask user wich configuration to use...
    cout << "Choose configuration to use, 0 - Normal, 1 - Application Controlled , 2 - Battery Suspend" <<
        endl;
else
    cout << "Choose configuration to use, 0 - Normal, 1 - Application Controlled" << endl;

cin >> suspendConfiguration;
Battery_release(pBattery);

// Register that this application needs to delay suspend/shutdown
// This should be done as soon as possible.
// Then the app must poll getPowerMgrStatus() and allow the suspend/shutdown with
// setAppReadyForSuspendOrShutdown().
// Depending on application design, this might be best handled in a separate thread.
err = PowerMgr_registerControlledSuspendOrShutDown(pPowerMgr,
    (PowerMgrConf) suspendConfiguration);

cout << "suspendConfiguration " << suspendConfiguration << endl;

if(err == ERR_SUCCESS)
    cout << "Registered to powerMgr." << endl;
else
    cout << "Error(" << err << ") in function registerControlledSuspendOrShutDown: " <<
        GetErrorStringA(err) << endl;

test_powermgr(pPowerMgr);

PowerMgr_release(pPowerMgr);

```

### 5.1.3.168 EXTERN\_C CCAUXDLL\_API SMARTHANDLE CCAUXDLL\_CALLING\_CONV CrossControl::GetSmart ( void )

Factory function that creates instances of the Smart object.

Supported Platform(s): XM, XA, XS

**Returns**

SMARTHANDLE to an allocated AuxVersion structure. The returned handle needs to be deallocated using the Smart::Release() method when it's no longer needed. Returns NULL if it fails to allocate memory.

Example Usage:

```
SMARTHANDLE pSmart = ::GetSmart();
assert(pSmart);

show_card_data(pSmart);

Smart_release(pSmart);
```

#### 5.1.3.169 EXTERN\_C CCAUXDLL\_API char const\* CCAUXDLL\_CALLING\_CONV CrossControl::GetStartupReasonStringA ( unsigned short *code* )

Get a string description of a startup reason code returned from `getStartupReason`.

Parameters

<i>code</i>	A code for which to get a string description.
-------------	---

Returns

String description of a code.

#### 5.1.3.170 EXTERN\_C CCAUXDLL\_API wchar\_t const\* CCAUXDLL\_CALLING\_CONV CrossControl::GetStartupReasonStringW ( unsigned short *code* )

Get a string description of a startup reason code returned from `getStartupReason`.

Parameters

<i>code</i>	A code for which to get a string description.
-------------	---

Returns

String description of a code.

#### 5.1.3.171 EXTERN\_C CCAUXDLL\_API TELEMATICSHANDLE CCAUXDLL\_CALLING\_CONV CrossControl::GetTelematics ( void )

Factory function that creates instances of the Telematics object.

Supported Platform(s): XM, XA, XS

Returns

TELEMATICSHANDLE to an allocated Telematics object. The returned handle needs to be deallocated using the `Telematics_release(TELEMATICSHANDLE)` method when it's no longer needed. Returns NULL if it fails to allocate memory.

Example Usage:

```
TELEMATICSHANDLE pTelematics = ::GetTelematics();
assert(pTelematics);

telematics_example(pTelematics);

Telematics_release(pTelematics);
```

#### 5.1.3.172 EXTERN\_C CCAUXDLL\_API TOUCHSCREENHANDLE CCAUXDLL\_CALLING\_CONV CrossControl::GetTouchScreen ( void )

Factory function that creates instances of the TouchScreen object.

Supported Platform(s): XM, XA, XS

##### Returns

TOUCHSCREENHANDLE to an allocated TouchScreen object. The returned handle needs to be deallocated using the [TouchScreen\\_release\(TOUCHSCREENHANDLE\)](#) method when it's no longer needed. Returns NULL if it fails to allocate memory.

##### Example Usage:

```
TOUCHSCREENHANDLE pTouchScreen = ::GetTouchScreen();
assert(pTouchScreen);

touchscreen_example(pTouchScreen);

TouchScreen_release(pTouchScreen);
```

#### 5.1.3.173 EXTERN\_C CCAUXDLL\_API TOUCHSCREENCALIBHANDLE CCAUXDLL\_CALLING\_CONV CrossControl::GetTouchScreenCalib ( void )

Factory function that creates instances of the TouchScreenCalib object.

##### Returns

TOUCHSCREENCALIBHANDLE to an allocated TouchScreenCalib object. The returned handle needs to be deallocated using the [TouchScreenCalib\\_release\(TOUCHSCREENCALIBHANDLE\)](#) method when it's no longer needed. Returns NULL if it fails to allocate memory.

#### 5.1.3.174 EXTERN\_C CCAUXDLL\_API VIDEOHANDLE CCAUXDLL\_CALLING\_CONV CrossControl::GetVideo ( void )

Factory function that creates instances of the Video object.

Supported Platform(s): XM, XA, XS

**Returns**

VIDEOHANDLE to an allocated Video object. The returned handle needs to be deallocated using the [Video\\_release\(VIDEOHANDLE\)](#) method when it's no longer needed. Returns NULL if it fails to allocate memory.

**5.1.3.175** EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Lightsensor\_getAverageIlluminance ( LIGHTSENSORHANDLE ,  
 unsigned short \* *value* )

Get average illuminance (light) value from light sensor.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>value</i>	Illuminance value (Lux).
--------------	--------------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```
err = Lightsensor_getAverageIlluminance(pLightSensor, &value);
if(err != ERR_SUCCESS)
{
    cout << "Error(" << err << ") in function getAverageIlluminance: " <<
        GetErrorStringA(err) << endl;
}
```

**5.1.3.176** EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Lightsensor\_getIlluminance ( LIGHTSENSORHANDLE , unsigned short  
 \* *value* )

Get illuminance (light) value from light sensor.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>value</i>	Illuminace value (Lux).
--------------	-------------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**



```
err = Lightsensor_getIlluminance(pLightSensor, &value);
if(err != ERR_SUCCESS)
{
    cout << "Error(" << err << ") in function getIlluminance: " <<
        GetErrorStringA(err) << endl;
}
```

**5.1.3.177 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::Lightsensor\_getIlluminance2 ( LIGHTSENSORHANDLE , unsigned  
short \* *value*, unsigned char \* *ch0*, unsigned char \* *ch1* )**

Get illuminance (light) value from light sensor. The parameters *cho* and *ch1* are raw ADC values read from a TAOS TSL2550 lightsensor.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>value</i>	Illuminance value (Lux).
<i>ch0</i>	Channel0 value.
<i>ch1</i>	Channel1 value.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.178 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::Lightsensor\_getOperatingRange ( LIGHTSENSORHANDLE ,  
LightSensorOperationRange \* *range* )**

Get operating range. The light sensor can operate in two ranges. Standard and extended range. In standard range, the range is smaller but resolution higher. See the TSL2550 data sheet for more information.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>range</i>	Operating range. RangeStandard or RangeExtended.
--------------	--

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**5.1.3.179 EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV  
CrossControl::Lightsensor\_release ( LIGHTSENSORHANDLE )**

Delete the Lightsensor object.

Supported Platform(s): XM, XA, XS

#### Returns

-

Example Usage:

```

LIGHTSENSORHANDLE pLightSensor = ::GetLightSensor();
assert (pLightSensor);

ls_example (pLightSensor);

Lightsensor_release (pLightSensor);

```

#### 5.1.3.180 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::Lightsensor\_setOperatingRange ( LIGHTSENSORHANDLE , LightSensorOperationRange *range* )

Set operating range. The light sensor can operate in two ranges. Standard and extended range. In standard range, the range is smaller but resolution higher. See the TSL2550 data sheet for more information.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>range</i>	Operating range to set. RangeStandard or RangeExtended.
--------------	---

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

#### 5.1.3.181 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::Lightsensor\_startAverageCalc ( LIGHTSENSORHANDLE , unsigned long *averageWndSize*, unsigned long *rejectWndSize*, unsigned long *rejectDeltaInLux*, LightSensorSamplingMode *mode* )

Start average calculation.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>average-WndSize</i>	The average window size in nr of samples.
<i>rejectWnd-Size</i>	The reject window size in nr of samples.

<i>rejectDelta- InLux</i>	The reject delta in lux.
<i>mode</i>	The configured sampling mode.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```
// Start the average calculation background function
// This cannot be used if the automatic backlight function is running.
err = Lightsensor_startAverageCalc(pLightSensor, 5, 5, 50,
    SamplingModeAuto);
if(err == ERR_AVERAGE_CALC_STARTED)
{
    cout << "Error(" << err << ") in function startAverageCalc: " <<
        GetErrorStringA(err) << endl;
    cout << endl << "Please turn off Automatic backlight! (CCsettings - Display tab)" << endl;
    return;
}
else if(err != ERR_SUCCESS)
{
    cout << "Error(" << err << ") in function startAverageCalc: " <<
        GetErrorStringA(err) << endl;
}
```

### 5.1.3.182 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::Lightsensor\_stopAverageCalc ( LIGHTSENSORHANDLE )

Stop average calculation.

Supported Platform(s): XM, XA, XS

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```
err = Lightsensor_stopAverageCalc(pLightSensor);
if(err != ERR_SUCCESS)
{
    cout << "Error(" << err << ") in function stopAverageCalc: " <<
        GetErrorStringA(err) << endl;
}
```

### 5.1.3.183 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::Power\_ackPowerRequest ( POWERHANDLE )

Acknowledge a power request from the system supervisor. This is handled by the service/daemon and should normally not be used by applications unless the [Cross-Control](#) service/daemon is not being run on the system. If that is the case, the following

requests (read by `getButtonPowerTransitionStatus`) should be acknowledged: `BPTS_ShutDown`, `BPTS_Suspend` and `BPTS_Restart`

Supported Platform(s): XM, XA, XS

#### Returns

error status. 0 = `ERR_SUCCESS`, otherwise error code. See the enum `eErr` for details.

**5.1.3.184** `EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV`  
`CrossControl::Power_getBLPowerStatus ( POWERHANDLE , CCStatus * status )`

Get backlight power status.

Supported Platform(s): XM

#### Parameters

<i>status</i>	Backlight power status.
---------------	-------------------------

#### Returns

error status. 0 = `ERR_SUCCESS`, otherwise error code. See the enum `eErr` for details.

#### Example Usage:

```
err = Power_getBLPowerStatus(pPower, &status);
if(err == ERR_SUCCESS)
{
    cout << "Backlight power is " << ((status == Enabled)? "ON" : "OFF") << endl;
}
else
{
    cout << "Error(" << err << ") in function Power_getBLPowerStatus: " <<
        GetErrorStringA(err) << endl;
}
```

**5.1.3.185** `EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV`  
`CrossControl::Power_getButtonPowerTransitionStatus ( POWERHANDLE ,`  
`ButtonPowerTransitionStatus * status )`

Get the current status for front panel button and on/off signal.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>status</i>	The current status. See the definition of <code>ButtonPowerTransitionStatus</code> for details.
---------------	---

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.186 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Power\_getCanOCDStatus ( POWERHANDLE , OCDStatus \* status )**

Get Can power overcurrent detection status. Find out if the Can power supervision has detected overcurrent, likely caused by short circuit problems. The overcurrent detection system will immediately turn of the power if such a condition occurs. After a short while, the system will test again, and if there still is overcurrent, Can power is turned off permanently until the unit is restarted.

Supported Platform(s): XA, XS

**Parameters**

<i>status</i>	The current overcurrent detection status
---------------	--

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```

cout << "Checking overcurrent status... " << endl;
OCDStatus ocdstatus;
err = Power_getCanOCDStatus(pPower, &ocdstatus);
if(err == ERR_NOT_SUPPORTED)
{
    cout << "Not supported." << endl;
}
else if(err != ERR_SUCCESS)
{
    cout << "Error(" << err << ") in function Power_getCanOCDStatus: " <<
        GetErrorStringA(err) << endl;
}
else
{
    cout << "Power_getCanOCDStatus: Can OCD status is: ";
    switch(ocdstatus)
    {
        case OCD_OK: cout << "OCD_OK" << std::endl; break;
        case OCD_OC: cout << "OCD_OC" << std::endl; break;
        case OCD_POWER_OFF: cout << "OCD_POWER_OFF" << std::endl; break;
        default: cout << "ERROR" << std::endl; break;
    }
}

```

**5.1.3.187 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Power\_getCanPowerStatus ( POWERHANDLE , CCStatus \* status )**

Get can power status.

Supported Platform(s): XM, XA, XS

## Parameters

<i>status</i>	Can power status.
---------------	-------------------

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

## 5.1.3.188 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV

CrossControl::Power\_getExtFanPowerStatus ( POWERHANDLE , CCStatus \* *status* )

Get external fan power status.

Supported Platform(s): XM

## Parameters

<i>status</i>	Fan power status.
---------------	-------------------

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

## 5.1.3.189 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV

CrossControl::Power\_getVideoOCDStatus ( POWERHANDLE , OCDStatus \* *status* )

Get Video power overcurrent detection status. Find out if the video power supervision has detected overcurrent, likely caused by short circuit problems. The overcurrent detection system will immediately turn of the power if such a condition occurs. After a short while, the system will test again, and if there still is overcurrent, video power is turned off permanently until the unit is restarted.

Supported Platform(s): XA, XS

## Parameters

<i>status</i>	The current overcurrent detection status
---------------	--

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

## Example Usage:

```
err = Power_getVideoOCDStatus(pPower, &ocdstatus);
if(err == ERR_NOT_SUPPORTED)
```

```

{
    /* Don't print anything */
}
else
    if(err != ERR_SUCCESS)
    {
        cout << "Error(" << err << ") in function Power_getVideoOCDStatus: " <<
            GetErrorStringA(err) << endl;
    }
    else
    {
        cout << "Power_getVideoOCDStatus: Video OCD status is: ";
        switch(ocdstatus)
        {
            case OCD_OK: cout << "OCD_OK" << std::endl; break;
            case OCD_OC: cout << "OCD_OC" << std::endl; break;
            case OCD_POWER_OFF: cout << "OCD_POWER_OFF" << std::endl; break;
            default: cout << "ERROR" << std::endl; break;
        }
    }
}

```

**5.1.3.190 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**  
**CrossControl::Power\_getVideoPowerStatus ( POWERHANDLE , unsigned char \***  
**videoStatus )**

Get Video power status.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>videoStatus</i>	Video power status. Bit0: Video 1. Bit1: Video 2. Bit2: Video 3. Bit3: Video 4. (1=on, 0=off)
--------------------	---

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

#### Example Usage:

```

err = Power_getVideoPowerStatus(pPower, &value);
if(err == ERR_SUCCESS)
{
    cout << "Video power status: " << endl;
    cout << "Video1: " << ((value & 0x01)? "ON" : "OFF") << endl;
    cout << "Video2: " << ((value & 0x02)? "ON" : "OFF") << endl;
    cout << "Video3: " << ((value & 0x04)? "ON" : "OFF") << endl;
    cout << "Video4: " << ((value & 0x08)? "ON" : "OFF") << endl;
}
else
{
    cout << "Error(" << err << ") in function Power_getVideoPowerStatus: " <<
        GetErrorStringA(err) << endl;
}

```

**5.1.3.191 EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV**  
**CrossControl::Power\_release ( POWERHANDLE )**

Delete the Power object.

Supported Platform(s): XM, XA, XS

#### Returns

-

Example Usage:

```
POWERHANDLE pPower = ::GetPower();
assert(pPower);

power_example(pPower);

Power_release(pPower);
```

#### 5.1.3.192 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::Power\_setBLPowerStatus ( POWERHANDLE , CCStatus status )

Set backlight power status.

Supported Platform(s): XM

#### Parameters

<i>status</i>	Backlight power status.
---------------	-------------------------

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

```
cout << "Blinking backlight... " << endl;
cin.sync();
cout << endl << "Press Enter to to turn off the Backlight and then Enter to turn it on again..." << endl;
cin.get();
err = Power_setBLPowerStatus(pPower, Disabled);
cin.sync();
cin.get();
err = Power_setBLPowerStatus(pPower, Enabled);
if(err != ERR_SUCCESS)
{
    cout << "Error(" << err << ") in function Power_setBLPowerStatus: " <<
        GetErrorStringA(err) << endl;
}
```

#### 5.1.3.193 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::Power\_setCanPowerStatus ( POWERHANDLE , CCStatus status )

Set can power status.

Supported Platform(s): XM, XA, XS



**Parameters**

<i>status</i>	Can power status.
---------------	-------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.194 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::Power\_setExtFanPowerStatus ( POWERHANDLE , CCStatus *status* )

Set external fan power status.

Supported Platform(s): XM

**Parameters**

<i>status</i>	Fan power status.
---------------	-------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.195 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::Power\_setVideoPowerStatus ( POWERHANDLE , unsigned char *status* )

Set Video power status.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>status</i>	Video power status. Bit0: Video 1. Bit1: Video 2. Bit2: Video 3. Bit3: Video 4. (1=on, 0=off)
---------------	---

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.196 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::PowerMgr\_getConfiguration ( POWERMGRHANDLE , PowerMgrConf \* *conf* )

Get the configuration that is in use.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>conf</i>	The configuration in use.
-------------	---------------------------

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

#### Example Usage:

```
CrossControl::PowerMgrConf conf;
err = PowerMgr_getConfiguration(pPowerMgr, &conf);
if(err == ERR_SUCCESS)
{
    switch (conf)
    {
        case Normal:
            cout << "PowerMgrConf is now: Normal" << endl; break;
        case ApplicationControlled:
            cout << "PowerMgrConf is now: ApplicationControlled" << endl; break;
        case BatterySuspend:
            cout << "PowerMgrConf is now: BatterySuspend" << endl; break;
    }
}
else
{
    cout << "Error(" << err << ") in function getConfiguration: " <<
    GetErrorStringA(err) << endl;
}
```

#### 5.1.3.197 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::PowerMgr\_getPowerMgrStatus ( POWERMGRHANDLE , PowerMgrStatus \* *status* )

Get the current status of the PowerMgr. This functions should be called periodically, to detect when suspend or shutdown requests arrive.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>status</i>	The current status.
---------------	---------------------

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

#### Example Usage:

```

while(1)
{
    OSSleep(500);

    PowerMgrStatus status;
    err = PowerMgr_getPowerMgrStatus(pPowerMgr, &status);
    if(err == ERR_SUCCESS)
    {
        switch(status)
        {
            case NoRequestsPending: // Wait until a PowerMgr request arrives...
                break;

            case ShutdownPending:
                {
                    // Shutdown by means of power button or on/off signal are caught here.
                    os_shutdown = false;

                    cout << "A shutdown request detected. App should now do what it needs to do before shutdown can
                    be performed." << endl;
                    cout << "Press Enter when ready to shutdown... " << endl;

                    // Make sure to clear cin buffer before read
                    std::cin.clear();
                    std::cin.ignore(100, '\n');
                    cin.get();
                    cout << "Signalling that app is ready..." << endl;
                    err = PowerMgr_setAppReadyForSuspendOrShutdown(pPowerMgr)
                ;
                    if(err != ERR_SUCCESS)
                    {
                        cout << "Error(" << err << ") in function setAppReadyForSuspendOrShutdown: " <<
                        GetErrorStringA(err) << endl;
                    }
                    return; //exit test app
                }
            case SuspendPending:
                {
                    os_shutdown = false;

                    cout << "A suspend request detected. App should now do what it needs to do before suspend can be
                    performed." << endl;
                    cout << "Press Enter when ready to suspend... " << endl;

                    // Make sure to clear cin buffer before read
                    std::cin.clear();
                    std::cin.ignore(100, '\n');
                    cin.get();
                    cout << "Signalling that app is ready..." << endl;
                    err = PowerMgr_setAppReadyForSuspendOrShutdown(pPowerMgr)
                ;
                    if(err != ERR_SUCCESS)
                    {
                        cout << "Error(" << err << ") in function setAppReadyForSuspendOrShutdown: " <<
                        GetErrorStringA(err) << endl;
                    }
                    break;

                }
            default:
                cout << "Error: Invalid status returned from getPowerMgrStatus!" << endl;
                break;
        }
    }

    //Wait for resume after notifying that we are ready to suspend
    if(status == SuspendPending)
    {
        bool b = false;
        while(!b)
        {
            OSSleep(100);
            cout << "." << endl;

            err = PowerMgr_hasResumed(pPowerMgr, &b);
        }
    }
}

```

```

        if(err != ERR_SUCCESS)
        {
            cout << "Error(" << err << ") in function hasResumed: " <<
            GetErrorStringA(err) << endl;
        }
    }
    cout << "System is now resumed from suspend mode!" << endl <<
        "Now we will soon re-register using the registerControlledSuspendOrShutDown function!" << endl;

    // Expecting to get configuration Normal after resume from suspend

    CrossControl::PowerMgrConf conf;
    err = PowerMgr_getConfiguration(pPowerMgr, &conf);
    if(err == ERR_SUCCESS)
    {
        switch (conf)
        {
            case Normal:
                cout << "PowerMgrConf is now: Normal" << endl; break;
            case ApplicationControlled:
                cout << "PowerMgrConf is now: ApplicationControlled" << endl; break;
            case BatterySuspend:
                cout << "PowerMgrConf is now: BatterySuspend" << endl; break;
        }
    }
    else
    {
        cout << "Error(" << err << ") in function getConfiguration: " <<
        GetErrorStringA(err) << endl;
    }

    // Re-register, do this as soon as possible after resume/startup
    PowerMgr_registerControlledSuspendOrShutDown(pPowerMgr,
    setConfiguration);
    if(err == ERR_SUCCESS)
        cout << "Re-registered to powerMgr. Ctrl-C to exit." << endl;
    else
        cout << "Error(" << err << ") in function registerControlledSuspendOrShutDown: " <<
        GetErrorStringA(err) << endl;
    }
}
else
{
    cout << "Error(" << err << ") in function getPowerMgrStatus: " <<
    GetErrorStringA(err) << endl;
}
}
}

```

### 5.1.3.198 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV

**CrossControl::PowerMgr\_hasResumed ( POWERMGRHANDLE , bool \* resumed )**

This function can be used in a suspend-resume scenario. After the application has used `setAppReadyForSuspendOrShutdown()` to init the suspend, this function may be polled in order to detect when the system is up and running again. Calling this function before calling `setAppReadyForSuspendOrShutdown` will return `resumed = true`.

Supported Platform(s): XM, XA, XS

#### Returns

error status. 0 = `ERR_SUCCESS`, otherwise error code. See the enum `eErr` for details.

Example Usage:

```

while(1)
{
    OSSleep(500);

    PowerMgrStatus status;
    err = PowerMgr_getPowerMgrStatus(pPowerMgr, &status);
    if(err == ERR_SUCCESS)
    {
        switch(status)
        {
            case NoRequestsPending: // Wait until a PowerMgr request arrives...
                break;

            case ShutdownPending:
                {
                    // Shutdown by means of power button or on/off signal are caught here.
                    os_shutdown = false;

                    cout << "A shutdown request detected. App should now do what it needs to do before shutdown can
                    be performed." << endl;
                    cout << "Press Enter when ready to shutdown... " << endl;

                    // Make sure to clear cin buffer before read
                    std::cin.clear();
                    std::cin.ignore(100, '\n');
                    cin.get();
                    cout << "Signalling that app is ready..." << endl;
                    err = PowerMgr_setAppReadyForSuspendOrShutdown(pPowerMgr)
                ;
                    if(err != ERR_SUCCESS)
                    {
                        cout << "Error(" << err << ") in function setAppReadyForSuspendOrShutdown: " <<
                        GetErrorStringA(err) << endl;
                    }
                    return; //exit test app
                }
            case SuspendPending:
                {
                    os_shutdown = false;

                    cout << "A suspend request detected. App should now do what it needs to do before suspend can be
                    performed." << endl;
                    cout << "Press Enter when ready to suspend... " << endl;

                    // Make sure to clear cin buffer before read
                    std::cin.clear();
                    std::cin.ignore(100, '\n');
                    cin.get();
                    cout << "Signalling that app is ready..." << endl;
                    err = PowerMgr_setAppReadyForSuspendOrShutdown(pPowerMgr)
                ;
                    if(err != ERR_SUCCESS)
                    {
                        cout << "Error(" << err << ") in function setAppReadyForSuspendOrShutdown: " <<
                        GetErrorStringA(err) << endl;
                    }
                    break;
                }
            default:
                cout << "Error: Invalid status returned from getPowerMgrStatus!" << endl;
                break;
        }
    }

    //Wait for resume after notifying that we are ready to suspend
    if(status == SuspendPending)
    {
        bool b = false;
        while(!b)
        {
            OSSleep(100);
            cout << "." << endl;

            err = PowerMgr_hasResumed(pPowerMgr, &b);
        }
    }
}

```

```

        if(err != ERR_SUCCESS)
        {
            cout << "Error(" << err << ") in function hasResumed: " <<
                GetErrorStringA(err) << endl;
        }
    }
    cout << "System is now resumed from suspend mode!" << endl <<
        "Now we will soon re-register using the registerControlledSuspendOrShutDown function!" << endl;

    // Expecting to get configuration Normal after resume from suspend

    CrossControl::PowerMgrConf conf;
    err = PowerMgr_getConfiguration(pPowerMgr, &conf);
    if(err == ERR_SUCCESS)
    {
        switch (conf)
        {
            case Normal:
                cout << "PowerMgrConf is now: Normal" << endl; break;
            case ApplicationControlled:
                cout << "PowerMgrConf is now: ApplicationControlled" << endl; break;
            case BatterySuspend:
                cout << "PowerMgrConf is now: BatterySuspend" << endl; break;
        }
    }
    else
    {
        cout << "Error(" << err << ") in function getConfiguration: " <<
            GetErrorStringA(err) << endl;
    }

    // Re-register, do this as soon as possible after resume/startup
    PowerMgr_registerControlledSuspendOrShutDown(pPowerMgr,
        setConfiguration);
    if(err == ERR_SUCCESS)
        cout << "Re-registered to powerMgr. Ctrl-C to exit." << endl;
    else
        cout << "Error(" << err << ") in function registerControlledSuspendOrShutDown: " <<
            GetErrorStringA(err) << endl;
    }
}
else
{
    cout << "Error(" << err << ") in function getPowerMgrStatus: " <<
        GetErrorStringA(err) << endl;
}
}
}

```

### 5.1.3.199 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::PowerMgr\_registerControlledSuspendOrShutDown ( POWERMGRHANDLE , PowerMgrConf *conf* )

Configure the PowerMgr. Call this function once initially to turn on the functionality.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>conf</i>	The configuration to use.
-------------	---------------------------

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

## Example Usage:

```

CrossControl::eErr err;
POWERMGRHANDLE pPowerMgr = ::GetPowerMgr();
BATTERYHANDLE pBattery = ::GetBattery();

assert(pPowerMgr);
assert(pBattery);

// Register a separate exit handler for the case where OS is initiating the shutdown. The Application
// must handle this case itself.
atexit(fnExit);

bool bBatt = false;
Battery_isBatteryPresent(pBattery, &bBatt);
if(bBatt) // Ask user wich configuration to use...
    cout << "Choose configuration to use, 0 - Normal, 1 - Application Controlled , 2 - Battery Suspend" <<
        endl;
else
    cout << "Choose configuration to use, 0 - Normal, 1 - Application Controlled" << endl;

cin >> suspendConfiguration;
Battery_release(pBattery);

// Register that this application needs to delay suspend/shutdown
// This should be done as soon as possible.
// Then the app must poll getPowerMgrStatus() and allow the suspend/shutdown with
// setAppReadyForSuspendOrShutdown().
// Depending on application design, this might be best handled in a separate thread.
err = PowerMgr_registerControlledSuspendOrShutDown(pPowerMgr,
    (PowerMgrConf) suspendConfiguration);

cout << "suspendConfiguration " << suspendConfiguration << endl;

if(err == ERR_SUCCESS)
    cout << "Registered to powerMgr." << endl;
else
    cout << "Error(" << err << ") in function registerControlledSuspendOrShutDown: " <<
        GetErrorStringA(err) << endl;

test_powermgr(pPowerMgr);

PowerMgr_release(pPowerMgr);

```

### 5.1.3.200 EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV CrossControl::PowerMgr\_release ( POWERMGRHANDLE )

Delete the PowerMgr object.

Supported Platform(s): XM, XA, XS

#### Returns

-

#### Example Usage:

```

CrossControl::eErr err;
POWERMGRHANDLE pPowerMgr = ::GetPowerMgr();
BATTERYHANDLE pBattery = ::GetBattery();

assert(pPowerMgr);
assert(pBattery);

```

```

// Register a separate exit handler for the case where OS is initiating the shutdown. The Application
// must handle this case itself.
atexit(fnExit);

bool bBatt = false;
Battery_isBatteryPresent(pBattery, &bBatt);
if(bBatt) // Ask user wich configuration to use...
    cout << "Choose configuration to use, 0 - Normal, 1 - Application Controlled , 2 - Battery Suspend" <<
        endl;
else
    cout << "Choose configuration to use, 0 - Normal, 1 - Application Controlled" << endl;

cin >> suspendConfiguration;
Battery_release(pBattery);

// Register that this application needs to delay suspend/shutdown
// This should be done as soon as possible.
// Then the app must poll getPowerMgrStatus() and allow the suspend/shutdown with
// setAppReadyForSuspendOrShutdown().
// Depending on application design, this might be best handled in a separate thread.
err = PowerMgr_registerControlledSuspendOrShutDown(pPowerMgr,
    (PowerMgrConf) suspendConfiguration);

cout << "suspendConfiguration " << suspendConfiguration << endl;

if(err == ERR_SUCCESS)
    cout << "Registered to powerMgr." << endl;
else
    cout << "Error(" << err << ") in function registerControlledSuspendOrShutDown: " <<
        GetErrorStringA(err) << endl;

test_powermgr(pPowerMgr);

PowerMgr_release(pPowerMgr);

```

### 5.1.3.201 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::PowerMgr\_setAppReadyForSuspendOrShutdown ( POWERMGRHANDLE )

Acknowledge that the application is ready for suspend/shutdown. Should be called after a request has been received in order to execute the request. The application must acknowledge a request within 20s from when it arrives.

Supported Platform(s): XM, XA, XS

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

#### Example Usage:

```

while(1)
{
    OSSleep(500);

    PowerMgrStatus status;
    err = PowerMgr_getPowerMgrStatus(pPowerMgr, &status);
    if(err == ERR_SUCCESS)
    {
        switch(status)
        {

```



```

case NoRequestsPending: // Wait until a PowerMgr request arrives...
    break;

case ShutdownPending:
    {
        // Shutdown by means of power button or on/off signal are caught here.
        os_shutdown = false;

        cout << "A shutdown request detected. App should now do what it needs to do before shutdown can
        be performed." << endl;
        cout << "Press Enter when ready to shutdown... " << endl;

        // Make sure to clear cin buffer before read
        std::cin.clear();
        std::cin.ignore(100, '\n');
        cin.get();
        cout << "Signalling that app is ready..." << endl;
        err = PowerMgr_setAppReadyForSuspendOrShutdown(pPowerMgr);
    };

    if(err != ERR_SUCCESS)
    {
        cout << "Error(" << err << ") in function setAppReadyForSuspendOrShutdown: " <<
        GetErrorStringA(err) << endl;
    }
    return; //exit test app
}
case SuspendPending:
    {
        os_shutdown = false;

        cout << "A suspend request detected. App should now do what it needs to do before suspend can be
        performed." << endl;
        cout << "Press Enter when ready to suspend... " << endl;

        // Make sure to clear cin buffer before read
        std::cin.clear();
        std::cin.ignore(100, '\n');
        cin.get();
        cout << "Signalling that app is ready..." << endl;
        err = PowerMgr_setAppReadyForSuspendOrShutdown(pPowerMgr);
    };

    if(err != ERR_SUCCESS)
    {
        cout << "Error(" << err << ") in function setAppReadyForSuspendOrShutdown: " <<
        GetErrorStringA(err) << endl;
    }
    }
    break;

default:
    cout << "Error: Invalid status returned from getPowerMgrStatus!" << endl;
    break;
}

//Wait for resume after notifying that we are ready to suspend
if(status == SuspendPending)
{
    bool b = false;
    while(!b)
    {
        OSSleep(100);
        cout << "." << endl;

        err = PowerMgr_hasResumed(pPowerMgr, &b);
        if(err != ERR_SUCCESS)
        {
            cout << "Error(" << err << ") in function hasResumed: " <<
            GetErrorStringA(err) << endl;
        }
    }
    cout << "System is now resumed from suspend mode!" << endl <<
    "Now we will soon re-register using the registerControlledSuspendOrShutDown function!" << endl;

    // Expecting to get configuration Normal after resume from suspend

```

```

CrossControl::PowerMgrConf conf;
err = PowerMgr_getConfiguration(pPowerMgr, &conf);
if(err == ERR_SUCCESS)
{
    switch (conf)
    {
        case Normal:
            cout << "PowerMgrConf is now: Normal" << endl; break;
        case ApplicationControlled:
            cout << "PowerMgrConf is now: ApplicationControlled" << endl; break;
        case BatterySuspend:
            cout << "PowerMgrConf is now: BatterySuspend" << endl; break;
    }
}
else
{
    cout << "Error(" << err << ") in function getConfiguration: " <<
    GetErrorStringA(err) << endl;
}

// Re-register, do this as soon as possible after resume/startup
PowerMgr_registerControlledSuspendOrShutDown(pPowerMgr,
setConfiguration);
if(err == ERR_SUCCESS)
    cout << "Re-registered to powerMgr. Ctrl-C to exit." << endl;
else
    cout << "Error(" << err << ") in function registerControlledSuspendOrShutDown: " <<
    GetErrorStringA(err) << endl;
}
else
{
    cout << "Error(" << err << ") in function getPowerMgrStatus: " <<
    GetErrorStringA(err) << endl;
}
}
}

```

### 5.1.3.202 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::Smart\_getDeviceSerial ( SMARTHANDLE , char \* buff, int len )

Get serial number of the secondary storage device.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>buff</i>	Text output buffer.
<i>len</i>	Maximum length of the output buffer. If the actual length of the data is greater, an error will be returned. At least an 21 bytes buffer size must be used since the serial number can be 20 bytes + trailing zero.

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

#### Example Usage:

```
char serial[21];
```

```

err = Smart_getDeviceSerial (pSmart, serial, sizeof(serial));
if (ERR_SUCCESS == err)
{
    cout << "Device serial number: " << serial << endl;
}
else
{
    cout << "Error(" << err << ") in function getDeviceSerial: " <<
        GetErrorStringA(err) << endl;
}

```

### 5.1.3.203 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::Smart\_getInitialTime ( SMARTHANDLE , time\_t \* time )

Get the date/time when the SMART monitoring began for this storage device. This time is either when the card first was used or when the system software was updated to support S.M.A.R.T. monitoring for the first time. Logging of time is based on the local time of the computer at the time of logging and may therefore not always be accurate.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>time</i>	A 32bit time_t value representing the number of seconds elapsed since 00:00 hours, Jan 1, 1970 UTC.
-------------	---

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

#### Example Usage:

```

time_t initialTime;
struct tm * timeinfo;
err = Smart_getInitialTime (pSmart, &initialTime);
if (ERR_SUCCESS == err)
{
    cout << "Device was initially timestamped on: ";
    timeinfo = localtime (&initialTime);
    cout << asctime(timeinfo) << endl;
}
else
{
    cout << "Error(" << err << ") in function getInitialTime: " <<
        GetErrorStringA(err) << endl;
}

```

### 5.1.3.204 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::Smart\_getRemainingLifeTime ( SMARTHANDLE , unsigned char \* lifetimepercent )

Get remaining lifetime of the secondary storage device.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>lifetimepercent</i>	The expected remaining lifetime (0..100%).
------------------------	--

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```

unsigned char life;
err = Smart_getRemainingLifeTime (pSmart, &life);
if (ERR_SUCCESS == err)
{
    cout << "Estimated remaining lifetime: " << (int)life << "%" << endl;
}
else
{
    cout << "Error(" << err << ") in function getRemainingLifeTime: " <<
        GetErrorStringA(err) << endl;
}

```

**5.1.3.205 EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV  
 CrossControl::Smart\_release ( SMARTHANDLE )**

Delete the Smart object.

Supported Platform(s): XM, XA, XS

**Returns**

-

**Example Usage:**

```

SMARTHANDLE pSmart = ::GetSmart();
assert(pSmart);

show_card_data(pSmart);

Smart_release(pSmart);

```

**5.1.3.206 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Telematics\_getBTPowerStatus ( TELEMATICSHANDLE , CCStatus \*  
 status )**

Get Bluetooth power status.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>status</i>	Bluetooth power status.
---------------	-------------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```
err = Telematics_getBTPowerStatus(pTelematics, &status);
if(err == ERR_SUCCESS)
{
    cout << "Bluetooth power is " << ((status == Enabled)? "ON" : "OFF") << endl;
}
else if(err == ERR_TELEMATICS_BT_NOT_AVAILABLE)
{
    cout << "getBLPowerStatus: Bluetooth is not available on this platform" << endl;
}
else
{
    cout << "Error(" << err << ") in function getBLPowerStatus: " <<
        GetErrorStringA(err) << endl;
}
```

### 5.1.3.207 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::Telematics\_getBTStartupPowerStatus ( TELEMATICSHANDLE , CCStatus \* *status* )

Get Bluetooth power status at startup and at resume from suspended mode.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>status</i>	Bluetooth power status.
---------------	-------------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```
err = Telematics_getBTStartupPowerStatus(pTelematics, &status);
if(err == ERR_SUCCESS)
{
    cout << "Bluetooth power is " << ((status == Enabled)? "Enabled" : "Disabled") << " at start-up"
        << endl;
}
else if(err == ERR_TELEMATICS_BT_NOT_AVAILABLE)
{
    cout << "getBTStartupPowerStatus: Bluetooth is not available on this platform" << endl;
}
else
{

```

```

    cout << "Error(" << err << ") in function getBTStartUpPowerStatus: " <<
        GetErrorStringA(err) << endl;
}

```

### 5.1.3.208 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::Telematics\_getGPRSPowerStatus ( TELEMATICHANDLE , CCStatus \* status )

Get GPRS power status.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>status</i>	GPRS power status.
---------------	--------------------

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

#### Example Usage:

```

err = Telematics_getGPRSPowerStatus(pTelematics, &status);
if(err == ERR_SUCCESS)
{
    cout << "GSM/GPRS power is " << ((status == Enabled)? "ON" : "OFF") << endl;
}
else if(err == ERR_TELEMATICS_GPRS_NOT_AVAILABLE)
{
    cout << "getGPRSPowerStatus: GSM/GPRS is not available on this platform" << endl;
}
else
{
    cout << "Error(" << err << ") in function getGPRSPowerStatus: " <<
        GetErrorStringA(err) << endl;
}

```

### 5.1.3.209 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::Telematics\_getGPRSStartUpPowerStatus ( TELEMATICHANDLE , CCStatus \* status )

Get GPRS power status at startup and at resume from suspended mode.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>status</i>	GPRS power status.
---------------	--------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```
err = Telematics_getGPRSStartupPowerStatus(pTelematics, &status);
if(err == ERR_SUCCESS)
{
    cout << "GSM/GPRS power is " << ((status == Enabled)? "Enabled" : "Disabled") << " at start-up"
        << endl;
}
else if(err == ERR_TELEMATICS_GPRS_NOT_AVAILABLE)
{
    cout << "getGPRSStartupPowerStatus: GSM/GPRS is not available on this platform" << endl;
}
else
{
    cout << "Error(" << err << ") in function getGPRSStartupPowerStatus: " <<
        GetErrorStringA(err) << endl;
}
```

**5.1.3.210 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**

**CrossControl::Telematics\_getGPSAntennaStatus ( TELEMATICSHANDLE , CCstatus  
\* status )**

Get GPS antenna status. Antenna open/short detection. The status is set to disabled if no antenna is present or a short is detected.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>status</i>	GPS antenna power status.
---------------	---------------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```
err = Telematics_getGPSAntennaStatus(pTelematics, &status);
if(err == ERR_SUCCESS)
{
    cout << "GPS antenna status: " << ((status == Enabled)? "OK" : "ERROR: Open connection or
        short-circuit") << endl;
}
else if(err == ERR_TELEMATICS_GPS_NOT_AVAILABLE)
{
    cout << "getGPSAntennaStatus: GPS is not available on this platform" << endl;
}
else
{
    cout << "Error(" << err << ") in function getGPSAntennaStatus: " <<
        GetErrorStringA(err) << endl;
}
```

**5.1.3.211 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**  
**CrossControl::Telematics\_getGPSPowerStatus ( TELEMATICSHANDLE , CCStatus \* status )**

Get GPS power status. Note that it can take some time after calling setGPSPowerStatus before the status is reported correctly.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>status</i>	GPS power status.
---------------	-------------------

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

#### Example Usage:

```
err = Telematics_getGPSPowerStatus(pTelematics, &status);
if(err == ERR_SUCCESS)
{
    cout << "GPS power is " << ((status == Enabled)? "ON" : "OFF") << endl;
}
else if(err == ERR_TELEMATICS_GPS_NOT_AVAILABLE)
{
    cout << "getGPSPowerStatus: GPS is not available on this platform" << endl;
}
else
{
    cout << "Error(" << err << ") in function getGPSPowerStatus: " <<
        GetErrorStringA(err) << endl;
}
```

**5.1.3.212 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**  
**CrossControl::Telematics\_getGPSStartupPowerStatus ( TELEMATICSHANDLE , CCStatus \* status )**

Get GPS power status at startup and at resume from suspended mode.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>status</i>	GPS power status.
---------------	-------------------

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

#### Example Usage:



```

err = Telematics_getGPSStartUpPowerStatus(pTelematics, &status);
if(err == ERR_SUCCESS)
{
    cout << "GPS power is " << ((status == Enabled)? "Enabled" : "Disabled") << " at start-up" <<
        endl;
}
else if(err == ERR_TELEMATICS_GPS_NOT_AVAILABLE)
{
    cout << "getGPSStartUpPowerStatus: GPS is not available on this platform" << endl;
}
else
{
    cout << "Error(" << err << ") in function getGPSStartUpPowerStatus: " <<
        GetErrorStringA(err) << endl;
}

```

### 5.1.3.213 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::Telematics\_getTelematicsAvailable ( TELEMATICSHANDLE , CCStatus \* status )

Is a telematics add-on card installed?

Supported Platform(s): XM, XA, XS

#### Parameters

<i>status</i>	Enabled if a telematics add-on card is installed, otherwise Disabled.
---------------	---

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

#### Example Usage:

```

err = Telematics_getTelematicsAvailable(pTelematics, &status);
if(err == ERR_SUCCESS)
{
    cout << "Telematics add-on board: " << ((status == Enabled)? "available" : "not available") <<
        endl;
    if(status == Disabled)
        return;
}
else
{
    cout << "Error(" << err << ") in function getTelematicsAvailable: " <<
        GetErrorStringA(err) << endl;
    return;
}

```

### 5.1.3.214 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::Telematics\_getWLANPowerStatus ( TELEMATICSHANDLE , CCStatus \* status )

Get WLAN power status.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>status</i>	WLAN power status.
---------------	--------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```
err = Telematics_getWLANPowerStatus(pTelematics, &status);
if(err == ERR_SUCCESS)
{
    cout << "WLAN power is " << ((status == Enabled)? "ON" : "OFF") << endl;
}
else if(err == ERR_TELEMATICS_WLAN_NOT_AVAILABLE)
{
    cout << "getWLANPowerStatus: WLAN is not available on this platform" << endl;
}
else
{
    cout << "Error(" << err << ") in function getWLANPowerStatus: " <<
        GetErrorStringA(err) << endl;
}
```

### 5.1.3.215 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::Telematics\_getWLANStartUpPowerStatus ( TELEMATICSHANDLE , CCStatus \* *status* )

Get WLAN power status at startup and at resume from suspended mode.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>status</i>	WLAN power status.
---------------	--------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```
err = Telematics_getWLANStartUpPowerStatus(pTelematics, &status);
if(err == ERR_SUCCESS)
{
    cout << "WLAN power is " << ((status == Enabled)? "Enabled" : "Disabled") << " at start-up" <<
        endl;
}
else if(err == ERR_TELEMATICS_WLAN_NOT_AVAILABLE)
{
    cout << "getWLANStartUpPowerStatus: WLAN is not available on this platform" << endl;
}
else
{

```

```

    cout << "Error(" << err << ") in function getWLANStartUpPowerStatus: " <<
        GetErrorStringA(err) << endl;
}

```

#### 5.1.3.216 EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV CrossControl::Telematics\_release ( TELEMATICSHANDLE )

Delete the Telematics object.

Supported Platform(s): XM, XA, XS

#### Returns

-

Example Usage:

```

TELEMATICSHANDLE pTelematics = ::GetTelematics();
assert(pTelematics);

telematics_example(pTelematics);

Telematics_release(pTelematics);

```

#### 5.1.3.217 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::Telematics\_setBTPowerStatus ( TELEMATICSHANDLE , CCStatus status )

Set Bluetooth power status.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>status</i>	Bluetooth power status.
---------------	-------------------------

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

#### 5.1.3.218 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::Telematics\_setBTStartupPowerStatus ( TELEMATICSHANDLE , CCStatus status )

Set Bluetooth power status at startup and at resume from suspended mode.

Supported Platform(s): XM, XA, XS

## Parameters

<i>status</i>	Bluetooth power status.
---------------	-------------------------

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.219 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::Telematics\_setGPRSPowerStatus ( TELEMATICSHANDLE , CCStatus *status* )

Set GPRS modem power status.

Supported Platform(s): XM, XA, XS

## Parameters

<i>status</i>	GPRS modem power status.
---------------	--------------------------

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.220 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::Telematics\_setGPRSStartupPowerStatus ( TELEMATICSHANDLE , CCStatus *status* )

Set GPRS power status at startup and at resume from suspended mode.

Supported Platform(s): XM, XA, XS

## Parameters

<i>status</i>	GPRS power status.
---------------	--------------------

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.221 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::Telematics\_setGPSPowerStatus ( TELEMATICSHANDLE , CCStatus *status* )

Set GPS power status.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>status</i>	GPS power status.
---------------	-------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.222 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Telematics.setGPSStartUpPowerStatus ( TELEMATICSHANDLE ,  
 CCStatus *status* )

Set GPS power status at startup and at resume from suspended mode.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>status</i>	GPS power status.
---------------	-------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.223 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Telematics.setWLANPowerStatus ( TELEMATICSHANDLE , CCStatus  
*status* )

Set WLAN power status.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>status</i>	WLAN power status.
---------------	--------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.224 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Telematics\_setWLANStartUpPowerStatus ( TELEMATICSHANDLE ,  
 CCStatus *status* )

Set WLAN power status at startup and at resume from suspended mode.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>status</i>	WLAN power status.
---------------	--------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.225 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::TouchScreen\_getAdvancedSetting ( TOUCHSCREENHANDLE ,  
 TSAdvancedSettingsParameter *param*, unsigned short \* *data* )

Get advanced touch screen settings. See the description of TSAdvancedSettingsParameter for a description of the parameters.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>param</i>	The setting to get.
<i>data</i>	The current data for the setting.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```
err = TouchScreen_getAdvancedSetting(pTouchScreen,
    TS_DEBOUNCE_TIME, &debouncetime);
if(err == ERR_SUCCESS)
{
    cout << "Touchscreen debounce time is set to: " << (int)debouncetime << " ms" << endl;
}
else
{
```

```

    cout << "Error(" << err << ") in function getAdvancedSetting: " <<
        GetErrorStringA(err) << endl;
}

```

### 5.1.3.226 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::TouchScreen\_getMode ( TOUCHSCREENHANDLE , TouchScreenModeSettings \* *config* )

Get Touch Screen mode. Gets the current mode of the USB profile.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>config</i>	The current mode.
---------------	-------------------

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

#### Example Usage:

```

err = TouchScreen_getMode(pTouchScreen, &ts_mode);
if(err == ERR_SUCCESS)
{
    switch(ts_mode)
    {
        case MOUSE_NEXT_BOOT: cout << "USB profile is set to Mouse profile (active next boot)" <
            < endl; break;
        case TOUCH_NEXT_BOOT: cout << "USB profile is set to Touch profile (active next boot)" <
            < endl; break;
        case MOUSE_NOW: cout << "USB profile is set to Mouse profile" << endl; break;
        case TOUCH_NOW: cout << "USB profile is set to Touch profile" << endl; break;
        default: cout << "Error: invalid setting returned from getMode" << endl; break;
    }
}
else if (err == ERR_NOT_SUPPORTED) {
    cout << "Function TouchScreen_getMode() is not supported on this platform";
}
else
{
    cout << "Error(" << err << ") in function getMode: " << GetErrorStringA(err) << endl;
}

```

### 5.1.3.227 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::TouchScreen\_getMouseRightClickTime ( TOUCHSCREENHANDLE , unsigned short \* *time* )

Get mouse right click time. Applies only to the mouse profile. Use the OS settings for the touch profile.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>time</i>	The right click time, in milliseconds.
-------------	--

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**Example Usage:**

```
err = TouchScreen_getMouseRightClickTime(pTouchScreen, &rightclicktime)
;
if(err == ERR_SUCCESS)
{
    cout << "Right click time is set to: " << (int)rightclicktime << " ms" << endl;
}
else
{
    cout << "Error(" << err << ") in function getMouseRightClickTime: " <<
        GetErrorStringA(err) << endl;
}
```

**5.1.3.228 EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV  
 CrossControl::TouchScreen\_release ( TOUCHSCREENHANDLE )**

Delete the TouchScreen object.

Supported Platform(s): XM, XA, XS

**Returns**

-

**Example Usage:**

```
TOUCHSCREENHANDLE pTouchScreen = ::GetTouchScreen();
assert(pTouchScreen);

touchscreen_example(pTouchScreen);

TouchScreen_release(pTouchScreen);
```

**5.1.3.229 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::TouchScreen\_setAdvancedSetting ( TOUCHSCREENHANDLE ,  
 TSAdvancedSettingsParameter param, unsigned short data )**

Set advanced touch screen settings. See the description of TSAdvancedSettingsParameter for a description of the parameters.

Supported Platform(s): XM, XA, XS

**Parameters**



<i>param</i>	The setting to set.
<i>data</i>	The data value to set.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.230 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::TouchScreen\_setMode ( TOUCHSCREENHANDLE ,  
TouchScreenModeSettings *config* )

Set Touch Screen mode. Sets the mode of the USB profile.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>config</i>	The mode to set.
---------------	------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.231 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::TouchScreen\_setMouseRightClickTime ( TOUCHSCREENHANDLE ,  
unsigned short *time* )

Set mouse right click time. Applies only to the mouse profile. Use the OS settings for the touch profile.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>time</i>	The right click time, in milliseconds.
-------------	--

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.232 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::TouchScreenCalib\_checkCalibrationPointFinished ( TOUCHSCREENCALIBHANDLE , bool \* *finished*, unsigned char *pointNr* )

Check if a calibration point is finished

## Parameters

<i>finished</i>	Is current point finished?
<i>pointNr</i>	Calibration point number (1 to total number of points)

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

5.1.3.233 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::TouchScreenCalib\_getConfigParam ( TOUCHSCREENCALIBHANDLE , CalibrationConfigParam *param*, unsigned short \* *value* )

Get calibration config parameters

## Parameters

<i>param</i>	Config parameter
<i>value</i>	Parameter value

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

5.1.3.234 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::TouchScreenCalib\_getMode ( TOUCHSCREENCALIBHANDLE , CalibrationModeSettings \* *mode* )

Get mode of front controller.

## Parameters

<i>mode</i>	Current calibration mode
-------------	--------------------------

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

5.1.3.235 EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV  
CrossControl::TouchScreenCalib\_release ( TOUCHSCREENCALIBHANDLE )

Delete the TouchScreenCalib object.

## Returns

-

5.1.3.236 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::TouchScreenCalib\_setCalibrationPoint ( TOUCHSCREENCALIBHANDLE  
, unsigned char *pointNr* )

Set calibration point

## Parameters

<i>pointNr</i>	Calibration point number (1 to total number of points)
----------------	--

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

5.1.3.237 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::TouchScreenCalib\_setConfigParam ( TOUCHSCREENCALIBHANDLE ,  
CalibrationConfigParam *param*, unsigned short *value* )

Set calibration config parameters

## Parameters

<i>param</i>	Config parameter
<i>value</i>	parameter value

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**5.1.3.238** EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::TouchScreenCalib\_setMode ( TOUCHSCREENCALIBHANDLE ,  
 CalibrationModeSettings *mode* )

Set mode of front controller.

**Parameters**

<i>mode</i>	Selected calibration mode
-------------	---------------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**5.1.3.239** EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Video\_activateSnapshot ( VIDEOHANDLE , bool *activate* )

To be able to take snapshot the snapshot function has to be active. After activation it takes 120ms before first snapshot can be taken. The Snapshot function can be active all the time. If power consumption and heat is an issue, snapshot may be turned off.

Supported Platform(s): XM (Windows)

**Parameters**

<i>activate</i>	Set to true if the snapshot function shall be active.
-----------------	---

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.240** EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Video\_createBitmap ( VIDEOHANDLE , char \*\* *bmpBuffer*, unsigned  
 long \* *bmpBufSize*, const char \* *rawImgBuffer*, unsigned long *rawImgBufSize*, bool  
*bInterlaced*, bool *bNTSCFormat* )

Create a bitmap from a raw image buffer. The bmp buffer is allocated in the function and has to be deallocated by the application.

Supported Platform(s): XM (Windows)

## Parameters

<i>bmpBuffer</i>	Bitmap ram buffer allocated by the API, has to be deallocated with freeBmpBuffer() by the application.
<i>bmpBufSize</i>	Size of the returned bitmap buffer.
<i>rawImg-Buffer</i>	Raw image buffer from takeSnapshotRaw.
<i>rawImgBuf-Size</i>	Size of the raw image buffer.
<i>bInterlaced</i>	Interlaced, if true the bitmap only contains every second line in the image, to save bandwidth.
<i>bNTSC-Format</i>	True if the video format in rawImageBuffer is NTSC format.

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.241 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::Video\_freeBmpBuffer ( VIDEOHANDLE , char \* *bmpBuffer* )

Free the memory allocated for BMP buffer.

Supported Platform(s): XM (Windows)

## Parameters

<i>bmpBuffer</i>	The bmp buffer to free.
------------------	-------------------------

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.242 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::Video\_getActiveChannel ( VIDEOHANDLE , VideoChannel \* *channel* )

Get the current video channel.

Supported Platform(s): XM, XA, XS

## Parameters

<i>channel</i>	Enum defining available channels.
----------------	-----------------------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.243 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**  
**CrossControl::Video\_getColorKeys ( VIDEOHANDLE , unsigned char \* *rKey*,  
 unsigned char \* *gKey*, unsigned char \* *bKey* )**

Get color key values. Note that the system uses 18 bit colors, so the two least significant bits are not used.

Supported Platform(s): XM

**Parameters**

<i>rKey</i>	Red value.
<i>gKey</i>	Green value.
<i>bKey</i>	Blue value.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.244 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**  
**CrossControl::Video\_getCropping ( VIDEOHANDLE , unsigned char \* *top*, unsigned  
 char \* *left*, unsigned char \* *bottom*, unsigned char \* *right* )**

Get Crop parameters.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>top</i>	Crop top (lines).
<i>left</i>	Crop left (lines).
<i>bottom</i>	Crop bottom (lines).
<i>right</i>	Crop right (lines).

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.245** EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Video\_getDecoderReg ( VIDEOHANDLE , unsigned char  
*decoderRegister*, unsigned char \* *registerValue* )

Get Video decoder bus register. Advanced function for direct access to the video decoder TVP5150AM1 registers.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>decoder-Register</i>	Decoder Register Address.
<i>register-Value</i>	register value.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.246** EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Video\_getDeInterlaceMode ( VIDEOHANDLE , DeInterlaceMode \* *mode*  
 )

Get the deinterlace mode used when decoding the interlaced video stream.

Supported Platform(s): XM

**Parameters**

<i>mode</i>	The current mode. See enum DeInterlaceMode for descriptions of the modes.
-------------	---

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.247** EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Video\_getMirroring ( VIDEOHANDLE , CCStatus \* *mode* )

Get the current mirroring mode of the video image.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>mode</i>	The current mode. Enabled or Disabled.
-------------	--

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.248 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::Video\_getRawImage ( VIDEOHANDLE , unsigned short \* *width*,  
unsigned short \* *height*, float \* *frameRate* )**

Get the raw image size of moving image before any scaling and frame rate. For snapshot the height is 4 row less.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>width</i>	Width of raw image.
<i>height</i>	Height of raw moving image, snapshot are 4 bytes less.
<i>frameRate</i>	Received video frame rate.

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.249 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::Video\_getRotation ( VIDEOHANDLE , VideoRotation \* *rotation* )**

Get the current rotation of the video image.

Supported Platform(s): XA, XS

#### Parameters

<i>rotation</i>	Enum defining the current rotation.
-----------------	-------------------------------------

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.



5.1.3.250 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Video\_getScaling ( VIDEOHANDLE , float \* x, float \* y )

Get Video Scaling (image size). If the deinterlace mode is set to DeInterlace\_Even or DeInterlace\_Odd, this function divides the actual vertical scaling by a factor of two, to get the same scaling factor as set with setScaling.

Supported Platform(s): XM

#### Parameters

<i>x</i>	Horizontal scaling (0.25-4).
<i>y</i>	Vertical scaling (0.25-4 DeInterlace_BOB) (0.125-2 DeInterlace_Even, DeInterlace_Odd).

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.251 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Video\_getStatus ( VIDEOHANDLE , unsigned char \* status )

Video status byte.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>status</i>	Status byte Bit 0: video on/off 0 = Off, 1 = On. Bit 2-1: De-interlacing method, 0 = Only even rows, 1 = Only odd rows, 2 = BOB, 3 = invalid. Bit 3: Mirroring mode, 0 = Off, 1 = On Bit 4: Read or write operation to analogue video decoder in progress. Bit 5: Analogue video decoder ready bit.
---------------	---

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.252 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Video\_getVideoArea ( VIDEOHANDLE , unsigned short \* topLeftX, unsigned short \* topLeftY, unsigned short \* bottomRighthX, unsigned short \* bottomRighthY )

Get the area where video is shown.

Supported Platform(s): XM, XA, XS

## Parameters

<i>topLeftX</i>	Top left X coordinate on screen.
<i>topLeftY</i>	Top left Y coordinate on screen.
<i>bottom-RigthX</i>	Bottom right X coordinate on screen.
<i>bottom-RigthY</i>	Bottom right Y coordinate on screen.

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

## 5.1.3.253 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV

CrossControl::Video\_getVideoStandard ( VIDEOHANDLE , videoStandard \* *standard* )

Get video standard. The video decoder auto detects the video standard of the source.

Supported Platform(s): XM, XA, XS

## Parameters

<i>standard</i>	Video standard.
-----------------	-----------------

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

## 5.1.3.254 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV

CrossControl::Video\_init ( VIDEOHANDLE , unsigned char *deviceNr* )

Initialize a video device. The video device will initially use the following settings: DeInterlace\_BOB and mirroring disabled.

Supported Platform(s): XM, XA, XS

## Parameters

<i>deviceNr</i>	Device to connect to (1,2). Select one of 2 devices to connect to.
-----------------	--

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.255 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::Video\_minimize ( VIDEOHANDLE )

Minimizes the video area. Restore with restore() call.

Supported Platform(s): XM, XA, XS

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.256 EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV  
CrossControl::Video\_release ( VIDEOHANDLE )

Delete the Video object.

Supported Platform(s): XM, XA, XS

**Returns**

-

5.1.3.257 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::Video\_restore ( VIDEOHANDLE )

Restores the video area to the size it was before a minimize() call. Don't use restore if minimize has not been used first.

Supported Platform(s): XM, XA, XS

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.258 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::Video\_setActiveChannel ( VIDEOHANDLE , VideoChannel *channel* )

Sets the active video channel.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>channel</i>	Enum defining available channels.
----------------	-----------------------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.259 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**

**CrossControl::Video.setColorKeys ( VIDEOHANDLE , unsigned char *rKey*, unsigned char *gKey*, unsigned char *bKey* )**

Set color keys. Writes RGB color key values. Note that the system uses 18 bit colors, so the two least significant bits are not used.

Supported Platform(s): XM

**Parameters**

<i>rKey</i>	Red key value.
<i>gKey</i>	Green key value.
<i>bKey</i>	Blue key value.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.260 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**

**CrossControl::Video.setCropping ( VIDEOHANDLE , unsigned char *top*, unsigned char *left*, unsigned char *bottom*, unsigned char *right* )**

Crop video image. Note that the video chip manual says the following about horizontal cropping: The number of pixels of active video must be an even number. The parameters *top* and *bottom* are internally converted to an even number. This is due to the input video being interlaced, a pair of odd/even lines are always cropped together. On XA/XS platforms, cropping from *top*/*bottom* on device 2 (channels 3 and 4) is not supported.

Supported Platform(s): XM, XA, XS

**Parameters**

<i>top</i>	Crop top (0-255 lines).
<i>left</i>	Crop left (0-127 lines).
<i>bottom</i>	Crop bottom (0-255 lines).
<i>right</i>	Crop right (0-127 lines).

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.261 **EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**  
**CrossControl::Video\_setDecoderReg ( VIDEOHANDLE , unsigned char**  
**decoderRegister, unsigned char registerValue )**

Set Video decoder bus register. Advanced function for direct access to the video decoder TVP5150AM1 registers.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>decoder-Register</i>	Decoder Register Address.
<i>register-Value</i>	register value.

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.262 **EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**  
**CrossControl::Video\_setDeInterlaceMode ( VIDEOHANDLE , DeInterlaceMode mode )**

Set the deinterlace mode used when decoding the interlaced video stream.

Supported Platform(s): XM

#### Parameters

<i>mode</i>	The mode to set. See enum DeInterlaceMode for descriptions of the modes.
-------------	--

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.263 **EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**  
**CrossControl::Video\_setMirroring ( VIDEOHANDLE , CCStatus mode )**

Enable or disable mirroring of the video image.

Supported Platform(s): XM, XA, XS

#### Parameters

<i>mode</i>	The mode to set. Enabled or Disabled.
-------------	---------------------------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.264 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Video\_setRotation ( VIDEOHANDLE , VideoRotation rotation )

Set the current rotation of the video image.

Supported Platform(s): XA, XS

**Parameters**

<i>rotation</i>	Enum defining the rotation to set.
-----------------	------------------------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.265 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Video\_setScaling ( VIDEOHANDLE , float x, float y )

Set Video Scaling (image size). If the deinterlace mode is set to DeInterlace\_Even or DeInterlace\_Odd, this function multiplies the vertical scaling by a factor of two, to get the correct image proportions.

Supported Platform(s): XM

**Parameters**

<i>x</i>	Horizontal scaling (0.25-4).
<i>y</i>	Vertical scaling (0.25-4 DeInterlace_BOB) (0.125-2 DeInterlace_Even, DeInterlace_Odd).

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.266 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Video\_setVideoArea ( VIDEOHANDLE , unsigned short *topLeftX*,  
 unsigned short *topLeftY*, unsigned short *bottomRightX*, unsigned short *bottomRightY*  
 )

Set the area where video is shown.

Supported Platform(s): XM, XA, XS

## Parameters

<i>topLeftX</i>	Top left X coordinate on screen.
<i>topLeftY</i>	Top left Y coordinate on screen.
<i>bottom-RightX</i>	Bottom right X coordinate on screen.
<i>bottom-RightY</i>	Bottom right Y coordinate on screen.

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.267** EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::Video\_showFrame ( VIDEOHANDLE )

Copy one frame from camera to the display.

Supported Platform(s): XA, XS

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.268** EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::Video\_showVideo ( VIDEOHANDLE , bool show )

Show or hide the video image. Note that it may take some time before the video is shown and correct input info can be read by getRawImage.

Supported Platform(s): XM, XA, XS

## Parameters

<i>show</i>	True shows the video image.
-------------	-----------------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.269 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**  
**CrossControl::Video\_takeSnapshot ( VIDEOHANDLE , const char \* path, bool bInterlaced )**

Takes a snapshot of the current video image and stores it to a bitmap file. This is a combination of takeSnapShotRaw, getVideoStandard and createBitMap and then storing of the bmpBuffer to file. To be able to take a snapshot, the snapshot function has to be active.

Supported Platform(s): XM (Windows)

**Parameters**

<i>path</i>	The file path to where the image should be stored.
<i>bInterlaced</i>	If true the bitmap only contains every second line in the image, to save bandwidth.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.270 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**  
**CrossControl::Video\_takeSnapshotBmp ( VIDEOHANDLE , char \*\* bmpBuffer, unsigned long \* bmpBufSize, bool bInterlaced, bool bNTSCFormat )**

Takes a snapshot of the current video image and return a data buffer with a bitmap image. The bmp buffer is allocated in the function and has to be deallocated with freeBmpBuffer() by the application. This is a combination of the function takeSnapShotRaw and createBitMap. To be able to take a snapshot, the snapshot function has to be active.

Supported Platform(s): XM (Windows)

**Parameters**

<i>bmpBuffer</i>	Bitmap ram buffer allocated by the API, has to be deallocated with freeBmpBuffer() by the application.
<i>bmpBufSize</i>	Size of the returned bitmap buffer.
<i>bInterlaced</i>	If true the bitmap only contains every second line in the image, to save bandwidth.
<i>bNTSC-Format</i>	True if the video format in rawImageBuffer is NTSC format.



**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.3.271 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::Video\_takeSnapshotRaw ( VIDEOHANDLE , char \* rawImgBuffer,  
unsigned long rawImgBuffSize, bool bInterlaced )**

Takes a snapshot of the current video image and return raw image data. The size of the raw image is when interlaced = false 0x100 + line count \* row count \* 4. The size of the raw image is when interlaced = true 0x100 + line count \* row count \* 2. To be able to take a snapshot, the snapshot function has to be active. This function is blocking until a new frame is available from the decoder. An error will be returned if the decoder doesn't return any frames before a timeout.

Supported Platform(s): XM (Windows)

**Parameters**

<i>rawImg-Buffer</i>	Buffer for image to be stored in.
<i>rawImgBuff-Size</i>	Size of the buffer.
<i>bInterlaced</i>	If true the bitmap only contains every second line in the image, to save bandwidth.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code. See the enum eErr for details.

**5.1.4 Variable Documentation**

**5.1.4.1 const unsigned char DigitalIn\_1 = (1 << 0)**

Bit defines for getDigIO

**5.1.4.2 const unsigned char DigitalIn\_2 = (1 << 1)**

**5.1.4.3 const unsigned char DigitalIn\_3 = (1 << 2)**

**5.1.4.4 const unsigned char DigitalIn\_4 = (1 << 3)**

**5.1.4.5 const unsigned char Video1Conf = (1 << 0)**

Bit defines for getVideoStartupPowerConfig and setVideoStartupPowerConfig

5.1.4.6 `const unsigned char Video2Conf = (1 << 1)`

Video channel 1 config

5.1.4.7 `const unsigned char Video3Conf = (1 << 2)`

Video channel 2 config

5.1.4.8 `const unsigned char Video4Conf = (1 << 3)`

Video channel 3 config

## Chapter 6

# Data Structure Documentation

### 6.1 BatteryTimerType Struct Reference

```
#include <Battery.h>
```

#### Data Fields

- unsigned long [TotRunTimeMain](#)
- unsigned long [TotRunTimeBattery](#)
- unsigned long [RunTime\\_m20](#)
- unsigned long [RunTime\\_m20\\_0](#)
- unsigned long [RunTime\\_0\\_40](#)
- unsigned long [RunTime\\_40\\_60](#)
- unsigned long [RunTime\\_60\\_70](#)
- unsigned long [RunTime\\_70\\_80](#)
- unsigned long [RunTime\\_Above80](#)

#### 6.1.1 Field Documentation

##### 6.1.1.1 unsigned long [RunTime\\_0\\_40](#)

Total runtime in range 0 to -20 deg C (minutes)

##### 6.1.1.2 unsigned long [RunTime\\_40\\_60](#)

Total runtime in range 0 to 40 deg C (minutes)

##### 6.1.1.3 unsigned long [RunTime\\_60\\_70](#)

Total runtime in range 40 to 60 deg C (minutes)

**6.1.1.4 unsigned long RunTime\_70\_80**

Total runtime in range 60 to 70 deg C (minutes)

**6.1.1.5 unsigned long RunTime\_Above80**

Total runtime in range 70 to 80 deg C (minutes)

**6.1.1.6 unsigned long RunTime\_m20**

Total running time on battery power (minutes)

**6.1.1.7 unsigned long RunTime\_m20\_0**

Total runtime below -20 deg C (minutes)

**6.1.1.8 unsigned long TotRunTimeBattery**

Total running time on main power (minutes)

**6.1.1.9 unsigned long TotRunTimeMain**

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

- [IncludeFiles/Battery.h](#)

**6.2 BuzzerSetup Struct Reference**

```
#include <CCAuxTypes.h>
```

**Data Fields**

- unsigned short [frequency](#)
- unsigned short [volume](#)

**6.2.1 Field Documentation****6.2.1.1 unsigned short frequency**

buzzer frequency

### 6.2.1.2 unsigned short volume

buzzer volume

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

- [IncludeFiles/CCAuxTypes.h](#)

## 6.3 FpgaLedTimingType Struct Reference

```
#include <CCAuxTypes.h>
```

### Data Fields

- unsigned char [ledNbr](#)
- unsigned char [onTime](#)
- unsigned char [offTime](#)
- unsigned char [idleTime](#)
- unsigned char [nrOfPulses](#)

### 6.3.1 Field Documentation

#### 6.3.1.1 unsigned char idleTime

LED idle time in 100ms

#### 6.3.1.2 unsigned char ledNbr

Number of LED

#### 6.3.1.3 unsigned char nrOfPulses

Pulses per sequences

#### 6.3.1.4 unsigned char offTime

LED off time in 10ms

#### 6.3.1.5 unsigned char onTime

LED on time in 10ms

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

- [IncludeFiles/CCAuxTypes.h](#)

## 6.4 LedColorMixType Struct Reference

```
#include <CCAuxTypes.h>
```

### Data Fields

- unsigned char [red](#)
- unsigned char [green](#)
- unsigned char [blue](#)

### 6.4.1 Field Documentation

#### 6.4.1.1 unsigned char blue

Blue color intensity 0-0x0F

#### 6.4.1.2 unsigned char green

Green color intensity 0-0x0F

#### 6.4.1.3 unsigned char red

Red color intensity 0-0x0F

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

- [IncludeFiles/CCAuxTypes.h](#)

## 6.5 LedTimingType Struct Reference

```
#include <CCAuxTypes.h>
```

### Data Fields

- unsigned char [onTime](#)
- unsigned char [offTime](#)
- unsigned char [idleTime](#)
- unsigned char [nrOfPulses](#)

### 6.5.1 Field Documentation

#### 6.5.1.1 unsigned char idleTime

LED idle time in 100ms

### 6.5.1.2 unsigned char nrOfPulses

Pulses per sequences

### 6.5.1.3 unsigned char offTime

LED off time in 10ms

### 6.5.1.4 unsigned char onTime

LED on time in 10ms

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

- [IncludeFiles/CCAuxTypes.h](#)

## 6.6 received\_video Struct Reference

```
#include <CCAuxTypes.h>
```

### Data Fields

- unsigned short [received\\_width](#)
- unsigned short [received\\_height](#)
- unsigned char [received\\_framerate](#)

### 6.6.1 Field Documentation

#### 6.6.1.1 unsigned char received\_framerate

#### 6.6.1.2 unsigned short received\_height

#### 6.6.1.3 unsigned short received\_width

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

- [IncludeFiles/CCAuxTypes.h](#)

## 6.7 TimerType Struct Reference

```
#include <CCAuxTypes.h>
```

## Data Fields

- unsigned long [TotRunTime](#)
- unsigned long [TotSuspTime](#)
- unsigned long [TotHeatTime](#)
- unsigned long [RunTime40\\_60](#)
- unsigned long [RunTime60\\_70](#)
- unsigned long [RunTime70\\_80](#)
- unsigned long [Above80RunTime](#)

### 6.7.1 Detailed Description

Diagnostic timer data

### 6.7.2 Field Documentation

#### 6.7.2.1 unsigned long [Above80RunTime](#)

Total runtime in 70-80deg (minutes)

#### 6.7.2.2 unsigned long [RunTime40\\_60](#)

Total heating time (minutes)

#### 6.7.2.3 unsigned long [RunTime60\\_70](#)

Total runtime in 40-60deg (minutes)

#### 6.7.2.4 unsigned long [RunTime70\\_80](#)

Total runtime in 60-70deg (minutes)

#### 6.7.2.5 unsigned long [TotHeatTime](#)

Total suspend time (minutes)

#### 6.7.2.6 unsigned long [TotRunTime](#)

#### 6.7.2.7 unsigned long [TotSuspTime](#)

Total running time (minutes)

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

- [IncludeFiles/CCAuxTypes.h](#)



## 6.8 UpgradeStatus Struct Reference

```
#include <CCAuxTypes.h>
```

### Data Fields

- enum [UpgradeAction](#) `currentAction`
- unsigned char [percent](#)
- [eErr](#) `errorCode`

### 6.8.1 Detailed Description

Upgrade Status

### 6.8.2 Field Documentation

6.8.2.1 enum [UpgradeAction](#) `currentAction`

6.8.2.2 [eErr](#) `errorCode`

Represents the percentage of completion of the current action

6.8.2.3 unsigned char `percent`

The current action.

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

- [IncludeFiles/CCAuxTypes.h](#)

## 6.9 version\_info Struct Reference

```
#include <CCAuxTypes.h>
```

### Data Fields

- unsigned char [major](#)
- unsigned char [minor](#)
- unsigned char [release](#)
- unsigned char [build](#)

### 6.9.1 Field Documentation

#### 6.9.1.1 unsigned char build

version build number

#### 6.9.1.2 unsigned char major

version major number

#### 6.9.1.3 unsigned char minor

version minor number

#### 6.9.1.4 unsigned char release

version release number

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

- [IncludeFiles/CCAuxTypes.h](#)

## 6.10 video\_dec\_command Struct Reference

```
#include <CCAuxTypes.h>
```

### Data Fields

- unsigned char [decoder\\_register](#)
- unsigned char [register\\_value](#)

### 6.10.1 Field Documentation

#### 6.10.1.1 unsigned char decoder\_register

#### 6.10.1.2 unsigned char register\_value

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

- [IncludeFiles/CCAuxTypes.h](#)

## Chapter 7

# File Documentation

### 7.1 IncludeFiles/About.h File Reference

#### Namespaces

- namespace [CrossControl](#)

#### Typedefs

- typedef void \* [ABOUTHANDLE](#)

#### Functions

- EXTERN\_C CCAUXDLL\_API ABOUTHANDLE CCAUXDLL\_CALLING\_CONV [GetAbout](#) (void)
- EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV [About\\_release](#) (ABOUTHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [About\\_getMainPCBSerial](#) (ABOUTHANDLE, char \*buff, int len)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [About\\_getUnitSerial](#) (ABOUTHANDLE, char \*buff, int len)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [About\\_getMainPCBArt](#) (ABOUTHANDLE, char \*buff, int length)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [About\\_getMainManufacturingDate](#) (ABOUTHANDLE, char \*buff, int len)

- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getMainHWversion](#) (ABOUTHANDLE, char \*buff, int len)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getMainProdRev](#) (ABOUTHANDLE, char \*buff, int len)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getMainProdArtNr](#) (ABOUTHANDLE, char \*buff, int len)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getNrOfETHConnections](#) (ABOUTHANDLE, unsigned char \*NrOfConnections)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getNrOfCANConnections](#) (ABOUTHANDLE, unsigned char \*NrOfConnections)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getNrOfVideoConnections](#) (ABOUTHANDLE, unsigned char \*NrOfConnections)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getNrOfUSBConnections](#) (ABOUTHANDLE, unsigned char \*NrOfConnections)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getNrOfSerialConnections](#) (ABOUTHANDLE, unsigned char \*NrOfConnections)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getNrOfDigIOConnections](#) (ABOUTHANDLE, unsigned char \*NrOfConnections)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getIsDisplayAvailable](#) (ABOUTHANDLE, bool \*available)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getIsTouchScreenAvailable](#) (ABOUTHANDLE, bool \*available)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getDisplayResolution](#) (ABOUTHANDLE, char \*buff, int len)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getAddOnPCBSerial](#) (ABOUTHANDLE, char \*buff, int len)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getAddOnPCBArt](#) (ABOUTHANDLE, char \*buff, int length)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getAddOnManufacturingDate](#) (ABOUTHANDLE, char \*buff, int len)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getAddOnHWversion](#) (ABOUTHANDLE, char \*buff, int len)

- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getIsWLANMounted](#) (ABOUTHANDLE, bool \*mounted)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getIsGPSPMounted](#) (ABOUTHANDLE, bool \*mounted)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getIsGPRSMounted](#) (ABOUTHANDLE, bool \*mounted)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getIsBTMounted](#) (ABOUTHANDLE, bool \*mounted)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getFrontPcbRev](#) (ABOUTHANDLE, unsigned char \*major, unsigned char \*minor)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getIsIOExpanderMounted](#) (ABOUTHANDLE, bool \*mounted)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getIOExpanderValue](#) (ABOUTHANDLE, unsigned short \*value)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_hasOsBooted](#) (ABOUTHANDLE, bool \*bootComplete)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [About\\_getIsAnybusMounted](#) (ABOUTHANDLE, bool \*mounted)

## 7.2 IncludeFiles/Adc.h File Reference

### Namespaces

- namespace [CrossControl](#)

### Typedefs

- typedef void \* [ADCHANDLE](#)

### Functions

- EXTERN\_C CCAUXDLL\_API  
ADCHANDLE  
CCAUXDLL\_CALLING\_CONV [GetAdc](#) (void)
- EXTERN\_C CCAUXDLL\_API void  
CCAUXDLL\_CALLING\_CONV [Adc\\_release](#) (ADCHANDLE)

- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Adc\\_getVoltage](#) (ADCHANDLE, VoltageEnum selection, double \*value)

## 7.3 IncludeFiles/AuxVersion.h File Reference

### Namespaces

- namespace [CrossControl](#)

### Typedefs

- typedef void \* [AUXVERSIONHANDLE](#)

### Functions

- EXTERN\_C CCAUXDLL\_API  
AUXVERSIONHANDLE  
CCAUXDLL\_CALLING\_CONV [GetAuxVersion](#) (void)
- EXTERN\_C CCAUXDLL\_API void  
CCAUXDLL\_CALLING\_CONV [AuxVersion\\_release](#) (AUXVERSIONHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [AuxVersion\\_getFPGAVersion](#) (AUXVERSIONHANDLE, unsigned char \*major, unsigned char \*minor, unsigned char \*release, unsigned char \*build)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [AuxVersion\\_getSSVersion](#) (AUXVERSIONHANDLE, unsigned char \*major, unsigned char \*minor, unsigned char \*release, unsigned char \*build)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [AuxVersion\\_getFrontVersion](#) (AUXVERSIONHANDLE, unsigned char \*major, unsigned char \*minor, unsigned char \*release, unsigned char \*build)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [AuxVersion\\_getCCAuxVersion](#) (AUXVERSIONHANDLE, unsigned char \*major, unsigned char \*minor, unsigned char \*release, unsigned char \*build)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [AuxVersion\\_getOSVersion](#) (AUXVERSIONHANDLE, unsigned char \*major, unsigned char \*minor, unsigned char \*release, unsigned char \*build)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [AuxVersion\\_getCCAuxDrvVersion](#) (AUXVERSIONHANDLE, unsigned char \*major, unsigned char \*minor, unsigned char \*release, unsigned char \*build)

## 7.4 IncludeFiles/Backlight.h File Reference

### Namespaces

- namespace [CrossControl](#)

### Typedefs

- typedef void \* [BACKLIGHTHANDLE](#)

### Functions

- EXTERN\_C CCAUXDLL\_API  
BACKLIGHTHANDLE  
CCAUXDLL\_CALLING\_CONV [GetBacklight](#) (void)
- EXTERN\_C CCAUXDLL\_API void  
CCAUXDLL\_CALLING\_CONV [Backlight\\_release](#) (BACKLIGHTHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Backlight\\_getIntensity](#) (BACKLIGHTHANDLE, unsigned char \*intensity)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Backlight\\_setIntensity](#) (BACKLIGHTHANDLE, unsigned char intensity)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Backlight\\_getStatus](#) (BACKLIGHTHANDLE, unsigned char \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Backlight\\_getHWStatus](#) (BACKLIGHTHANDLE, bool \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Backlight\\_startAutomaticBL](#) (BACKLIGHTHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Backlight\\_stopAutomaticBL](#) (BACKLIGHTHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Backlight\\_getAutomaticBLStatus](#) (BACKLIGHTHANDLE, unsigned char \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Backlight\\_setAutomaticBLParams](#) (BACKLIGHTHANDLE, bool bSoftTransitions)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Backlight\\_getAutomaticBLParams](#) (BACKLIGHTHANDLE, bool \*bSoftTransitions, double \*k)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Backlight\\_setAutomaticBLFilter](#) (BACKLIGHTHANDLE, unsigned long averageWndSize, unsigned long rejectWndSize, unsigned long rejectDeltaInLux, LightSensorSamplingMode mode)

- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Backlight\\_getAutomaticBLFilter](#) (BACKLIGHTHANDLE, unsigned long \*averageWndSize, unsigned long \*rejectWndSize, unsigned long \*rejectDeltaInLux, LightSensorSamplingMode \*mode)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Backlight\\_getLedDimming](#) (BACKLIGHTHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Backlight\\_setLedDimming](#) (BACKLIGHTHANDLE, CCStatus status)

## 7.5 IncludeFiles/Battery.h File Reference

### Data Structures

- struct [BatteryTimerType](#)

### Namespaces

- namespace [CrossControl](#)

### Typedefs

- typedef void \* [BATTERYHANDLE](#)

### Enumerations

- enum [ChargingStatus](#) {  
[ChargingStatus\\_NoCharge](#) = 0, [ChargingStatus\\_Charging](#) = 1, [ChargingStatus\\_FullyCharged](#) = 2, [ChargingStatus\\_TempLow](#) = 3,  
[ChargingStatus\\_TempHigh](#) = 4, [ChargingStatus\\_Unknown](#) = 5 }
- enum [PowerSource](#) { [PowerSource\\_Battery](#) = 0, [PowerSource\\_ExternalPower](#) = 1 }
- enum [ErrorStatus](#) {  
[ErrorStatus\\_NoError](#) = 0, [ErrorStatus\\_ThermistorTempSensor](#) = 1, [ErrorStatus\\_SecondaryTempSensor](#) = 2, [ErrorStatus\\_ChargeFail](#) = 3,  
[ErrorStatus\\_Overcurrent](#) = 4, [ErrorStatus\\_Init](#) = 5 }

### Functions

- EXTERN\_C CCAUXDLL\_API  
BATTERYHANDLE  
CCAUXDLL\_CALLING\_CONV [GetBattery](#) (void)
- EXTERN\_C CCAUXDLL\_API void  
CCAUXDLL\_CALLING\_CONV [Battery\\_release](#) (BATTERYHANDLE)



- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Battery\\_isBatteryPresent](#) (BATTERYHANDLE, bool \*batteryIsPresent)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Battery\\_getBatteryVoltageStatus](#) (BATTERYHANDLE, unsigned char \*batteryVoltagePercent)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Battery\\_getBatteryChargingStatus](#) (BATTERYHANDLE, ChargingStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Battery\\_getPowerSource](#) (BATTERYHANDLE, PowerSource \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Battery\\_getBatteryTemp](#) (BATTERYHANDLE, signed short \*temperature)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Battery\\_getHwErrorStatus](#) (BATTERYHANDLE, ErrorStatus \*errorCode)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Battery\\_getTimer](#) (BATTERYHANDLE, BatteryTimerType \*times)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Battery\\_getMinMaxTemp](#) (BATTERYHANDLE, signed short \*minTemp, signed short \*maxTemp)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Battery\\_getBatteryHWversion](#) (BATTERYHANDLE, char \*buff, int len)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Battery\\_getBatterySwVersion](#) (BATTERYHANDLE, unsigned short \*major, unsigned short \*minor, unsigned short \*release, unsigned short \*build)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Battery\\_getBatterySerial](#) (BATTERYHANDLE, char \*buff, int len)

## 7.6 IncludeFiles/Buzzer.h File Reference

### Namespaces

- namespace [CrossControl](#)

### Typedefs

- typedef void \* [BUZZERHANDLE](#)

## Functions

- EXTERN\_C CCAUXDLL\_API BUZZERHANDLE CCAUXDLL\_CALLING\_CONV [GetBuzzer](#) (void)
- EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV [Buzzer\\_release](#) (BUZZERHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Buzzer\\_getFrequency](#) (BUZZERHANDLE, unsigned short \*frequency)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Buzzer\\_getVolume](#) (BUZZERHANDLE, unsigned short \*volume)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Buzzer\\_getTrigger](#) (BUZZERHANDLE, bool \*trigger)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Buzzer\\_setFrequency](#) (BUZZERHANDLE, unsigned short frequency)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Buzzer\\_setVolume](#) (BUZZERHANDLE, unsigned short volume)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Buzzer\\_setTrigger](#) (BUZZERHANDLE, bool trigger)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Buzzer\\_buzze](#) (BUZZERHANDLE, int time, bool blocking)

## 7.7 IncludeFiles/CanSetting.h File Reference

### Namespaces

- namespace [CrossControl](#)

### Typedefs

- typedef void \* [CANSETTINGHANDLE](#)

### Functions

- EXTERN\_C CCAUXDLL\_API CANSETTINGHANDLE CCAUXDLL\_CALLING\_CONV [GetCanSetting](#) (void)

- EXTERN\_C CCAUXDLL\_API void  
CCAUXDLL\_CALLING\_CONV [CanSetting\\_release](#) (CANSETTINGHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [CanSetting\\_getBaudrate](#) (CANSETTINGHANDLE, unsigned char net, unsigned short \*baudrate)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [CanSetting\\_getFrameType](#) (CANSETTINGHANDLE, unsigned char net, CanFrameType \*frameType)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [CanSetting\\_setBaudrate](#) (CANSETTINGHANDLE, unsigned char net, unsigned short baudrate)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [CanSetting\\_setFrameType](#) (CANSETTINGHANDLE, unsigned char net, CanFrameType frameType)

## 7.8 IncludeFiles/CCAuxErrors.h File Reference

### Namespaces

- namespace [CrossControl](#)

### Functions

- EXTERN\_C CCAUXDLL\_API char  
const \*CCAUXDLL\_CALLING\_CONV [GetErrorStringA](#) (eErr errCode)
- EXTERN\_C CCAUXDLL\_API wchar\_t  
const \*CCAUXDLL\_CALLING\_CONV [GetErrorStringW](#) (eErr errCode)

## 7.9 IncludeFiles/CCAuxTypes.h File Reference

### Data Structures

- struct [received\\_video](#)
- struct [video\\_dec\\_command](#)
- struct [version\\_info](#)
- struct [BuzzerSetup](#)
- struct [LedTimingType](#)
- struct [FpgaLedTimingType](#)
- struct [LedColorMixType](#)
- struct [TimerType](#)
- struct [UpgradeStatus](#)

## Namespaces

- namespace [CrossControl](#)

## Typedefs

- typedef struct version\_info [VersionType](#)

## Enumerations

- enum [VoltageEnum](#) {  
[VOLTAGE\\_24VIN](#) = 0, [VOLTAGE\\_24V](#), [VOLTAGE\\_12V](#), [VOLTAGE\\_12-VID](#),  
[VOLTAGE\\_5V](#), [VOLTAGE\\_3V3](#), [VOLTAGE\\_VTFT](#), [VOLTAGE\\_5VSTB](#),  
[VOLTAGE\\_1V9](#), [VOLTAGE\\_1V8](#), [VOLTAGE\\_1V5](#), [VOLTAGE\\_1V2](#),  
[VOLTAGE\\_1V05](#), [VOLTAGE\\_1V0](#), [VOLTAGE\\_0V9](#), [VOLTAGE\\_VREF\\_I-NT](#),  
[VOLTAGE\\_24V\\_BACKUP](#), [VOLTAGE\\_2V5](#), [VOLTAGE\\_1V1](#), [VOLTAGE-\\_1V3\\_PER](#),  
[VOLTAGE\\_1V3\\_VDDA](#) }
- enum [LightSensorOperationRange](#) { [RangeStandard](#) = 0, [RangeExtended](#) = 1 }
- enum [LightSensorSamplingMode](#) { [SamplingModeStandard](#) = 0, [SamplingMode-Extended](#), [SamplingModeAuto](#) }
- enum [CCStatus](#) { [Disabled](#) = 0, [Enabled](#) = 1 }
- enum [eErr](#) {  
[ERR\\_SUCCESS](#) = 0, [ERR\\_OPEN\\_FAILED](#) = 1, [ERR\\_NOT\\_SUPPORTED](#) =  
2, [ERR\\_UNKNOWN\\_FEATURE](#) = 3,  
[ERR\\_DATATYPE\\_MISMATCH](#) = 4, [ERR\\_CODE\\_NOT\\_EXIST](#) = 5, [ERR\\_-  
BUFFER\\_SIZE](#) = 6, [ERR\\_IOCTL\\_FAILED](#) = 7,  
[ERR\\_INVALID\\_DATA](#) = 8, [ERR\\_INVALID\\_PARAMETER](#) = 9, [ERR\\_CRE-  
ATE\\_THREAD](#) = 10, [ERR\\_IN\\_PROGRESS](#) = 11,  
[ERR\\_CHECKSUM](#) = 12, [ERR\\_INIT\\_FAILED](#) = 13, [ERR\\_VERIFY\\_FAILED](#)  
= 14, [ERR\\_DEVICE\\_READ\\_DATA\\_FAILED](#) = 15,  
[ERR\\_DEVICE\\_WRITE\\_DATA\\_FAILED](#) = 16, [ERR\\_COMMAND\\_FAILED](#)  
= 17, [ERR\\_EEPROM](#) = 18, [ERR\\_JIDA\\_TEMP](#) = 19,  
[ERR\\_AVERAGE\\_CALC\\_STARTED](#) = 20, [ERR\\_NOT\\_RUNNING](#) = 21, [ER-  
R\\_I2C\\_EXPANDER\\_READ\\_FAILED](#) = 22, [ERR\\_I2C\\_EXPANDER\\_WRITE-  
\\_FAILED](#) = 23,  
[ERR\\_I2C\\_EXPANDER\\_INIT\\_FAILED](#) = 24, [ERR\\_NEWER\\_SS\\_VERSION-  
\\_REQUIRED](#) = 25, [ERR\\_NEWER\\_FPGA\\_VERSION\\_REQUIRED](#) = 26, [ER-  
R\\_NEWER\\_FRONT\\_VERSION\\_REQUIRED](#) = 27,  
[ERR\\_TELEMATICS\\_GPRS\\_NOT\\_AVAILABLE](#) = 28, [ERR\\_TELEMATICS-  
\\_WLAN\\_NOT\\_AVAILABLE](#) = 29, [ERR\\_TELEMATICS\\_BT\\_NOT\\_AVAIL-  
ABLE](#) = 30, [ERR\\_TELEMATICS\\_GPS\\_NOT\\_AVAILABLE](#) = 31,  
[ERR\\_MEM\\_ALLOC\\_FAIL](#) = 32, [ERR\\_JOIN\\_THREAD](#) = 33 }
- enum [DeInterlaceMode](#) { [DeInterlace\\_Even](#) = 0, [DeInterlace\\_Odd](#) = 1, [DeInterlace-\\_BOB](#) = 2 }

- enum VideoChannel { Analog\_Channel\_1 = 0, Analog\_Channel\_2 = 1, Analog\_Channel\_3 = 2, Analog\_Channel\_4 = 3 }
- enum videoStandard { STD\_M\_J\_NTSC = 0, STD\_B\_D\_G\_H\_I\_N\_PAL = 1, STD\_M\_PAL = 2, STD\_PAL = 3, STD\_NTSC = 4, STD\_SECAM = 5 }
- enum VideoRotation { RotNone = 0, Rot90, Rot180, Rot270 }
- enum CanFrameType { FrameStandard, FrameExtended, FrameStandardExtended }
- enum TriggerConf { Front\_Button\_Enabled = 1, OnOff\_Signal\_Enabled = 2, Both\_Button\_And\_Signal\_Enabled = 3 }
- enum PowerAction { NoAction = 0, ActionSuspend = 1, ActionShutDown = 2 }
- enum ButtonPowerTransitionStatus { BPTS\_No\_Change = 0, BPTS\_ShutDown = 1, BPTS\_Suspend = 2, BPTS\_Restart = 3, BPTS\_BtnPressed = 4, BPTS\_BtnPressedLong = 5, BPTS\_SignalOff = 6 }
- enum OCDStatus { OCD\_OK = 0, OCD\_OC = 1, OCD\_POWER\_OFF = 2 }
- enum JidaSensorType { TEMP\_CPU = 0, TEMP\_BOX = 1, TEMP\_ENV = 2, TEMP\_BOARD = 3, TEMP\_BACKPLANE = 4, TEMP\_CHIPSETS = 5, TEMP\_VIDEO = 6, TEMP\_OTHER = 7 }
- enum UpgradeAction { UPGRADE\_INIT, UPGRADE\_PREP\_COM, UPGRADE\_READING\_FILE, UPGRADE\_CONVERTING\_FILE, UPGRADE\_FLASHING, UPGRADE\_VERIFYING, UPGRADE\_COMPLETE, UPGRADE\_COMPLETE\_WITH\_ERRORS }
- enum CCAuxColor { RED = 0, GREEN, BLUE, CYAN, MAGENTA, YELLOW, UNDEFINED\_COLOR }
- enum RS4XXPort { RS4XXPort1 = 1, RS4XXPort2, RS4XXPort3, RS4XXPort4 }

## 7.10 IncludeFiles/CCPlatform.h File Reference

## 7.11 IncludeFiles/Config.h File Reference

### Namespaces

- namespace [CrossControl](#)

### Typedefs

- typedef void \* [CONFIGHANDLE](#)

## Functions

- EXTERN\_C CCAUXDLL\_API  
CONFIGHANDLE  
CCAUXDLL\_CALLING\_CONV [GetConfig \(\)](#)
- EXTERN\_C CCAUXDLL\_API void  
CCAUXDLL\_CALLING\_CONV [Config\\_release \(CONFIGHANDLE\)](#)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_getStartupTriggerConfig \(CONFIGHANDLE, TriggerConf \\*config\)](#)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_getShortButtonPressAction \(CONFIGHANDLE, PowerAction \\*action\)](#)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_getLongButtonPressAction \(CONFIGHANDLE, PowerAction \\*action\)](#)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_getOnOffSigAction \(CONFIGHANDLE, PowerAction \\*action\)](#)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_getFrontBtnTrigTime \(CONFIGHANDLE, unsigned short \\*triggertime\)](#)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_getExtOnOffSigTrigTime \(CONFIGHANDLE, unsigned long \\*triggertime\)](#)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_getSuspendMaxTime \(CONFIGHANDLE, unsigned short \\*maxTime\)](#)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_getCanStartupPowerConfig \(CONFIGHANDLE, CCStatus \\*status\)](#)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_getVideoStartupPowerConfig \(CONFIGHANDLE, unsigned char \\*config\)](#)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_getExtFanStartupPowerConfig \(CONFIGHANDLE, CCStatus \\*status\)](#)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_getStartupVoltageConfig \(CONFIGHANDLE, double \\*voltage\)](#)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_getHeatingTempLimit \(CONFIGHANDLE, signed short \\*temperature\)](#)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_getPowerOnStartup \(CONFIGHANDLE, CCStatus \\*status\)](#)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_setStartupTriggerConfig \(CONFIGHANDLE, TriggerConf conf\)](#)

- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_setShortButtonPressAction](#) (CONFIGHANDLE, PowerAction action)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_setLongButtonPressAction](#) (CONFIGHANDLE, PowerAction action)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_setOnOffSigAction](#) (CONFIGHANDLE, PowerAction action)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_setFrontBtnTrigTime](#) (CONFIGHANDLE, unsigned short triggertime)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_setExtOnOffSigTrigTime](#) (CONFIGHANDLE, unsigned long triggertime)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_setSuspendMaxTime](#) (CONFIGHANDLE, unsigned short maxTime)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_setCanStartupPowerConfig](#) (CONFIGHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_setVideoStartupPowerConfig](#) (CONFIGHANDLE, unsigned char config)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_setExtFanStartupPowerConfig](#) (CONFIGHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_setStartupVoltageConfig](#) (CONFIGHANDLE, double voltage)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_setHeatingTempLimit](#) (CONFIGHANDLE, signed short temperature)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_setPowerOnStartup](#) (CONFIGHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_setRS485Enabled](#) (CONFIGHANDLE, RS4XXPort port, bool enabled)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Config\\_getRS485Enabled](#) (CONFIGHANDLE, RS4XXPort port, bool \*enabled)

### Variables

- const unsigned char [Video1Conf](#) = (1 << 0)
- const unsigned char [Video2Conf](#) = (1 << 1)
- const unsigned char [Video3Conf](#) = (1 << 2)
- const unsigned char [Video4Conf](#) = (1 << 3)

## 7.12 IncludeFiles/Diagnostic.h File Reference

### Namespaces

- namespace [CrossControl](#)

### Typedefs

- typedef void \* [DIAGNOSTICHANDLE](#)

### Functions

- EXTERN\_C CCAUXDLL\_API  
DIAGNOSTICHANDLE  
CCAUXDLL\_CALLING\_CONV [GetDiagnostic](#) (void)
- EXTERN\_C CCAUXDLL\_API void  
CCAUXDLL\_CALLING\_CONV [Diagnostic\\_release](#) (DIAGNOSTICHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Diagnostic\\_getSSTemp](#) (DIAGNOSTICHANDLE, signed short \*temperature)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Diagnostic\\_getPCBTemp](#) (DIAGNOSTICHANDLE, signed short \*temperature)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Diagnostic\\_getPMTemp](#) (DIAGNOSTICHANDLE, unsigned char index, signed short \*temperature, JidaSensorType \*jst)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Diagnostic\\_getStartupReason](#) (DIAGNOSTICHANDLE, unsigned short \*reason)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Diagnostic\\_getShutDownReason](#) (DIAGNOSTICHANDLE, unsigned short \*reason)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Diagnostic\\_getHwErrorStatus](#) (DIAGNOSTICHANDLE, unsigned short \*errorCode)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Diagnostic\\_getTimer](#) (DIAGNOSTICHANDLE, TimerType \*times)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Diagnostic\\_getMinMaxTemp](#) (DIAGNOSTICHANDLE, signed short \*minTemp, signed short \*maxTemp)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Diagnostic\\_getPowerCycles](#) (DIAGNOSTICHANDLE, unsigned short \*powerCycles)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Diagnostic\\_clearHwErrorStatus](#) (DIAGNOSTICHANDLE)



## 7.13 IncludeFiles/DiagnosticCodes.h File Reference

### Namespaces

- namespace [CrossControl](#)

### Enumerations

- enum [startupReasonCodes](#) {  
  [startupReasonCodeUndefined](#) = 0x0000, [startupReasonCodeButtonPress](#) = 0x0055,  
  [startupReasonCodeExtCtrl](#) = 0x00AA, [startupReasonCodeMPRestart](#) = 0x00F0,  
  [startupReasonCodePowerOnStartup](#) = 0x000F }
- enum [shutDownReasonCodes](#) { [shutdownReasonCodeNoError](#) = 0x001F }
- enum [hwErrorStatusCodes](#) { [errCodeNoErr](#) = 0 }

### Functions

- EXTERN\_C CCAUXDLL\_API char  
  const \*CCAUXDLL\_CALLING\_CONV [GetHwErrorStatusStringA](#) (unsigned short errCode)
- EXTERN\_C CCAUXDLL\_API wchar\_t  
  const \*CCAUXDLL\_CALLING\_CONV [GetHwErrorStatusStringW](#) (unsigned short errCode)
- EXTERN\_C CCAUXDLL\_API char  
  const \*CCAUXDLL\_CALLING\_CONV [GetStartupReasonStringA](#) (unsigned short code)
- EXTERN\_C CCAUXDLL\_API wchar\_t  
  const \*CCAUXDLL\_CALLING\_CONV [GetStartupReasonStringW](#) (unsigned short code)

## 7.14 IncludeFiles/DigIO.h File Reference

### Namespaces

- namespace [CrossControl](#)

### Typedefs

- typedef void \* [DIGIOHANDLE](#)

### Functions

- EXTERN\_C CCAUXDLL\_API  
  DIGIOHANDLE  
  CCAUXDLL\_CALLING\_CONV [GetDigIO](#) (void)

- EXTERN\_C CCAUXDLL\_API void  
CCAUXDLL\_CALLING\_CONV [DigIO\\_release](#) (DIGIOHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [DigIO\\_getDigIO](#) (DIGIOHANDLE, unsigned char \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [DigIO\\_setDigIO](#) (DIGIOHANDLE, unsigned char state)

### Variables

- const unsigned char [DigitalIn\\_1](#) = (1 << 0)
- const unsigned char [DigitalIn\\_2](#) = (1 << 1)
- const unsigned char [DigitalIn\\_3](#) = (1 << 2)
- const unsigned char [DigitalIn\\_4](#) = (1 << 3)

## 7.15 IncludeFiles/FirmwareUpgrade.h File Reference

### Namespaces

- namespace [CrossControl](#)

### Typedefs

- typedef void \* [FIRMWAREUPGHANDLE](#)

### Functions

- EXTERN\_C CCAUXDLL\_API  
FIRMWAREUPGHANDLE  
CCAUXDLL\_CALLING\_CONV [GetFirmwareUpgrade](#) (void)
- EXTERN\_C CCAUXDLL\_API void  
CCAUXDLL\_CALLING\_CONV [FirmwareUpgrade\\_release](#) (FIRMWAREUPGHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [FirmwareUpgrade\\_startFpgaUpgrade](#) (FIRMWAREUPGHANDLE, const char \*filename, bool blocking)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [FirmwareUpgrade\\_startFpgaVerification](#) (FIRMWAREUPGHANDLE, const char \*filename, bool blocking)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [FirmwareUpgrade\\_startSSUpgrade](#) (FIRMWAREUPGHANDLE, const char \*filename, bool blocking)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [FirmwareUpgrade\\_startSSVerification](#) (FIRMWAREUPGHANDLE, const char \*filename, bool blocking)

- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [FirmwareUpgrade\\_startFrontUpgrade](#) (FIRMWAREUPGHANDLE, const char \*filename, bool blocking)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [FirmwareUpgrade\\_startFrontVerification](#) (FIRMWAREUPGHANDLE, const char \*filename, bool blocking)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [FirmwareUpgrade\\_getUpgradeStatus](#) (FIRMWAREUPGHANDLE, UpgradeStatus \*status, bool blocking)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [FirmwareUpgrade\\_shutDown](#) (FIRMWAREUPGHANDLE)

## 7.16 IncludeFiles/FrontLED.h File Reference

### Namespaces

- namespace [CrossControl](#)

### Typedefs

- typedef void \* [FRONTLEDHANDLE](#)

### Functions

- EXTERN\_C CCAUXDLL\_API  
FRONTLEDHANDLE  
CCAUXDLL\_CALLING\_CONV [GetFrontLED](#) (void)
- EXTERN\_C CCAUXDLL\_API void  
CCAUXDLL\_CALLING\_CONV [FrontLED\\_release](#) (FRONTLEDHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [FrontLED\\_getSignal](#) (FRONTLEDHANDLE, double \*frequency, unsigned char \*dutyCycle)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [FrontLED\\_getOnTime](#) (FRONTLEDHANDLE, unsigned char \*onTime)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [FrontLED\\_getOffTime](#) (FRONTLEDHANDLE, unsigned char \*offTime)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [FrontLED\\_getIdleTime](#) (FRONTLEDHANDLE, unsigned char \*idleTime)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [FrontLED\\_getNrOfPulses](#) (FRONTLEDHANDLE, unsigned char \*nrOfPulses)

- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [FrontLED\\_getColor](#) (FRONTLEDHANDLE, unsigned char \*red, unsigned char \*green, unsigned char \*blue)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [FrontLED\\_getStandardColor](#) (FRONTLEDHANDLE, CCAuxColor \*color)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [FrontLED\\_getEnabledDuringStartup](#) (FRONTLEDHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [FrontLED\\_setSignal](#) (FRONTLEDHANDLE, double frequency, unsigned char dutyCycle)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [FrontLED\\_setOnTime](#) (FRONTLEDHANDLE, unsigned char onTime)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [FrontLED\\_setOffTime](#) (FRONTLEDHANDLE, unsigned char offTime)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [FrontLED\\_setIdleTime](#) (FRONTLEDHANDLE, unsigned char idleTime)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [FrontLED\\_setNrOfPulses](#) (FRONTLEDHANDLE, unsigned char nrOfPulses)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [FrontLED\\_setColor](#) (FRONTLEDHANDLE, unsigned char red, unsigned char green, unsigned char blue)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [FrontLED\\_setStandardColor](#) (FRONTLEDHANDLE, CCAuxColor color)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [FrontLED\\_setOff](#) (FRONTLEDHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [FrontLED\\_setEnabledDuringStartup](#) (FRONTLEDHANDLE, CCStatus status)

## 7.17 IncludeFiles/Lightsensor.h File Reference

### Namespaces

- namespace [CrossControl](#)

### Typedefs

- typedef void \* [LIGHTSENSORHANDLE](#)

**Functions**

- EXTERN\_C CCAUXDLL\_API  
LIGHTSENSORHANDLE  
CCAUXDLL\_CALLING\_CONV [GetLightsensor](#) (void)
- EXTERN\_C CCAUXDLL\_API void  
CCAUXDLL\_CALLING\_CONV [Lightsensor\\_release](#) (LIGHTSENSORHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Lightsensor\\_getIlluminance](#) (LIGHTSENSORHANDLE, unsigned short \*value)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Lightsensor\\_getIlluminance2](#) (LIGHTSENSORHANDLE, unsigned short \*value, unsigned char \*ch0, unsigned char \*ch1)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Lightsensor\\_getAverageIlluminance](#) (LIGHTSENSORHANDLE, unsigned short \*value)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Lightsensor\\_startAverageCalc](#) (LIGHTSENSORHANDLE, unsigned long averageWndSize, unsigned long rejectWndSize, unsigned long rejectDeltaInLux, LightSensorSamplingMode mode)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Lightsensor\\_stopAverageCalc](#) (LIGHTSENSORHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Lightsensor\\_getOperatingRange](#) (LIGHTSENSORHANDLE, LightSensorOperationRange \*range)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Lightsensor\\_setOperatingRange](#) (LIGHTSENSORHANDLE, LightSensorOperationRange range)

**7.18 IncludeFiles/Power.h File Reference****Namespaces**

- namespace [CrossControl](#)

**Typedefs**

- typedef void \* [POWERHANDLE](#)

**Functions**

- EXTERN\_C CCAUXDLL\_API  
POWERHANDLE  
CCAUXDLL\_CALLING\_CONV [GetPower](#) (void)

- EXTERN\_C CCAUXDLL\_API void  
CCAUXDLL\_CALLING\_CONV [Power\\_release](#) (POWERHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Power\\_getBLPowerStatus](#) (POWERHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Power\\_getCanPowerStatus](#) (POWERHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Power\\_getVideoPowerStatus](#) (POWERHANDLE, unsigned char \*videoStatus)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Power\\_getExtFanPowerStatus](#) (POWERHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Power\\_getButtonPowerTransitionStatus](#) (POWERHANDLE, ButtonPowerTransitionStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Power\\_getVideoOCDStatus](#) (POWERHANDLE, OCDStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Power\\_getCanOCDStatus](#) (POWERHANDLE, OCDStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Power\\_setBLPowerStatus](#) (POWERHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Power\\_setCanPowerStatus](#) (POWERHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Power\\_setVideoPowerStatus](#) (POWERHANDLE, unsigned char status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Power\\_setExtFanPowerStatus](#) (POWERHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Power\\_ackPowerRequest](#) (POWERHANDLE)

## 7.19 IncludeFiles/PowerMgr.h File Reference

### Namespaces

- namespace [CrossControl](#)

## Typedefs

- typedef enum  
[CrossControl::PowerMgrConf](#) \_PowerMgrConf
- typedef enum  
[CrossControl::PowerMgrStatus](#) \_PowerMgrStatus
- typedef void \* [POWERMGRHANDLE](#)

## Enumerations

- enum [PowerMgrConf](#) { [Normal](#) = 0, [ApplicationControlled](#) = 1, [BatterySuspend](#) = 2 }
- enum [PowerMgrStatus](#) { [NoRequestsPending](#) = 0, [SuspendPending](#) = 1, [ShutdownPending](#) = 2 }

## Functions

- [EXTERN\\_C](#) [CCAUXDLL\\_API](#) [POWERMGRHANDLE](#) [CCAUXDLL\\_CALLING\\_CONV](#) [GetPowerMgr](#) (void)
- [EXTERN\\_C](#) [CCAUXDLL\\_API](#) void [CCAUXDLL\\_CALLING\\_CONV](#) [PowerMgr\\_release](#) ([POWERMGRHANDLE](#))
- [EXTERN\\_C](#) [CCAUXDLL\\_API](#) eErr [CCAUXDLL\\_CALLING\\_CONV](#) [PowerMgr\\_registerControlledSuspendOrShutdown](#) ([POWERMGRHANDLE](#), [PowerMgrConf](#) conf)
- [EXTERN\\_C](#) [CCAUXDLL\\_API](#) eErr [CCAUXDLL\\_CALLING\\_CONV](#) [PowerMgr\\_getConfiguration](#) ([POWERMGRHANDLE](#), [PowerMgrConf](#) \*conf)
- [EXTERN\\_C](#) [CCAUXDLL\\_API](#) eErr [CCAUXDLL\\_CALLING\\_CONV](#) [PowerMgr\\_getPowerMgrStatus](#) ([POWERMGRHANDLE](#), [PowerMgrStatus](#) \*status)
- [EXTERN\\_C](#) [CCAUXDLL\\_API](#) eErr [CCAUXDLL\\_CALLING\\_CONV](#) [PowerMgr\\_setAppReadyForSuspendOrShutdown](#) ([POWERMGRHANDLE](#))
- [EXTERN\\_C](#) [CCAUXDLL\\_API](#) eErr [CCAUXDLL\\_CALLING\\_CONV](#) [PowerMgr\\_hasResumed](#) ([POWERMGRHANDLE](#), bool \*resumed)

## 7.20 IncludeFiles/Releasenotes.dox File Reference

### 7.21 IncludeFiles/Smart.h File Reference

#### Namespaces

- namespace [CrossControl](#)

### Typedefs

- typedef void \* [SMARTHANDLE](#)

### Functions

- EXTERN\_C CCAUXDLL\_API SMARTHANDLE CCAUXDLL\_CALLING\_CONV [GetSmart](#) (void)
- EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV [Smart\\_release](#) (SMARTHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Smart\\_getRemainingLifeTime](#) (SMARTHANDLE, unsigned char \*lifetimepercent)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Smart\\_getDeviceSerial](#) (SMARTHANDLE, char \*buff, int len)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Smart\\_getInitialTime](#) (SMARTHANDLE, time\_t \*time)

## 7.22 IncludeFiles/Telematics.h File Reference

### Namespaces

- namespace [CrossControl](#)

### Typedefs

- typedef void \* [TELEMATICSHANDLE](#)

### Functions

- EXTERN\_C CCAUXDLL\_API TELEMATICSHANDLE CCAUXDLL\_CALLING\_CONV [GetTelematics](#) (void)
- EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV [Telematics\\_release](#) (TELEMATICSHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Telematics\\_getTelematicsAvailable](#) (TELEMATICSHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Telematics\\_getGPRSPowerStatus](#) (TELEMATICSHANDLE, CCStatus \*status)



- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Telematics\\_getGPRSStartupPowerStatus](#) (TELEMATICSHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Telematics\\_getWLANPowerStatus](#) (TELEMATICSHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Telematics\\_getWLANStartupPowerStatus](#) (TELEMATICSHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Telematics\\_getBTPowerStatus](#) (TELEMATICSHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Telematics\\_getBTStartupPowerStatus](#) (TELEMATICSHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Telematics\\_getGPSPowerStatus](#) (TELEMATICSHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Telematics\\_getGPSStartupPowerStatus](#) (TELEMATICSHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Telematics\\_getGPSAntennaStatus](#) (TELEMATICSHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Telematics\\_setGPRSPowerStatus](#) (TELEMATICSHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Telematics\\_setGPRSStartupPowerStatus](#) (TELEMATICSHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Telematics\\_setWLANPowerStatus](#) (TELEMATICSHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Telematics\\_setWLANStartupPowerStatus](#) (TELEMATICSHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Telematics\\_setBTPowerStatus](#) (TELEMATICSHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Telematics\\_setBTStartupPowerStatus](#) (TELEMATICSHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Telematics\\_setGPSPowerStatus](#) (TELEMATICSHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Telematics\\_setGPSStartupPowerStatus](#) (TELEMATICSHANDLE, CCStatus status)

## 7.23 IncludeFiles/TouchScreen.h File Reference

### Namespaces

- namespace [CrossControl](#)

### Typedefs

- typedef void \* [TOUCHSCREENHANDLE](#)

### Enumerations

- enum [TouchScreenModeSettings](#) { [MOUSE\\_NEXT\\_BOOT](#) = 0, [TOUCH\\_NEXT\\_BOOT](#) = 1, [MOUSE\\_NOW](#) = 2, [TOUCH\\_NOW](#) = 3 }
- enum [TSAdvancedSettingsParameter](#) { [TS\\_RIGHT\\_CLICK\\_TIME](#) = 0, [TS\\_LOW\\_LEVEL](#) = 1, [TS\\_UNTOUCHLEVEL](#) = 2, [TS\\_DEBOUNCE\\_TIME](#) = 3, [TS\\_DEBOUNCE\\_TIMEOUT\\_TIME](#) = 4, [TS\\_DOUBLECLICK\\_MAX\\_CLICK\\_TIME](#) = 5, [TS\\_DOUBLE\\_CLICK\\_TIME](#) = 6, [TS\\_MAX\\_RIGHTCLICK\\_DISTANCE](#) = 7, [TS\\_USE\\_DEJITTER](#) = 8, [TS\\_CALIBTATION\\_WIDTH](#) = 9, [TS\\_CALIBRATION\\_MEASUREMENTS](#) = 10, [TS\\_RESTORE\\_DEFAULT\\_SETTINGS](#) = 11, [TS\\_TCHAUTOCAL](#) = 12 }

### Functions

- [EXTERN\\_C CCAUXDLL\\_API TOUCHSCREENHANDLE CCAUXDLL\\_CALLING\\_CONV \[GetTouchScreen\]\(#\) \(void\)](#)
- [EXTERN\\_C CCAUXDLL\\_API void CCAUXDLL\\_CALLING\\_CONV \[TouchScreen\\\_release\]\(#\) \(TOUCHSCREENHANDLE\)](#)
- [EXTERN\\_C CCAUXDLL\\_API eErr CCAUXDLL\\_CALLING\\_CONV \[TouchScreen\\\_getMode\]\(#\) \(TOUCHSCREENHANDLE, TouchScreenModeSettings \\*config\)](#)
- [EXTERN\\_C CCAUXDLL\\_API eErr CCAUXDLL\\_CALLING\\_CONV \[TouchScreen\\\_getMouseRightClickTime\]\(#\) \(TOUCHSCREENHANDLE, unsigned short \\*time\)](#)
- [EXTERN\\_C CCAUXDLL\\_API eErr CCAUXDLL\\_CALLING\\_CONV \[TouchScreen\\\_setMode\]\(#\) \(TOUCHSCREENHANDLE, TouchScreenModeSettings config\)](#)
- [EXTERN\\_C CCAUXDLL\\_API eErr CCAUXDLL\\_CALLING\\_CONV \[TouchScreen\\\_setMouseRightClickTime\]\(#\) \(TOUCHSCREENHANDLE, unsigned short time\)](#)
- [EXTERN\\_C CCAUXDLL\\_API eErr CCAUXDLL\\_CALLING\\_CONV \[TouchScreen\\\_setAdvancedSetting\]\(#\) \(TOUCHSCREENHANDLE, TSAdvancedSettingsParameter param, unsigned short data\)](#)

- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [TouchScreen\\_getAdvancedSetting](#) (TOUCHSCREENHANDLE, TSAAdvancedSettingsParameter param, unsigned short \*data)

## 7.24 IncludeFiles/TouchScreenCalib.h File Reference

### Namespaces

- namespace [CrossControl](#)

### Typedefs

- typedef void \* [TOUCHSCREENCALIBHANDLE](#)

### Enumerations

- enum [CalibrationModeSettings](#) {  
MODE\_UNKNOWN = 0, MODE\_NORMAL = 1, MODE\_CALIBRATION\_5P = 2, MODE\_CALIBRATION\_9P = 3,  
MODE\_CALIBRATION\_13P = 4 }
- enum [CalibrationConfigParam](#) {  
CONFIG\_CALIBRATION\_WITH = 0, CONFIG\_CALIBRATION\_MEASUREMENTS = 1, CONFIG\_5P\_CALIBRATION\_POINT\_BORDER = 2, CONFIG\_13P\_CALIBRATION\_POINT\_BORDER = 3,  
CONFIG\_13P\_CALIBRATION\_TRANSITION\_MIN = 4, CONFIG\_13P\_CALIBRATION\_TRANSITION\_MAX = 5 }

### Functions

- EXTERN\_C CCAUXDLL\_API  
TOUCHSCREENCALIBHANDLE  
CCAUXDLL\_CALLING\_CONV [GetTouchScreenCalib](#) (void)
- EXTERN\_C CCAUXDLL\_API void  
CCAUXDLL\_CALLING\_CONV [TouchScreenCalib\\_release](#) (TOUCHSCREENCALIBHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [TouchScreenCalib\\_setMode](#) (TOUCHSCREENCALIBHANDLE, CalibrationModeSettings mode)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [TouchScreenCalib\\_getMode](#) (TOUCHSCREENCALIBHANDLE, CalibrationModeSettings \*mode)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [TouchScreenCalib\\_setCalibrationPoint](#) (TOUCHSCREENCALIBHANDLE, unsigned char pointNr)

- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [TouchScreenCalib\\_checkCalibrationPointFinished](#) (TOUCHSCREENCALIBHANDLE, bool \*finished, unsigned char pointNr)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [TouchScreenCalib\\_getConfigParam](#) (TOUCHSCREENCALIBHANDLE, CalibrationConfigParam param, unsigned short \*value)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [TouchScreenCalib\\_setConfigParam](#) (TOUCHSCREENCALIBHANDLE, CalibrationConfigParam param, unsigned short value)

## 7.25 IncludeFiles/Video.h File Reference

### Namespaces

- namespace [CrossControl](#)

### Typedefs

- typedef void \* [VIDEOHANDLE](#)

### Functions

- EXTERN\_C CCAUXDLL\_API  
VIDEOHANDLE  
CCAUXDLL\_CALLING\_CONV [GetVideo](#) (void)
- EXTERN\_C CCAUXDLL\_API void  
CCAUXDLL\_CALLING\_CONV [Video\\_release](#) (VIDEOHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_init](#) (VIDEOHANDLE, unsigned char deviceNr)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_showVideo](#) (VIDEOHANDLE, bool show)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_setDeInterlaceMode](#) (VIDEOHANDLE, DeInterlaceMode mode)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_getDeInterlaceMode](#) (VIDEOHANDLE, DeInterlaceMode \*mode)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_setMirroring](#) (VIDEOHANDLE, CC-Status mode)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_getMirroring](#) (VIDEOHANDLE, CC-Status \*mode)

- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_setRotation](#) (VIDEOHANDLE, VideoRotation rotation)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_getRotation](#) (VIDEOHANDLE, VideoRotation \*rotation)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_setActiveChannel](#) (VIDEOHANDLE, VideoChannel channel)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_getActiveChannel](#) (VIDEOHANDLE, VideoChannel \*channel)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_setColorKeys](#) (VIDEOHANDLE, unsigned char rKey, unsigned char gKey, unsigned char bKey)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_getColorKeys](#) (VIDEOHANDLE, unsigned char \*rKey, unsigned char \*gKey, unsigned char \*bKey)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_setVideoArea](#) (VIDEOHANDLE, unsigned short topLeftX, unsigned short topLeftY, unsigned short bottomRightX, unsigned short bottomRightY)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_getRawImage](#) (VIDEOHANDLE, unsigned short \*width, unsigned short \*height, float \*frameRate)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_getVideoArea](#) (VIDEOHANDLE, unsigned short \*topLeftX, unsigned short \*topLeftY, unsigned short \*bottomRightX, unsigned short \*bottomRightY)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_getVideoStandard](#) (VIDEOHANDLE, videoStandard \*standard)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_getStatus](#) (VIDEOHANDLE, unsigned char \*status)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_setScaling](#) (VIDEOHANDLE, float x, float y)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_getScaling](#) (VIDEOHANDLE, float \*x, float \*y)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_activateSnapshot](#) (VIDEOHANDLE, bool activate)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_takeSnapshot](#) (VIDEOHANDLE, const char \*path, bool bInterlaced)

- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_takeSnapshotRaw](#) (VIDEOHANDLE, char \*rawImgBuffer, unsigned long rawImgBuffSize, bool bInterlaced)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_takeSnapshotBmp](#) (VIDEOHANDLE, char \*\*bmpBuffer, unsigned long \*bmpBufSize, bool bInterlaced, bool bNTSC-Format)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_createBitmap](#) (VIDEOHANDLE, char \*\*bmpBuffer, unsigned long \*bmpBufSize, const char \*rawImgBuffer, unsigned long rawImgBufSize, bool bInterlaced, bool bNTSCFormat)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_freeBmpBuffer](#) (VIDEOHANDLE, char \*bmpBuffer)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_minimize](#) (VIDEOHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_restore](#) (VIDEOHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_setDecoderReg](#) (VIDEOHANDLE, unsigned char decoderRegister, unsigned char registerValue)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_getDecoderReg](#) (VIDEOHANDLE, unsigned char decoderRegister, unsigned char \*registerValue)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_setCropping](#) (VIDEOHANDLE, unsigned char top, unsigned char left, unsigned char bottom, unsigned char right)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_getCropping](#) (VIDEOHANDLE, unsigned char \*top, unsigned char \*left, unsigned char \*bottom, unsigned char \*right)
- EXTERN\_C CCAUXDLL\_API eErr  
CCAUXDLL\_CALLING\_CONV [Video\\_showFrame](#) (VIDEOHANDLE)

# Index

- [\\_PowerMgrConf](#)
      - [CrossControl, 29](#)
    - [\\_PowerMgrStatus](#)
      - [CrossControl, 29](#)
  - [ABOUTHANDLE](#)
    - [CrossControl, 29](#)
  - [ADCHANDLE](#)
    - [CrossControl, 29](#)
  - [AUXVERSIONHANDLE](#)
    - [CrossControl, 29](#)
  - [About\\_getAddOnHWversion](#)
    - [CrossControl, 41](#)
  - [About\\_getAddOnManufacturingDate](#)
    - [CrossControl, 41](#)
  - [About\\_getAddOnPCBArt](#)
    - [CrossControl, 41](#)
  - [About\\_getAddOnPCBSerial](#)
    - [CrossControl, 42](#)
  - [About\\_getDisplayResolution](#)
    - [CrossControl, 42](#)
  - [About\\_getFrontPcbRev](#)
    - [CrossControl, 43](#)
  - [About\\_getIOExpanderValue](#)
    - [CrossControl, 43](#)
  - [About\\_getIsAnybusMounted](#)
    - [CrossControl, 44](#)
  - [About\\_getIsBTMounted](#)
    - [CrossControl, 44](#)
  - [About\\_getIsDisplayAvailable](#)
    - [CrossControl, 45](#)
  - [About\\_getIsGPRSMounted](#)
    - [CrossControl, 45](#)
  - [About\\_getIsGPSMounted](#)
    - [CrossControl, 45](#)
  - [About\\_getIsIOExpanderMounted](#)
    - [CrossControl, 46](#)
  - [About\\_getIsTouchScreenAvailable](#)
    - [CrossControl, 46](#)
  - [About\\_getIsWLANMounted](#)
    - [CrossControl, 47](#)
- [About\\_getMainHWversion](#)
    - [CrossControl, 47](#)
  - [About\\_getMainManufacturingDate](#)
    - [CrossControl, 48](#)
  - [About\\_getMainPCBArt](#)
    - [CrossControl, 48](#)
  - [About\\_getMainPCBSerial](#)
    - [CrossControl, 49](#)
  - [About\\_getMainProdArtNr](#)
    - [CrossControl, 49](#)
  - [About\\_getMainProdRev](#)
    - [CrossControl, 50](#)
  - [About\\_getNrOfCANConnections](#)
    - [CrossControl, 50](#)
  - [About\\_getNrOfDigIOConnections](#)
    - [CrossControl, 51](#)
  - [About\\_getNrOfETHConnections](#)
    - [CrossControl, 51](#)
  - [About\\_getNrOfSerialConnections](#)
    - [CrossControl, 52](#)
  - [About\\_getNrOfUSBConnections](#)
    - [CrossControl, 52](#)
  - [About\\_getNrOfVideoConnections](#)
    - [CrossControl, 53](#)
  - [About\\_getUnitSerial](#)
    - [CrossControl, 53](#)
  - [About\\_hasOsBooted](#)
    - [CrossControl, 54](#)
  - [About\\_release](#)
    - [CrossControl, 54](#)
  - [Above80RunTime](#)
    - [CrossControl::TimerType, 181](#)
  - [ActionShutDown](#)
    - [CrossControl, 36](#)
  - [ActionSuspend](#)
    - [CrossControl, 36](#)
  - [Adc\\_getVoltage](#)
    - [CrossControl, 55](#)
  - [Adc\\_release](#)
    - [CrossControl, 55](#)
  - [Analog\\_Channel\\_1](#)

- CrossControl, 39
- Analog\_Channel\_2
  - CrossControl, 39
- Analog\_Channel\_3
  - CrossControl, 39
- Analog\_Channel\_4
  - CrossControl, 39
- ApplicationControlled
  - CrossControl, 36
- AuxVersion\_getCCAuxDrvVersion
  - CrossControl, 56
- AuxVersion\_getCCAuxVersion
  - CrossControl, 57
- AuxVersion\_getFPGAVersion
  - CrossControl, 57
- AuxVersion\_getFrontVersion
  - CrossControl, 58
- AuxVersion\_getOSVersion
  - CrossControl, 59
- AuxVersion\_getSSVersion
  - CrossControl, 60
- AuxVersion\_release
  - CrossControl, 60
- BLUE
  - CrossControl, 32
- BPTS\_BtnPressed
  - CrossControl, 31
- BPTS\_BtnPressedLong
  - CrossControl, 31
- BPTS\_No\_Change
  - CrossControl, 30
- BPTS\_Restart
  - CrossControl, 31
- BPTS\_ShutDown
  - CrossControl, 30
- BPTS\_SignalOff
  - CrossControl, 31
- BPTS\_Suspend
  - CrossControl, 31
- BACKLIGHTHANDLE
  - CrossControl, 30
- BATTERYHANDLE
  - CrossControl, 30
- BUZZERHANDLE
  - CrossControl, 30
- Backlight\_getAutomaticBLFilter
  - CrossControl, 61
- Backlight\_getAutomaticBLParams
  - CrossControl, 61
- Backlight\_getAutomaticBLStatus
  - CrossControl, 62
- Backlight\_getHWStatus
  - CrossControl, 62
- Backlight\_getIntensity
  - CrossControl, 63
- Backlight\_getLedDimming
  - CrossControl, 63
- Backlight\_getStatus
  - CrossControl, 64
- Backlight\_release
  - CrossControl, 64
- Backlight\_setAutomaticBLFilter
  - CrossControl, 65
- Backlight\_setAutomaticBLParams
  - CrossControl, 65
- Backlight\_setIntensity
  - CrossControl, 66
- Backlight\_setLedDimming
  - CrossControl, 66
- Backlight\_startAutomaticBL
  - CrossControl, 67
- Backlight\_stopAutomaticBL
  - CrossControl, 67
- BatterySuspend
  - CrossControl, 36
- Battery\_getBatteryChargingStatus
  - CrossControl, 67
- Battery\_getBatteryHWversion
  - CrossControl, 68
- Battery\_getBatterySerial
  - CrossControl, 69
- Battery\_getBatterySwVersion
  - CrossControl, 69
- Battery\_getBatteryTemp
  - CrossControl, 70
- Battery\_getBatteryVoltageStatus
  - CrossControl, 71
- Battery\_getHwErrorStatus
  - CrossControl, 71
- Battery\_getMinMaxTemp
  - CrossControl, 72
- Battery\_getPowerSource
  - CrossControl, 73
- Battery\_getTimer
  - CrossControl, 74
- Battery\_isBatteryPresent
  - CrossControl, 74
- Battery\_release
  - CrossControl, 75



- BatteryTimerType, 176
- blue
  - CrossControl::LedColorMixType, 179
- Both\_Button\_And\_Signal\_Enabled
  - CrossControl, 38
- build
  - CrossControl::version\_info, 183
- ButtonPowerTransitionStatus
  - CrossControl, 30
- Buzzer\_buzze
  - CrossControl, 75
- Buzzer\_getFrequency
  - CrossControl, 76
- Buzzer\_getTrigger
  - CrossControl, 76
- Buzzer\_getVolume
  - CrossControl, 77
- Buzzer\_release
  - CrossControl, 77
- Buzzer\_setFrequency
  - CrossControl, 77
- Buzzer\_setTrigger
  - CrossControl, 78
- Buzzer\_setVolume
  - CrossControl, 78
- BuzzerSetup, 177
- CONFIG\_13P\_CALIBRATION\_POINT-  
\_BORDER
  - CrossControl, 31
- CONFIG\_13P\_CALIBRATION\_TRANS-  
ITION\_MAX
  - CrossControl, 31
- CONFIG\_13P\_CALIBRATION\_TRANS-  
ITION\_MIN
  - CrossControl, 31
- CONFIG\_5P\_CALIBRATION\_POINT-  
BORDER
  - CrossControl, 31
- CONFIG\_CALIBRATION\_MEASUREM-  
ENTS
  - CrossControl, 31
- CONFIG\_CALIBRATION\_WITH
  - CrossControl, 31
- CYAN
  - CrossControl, 32
- CANSETTINGHANDLE
  - CrossControl, 30
- CCAuxColor
  - CrossControl, 32
- CCStatus
  - CrossControl, 32
- CONFIGHANDLE
  - CrossControl, 30
- CalibrationConfigParam
  - CrossControl, 31
- CalibrationModeSettings
  - CrossControl, 31
- CanFrameType
  - CrossControl, 31
- CanSetting\_getBaudrate
  - CrossControl, 79
- CanSetting\_getFrameType
  - CrossControl, 79
- CanSetting\_release
  - CrossControl, 80
- CanSetting\_setBaudrate
  - CrossControl, 80
- CanSetting\_setFrameType
  - CrossControl, 81
- ChargingStatus\_Charging
  - CrossControl, 32
- ChargingStatus\_FullyCharged
  - CrossControl, 32
- ChargingStatus\_NoCharge
  - CrossControl, 32
- ChargingStatus\_TempHigh
  - CrossControl, 33
- ChargingStatus\_TempLow
  - CrossControl, 32
- ChargingStatus\_Unknown
  - CrossControl, 33
- ChargingStatus
  - CrossControl, 32
- Config\_getCanStartupPowerConfig
  - CrossControl, 81
- Config\_getExtFanStartupPowerConfig
  - CrossControl, 81
- Config\_getExtOnOffSigTrigTime
  - CrossControl, 82
- Config\_getFrontBtnTrigTime
  - CrossControl, 82
- Config\_getHeatingTempLimit
  - CrossControl, 83
- Config\_getLongButtonPressAction
  - CrossControl, 83
- Config\_getOnOffSigAction
  - CrossControl, 83
- Config\_getPowerOnStartup
  - CrossControl, 84

- Config\_getRS485Enabled
  - CrossControl, 84
- Config\_getShortButtonPressAction
  - CrossControl, 85
- Config\_getStartupTriggerConfig
  - CrossControl, 85
- Config\_getStartupVoltageConfig
  - CrossControl, 86
- Config\_getSuspendMaxTime
  - CrossControl, 86
- Config\_getVideoStartupPowerConfig
  - CrossControl, 86
- Config\_release
  - CrossControl, 87
- Config\_setCanStartupPowerConfig
  - CrossControl, 87
- Config\_setExtFanStartupPowerConfig
  - CrossControl, 88
- Config\_setExtOnOffSigTrigTime
  - CrossControl, 88
- Config\_setFrontBtnTrigTime
  - CrossControl, 88
- Config\_setHeatingTempLimit
  - CrossControl, 89
- Config\_setLongButtonPressAction
  - CrossControl, 89
- Config\_setOnOffSigAction
  - CrossControl, 90
- Config\_setPowerOnStartup
  - CrossControl, 90
- Config\_setRS485Enabled
  - CrossControl, 90
- Config\_setShortButtonPressAction
  - CrossControl, 91
- Config\_setStartupTriggerConfig
  - CrossControl, 91
- Config\_setStartupVoltageConfig
  - CrossControl, 92
- Config\_setSuspendMaxTime
  - CrossControl, 92
- Config\_setVideoStartupPowerConfig
  - CrossControl, 93
- CrossControl
  - ActionShutDown, 36
  - ActionSuspend, 36
  - Analog\_Channel\_1, 39
  - Analog\_Channel\_2, 39
  - Analog\_Channel\_3, 39
  - Analog\_Channel\_4, 39
  - ApplicationControlled, 36
  - BLUE, 32
  - BPTS\_BtnPressed, 31
  - BPTS\_BtnPressedLong, 31
  - BPTS\_No\_Change, 30
  - BPTS\_Restart, 31
  - BPTS\_ShutDown, 30
  - BPTS\_SignalOff, 31
  - BPTS\_Suspend, 31
  - BatterySuspend, 36
  - Both\_Button\_And\_Signal\_Enabled, 38
  - CONFIG\_13P\_CALIBRATION\_PO-  
INT\_BORDER, 31
  - CONFIG\_13P\_CALIBRATION\_TR-  
ANSITION\_MAX, 31
  - CONFIG\_13P\_CALIBRATION\_TR-  
ANSITION\_MIN, 31
  - CONFIG\_5P\_CALIBRATION\_POI-  
NT\_BORDER, 31
  - CONFIG\_CALIBRATION\_MEASU-  
REMENTS, 31
  - CONFIG\_CALIBRATION\_WITH, 31
  - CYAN, 32
  - ChargingStatus\_Charging, 32
  - ChargingStatus\_FullyCharged, 32
  - ChargingStatus\_NoCharge, 32
  - ChargingStatus\_TempHigh, 33
  - ChargingStatus\_TempLow, 32
  - ChargingStatus\_Unknown, 33
  - DeInterlace\_BOB, 33
  - DeInterlace\_Even, 33
  - DeInterlace\_Odd, 33
  - Disabled, 32
  - ERR\_AVERAGE\_CALC\_START-  
D, 33
  - ERR\_BUFFER\_SIZE, 33
  - ERR\_CHECKSUM, 33
  - ERR\_CODE\_NOT\_EXIST, 33
  - ERR\_COMMAND\_FAILED, 33
  - ERR\_CREATE\_THREAD, 33
  - ERR\_DATATYPE\_MISMATCH, 33
  - ERR\_DEVICE\_READ\_DATA\_FA-  
ILED, 33
  - ERR\_DEVICE\_WRITE\_DATA\_FA-  
ILED, 33
  - ERR\_EEPROM, 33
  - ERR\_I2C\_EXPANDER\_INIT\_FAIL-  
ED, 34
  - ERR\_I2C\_EXPANDER\_READ\_FA-  
ILED, 34

- ERR\_I2C\_EXPANDER\_WRITE\_FAILED, 34
- ERR\_IN\_PROGRESS, 33
- ERR\_INIT\_FAILED, 33
- ERR\_INVALID\_DATA, 33
- ERR\_INVALID\_PARAMETER, 33
- ERR\_IOCTL\_FAILED, 33
- ERR\_JIDA\_TEMP, 33
- ERR\_JOIN\_THREAD, 34
- ERR\_MEM\_ALLOC\_FAIL, 34
- ERR\_NEWER\_FPGA\_VERSION\_REQUIRED, 34
- ERR\_NEWER\_FRONT\_VERSION\_REQUIRED, 34
- ERR\_NEWER\_SS\_VERSION\_REQUIRED, 34
- ERR\_NOT\_RUNNING, 33
- ERR\_NOT\_SUPPORTED, 33
- ERR\_OPEN\_FAILED, 33
- ERR\_SUCCESS, 33
- ERR\_TELEMATICS\_BT\_NOT\_AVAILABLE, 34
- ERR\_TELEMATICS\_GPRS\_NOT\_AVAILABLE, 34
- ERR\_TELEMATICS\_GPS\_NOT\_AVAILABLE, 34
- ERR\_TELEMATICS\_WLAN\_NOT\_AVAILABLE, 34
- ERR\_UNKNOWN\_FEATURE, 33
- ERR\_VERIFY\_FAILED, 33
- Enabled, 32
- errCodeNoErr, 34
- ErrorStatus\_ChargeFail, 34
- ErrorStatus\_Init, 34
- ErrorStatus\_NoError, 34
- ErrorStatus\_Overcurrent, 34
- ErrorStatus\_SecondaryTempSensor, 34
- ErrorStatus\_ThermistorTempSensor, 34
- FrameExtended, 32
- FrameStandard, 32
- FrameStandardExtended, 32
- Front\_Button\_Enabled, 38
- GREEN, 32
- MAGENTA, 32
- MODE\_CALIBRATION\_13P, 31
- MODE\_CALIBRATION\_5P, 31
- MODE\_CALIBRATION\_9P, 31
- MODE\_NORMAL, 31
- MODE\_UNKNOWN, 31
- MOUSE\_NEXT\_BOOT, 37
- MOUSE\_NOW, 37
- NoAction, 36
- NoRequestsPending, 36
- Normal, 36
- OCD\_OC, 35
- OCD\_OK, 35
- OCD\_POWER\_OFF, 35
- OnOff\_Signal\_Enabled, 38
- PowerSource\_Battery, 36
- PowerSource\_ExternalPower, 36
- RED, 32
- RS4XXPort1, 37
- RS4XXPort2, 37
- RS4XXPort3, 37
- RS4XXPort4, 37
- RangeExtended, 35
- RangeStandard, 35
- Rot180, 39
- Rot270, 39
- Rot90, 39
- RotNone, 39
- STD\_B\_D\_G\_H\_I\_N\_PAL, 40
- STD\_M\_J\_NTSC, 40
- STD\_M\_PAL, 40
- STD\_NTSC, 40
- STD\_PAL, 40
- STD\_SECAM, 40
- SamplingModeAuto, 35
- SamplingModeExtended, 35
- SamplingModeStandard, 35
- ShutdownPending, 36
- shutdownReasonCodeNoError, 37
- startupReasonCodeButtonPress, 37
- startupReasonCodeExtCtrl, 37
- startupReasonCodeMPRestart, 37
- startupReasonCodePowerOnStartup, 37
- startupReasonCodeUndefined, 37
- SuspendPending, 36
- TEMP\_BACKPLANE, 35
- TEMP\_BOARD, 35
- TEMP\_BOX, 34
- TEMP\_CHIPSETS, 35
- TEMP\_CPU, 34
- TEMP\_ENV, 35
- TEMP\_OTHER, 35
- TEMP\_VIDEO, 35
- TOUCH\_NEXT\_BOOT, 37
- TOUCH\_NOW, 37
- TS\_CALIBRATION\_MEASUREMENTS, 38

- TS\_CALIBTATION\_WIDTH, 38
- TS\_DEBOUNCE\_TIME, 38
- TS\_DEBOUNCE\_TIMEOUT\_TIME, 38
- TS\_DOUBLE\_CLICK\_TIME, 38
- TS\_DOUBLECLICK\_MAX\_CLICK\_TIME, 38
- TS\_LOW\_LEVEL, 38
- TS\_MAX\_RIGHTCLICK\_DISTANCE, 38
- TS\_RESTORE\_DEFAULT\_SETTINGS, 38
- TS\_RIGHT\_CLICK\_TIME, 38
- TS\_TCHAUTOCAL, 39
- TS\_UNTOUCHLEVEL, 38
- TS\_USE\_DEJITTER, 38
- UNDEFINED\_COLOR, 32
- UPGRADE\_COMPLETE, 39
- UPGRADE\_COMPLETE\_WITH\_ERRORS, 39
- UPGRADE\_CONVERTING\_FILE, 39
- UPGRADE\_FLASHING, 39
- UPGRADE\_INIT, 39
- UPGRADE\_PREP\_COM, 39
- UPGRADE\_READING\_FILE, 39
- UPGRADE\_VERIFYING, 39
- VOLTAGE\_0V9, 40
- VOLTAGE\_12V, 40
- VOLTAGE\_12VID, 40
- VOLTAGE\_1V0, 40
- VOLTAGE\_1V05, 40
- VOLTAGE\_1V1, 40
- VOLTAGE\_1V2, 40
- VOLTAGE\_1V3\_PER, 40
- VOLTAGE\_1V3\_VDDA, 40
- VOLTAGE\_1V5, 40
- VOLTAGE\_1V8, 40
- VOLTAGE\_1V9, 40
- VOLTAGE\_24V, 40
- VOLTAGE\_24V\_BACKUP, 40
- VOLTAGE\_24VIN, 40
- VOLTAGE\_2V5, 40
- VOLTAGE\_3V3, 40
- VOLTAGE\_5V, 40
- VOLTAGE\_5VSTB, 40
- VOLTAGE\_VREF\_INT, 40
- VOLTAGE\_VTFT, 40
- YELLOW, 32
- CrossControl, 9
  - \_PowerMgrConf, 29
  - \_PowerMgrStatus, 29
- ABOUTHANDLE, 29
- ADCHANDLE, 29
- AUXVERSIONHANDLE, 29
- About\_getAddOnHWversion, 41
- About\_getAddOnManufacturingDate, 41
- About\_getAddOnPCBArt, 41
- About\_getAddOnPCBSerial, 42
- About\_getDisplayResolution, 42
- About\_getFrontPcbRev, 43
- About\_getIOExpanderValue, 43
- About\_getIsAnybusMounted, 44
- About\_getIsBTMounted, 44
- About\_getIsDisplayAvailable, 45
- About\_getIsGPRSMounted, 45
- About\_getIsGPSMounted, 45
- About\_getIsIOExpanderMounted, 46
- About\_getIsTouchScreenAvailable, 46
- About\_getIsWLANMounted, 47
- About\_getMainHWversion, 47
- About\_getMainManufacturingDate, 48
- About\_getMainPCBArt, 48
- About\_getMainPCBSerial, 49
- About\_getMainProdArtNr, 49
- About\_getMainProdRev, 50
- About\_getNrOfCANConnections, 50
- About\_getNrOfDigIOConnections, 51
- About\_getNrOfETHConnections, 51
- About\_getNrOfSerialConnections, 52
- About\_getNrOfUSBConnections, 52
- About\_getNrOfVideoConnections, 53
- About\_getUnitSerial, 53
- About\_hasOsBooted, 54
- About\_release, 54
- Adc\_getVoltage, 55
- Adc\_release, 55
- AuxVersion\_getCCAuxDrvVersion, 56
- AuxVersion\_getCCAuxVersion, 57
- AuxVersion\_getFPGAVersion, 57
- AuxVersion\_getFrontVersion, 58
- AuxVersion\_getOSVersion, 59
- AuxVersion\_getSSVersion, 60
- AuxVersion\_release, 60
- BACKLIGHTHANDLE, 30
- BATTERYHANDLE, 30
- BUZZERHANDLE, 30
- Backlight\_getAutomaticBLFilter, 61
- Backlight\_getAutomaticBLParams, 61
- Backlight\_getAutomaticBLStatus, 62

- Backlight\_getHWStatus, 62
- Backlight\_getIntensity, 63
- Backlight\_getLedDimming, 63
- Backlight\_getStatus, 64
- Backlight\_release, 64
- Backlight\_setAutomaticBLFilter, 65
- Backlight\_setAutomaticBLParams, 65
- Backlight\_setIntensity, 66
- Backlight\_setLedDimming, 66
- Backlight\_startAutomaticBL, 67
- Backlight\_stopAutomaticBL, 67
- Battery\_getBatteryChargingStatus, 67
- Battery\_getBatteryHWversion, 68
- Battery\_getBatterySerial, 69
- Battery\_getBatterySwVersion, 69
- Battery\_getBatteryTemp, 70
- Battery\_getBatteryVoltageStatus, 71
- Battery\_getHwErrorStatus, 71
- Battery\_getMinMaxTemp, 72
- Battery\_getPowerSource, 73
- Battery\_getTimer, 74
- Battery\_isBatteryPresent, 74
- Battery\_release, 75
- ButtonPowerTransitionStatus, 30
- Buzzer\_buzze, 75
- Buzzer\_getFrequency, 76
- Buzzer\_getTrigger, 76
- Buzzer\_getVolume, 77
- Buzzer\_release, 77
- Buzzer\_setFrequency, 77
- Buzzer\_setTrigger, 78
- Buzzer\_setVolume, 78
- CANSETTINGHANDLE, 30
- CCAuxColor, 32
- CCStatus, 32
- CONFIGHANDLE, 30
- CalibrationConfigParam, 31
- CalibrationModeSettings, 31
- CanFrameType, 31
- CanSetting\_getBaudrate, 79
- CanSetting\_getFrameType, 79
- CanSetting\_release, 80
- CanSetting\_setBaudrate, 80
- CanSetting\_setFrameType, 81
- ChargingStatus, 32
- Config\_getCanStartupPowerConfig, 81
- Config\_getExtFanStartupPowerConfig, 81
- Config\_getExtOnOffSigTrigTime, 82
- Config\_getFrontBtnTrigTime, 82
- Config\_getHeatingTempLimit, 83
- Config\_getLongButtonPressAction, 83
- Config\_getOnOffSigAction, 83
- Config\_getPowerOnStartup, 84
- Config\_getRS485Enabled, 84
- Config\_getShortButtonPressAction, 85
- Config\_getStartupTriggerConfig, 85
- Config\_getStartupVoltageConfig, 86
- Config\_getSuspendMaxTime, 86
- Config\_getVideoStartupPowerConfig, 86
- Config\_release, 87
- Config\_setCanStartupPowerConfig, 87
- Config\_setExtFanStartupPowerConfig, 88
- Config\_setExtOnOffSigTrigTime, 88
- Config\_setFrontBtnTrigTime, 88
- Config\_setHeatingTempLimit, 89
- Config\_setLongButtonPressAction, 89
- Config\_setOnOffSigAction, 90
- Config\_setPowerOnStartup, 90
- Config\_setRS485Enabled, 90
- Config\_setShortButtonPressAction, 91
- Config\_setStartupTriggerConfig, 91
- Config\_setStartupVoltageConfig, 92
- Config\_setSuspendMaxTime, 92
- Config\_setVideoStartupPowerConfig, 93
- DIAGNOSTICHANDLE, 30
- DIGIOHANDLE, 30
- DeInterlaceMode, 33
- Diagnostic\_clearHwErrorStatus, 93
- Diagnostic\_getHwErrorStatus, 93
- Diagnostic\_getMinMaxTemp, 94
- Diagnostic\_getPCBTemp, 94
- Diagnostic\_getPMTemp, 94
- Diagnostic\_getPowerCycles, 95
- Diagnostic\_getSSTemp, 96
- Diagnostic\_getShutDownReason, 95
- Diagnostic\_getStartupReason, 96
- Diagnostic\_getTimer, 97
- Diagnostic\_release, 97
- DigIO\_getDigIO, 97
- DigIO\_release, 98
- DigIO\_setDigIO, 98
- DigitalIn\_1, 174
- DigitalIn\_2, 174
- DigitalIn\_3, 174
- DigitalIn\_4, 174
- eErr, 33

- ErrorStatus, 34
- FIRMWAREUPGHANDLE, 30
- FRONTLEDHANDLE, 30
- FirmwareUpgrade\_getUpgradeStatus, 99
- FirmwareUpgrade\_release, 99
- FirmwareUpgrade\_shutDown, 100
- FirmwareUpgrade\_startFpgaUpgrade, 100
- FirmwareUpgrade\_startFpgaVerification, 101
- FirmwareUpgrade\_startFrontUpgrade, 102
- FirmwareUpgrade\_startFrontVerification, 103
- FirmwareUpgrade\_startSSUpgrade, 104
- FirmwareUpgrade\_startSSVerification, 105
- FrontLED\_getColor, 106
- FrontLED\_getEnabledDuringStartup, 107
- FrontLED\_getIdleTime, 107
- FrontLED\_getNrOfPulses, 107
- FrontLED\_getOffTime, 108
- FrontLED\_getOnTime, 108
- FrontLED\_getSignal, 109
- FrontLED\_getStandardColor, 109
- FrontLED\_release, 109
- FrontLED\_setColor, 110
- FrontLED\_setEnabledDuringStartup, 110
- FrontLED\_setIdleTime, 111
- FrontLED\_setNrOfPulses, 111
- FrontLED\_setOff, 112
- FrontLED\_setOffTime, 112
- FrontLED\_setOnTime, 112
- FrontLED\_setSignal, 113
- FrontLED\_setStandardColor, 113
- GetAbout, 114
- GetAdc, 114
- GetAuxVersion, 115
- GetBacklight, 115
- GetBattery, 116
- GetBuzzer, 116
- GetCanSetting, 117
- GetConfig, 117
- GetDiagnostic, 118
- GetDigIO, 118
- GetErrorStringA, 118
- GetErrorStringW, 119
- GetFirmwareUpgrade, 119
- GetFrontLED, 119
- GetHwErrorStatusStringA, 120
- GetHwErrorStatusStringW, 120
- GetLightsensor, 120
- GetPower, 121
- GetPowerMgr, 121
- GetSmart, 122
- GetStartupReasonStringA, 123
- GetStartupReasonStringW, 123
- GetTelematics, 123
- GetTouchScreen, 124
- GetTouchScreenCalib, 124
- GetVideo, 124
- hwErrorStatusCodes, 34
- JidaSensorType, 34
- LIGHTSENSORHANDLE, 30
- LightSensorOperationRange, 35
- LightSensorSamplingMode, 35
- Lightsensor\_getAverageIlluminance, 125
- Lightsensor\_getIlluminance, 125
- Lightsensor\_getIlluminance2, 126
- Lightsensor\_getOperatingRange, 126
- Lightsensor\_release, 126
- Lightsensor\_setOperatingRange, 127
- Lightsensor\_startAverageCalc, 127
- Lightsensor\_stopAverageCalc, 128
- OCDStatus, 35
- POWERHANDLE, 30
- POWERMGRHANDLE, 30
- Power\_ackPowerRequest, 128
- Power\_getBLPowerStatus, 129
- Power\_getButtonPowerTransitionStatus, 129
- Power\_getCanOCDStatus, 130
- Power\_getCanPowerStatus, 130
- Power\_getExtFanPowerStatus, 131
- Power\_getVideoOCDStatus, 131
- Power\_getVideoPowerStatus, 132
- Power\_release, 132
- Power\_setBLPowerStatus, 133
- Power\_setCanPowerStatus, 133
- Power\_setExtFanPowerStatus, 134
- Power\_setVideoPowerStatus, 134
- PowerAction, 35
- PowerMgr\_getConfiguration, 134
- PowerMgr\_getPowerMgrStatus, 135
- PowerMgr\_hasResumed, 137

- PowerMgr\_registerControlledSuspend-OrShutDown, 139
- PowerMgr\_release, 140
- PowerMgr\_setAppReadyForSuspend-OrShutdown, 141
- PowerMgrConf, 36
- PowerMgrStatus, 36
- PowerSource, 36
- RS4XXPort, 36
- SMARTHANDLE, 30
- shutDownReasonCodes, 37
- Smart\_getDeviceSerial, 143
- Smart\_getInitialTime, 144
- Smart\_getRemainingLifeTime, 144
- Smart\_release, 145
- startupReasonCodes, 37
- TELEMATICSHANDLE, 30
- TOUCHSCREENHANDLE, 30
- TSAdvancedSettingsParameter, 38
- Telematics\_getBTPowerStatus, 145
- Telematics\_getBTStartUpPowerStatus, 146
- Telematics\_getGPRSPowerStatus, 147
- Telematics\_getGPRSStartUpPowerStatus, 147
- Telematics\_getGPSAntennaStatus, 148
- Telematics\_getGPSPowerStatus, 148
- Telematics\_getGPSStartUpPowerStatus, 149
- Telematics\_getTelematicsAvailable, 150
- Telematics\_getWLANPowerStatus, 150
- Telematics\_getWLANStartUpPowerStatus, 151
- Telematics\_release, 152
- Telematics\_setBTPowerStatus, 152
- Telematics\_setBTStartUpPowerStatus, 152
- Telematics\_setGPRSPowerStatus, 153
- Telematics\_setGPRSStartUpPowerStatus, 153
- Telematics\_setGPSPowerStatus, 153
- Telematics\_setGPSStartUpPowerStatus, 154
- Telematics\_setWLANPowerStatus, 154
- Telematics\_setWLANStartUpPowerStatus, 155
- TouchScreen\_getAdvancedSetting, 155
- TouchScreen\_getMode, 156
- TouchScreen\_getMouseRightClickTime, 156
- TouchScreen\_release, 157
- TouchScreen\_setAdvancedSetting, 157
- TouchScreen\_setMode, 158
- TouchScreen\_setMouseRightClickTime, 158
- TouchScreenCalib\_checkCalibration-PointFinished, 159
- TouchScreenCalib\_getConfigParam, 159
- TouchScreenCalib\_getMode, 159
- TouchScreenCalib\_release, 160
- TouchScreenCalib\_setCalibrationPoint, 160
- TouchScreenCalib\_setConfigParam, 160
- TouchScreenCalib\_setMode, 161
- TouchScreenModeSettings, 37
- TriggerConf, 37
- UpgradeAction, 39
- VIDEOHANDLE, 30
- VersionType, 30
- Video1Conf, 174
- Video2Conf, 174
- Video3Conf, 175
- Video4Conf, 175
- Video\_activateSnapshot, 161
- Video\_createBitmap, 161
- Video\_freeBmpBuffer, 162
- Video\_getActiveChannel, 162
- Video\_getColorKeys, 163
- Video\_getCropping, 163
- Video\_getDeInterlaceMode, 164
- Video\_getDecoderReg, 164
- Video\_getMirroring, 164
- Video\_getRawImage, 165
- Video\_getRotation, 165
- Video\_getScaling, 165
- Video\_getStatus, 166
- Video\_getVideoArea, 166
- Video\_getVideoStandard, 167
- Video\_init, 167
- Video\_minimize, 167
- Video\_release, 168
- Video\_restore, 168
- Video\_setActiveChannel, 168
- Video\_setColorKeys, 169
- Video\_setCropping, 169
- Video\_setDeInterlaceMode, 170
- Video\_setDecoderReg, 169
- Video\_setMirroring, 170
- Video\_setRotation, 171
- Video\_setScaling, 171

- Video\_setVideoArea, 171
- Video\_showFrame, 172
- Video\_showVideo, 172
- Video\_takeSnapshot, 173
- Video\_takeSnapshotBmp, 173
- Video\_takeSnapshotRaw, 174
- VideoChannel, 39
- VideoRotation, 39
- videoStandard, 39
- VoltageEnum, 40
- CrossControl::BatteryTimerType
  - RunTime\_0\_40, 176
  - RunTime\_40\_60, 176
  - RunTime\_60\_70, 176
  - RunTime\_70\_80, 176
  - RunTime\_Above80, 177
  - RunTime\_m20, 177
  - RunTime\_m20\_0, 177
  - TotRunTimeBattery, 177
  - TotRunTimeMain, 177
- CrossControl::BuzzerSetup
  - frequency, 177
  - volume, 177
- CrossControl::FpgaLedTimingType
  - idleTime, 178
  - ledNbr, 178
  - nrOfPulses, 178
  - offTime, 178
  - onTime, 178
- CrossControl::LedColorMixType
  - blue, 179
  - green, 179
  - red, 179
- CrossControl::LedTimingType
  - idleTime, 179
  - nrOfPulses, 179
  - offTime, 180
  - onTime, 180
- CrossControl::TimerType
  - Above80RunTime, 181
  - RunTime40\_60, 181
  - RunTime60\_70, 181
  - RunTime70\_80, 181
  - TotHeatTime, 181
  - TotRunTime, 181
  - TotSuspTime, 181
- CrossControl::UpgradeStatus
  - currentAction, 182
  - errorCode, 182
  - percent, 182
- CrossControl::received\_video
  - received\_framerate, 180
  - received\_height, 180
  - received\_width, 180
- CrossControl::version\_info
  - build, 183
  - major, 183
  - minor, 183
  - release, 183
- CrossControl::video\_dec\_command
  - decoder\_register, 183
  - register\_value, 183
- currentAction
  - CrossControl::UpgradeStatus, 182
- DIAGNOSTICHANDLE
  - CrossControl, 30
- DIGIOHANDLE
  - CrossControl, 30
- DeInterlace\_BOB
  - CrossControl, 33
- DeInterlace\_Even
  - CrossControl, 33
- DeInterlace\_Odd
  - CrossControl, 33
- DeInterlaceMode
  - CrossControl, 33
- decoder\_register
  - CrossControl::video\_dec\_command, 183
- Diagnostic\_clearHwErrorStatus
  - CrossControl, 93
- Diagnostic\_getHwErrorStatus
  - CrossControl, 93
- Diagnostic\_getMinMaxTemp
  - CrossControl, 94
- Diagnostic\_getPCBTemp
  - CrossControl, 94
- Diagnostic\_getPMTemp
  - CrossControl, 94
- Diagnostic\_getPowerCycles
  - CrossControl, 95
- Diagnostic\_getSSTemp
  - CrossControl, 96
- Diagnostic\_getShutDownReason
  - CrossControl, 95
- Diagnostic\_getStartupReason
  - CrossControl, 96
- Diagnostic\_getTimer
  - CrossControl, 97
- Diagnostic\_release



- CrossControl, 97
- DigIO\_getDigIO
  - CrossControl, 97
- DigIO\_release
  - CrossControl, 98
- DigIO\_setDigIO
  - CrossControl, 98
- DigitalIn\_1
  - CrossControl, 174
- DigitalIn\_2
  - CrossControl, 174
- DigitalIn\_3
  - CrossControl, 174
- DigitalIn\_4
  - CrossControl, 174
- Disabled
  - CrossControl, 32
- ERR\_AVERAGE\_CALC\_STARTED
  - CrossControl, 33
- ERR\_BUFFER\_SIZE
  - CrossControl, 33
- ERR\_CHECKSUM
  - CrossControl, 33
- ERR\_CODE\_NOT\_EXIST
  - CrossControl, 33
- ERR\_COMMAND\_FAILED
  - CrossControl, 33
- ERR\_CREATE\_THREAD
  - CrossControl, 33
- ERR\_DATATYPE\_MISMATCH
  - CrossControl, 33
- ERR\_DEVICE\_READ\_DATA\_FAILED
  - CrossControl, 33
- ERR\_DEVICE\_WRITE\_DATA\_FAILED
  - CrossControl, 33
- ERR\_EEPROM
  - CrossControl, 33
- ERR\_I2C\_EXPANDER\_INIT\_FAILED
  - CrossControl, 34
- ERR\_I2C\_EXPANDER\_READ\_FAILED
  - CrossControl, 34
- ERR\_I2C\_EXPANDER\_WRITE\_FAILED
  - CrossControl, 34
- ERR\_IN\_PROGRESS
  - CrossControl, 33
- ERR\_INIT\_FAILED
  - CrossControl, 33
- ERR\_INVALID\_DATA
  - CrossControl, 33
- ERR\_INVALID\_PARAMETER
  - CrossControl, 33
- ERR\_IOCTL\_FAILED
  - CrossControl, 33
- ERR\_JIDA\_TEMP
  - CrossControl, 33
- ERR\_JOIN\_THREAD
  - CrossControl, 34
- ERR\_MEM\_ALLOC\_FAIL
  - CrossControl, 34
- ERR\_NEWER\_FPGA\_VERSION\_REQUIRED
  - CrossControl, 34
- ERR\_NEWER\_FRONT\_VERSION\_REQUIRED
  - CrossControl, 34
- ERR\_NEWER\_SS\_VERSION\_REQUIRED
  - CrossControl, 34
- ERR\_NOT\_RUNNING
  - CrossControl, 33
- ERR\_NOT\_SUPPORTED
  - CrossControl, 33
- ERR\_OPEN\_FAILED
  - CrossControl, 33
- ERR\_SUCCESS
  - CrossControl, 33
- ERR\_TELEMATICS\_BT\_NOT\_AVAILABLE
  - CrossControl, 34
- ERR\_TELEMATICS\_GPRS\_NOT\_AVAILABLE
  - CrossControl, 34
- ERR\_TELEMATICS\_GPS\_NOT\_AVAILABLE
  - CrossControl, 34
- ERR\_TELEMATICS\_WLAN\_NOT\_AVAILABLE
  - CrossControl, 34
- ERR\_UNKNOWN\_FEATURE
  - CrossControl, 33
- ERR\_VERIFY\_FAILED
  - CrossControl, 33
- eErr
  - CrossControl, 33
- Enabled
  - CrossControl, 32
- errCodeNoErr
  - CrossControl, 34

- ErrorStatus\_ChargeFail
  - CrossControl, 34
- ErrorStatus\_Init
  - CrossControl, 34
- ErrorStatus\_NoError
  - CrossControl, 34
- ErrorStatus\_Overcurrent
  - CrossControl, 34
- ErrorStatus\_SecondaryTempSensor
  - CrossControl, 34
- ErrorStatus\_ThermistorTempSensor
  - CrossControl, 34
- errorCode
  - CrossControl::UpgradeStatus, 182
- ErrorStatus
  - CrossControl, 34
- FIRMWAREUPGHANDLE
  - CrossControl, 30
- FRONTLEDHANDLE
  - CrossControl, 30
- FirmwareUpgrade\_getUpgradeStatus
  - CrossControl, 99
- FirmwareUpgrade\_release
  - CrossControl, 99
- FirmwareUpgrade\_shutDown
  - CrossControl, 100
- FirmwareUpgrade\_startFpgaUpgrade
  - CrossControl, 100
- FirmwareUpgrade\_startFpgaVerification
  - CrossControl, 101
- FirmwareUpgrade\_startFrontUpgrade
  - CrossControl, 102
- FirmwareUpgrade\_startFrontVerification
  - CrossControl, 103
- FirmwareUpgrade\_startSSUpgrade
  - CrossControl, 104
- FirmwareUpgrade\_startSSVerification
  - CrossControl, 105
- FpgaLedTimingType, 178
- FrameExtended
  - CrossControl, 32
- FrameStandard
  - CrossControl, 32
- FrameStandardExtended
  - CrossControl, 32
- frequency
  - CrossControl::BuzzerSetup, 177
- Front\_Button\_Enabled
  - CrossControl, 38
- FrontLED\_getColor
  - CrossControl, 106
- FrontLED\_getEnabledDuringStartup
  - CrossControl, 107
- FrontLED\_getIdleTime
  - CrossControl, 107
- FrontLED\_getNrOfPulses
  - CrossControl, 107
- FrontLED\_getOffTime
  - CrossControl, 108
- FrontLED\_getOnTime
  - CrossControl, 108
- FrontLED\_getSignal
  - CrossControl, 109
- FrontLED\_getStandardColor
  - CrossControl, 109
- FrontLED\_release
  - CrossControl, 109
- FrontLED\_setColor
  - CrossControl, 110
- FrontLED\_setEnabledDuringStartup
  - CrossControl, 110
- FrontLED\_setIdleTime
  - CrossControl, 111
- FrontLED\_setNrOfPulses
  - CrossControl, 111
- FrontLED\_setOff
  - CrossControl, 112
- FrontLED\_setOffTime
  - CrossControl, 112
- FrontLED\_setOnTime
  - CrossControl, 112
- FrontLED\_setSignal
  - CrossControl, 113
- FrontLED\_setStandardColor
  - CrossControl, 113
- GREEN
  - CrossControl, 32
- GetAbout
  - CrossControl, 114
- GetAdc
  - CrossControl, 114
- GetAuxVersion
  - CrossControl, 115
- GetBacklight
  - CrossControl, 115
- GetBattery
  - CrossControl, 116
- GetBuzzer

- CrossControl, 116
- GetCanSetting
  - CrossControl, 117
- GetConfig
  - CrossControl, 117
- GetDiagnostic
  - CrossControl, 118
- GetDigIO
  - CrossControl, 118
- GetErrorStringA
  - CrossControl, 118
- GetErrorStringW
  - CrossControl, 119
- GetFirmwareUpgrade
  - CrossControl, 119
- GetFrontLED
  - CrossControl, 119
- GetHwErrorStatusStringA
  - CrossControl, 120
- GetHwErrorStatusStringW
  - CrossControl, 120
- GetLightsensor
  - CrossControl, 120
- GetPower
  - CrossControl, 121
- GetPowerMgr
  - CrossControl, 121
- GetSmart
  - CrossControl, 122
- GetStartupReasonStringA
  - CrossControl, 123
- GetStartupReasonStringW
  - CrossControl, 123
- GetTelematics
  - CrossControl, 123
- GetTouchScreen
  - CrossControl, 124
- GetTouchScreenCalib
  - CrossControl, 124
- GetVideo
  - CrossControl, 124
- green
  - CrossControl::LedColorMixType, 179
- hwErrorStatusCodes
  - CrossControl, 34
- idleTime
  - CrossControl::FpgaLedTimingType, 178
  - CrossControl::LedTimingType, 179
- IncludeFiles/About.h, 184
- IncludeFiles/Adc.h, 186
- IncludeFiles/AuxVersion.h, 187
- IncludeFiles/Backlight.h, 188
- IncludeFiles/Battery.h, 189
- IncludeFiles/Buzzer.h, 190
- IncludeFiles/CCAuxErrors.h, 192
- IncludeFiles/CCAuxTypes.h, 192
- IncludeFiles/CCPlatform.h, 194
- IncludeFiles/CanSetting.h, 191
- IncludeFiles/Config.h, 194
- IncludeFiles/Diagnostic.h, 197
- IncludeFiles/DiagnosticCodes.h, 198
- IncludeFiles/DigIO.h, 198
- IncludeFiles/FirmwareUpgrade.h, 199
- IncludeFiles/FrontLED.h, 200
- IncludeFiles/Lightsensor.h, 201
- IncludeFiles/Power.h, 202
- IncludeFiles/PowerMgr.h, 203
- IncludeFiles/Releasenotes.dox, 204
- IncludeFiles/Smart.h, 204
- IncludeFiles/Telematics.h, 205
- IncludeFiles/TouchScreen.h, 207
- IncludeFiles/TouchScreenCalib.h, 208
- IncludeFiles/Video.h, 209
- JidaSensorType
  - CrossControl, 34
- LIGHTSENSORHANDLE
  - CrossControl, 30
- LedColorMixType, 179
- ledNbr
  - CrossControl::FpgaLedTimingType, 178
- LedTimingType, 179
- LightSensorOperationRange
  - CrossControl, 35
- LightSensorSamplingMode
  - CrossControl, 35
- Lightsensor\_getAverageIlluminance
  - CrossControl, 125
- Lightsensor\_getIlluminance
  - CrossControl, 125
- Lightsensor\_getIlluminance2
  - CrossControl, 126
- Lightsensor\_getOperatingRange
  - CrossControl, 126
- Lightsensor\_release
  - CrossControl, 126
- Lightsensor\_setOperatingRange

- CrossControl, 127
- Lightsensor\_startAverageCalc
  - CrossControl, 127
- Lightsensor\_stopAverageCalc
  - CrossControl, 128
- MAGENTA
  - CrossControl, 32
- MODE\_CALIBRATION\_13P
  - CrossControl, 31
- MODE\_CALIBRATION\_5P
  - CrossControl, 31
- MODE\_CALIBRATION\_9P
  - CrossControl, 31
- MODE\_NORMAL
  - CrossControl, 31
- MODE\_UNKNOWN
  - CrossControl, 31
- MOUSE\_NEXT\_BOOT
  - CrossControl, 37
- MOUSE\_NOW
  - CrossControl, 37
- major
  - CrossControl::version\_info, 183
- minor
  - CrossControl::version\_info, 183
- NoAction
  - CrossControl, 36
- NoRequestsPending
  - CrossControl, 36
- Normal
  - CrossControl, 36
- nrOfPulses
  - CrossControl::FpgaLedTimingType, 178
  - CrossControl::LedTimingType, 179
- OCD\_OC
  - CrossControl, 35
- OCD\_OK
  - CrossControl, 35
- OCD\_POWER\_OFF
  - CrossControl, 35
- OCDStatus
  - CrossControl, 35
- offTime
  - CrossControl::FpgaLedTimingType, 178
  - CrossControl::LedTimingType, 180
- OnOff\_Signal\_Enabled
  - CrossControl, 38
- onTime
  - CrossControl::FpgaLedTimingType, 178
  - CrossControl::LedTimingType, 180
- POWERHANDLE
  - CrossControl, 30
- POWERMGRHANDLE
  - CrossControl, 30
- percent
  - CrossControl::UpgradeStatus, 182
- PowerSource\_Battery
  - CrossControl, 36
- PowerSource\_ExternalPower
  - CrossControl, 36
- Power\_ackPowerRequest
  - CrossControl, 128
- Power\_getBLPowerStatus
  - CrossControl, 129
- Power\_getButtonPowerTransitionStatus
  - CrossControl, 129
- Power\_getCanOCDSStatus
  - CrossControl, 130
- Power\_getCanPowerStatus
  - CrossControl, 130
- Power\_getExtFanPowerStatus
  - CrossControl, 131
- Power\_getVideoOCDSStatus
  - CrossControl, 131
- Power\_getVideoPowerStatus
  - CrossControl, 132
- Power\_release
  - CrossControl, 132
- Power\_setBLPowerStatus
  - CrossControl, 133
- Power\_setCanPowerStatus
  - CrossControl, 133
- Power\_setExtFanPowerStatus
  - CrossControl, 134
- Power\_setVideoPowerStatus
  - CrossControl, 134
- PowerAction
  - CrossControl, 35
- PowerMgr\_getConfiguration
  - CrossControl, 134
- PowerMgr\_getPowerMgrStatus
  - CrossControl, 135
- PowerMgr\_hasResumed
  - CrossControl, 137
- PowerMgr\_registerControlledSuspendOrShutDown

- CrossControl, 139
- PowerMgr\_release
  - CrossControl, 140
- PowerMgr\_setAppReadyForSuspendOrShutdown
  - CrossControl, 141
- PowerMgrConf
  - CrossControl, 36
- PowerMgrStatus
  - CrossControl, 36
- PowerSource
  - CrossControl, 36
- RED
  - CrossControl, 32
- RS4XXPort1
  - CrossControl, 37
- RS4XXPort2
  - CrossControl, 37
- RS4XXPort3
  - CrossControl, 37
- RS4XXPort4
  - CrossControl, 37
- RS4XXPort
  - CrossControl, 36
- RangeExtended
  - CrossControl, 35
- RangeStandard
  - CrossControl, 35
- received\_framerate
  - CrossControl::received\_video, 180
- received\_height
  - CrossControl::received\_video, 180
- received\_video, 180
- received\_width
  - CrossControl::received\_video, 180
- red
  - CrossControl::LedColorMixType, 179
- register\_value
  - CrossControl::video\_dec\_command, 188
- release
  - CrossControl::version\_info, 183
- Rot180
  - CrossControl, 39
- Rot270
  - CrossControl, 39
- Rot90
  - CrossControl, 39
- RotNone
  - CrossControl, 39
- RunTime40\_60
  - CrossControl::TimerType, 181
- RunTime60\_70
  - CrossControl::TimerType, 181
- RunTime70\_80
  - CrossControl::TimerType, 181
- RunTime\_0\_40
  - CrossControl::BatteryTimerType, 176
- RunTime\_40\_60
  - CrossControl::BatteryTimerType, 176
- RunTime\_60\_70
  - CrossControl::BatteryTimerType, 176
- RunTime\_70\_80
  - CrossControl::BatteryTimerType, 176
- RunTime\_Above80
  - CrossControl::BatteryTimerType, 177
- RunTime\_m20
  - CrossControl::BatteryTimerType, 177
- RunTime\_m20\_0
  - CrossControl::BatteryTimerType, 177
- STD\_B\_D\_G\_H\_I\_N\_PAL
  - CrossControl, 40
- STD\_M\_J\_NTSC
  - CrossControl, 40
- STD\_M\_PAL
  - CrossControl, 40
- STD\_NTSC
  - CrossControl, 40
- STD\_PAL
  - CrossControl, 40
- STD\_SECAM
  - CrossControl, 40
- SMARTHANDLE
  - CrossControl, 30
- SamplingModeAuto
  - CrossControl, 35
- SamplingModeExtended
  - CrossControl, 35
- SamplingModeStandard
  - CrossControl, 35
- shutDownReasonCodes
  - CrossControl, 37
- ShutdownPending
  - CrossControl, 36
- shutdownReasonCodeNoError
  - CrossControl, 37
- Smart\_getDeviceSerial
  - CrossControl, 143
- Smart\_getInitialTime
  - CrossControl, 144

- Smart\_getRemainingLifeTime
  - CrossControl, [144](#)
- Smart\_release
  - CrossControl, [145](#)
- startupReasonCodeButtonPress
  - CrossControl, [37](#)
- startupReasonCodeExtCtrl
  - CrossControl, [37](#)
- startupReasonCodeMPRestart
  - CrossControl, [37](#)
- startupReasonCodePowerOnStartup
  - CrossControl, [37](#)
- startupReasonCodeUndefined
  - CrossControl, [37](#)
- startupReasonCodes
  - CrossControl, [37](#)
- SuspendPending
  - CrossControl, [36](#)
- TEMP\_BACKPLANE
  - CrossControl, [35](#)
- TEMP\_BOARD
  - CrossControl, [35](#)
- TEMP\_BOX
  - CrossControl, [34](#)
- TEMP\_CHIPSETS
  - CrossControl, [35](#)
- TEMP\_CPU
  - CrossControl, [34](#)
- TEMP\_ENV
  - CrossControl, [35](#)
- TEMP\_OTHER
  - CrossControl, [35](#)
- TEMP\_VIDEO
  - CrossControl, [35](#)
- TOUCH\_NEXT\_BOOT
  - CrossControl, [37](#)
- TOUCH\_NOW
  - CrossControl, [37](#)
- TS\_CALIBRATION\_MEASUREMENT-
  - S
  - CrossControl, [38](#)
- TS\_CALIBTATION\_WIDTH
  - CrossControl, [38](#)
- TS\_DEBOUNCE\_TIME
  - CrossControl, [38](#)
- TS\_DEBOUNCE\_TIMEOUT\_TIME
  - CrossControl, [38](#)
- TS\_DOUBLE\_CLICK\_TIME
  - CrossControl, [38](#)
- TS\_DOUBLECLICK\_MAX\_CLICK\_TI-
  - ME
  - CrossControl, [38](#)
- TS\_LOW\_LEVEL
  - CrossControl, [38](#)
- TS\_MAX\_RIGHTCLICK\_DISTANCE
  - CrossControl, [38](#)
- TS\_RESTORE\_DEFAULT\_SETTINGS
  - CrossControl, [38](#)
- TS\_RIGHT\_CLICK\_TIME
  - CrossControl, [38](#)
- TS\_TCHAUTOCAL
  - CrossControl, [39](#)
- TS\_UNTOUCHLEVEL
  - CrossControl, [38](#)
- TS\_USE\_DEJITTER
  - CrossControl, [38](#)
- TELEMATICSHANDLE
  - CrossControl, [30](#)
- TOUCHSCREENHANDLE
  - CrossControl, [30](#)
- TSAdvancedSettingsParameter
  - CrossControl, [38](#)
- Telematics\_getBTPowerStatus
  - CrossControl, [145](#)
- Telematics\_getBTStartUpPowerStatus
  - CrossControl, [146](#)
- Telematics\_getGPRSPowerStatus
  - CrossControl, [147](#)
- Telematics\_getGPRSStartUpPowerStatus
  - CrossControl, [147](#)
- Telematics\_getGPSAntennaStatus
  - CrossControl, [148](#)
- Telematics\_getGPSPowerStatus
  - CrossControl, [148](#)
- Telematics\_getGPSStartUpPowerStatus
  - CrossControl, [149](#)
- Telematics\_getTelematicsAvailable
  - CrossControl, [150](#)
- Telematics\_getWLANPowerStatus
  - CrossControl, [150](#)
- Telematics\_getWLANStartUpPowerStatus
  - CrossControl, [151](#)
- Telematics\_release
  - CrossControl, [152](#)
- Telematics\_setBTPowerStatus
  - CrossControl, [152](#)
- Telematics\_setBTStartUpPowerStatus
  - CrossControl, [152](#)
- Telematics\_setGPRSPowerStatus

- CrossControl, 153
- Telematics\_setGPRSStartupPowerStatus
  - CrossControl, 153
- Telematics\_setGPSPowerStatus
  - CrossControl, 153
- Telematics\_setGPSStartUpPowerStatus
  - CrossControl, 154
- Telematics\_setWLANPowerStatus
  - CrossControl, 154
- Telematics\_setWLANStartupPowerStatus
  - CrossControl, 155
- TimerType, 180
- TotHeatTime
  - CrossControl::TimerType, 181
- TotRunTime
  - CrossControl::TimerType, 181
- TotRunTimeBattery
  - CrossControl::BatteryTimerType, 177
- TotRunTimeMain
  - CrossControl::BatteryTimerType, 177
- TotSuspTime
  - CrossControl::TimerType, 181
- TouchScreen\_getAdvancedSetting
  - CrossControl, 155
- TouchScreen\_getMode
  - CrossControl, 156
- TouchScreen\_getMouseRightClickTime
  - CrossControl, 156
- TouchScreen\_release
  - CrossControl, 157
- TouchScreen\_setAdvancedSetting
  - CrossControl, 157
- TouchScreen\_setMode
  - CrossControl, 158
- TouchScreen\_setMouseRightClickTime
  - CrossControl, 158
- TouchScreenCalib\_checkCalibrationPoint-  
Finished
  - CrossControl, 159
- TouchScreenCalib\_getConfigParam
  - CrossControl, 159
- TouchScreenCalib\_getMode
  - CrossControl, 159
- TouchScreenCalib\_release
  - CrossControl, 160
- TouchScreenCalib\_setCalibrationPoint
  - CrossControl, 160
- TouchScreenCalib\_setConfigParam
  - CrossControl, 160
- TouchScreenCalib\_setMode
  - CrossControl, 161
- TouchScreenModeSettings
  - CrossControl, 37
- TriggerConf
  - CrossControl, 37
- UNDEFINED\_COLOR
  - CrossControl, 32
- UPGRADE\_COMPLETE
  - CrossControl, 39
- UPGRADE\_COMPLETE\_WITH\_ERRORS
  - CrossControl, 39
- UPGRADE\_CONVERTING\_FILE
  - CrossControl, 39
- UPGRADE\_FLASHING
  - CrossControl, 39
- UPGRADE\_INIT
  - CrossControl, 39
- UPGRADE\_PREP\_COM
  - CrossControl, 39
- UPGRADE\_READING\_FILE
  - CrossControl, 39
- UPGRADE\_VERIFYING
  - CrossControl, 39
- UpgradeAction
  - CrossControl, 39
- UpgradeStatus, 182
- VOLTAGE\_0V9
  - CrossControl, 40
- VOLTAGE\_12V
  - CrossControl, 40
- VOLTAGE\_12VID
  - CrossControl, 40
- VOLTAGE\_1V0
  - CrossControl, 40
- VOLTAGE\_1V05
  - CrossControl, 40
- VOLTAGE\_1V1
  - CrossControl, 40
- VOLTAGE\_1V2
  - CrossControl, 40
- VOLTAGE\_1V3\_PER
  - CrossControl, 40
- VOLTAGE\_1V3\_VDDA
  - CrossControl, 40
- VOLTAGE\_1V5
  - CrossControl, 40
- VOLTAGE\_1V8

- CrossControl, [40](#)
- VOLTAGE\_1V9
  - CrossControl, [40](#)
- VOLTAGE\_24V
  - CrossControl, [40](#)
- VOLTAGE\_24V\_BACKUP
  - CrossControl, [40](#)
- VOLTAGE\_24VIN
  - CrossControl, [40](#)
- VOLTAGE\_2V5
  - CrossControl, [40](#)
- VOLTAGE\_3V3
  - CrossControl, [40](#)
- VOLTAGE\_5V
  - CrossControl, [40](#)
- VOLTAGE\_5VSTB
  - CrossControl, [40](#)
- VOLTAGE\_VREF\_INT
  - CrossControl, [40](#)
- VOLTAGE\_VTFT
  - CrossControl, [40](#)
- VIDEOHANDLE
  - CrossControl, [30](#)
- version\_info, [182](#)
- VersionType
  - CrossControl, [30](#)
- Video1Conf
  - CrossControl, [174](#)
- Video2Conf
  - CrossControl, [174](#)
- Video3Conf
  - CrossControl, [175](#)
- Video4Conf
  - CrossControl, [175](#)
- Video\_activateSnapshot
  - CrossControl, [161](#)
- Video\_createBitmap
  - CrossControl, [161](#)
- video\_dec\_command, [183](#)
- Video\_freeBmpBuffer
  - CrossControl, [162](#)
- Video\_getActiveChannel
  - CrossControl, [162](#)
- Video\_getColorKeys
  - CrossControl, [163](#)
- Video\_getCropping
  - CrossControl, [163](#)
- Video\_getDeInterlaceMode
  - CrossControl, [164](#)
- Video\_getDecoderReg
  - CrossControl, [164](#)
- Video\_getMirroring
  - CrossControl, [164](#)
- Video\_getRawImage
  - CrossControl, [165](#)
- Video\_getRotation
  - CrossControl, [165](#)
- Video\_getScaling
  - CrossControl, [165](#)
- Video\_getStatus
  - CrossControl, [166](#)
- Video\_getVideoArea
  - CrossControl, [166](#)
- Video\_getVideoStandard
  - CrossControl, [167](#)
- Video\_init
  - CrossControl, [167](#)
- Video\_minimize
  - CrossControl, [167](#)
- Video\_release
  - CrossControl, [168](#)
- Video\_restore
  - CrossControl, [168](#)
- Video\_setActiveChannel
  - CrossControl, [168](#)
- Video\_setColorKeys
  - CrossControl, [169](#)
- Video\_setCropping
  - CrossControl, [169](#)
- Video\_setDeInterlaceMode
  - CrossControl, [170](#)
- Video\_setDecoderReg
  - CrossControl, [169](#)
- Video\_setMirroring
  - CrossControl, [170](#)
- Video\_setRotation
  - CrossControl, [171](#)
- Video\_setScaling
  - CrossControl, [171](#)
- Video\_setVideoArea
  - CrossControl, [171](#)
- Video\_showFrame
  - CrossControl, [172](#)
- Video\_showVideo
  - CrossControl, [172](#)
- Video\_takeSnapshot
  - CrossControl, [173](#)
- Video\_takeSnapshotBmp
  - CrossControl, [173](#)
- Video\_takeSnapshotRaw



- CrossControl, [174](#)
- VideoChannel
  - CrossControl, [39](#)
- VideoRotation
  - CrossControl, [39](#)
- videoStandard
  - CrossControl, [39](#)
- VoltageEnum
  - CrossControl, [40](#)
- volume
  - CrossControl::BuzzerSetup, [177](#)
- YELLOW
  - CrossControl, [32](#)