

**cr•ssc•ntr•l**

CCAux  
2.2.0.0

Mon Jan 28 2013

# Contents

<b>1</b>	<b>Namespace Index</b>	<b>1</b>
1.1	Namespace List . . . . .	1
<b>2</b>	<b>Class Index</b>	<b>2</b>
2.1	Class List . . . . .	2
<b>3</b>	<b>File Index</b>	<b>3</b>
3.1	File List . . . . .	3
<b>4</b>	<b>Namespace Documentation</b>	<b>4</b>
4.1	CrossControl Namespace Reference . . . . .	4
4.1.1	Typedef Documentation . . . . .	18
4.1.1.1	ABOUTHANDLE . . . . .	18
4.1.1.2	ADCHANDLE . . . . .	18
4.1.1.3	AUXVERSIONHANDLE . . . . .	18
4.1.1.4	BACKLIGHTHANDLE . . . . .	18
4.1.1.5	BUZZERHANDLE . . . . .	18
4.1.1.6	CANSETTINGHANDLE . . . . .	18
4.1.1.7	CONFIGHANDLE . . . . .	18
4.1.1.8	DIAGNOSTICHANDLE . . . . .	18
4.1.1.9	DIGIOHANDLE . . . . .	18
4.1.1.10	FIRMWAREUPGHANDLE . . . . .	19
4.1.1.11	FRONTLEDHANDLE . . . . .	19
4.1.1.12	LIGHTSENSORHANDLE . . . . .	19
4.1.1.13	POWERHANDLE . . . . .	19
4.1.1.14	TELEMATICSHANDLE . . . . .	19
4.1.1.15	TOUCHSCREENCALIBHANDLE . . . . .	19
4.1.1.16	TOUCHSCREENHANDLE . . . . .	19
4.1.1.17	VersionType . . . . .	19
4.1.1.18	VIDEOHANDLE . . . . .	19
4.1.2	Enumeration Type Documentation . . . . .	19
4.1.2.1	ButtonPowerTransitionStatus . . . . .	19
4.1.2.2	CalibrationConfigParam . . . . .	19
4.1.2.3	CalibrationModeSettings . . . . .	20
4.1.2.4	CanFrameType . . . . .	20
4.1.2.5	CCAuxColor . . . . .	20
4.1.2.6	CCStatus . . . . .	21
4.1.2.7	DeInterlaceMode . . . . .	21
4.1.2.8	eErr . . . . .	21

---

4.1.2.9	hwErrorStatusCodes	22
4.1.2.10	JidaSensorType	22
4.1.2.11	LightSensorOperationRange	23
4.1.2.12	LightSensorSamplingMode	23
4.1.2.13	OCDStatus	23
4.1.2.14	PowerAction	23
4.1.2.15	shutDownReasonCodes	24
4.1.2.16	startupReasonCodes	24
4.1.2.17	TouchScreenModeSettings	24
4.1.2.18	TriggerConf	25
4.1.2.19	TSAdvancedSettingsParameter	25
4.1.2.20	UpgradeAction	26
4.1.2.21	VideoChannel	26
4.1.2.22	videoStandard	26
4.1.2.23	VoltageEnum	27
4.1.3	Function Documentation	27
4.1.3.1	About_getAddOnHWversion	27
4.1.3.2	About_getAddOnManufacturingDate	28
4.1.3.3	About_getAddOnPCBArt	28
4.1.3.4	About_getAddOnPCBSerial	29
4.1.3.5	About_getDisplayResolution	29
4.1.3.6	About_getFrontPcbRev	30
4.1.3.7	About_getIOExpanderValue	30
4.1.3.8	About_getIsAnybusMounted	30
4.1.3.9	About_getIsBTMounted	31
4.1.3.10	About_getIsDisplayAvailable	31
4.1.3.11	About_getIsGPRSMounted	31
4.1.3.12	About_getIsGPSPMounted	32
4.1.3.13	About_getIsIOExpanderMounted	32
4.1.3.14	About_getIsTouchScreenAvailable	33
4.1.3.15	About_getIsWLANMounted	33
4.1.3.16	About_getMainHWversion	33
4.1.3.17	About_getMainManufacturingDate	34
4.1.3.18	About_getMainPCBArt	34
4.1.3.19	About_getMainPCBSerial	35
4.1.3.20	About_getMainProdArtNr	35
4.1.3.21	About_getMainProdRev	36
4.1.3.22	About_getNrOfCANConnections	36
4.1.3.23	About_getNrOfDigIOConnections	37
4.1.3.24	About_getNrOfETHConnections	37
4.1.3.25	About_getNrOfSerialConnections	37
4.1.3.26	About_getNrOfUSBConnections	38
4.1.3.27	About_getNrOfVideoConnections	38
4.1.3.28	About_getUnitSerial	39
4.1.3.29	About_hasOsBooted	39
4.1.3.30	About_release	40
4.1.3.31	Adc_getVoltage	40
4.1.3.32	Adc_release	40
4.1.3.33	AuxVersion_getCCAuxDrvVersion	41
4.1.3.34	AuxVersion_getCCAuxVersion	42

4.1.3.35	AuxVersion_getFPGAVersion	43
4.1.3.36	AuxVersion_getFrontVersion	43
4.1.3.37	AuxVersion_getOSVersion	44
4.1.3.38	AuxVersion_getSSVersion	45
4.1.3.39	AuxVersion_release	45
4.1.3.40	Backlight_getAutomaticBLFilter	46
4.1.3.41	Backlight_getAutomaticBLParams	46
4.1.3.42	Backlight_getAutomaticBLStatus	46
4.1.3.43	Backlight_getIntensity	47
4.1.3.44	Backlight_getLedDimming	47
4.1.3.45	Backlight_getStatus	48
4.1.3.46	Backlight_release	48
4.1.3.47	Backlight_setAutomaticBLFilter	49
4.1.3.48	Backlight_setAutomaticBLParams	49
4.1.3.49	Backlight_setIntensity	49
4.1.3.50	Backlight_setLedDimming	50
4.1.3.51	Backlight_startAutomaticBL	50
4.1.3.52	Backlight_stopAutomaticBL	50
4.1.3.53	Buzzer_buzze	51
4.1.3.54	Buzzer_getFrequency	51
4.1.3.55	Buzzer_getTrigger	52
4.1.3.56	Buzzer_getVolume	52
4.1.3.57	Buzzer_release	52
4.1.3.58	Buzzer_setFrequency	53
4.1.3.59	Buzzer_setTrigger	53
4.1.3.60	Buzzer_setVolume	54
4.1.3.61	CanSetting_getBaudrate	54
4.1.3.62	CanSetting_getFrameType	55
4.1.3.63	CanSetting_release	55
4.1.3.64	CanSetting_setBaudrate	55
4.1.3.65	CanSetting_setFrameType	56
4.1.3.66	Config_getCanStartupPowerConfig	56
4.1.3.67	Config_getExtFanStartupPowerConfig	57
4.1.3.68	Config_getExtOnOffSigTrigTime	57
4.1.3.69	Config_getFrontBtnTrigTime	57
4.1.3.70	Config_getHeatingTempLimit	58
4.1.3.71	Config_getLongButtonPressAction	58
4.1.3.72	Config_getOnOffSigAction	58
4.1.3.73	Config_getPowerOnStartup	59
4.1.3.74	Config_getShortButtonPressAction	59
4.1.3.75	Config_getStartupTriggerConfig	59
4.1.3.76	Config_getStartupVoltageConfig	60
4.1.3.77	Config_getSuspendMaxTime	60
4.1.3.78	Config_getTFTMirror	61
4.1.3.79	Config_getTFTMode	61
4.1.3.80	Config_getTFTScan	61
4.1.3.81	Config_getVideoStartupPowerConfig	61
4.1.3.82	Config_release	62
4.1.3.83	Config_setCanStartupPowerConfig	62
4.1.3.84	Config_setExtFanStartupPowerConfig	62

---

4.1.3.85	Config_setExtOnOffSigTrigTime	63
4.1.3.86	Config_setFrontBtnTrigTime	63
4.1.3.87	Config_setHeatingTempLimit	63
4.1.3.88	Config_setLongButtonPressAction	64
4.1.3.89	Config_setOnOffSigAction	64
4.1.3.90	Config_setPowerOnStartup	65
4.1.3.91	Config_setShortButtonPressAction	65
4.1.3.92	Config_setStartupTriggerConfig	66
4.1.3.93	Config_setStartupVoltageConfig	66
4.1.3.94	Config_setSuspendMaxTime	66
4.1.3.95	Config_setTFTMirror	67
4.1.3.96	Config_setTFTMode	67
4.1.3.97	Config_setTFTScan	67
4.1.3.98	Config_setVideoStartupPowerConfig	67
4.1.3.99	Diagnostic_clearHwErrorStatus	68
4.1.3.100	Diagnostic_getHwErrorStatus	68
4.1.3.101	Diagnostic_getMinMaxTemp	68
4.1.3.102	Diagnostic_getPCBTemp	69
4.1.3.103	Diagnostic_getPMTemp	69
4.1.3.104	Diagnostic_getPowerCycles	69
4.1.3.105	Diagnostic_getShutDownReason	70
4.1.3.106	Diagnostic_getSSTemp	70
4.1.3.107	Diagnostic_getStartupReason	70
4.1.3.108	Diagnostic_getTimer	71
4.1.3.109	Diagnostic_release	71
4.1.3.110	DigIO_getDigIO	71
4.1.3.111	DigIO_release	72
4.1.3.112	DigIO_setDigIO	72
4.1.3.113	FirmwareUpgrade_getUpgradeStatus	73
4.1.3.114	FirmwareUpgrade_release	73
4.1.3.115	FirmwareUpgrade_shutDown	73
4.1.3.116	FirmwareUpgrade_startFpgaUpgrade	74
4.1.3.117	FirmwareUpgrade_startFpgaVerification	75
4.1.3.118	FirmwareUpgrade_startFrontUpgrade	76
4.1.3.119	FirmwareUpgrade_startFrontVerification	77
4.1.3.120	FirmwareUpgrade_startSSUpgrade	78
4.1.3.121	FirmwareUpgrade_startSSVerification	79
4.1.3.122	FrontLED_getColor	80
4.1.3.123	FrontLED_getEnabledDuringStartup	80
4.1.3.124	FrontLED_getFrontPcbRev	80
4.1.3.125	FrontLED_getIdleTime	81
4.1.3.126	FrontLED_getNrOfPulses	81
4.1.3.127	FrontLED_getOffTime	81
4.1.3.128	FrontLED_getOnTime	82
4.1.3.129	FrontLED_getSignal	82
4.1.3.130	FrontLED_getStandardColor	82
4.1.3.131	FrontLED_release	83
4.1.3.132	FrontLED_setColor	83
4.1.3.133	FrontLED_setEnabledDuringStartup	84
4.1.3.134	FrontLED_setIdleTime	84

---

4.1.3.135 FrontLED_setNrOfPulses . . . . .	84
4.1.3.136 FrontLED_setOff . . . . .	84
4.1.3.137 FrontLED_setOffTime . . . . .	85
4.1.3.138 FrontLED_setOnTime . . . . .	85
4.1.3.139 FrontLED_setSignal . . . . .	86
4.1.3.140 FrontLED_setStandardColor . . . . .	86
4.1.3.141 GetAbout . . . . .	86
4.1.3.142 GetAdc . . . . .	87
4.1.3.143 GetAuxVersion . . . . .	87
4.1.3.144 GetBacklight . . . . .	88
4.1.3.145 GetBuzzer . . . . .	88
4.1.3.146 GetCanSetting . . . . .	89
4.1.3.147 GetConfig . . . . .	89
4.1.3.148 GetDiagnostic . . . . .	89
4.1.3.149 GetDigIO . . . . .	90
4.1.3.150 GetErrorStringA . . . . .	90
4.1.3.151 GetErrorStringW . . . . .	91
4.1.3.152 GetFirmwareUpgrade . . . . .	91
4.1.3.153 GetFrontLED . . . . .	91
4.1.3.154 GetHwErrorStatusStringA . . . . .	92
4.1.3.155 GetHwErrorStatusStringW . . . . .	92
4.1.3.156 GetLightsensor . . . . .	92
4.1.3.157 GetPower . . . . .	93
4.1.3.158 GetStartupReasonStringA . . . . .	93
4.1.3.159 GetStartupReasonStringW . . . . .	93
4.1.3.160 GetTelematics . . . . .	94
4.1.3.161 GetTouchScreen . . . . .	94
4.1.3.162 GetTouchScreenCalib . . . . .	94
4.1.3.163 GetVideo . . . . .	95
4.1.3.164 Lightsensor_getAverageIlluminance . . . . .	95
4.1.3.165 Lightsensor_getIlluminance . . . . .	95
4.1.3.166 Lightsensor_getIlluminance2 . . . . .	96
4.1.3.167 Lightsensor_getOperatingRange . . . . .	96
4.1.3.168 Lightsensor_release . . . . .	96
4.1.3.169 Lightsensor_setOperatingRange . . . . .	97
4.1.3.170 Lightsensor_startAverageCalc . . . . .	97
4.1.3.171 Lightsensor_stopAverageCalc . . . . .	98
4.1.3.172 Power_ackPowerRequest . . . . .	98
4.1.3.173 Power_getBLPowerStatus . . . . .	99
4.1.3.174 Power_getButtonPowerTransitionStatus . . . . .	99
4.1.3.175 Power_getCanOCDStatus . . . . .	99
4.1.3.176 Power_getCanPowerStatus . . . . .	100
4.1.3.177 Power_getExtFanPowerStatus . . . . .	101
4.1.3.178 Power_getVideoOCDStatus . . . . .	101
4.1.3.179 Power_getVideoPowerStatus . . . . .	102
4.1.3.180 Power_release . . . . .	102
4.1.3.181 Power_setBLPowerStatus . . . . .	103
4.1.3.182 Power_setCanPowerStatus . . . . .	103
4.1.3.183 Power_setExtFanPowerStatus . . . . .	104
4.1.3.184 Power_setVideoPowerStatus . . . . .	104

---

4.1.3.185 Telematics_getBTPowerStatus . . . . .	104
4.1.3.186 Telematics_getBTStartUpPowerStatus . . . . .	105
4.1.3.187 Telematics_getGPRSPowerStatus . . . . .	105
4.1.3.188 Telematics_getGPRSStartUpPowerStatus . . . . .	106
4.1.3.189 Telematics_getGPSAntennaStatus . . . . .	107
4.1.3.190 Telematics_getGPSPowerStatus . . . . .	107
4.1.3.191 Telematics_getGPSStartUpPowerStatus . . . . .	108
4.1.3.192 Telematics_getTelematicsAvailable . . . . .	109
4.1.3.193 Telematics_getWLANPowerStatus . . . . .	109
4.1.3.194 Telematics_getWLANStartUpPowerStatus . . . . .	110
4.1.3.195 Telematics_release . . . . .	110
4.1.3.196 Telematics_setBTPowerStatus . . . . .	111
4.1.3.197 Telematics_setBTStartUpPowerStatus . . . . .	111
4.1.3.198 Telematics_setGPRSPowerStatus . . . . .	111
4.1.3.199 Telematics_setGPRSStartUpPowerStatus . . . . .	112
4.1.3.200 Telematics_setGPSPowerStatus . . . . .	112
4.1.3.201 Telematics_setGPSStartUpPowerStatus . . . . .	112
4.1.3.202 Telematics_setWLANPowerStatus . . . . .	113
4.1.3.203 Telematics_setWLANStartUpPowerStatus . . . . .	113
4.1.3.204 TouchScreen_getAdvancedSetting . . . . .	113
4.1.3.205 TouchScreen_getMode . . . . .	114
4.1.3.206 TouchScreen_getMouseRightClickTime . . . . .	115
4.1.3.207 TouchScreen_release . . . . .	115
4.1.3.208 TouchScreen_setAdvancedSetting . . . . .	115
4.1.3.209 TouchScreen_setMode . . . . .	116
4.1.3.210 TouchScreen_setMouseRightClickTime . . . . .	116
4.1.3.211 TouchScreenCalib_checkCalibrationPointFinished . . . . .	116
4.1.3.212 TouchScreenCalib_getConfigParam . . . . .	117
4.1.3.213 TouchScreenCalib_getMode . . . . .	117
4.1.3.214 TouchScreenCalib_release . . . . .	117
4.1.3.215 TouchScreenCalib_setCalibrationPoint . . . . .	118
4.1.3.216 TouchScreenCalib_setConfigParam . . . . .	118
4.1.3.217 TouchScreenCalib_setMode . . . . .	118
4.1.3.218 Video_activateSnapshot . . . . .	119
4.1.3.219 Video_createBitmap . . . . .	119
4.1.3.220 Video_freeBmpBuffer . . . . .	120
4.1.3.221 Video_getActiveChannel . . . . .	120
4.1.3.222 Video_getColorKeys . . . . .	120
4.1.3.223 Video_getCropping . . . . .	121
4.1.3.224 Video_getDecoderReg . . . . .	121
4.1.3.225 Video_getDeInterlaceMode . . . . .	121
4.1.3.226 Video_getMirroring . . . . .	122
4.1.3.227 Video_getRawImage . . . . .	122
4.1.3.228 Video_getScaling . . . . .	122
4.1.3.229 Video_getStatus . . . . .	123
4.1.3.230 Video_getVideoArea . . . . .	123
4.1.3.231 Video_getVideoStandard . . . . .	124
4.1.3.232 Video_init . . . . .	124
4.1.3.233 Video_minimize . . . . .	124
4.1.3.234 Video_release . . . . .	124

4.1.3.235	Video_restore	125
4.1.3.236	Video_setActiveChannel	125
4.1.3.237	Video_setColorKeys	125
4.1.3.238	Video_setCropping	125
4.1.3.239	Video_setDecoderReg	126
4.1.3.240	Video_setDeInterlaceMode	126
4.1.3.241	Video_setMirroring	127
4.1.3.242	Video_setScaling	127
4.1.3.243	Video_setVideoArea	127
4.1.3.244	Video_showFrame	128
4.1.3.245	Video_showVideo	128
4.1.3.246	Video_takeSnapshot	128
4.1.3.247	Video_takeSnapshotBmp	129
4.1.3.248	Video_takeSnapshotRaw	129
4.1.4	Variable Documentation	130
4.1.4.1	DigitalIn_1	130
4.1.4.2	DigitalIn_2	130
4.1.4.3	DigitalIn_3	130
4.1.4.4	DigitalIn_4	130
4.1.4.5	Video1Conf	130
4.1.4.6	Video2Conf	130
4.1.4.7	Video3Conf	130
4.1.4.8	Video4Conf	130
<b>5</b>	<b>Class Documentation</b>	<b>131</b>
5.1	CrossControl::BuzzerSetup Struct Reference	131
5.1.1	Member Data Documentation	131
5.1.1.1	frequency	131
5.1.1.2	volume	131
5.2	CrossControl::FpgaLedTimingType Struct Reference	131
5.2.1	Member Data Documentation	132
5.2.1.1	idleTime	132
5.2.1.2	ledNbr	132
5.2.1.3	nrOfPulses	132
5.2.1.4	offTime	132
5.2.1.5	onTime	132
5.3	CrossControl::LedColorMixType Struct Reference	132
5.3.1	Member Data Documentation	133
5.3.1.1	blue	133
5.3.1.2	green	133
5.3.1.3	red	133
5.4	CrossControl::LedTimingType Struct Reference	133
5.4.1	Member Data Documentation	133
5.4.1.1	idleTime	133
5.4.1.2	nrOfPulses	133
5.4.1.3	offTime	133
5.4.1.4	onTime	134
5.5	CrossControl::received_video Struct Reference	134
5.5.1	Member Data Documentation	134
5.5.1.1	received_framerate	134



5.5.1.2	received_height	134
5.5.1.3	received_width	134
5.6	CrossControl::TimerType Struct Reference	134
5.6.1	Detailed Description	135
5.6.2	Member Data Documentation	135
5.6.2.1	Above80RunTime	135
5.6.2.2	RunTime40_60	135
5.6.2.3	RunTime60_70	135
5.6.2.4	RunTime70_80	135
5.6.2.5	TotHeatTime	135
5.6.2.6	TotRunTime	135
5.6.2.7	TotSuspTime	135
5.7	CrossControl::UpgradeStatus Struct Reference	135
5.7.1	Detailed Description	136
5.7.2	Member Data Documentation	136
5.7.2.1	currentAction	136
5.7.2.2	errorCode	136
5.7.2.3	percent	136
5.8	CrossControl::version_info Struct Reference	136
5.8.1	Member Data Documentation	136
5.8.1.1	build	136
5.8.1.2	major	136
5.8.1.3	minor	137
5.8.1.4	release	137
5.9	CrossControl::video_dec_command Struct Reference	137
5.9.1	Member Data Documentation	137
5.9.1.1	decoder_register	137
5.9.1.2	register_value	137
<b>6</b>	<b>File Documentation</b>	<b>138</b>
6.1	fixedIncludeFiles/About.h File Reference	138
6.2	fixedIncludeFiles/Adc.h File Reference	140
6.3	fixedIncludeFiles/AuxVersion.h File Reference	140
6.4	fixedIncludeFiles/Backlight.h File Reference	141
6.5	fixedIncludeFiles/Buzzer.h File Reference	142
6.6	fixedIncludeFiles/CanSetting.h File Reference	143
6.7	fixedIncludeFiles/CCAuxErrors.h File Reference	143
6.8	fixedIncludeFiles/CCAuxTypes.h File Reference	144
6.9	fixedIncludeFiles/CCPlatform.h File Reference	146
6.10	fixedIncludeFiles/Config.h File Reference	146
6.11	fixedIncludeFiles/Diagnostic.h File Reference	148
6.12	fixedIncludeFiles/DiagnosticCodes.h File Reference	149
6.13	fixedIncludeFiles/DigIO.h File Reference	150
6.14	fixedIncludeFiles/FirmwareUpgrade.h File Reference	150
6.15	fixedIncludeFiles/FrontLED.h File Reference	151
6.16	fixedIncludeFiles/Lightsensor.h File Reference	153
6.17	fixedIncludeFiles/Power.h File Reference	153
6.18	fixedIncludeFiles/Telematics.h File Reference	154
6.19	fixedIncludeFiles/TouchScreen.h File Reference	156
6.20	fixedIncludeFiles/TouchScreenCalib.h File Reference	157

---

6.21 [fixedIncludeFiles/Video.h File Reference](#) . . . . . 158

# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">CrossControl</a> . . . . .	4
--	---

## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">CrossControl::BuzzerSetup</a>	131
<a href="#">CrossControl::FpgaLedTimingType</a>	131
<a href="#">CrossControl::LedColorMixType</a>	132
<a href="#">CrossControl::LedTimingType</a>	133
<a href="#">CrossControl::received_video</a>	134
<a href="#">CrossControl::TimerType</a>	134
<a href="#">CrossControl::UpgradeStatus</a>	135
<a href="#">CrossControl::version_info</a>	136
<a href="#">CrossControl::video_dec_command</a>	137

## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

<a href="#">fixedIncludeFiles/About.h</a>	138
<a href="#">fixedIncludeFiles/Adc.h</a>	140
<a href="#">fixedIncludeFiles/AuxVersion.h</a>	140
<a href="#">fixedIncludeFiles/Backlight.h</a>	141
<a href="#">fixedIncludeFiles/Buzzer.h</a>	142
<a href="#">fixedIncludeFiles/CanSetting.h</a>	143
<a href="#">fixedIncludeFiles/CCAuxErrors.h</a>	143
<a href="#">fixedIncludeFiles/CCAuxTypes.h</a>	144
<a href="#">fixedIncludeFiles/CCPlatform.h</a>	146
<a href="#">fixedIncludeFiles/Config.h</a>	146
<a href="#">fixedIncludeFiles/Diagnostic.h</a>	148
<a href="#">fixedIncludeFiles/DiagnosticCodes.h</a>	149
<a href="#">fixedIncludeFiles/DigIO.h</a>	150
<a href="#">fixedIncludeFiles/FirmwareUpgrade.h</a>	150
<a href="#">fixedIncludeFiles/FrontLED.h</a>	151
<a href="#">fixedIncludeFiles/Lightsensor.h</a>	153
<a href="#">fixedIncludeFiles/Power.h</a>	153
<a href="#">fixedIncludeFiles/Telematics.h</a>	154
<a href="#">fixedIncludeFiles/TouchScreen.h</a>	156
<a href="#">fixedIncludeFiles/TouchScreenCalib.h</a>	157
<a href="#">fixedIncludeFiles/Video.h</a>	158

## Chapter 4

# Namespace Documentation

### 4.1 CrossControl Namespace Reference

#### Classes

- 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 \* [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 void \* [TELEMATICSHANDLE](#)
- typedef void \* [TOUCHSCREENHANDLE](#)
- typedef void \* [TOUCHSCREENCALIBHANDLE](#)
- typedef void \* [VIDEOHANDLE](#)

## Enumerations

- enum [VoltageEnum](#) { [VOLTAGE\\_24VIN](#) = 0, [VOLTAGE\\_24V](#), [VOLTAGE\\_12V](#), [VOLTAGE\\_12VID](#), [VOLTAGE\\_5V](#), [VOLTAGE\\_3V3](#), [VOLTAGE\\_VTFT](#), [VOLTAGE\\_5VSTB](#), [VOLTAGE\\_1V9](#), [VOLTAGE\\_1V8](#), [VOLTAGE\\_1V5](#), [VOLTAGE\\_1V2](#), [VOLTAGE\\_1V05](#), [VOLTAGE\\_1V0](#), [VOLTAGE\\_0V9](#), [VOLTAGE\\_VREF\\_INT](#), [VOLTAGE\\_24V\\_BACKUP](#), [VOLTAGE\\_2V5](#), [VOLTAGE\\_1V1](#), [VOLTAGE\\_1V3\\_PER](#), [VOLTAGE\\_1V3\\_VDDA](#) }
- enum [LightSensorOperationRange](#) { [RangeStandard](#) = 0, [RangeExtended](#) = 1 }
- enum [LightSensorSamplingMode](#) { [SamplingModeStandard](#) = 0, [SamplingModeExtended](#), [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\\_CREATE\\_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\\_PAL](#) = 3, [STD\\_NTSC](#) = 4, [STD\\_SECAM](#) = 5 }
- 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 `startupReasonCodes` { `startupReasonCodeUndefined` = 0x0000, `startupReasonCodeButtonPress` = 0x0055, `startupReasonCodeExtCtrl` = 0x00AA, `startupReasonCodeMPRestart` = 0x00F0, `startupReasonCodePowerOnStartup` = 0x000F }
- enum `shutDownReasonCodes` { `shutdownReasonCodeNoError` = 0x001F }
- enum `hwErrorStatusCodes` { `errCodeNoErr` = 0 }
- 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 }
- 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 \(ABOUTHANDLE, char \\*buff, int len\)](#)
- [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)
- 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\\_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 [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-\\_setTFTMode \(CONFIGHANDLE, bool enable\)](#)
- [EXTERN\\_C CCAUXDLL\\_API eErr CCAUXDLL\\_CALLING\\_CONV Config-\\_setTFTScan \(CONFIGHANDLE, bool enable\)](#)
- [EXTERN\\_C CCAUXDLL\\_API eErr CCAUXDLL\\_CALLING\\_CONV Config-\\_setTFTMirror \(CONFIGHANDLE, bool enable\)](#)
- [EXTERN\\_C CCAUXDLL\\_API eErr CCAUXDLL\\_CALLING\\_CONV Config-\\_getTFTMode \(CONFIGHANDLE, bool \\*enable\)](#)
- [EXTERN\\_C CCAUXDLL\\_API eErr CCAUXDLL\\_CALLING\\_CONV Config-\\_getTFTScan \(CONFIGHANDLE, bool \\*enable\)](#)
- [EXTERN\\_C CCAUXDLL\\_API eErr CCAUXDLL\\_CALLING\\_CONV Config-\\_getTFTMirror \(CONFIGHANDLE, bool \\*enable\)](#)
- [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 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)
- 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 eErr CCAUXDLL\_CALLING\_CONV FrontLED\_getFrontPcbRev (FRONTLEDHANDLE, unsigned char \*major, unsigned char \*minor)
- 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\\_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)
- 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](#), [CCStatus](#) mode)

- [EXTERN\\_C CCAUXDLL\\_API eErr CCAUXDLL\\_CALLING\\_CONV Video\\_getMirroring \(VIDEOHANDLE, CCStatus \\*mode\)](#)
- [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 \(VIDEOHANDLE, char \\*rawImgBuffer, unsigned long rawImgBufSize, bool bInterlaced\)](#)
- [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)

### 4.1.1 Typedef Documentation

4.1.1.1 typedef void\* [CrossControl::ABOUTHANDLE](#)

4.1.1.2 typedef void\* [CrossControl::ADCHANDLE](#)

4.1.1.3 typedef void\* [CrossControl::AUXVERSIONHANDLE](#)

4.1.1.4 typedef void\* [CrossControl::BACKLIGHTHANDLE](#)

4.1.1.5 typedef void\* [CrossControl::BUZZERHANDLE](#)

4.1.1.6 typedef void\* [CrossControl::CANSETTINGHANDLE](#)

4.1.1.7 typedef void\* [CrossControl::CONFIGHANDLE](#)

4.1.1.8 typedef void\* [CrossControl::DIAGNOSTICHANDLE](#)

4.1.1.9 typedef void\* [CrossControl::DIGIOHANDLE](#)

4.1.1.10 typedef void\* CrossControl::FIRMWAREUPGHANDLE

4.1.1.11 typedef void\* CrossControl::FRONTLEDHANDLE

4.1.1.12 typedef void\* CrossControl::LIGHTSENSORHANDLE

4.1.1.13 typedef void\* CrossControl::POWERHANDLE

4.1.1.14 typedef void\* CrossControl::TELEMATICSHANDLE

4.1.1.15 typedef void\* CrossControl::TOUCHSCREENCALIBHANDLE

4.1.1.16 typedef void\* CrossControl::TOUCHSCREENHANDLE

4.1.1.17 typedef struct version\_info CrossControl::VersionType

4.1.1.18 typedef void\* CrossControl::VIDEOHANDLE

## 4.1.2 Enumeration Type Documentation

### 4.1.2.1 enum CrossControl::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.

### 4.1.2.2 enum CrossControl::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.

#### 4.1.2.3 enum CrossControl::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.

#### 4.1.2.4 enum CrossControl::CanFrameType

Can frame type settings

Enumerator:

***FrameStandard***

***FrameExtended***

***FrameStandardExtended***

#### 4.1.2.5 enum CrossControl::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

#### 4.1.2.6 enum CrossControl::CCStatus

Enable/disable enumeration

Enumerator:

*Disabled*  
*Enabled* The setting is disabled or turned off

#### 4.1.2.7 enum CrossControl::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

#### 4.1.2.8 enum CrossControl::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

#### 4.1.2.9 enum CrossControl::hwErrorStatusCodes

The error codes returned by getHWErrorStatus.

Enumerator:

***errCodeNoErr***

#### 4.1.2.10 enum CrossControl::JidaSensorType

Jida temperature sensor types

Enumerator:

***TEMP\_CPU***

***TEMP\_BOX***

***TEMP\_ENV***



*TEMP\_BOARD*

*TEMP\_BACKPLANE*

*TEMP\_CHIPSETS*

*TEMP\_VIDEO*

*TEMP\_OTHER*

#### 4.1.2.11 enum CrossControl::LightSensorOperationRange

Light sensor operation ranges.

Enumerator:

*RangeStandard*

*RangeExtended* Light sensor operation range standard

#### 4.1.2.12 enum CrossControl::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.

#### 4.1.2.13 enum CrossControl::OCDDStatus

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

#### 4.1.2.14 enum CrossControl::PowerAction

Button and on/off signal actions.

Enumerator:

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

#### 4.1.2.15 enum CrossControl::shutDownReasonCodes

The shutdown codes returned by getShutDownReason.

Enumerator:

- shutdownReasonCodeNoError*

#### 4.1.2.16 enum CrossControl::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

#### 4.1.2.17 enum CrossControl::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.

#### 4.1.2.18 enum CrossControl::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

#### 4.1.2.19 enum CrossControl::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.

#### 4.1.2.20 enum CrossControl::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

#### 4.1.2.21 enum CrossControl::VideoChannel

The available analog video channels

Enumerator:

***Analog\_Channel\_1***

***Analog\_Channel\_2***

***Analog\_Channel\_3***

***Analog\_Channel\_4***

#### 4.1.2.22 enum CrossControl::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

## 4.1.2.23 enum CrossControl::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

## 4.1.3 Function Documentation

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

Get Add on hardware version.

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.

**Example Usage:**

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

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

Get Add on manufacturing date.

**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.

**Example Usage:**

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

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

Get Add on PCB article number.

**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.

**Example Usage:**

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

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

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

Get Add on PCB serial number.

##### 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.

##### Example Usage:

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

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

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

Get display resolution.

##### 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.

##### Example Usage:

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

4.1.3.6 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).

#### Parameters

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

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

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

Get Value for IO Expander

#### Parameters

<i>is</i>	Value for IO Expander.
-----------	------------------------

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.8 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::About\_getIsAnybusMounted ( ABOUTHANDLE , bool \*  
*mounted* )

Get Anybus mounting status.

#### Parameters

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

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

Example Usage:



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

Get BlueTooth module mounting status.

##### Parameters

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

##### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

##### Example Usage:

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

#### 4.1.3.10 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)

##### Parameters

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

##### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

##### Example Usage:

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

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

Get GPRS module mounting status.

##### Parameters

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**Example Usage:**

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

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

Get GPS module mounting status.

**Parameters**

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**Example Usage:**

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

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

Get IO Expander mounting status.

**Parameters**

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**Example Usage:**

**4.1.3.14 EXTERN.C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**  
**CrossControl::About\_getIsTouchScreenAvailable ( ABOUTHANDLE , bool \*  
*available* )**

Get Display TouchScreen status.

#### Parameters

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

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

#### Example Usage:

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

**4.1.3.15 EXTERN.C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**  
**CrossControl::About\_getIsWLANMounted ( ABOUTHANDLE , bool \*  
*mounted* )**

Get WLAN module mounting status.

#### Parameters

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

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

#### Example Usage:

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

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

Get main hardware version (PCB revision).

**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.

**Example Usage:**

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

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

Get main manufacturing date.

**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.

**Example Usage:**

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

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

Get main PCB article number.

**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.

**Example Usage:**

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

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

Get main PCB serial number.

**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.

**Example Usage:**

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

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

Get main product article number.

**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.

**Example Usage:**

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

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

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

Get main product revision.

##### 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.

##### Example Usage:

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

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

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

Get number of CAN connections present.

##### Parameters

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

##### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

##### Example Usage:

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

**4.1.3.23 EXTERN.C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::About\_getNrOfDigIOConnections ( ABOUTHANDLE ,  
unsigned char \* *NrOfConnections* )**

Get number of digital I/O connections present.

**Parameters**

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**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;
```

**4.1.3.24 EXTERN.C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::About\_getNrOfETHConnections ( ABOUTHANDLE ,  
unsigned char \* *NrOfConnections* )**

Get number of ethernet connections present.

**Parameters**

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**Example Usage:**

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

**4.1.3.25 EXTERN.C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::About\_getNrOfSerialConnections ( ABOUTHANDLE ,  
unsigned char \* *NrOfConnections* )**

Get number of serial port (RS232) connections present.

## Parameters

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

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

## Example Usage:

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

**4.1.3.26 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::About\_getNrOfUSBConnections ( ABOUTHANDLE ,  
unsigned char \* *NrOfConnections* )**

Get number of USB connections present.

## Parameters

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

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

## Example Usage:

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

**4.1.3.27 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::About\_getNrOfVideoConnections ( ABOUTHANDLE ,  
unsigned char \* *NrOfConnections* )**

Get number of Video connections present.

## Parameters

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



**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**Example Usage:**

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

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

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

Get unit serial number.

**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.

**Example Usage:**

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

**4.1.3.29 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.

**Parameters**

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**Example Usage:**

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

Delete the About object.

##### Returns

-

##### Example Usage:

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

list_about_information(pAbout);

About_release(pAbout);
```

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

Read measured voltage.

##### 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.

##### Example Usage:

```
err = Adc_getVoltage(pAdc, selection, &voltage);
if (CrossControl::ERR_SUCCESS == err)
{
    cout << left << setw(7) << description << ":" <<
        fixed << setprecision(2) << voltage << "V" << endl;
}
```

#### 4.1.3.32 EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV CrossControl::Adc\_release ( ADCHANDLE )

Delete the ADC object.

**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);

Adc_release(pAdc);

```

#### 4.1.3.33 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).

**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.

**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;

```

#### 4.1.3.34 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.

##### 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.

##### 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;

```

**4.1.3.35** 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

**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.

**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;
```

**4.1.3.36** 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

**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.

**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;
```

**4.1.3.37 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.

**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.

**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;
```

**4.1.3.38 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

#### 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.

#### 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;
```

**4.1.3.39 EXTERN.C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV**  
**CrossControl::AuxVersion\_release ( AUXVERSIONHANDLE )**

Delete the AuxVersion object.

#### Returns

-

#### Example Usage:

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

output_versions(pAuxVersion);

AuxVersion_release(pAuxVersion);
```

4.1.3.40 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.

**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.

4.1.3.41 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
**CrossControl::Backlight\_getAutomaticBLParams** ( BACKLIGHTHANDLE ,  
 bool \* *bSoftTransitions*, double \* *k* )

Get parameters for automatic backlight control.

**Parameters**

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.42 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
**CrossControl::Backlight\_getAutomaticBLStatus** ( BACKLIGHTHANDLE ,  
 unsigned char \* *status* )

Get status from automatic backlight control.

**Parameters**

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



**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**4.1.3.43 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::Backlight\_getIntensity ( BACKLIGHTHANDLE , unsigned char \*  
intensity )**

Get backlight intensity. Note that the lowest value returned is 3.

**Parameters**

<i>intensity</i>	The current backlight intensity (3..255).
------------------	---

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**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 getIntensity: %s\n", err, GetErrorStringA(err));
}
```

**4.1.3.44 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.

**Parameters**

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**4.1.3.45 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**  
**CrossControl::Backlight\_getStatus ( BACKLIGHTHANDLE , unsigned char \*  
*status* )**

Get backlight controller status.

#### 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.

#### 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
{
    printf("Error(%d) in function getStatus: %s\n", err, GetErrorStringA(err));
}
```

**4.1.3.46 EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV**  
**CrossControl::Backlight\_release ( BACKLIGHTHANDLE )**

Delete the backlight object.

#### Returns

-

#### Example Usage:

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

change_backlight(pBacklight);

Backlight_release(pBacklight);
```

4.1.3.47 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.

#### 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.

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

Set parameters for automatic backlight control.

#### Parameters

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

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.49 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Backlight\_setIntensity ( BACKLIGHTHANDLE , unsigned char  
*intensity* )

Set backlight intensity. Note that setting a lower value than 3 actually sets the value 3.  
 This is a hardware design limit.

#### Parameters

<i>intensity</i>	The backlight intensity to set (3..255).
------------------	--

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**Example Usage:**

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

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

**4.1.3.50 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.

**Parameters**

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**4.1.3.51 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.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

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

Stop automatic backlight control.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**4.1.3.53 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Buzzer\_buzze ( BUZZERHANDLE , int *time*, bool *blocking* )**

Buzzes for a specified time.

**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.

**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;
    }
}
```

**4.1.3.54 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Buzzer\_getFrequency ( BUZZERHANDLE , unsigned short \*  
*frequency* )**

Get buzzer frequency.

**Parameters**

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**4.1.3.55 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.

**Parameters**

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**4.1.3.56 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::Buzzer\_getVolume ( BUZZERHANDLE , unsigned short \* *volume*  
)**

Get buzzer volume.

**Parameters**

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**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;
}
```

**4.1.3.57 EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV  
CrossControl::Buzzer\_release ( BUZZERHANDLE )**

Delete the Buzzer object.

**Returns**

-

**Example Usage:**

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

play_beeeps (pBuzzer);

Buzzer_release (pBuzzer);
```

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

Set buzzer frequency.

**Parameters**

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**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;
    }
}
```

**4.1.3.59 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.

**Parameters**

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

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

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

Set buzzer volume.

**Parameters**

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**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;
}
```

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

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

Get Baud rate

**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.

**Example Usage:**

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



```

    GetErrorStringA(err) << endl;
    break;
}

```

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

Get frame type

#### 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.

Example Usage:

```

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

```

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

Delete the CanSetting object.

#### Returns

-

Example Usage:

```

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

read_cansettings(pCanSetting);

CanSetting_release(pCanSetting);

```

**4.1.3.64 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.

## 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.

4.1.3.65 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.

## Parameters

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

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.66 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.

## Parameters

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**4.1.3.67** 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.

**Parameters**

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

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

Get external on/off signal trigger time.

**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.

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

Get front button trigger time for long press.

**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.

**4.1.3.70 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.

**Parameters**

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**4.1.3.71 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.

**Parameters**

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**4.1.3.72 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.

**Parameters**

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.73 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.

**Parameters**

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.74 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.

**Parameters**

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.75 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?

**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.

**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;
}
}
```

**4.1.3.76 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.

**Parameters**

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

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

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

Get suspend mode maximum time.

**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.

4.1.3.78 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Config\_getTFTMirror ( CONFIGHANDLE , bool \* *enable* )

Get TFT Mirror configuration. Will mirror display image when enabled.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.79 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Config\_getTFTMode ( CONFIGHANDLE , bool \* *enable* )

Get TFT Mode configuration. Will select 18/24-bit interface. Should be low for both.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.80 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Config\_getTFTScan ( CONFIGHANDLE , bool \* *enable* )

Get TFT Scan configuration. For 10" display will rotate picture 180deg and on 7" display will flip picture up/down.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.81 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.

**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.

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

Delete the Config object.

**Returns**

-

**Example Usage:**

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

conf_example(pConfig);

Config_release(pConfig);
```

**4.1.3.83 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.

**Parameters**

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**4.1.3.84 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.



## Parameters

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

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

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

Set external on/off signal trigger time.

## 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.

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

Set front button trigger time for long press.

## 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.

**4.1.3.87** 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.

#### Parameters

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

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

**4.1.3.88** 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.

#### Parameters

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

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

**4.1.3.89** 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.

#### Parameters

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**4.1.3.90** 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.

**Parameters**

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**4.1.3.91** 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.

**Parameters**

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**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;
}
```

4.1.3.92 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?

#### 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.

4.1.3.93 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.

#### Parameters

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

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

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

Set suspend mode maximum time.

#### 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.

4.1.3.95 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Config\_setTFTMirror ( CONFIGHANDLE , bool *enable* )

Set TFT Mirror configuration. Will mirror display image when enabled.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.96 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Config\_setTFTMode ( CONFIGHANDLE , bool *enable* )

Set TFT Mode configuration. Will select 18/24-bit interface. Should be low for both.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.97 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Config\_setTFTScan ( CONFIGHANDLE , bool *enable* )

Set TFT Scan configuration. For 10" display will rotate picture 180deg and on 7" display will flip picture up/down.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.98 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.

**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.

4.1.3.99 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)

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.100 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.

#### Parameters

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

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.101 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.

#### Parameters

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

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

#### Example Usage:

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

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

Get PCB temperature.

Parameters

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

Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.103 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.
<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.

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

Get number of power cycles.

Parameters

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

Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

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

Get shutdown reason.

**Parameters**

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

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

Get System Supervisor temperature.

**Parameters**

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

Example Usage:

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

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

Get startup reason.

**Parameters**

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.



**4.1.3.108 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**  
**CrossControl::Diagnostic\_getTimer ( DIAGNOSTICHANDLE , TimerType \*  
*times* )**

Get diagnostic timer.

#### Parameters

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

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

#### 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);
```

**4.1.3.109 EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV**  
**CrossControl::Diagnostic\_release ( DIAGNOSTICHANDLE )**

Delete the Diagnostic object.

#### Returns

-

#### Example Usage:

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

diagnostic_example(pDiagnostic);

Diagnostic_release(pDiagnostic);
```

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

Get Digital inputs. Supported platform(s): XA, XM.

#### 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
Generated on Mon Jan 28 2013 14:16:09 for CCAux by Doxygen	

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**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;
}
```

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

Delete the DigIO object.

**Returns**

-

**Example Usage:**

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

list_digital_inputs(pDigIO);

DigIO_release(pDigIO);
```

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

Set Digital outputs. Supported platform(s): XA.

**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.

**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;
}
```

**4.1.3.113 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.

**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.

**4.1.3.114 EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV  
CrossControl::FirmwareUpgrade\_release ( FIRMWAREUPGHANDLE )**

Delete the FirmwareUpgrade object.

**Returns**

-

**Example Usage:**

```
FirmwareUpgrade_release (pFirmwareUpgrade);
```

**4.1.3.115 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::FirmwareUpgrade\_shutDown ( FIRMWAREUPGHANDLE )**

Shut down the operating system.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**4.1.3.116 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.

**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.

**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;
        }
    }
}

```

```

    }
}

```

#### 4.1.3.117 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.

##### 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.

##### 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;
        }
    }
}
}

```

4.1.3.118 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.

#### 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.

#### 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;
        }
    }
}
```

#### 4.1.3.119 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.

##### 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.

##### 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;
        }
    }
}
}

```

**4.1.3.120 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.

**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.

**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;
        }
    }
}

```



4.1.3.121 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.

**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.

**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;
        }
    }
}
```

4.1.3.122 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.

#### 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.

#### Example Usage:

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

4.1.3.123 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.

#### Parameters

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

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.124 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
**CrossControl::FrontLED\_getFrontPcbRev** ( FRONTLEDHANDLE , unsigned  
char \* *major*, unsigned char \* *minor* )

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

#### Parameters

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

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

Get front LED idle time.

**Parameters**

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

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

Get number of pulses during a blink sequence.

**Parameters**

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

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

Get front LED off time.

**Parameters**

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

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

Get front LED on time.

#### Parameters

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

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.129 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 set-Signal. This is due to precision-loss in approximations.

#### 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.

#### Example Usage:

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

4.1.3.130 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.

#### Parameters

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**4.1.3.131 EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV  
CrossControl::FrontLED\_release ( FRONTLEDHANDLE )**

Delete the FrontLED object.

**Returns**

-

**Example Usage:**

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

led_example(pFrontLED);

FrontLED_release(pFrontLED);
```

**4.1.3.132 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.

**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.

**Example Usage:**

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

4.1.3.133 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.

Parameters

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

Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

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

Get front LED idle time.

Parameters

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

Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.135 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.

Parameters

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

Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

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

Set front LED off.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

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

Set front LED off time.

**Parameters**

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**Example Usage:**

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

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

Set front LED on time.

**Parameters**

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**Example Usage:**

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

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

Set front LED signal.

**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.

**Example Usage:**

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

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

Set one of the front LED standard colors.

**Parameters**

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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**Example Usage:**

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

**4.1.3.141** EXTERN\_C CCAUXDLL\_API ABOUTHANDLE CCAUXDLL\_CALLING\_CONV  
**CrossControl::GetAbout** ( void )

Factory function that creates instances of the About object.



**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);
```

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

Factory function that creates instances of the Adc object.

**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);

Adc_release(pAdc);
```

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

Factory function that creates instances of the AuxVersion object.

**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);
```

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

Factory function that creates instances of the Backlight object.

**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);
```

**4.1.3.145 EXTERN\_C CCAUXDLL\_API BUZZERHANDLE CCAUXDLL\_CALLING\_CONV  
CrossControl::GetBuzzer ( void )**

Factory function that creates instances of the Buzzer object.

**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);
```

**4.1.3.146 EXTERN\_C CCAUXDLL\_API CANSETTINGHANDLE  
CCAUXDLL\_CALLING\_CONV CrossControl::GetCanSetting ( void )**

Factory function that creates instances of the CanSetting object.

**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);
```

**4.1.3.147 EXTERN\_C CCAUXDLL\_API CONFIGHANDLE CCAUXDLL\_CALLING\_CONV  
CrossControl::GetConfig ( )**

Video channel 4 config

Factory function that creates instances of the Config object.

**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);
```

**4.1.3.148 EXTERN\_C CCAUXDLL\_API DIAGNOSTICHANDLE  
CCAUXDLL\_CALLING\_CONV CrossControl::GetDiagnostic ( void )**

Factory function that creates instances of the Diagnostic object.

**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);
```

#### 4.1.3.149 EXTERN\_C CCAUXDLL\_API DIGIOHANDLE CCAUXDLL\_CALLING\_CONV CrossControl::GetDigIO ( void )

Factory function that creates instances of the DigIO object.

**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);
```

#### 4.1.3.150 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.

**4.1.3.151 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.

**4.1.3.152 EXTERN\_C CCAUXDLL\_API FIRMWAREUPGHANDLE  
CCAUXDLL\_CALLING\_CONV CrossControl::GetFirmwareUpgrade ( void )**

Factory function that creates instances of the Adc object.

**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);
```

**4.1.3.153 EXTERN\_C CCAUXDLL\_API FRONTLEDHANDLE CCAUXDLL\_CALLING\_CONV  
CrossControl::GetFrontLED ( void )**

Factory function that creates instances of the FrontLED object.

**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);
```

4.1.3.154 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.

4.1.3.155 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.

4.1.3.156 EXTERN\_C CCAUXDLL\_API LIGHTSENSORHANDLE  
CCAUXDLL\_CALLING\_CONV CrossControl::GetLightsensor ( void )

Factory function that creates instances of the Lightsensor object.

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);
```

**4.1.3.157** `EXTERN_C CCAUXDLL_API POWERHANDLE CCAUXDLL_CALLING_CONV  
CrossControl::GetPower ( void )`

Factory function that creates instances of the Power object.

**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);
```

**4.1.3.158** `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.

**4.1.3.159** `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.

#### 4.1.3.160 EXTERN\_C CCAUXDLL\_API TELEMATICSHANDLE CCAUXDLL\_CALLING\_CONV CrossControl::GetTelematics ( void )

Factory function that creates instances of the Telematics object.

##### 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);
```

#### 4.1.3.161 EXTERN\_C CCAUXDLL\_API TOUCHSCREENHANDLE CCAUXDLL\_CALLING\_CONV CrossControl::GetTouchScreen ( void )

Factory function that creates instances of the TouchScreen object.

##### 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);
```

#### 4.1.3.162 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.



**4.1.3.163** EXTERN\_C CCAUXDLL\_API VIDEOHANDLE CCAUXDLL\_CALLING\_CONV  
**CrossControl::GetVideo** ( void )

Factory function that creates instances of the Video object.

**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.

**4.1.3.164** EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
**CrossControl::Lightsensor\_getAverageIlluminance** ( LIGHTSENSORHANDLE , unsigned short \* *value* )

Get average illuminance (light) value from light sensor.

**Parameters**

<i>value</i>	Illuminance value (Lux).
--------------	--------------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**Example Usage:**

```
err = Lightsensor_getAverageIlluminance(pLightSensor, &value);
if (err != ERR_SUCCESS)
{
    cout << "Error(" << err << ") in function getAverageIlluminance: " <<
        GetErrorStringA(err) << endl;
}
```

**4.1.3.165** EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
**CrossControl::Lightsensor\_getIlluminance** ( LIGHTSENSORHANDLE , unsigned short \* *value* )

Get illuminance (light) value from light sensor.

**Parameters**

<i>value</i>	Illuminace value (Lux).
--------------	-------------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**Example Usage:**

```
err = Lightsensor_getIlluminance(pLightSensor, &value);
if(err != ERR_SUCCESS)
{
    cout << "Error(" << err << ") in function getIlluminance: " <<
        GetErrorStringA(err) << endl;
}
```

**4.1.3.166 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.

**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.

**4.1.3.167 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.

**Parameters**

<i>range</i>	Operating range. RangeStandard or RangeExtended.
--------------	--

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**4.1.3.168 EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV  
CrossControl::Lightsensor\_release ( LIGHTSENSORHANDLE )**

Delete the Lightsensor object.

**Returns**

-

**Example Usage:**

```

LIGHTSENSORHANDLE pLightSensor = ::GetLightsensor();
assert (pLightSensor);

ls_example (pLightSensor);

Lightsensor_release (pLightSensor);

```

**4.1.3.169 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.

**Parameters**

<i>range</i>	Operating range to set. RangeStandard or RangeExtended.
--------------	---

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**4.1.3.170 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.

**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.

**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;
}
```

**4.1.3.171 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::Lightsensor\_stopAverageCalc ( LIGHTSENSORHANDLE )**

Stop average calculation.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**Example Usage:**

```
err = Lightsensor_stopAverageCalc(pLightSensor);
if(err != ERR_SUCCESS)
{
    cout << "Error(" << err << ") in function stopAverageCalc: " <<
        GetErrorStringA(err) << endl;
}
```

**4.1.3.172 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

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.173 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
**CrossControl::Power\_getBLPowerStatus** ( POWERHANDLE , CCStatus \*  
*status* )

Get backlight power status.

#### Parameters

<i>status</i>	Backlight power status.
---------------	-------------------------

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

#### 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;
}
```

4.1.3.174 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.

#### Parameters

<i>status</i>	The current status. See the definition of ButtonPowerTransitionStatus for details.
---------------	--

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.175 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.

**Parameters**

<i>status</i>	The current overcurrent detection status
---------------	--

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**Example Usage:**

```

cout << "Checking overcurrent status... " << endl;
OCDStatus ocdstatus;
err = Power_getCanOCDStatus(pPower, &ocdstatus);
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;
    }
}

```

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

**CrossControl::Power\_getCanPowerStatus ( POWERHANDLE , CCStatus \*  
status )**

Get can power status.

**Parameters**

<i>status</i>	Can power status.
---------------	-------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.177 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
**CrossControl::Power\_getExtFanPowerStatus** ( POWERHANDLE , CCStatus \* *status* )

Get external fan power status.

**Parameters**

<i>status</i>	Fan power status.
---------------	-------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.178 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.

**Parameters**

<i>status</i>	The current overcurrent detection status
---------------	--

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**Example Usage:**

```
err = Power_getVideoOCDStatus(pPower, &ocdstatus);
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;
  }
}

```

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

**CrossControl::Power\_getVideoPowerStatus ( POWERHANDLE , unsigned char \* videoStatus )**

Get Video power status.

##### 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.

##### 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;
}

```

#### 4.1.3.180 EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV

**CrossControl::Power\_release ( POWERHANDLE )**

Delete the Power object.

##### Returns

-

##### Example Usage:

```

POWERHANDLE pPower = ::GetPower();
assert(pPower);

```



```
power_example(pPower);
Power_release(pPower);
```

#### 4.1.3.181 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::Power\_setBLPowerStatus ( POWERHANDLE , CCStatus status )

Set backlight power status.

##### Parameters

<i>status</i>	Backlight power status.
---------------	-------------------------

##### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

##### 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;
}
```

#### 4.1.3.182 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::Power\_setCanPowerStatus ( POWERHANDLE , CCStatus status )

Set can power status.

##### Parameters

<i>status</i>	Can power status.
---------------	-------------------

##### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.183 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
**CrossControl::Power\_setExtFanPowerStatus ( POWERHANDLE , CCStatus  
*status* )**

Set external fan power status.

**Parameters**

<i>status</i>	Fan power status.
---------------	-------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.184 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
**CrossControl::Power\_setVideoPowerStatus ( POWERHANDLE , unsigned  
char *status* )**

Set Video power status.

**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.

4.1.3.185 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
**CrossControl::Telematics\_getBTPowerStatus ( TELEMATICSHANDLE ,  
CCStatus \* *status* )**

Get Bluetooth power status.

**Parameters**

<i>status</i>	Bluetooth power status.
---------------	-------------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**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;
    }
}

```

#### 4.1.3.186 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.

##### Parameters

<i>status</i>	Bluetooth power status.
---------------	-------------------------

##### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

##### 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;
}
}

```

#### 4.1.3.187 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::Telematics\_getGPRSPowerStatus ( TELEMATICSHANDLE , CCStatus \* status )

Get GPRS power status.

**Parameters**

<i>status</i>	GPRS power status.
---------------	--------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**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;
}
}
```

#### 4.1.3.188 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV CrossControl::Telematics\_getGPRSStartupPowerStatus (TELEMATICSHANDLE , CCStatus \* status )

Get GPRS power status at startup and at resume from suspended mode.

**Parameters**

<i>status</i>	GPRS power status.
---------------	--------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**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;
}

```

#### 4.1.3.189 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.

##### Parameters

<i>status</i>	GPS antenna power status.
---------------	---------------------------

##### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

##### 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;
}

```

#### 4.1.3.190 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.

##### Parameters

<i>status</i>	GPS power status.
---------------	-------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**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;
}
```

#### 4.1.3.191 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.

**Parameters**

<i>status</i>	GPS power status.
---------------	-------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**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;
}
```

**4.1.3.192 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**  
**CrossControl::Telematics\_getTelematicsAvailable ( TELEMATICSHANDLE ,**  
**CCStatus \* status )**

Is a telematics add-on card installed?

#### Parameters

<i>status</i>	Enabled if a telematics add-on card is installed, otherwise Disabled.
---------------	---

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

#### 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;
}
```

**4.1.3.193 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV**  
**CrossControl::Telematics\_getWLANPowerStatus ( TELEMATICSHANDLE ,**  
**CCStatus \* status )**

Get WLAN power status.

#### Parameters

<i>status</i>	WLAN power status.
---------------	--------------------

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

#### 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;
}

```

#### 4.1.3.194 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.

##### Parameters

<i>status</i>	WLAN power status.
---------------	--------------------

##### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

##### 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;
}

```

#### 4.1.3.195 EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV CrossControl::Telematics\_release ( TELEMATICSHANDLE )

Delete the Telematics object.

##### Returns

-

##### Example Usage:



```

TELEMATICSHANDLE pTelematics = ::GetTelematics();
assert(pTelematics);

telematics_example(pTelematics);

Telematics_release(pTelematics);

```

**4.1.3.196 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::Telematics\_setBTPowerStatus ( TELEMATICSHANDLE ,  
CCStatus *status* )**

Set Bluetooth power status.

**Parameters**

<i>status</i>	Bluetooth power status.
---------------	-------------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**4.1.3.197 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.

**Parameters**

<i>status</i>	Bluetooth power status.
---------------	-------------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**4.1.3.198 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::Telematics\_setGPRSPowerStatus ( TELEMATICSHANDLE ,  
CCStatus *status* )**

Set GPRS modem power status.

**Parameters**

<i>status</i>	GPRS modem power status.
---------------	--------------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.199 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.

**Parameters**

<i>status</i>	GPRS power status.
---------------	--------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.200 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::Telematics\_setGPSPowerStatus ( TELEMATICSHANDLE ,   
CCStatus *status* )

Set GPS power status.

**Parameters**

<i>status</i>	GPS power status.
---------------	-------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.201 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.

**Parameters**

<i>status</i>	GPS power status.
---------------	-------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.202 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Telematics\_setWLANPowerStatus ( TELEMATICSHANDLE ,  
 CCStatus *status* )

Set WLAN power status.

Parameters

<i>status</i>	WLAN power status.
---------------	--------------------

Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.203 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.

Parameters

<i>status</i>	WLAN power status.
---------------	--------------------

Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.204 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.

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.

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;
}

```

#### 4.1.3.205 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.

##### Parameters

<i>config</i>	The current mode.
---------------	-------------------

##### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

##### 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;
}

```

**4.1.3.206** 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.

#### Parameters

<i>time</i>	The right click time, in milliseconds.
-------------	--

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

#### 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;
}
```

**4.1.3.207** EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV  
**CrossControl::TouchScreen\_release** ( TOUCHSCREENHANDLE )

Delete the TouchScreen object.

#### Returns

-

#### Example Usage:

```
TOUCHSCREENHANDLE pTouchScreen = ::GetTouchScreen();
assert(pTouchScreen);

touchscreen_example(pTouchScreen);

TouchScreen_release(pTouchScreen);
```

**4.1.3.208** 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 TSAAdvancedSettingsParameter for a description of the parameters.

## 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.

4.1.3.209 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.

## Parameters

<i>config</i>	The mode to set.
---------------	------------------

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.210 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.

## Parameters

<i>time</i>	The right click time, in milliseconds.
-------------	--

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.211 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.

4.1.3.212 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.

4.1.3.213 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.

4.1.3.214 EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV  
 CrossControl::TouchScreenCalib\_release ( TOUCHSCREENCALIBHANDLE )

Delete the TouchScreenCalib object.

## Returns

-

4.1.3.215 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.

4.1.3.216 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.

4.1.3.217 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.

**4.1.3.218 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.

**Parameters**

<i>activate</i>	Set to true if the snapshot function shall be active.
-----------------	---

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**4.1.3.219 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.

**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.

4.1.3.220 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Video\_freeBmpBuffer ( VIDEOHANDLE , char \* *bmpBuffer* )

Free the memory allocated for BMP buffer.

Parameters

<i>bmpBuffer</i>	The bmp buffer to free.
------------------	-------------------------

Returns

error status.

4.1.3.221 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Video\_getActiveChannel ( VIDEOHANDLE , VideoChannel \* *channel* )

Get the current video channel.

Parameters

<i>channel</i>	Enum defining available channels.
----------------	-----------------------------------

Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.222 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.

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.

**4.1.3.223** 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.

**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.

**4.1.3.224** 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.

**Parameters**

<i>decoder-Register</i>	Decoder Register Address.
<i>register-Value</i>	register value.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**4.1.3.225** 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.

## Parameters

<i>mode</i>	The current mode. See enum DeInterlaceMode for descriptions of the modes.
-------------	---

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

**4.1.3.226 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::Video\_getMirroring ( VIDEOHANDLE , CCStatus \* mode )**

Get the current mirroring mode of the video image.

## Parameters

<i>mode</i>	The current mode. Enabled or Disabled.
-------------	--

## Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

**4.1.3.227 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.

## 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.

**4.1.3.228 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.

## 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.

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

CrossControl::Video\_getStatus ( VIDEOHANDLE , unsigned char \* *status* )

Video status byte.

## 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.

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

CrossControl::Video\_getVideoArea ( VIDEOHANDLE , unsigned short \* *topLeftX*, unsigned short \* *topLeftY*, unsigned short \* *bottomRigthX*, unsigned short \* *bottomRigthY* )

Get the area where video is shown.

## 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.

4.1.3.231 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.

Parameters

<i>standard</i>	Video standard.
-----------------	-----------------

Returns

error status.

4.1.3.232 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.

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.

4.1.3.233 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
CrossControl::Video\_minimize ( VIDEOHANDLE )

Minimizes the video area. Restore with restore() call.

Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.234 EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV  
CrossControl::Video\_release ( VIDEOHANDLE )

Delete the Video object.

Returns

-

4.1.3.235 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.

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.236 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Video\_setActiveChannel ( VIDEOHANDLE , VideoChannel  
*channel* )

Sets the active video channel.

#### Parameters

<i>channel</i>	Enum defining available channels.
----------------	-----------------------------------

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.237 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.

#### 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.

4.1.3.238 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 param-

ters 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.

**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.

**4.1.3.239** 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.

**Parameters**

<i>decoder-Register</i>	Decoder Register Address.
<i>register-Value</i>	register value.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**4.1.3.240** 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.

**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.



4.1.3.241 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Video\_setMirroring ( VIDEOHANDLE , CCStatus *mode* )

Enable or disable mirroring of the video image.

#### Parameters

<i>mode</i>	The mode to set. Enabled or Disabled.
-------------	---------------------------------------

#### Returns

error status. 0 = ERR\_SUCCESS, otherwise error code.

4.1.3.242 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.

#### 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.

4.1.3.243 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.

#### 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.

**4.1.3.244 EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV  
 CrossControl::Video\_showFrame ( VIDEOHANDLE )**

Copy one frame from camera to the display.

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**4.1.3.245 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.

**Parameters**

<i>show</i>	True shows the video image.
-------------	-----------------------------

**Returns**

error status. 0 = ERR\_SUCCESS, otherwise error code.

**4.1.3.246 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.

**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.

4.1.3.247 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.

#### 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.

4.1.3.248 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.

#### 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.

**4.1.4 Variable Documentation**

4.1.4.1 `const unsigned char CrossControl::DigitalIn_1 = (1 << 0)`

Bit defines for getDigIO

4.1.4.2 `const unsigned char CrossControl::DigitalIn_2 = (1 << 1)`

4.1.4.3 `const unsigned char CrossControl::DigitalIn_3 = (1 << 2)`

4.1.4.4 `const unsigned char CrossControl::DigitalIn_4 = (1 << 3)`

4.1.4.5 `const unsigned char CrossControl::Video1Conf = (1 << 0)`

Bit defines for getVideoStartupPowerConfig and setVideoStartupPowerConfig

4.1.4.6 `const unsigned char CrossControl::Video2Conf = (1 << 1)`

Video channel 1 config

4.1.4.7 `const unsigned char CrossControl::Video3Conf = (1 << 2)`

Video channel 2 config

4.1.4.8 `const unsigned char CrossControl::Video4Conf = (1 << 3)`

Video channel 3 config

## Chapter 5

# Class Documentation

### 5.1 CrossControl::BuzzerSetup Struct Reference

```
#include <CCAuxTypes.h>
```

#### Public Attributes

- unsigned short [frequency](#)
- unsigned short [volume](#)

#### 5.1.1 Member Data Documentation

##### 5.1.1.1 unsigned short CrossControl::BuzzerSetup::frequency

buzzer frequency

##### 5.1.1.2 unsigned short CrossControl::BuzzerSetup::volume

buzzer volume

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

- [fixedIncludeFiles/CCAuxTypes.h](#)

### 5.2 CrossControl::FpgaLedTimingType Struct Reference

```
#include <CCAuxTypes.h>
```

### Public Attributes

- unsigned char [ledNbr](#)
- unsigned char [onTime](#)
- unsigned char [offTime](#)
- unsigned char [idleTime](#)
- unsigned char [nrOfPulses](#)

### 5.2.1 Member Data Documentation

#### 5.2.1.1 unsigned char CrossControl::FpgaLedTimingType::idleTime

LED idle time in 100ms

#### 5.2.1.2 unsigned char CrossControl::FpgaLedTimingType::ledNbr

Number of LED

#### 5.2.1.3 unsigned char CrossControl::FpgaLedTimingType::nrOfPulses

Pulses per sequences

#### 5.2.1.4 unsigned char CrossControl::FpgaLedTimingType::offTime

LED off time in 10ms

#### 5.2.1.5 unsigned char CrossControl::FpgaLedTimingType::onTime

LED on time in 10ms

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

- [fixedIncludeFiles/CCAuxTypes.h](#)

### 5.3 CrossControl::LedColorMixType Struct Reference

```
#include <CCAuxTypes.h>
```

### Public Attributes

- unsigned char [red](#)
- unsigned char [green](#)
- unsigned char [blue](#)

### 5.3.1 Member Data Documentation

#### 5.3.1.1 unsigned char CrossControl::LedColorMixType::blue

Blue color intensity 0-0x0F

#### 5.3.1.2 unsigned char CrossControl::LedColorMixType::green

Green color intensity 0-0x0F

#### 5.3.1.3 unsigned char CrossControl::LedColorMixType::red

Red color intensity 0-0x0F

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

- [fixedIncludeFiles/CCAuxTypes.h](#)

## 5.4 CrossControl::LedTimingType Struct Reference

```
#include <CCAuxTypes.h>
```

### Public Attributes

- unsigned char [onTime](#)
- unsigned char [offTime](#)
- unsigned char [idleTime](#)
- unsigned char [nrOfPulses](#)

### 5.4.1 Member Data Documentation

#### 5.4.1.1 unsigned char CrossControl::LedTimingType::idleTime

LED idle time in 100ms

#### 5.4.1.2 unsigned char CrossControl::LedTimingType::nrOfPulses

Pulses per sequences

#### 5.4.1.3 unsigned char CrossControl::LedTimingType::offTime

LED off time in 10ms

#### 5.4.1.4 unsigned char CrossControl::LedTimingType::onTime

LED on time in 10ms

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

- [fixedIncludeFiles/CCAuxTypes.h](#)

## 5.5 CrossControl::received\_video Struct Reference

```
#include <CCAuxTypes.h>
```

### Public Attributes

- unsigned short [received\\_width](#)
- unsigned short [received\\_height](#)
- unsigned char [received\\_framerate](#)

### 5.5.1 Member Data Documentation

#### 5.5.1.1 unsigned char CrossControl::received\_video::received\_framerate

#### 5.5.1.2 unsigned short CrossControl::received\_video::received\_height

#### 5.5.1.3 unsigned short CrossControl::received\_video::received\_width

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

- [fixedIncludeFiles/CCAuxTypes.h](#)

## 5.6 CrossControl::TimerType Struct Reference

```
#include <CCAuxTypes.h>
```

### Public Attributes

- unsigned long [TotRunTime](#)
- unsigned long [TotSuspTime](#)
- unsigned long [TotHeatTime](#)
- unsigned long [RunTime40\\_60](#)
- unsigned long [RunTime60\\_70](#)
- unsigned long [RunTime70\\_80](#)
- unsigned long [Above80RunTime](#)



### 5.6.1 Detailed Description

Diagnostic timer data

### 5.6.2 Member Data Documentation

#### 5.6.2.1 unsigned long CrossControl::TimerType::Above80RunTime

Total runtime in 70-80deg (minutes)

#### 5.6.2.2 unsigned long CrossControl::TimerType::RunTime40\_60

Total heating time (minutes)

#### 5.6.2.3 unsigned long CrossControl::TimerType::RunTime60\_70

Total runtime in 40-60deg (minutes)

#### 5.6.2.4 unsigned long CrossControl::TimerType::RunTime70\_80

Total runtime in 60-70deg (minutes)

#### 5.6.2.5 unsigned long CrossControl::TimerType::TotHeatTime

Total suspend time (minutes)

#### 5.6.2.6 unsigned long CrossControl::TimerType::TotRunTime

#### 5.6.2.7 unsigned long CrossControl::TimerType::TotSuspTime

Total running time (minutes)

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

- [fixedIncludeFiles/CCAuxTypes.h](#)

## 5.7 CrossControl::UpgradeStatus Struct Reference

```
#include <CCAuxTypes.h>
```

### Public Attributes

- enum [UpgradeAction](#) `currentAction`

- unsigned char [percent](#)
- [eErr](#) [errorCode](#)

### 5.7.1 Detailed Description

Upgrade Status

### 5.7.2 Member Data Documentation

5.7.2.1 enum [UpgradeAction](#) [CrossControl::UpgradeStatus::currentAction](#)

5.7.2.2 [eErr](#) [CrossControl::UpgradeStatus::errorCode](#)

Represents the percentage of completion of the current action

5.7.2.3 unsigned char [CrossControl::UpgradeStatus::percent](#)

The current action.

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

- [fixedIncludeFiles/CCAuxTypes.h](#)

## 5.8 CrossControl::version\_info Struct Reference

```
#include <CCAuxTypes.h>
```

### Public Attributes

- unsigned char [major](#)
- unsigned char [minor](#)
- unsigned char [release](#)
- unsigned char [build](#)

### 5.8.1 Member Data Documentation

5.8.1.1 unsigned char [CrossControl::version\\_info::build](#)

version build number

5.8.1.2 unsigned char [CrossControl::version\\_info::major](#)

version major number

### 5.8.1.3 unsigned char CrossControl::version\_info::minor

version minor number

### 5.8.1.4 unsigned char CrossControl::version\_info::release

version release number

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

- [fixedIncludeFiles/CCAuxTypes.h](#)

## 5.9 CrossControl::video\_dec\_command Struct Reference

```
#include <CCAuxTypes.h>
```

### Public Attributes

- unsigned char [decoder\\_register](#)
- unsigned char [register\\_value](#)

### 5.9.1 Member Data Documentation

#### 5.9.1.1 unsigned char CrossControl::video\_dec\_command::decoder\_register

#### 5.9.1.2 unsigned char CrossControl::video\_dec\_command::register\_value

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

- [fixedIncludeFiles/CCAuxTypes.h](#)

# Chapter 6

## File Documentation

### 6.1 fixedIncludeFiles/About.h File Reference

#### Namespaces

- namespace [CrossControl](#)

#### Typedefs

- typedef void \* [CrossControl::ABOUTHANDLE](#)

#### Functions

- EXTERN\_C CCAUXDLL\_API ABOUTHANDLE CCAUXDLL\_CALLING\_CONV [CrossControl::GetAbout](#) (void)
- EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV [CrossControl::About\\_release](#) (ABOUTHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::About\\_getMainPCBSerial](#) (ABOUTHANDLE, char \*buff, int len)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::About\\_getUnitSerial](#) (ABOUTHANDLE, char \*buff, int len)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::About\\_getMainPCBArt](#) (ABOUTHANDLE, char \*buff, int length)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::About\\_getMainManufacturingDate](#) (ABOUTHANDLE, char \*buff, int len)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::About\\_getMainHWversion](#) (ABOUTHANDLE, char \*buff, int len)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::About\\_getMainProdRev](#) (ABOUTHANDLE, char \*buff, int len)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::About\\_getMainProdArtNr](#) (ABOUTHANDLE, char \*buff, int len)

- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::About\\_getNrOfETHConnections](#) (ABOUTHANDLE, unsigned char \*NrOfConnections)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::About\\_getNrOfCANConnections](#) (ABOUTHANDLE, unsigned char \*NrOfConnections)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::About\\_getNrOfVideoConnections](#) (ABOUTHANDLE, unsigned char \*NrOfConnections)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::About\\_getNrOfUSBConnections](#) (ABOUTHANDLE, unsigned char \*NrOfConnections)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::About\\_getNrOfSerialConnections](#) (ABOUTHANDLE, unsigned char \*NrOfConnections)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::About\\_getNrOfDigIOConnections](#) (ABOUTHANDLE, unsigned char \*NrOfConnections)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::About\\_getIsDisplayAvailable](#) (ABOUTHANDLE, bool \*available)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::About\\_getIsTouchScreenAvailable](#) (ABOUTHANDLE, bool \*available)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::About\\_getDisplayResolution](#) (ABOUTHANDLE, char \*buff, int len)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::About\\_getAddOnPCBSerial](#) (ABOUTHANDLE, char \*buff, int len)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::About\\_getAddOnPCB Art](#) (ABOUTHANDLE, char \*buff, int length)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::About\\_getAddOnManufacturingDate](#) (ABOUTHANDLE, char \*buff, int len)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::About\\_getAddOnHWversion](#) (ABOUTHANDLE, char \*buff, int len)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::About\\_getIsWLANMounted](#) (ABOUTHANDLE, bool \*mounted)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::About\\_getIsGPSPMounted](#) (ABOUTHANDLE, bool \*mounted)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::About\\_getIsGPRSMounted](#) (ABOUTHANDLE, bool \*mounted)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::About\\_getIsBTMounted](#) (ABOUTHANDLE, bool \*mounted)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::About\\_getFrontPcbRev](#) (ABOUTHANDLE, unsigned char \*major, unsigned char \*minor)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::About\\_getIsIOExpanderMounted](#) (ABOUTHANDLE, bool \*mounted)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::About\\_getIOExpanderValue](#) (ABOUTHANDLE, unsigned short \*value)

- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::About\\_hasOsBooted](#) (ABOUTHANDLE, bool \*bootComplete)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::About\\_getIsAnybusMounted](#) (ABOUTHANDLE, bool \*mounted)

## 6.2 fixedIncludeFiles/Adc.h File Reference

### Namespaces

- namespace [CrossControl](#)

### Typedefs

- typedef void \* [CrossControl::ADCHANDLE](#)

### Functions

- EXTERN\_C CCAUXDLL\_API ADCHANDLE CCAUXDLL\_CALLING\_CONV [CrossControl::GetAdc](#) (void)
- EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV [CrossControl::Adc\\_release](#) (ADCHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Adc\\_getVoltage](#) (ADCHANDLE, VoltageEnum selection, double \*value)

## 6.3 fixedIncludeFiles/AuxVersion.h File Reference

### Namespaces

- namespace [CrossControl](#)

### Typedefs

- typedef void \* [CrossControl::AUXVERSIONHANDLE](#)

### Functions

- EXTERN\_C CCAUXDLL\_API AUXVERSIONHANDLE CCAUXDLL\_CALLING\_CONV [CrossControl::GetAuxVersion](#) (void)
- EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV [CrossControl::AuxVersion\\_release](#) (AUXVERSIONHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::AuxVersion\\_getFPGAVersion](#) (AUXVERSIONHANDLE, unsigned char \*major, unsigned char \*minor, unsigned char \*release, unsigned char \*build)

- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::AuxVersion\\_getSSVersion](#) (AUXVERSIONHANDLE, unsigned char \*major, unsigned char \*minor, unsigned char \*release, unsigned char \*build)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::AuxVersion\\_getFrontVersion](#) (AUXVERSIONHANDLE, unsigned char \*major, unsigned char \*minor, unsigned char \*release, unsigned char \*build)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::AuxVersion\\_getCCAuxVersion](#) (AUXVERSIONHANDLE, unsigned char \*major, unsigned char \*minor, unsigned char \*release, unsigned char \*build)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::AuxVersion\\_getOSVersion](#) (AUXVERSIONHANDLE, unsigned char \*major, unsigned char \*minor, unsigned char \*release, unsigned char \*build)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::AuxVersion\\_getCCAuxDrvVersion](#) (AUXVERSIONHANDLE, unsigned char \*major, unsigned char \*minor, unsigned char \*release, unsigned char \*build)

## 6.4 fixedIncludeFiles/Backlight.h File Reference

### Namespaces

- namespace [CrossControl](#)

### Typedefs

- typedef void \* [CrossControl::BACKLIGHTHANDLE](#)

### Functions

- EXTERN\_C CCAUXDLL\_API BACKLIGHTHANDLE CCAUXDLL\_CALLING\_CONV [CrossControl::GetBacklight](#) (void)
- EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV [CrossControl::Backlight\\_release](#) (BACKLIGHTHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Backlight\\_getIntensity](#) (BACKLIGHTHANDLE, unsigned char \*intensity)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Backlight\\_setIntensity](#) (BACKLIGHTHANDLE, unsigned char intensity)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Backlight\\_getStatus](#) (BACKLIGHTHANDLE, unsigned char \*status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Backlight\\_startAutomaticBL](#) (BACKLIGHTHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Backlight\\_stopAutomaticBL](#) (BACKLIGHTHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Backlight\\_getAutomaticBLStatus](#) (BACKLIGHTHANDLE, unsigned char \*status)

- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Backlight\\_setAutomaticBLParams](#) (BACKLIGHTHANDLE, bool bSoftTransitions)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Backlight\\_getAutomaticBLParams](#) (BACKLIGHTHANDLE, bool \*bSoftTransitions, double \*k)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Backlight\\_setAutomaticBLFilter](#) (BACKLIGHTHANDLE, unsigned long averageWndSize, unsigned long rejectWndSize, unsigned long rejectDeltaInLux, LightSensorSamplingMode mode)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Backlight\\_getAutomaticBLFilter](#) (BACKLIGHTHANDLE, unsigned long \*averageWndSize, unsigned long \*rejectWndSize, unsigned long \*rejectDeltaInLux, LightSensorSamplingMode \*mode)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Backlight\\_getLedDimming](#) (BACKLIGHTHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Backlight\\_setLedDimming](#) (BACKLIGHTHANDLE, CCStatus status)

## 6.5 fixedIncludeFiles/Buzzer.h File Reference

### Namespaces

- namespace [CrossControl](#)

### Typedefs

- typedef void \* [CrossControl::BUZZERHANDLE](#)

### Functions

- EXTERN\_C CCAUXDLL\_API BUZZERHANDLE CCAUXDLL\_CALLING\_CONV [CrossControl::GetBuzzer](#) (void)
- EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV [CrossControl::Buzzer\\_release](#) (BUZZERHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Buzzer\\_getFrequency](#) (BUZZERHANDLE, unsigned short \*frequency)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Buzzer\\_getVolume](#) (BUZZERHANDLE, unsigned short \*volume)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Buzzer\\_getTrigger](#) (BUZZERHANDLE, bool \*trigger)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Buzzer\\_setFrequency](#) (BUZZERHANDLE, unsigned short frequency)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Buzzer\\_setVolume](#) (BUZZERHANDLE, unsigned short volume)



- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Buzzer\\_setTrigger](#) (BUZZERHANDLE, bool trigger)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Buzzer\\_buzze](#) (BUZZERHANDLE, int time, bool blocking)

## 6.6 fixedIncludeFiles/CanSetting.h File Reference

### Namespaces

- namespace [CrossControl](#)

### Typedefs

- typedef void \* [CrossControl::CANSETTINGHANDLE](#)

### Functions

- EXTERN\_C CCAUXDLL\_API CANSETTINGHANDLE CCAUXDLL\_CALLING\_CONV [CrossControl::GetCanSetting](#) (void)
- EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV [CrossControl::CanSetting\\_release](#) (CANSETTINGHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::CanSetting\\_getBaudrate](#) (CANSETTINGHANDLE, unsigned char net, unsigned short \*baudrate)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::CanSetting\\_getFrameType](#) (CANSETTINGHANDLE, unsigned char net, CanFrameType \*frameType)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::CanSetting\\_setBaudrate](#) (CANSETTINGHANDLE, unsigned char net, unsigned short baudrate)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::CanSetting\\_setFrameType](#) (CANSETTINGHANDLE, unsigned char net, CanFrameType frameType)

## 6.7 fixedIncludeFiles/CCAuxErrors.h File Reference

### Namespaces

- namespace [CrossControl](#)

### Functions

- EXTERN\_C CCAUXDLL\_API char const \*CCAUXDLL\_CALLING\_CONV [CrossControl::GetErrorStringA](#) (eErr errCode)

- `EXTERN_C CCAUXDLL_API wchar_t const *CCAUXDLL_CALLING_C-ONV CrossControl::GetErrorStringW (eErr errCode)`

## 6.8 fixedIncludeFiles/CCAuxTypes.h File Reference

### Classes

- struct `CrossControl::received_video`
- struct `CrossControl::video_dec_command`
- struct `CrossControl::version_info`
- struct `CrossControl::BuzzerSetup`
- struct `CrossControl::LedTimingType`
- struct `CrossControl::FpgaLedTimingType`
- struct `CrossControl::LedColorMixType`
- struct `CrossControl::TimerType`
- struct `CrossControl::UpgradeStatus`

### Namespaces

- namespace `CrossControl`

### Typedefs

- typedef struct version\_info `CrossControl::VersionType`

### Enumerations

- enum `CrossControl::VoltageEnum` { `CrossControl::VOLTAGE_24VIN` = 0, `CrossControl::VOLTAGE_24V`, `CrossControl::VOLTAGE_12V`, `CrossControl::VOLTAGE_12VID`, `CrossControl::VOLTAGE_5V`, `CrossControl::VOLTAGE_3V3`, `CrossControl::VOLTAGE_VTFT`, `CrossControl::VOLTAGE_5VSTB`, `CrossControl::VOLTAGE_1V9`, `CrossControl::VOLTAGE_1V8`, `CrossControl::VOLTAGE_1V5`, `CrossControl::VOLTAGE_1V2`, `CrossControl::VOLTAGE_1V05`, `CrossControl::VOLTAGE_1V0`, `CrossControl::VOLTAGE_0V9`, `CrossControl::VOLTAGE_VREF_INT`, `CrossControl::VOLTAGE_24V_BACKUP`, `CrossControl::VOLTAGE_2V5`, `CrossControl::VOLTAGE_1V1`, `CrossControl::VOLTAGE_1V3_PER`, `CrossControl::VOLTAGE_1V3_VDDA` }
- enum `CrossControl::LightSensorOperationRange` { `CrossControl::RangeStandard` = 0, `CrossControl::RangeExtended` = 1 }
- enum `CrossControl::LightSensorSamplingMode` { `CrossControl::SamplingModeStandard` = 0, `CrossControl::SamplingModeExtended`, `CrossControl::SamplingModeAuto` }
- enum `CrossControl::CCStatus` { `CrossControl::Disabled` = 0, `CrossControl::Enabled` = 1 }

- enum `CrossControl::eErr` { `CrossControl::ERR_SUCCESS` = 0, `CrossControl::ERR_OPEN_FAILED` = 1, `CrossControl::ERR_NOT_SUPPORTED` = 2, `CrossControl::ERR_UNKNOWN_FEATURE` = 3, `CrossControl::ERR_DATATYPE_MISMATCH` = 4, `CrossControl::ERR_CODE_NOT_EXIST` = 5, `CrossControl::ERR_BUFFER_SIZE` = 6, `CrossControl::ERR_IOCTL_FAILED` = 7, `CrossControl::ERR_INVALID_DATA` = 8, `CrossControl::ERR_INVALID_PARAMETER` = 9, `CrossControl::ERR_CREATE_THREAD` = 10, `CrossControl::ERR_IN_PROGRESS` = 11, `CrossControl::ERR_CHECKSUM` = 12, `CrossControl::ERR_INIT_FAILED` = 13, `CrossControl::ERR_VERIFY_FAILED` = 14, `CrossControl::ERR_DEVICE_READ_DATA_FAILED` = 15, `CrossControl::ERR_DEVICE_WRITE_DATA_FAILED` = 16, `CrossControl::ERR_COMMAND_FAILED` = 17, `CrossControl::ERR_EEPROM` = 18, `CrossControl::ERR_JIDA_TEMP` = 19, `CrossControl::ERR_AVERAGE_CALC_STARTED` = 20, `CrossControl::ERR_NOT_RUNNING` = 21, `CrossControl::ERR_I2C_EXPANDER_READ_FAILED` = 22, `CrossControl::ERR_I2C_EXPANDER_WRITE_FAILED` = 23, `CrossControl::ERR_I2C_EXPANDER_INIT_FAILED` = 24, `CrossControl::ERR_NEWER_SS_VERSION_REQUIRED` = 25, `CrossControl::ERR_NEWER_FPGA_VERSION_REQUIRED` = 26, `CrossControl::ERR_NEWER_FRONT_VERSION_REQUIRED` = 27, `CrossControl::ERR_TELEMATICS_GPRS_NOT_AVAILABLE` = 28, `CrossControl::ERR_TELEMATICS_WLAN_NOT_AVAILABLE` = 29, `CrossControl::ERR_TELEMATICS_BT_NOT_AVAILABLE` = 30, `CrossControl::ERR_TELEMATICS_GPS_NOT_AVAILABLE` = 31, `CrossControl::ERR_MEM_ALLOC_FAIL` = 32, `CrossControl::ERR_JOIN_THREAD` = 33 }
- enum `CrossControl::DeInterlaceMode` { `CrossControl::DeInterlace_Even` = 0, `CrossControl::DeInterlace_Odd` = 1, `CrossControl::DeInterlace_BOB` = 2 }
- enum `CrossControl::VideoChannel` { `CrossControl::Analog_Channel_1` = 0, `CrossControl::Analog_Channel_2` = 1, `CrossControl::Analog_Channel_3` = 2, `CrossControl::Analog_Channel_4` = 3 }
- enum `CrossControl::videoStandard` { `CrossControl::STD_M_J_NTSC` = 0, `CrossControl::STD_B_D_G_H_I_N_PAL` = 1, `CrossControl::STD_M_PAL` = 2, `CrossControl::STD_PAL` = 3, `CrossControl::STD_NTSC` = 4, `CrossControl::STD_SECAM` = 5 }
- enum `CrossControl::CanFrameType` { `CrossControl::FrameStandard`, `CrossControl::FrameExtended`, `CrossControl::FrameStandardExtended` }
- enum `CrossControl::TriggerConf` { `CrossControl::Front_Button_Enabled` = 1, `CrossControl::OnOff_Signal_Enabled` = 2, `CrossControl::Both_Button_And_Signal_Enabled` = 3 }
- enum `CrossControl::PowerAction` { `CrossControl::NoAction` = 0, `CrossControl::ActionSuspend` = 1, `CrossControl::ActionShutDown` = 2 }
- enum `CrossControl::ButtonPowerTransitionStatus` { `CrossControl::BPTS_No_Change` = 0, `CrossControl::BPTS_ShutDown` = 1, `CrossControl::BPTS_Suspend` = 2, `CrossControl::BPTS_Restart` = 3, `CrossControl::BPTS_BtnPressed` = 4, `CrossControl::BPTS_BtnPressedLong` = 5, `CrossControl::BPTS_SignalOff` = 6 }
- enum `CrossControl::OCDStatus` { `CrossControl::OCD_OK` = 0, `CrossControl::OCD_OC` = 1, `CrossControl::OCD_POWER_OFF` = 2 }
- enum `CrossControl::JidaSensorType` { `CrossControl::TEMP_CPU` = 0, `CrossControl::TEMP_BOX` = 1, `CrossControl::TEMP_ENV` = 2, `CrossControl::TE-`

MP\_BOARD = 3, CrossControl::TEMP\_BACKPLANE = 4, CrossControl::TEMP\_CHIPSETS = 5, CrossControl::TEMP\_VIDEO = 6, CrossControl::TEMP\_OTHER = 7 }

- enum CrossControl::UpgradeAction { CrossControl::UPGRADE\_INIT, CrossControl::UPGRADE\_PREP\_COM, CrossControl::UPGRADE\_READING\_FILE, CrossControl::UPGRADE\_CONVERTING\_FILE, CrossControl::UPGRADE\_FLASHING, CrossControl::UPGRADE\_VERIFYING, CrossControl::UPGRADE\_COMPLETE, CrossControl::UPGRADE\_COMPLETE\_WITH\_ERRORS }
- enum CrossControl::CCAuxColor { CrossControl::RED = 0, CrossControl::GREEN, CrossControl::BLUE, CrossControl::CYAN, CrossControl::MAGENTA, CrossControl::YELLOW, CrossControl::UNDEFINED\_COLOR }

## 6.9 fixedIncludeFiles/CCPlatform.h File Reference

### 6.10 fixedIncludeFiles/Config.h File Reference

#### Namespaces

- namespace [CrossControl](#)

#### Typedefs

- typedef void \* [CrossControl::CONFIGHANDLE](#)

#### Functions

- EXTERN\_C CCAUXDLL\_API CONFIGHANDLE CCAUXDLL\_CALLING\_CONV [CrossControl::GetConfig](#) ()
- EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV [CrossControl::Config\\_release](#) (CONFIGHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Config\\_getStartupTriggerConfig](#) (CONFIGHANDLE, TriggerConf \*config)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Config\\_getShortButtonPressAction](#) (CONFIGHANDLE, PowerAction \*action)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Config\\_getLongButtonPressAction](#) (CONFIGHANDLE, PowerAction \*action)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Config\\_getOnOffSigAction](#) (CONFIGHANDLE, PowerAction \*action)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Config\\_getFrontBtnTrigTime](#) (CONFIGHANDLE, unsigned short \*triggertime)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Config\\_getExtOnOffSigTrigTime](#) (CONFIGHANDLE, unsigned long \*triggertime)

- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::Config\\_getSuspendMaxTime](#) (CONFIGHANDLE, unsigned short \*maxTime)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::Config\\_getCanStartupPowerConfig](#) (CONFIGHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::Config\\_getVideoStartupPowerConfig](#) (CONFIGHANDLE, unsigned char \*config)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::Config\\_getExtFanStartupPowerConfig](#) (CONFIGHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::Config\\_getStartupVoltageConfig](#) (CONFIGHANDLE, double \*voltage)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::Config\\_getHeatingTempLimit](#) (CONFIGHANDLE, signed short \*temperature)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::Config\\_getPowerOnStartup](#) (CONFIGHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::Config\\_setStartupTriggerConfig](#) (CONFIGHANDLE, TriggerConf conf)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::Config\\_setShortButtonPressAction](#) (CONFIGHANDLE, PowerAction action)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::Config\\_setLongButtonPressAction](#) (CONFIGHANDLE, PowerAction action)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::Config\\_setOnOffSigAction](#) (CONFIGHANDLE, PowerAction action)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::Config\\_setFrontBtnTrigTime](#) (CONFIGHANDLE, unsigned short triggertime)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::Config\\_setExtOnOffSigTrigTime](#) (CONFIGHANDLE, unsigned long triggertime)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::Config\\_setSuspendMaxTime](#) (CONFIGHANDLE, unsigned short maxTime)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::Config\\_setCanStartupPowerConfig](#) (CONFIGHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::Config\\_setVideoStartupPowerConfig](#) (CONFIGHANDLE, unsigned char config)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::Config\\_setExtFanStartupPowerConfig](#) (CONFIGHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::Config\\_setStartupVoltageConfig](#) (CONFIGHANDLE, double voltage)

- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Config\\_setHeatingTempLimit](#) (CONFIGHANDLE, signed short temperature)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Config\\_setPowerOnStartup](#) (CONFIGHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Config\\_setTFTMode](#) (CONFIGHANDLE, bool enable)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Config\\_setTFTScan](#) (CONFIGHANDLE, bool enable)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Config\\_setTFTMirror](#) (CONFIGHANDLE, bool enable)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Config\\_getTFTMode](#) (CONFIGHANDLE, bool \*enable)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Config\\_getTFTScan](#) (CONFIGHANDLE, bool \*enable)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Config\\_getTFTMirror](#) (CONFIGHANDLE, bool \*enable)

### Variables

- const unsigned char [CrossControl::Video1Conf](#) = (1 << 0)
- const unsigned char [CrossControl::Video2Conf](#) = (1 << 1)
- const unsigned char [CrossControl::Video3Conf](#) = (1 << 2)
- const unsigned char [CrossControl::Video4Conf](#) = (1 << 3)

## 6.11 fixedIncludeFiles/Diagnostic.h File Reference

### Namespaces

- namespace [CrossControl](#)

### Typedefs

- typedef void \* [CrossControl::DIAGNOSTICHANDLE](#)

### Functions

- EXTERN\_C CCAUXDLL\_API DIAGNOSTICHANDLE CCAUXDLL\_CALLING\_CONV [CrossControl::GetDiagnostic](#) (void)
- EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV [CrossControl::Diagnostic\\_release](#) (DIAGNOSTICHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Diagnostic\\_getSSTemp](#) (DIAGNOSTICHANDLE, signed short \*temperature)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Diagnostic\\_getPCBTemp](#) (DIAGNOSTICHANDLE, signed short \*temperature)

- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Diagnostic\\_getPMTemp](#) (DIAGNOSTICHANDLE, unsigned char index, signed short \*temperature, JidaSensorType \*jst)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Diagnostic\\_getStartupReason](#) (DIAGNOSTICHANDLE, unsigned short \*reason)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Diagnostic\\_getShutdownReason](#) (DIAGNOSTICHANDLE, unsigned short \*reason)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Diagnostic\\_getHwErrorStatus](#) (DIAGNOSTICHANDLE, unsigned short \*errorCode)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Diagnostic\\_getTimer](#) (DIAGNOSTICHANDLE, TimerType \*times)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Diagnostic\\_getMinMaxTemp](#) (DIAGNOSTICHANDLE, signed short \*minTemp, signed short \*maxTemp)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Diagnostic\\_getPowerCycles](#) (DIAGNOSTICHANDLE, unsigned short \*powerCycles)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Diagnostic\\_clearHwErrorStatus](#) (DIAGNOSTICHANDLE)

## 6.12 fixedIncludeFiles/DiagnosticCodes.h File Reference

### Namespaces

- namespace [CrossControl](#)

### Enumerations

- enum [CrossControl::startupReasonCodes](#) { [CrossControl::startupReasonCodeUndefined](#) = 0x0000, [CrossControl::startupReasonCodeButtonPress](#) = 0x0055, [CrossControl::startupReasonCodeExtCtrl](#) = 0x00AA, [CrossControl::startupReasonCodeMPRestart](#) = 0x00F0, [CrossControl::startupReasonCodePowerOnStartup](#) = 0x000F }
- enum [CrossControl::shutdownReasonCodes](#) { [CrossControl::shutdownReasonCodeNoError](#) = 0x001F }
- enum [CrossControl::hwErrorStatusCodes](#) { [CrossControl::errCodeNoErr](#) = 0 }

### Functions

- EXTERN\_C CCAUXDLL\_API char const \*CCAUXDLL\_CALLING\_CONV [CrossControl::GetHwErrorStatusStringA](#) (unsigned short errorCode)
- EXTERN\_C CCAUXDLL\_API wchar\_t const \*CCAUXDLL\_CALLING\_CONV [CrossControl::GetHwErrorStatusStringW](#) (unsigned short errorCode)

- EXTERN\_C CCAUXDLL\_API char const \*CCAUXDLL\_CALLING\_CONV [CrossControl::GetStartupReasonStringA](#) (unsigned short code)
- EXTERN\_C CCAUXDLL\_API wchar\_t const \*CCAUXDLL\_CALLING\_CONV [CrossControl::GetStartupReasonStringW](#) (unsigned short code)

## 6.13 fixedIncludeFiles/DigIO.h File Reference

### Namespaces

- namespace [CrossControl](#)

### Typedefs

- typedef void \* [CrossControl::DIGIOHANDLE](#)

### Functions

- EXTERN\_C CCAUXDLL\_API DIGIOHANDLE CCAUXDLL\_CALLING\_CONV [CrossControl::GetDigIO](#) (void)
- EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV [CrossControl::DigIO\\_release](#) (DIGIOHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::DigIO\\_getDigIO](#) (DIGIOHANDLE, unsigned char \*status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::DigIO\\_setDigIO](#) (DIGIOHANDLE, unsigned char state)

### Variables

- const unsigned char [CrossControl::DigitalIn\\_1](#) = (1 << 0)
- const unsigned char [CrossControl::DigitalIn\\_2](#) = (1 << 1)
- const unsigned char [CrossControl::DigitalIn\\_3](#) = (1 << 2)
- const unsigned char [CrossControl::DigitalIn\\_4](#) = (1 << 3)

## 6.14 fixedIncludeFiles/FirmwareUpgrade.h File Reference

### Namespaces

- namespace [CrossControl](#)

### Typedefs

- typedef void \* [CrossControl::FIRMWAREUPGHANDLE](#)



## Functions

- EXTERN\_C CCAUXDLL\_API FIRMWAREUPGHANDLE CCAUXDLL\_CALLING\_CONV [CrossControl::GetFirmwareUpgrade](#) (void)
- EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV [CrossControl::FirmwareUpgrade\\_release](#) (FIRMWAREUPGHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::FirmwareUpgrade\\_startFpgaUpgrade](#) (FIRMWAREUPGHANDLE, const char \*filename, bool blocking)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::FirmwareUpgrade\\_startFpgaVerification](#) (FIRMWAREUPGHANDLE, const char \*filename, bool blocking)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::FirmwareUpgrade\\_startSSUpgrade](#) (FIRMWAREUPGHANDLE, const char \*filename, bool blocking)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::FirmwareUpgrade\\_startSSVerification](#) (FIRMWAREUPGHANDLE, const char \*filename, bool blocking)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::FirmwareUpgrade\\_startFrontUpgrade](#) (FIRMWAREUPGHANDLE, const char \*filename, bool blocking)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::FirmwareUpgrade\\_startFrontVerification](#) (FIRMWAREUPGHANDLE, const char \*filename, bool blocking)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::FirmwareUpgrade\\_getUpgradeStatus](#) (FIRMWAREUPGHANDLE, Upgrade-Status \*status, bool blocking)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::FirmwareUpgrade\\_shutDown](#) (FIRMWAREUPGHANDLE)

## 6.15 fixedIncludeFiles/FrontLED.h File Reference

### Namespaces

- namespace [CrossControl](#)

### Typedefs

- typedef void \* [CrossControl::FRONTLEDHANDLE](#)

### Functions

- EXTERN\_C CCAUXDLL\_API FRONTLEDHANDLE CCAUXDLL\_CALLING\_CONV [CrossControl::GetFrontLED](#) (void)
- EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV [CrossControl::FrontLED\\_release](#) (FRONTLEDHANDLE)

- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::FrontLED\\_getSignal](#) (FRONTLEDHANDLE, double \*frequency, unsigned char \*dutyCycle)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::FrontLED\\_getOnTime](#) (FRONTLEDHANDLE, unsigned char \*onTime)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::FrontLED\\_getOffTime](#) (FRONTLEDHANDLE, unsigned char \*offTime)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::FrontLED\\_getIdleTime](#) (FRONTLEDHANDLE, unsigned char \*idleTime)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::FrontLED\\_getNrOfPulses](#) (FRONTLEDHANDLE, unsigned char \*nrOfPulses)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::FrontLED\\_getColor](#) (FRONTLEDHANDLE, unsigned char \*red, unsigned char \*green, unsigned char \*blue)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::FrontLED\\_getStandardColor](#) (FRONTLEDHANDLE, CCAuxColor \*color)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::FrontLED\\_getEnabledDuringStartup](#) (FRONTLEDHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::FrontLED\\_setSignal](#) (FRONTLEDHANDLE, double frequency, unsigned char dutyCycle)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::FrontLED\\_setOnTime](#) (FRONTLEDHANDLE, unsigned char onTime)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::FrontLED\\_setOffTime](#) (FRONTLEDHANDLE, unsigned char offTime)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::FrontLED\\_setIdleTime](#) (FRONTLEDHANDLE, unsigned char idleTime)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::FrontLED\\_setNrOfPulses](#) (FRONTLEDHANDLE, unsigned char nrOfPulses)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::FrontLED\\_setColor](#) (FRONTLEDHANDLE, unsigned char red, unsigned char green, unsigned char blue)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::FrontLED\\_setStandardColor](#) (FRONTLEDHANDLE, CCAuxColor color)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::FrontLED\\_setOff](#) (FRONTLEDHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::FrontLED\\_setEnabledDuringStartup](#) (FRONTLEDHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [Cross-Control::FrontLED\\_getFrontPcbRev](#) (FRONTLEDHANDLE, unsigned char \*major, unsigned char \*minor)

## 6.16 fixedIncludeFiles/Lightsensor.h File Reference

### Namespaces

- namespace [CrossControl](#)

### Typedefs

- typedef void \* [CrossControl::LIGHTSENSORHANDLE](#)

### Functions

- EXTERN\_C CCAUXDLL\_API LIGHTSENSORHANDLE CCAUXDLL\_CALLING\_CONV [CrossControl::GetLightsensor](#) (void)
- EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV [CrossControl::Lightsensor\\_release](#) (LIGHTSENSORHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Lightsensor\\_getIlluminance](#) (LIGHTSENSORHANDLE, unsigned short \*value)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Lightsensor\\_getIlluminance2](#) (LIGHTSENSORHANDLE, unsigned short \*value, unsigned char \*ch0, unsigned char \*ch1)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Lightsensor\\_getAverageIlluminance](#) (LIGHTSENSORHANDLE, unsigned short \*value)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Lightsensor\\_startAverageCalc](#) (LIGHTSENSORHANDLE, unsigned long averageWndSize, unsigned long rejectWndSize, unsigned long rejectDeltaInLux, LightSensorSamplingMode mode)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Lightsensor\\_stopAverageCalc](#) (LIGHTSENSORHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Lightsensor\\_getOperatingRange](#) (LIGHTSENSORHANDLE, LightSensorOperationRange \*range)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Lightsensor\\_setOperatingRange](#) (LIGHTSENSORHANDLE, LightSensorOperationRange range)

## 6.17 fixedIncludeFiles/Power.h File Reference

### Namespaces

- namespace [CrossControl](#)

## Typedefs

- typedef void \* [CrossControl::POWERHANDLE](#)

## Functions

- EXTERN\_C CCAUXDLL\_API POWERHANDLE CCAUXDLL\_CALLING\_CONV [CrossControl::GetPower](#) (void)
- EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV [CrossControl::Power\\_release](#) (POWERHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Power\\_getBLPowerStatus](#) (POWERHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Power\\_getCanPowerStatus](#) (POWERHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Power\\_getVideoPowerStatus](#) (POWERHANDLE, unsigned char \*videoStatus)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Power\\_getExtFanPowerStatus](#) (POWERHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Power\\_getButtonPowerTransitionStatus](#) (POWERHANDLE, ButtonPowerTransitionStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Power\\_getVideoOCDStatus](#) (POWERHANDLE, OCDStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Power\\_getCanOCDStatus](#) (POWERHANDLE, OCDStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Power\\_setBLPowerStatus](#) (POWERHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Power\\_setCanPowerStatus](#) (POWERHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Power\\_setVideoPowerStatus](#) (POWERHANDLE, unsigned char status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Power\\_setExtFanPowerStatus](#) (POWERHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Power\\_ackPowerRequest](#) (POWERHANDLE)

## 6.18 fixedIncludeFiles/Telematics.h File Reference

### Namespaces

- namespace [CrossControl](#)

### Typedefs

- typedef void \* [CrossControl::TELEMATICSHANDLE](#)

## Functions

- EXTERN\_C CCAUXDLL\_API TELEMATICSHANDLE CCAUXDLL\_CALLING\_CONV [CrossControl::GetTelematics](#) (void)
- EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV [CrossControl::Telematics\\_release](#) (TELEMATICSHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Telematics\\_getTelematicsAvailable](#) (TELEMATICSHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Telematics\\_getGPRSPowerStatus](#) (TELEMATICSHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Telematics\\_getGPRSStartupPowerStatus](#) (TELEMATICSHANDLE, -CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Telematics\\_getWLANPowerStatus](#) (TELEMATICSHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Telematics\\_getWLANStartupPowerStatus](#) (TELEMATICSHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Telematics\\_getBTPowerStatus](#) (TELEMATICSHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Telematics\\_getBTStartupPowerStatus](#) (TELEMATICSHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Telematics\\_getGPSPowerStatus](#) (TELEMATICSHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Telematics\\_getGPSStartupPowerStatus](#) (TELEMATICSHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Telematics\\_getGPSAntennaStatus](#) (TELEMATICSHANDLE, CCStatus \*status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Telematics\\_setGPRSPowerStatus](#) (TELEMATICSHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Telematics\\_setGPRSStartupPowerStatus](#) (TELEMATICSHANDLE, -CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Telematics\\_setWLANPowerStatus](#) (TELEMATICSHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Telematics\\_setWLANStartupPowerStatus](#) (TELEMATICSHANDLE, -CCStatus status)

- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Telematics\\_setBTPowerStatus](#) (TELEMATICSHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Telematics\\_setBTStartUpPowerStatus](#) (TELEMATICSHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Telematics\\_setGPSPowerStatus](#) (TELEMATICSHANDLE, CCStatus status)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Telematics\\_setGPSStartUpPowerStatus](#) (TELEMATICSHANDLE, CCStatus status)

## 6.19 fixedIncludeFiles/TouchScreen.h File Reference

### Namespaces

- namespace [CrossControl](#)

### Typedefs

- typedef void \* [CrossControl::TOUCHSCREENHANDLE](#)

### Enumerations

- enum [CrossControl::TouchScreenModeSettings](#) { [CrossControl::MOUSE\\_NEXT\\_BOOT](#) = 0, [CrossControl::TOUCH\\_NEXT\\_BOOT](#) = 1, [CrossControl::MOUSE\\_NOW](#) = 2, [CrossControl::TOUCH\\_NOW](#) = 3 }
- enum [CrossControl::TSAdvancedSettingsParameter](#) { [CrossControl::TS\\_RIGHT\\_CLICK\\_TIME](#) = 0, [CrossControl::TS\\_LOW\\_LEVEL](#) = 1, [CrossControl::TS\\_UNTOUCHLEVEL](#) = 2, [CrossControl::TS\\_DEBOUNCE\\_TIME](#) = 3, [CrossControl::TS\\_DEBOUNCE\\_TIMEOUT\\_TIME](#) = 4, [CrossControl::TS\\_DOUBLECLICK\\_MAX\\_CLICK\\_TIME](#) = 5, [CrossControl::TS\\_DOUBLE\\_CLICK\\_TIME](#) = 6, [CrossControl::TS\\_MAX\\_RIGHTCLICK\\_DISTANCE](#) = 7, [CrossControl::TS\\_USE\\_DEJITTER](#) = 8, [CrossControl::TS\\_CALIBTATION\\_WIDTH](#) = 9, [CrossControl::TS\\_CALIBRATION\\_MEASUREMENTS](#) = 10, [CrossControl::TS\\_RESTORE\\_DEFAULT\\_SETTINGS](#) = 11 }

### Functions

- EXTERN\_C CCAUXDLL\_API TOUCHSCREENHANDLE CCAUXDLL\_CALLING\_CONV [CrossControl::GetTouchScreen](#) (void)
- EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV [CrossControl::TouchScreen\\_release](#) (TOUCHSCREENHANDLE)

- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::TouchScreen\\_getMode](#) (TOUCHSCREENHANDLE, TouchScreenModeSettings \*config)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::TouchScreen\\_getMouseRightClickTime](#) (TOUCHSCREENHANDLE, unsigned short \*time)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::TouchScreen\\_setMode](#) (TOUCHSCREENHANDLE, TouchScreenModeSettings config)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::TouchScreen\\_setMouseRightClickTime](#) (TOUCHSCREENHANDLE, unsigned short time)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::TouchScreen\\_setAdvancedSetting](#) (TOUCHSCREENHANDLE, TSAdvancedSettingsParameter param, unsigned short data)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::TouchScreen\\_getAdvancedSetting](#) (TOUCHSCREENHANDLE, TSAdvancedSettingsParameter param, unsigned short \*data)

## 6.20 fixedIncludeFiles/TouchScreenCalib.h File Reference

### Namespaces

- namespace [CrossControl](#)

### Typedefs

- typedef void \* [CrossControl::TOUCHSCREENCALIBHANDLE](#)

### Enumerations

- enum [CrossControl::CalibrationModeSettings](#) { [CrossControl::MODE\\_UNKNOWN](#) = 0, [CrossControl::MODE\\_NORMAL](#) = 1, [CrossControl::MODE\\_CALIBRATION\\_5P](#) = 2, [CrossControl::MODE\\_CALIBRATION\\_9P](#) = 3, [CrossControl::MODE\\_CALIBRATION\\_13P](#) = 4 }
- enum [CrossControl::CalibrationConfigParam](#) { [CrossControl::CONFIG\\_CALIBRATION\\_WITH](#) = 0, [CrossControl::CONFIG\\_CALIBRATION\\_MEASUREMENTS](#) = 1, [CrossControl::CONFIG\\_5P\\_CALIBRATION\\_POINT\\_BORDER](#) = 2, [CrossControl::CONFIG\\_13P\\_CALIBRATION\\_POINT\\_BORDER](#) = 3, [CrossControl::CONFIG\\_13P\\_CALIBRATION\\_TRANSITION\\_MIN](#) = 4, [CrossControl::CONFIG\\_13P\\_CALIBRATION\\_TRANSITION\\_MAX](#) = 5 }

### Functions

- EXTERN\_C CCAUXDLL\_API TOUCHSCREENCALIBHANDLE CCAUXDLL\_CALLING\_CONV [CrossControl::GetTouchScreenCalib](#) (void)

- EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV [CrossControl::TouchScreenCalib\\_release](#) (TOUCHSCREENCALIBHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::TouchScreenCalib\\_setMode](#) (TOUCHSCREENCALIBHANDLE, CalibrationModeSettings mode)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::TouchScreenCalib\\_getMode](#) (TOUCHSCREENCALIBHANDLE, CalibrationModeSettings \*mode)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::TouchScreenCalib\\_setCalibrationPoint](#) (TOUCHSCREENCALIBHANDLE, unsigned char pointNr)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::TouchScreenCalib\\_checkCalibrationPointFinished](#) (TOUCHSCREENCALIBHANDLE, bool \*finished, unsigned char pointNr)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::TouchScreenCalib\\_getConfigParam](#) (TOUCHSCREENCALIBHANDLE, CalibrationConfigParam param, unsigned short \*value)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::TouchScreenCalib\\_setConfigParam](#) (TOUCHSCREENCALIBHANDLE, CalibrationConfigParam param, unsigned short value)

## 6.21 fixedIncludeFiles/Video.h File Reference

### Namespaces

- namespace [CrossControl](#)

### Typedefs

- typedef void \* [CrossControl::VIDEOHANDLE](#)

### Functions

- EXTERN\_C CCAUXDLL\_API VIDEOHANDLE CCAUXDLL\_CALLING\_CONV [CrossControl::GetVideo](#) (void)
- EXTERN\_C CCAUXDLL\_API void CCAUXDLL\_CALLING\_CONV [CrossControl::Video\\_release](#) (VIDEOHANDLE)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Video\\_init](#) (VIDEOHANDLE, unsigned char deviceNr)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Video\\_showVideo](#) (VIDEOHANDLE, bool show)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Video\\_setDeInterlaceMode](#) (VIDEOHANDLE, DeInterlaceMode mode)
- EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV [CrossControl::Video\\_getDeInterlaceMode](#) (VIDEOHANDLE, DeInterlaceMode \*mode)



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

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

# Index

- ActionShutDown
  - CrossControl, 24
- ActionSuspend
  - CrossControl, 24
- Analog\_Channel\_1
  - CrossControl, 26
- Analog\_Channel\_2
  - CrossControl, 26
- Analog\_Channel\_3
  - CrossControl, 26
- Analog\_Channel\_4
  - CrossControl, 26
- BLUE
  - CrossControl, 20
- BPTS\_BtnPressed
  - CrossControl, 19
- BPTS\_BtnPressedLong
  - CrossControl, 19
- BPTS\_No\_Change
  - CrossControl, 19
- BPTS\_Restart
  - CrossControl, 19
- BPTS\_ShutDown
  - CrossControl, 19
- BPTS\_SignalOff
  - CrossControl, 19
- BPTS\_Suspend
  - CrossControl, 19
- Both\_Button\_And\_Signal\_Enabled
  - CrossControl, 25
- CONFIG\_13P\_CALIBRATION\_POINT\_BORDER
  - CrossControl, 20
- CONFIG\_13P\_CALIBRATION\_TRANSITION\_MAX
  - CrossControl, 20
- CONFIG\_13P\_CALIBRATION\_TRANSITION\_MIN
  - CrossControl, 20
- CONFIG\_5P\_CALIBRATION\_POINT\_BORDER
  - CrossControl, 20
- CONFIG\_CALIBRATION\_MEASUREMENTS
  - CrossControl, 20
- CONFIG\_CALIBRATION\_WITH
  - CrossControl, 20
- CYAN
  - CrossControl, 20
- CrossControl
  - ActionShutDown, 24
  - ActionSuspend, 24
  - Analog\_Channel\_1, 26
  - Analog\_Channel\_2, 26
  - Analog\_Channel\_3, 26
  - Analog\_Channel\_4, 26
  - BLUE, 20
  - BPTS\_BtnPressed, 19
  - BPTS\_BtnPressedLong, 19
  - BPTS\_No\_Change, 19
  - BPTS\_Restart, 19
  - BPTS\_ShutDown, 19
  - BPTS\_SignalOff, 19
  - BPTS\_Suspend, 19
  - Both\_Button\_And\_Signal\_Enabled, 25
  - CONFIG\_13P\_CALIBRATION\_POINT\_BORDER, 20
  - CONFIG\_13P\_CALIBRATION\_TRANSITION\_MAX, 20
  - CONFIG\_13P\_CALIBRATION\_TRANSITION\_MIN, 20
  - CONFIG\_5P\_CALIBRATION\_POINT\_BORDER, 20
  - CONFIG\_CALIBRATION\_MEASUREMENTS, 20
  - CONFIG\_CALIBRATION\_WITH, 20
  - CYAN, 20
  - DeInterlace\_BOB, 21
  - DeInterlace\_Even, 21
  - DeInterlace\_Odd, 21
  - Disabled, 21

- ERR\_AVERAGE\_CALC\_STARTED, 22
- ERR\_BUFFER\_SIZE, 21
- ERR\_CHECKSUM, 21
- ERR\_CODE\_NOT\_EXIST, 21
- ERR\_COMMAND\_FAILED, 22
- ERR\_CREATE\_THREAD, 21
- ERR\_DATATYPE\_MISMATCH, 21
- ERR\_DEVICE\_READ\_DATA\_FAILED, 22
- ERR\_DEVICE\_WRITE\_DATA\_FAILED, 22
- ERR\_EEPROM, 22
- ERR\_I2C\_EXPANDER\_INIT\_FAILED, 22
- ERR\_I2C\_EXPANDER\_READ\_FAILED, 22
- ERR\_I2C\_EXPANDER\_WRITE\_FAILED, 22
- ERR\_INIT\_FAILED, 21
- ERR\_INVALID\_DATA, 21
- ERR\_INVALID\_PARAMETER, 21
- ERR\_IN\_PROGRESS, 21
- ERR\_IOCTL\_FAILED, 21
- ERR\_JIDA\_TEMP, 22
- ERR\_JOIN\_THREAD, 22
- ERR\_MEM\_ALLOC\_FAIL, 22
- ERR\_NEWER\_FPGA\_VERSION\_REQUIRED, 22
- ERR\_NEWER\_FRONT\_VERSION\_REQUIRED, 22
- ERR\_NEWER\_SS\_VERSION\_REQUIRED, 22
- ERR\_NOT\_RUNNING, 22
- ERR\_NOT\_SUPPORTED, 21
- ERR\_OPEN\_FAILED, 21
- ERR\_SUCCESS, 21
- ERR\_TELEMATICS\_BT\_NOT\_AVAILABLE, 22
- ERR\_TELEMATICS\_GPRS\_NOT\_AVAILABLE, 22
- ERR\_TELEMATICS\_GPS\_NOT\_AVAILABLE, 22
- ERR\_TELEMATICS\_WLAN\_NOT\_AVAILABLE, 22
- ERR\_UNKNOWN\_FEATURE, 21
- ERR\_VERIFY\_FAILED, 22
- Enabled, 21
- FrameExtended, 20
- FrameStandard, 20
- FrameStandardExtended, 20
- Front\_Button\_Enabled, 25
- GREEN, 20
- MAGENTA, 21
- MODE\_CALIBRATION\_13P, 20
- MODE\_CALIBRATION\_5P, 20
- MODE\_CALIBRATION\_9P, 20
- MODE\_NORMAL, 20
- MODE\_UNKNOWN, 20
- MOUSE\_NEXT\_BOOT, 24
- MOUSE\_NOW, 24
- NoAction, 24
- OCD\_OC, 23
- OCD\_OK, 23
- OCD\_POWER\_OFF, 23
- OnOff\_Signal\_Enabled, 25
- RED, 20
- RangeExtended, 23
- RangeStandard, 23
- STD\_B\_D\_G\_H\_I\_N\_PAL, 26
- STD\_M\_J\_NTSC, 26
- STD\_M\_PAL, 26
- STD\_NTSC, 26
- STD\_PAL, 26
- STD\_SECAM, 26
- SamplingModeAuto, 23
- SamplingModeExtended, 23
- SamplingModeStandard, 23
- TEMP\_BACKPLANE, 23
- TEMP\_BOARD, 22
- TEMP\_BOX, 22
- TEMP\_CHIPSETS, 23
- TEMP\_CPU, 22
- TEMP\_ENV, 22
- TEMP\_OTHER, 23
- TEMP\_VIDEO, 23
- TOUCH\_NEXT\_BOOT, 24
- TOUCH\_NOW, 24
- TS\_CALIBRATION\_MEASUREMENTS, 25
- TS\_CALIBRATION\_WIDTH, 25
- TS\_DEBOUNCE\_TIME, 25
- TS\_DEBOUNCE\_TIMEOUT\_TIME, 25
- TS\_DOUBLECLICK\_MAX\_CLICK\_TIME, 25
- TS\_DOUBLE\_CLICK\_TIME, 25
- TS\_LOW\_LEVEL, 25
- TS\_MAX\_RIGHTCLICK\_DISTANCE, 25

- TS\_RESTORE\_DEFAULT\_SETTING- Disabled  
 GS, 25  
 TS\_RIGHT\_CLICK\_TIME, 25  
 TS\_UNTOUCHLEVEL, 25  
 TS\_USE\_DEJITTER, 25  
 UNDEFINED\_COLOR, 21  
 UPGRADE\_COMPLETE, 26  
 UPGRADE\_COMPLETE\_WITH\_ERRORS, 26  
 UPGRADE\_CONVERTING\_FILE, 26  
 UPGRADE\_FLASHING, 26  
 UPGRADE\_INIT, 26  
 UPGRADE\_PREP\_COM, 26  
 UPGRADE\_READING\_FILE, 26  
 UPGRADE\_VERIFYING, 26  
 VOLTAGE\_0V9, 27  
 VOLTAGE\_12V, 27  
 VOLTAGE\_12VID, 27  
 VOLTAGE\_1V0, 27  
 VOLTAGE\_1V05, 27  
 VOLTAGE\_1V1, 27  
 VOLTAGE\_1V2, 27  
 VOLTAGE\_1V3\_PER, 27  
 VOLTAGE\_1V3\_VDDA, 27  
 VOLTAGE\_1V5, 27  
 VOLTAGE\_1V8, 27  
 VOLTAGE\_1V9, 27  
 VOLTAGE\_24V, 27  
 VOLTAGE\_24VIN, 27  
 VOLTAGE\_24V\_BACKUP, 27  
 VOLTAGE\_2V5, 27  
 VOLTAGE\_3V3, 27  
 VOLTAGE\_5V, 27  
 VOLTAGE\_5VSTB, 27  
 VOLTAGE\_VREF\_INT, 27  
 VOLTAGE\_VTFT, 27  
 YELLOW, 21  
 errCodeNoErr, 22  
 shutdownReasonCodeNoError, 24  
 startupReasonCodeButtonPress, 24  
 startupReasonCodeExtCtrl, 24  
 startupReasonCodeMPRestart, 24  
 startupReasonCodePowerOnStartup, 24  
 startupReasonCodeUndefined, 24  
 DeInterlace\_BOB  
 CrossControl, 21  
 DeInterlace\_Even  
 CrossControl, 21  
 DeInterlace\_Odd  
 CrossControl, 21  
 Disabled  
 CrossControl, 21  
 ERR\_AVERAGE\_CALC\_STARTED  
 CrossControl, 22  
 ERR\_BUFFER\_SIZE  
 CrossControl, 21  
 ERR\_CHECKSUM  
 CrossControl, 21  
 ERR\_CODE\_NOT\_EXIST  
 CrossControl, 21  
 ERR\_COMMAND\_FAILED  
 CrossControl, 22  
 ERR\_CREATE\_THREAD  
 CrossControl, 21  
 ERR\_DATATYPE\_MISMATCH  
 CrossControl, 21  
 ERR\_DEVICE\_READ\_DATA\_FAILED  
 CrossControl, 22  
 ERR\_DEVICE\_WRITE\_DATA\_FAILED  
 CrossControl, 22  
 ERR\_EEPROM  
 CrossControl, 22  
 ERR\_I2C\_EXPANDER\_INIT\_FAILED  
 CrossControl, 22  
 ERR\_I2C\_EXPANDER\_READ\_FAILED  
 CrossControl, 22  
 ERR\_I2C\_EXPANDER\_WRITE\_FAILED  
 CrossControl, 22  
 ERR\_INIT\_FAILED  
 CrossControl, 21  
 ERR\_INVALID\_DATA  
 CrossControl, 21  
 ERR\_INVALID\_PARAMETER  
 CrossControl, 21  
 ERR\_IN\_PROGRESS  
 CrossControl, 21  
 ERR\_IOCTL\_FAILED  
 CrossControl, 21  
 ERR\_JIDA\_TEMP  
 CrossControl, 22  
 ERR\_JOIN\_THREAD  
 CrossControl, 22  
 ERR\_MEM\_ALLOC\_FAIL  
 CrossControl, 22  
 ERR\_NEWER\_FPGA\_VERSION\_REQUIRED  
 CrossControl, 22  
 ERR\_NEWER\_FRONT\_VERSION\_REQUIRED

- CrossControl, 22
- ERR\_NEWER\_SS\_VERSION\_REQUIRED
  - CrossControl, 22
- ERR\_NOT\_RUNNING
  - CrossControl, 22
- ERR\_NOT\_SUPPORTED
  - CrossControl, 21
- ERR\_OPEN\_FAILED
  - CrossControl, 21
- ERR\_SUCCESS
  - CrossControl, 21
- ERR\_TELEMATICS\_BT\_NOT\_AVAILABLE
  - CrossControl, 22
- ERR\_TELEMATICS\_GPRS\_NOT\_AVAILABLE
  - CrossControl, 22
- ERR\_TELEMATICS\_GPS\_NOT\_AVAILABLE
  - CrossControl, 22
- ERR\_TELEMATICS\_WLAN\_NOT\_AVAILABLE
  - CrossControl, 22
- ERR\_UNKNOWN\_FEATURE
  - CrossControl, 21
- ERR\_VERIFY\_FAILED
  - CrossControl, 22
- Enabled
  - CrossControl, 21
- FrameExtended
  - CrossControl, 20
- FrameStandard
  - CrossControl, 20
- FrameStandardExtended
  - CrossControl, 20
- Front\_Button\_Enabled
  - CrossControl, 25
- GREEN
  - CrossControl, 20
- MAGENTA
  - CrossControl, 21
- MODE\_CALIBRATION\_13P
  - CrossControl, 20
- MODE\_CALIBRATION\_5P
  - CrossControl, 20
- MODE\_CALIBRATION\_9P
  - CrossControl, 20
- MODE\_NORMAL
  - CrossControl, 20
- MODE\_UNKNOWN
  - CrossControl, 20
- MOUSE\_NEXT\_BOOT
  - CrossControl, 24
- MOUSE\_NOW
  - CrossControl, 24
- NoAction
  - CrossControl, 24
- OCD\_OC
  - CrossControl, 23
- OCD\_OK
  - CrossControl, 23
- OCD\_POWER\_OFF
  - CrossControl, 23
- OnOff\_Signal\_Enabled
  - CrossControl, 25
- RED
  - CrossControl, 20
- RangeExtended
  - CrossControl, 23
- RangeStandard
  - CrossControl, 23
- STD\_B\_D\_G\_H\_I\_N\_PAL
  - CrossControl, 26
- STD\_M\_J\_NTSC
  - CrossControl, 26
- STD\_M\_PAL
  - CrossControl, 26
- STD\_NTSC
  - CrossControl, 26
- STD\_PAL
  - CrossControl, 26
- STD\_SECAM
  - CrossControl, 26
- SamplingModeAuto
  - CrossControl, 23
- SamplingModeExtended
  - CrossControl, 23
- SamplingModeStandard
  - CrossControl, 23
- TEMP\_BACKPLANE
  - CrossControl, 23
- TEMP\_BOARD
  - CrossControl, 22
- TEMP\_BOX
  - CrossControl, 22
- TEMP\_CHIPSETS
  - CrossControl, 23
- TEMP\_CPU
  - CrossControl, 22

- TEMP\_ENV
  - CrossControl, 22
- TEMP\_OTHER
  - CrossControl, 23
- TEMP\_VIDEO
  - CrossControl, 23
- TOUCH\_NEXT\_BOOT
  - CrossControl, 24
- TOUCH\_NOW
  - CrossControl, 24
- TS\_CALIBRATION\_MEASUREMENTS
  - CrossControl, 25
- TS\_CALIBTATION\_WIDTH
  - CrossControl, 25
- TS\_DEBOUNCE\_TIME
  - CrossControl, 25
- TS\_DEBOUNCE\_TIMEOUT\_TIME
  - CrossControl, 25
- TS\_DOUBLECLICK\_MAX\_CLICK\_TIME
  - CrossControl, 25
- TS\_DOUBLE\_CLICK\_TIME
  - CrossControl, 25
- TS\_LOW\_LEVEL
  - CrossControl, 25
- TS\_MAX\_RIGHTCLICK\_DISTANCE
  - CrossControl, 25
- TS\_RESTORE\_DEFAULT\_SETTINGS
  - CrossControl, 25
- TS\_RIGHT\_CLICK\_TIME
  - CrossControl, 25
- TS\_UNTOUCHLEVEL
  - CrossControl, 25
- TS\_USE\_DEJITTER
  - CrossControl, 25
- UNDEFINED\_COLOR
  - CrossControl, 21
- UPGRADE\_COMPLETE
  - CrossControl, 26
- UPGRADE\_COMPLETE\_WITH\_ERRORS
  - CrossControl, 26
- UPGRADE\_CONVERTING\_FILE
  - CrossControl, 26
- UPGRADE\_FLASHING
  - CrossControl, 26
- UPGRADE\_INIT
  - CrossControl, 26
- UPGRADE\_PREP\_COM
  - CrossControl, 26
- UPGRADE\_READING\_FILE
  - CrossControl, 26
- UPGRADE\_VERIFYING
  - CrossControl, 26
- VOLTAGE\_0V9
  - CrossControl, 27
- VOLTAGE\_12V
  - CrossControl, 27
- VOLTAGE\_12VID
  - CrossControl, 27
- VOLTAGE\_1V0
  - CrossControl, 27
- VOLTAGE\_1V05
  - CrossControl, 27
- VOLTAGE\_1V1
  - CrossControl, 27
- VOLTAGE\_1V2
  - CrossControl, 27
- VOLTAGE\_1V3\_PER
  - CrossControl, 27
- VOLTAGE\_1V3\_VDDA
  - CrossControl, 27
- VOLTAGE\_1V5
  - CrossControl, 27
- VOLTAGE\_1V8
  - CrossControl, 27
- VOLTAGE\_1V9
  - CrossControl, 27
- VOLTAGE\_24V
  - CrossControl, 27
- VOLTAGE\_24VIN
  - CrossControl, 27
- VOLTAGE\_24V\_BACKUP
  - CrossControl, 27
- VOLTAGE\_2V5
  - CrossControl, 27
- VOLTAGE\_3V3
  - CrossControl, 27
- VOLTAGE\_5V
  - CrossControl, 27
- VOLTAGE\_5VSTB
  - CrossControl, 27
- VOLTAGE\_VREF\_INT
  - CrossControl, 27
- VOLTAGE\_VTFT
  - CrossControl, 27
- YELLOW
  - CrossControl, 21
- ABOUTHANDLE

- CrossControl, 18
- ADCHANDLE
  - CrossControl, 18
- AUXVERSIONHANDLE
  - CrossControl, 18
- About\_getAddOnHWversion
  - CrossControl, 27
- About\_getAddOnManufacturingDate
  - CrossControl, 28
- About\_getAddOnPCBArt
  - CrossControl, 28
- About\_getAddOnPCBSerial
  - CrossControl, 29
- About\_getDisplayResolution
  - CrossControl, 29
- About\_getFrontPcbRev
  - CrossControl, 29
- About\_getIOExpanderValue
  - CrossControl, 30
- About\_getIsAnybusMounted
  - CrossControl, 30
- About\_getIsBTMounted
  - CrossControl, 30
- About\_getIsDisplayAvailable
  - CrossControl, 31
- About\_getIsGPRSMounted
  - CrossControl, 31
- About\_getIsGPSMounted
  - CrossControl, 32
- About\_getIsIOExpanderMounted
  - CrossControl, 32
- About\_getIsTouchScreenAvailable
  - CrossControl, 32
- About\_getIsWLANMounted
  - CrossControl, 33
- About\_getMainHWversion
  - CrossControl, 33
- About\_getMainManufacturingDate
  - CrossControl, 34
- About\_getMainPCBArt
  - CrossControl, 34
- About\_getMainPCBSerial
  - CrossControl, 35
- About\_getMainProdArtNr
  - CrossControl, 35
- About\_getMainProdRev
  - CrossControl, 36
- About\_getNrOfCANConnections
  - CrossControl, 36
- About\_getNrOfDigIOConnections
  - CrossControl, 36
- About\_getNrOfETHConnections
  - CrossControl, 37
- About\_getNrOfSerialConnections
  - CrossControl, 37
- About\_getNrOfUSBConnections
  - CrossControl, 38
- About\_getNrOfVideoConnections
  - CrossControl, 38
- About\_getUnitSerial
  - CrossControl, 39
- About\_hasOsBooted
  - CrossControl, 39
- About\_release
  - CrossControl, 40
- Above80RunTime
  - CrossControl::TimerType, 135
- Adc\_getVoltage
  - CrossControl, 40
- Adc\_release
  - CrossControl, 40
- AuxVersion\_getCCAuxDrvVersion
  - CrossControl, 41
- AuxVersion\_getCCAuxVersion
  - CrossControl, 42
- AuxVersion\_getFPGAVersion
  - CrossControl, 42
- AuxVersion\_getFrontVersion
  - CrossControl, 43
- AuxVersion\_getOSVersion
  - CrossControl, 44
- AuxVersion\_getSSVersion
  - CrossControl, 44
- AuxVersion\_release
  - CrossControl, 45
- BACKLIGHTHANDLE
  - CrossControl, 18
- BUZZERHANDLE
  - CrossControl, 18
- Backlight\_getAutomaticBLFilter
  - CrossControl, 45
- Backlight\_getAutomaticBLParams
  - CrossControl, 46
- Backlight\_getAutomaticBLStatus
  - CrossControl, 46
- Backlight\_getIntensity
  - CrossControl, 47
- Backlight\_getLedDimming
  - CrossControl, 47
- Backlight\_getStatus



- CrossControl, 47
- Backlight\_release
  - CrossControl, 48
- Backlight\_setAutomaticBLFilter
  - CrossControl, 48
- Backlight\_setAutomaticBLParams
  - CrossControl, 49
- Backlight\_setIntensity
  - CrossControl, 49
- Backlight\_setLedDimming
  - CrossControl, 50
- Backlight\_startAutomaticBL
  - CrossControl, 50
- Backlight\_stopAutomaticBL
  - CrossControl, 50
- ButtonPowerTransitionStatus
  - CrossControl, 19
- Buzzer\_buzze
  - CrossControl, 51
- Buzzer\_getFrequency
  - CrossControl, 51
- Buzzer\_getTrigger
  - CrossControl, 51
- Buzzer\_getVolume
  - CrossControl, 52
- Buzzer\_release
  - CrossControl, 52
- Buzzer\_setFrequency
  - CrossControl, 53
- Buzzer\_setTrigger
  - CrossControl, 53
- Buzzer\_setVolume
  - CrossControl, 54
- CANSETTINGHANDLE
  - CrossControl, 18
- CCAuxColor
  - CrossControl, 20
- CCStatus
  - CrossControl, 21
- CONFIGHANDLE
  - CrossControl, 18
- CalibrationConfigParam
  - CrossControl, 19
- CalibrationModeSettings
  - CrossControl, 20
- CanFrameType
  - CrossControl, 20
- CanSetting\_getBaudrate
  - CrossControl, 54
- CanSetting\_getFrameType
  - CrossControl, 55
- CanSetting\_release
  - CrossControl, 55
- CanSetting\_setBaudrate
  - CrossControl, 55
- CanSetting\_setFrameType
  - CrossControl, 56
- Config\_getCanStartupPowerConfig
  - CrossControl, 56
- Config\_getExtFanStartupPowerConfig
  - CrossControl, 57
- Config\_getExtOnOffSigTrigTime
  - CrossControl, 57
- Config\_getFrontBtnTrigTime
  - CrossControl, 57
- Config\_getHeatingTempLimit
  - CrossControl, 58
- Config\_getLongButtonPressAction
  - CrossControl, 58
- Config\_getOnOffSigAction
  - CrossControl, 58
- Config\_getPowerOnStartup
  - CrossControl, 59
- Config\_getShortButtonPressAction
  - CrossControl, 59
- Config\_getStartupTriggerConfig
  - CrossControl, 59
- Config\_getStartupVoltageConfig
  - CrossControl, 60
- Config\_getSuspendMaxTime
  - CrossControl, 60
- Config\_getTFTMirror
  - CrossControl, 61
- Config\_getTFTMode
  - CrossControl, 61
- Config\_getTFTScan
  - CrossControl, 61
- Config\_getVideoStartupPowerConfig
  - CrossControl, 61
- Config\_release
  - CrossControl, 62
- Config\_setCanStartupPowerConfig
  - CrossControl, 62
- Config\_setExtFanStartupPowerConfig
  - CrossControl, 62
- Config\_setExtOnOffSigTrigTime
  - CrossControl, 63
- Config\_setFrontBtnTrigTime
  - CrossControl, 63
- Config\_setHeatingTempLimit
  - CrossControl, 63

- CrossControl, 63
- Config\_setLongButtonPressAction
  - CrossControl, 64
- Config\_setOnOffSigAction
  - CrossControl, 64
- Config\_setPowerOnStartup
  - CrossControl, 65
- Config\_setShortButtonPressAction
  - CrossControl, 65
- Config\_setStartupTriggerConfig
  - CrossControl, 65
- Config\_setStartupVoltageConfig
  - CrossControl, 66
- Config\_setSuspendMaxTime
  - CrossControl, 66
- Config\_setTFTMirror
  - CrossControl, 66
- Config\_setTFTMode
  - CrossControl, 67
- Config\_setTFTScan
  - CrossControl, 67
- Config\_setVideoStartupPowerConfig
  - CrossControl, 67
- CrossControl, 4
  - ABOUTHANDLE, 18
  - ADCHANDLE, 18
  - AUXVERSIONHANDLE, 18
  - About\_getAddOnHWversion, 27
  - About\_getAddOnManufacturingDate, 28
  - About\_getAddOnPCBArt, 28
  - About\_getAddOnPCBSerial, 29
  - About\_getDisplayResolution, 29
  - About\_getFrontPcbRev, 29
  - About\_getIOExpanderValue, 30
  - About\_getIsAnybusMounted, 30
  - About\_getIsBTMounted, 30
  - About\_getIsDisplayAvailable, 31
  - About\_getIsGPRSMounted, 31
  - About\_getIsGPSSMounted, 32
  - About\_getIsIOExpanderMounted, 32
  - About\_getIsTouchScreenAvailable, 32
  - About\_getIsWLANMounted, 33
  - About\_getMainHWversion, 33
  - About\_getMainManufacturingDate, 34
  - About\_getMainPCBArt, 34
  - About\_getMainPCBSerial, 35
  - About\_getMainProdArtNr, 35
  - About\_getMainProdRev, 36
  - About\_getNrOfCANConnections, 36
  - About\_getNrOfDigIOConnections, 36
  - About\_getNrOfETHConnections, 37
  - About\_getNrOfSerialConnections, 37
  - About\_getNrOfUSBConnections, 38
  - About\_getNrOfVideoConnections, 38
  - About\_getUnitSerial, 39
  - About\_hasOsBooted, 39
  - About\_release, 40
  - Adc\_getVoltage, 40
  - Adc\_release, 40
  - AuxVersion\_getCCAuxDrvVersion, 41
  - AuxVersion\_getCCAuxVersion, 42
  - AuxVersion\_getFPGAVersion, 42
  - AuxVersion\_getFrontVersion, 43
  - AuxVersion\_getOSVersion, 44
  - AuxVersion\_getSSVersion, 44
  - AuxVersion\_release, 45
  - BACKLIGHTHANDLE, 18
  - BUZZERHANDLE, 18
  - Backlight\_getAutomaticBLFilter, 45
  - Backlight\_getAutomaticBLParams, 46
  - Backlight\_getAutomaticBLStatus, 46
  - Backlight\_getIntensity, 47
  - Backlight\_getLedDimming, 47
  - Backlight\_getStatus, 47
  - Backlight\_release, 48
  - Backlight\_setAutomaticBLFilter, 48
  - Backlight\_setAutomaticBLParams, 49
  - Backlight\_setIntensity, 49
  - Backlight\_setLedDimming, 50
  - Backlight\_startAutomaticBL, 50
  - Backlight\_stopAutomaticBL, 50
  - ButtonPowerTransitionStatus, 19
  - Buzzer\_buzze, 51
  - Buzzer\_getFrequency, 51
  - Buzzer\_getTrigger, 51
  - Buzzer\_getVolume, 52
  - Buzzer\_release, 52
  - Buzzer\_setFrequency, 53
  - Buzzer\_setTrigger, 53
  - Buzzer\_setVolume, 54
  - CANSETTINGHANDLE, 18
  - CCAuxColor, 20
  - CCStatus, 21
  - CONFIGHANDLE, 18
  - CalibrationConfigParam, 19
  - CalibrationModeSettings, 20
  - CanFrameType, 20
  - CanSetting\_getBaudrate, 54
  - CanSetting\_getFrameType, 55

- CanSetting\_release, 55
- CanSetting\_setBaudrate, 55
- CanSetting\_setFrameType, 56
- Config\_getCanStartupPowerConfig, 56
- Config\_getExtFanStartupPowerConfig, 57
- Config\_getExtOnOffSigTrigTime, 57
- Config\_getFrontBtnTrigTime, 57
- Config\_getHeatingTempLimit, 58
- Config\_getLongButtonPressAction, 58
- Config\_getOnOffSigAction, 58
- Config\_getPowerOnStartup, 59
- Config\_getShortButtonPressAction, 59
- Config\_getStartupTriggerConfig, 59
- Config\_getStartupVoltageConfig, 60
- Config\_getSuspendMaxTime, 60
- Config\_getTFTMirror, 61
- Config\_getTFTMode, 61
- Config\_getTFTScan, 61
- Config\_getVideoStartupPowerConfig, 61
- Config\_release, 62
- Config\_setCanStartupPowerConfig, 62
- Config\_setExtFanStartupPowerConfig, 62
- Config\_setExtOnOffSigTrigTime, 63
- Config\_setFrontBtnTrigTime, 63
- Config\_setHeatingTempLimit, 63
- Config\_setLongButtonPressAction, 64
- Config\_setOnOffSigAction, 64
- Config\_setPowerOnStartup, 65
- Config\_setShortButtonPressAction, 65
- Config\_setStartupTriggerConfig, 65
- Config\_setStartupVoltageConfig, 66
- Config\_setSuspendMaxTime, 66
- Config\_setTFTMirror, 66
- Config\_setTFTMode, 67
- Config\_setTFTScan, 67
- Config\_setVideoStartupPowerConfig, 67
- DIAGNOSTICHANDLE, 18
- DIGIOHANDLE, 18
- DeInterlaceMode, 21
- Diagnostic\_clearHwErrorStatus, 67
- Diagnostic\_getHwErrorStatus, 68
- Diagnostic\_getMinMaxTemp, 68
- Diagnostic\_getPCBTemp, 68
- Diagnostic\_getPMTemp, 69
- Diagnostic\_getPowerCycles, 69
- Diagnostic\_getSSTemp, 70
- Diagnostic\_getShutDownReason, 69
- Diagnostic\_getStartupReason, 70
- Diagnostic\_getTimer, 70
- Diagnostic\_release, 71
- DigIO\_getDigIO, 71
- DigIO\_release, 72
- DigIO\_setDigIO, 72
- DigitalIn\_1, 130
- DigitalIn\_2, 130
- DigitalIn\_3, 130
- DigitalIn\_4, 130
- FIRMWAREUPGHANDLE, 18
- FRONTLEDHANDLE, 19
- FirmwareUpgrade\_getUpgradeStatus, 73
- FirmwareUpgrade\_release, 73
- FirmwareUpgrade\_shutDown, 73
- FirmwareUpgrade\_startFpgaUpgrade, 74
- FirmwareUpgrade\_startFpgaVerification, 75
- FirmwareUpgrade\_startFrontUpgrade, 75
- FirmwareUpgrade\_startFrontVerification, 76
- FirmwareUpgrade\_startSSUpgrade, 77
- FirmwareUpgrade\_startSSVerification, 78
- FrontLED\_getColor, 79
- FrontLED\_getEnabledDuringStartup, 80
- FrontLED\_getFrontPcbRev, 80
- FrontLED\_getIdleTime, 81
- FrontLED\_getNrOfPulses, 81
- FrontLED\_getOffTime, 81
- FrontLED\_getOnTime, 81
- FrontLED\_getSignal, 82
- FrontLED\_getStandardColor, 82
- FrontLED\_release, 83
- FrontLED\_setColor, 83
- FrontLED\_setEnabledDuringStartup, 83
- FrontLED\_setIdleTime, 84
- FrontLED\_setNrOfPulses, 84
- FrontLED\_setOff, 84
- FrontLED\_setOffTime, 85
- FrontLED\_setOnTime, 85
- FrontLED\_setSignal, 85
- FrontLED\_setStandardColor, 86
- GetAbout, 86

- GetAdc, 87
- GetAuxVersion, 87
- GetBacklight, 88
- GetBuzzer, 88
- GetCanSetting, 88
- GetConfig, 89
- GetDiagnostic, 89
- GetDigIO, 90
- GetErrorStringA, 90
- GetErrorStringW, 90
- GetFirmwareUpgrade, 91
- GetFrontLED, 91
- GetHwErrorStatusStringA, 91
- GetHwErrorStatusStringW, 92
- GetLightsensor, 92
- GetPower, 92
- GetStartupReasonStringA, 93
- GetStartupReasonStringW, 93
- GetTelematics, 93
- GetTouchScreen, 94
- GetTouchScreenCalib, 94
- GetVideo, 94
- JidaSensorType, 22
- LIGHTSENSORHANDLE, 19
- LightSensorOperationRange, 23
- LightSensorSamplingMode, 23
- Lightsensor\_getAverageIlluminance, 95
- Lightsensor\_getIlluminance, 95
- Lightsensor\_getIlluminance2, 96
- Lightsensor\_getOperatingRange, 96
- Lightsensor\_release, 96
- Lightsensor\_setOperatingRange, 97
- Lightsensor\_startAverageCalc, 97
- Lightsensor\_stopAverageCalc, 98
- OCDStatus, 23
- POWERHANDLE, 19
- PowerAction, 23
- Power\_ackPowerRequest, 98
- Power\_getBLPowerStatus, 98
- Power\_getButtonPowerTransitionStatus, 99
- Power\_getCanOCDStatus, 99
- Power\_getCanPowerStatus, 100
- Power\_getExtFanPowerStatus, 101
- Power\_getVideoOCDStatus, 101
- Power\_getVideoPowerStatus, 102
- Power\_release, 102
- Power\_setBLPowerStatus, 103
- Power\_setCanPowerStatus, 103
- Power\_setExtFanPowerStatus, 103
- Power\_setVideoPowerStatus, 104
- TELEMATICSHANDLE, 19
- TOUCHSCREENHANDLE, 19
- TSAAdvancedSettingsParameter, 25
- Telematics\_getBTPowerStatus, 104
- Telematics\_getBTStartupPowerStatus, 105
- Telematics\_getGPRSPowerStatus, 105
- Telematics\_getGPRSStartupPowerStatus, 106
- Telematics\_getGPSAntennaStatus, 107
- Telematics\_getGPSPowerStatus, 107
- Telematics\_getGPSStartupPowerStatus, 108
- Telematics\_getTelematicsAvailable, 108
- Telematics\_getWLANPowerStatus, 109
- Telematics\_getWLANStartupPowerStatus, 110
- Telematics\_release, 110
- Telematics\_setBTPowerStatus, 111
- Telematics\_setBTStartupPowerStatus, 111
- Telematics\_setGPRSPowerStatus, 111
- Telematics\_setGPRSStartupPowerStatus, 112
- Telematics\_setGPSPowerStatus, 112
- Telematics\_setGPSStartupPowerStatus, 112
- Telematics\_setWLANPowerStatus, 112
- Telematics\_setWLANStartupPowerStatus, 113
- TouchScreenCalib\_checkCalibrationPointFinished, 116
- TouchScreenCalib\_getConfigParam, 117
- TouchScreenCalib\_getMode, 117
- TouchScreenCalib\_release, 117
- TouchScreenCalib\_setCalibrationPoint, 117
- TouchScreenCalib\_setConfigParam, 118
- TouchScreenCalib\_setMode, 118
- TouchScreenModeSettings, 24
- TouchScreen\_getAdvancedSetting, 113
- TouchScreen\_getMode, 114
- TouchScreen\_getMouseRightClickTime, 114
- TouchScreen\_release, 115
- TouchScreen\_setAdvancedSetting, 115
- TouchScreen\_setMode, 116

- TouchScreen\_setMouseRightClickTime, 116
- TriggerConf, 24
- UpgradeAction, 26
- VIDEOHANDLE, 19
- VersionType, 19
- Video1Conf, 130
- Video2Conf, 130
- Video3Conf, 130
- Video4Conf, 130
- VideoChannel, 26
- Video\_activateSnapshot, 119
- Video\_createBitmap, 119
- Video\_freeBmpBuffer, 119
- Video\_getActiveChannel, 120
- Video\_getColorKeys, 120
- Video\_getCropping, 121
- Video\_getDeInterlaceMode, 121
- Video\_getDecoderReg, 121
- Video\_getMirroring, 122
- Video\_getRawImage, 122
- Video\_getScaling, 122
- Video\_getStatus, 123
- Video\_getVideoArea, 123
- Video\_getVideoStandard, 123
- Video\_init, 124
- Video\_minimize, 124
- Video\_release, 124
- Video\_restore, 124
- Video\_setActiveChannel, 125
- Video\_setColorKeys, 125
- Video\_setCropping, 125
- Video\_setDeInterlaceMode, 126
- Video\_setDecoderReg, 126
- Video\_setMirroring, 126
- Video\_setScaling, 127
- Video\_setVideoArea, 127
- Video\_showFrame, 128
- Video\_showVideo, 128
- Video\_takeSnapshot, 128
- Video\_takeSnapshotBmp, 128
- Video\_takeSnapshotRaw, 129
- VoltageEnum, 26
- eErr, 21
- hwErrorStatusCodes, 22
- shutDownReasonCodes, 24
- startupReasonCodes, 24
- videoStandard, 26
- CrossControl::BuzzerSetup, 131
  - frequency, 131
  - volume, 131
- CrossControl::FpgaLedTimingType, 131
  - idleTime, 132
  - ledNbr, 132
  - nrOfPulses, 132
  - offTime, 132
  - onTime, 132
- CrossControl::LedColorMixType, 132
  - blue, 133
  - green, 133
  - red, 133
- CrossControl::LedTimingType, 133
  - idleTime, 133
  - nrOfPulses, 133
  - offTime, 133
  - onTime, 133
- CrossControl::TimerType, 134
  - Above80RunTime, 135
  - RunTime40\_60, 135
  - RunTime60\_70, 135
  - RunTime70\_80, 135
  - TotHeatTime, 135
  - TotRunTime, 135
  - TotSuspTime, 135
- CrossControl::UpgradeStatus, 135
  - currentAction, 136
  - errorCode, 136
  - percent, 136
- CrossControl::received\_video, 134
  - received\_framerate, 134
  - received\_height, 134
  - received\_width, 134
- CrossControl::version\_info, 136
  - build, 136
  - major, 136
  - minor, 136
  - release, 137
- CrossControl::video\_dec\_command, 137
  - decoder\_register, 137
  - register\_value, 137
- DIAGNOSTICHANDLE
  - CrossControl, 18
- DIGIOHANDLE
  - CrossControl, 18
- DeInterlaceMode
  - CrossControl, 21
- Diagnostic\_clearHwErrorStatus
  - CrossControl, 67
- Diagnostic\_getHwErrorStatus
  - CrossControl, 68

- Diagnostic\_getMinMaxTemp
  - CrossControl, 68
- Diagnostic\_getPCBTemp
  - CrossControl, 68
- Diagnostic\_getPMTemp
  - CrossControl, 69
- Diagnostic\_getPowerCycles
  - CrossControl, 69
- Diagnostic\_getSSTemp
  - CrossControl, 70
- Diagnostic\_getShutDownReason
  - CrossControl, 69
- Diagnostic\_getStartupReason
  - CrossControl, 70
- Diagnostic\_getTimer
  - CrossControl, 70
- Diagnostic\_release
  - CrossControl, 71
- DigIO\_getDigIO
  - CrossControl, 71
- DigIO\_release
  - CrossControl, 72
- DigIO\_setDigIO
  - CrossControl, 72
- DigitalIn\_1
  - CrossControl, 130
- DigitalIn\_2
  - CrossControl, 130
- DigitalIn\_3
  - CrossControl, 130
- DigitalIn\_4
  - CrossControl, 130
- FIRMWAREUPGHANDLE
  - CrossControl, 18
- FRONTLEDHANDLE
  - CrossControl, 19
- FirmwareUpgrade\_getUpgradeStatus
  - CrossControl, 73
- FirmwareUpgrade\_release
  - CrossControl, 73
- FirmwareUpgrade\_shutDown
  - CrossControl, 73
- FirmwareUpgrade\_startFpgaUpgrade
  - CrossControl, 74
- FirmwareUpgrade\_startFpgaVerification
  - CrossControl, 75
- FirmwareUpgrade\_startFrontUpgrade
  - CrossControl, 75
- FirmwareUpgrade\_startFrontVerification
  - CrossControl, 76
- FirmwareUpgrade\_startSSUpgrade
  - CrossControl, 77
- FirmwareUpgrade\_startSSVerification
  - CrossControl, 78
- FrontLED\_getColor
  - CrossControl, 79
- FrontLED\_getEnabledDuringStartup
  - CrossControl, 80
- FrontLED\_getFrontPcbRev
  - CrossControl, 80
- FrontLED\_getIdleTime
  - CrossControl, 81
- FrontLED\_getNrOfPulses
  - CrossControl, 81
- FrontLED\_getOffTime
  - CrossControl, 81
- FrontLED\_getOnTime
  - CrossControl, 81
- FrontLED\_getSignal
  - CrossControl, 82
- FrontLED\_getStandardColor
  - CrossControl, 82
- FrontLED\_release
  - CrossControl, 83
- FrontLED\_setColor
  - CrossControl, 83
- FrontLED\_setEnabledDuringStartup
  - CrossControl, 83
- FrontLED\_setIdleTime
  - CrossControl, 84
- FrontLED\_setNrOfPulses
  - CrossControl, 84
- FrontLED\_setOff
  - CrossControl, 84
- FrontLED\_setOffTime
  - CrossControl, 85
- FrontLED\_setOnTime
  - CrossControl, 85
- FrontLED\_setSignal
  - CrossControl, 85
- FrontLED\_setStandardColor
  - CrossControl, 86
- GetAbout
  - CrossControl, 86
- GetAdc
  - CrossControl, 87
- GetAuxVersion
  - CrossControl, 87
- GetBacklight
  - CrossControl, 88

- GetBuzzer
  - CrossControl, [88](#)
- GetCanSetting
  - CrossControl, [88](#)
- GetConfig
  - CrossControl, [89](#)
- GetDiagnostic
  - CrossControl, [89](#)
- GetDigIO
  - CrossControl, [90](#)
- GetErrorStringA
  - CrossControl, [90](#)
- GetErrorStringW
  - CrossControl, [90](#)
- GetFirmwareUpgrade
  - CrossControl, [91](#)
- GetFrontLED
  - CrossControl, [91](#)
- GetHwErrorStatusStringA
  - CrossControl, [91](#)
- GetHwErrorStatusStringW
  - CrossControl, [92](#)
- GetLightsensor
  - CrossControl, [92](#)
- GetPower
  - CrossControl, [92](#)
- GetStartupReasonStringA
  - CrossControl, [93](#)
- GetStartupReasonStringW
  - CrossControl, [93](#)
- GetTelematics
  - CrossControl, [93](#)
- GetTouchScreen
  - CrossControl, [94](#)
- GetTouchScreenCalib
  - CrossControl, [94](#)
- GetVideo
  - CrossControl, [94](#)
- JidaSensorType
  - CrossControl, [22](#)
- LIGHTSENSORHANDLE
  - CrossControl, [19](#)
- LightSensorOperationRange
  - CrossControl, [23](#)
- LightSensorSamplingMode
  - CrossControl, [23](#)
- Lightsensor\_getAverageIlluminance
  - CrossControl, [95](#)
- Lightsensor\_getIlluminance
  - CrossControl, [95](#)
- Lightsensor\_getIlluminance2
  - CrossControl, [96](#)
- Lightsensor\_getOperatingRange
  - CrossControl, [96](#)
- Lightsensor\_release
  - CrossControl, [96](#)
- Lightsensor\_setOperatingRange
  - CrossControl, [97](#)
- Lightsensor\_startAverageCalc
  - CrossControl, [97](#)
- Lightsensor\_stopAverageCalc
  - CrossControl, [98](#)
- OCDStatus
  - CrossControl, [23](#)
- POWERHANDLE
  - CrossControl, [19](#)
- PowerAction
  - CrossControl, [23](#)
- Power\_ackPowerRequest
  - CrossControl, [98](#)
- Power\_getBLPowerStatus
  - CrossControl, [98](#)
- Power\_getButtonPowerTransitionStatus
  - CrossControl, [99](#)
- Power\_getCanOCDStatus
  - CrossControl, [99](#)
- Power\_getCanPowerStatus
  - CrossControl, [100](#)
- Power\_getExtFanPowerStatus
  - CrossControl, [101](#)
- Power\_getVideoOCDStatus
  - CrossControl, [101](#)
- Power\_getVideoPowerStatus
  - CrossControl, [102](#)
- Power\_release
  - CrossControl, [102](#)
- Power\_setBLPowerStatus
  - CrossControl, [103](#)
- Power\_setCanPowerStatus
  - CrossControl, [103](#)
- Power\_setExtFanPowerStatus
  - CrossControl, [103](#)
- Power\_setVideoPowerStatus
  - CrossControl, [104](#)
- RunTime40\_60
  - CrossControl::TimerType, [135](#)
- RunTime60\_70
  - CrossControl::TimerType, [135](#)
- RunTime70\_80
  - CrossControl::TimerType, [135](#)



- 
- TELEMATICSHANDLE
    - CrossControl, [19](#)
  - TOUCHSCREENHANDLE
    - CrossControl, [19](#)
  - TSAAdvancedSettingsParameter
    - CrossControl, [25](#)
  - Telematics\_getBTPowerStatus
    - CrossControl, [104](#)
  - Telematics\_getBTStartupPowerStatus
    - CrossControl, [105](#)
  - Telematics\_getGPRSPowerStatus
    - CrossControl, [105](#)
  - Telematics\_getGPRSStartupPowerStatus
    - CrossControl, [106](#)
  - Telematics\_getGPSAntennaStatus
    - CrossControl, [107](#)
  - Telematics\_getGPSPowerStatus
    - CrossControl, [107](#)
  - Telematics\_getGPSStartupPowerStatus
    - CrossControl, [108](#)
  - Telematics\_getTelematicsAvailable
    - CrossControl, [108](#)
  - Telematics\_getWLANPowerStatus
    - CrossControl, [109](#)
  - Telematics\_getWLANStartupPowerStatus
    - CrossControl, [110](#)
  - Telematics\_release
    - CrossControl, [110](#)
  - Telematics\_setBTPowerStatus
    - CrossControl, [111](#)
  - Telematics\_setBTStartupPowerStatus
    - CrossControl, [111](#)
  - Telematics\_setGPRSPowerStatus
    - CrossControl, [111](#)
  - Telematics\_setGPRSStartupPowerStatus
    - CrossControl, [112](#)
  - Telematics\_setGPSPowerStatus
    - CrossControl, [112](#)
  - Telematics\_setGPSStartupPowerStatus
    - CrossControl, [112](#)
  - Telematics\_setWLANPowerStatus
    - CrossControl, [112](#)
  - Telematics\_setWLANStartupPowerStatus
    - CrossControl, [113](#)
  - TotHeatTime
    - CrossControl::TimerType, [135](#)
  - TotRunTime
    - CrossControl::TimerType, [135](#)
  - TotSuspTime
    - CrossControl::TimerType, [135](#)
  - TouchScreenCalib\_checkCalibrationPoint-
    - Finished
    - CrossControl, [116](#)
  - TouchScreenCalib\_getConfigParam
    - CrossControl, [117](#)
  - TouchScreenCalib\_getMode
    - CrossControl, [117](#)
  - TouchScreenCalib\_release
    - CrossControl, [117](#)
  - TouchScreenCalib\_setCalibrationPoint
    - CrossControl, [117](#)
  - TouchScreenCalib\_setConfigParam
    - CrossControl, [118](#)
  - TouchScreenCalib\_setMode
    - CrossControl, [118](#)
  - TouchScreenModeSettings
    - CrossControl, [24](#)
  - TouchScreen\_getAdvancedSetting
    - CrossControl, [113](#)
  - TouchScreen\_getMode
    - CrossControl, [114](#)
  - TouchScreen\_getMouseRightClickTime
    - CrossControl, [114](#)
  - TouchScreen\_release
    - CrossControl, [115](#)
  - TouchScreen\_setAdvancedSetting
    - CrossControl, [115](#)
  - TouchScreen\_setMode
    - CrossControl, [116](#)
  - TouchScreen\_setMouseRightClickTime
    - CrossControl, [116](#)
  - TriggerConf
    - CrossControl, [24](#)
  - UpgradeAction
    - CrossControl, [26](#)
  - VIDEOHANDLE
    - CrossControl, [19](#)
  - VersionType
    - CrossControl, [19](#)
  - Video1Conf
    - CrossControl, [130](#)
  - Video2Conf
    - CrossControl, [130](#)
  - Video3Conf
    - CrossControl, [130](#)
  - Video4Conf
    - CrossControl, [130](#)
  - VideoChannel
    - CrossControl, [26](#)
  - Video\_activateSnapshot
-



- CrossControl, 119
- Video\_createBitmap
  - CrossControl, 119
- Video\_freeBmpBuffer
  - CrossControl, 119
- Video\_getActiveChannel
  - CrossControl, 120
- Video\_getColorKeys
  - CrossControl, 120
- Video\_getCropping
  - CrossControl, 121
- Video\_getDeInterlaceMode
  - CrossControl, 121
- Video\_getDecoderReg
  - CrossControl, 121
- Video\_getMirroring
  - CrossControl, 122
- Video\_getRawImage
  - CrossControl, 122
- Video\_getScaling
  - CrossControl, 122
- Video\_getStatus
  - CrossControl, 123
- Video\_getVideoArea
  - CrossControl, 123
- Video\_getVideoStandard
  - CrossControl, 123
- Video\_init
  - CrossControl, 124
- Video\_minimize
  - CrossControl, 124
- Video\_release
  - CrossControl, 124
- Video\_restore
  - CrossControl, 124
- Video\_setActiveChannel
  - CrossControl, 125
- Video\_setColorKeys
  - CrossControl, 125
- Video\_setCropping
  - CrossControl, 125
- Video\_setDeInterlaceMode
  - CrossControl, 126
- Video\_setDecoderReg
  - CrossControl, 126
- Video\_setMirroring
  - CrossControl, 126
- Video\_setScaling
  - CrossControl, 127
- Video\_setVideoArea
  - CrossControl, 127
- Video\_showFrame
  - CrossControl, 128
- Video\_showVideo
  - CrossControl, 128
- Video\_takeSnapshot
  - CrossControl, 128
- Video\_takeSnapshotBmp
  - CrossControl, 128
- Video\_takeSnapshotRaw
  - CrossControl, 129
- VoltageEnum
  - CrossControl, 26
- blue
  - CrossControl::LedColorMixType, 133
- build
  - CrossControl::version\_info, 136
- currentAction
  - CrossControl::UpgradeStatus, 136
- decoder\_register
  - CrossControl::video\_dec\_command, 137
- eErr
  - CrossControl, 21
- errCodeNoErr
  - CrossControl, 22
- errorCode
  - CrossControl::UpgradeStatus, 136
- fixedIncludeFiles/About.h, 138
- fixedIncludeFiles/Adc.h, 140
- fixedIncludeFiles/AuxVersion.h, 140
- fixedIncludeFiles/Backlight.h, 141
- fixedIncludeFiles/Buzzer.h, 142
- fixedIncludeFiles/CCAuxErrors.h, 143
- fixedIncludeFiles/CCAuxTypes.h, 144
- fixedIncludeFiles/CCPlatform.h, 146
- fixedIncludeFiles/CanSetting.h, 143
- fixedIncludeFiles/Config.h, 146
- fixedIncludeFiles/Diagnostic.h, 148
- fixedIncludeFiles/DiagnosticCodes.h, 149
- fixedIncludeFiles/DigIO.h, 150
- fixedIncludeFiles/FirmwareUpgrade.h, 150
- fixedIncludeFiles/FrontLED.h, 151
- fixedIncludeFiles/Lightsensor.h, 153
- fixedIncludeFiles/Power.h, 153
- fixedIncludeFiles/Telematics.h, 154
- fixedIncludeFiles/TouchScreen.h, 156

- fixedIncludeFiles/TouchScreenCalib.h, 157
- fixedIncludeFiles/Video.h, 158
- frequency
  - CrossControl::BuzzerSetup, 131
- green
  - CrossControl::LedColorMixType, 133
- hwErrorStatusCodes
  - CrossControl, 22
- idleTime
  - CrossControl::FpgaLedTimingType, 132
  - CrossControl::LedTimingType, 133
- ledNbr
  - CrossControl::FpgaLedTimingType, 132
- major
  - CrossControl::version\_info, 136
- minor
  - CrossControl::version\_info, 136
- nrOfPulses
  - CrossControl::FpgaLedTimingType, 132
  - CrossControl::LedTimingType, 133
- offTime
  - CrossControl::FpgaLedTimingType, 132
  - CrossControl::LedTimingType, 133
- onTime
  - CrossControl::FpgaLedTimingType, 132
  - CrossControl::LedTimingType, 133
- percent
  - CrossControl::UpgradeStatus, 136
- received\_framerate
  - CrossControl::received\_video, 134
- received\_height
  - CrossControl::received\_video, 134
- received\_width
  - CrossControl::received\_video, 134
- red
  - CrossControl::LedColorMixType, 133
- register\_value
  - CrossControl::video\_dec\_command, 137
- release
  - CrossControl::version\_info, 137
- shutDownReasonCodes
  - CrossControl, 24
- shutdownReasonCodeNoError
  - CrossControl, 24
- startupReasonCodeButtonPress
  - CrossControl, 24
- startupReasonCodeExtCtrl
  - CrossControl, 24
- startupReasonCodeMPRestart
  - CrossControl, 24
- startupReasonCodePowerOnStartup
  - CrossControl, 24
- startupReasonCodeUndefined
  - CrossControl, 24
- startupReasonCodes
  - CrossControl, 24
- videoStandard
  - CrossControl, 26
- volume
  - CrossControl::BuzzerSetup, 131