cr·ssc·ntr·l

CCAux

2.2.0.0

Mon Jan 28 2013

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

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_IOCTRL_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_getIsGPSMounted (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 *bottomRigthX, unsigned short *bottomRigthY)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_getVideoStandard (VIDEOHANDLE, videoStandard *standard)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_getStatus (VIDEOHANDLE, unsigned char *status)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_setScaling (VIDEOHANDLE, float x, float y)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_getScaling (VIDEOHANDLE, float *x, float *y)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_activateSnapshot (VIDEOHANDLE, bool activate)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_takeSnapshot (VIDEOHANDLE, const char *path, bool bInterlaced)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_takeSnapshotRaw (VIDEOHANDLE, char *rawImgBuffer, unsigned long rawImgBuffSize, bool bInterlaced)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_takeSnapshotBmp (VIDEOHANDLE, char **bmpBuffer, unsigned long *bmpBufSize, bool bInterlaced, bool bNTSCFormat)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_createBitmap (VIDEOHANDLE, char **bmpBuffer, unsigned long *bmpBufSize, const char *rawImgBuffer, unsigned long rawImgBufSize, bool bInterlaced, bool bNTSCFormat)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_freeBmpBuffer (VIDEOHANDLE, char *bmpBuffer)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_minimize (VIDEOHANDLE)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_restore (VIDEOHANDLE)

- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_setDecoderReg (VIDEOHANDLE, unsigned char decoderRegister, unsigned char registerValue)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_getDecoderReg (VIDEOHANDLE, unsigned char decoderRegister, unsigned char *registerValue)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_setCropping (VIDEOHANDLE, unsigned char top, unsigned char left, unsigned char bottom, unsigned char right)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_getCropping (VIDEOHANDLE, unsigned char *top, unsigned char *left, unsigned char *bottom, unsigned char *right)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_showFrame (VIDEOHANDLE)

Variables

- const unsigned char Video1Conf = (1 << 0)
- const unsigned char Video2Conf = (1 << 1)
- const unsigned char Video3Conf = (1 << 2)
- const unsigned char Video4Conf = (1 << 3)
- const unsigned char DigitalIn_1 = (1 << 0)
- const unsigned char DigitalIn_2 = (1 << 1)
- const unsigned char DigitalIn_3 = (1 << 2)
- const unsigned char DigitalIn_4 = (1 << 3)

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_IOCTRL_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 Initalization 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::OCDStatus

Overcurrent detection status.

Enumerator:

OCD_OK Normal operation, no overcurrent condition detected

OCD_OC Overcurrent has been detected, power has therefore been turned off, but may be functioning again if the problem disappeared after re-test

OCD_POWER_OFF Overcurrent has been detected, power has been permanently turned off

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

buff	Text output buffer.
len	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

buff	Text output buffer.
len	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

buff	Text output buffer.
length	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

buff	Text output buffer.
len	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

buff	Text output buffer.
len	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

major	The major pcb revision.
minor	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

is	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

mounted	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

mounted	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

available	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

mounted	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

mounted	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

mounted	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

available	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

mounted	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

buff	Text output buffer.
len	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

buff	Text output buffer.
len	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

buff	Text output buffer.
length	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

buff	Text output buffer.
len	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

buff	Text output buffer.
len	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

buff	Text output buffer.
len	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

NrOf-Connections	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

NrOfConnections	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

NrOfConnections	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

NrOf-Connections	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

NrOf-Connections	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

NrOf-Connections	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

buff	Text output buffer.
len	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

boot-Complete	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

selection	The type of voltage to get.
value	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_getCCAux-Version (Win32).

Parameters

major	Major version number
minor	Minor version number
release	Release version number
build	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

major	Major version number
minor	Minor version number
release	Release version number
build	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

major	Major version number
minor	Minor version number
release	Release version number
build	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

major	Major version number
minor	Minor version number
release	Release version number
build	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

major	Major version number
minor	Minor version number
release	Release version number
build	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

major	Major version number
minor	Minor version number
release	Release version number
build	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

averageWndSize	The average window size in nr of samples.
rejectWndSize	The reject window size in nr of samples.
rejectDeltaInLux	The reject delta in lux.
mode	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

bSoftTransitions	Soft transitions used?
k	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

status	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

intensity	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 dimms 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

status	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

status	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

averageWndSize	The average window size in nr of samples.
rejectWndSize	The reject window size in nr of samples.
rejectDeltaInLux	The reject delta in lux.
mode	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

bSoftTransitions	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

intensity	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 dimms 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

status	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

time	Time (ms) to buzz.
blocking	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

frequency	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

trigger	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

volume	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_beeps(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

frequency	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

trigger	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

volume	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

net	CAN net (1-4) to get settings for.
baudrate	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

net	CAN net (1-4) to get settings for.
frameType	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

net	CAN net (1-4).
baudrate	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

net	CAN net (1-4).
frameType	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

status	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

status	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

triggertime	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

triggertime	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

temperature	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

action	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

action	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 StartupTrigger-Config still applies if the unit is shut down or suspended, without removing the power supply.

Parameters

status	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

action	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

config	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

voltage	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

maxTime	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

config	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

status	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

status	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

triggertime	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

triggertime	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

temperature	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

action	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

action	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 StartupTrigger-Config still applies if the unit is shut down or suspended, without removing the power supply.

Parameters

status	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

action	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

conf	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

voltage	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

maxTime	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

config	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

errorCode	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

minTemp	Minimum measured PCB temperature.
maxTemp	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

temperature	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

index	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.
temperature	Temperature in degrees Celsius.
jst	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

powerCycles	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

reason	See DiagnosticCodes.h 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

temperature	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

reason	See DiagnosticCodes.h 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

times	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

status	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

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

state	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

status	The current status of the upgrade operation.
blocking	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

filename	Path and filename to the .mcs file to program.
blocking	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

filename	Path and filename to the .mcs file to verify against.
blocking	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

filename	Path and filename to the .hex file to program.
blocking	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

filename	Path and filename to the .hex file to verify against.
blocking	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

filename	Path and filename to the .hex file to program.
blocking	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

filename	Path and filename to the .hex file to verify against.
blocking	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

red	Red color intensity 0-0x0F.
green	Green color intensity 0-0x0F.
blue	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

status	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

major	The major pcb revision.
minor	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

idleTime	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

nrOfPulses	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

offTime	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

onTime	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 setSignal. This is due to precision-loss in approximations.

Parameters

frequency	LED blink frequency (0.2-50 Hz).
dutyCycle	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

color	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

red	Red color intensity 0-0x0F.
green	Green color intensity 0-0x0F.
blue	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

status	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

idleTime	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

nrOfPulses	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

offTime	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

onTime	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

frequency	LED blink frequency (0.2-50 Hz).
dutyCycle	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

color	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

errCode	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

errCode	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

errCode	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

errCode	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

code	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

code	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

value	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

value	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

value	Illuminance value (Lux).
ch0	Channel0 value.
ch1	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

range	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

range	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

average-WndSize	The average window size in nr of samples.
rejectWnd-Size	The reject window size in nr of samples.
rejectDelta-InLux	The reject delta in lux.
mode	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 backlihgt 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 CrossControl 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

status	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

status	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

status	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

status	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

status	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

status	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

videoStatus	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

status	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

status	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

status	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

status	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

status	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

status	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

status	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

status	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

status	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

status	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

status	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

status	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

status	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

status	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

status	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

status	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

status	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

status	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

status	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

status	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

status	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

status	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

param	The setting to get.
data	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

config	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

time	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 TSAdvancedSettingsParameter for a description of the parameters.

Parameters

param	The setting to set.
data	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

config	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

time	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

finished	Is current point finished?
pointNr	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

param	Config parameter
value	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

mode	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

pointNr	Calibartion 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

param	Config parameter
value	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

mode	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

activate	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

bmpBuffer	Bitmap ram buffer allocated by the API, has to be deallocated with freeBmpBuffer() by the application.
bmpBufSize	Size of the returned bitmap buffer.
rawImg-Buffer	Raw image buffer from takeSnapShotRaw.
rawImgBuf-Size	Size of the raw image buffer.
bInterlaced	Interlaced, if true the bitmap only contains every second line in the image, to save bandwidth.
bNTSC-Format	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

bmpBuffer	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

channel	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

rKey	Red value.
gKey	Green value.
bKey	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

top	Crop top (lines).
left	Crop left (lines).
bottom	Crop bottom (lines).
right	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

decoder-Register	Decoder Register Address.
register-Value	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

mode	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

mode	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

width	Width of raw image.
height	Height of raw moving image, snapshot are 4 bytes less.
frameRate	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

x	Horizontal scaling (0.25-4).
y	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

status	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

topLeftX	Top left X coordinate on screen.
topLeftY	Top left Y coordinate on screen.
bottom-RigthX	Bottom right X coordinate on screen.
bottom-RigthY	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

standard	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

deviceNr	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

channel	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

rKey	Red key value.
gKey	Green key value.
bKey	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 horisontal cropping: The number of pixels of active video must be an even number. The parame-

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 allways cropped together.

Parameters

top	Crop top (0-255 lines).
left	Crop left (0-127 lines).
bottom	Crop bottom (0-255 lines).
right	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

decoder-Register	Decoder Register Address.
register-Value	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

mode	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

mode	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

x	Horizontal scaling (0.25-4).
y	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

topLeftX	Top left X coordinate on screen.
topLeftY	Top left Y coordinate on screen.
bottom-RightX	Bottom right X coordinate on screen.
bottom-RightY	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

show	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

path	The file path to where the image should be stored.
bInterlaced	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

bmpBuffer	Bitmap ram buffer allocated by the API, has to be deallocated with freeBmpBuffer() by the application.
bmpBufSize	Size of the returned bitmap buffer.
bInterlaced	If true the bitmap only contains every second line in the image, to save bandwidth.
bNTSCFormat	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

rawImgBuffer	Buffer for image to be stored in.
rawImgBuffSize	Size of the buffer.
bInterlaced	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 CrossControl::About_getNrOfETHConnections (ABOUTHANDLE, unsigned char *NrOfConnections)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getNrOfCANConnections (ABOUTHANDLE, unsigned char *NrOfConnections)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getNrOfVideoConnections (ABOUTHANDLE, unsigned char *NrOfConnections)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getNrOfUSBConnections (ABOUTHANDLE, unsigned char *NrOfConnections)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getNrOfSerialConnections (ABOUTHANDLE, unsigned char *NrOfConnections)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getNrOfDigIOConnections (ABOUTHANDLE, unsigned char *NrOfConnections)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getIsDisplayAvailable (ABOUTHANDLE, bool *available)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getIsTouchScreenAvailable (ABOUTHANDLE, bool *available)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getDisplayResolution (ABOUTHANDLE, char *buff, int len)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getAddOnPCBSerial (ABOUTHANDLE, char *buff, int len)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getAddOnPCBArt (ABOUTHANDLE, char *buff, int length)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getAddOnManufacturingDate (ABOUTHANDLE, char *buff, int len)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getAddOnHWversion (ABOUTHANDLE, char *buff, int len)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getIsWLANMounted (ABOUTHANDLE, bool *mounted)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getIsGPSMounted (ABOUTHANDLE, bool *mounted)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getIsGPRSMounted (ABOUTHANDLE, bool *mounted)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getIsBTMounted (ABOUTHANDLE, bool *mounted)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getFrontPcbRev (ABOUTHANDLE, unsigned char *major, unsigned char *minor)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getIsIOExpanderMounted (ABOUTHANDLE, bool *mounted)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::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_CONV 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_DATATYP-E_MISMATCH = 4, CrossControl::ERR_CODE_NOT_EXIST = 5, CrossControl::ERR_BUFFER_SIZE = 6, CrossControl::ERR_IOCTRL_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 CrossControl::Config_getSuspendMaxTime (CONFIGHANDLE, unsigned short *maxTime)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_getCanStartupPowerConfig (CONFIGHANDLE, CCStatus *status)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_getVideoStartupPowerConfig (CONFIGHANDLE, unsigned char *config)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_getExtFanStartupPowerConfig (CONFIGHANDLE, CCStatus *status)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_getStartupVoltageConfig (CONFIGHANDLE, double *voltage)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_getHeatingTempLimit (CONFIGHANDLE, signed short *temperature)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_getPowerOnStartup (CONFIGHANDLE, CCStatus *status)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_setStartupTriggerConfig (CONFIGHANDLE, TriggerConf conf)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_setShortButtonPressAction (CONFIGHANDLE, PowerAction action)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_setLongButtonPressAction (CONFIGHANDLE, PowerAction action)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_setOnOffSigAction (CONFIGHANDLE, PowerAction action)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_setFrontBtnTrigTime (CONFIGHANDLE, unsigned short triggertime)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_setExtOnOffSigTrigTime (CONFIGHANDLE, unsigned long triggertime)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_setSuspendMaxTime (CONFIGHANDLE, unsigned short maxTime)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_setCanStartupPowerConfig (CONFIGHANDLE, CCStatus status)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_setVideoStartupPowerConfig (CONFIGHANDLE, unsigned char config)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_setExtFanStartupPowerConfig (CONFIGHANDLE, CCStatus status)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::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 errCode)
- EXTERN_C CCAUXDLL_API wchar_t const *CCAUXDLL_CALLING_CONV CrossControl::GetHwErrorStatusStringW (unsigned short errCode)

- 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, UpgradeStatus *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 CrossControl::FrontLED_getSignal (FRONTLEDHANDLE, double *frequency, unsigned char *dutyCycle)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FrontLED_getOnTime (FRONTLEDHANDLE, unsigned char *onTime)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FrontLED_getOffTime (FRONTLEDHANDLE, unsigned char *offTime)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FrontLED_getIdleTime (FRONTLEDHANDLE, unsigned char *idleTime)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FrontLED_getNrOfPulses (FRONTLEDHANDLE, unsigned char *nrOfPulses)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FrontLED_getColor (FRONTLEDHANDLE, unsigned char *red, unsigned char *green, unsigned char *blue)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FrontLED_getStandardColor (FRONTLEDHANDLE, CCAuxColor *color)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FrontLED_getEnabledDuringStartup (FRONTLEDHANDLE, CCStatus *status)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FrontLED_setSignal (FRONTLEDHANDLE, double frequency, unsigned char dutyCycle)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FrontLED_setOnTime (FRONTLEDHANDLE, unsigned char onTime)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FrontLED_setOffTime (FRONTLEDHANDLE, unsigned char offTime)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FrontLED_setIdleTime (FRONTLEDHANDLE, unsigned char idleTime)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FrontLED_setNrOfPulses (FRONTLEDHANDLE, unsigned char nrOfPulses)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FrontLED_setColor (FRONTLEDHANDLE, unsigned char red, unsigned char green, unsigned char blue)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FrontLED_setStandardColor (FRONTLEDHANDLE, CCAuxColor color)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FrontLED_setOff (FRONTLEDHANDLE)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FrontLED_setEnabledDuringStartup (FRONTLEDHANDLE, CCStatus status)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::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 CrossControl::Video_setMirroring (VIDEOHANDLE, CCStatus mode)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_getMirroring (VIDEOHANDLE, CCStatus *mode)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_setActiveChannel (VIDEOHANDLE, VideoChannel channel)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_getActiveChannel (VIDEOHANDLE, VideoChannel *channel)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_setColorKeys (VIDEOHANDLE, unsigned char rKey, unsigned char gKey, unsigned char bKey)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_getColorKeys (VIDEOHANDLE, unsigned char *rKey, unsigned char *gKey, unsigned char *bKey)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_setVideoArea (VIDEOHANDLE, unsigned short topLeftX, unsigned short topLeftY, unsigned short bottomRightX, unsigned short bottomRightY)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_getRawImage (VIDEOHANDLE, unsigned short *width, unsigned short *height, float *frameRate)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_getVideoArea (VIDEOHANDLE, unsigned short *topLeftX, unsigned short *topLeftY, unsigned short *bottomRigthX, unsigned short *bottomRigthY)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_getVideoStandard (VIDEOHANDLE, videoStandard *standard)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_getStatus (VIDEOHANDLE, unsigned char *status)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_setScaling (VIDEOHANDLE, float x, float y)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_getScaling (VIDEOHANDLE, float *x, float *y)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_activateSnapshot (VIDEOHANDLE, bool activate)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_takeSnapshot (VIDEOHANDLE, const char *path, bool bInterlaced)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_takeSnapshotRaw (VIDEOHANDLE, char *rawImgBuffer, unsigned long rawImgBuffSize, bool bInterlaced)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_takeSnapshotBmp (VIDEOHANDLE, char **bmpBuffer, unsigned long *bmpBufSize, bool bInterlaced, bool bNTSCFormat)
- 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)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_freeBmpBuffer (VIDEOHANDLE, char *bmpBuffer)

- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_minimize (VIDEOHANDLE)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_restore (VIDEOHANDLE)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_setDecoderReg (VIDEOHANDLE, unsigned char decoderRegister, unsigned char registerValue)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_getDecoderReg (VIDEOHANDLE, unsigned char decoderRegister, unsigned char *registerValue)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_setCropping (VIDEOHANDLE, unsigned char top, unsigned char left, unsigned char bottom, unsigned char right)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_getCropping (VIDEOHANDLE, unsigned char *top, unsigned char *left, unsigned char *bottom, unsigned char *right)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_showFrame (VIDEOHANDLE)

Index