CCAux

2.4.0.0

Thu May 23 2013

Contents

Chapter 1

Main Page

1.1 Introduction

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

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

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

- CCSettings
- ccvideo
- ccsettingsconsole
- touchcalibrator
- ccauxd

1.2 Changelog

1.2.1 Version 2.4.0.0 - XA/XS, XM platforms

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

- CCSettings and StartupGUI rebranded for maximatecc
- CCSettings now adapts to the number of CAN ports available
- Added the following function blocks: Battery, PowerMgr and Smart from 1.x API
- XM: CCAux2 is now fully supported on the XM platform with the same functionality as in the 1.6.4.0 release.
- XM: CCAux api 1.6.4.0 will be available for backwards compability. It is compatible back to the 1.3.1.0 release.
- XA/XS: Config_setRS485Enabled now sets MP_RS422_MODE GPIO pins to correct state
- XA/XS: Video_setMirroring implemented
- XA/XS: Playing two video channels simultaneously now works (1/2+3/4)
- XA/XS: Video can be cropped from left/right for channels 3/4
- XA/XS: Various other improvements for video channels 3/4
- XA/XS: Video standard now reported correctly
- XA/XS: ccvideo context menu now appearing consistently
- XA/XS: ccvideo context menu hanging now fixed
- XA/XS: ccvideo blanking now fixed
- XA/XS: ccvideo now handles rotation
- XA/XS: ccsettingsconsole now up to date
- XA/XS: Context menu no longer opened while calibrating
- XA/XS: The PowerOnAtStartup setting ("Always start when power turned on" in CCsettings) was always read as Enabled
- XA/XS: 1V2 is now a supported ADC channel on some instances
- XA/XS: Added TS_TCHAUTOCAL in TouchScreen class
- ccauxd: Fixed issues that caused crash when shutting the daemon off
- ccauxd: Added support for PowerMgr

1.2.2 Version 2.3.0.0 - XA/XS platform

- Functions added: Backlight_getHWStatus, Config_getRS485Enabled and Config-_setRS485Enabled
- CCSettings: Led tab improved
- CCSettings: Hide unsupported options in Power tab
- CCSettings: Hide suspend options if unsupported by HW
- CCSettings: Fixed rotation glitches
- Bugfixes
- Known issues:
  - Same as 2.2.0.0 release, plus:
  - ccvideo: Rightclick (long press) menu not appearing consistently

1.2.3 Version 2.2.0.0 - XA/XS platform

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

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

1.2.4 Version 2.1.0.0 - XA/XS platform

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

1.2.5 Version 2.0.0.0 - XA/XS platform

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

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

1.3 Known Issues

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

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 CrossControl Namespace Reference

Data Structures

- struct BatteryTimerType
- struct received_video
- struct video_dec_command
- struct version_info
- struct BuzzerSetup
- struct LedTimingType
- struct FpgaLedTimingType
- struct LedColorMixType
- struct TimerType
- struct UpgradeStatus

Typedefs

- typedef void ∗ ABOUTHANDLE
- typedef void ∗ ADCHANDLE
- typedef void ∗ AUXVERSIONHANDLE
- typedef void ∗ BACKLIGHTHANDLE
- typedef void ∗ BATTERYHANDLE
- typedef void ∗ BUZZERHANDLE
- typedef void ∗ CANSETTINGHANDLE
- typedef struct version_info VersionType
- typedef void ∗ CONFIGHANDLE
- typedef void ∗ DIAGNOSTICHANDLE
- typedef void ∗ DIGIOHANDLE
- typedef void ∗ FIRMWAREUPGHANDLE
- typedef void ∗ FRONTLEDHANDLE

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

Enumerations

- enum ChargingStatus {
  ChargingStatus_NoCharge = 0, ChargingStatus_Charging = 1, ChargingStatus-_FullyCharged = 2, ChargingStatus_TempLow = 3,
  ChargingStatus_TempHigh = 4, ChargingStatus_Unknown = 5 }
- enum PowerSource { PowerSource_Battery = 0, PowerSource_ExternalPower = 1 }
- enum ErrorStatus {
  ErrorStatus_NoError = 0, ErrorStatus_ThermistorTempSensor = 1, ErrorStatus-_SecondaryTempSensor = 2, ErrorStatus_ChargeFail = 3,
  ErrorStatus_Overcurrent = 4, ErrorStatus_Init = 5 }
- enum VoltageEnum {
  VOLTAGE_24VIN = 0, VOLTAGE_24V, VOLTAGE_12V, VOLTAGE_12-VID,
  VOLTAGE_5V, VOLTAGE_3V3, VOLTAGE_VTFT, VOLTAGE_5VSTB,
  VOLTAGE_1V9, VOLTAGE_1V8, VOLTAGE_1V5, VOLTAGE_1V2,
  VOLTAGE_1V05, VOLTAGE_1V0, VOLTAGE_0V9, VOLTAGE_VREF_I-NT,
  VOLTAGE_24V_BACKUP, VOLTAGE_2V5, VOLTAGE_1V1, VOLTAGE-_1V3_PER,
  VOLTAGE_1V3_VDDA }
- enum LightSensorOperationRange { RangeStandard = 0, RangeExtended = 1 }
- enum LightSensorSamplingMode { SamplingModeStandard = 0, SamplingMode-Extended, SamplingModeAuto }
- enum CCStatus { Disabled = 0, Enabled = 1 }
- enum eErr {
  ERR_SUCCESS = 0, ERR_OPEN_FAILED = 1, ERR_NOT_SUPPORTED = 2, ERR_UNKNOWN_FEATURE = 3,
  ERR_DATATYPE_MISMATCH = 4, ERR_CODE_NOT_EXIST = 5, ERR_-BUFFER_SIZE = 6, ERR_IOCTRL_FAILED = 7,
  ERR_INVALID_DATA = 8, ERR_INVALID_PARAMETER = 9, ERR_CRE-ATE_THREAD = 10, ERR_IN_PROGRESS = 11,
  ERR_CHECKSUM = 12, ERR_INIT_FAILED = 13, ERR_VERIFY_FAILED

= 14, ERR_DEVICE_READ_DATA_FAILED = 15,
ERR_DEVICE_WRITE_DATA_FAILED = 16, ERR_COMMAND_FAILED = 17, ERR_EEPROM = 18, ERR_JIDA_TEMP = 19,
ERR_AVERAGE_CALC_STARTED = 20, ERR_NOT_RUNNING = 21, ERR_I2C_EXPANDER_READ_FAILED = 22, ERR_I2C_EXPANDER_WRITE_FAILED = 23,
ERR_I2C_EXPANDER_INIT_FAILED = 24, ERR_NEWER_SS_VERSION_REQUIRED = 25, ERR_NEWER_FPGA_VERSION_REQUIRED = 26, ERR_NEWER_FRONT_VERSION_REQUIRED = 27,
ERR_TELEMATICS_GPRS_NOT_AVAILABLE = 28, ERR_TELEMATICS_WLAN_NOT_AVAILABLE = 29, ERR_TELEMATICS_BT_NOT_AVAILABLE = 30, ERR_TELEMATICS_GPS_NOT_AVAILABLE = 31,
ERR_MEM_ALLOC_FAIL = 32, ERR_JOIN_THREAD = 33 }

- enum DeInterlaceMode { DeInterlace_Even = 0, DeInterlace_Odd = 1, DeInterlace_BOB = 2 }
- enum VideoChannel { Analog_Channel_1 = 0, Analog_Channel_2 = 1, Analog_Channel_3 = 2, Analog_Channel_4 = 3 }
- enum videoStandard {
  STD_M_J_NTSC = 0, STD_B_D_G_H_I_N_PAL = 1, STD_M_PAL = 2, STD_PAL = 3,
  STD_NTSC = 4, STD_SECAM = 5 }
- enum VideoRotation { RotNone = 0, Rot90, Rot180, Rot270 }
- enum CanFrameType { FrameStandard, FrameExtended, FrameStandardExtended }
- enum TriggerConf { Front_Button_Enabled = 1, OnOff_Signal_Enabled = 2, Both_Button_And_Signal_Enabled = 3 }
- enum PowerAction { NoAction = 0, ActionSuspend = 1, ActionShutDown = 2 }
- enum ButtonPowerTransitionStatus {
  BPTS_No_Change = 0, BPTS_ShutDown = 1, BPTS_Suspend = 2, BPTS_Restart = 3,
  BPTS_BtnPressed = 4, BPTS_BtnPressedLong = 5, BPTS_SignalOff = 6 }
- enum OCDStatus { OCD_OK = 0, OCD_OC = 1, OCD_POWER_OFF = 2 }
- enum JidaSensorType {
  TEMP_CPU = 0, TEMP_BOX = 1, TEMP_ENV = 2, TEMP_BOARD = 3,
  TEMP_BACKPLANE = 4, TEMP_CHIPSETS = 5, TEMP_VIDEO = 6, TEMP_OTHER = 7 }
- enum UpgradeAction {
  UPGRADE_INIT, UPGRADE_PREP_COM, UPGRADE_READING_FILE, UPGRADE_CONVERTING_FILE,
  UPGRADE_FLASHING, UPGRADE_VERIFYING, UPGRADE_COMPLETE, UPGRADE_COMPLETE_WITH_ERRORS }
- enum CCAuxColor {
  RED = 0, GREEN, BLUE, CYAN,
  MAGENTA, YELLOW, UNDEFINED_COLOR }
- enum RS4XXPort { RS4XXPort1 = 1, RS4XXPort2, RS4XXPort3, RS4XXPort4 }
- enum startupReasonCodes {
  startupReasonCodeUndefined = 0x0000, startupReasonCodeButtonPress = 0x0055,

startupReasonCodeExtCtrl = 0x00AA, startupReasonCodeMPRestart = 0x00F0, startupReasonCodePowerOnStartup = 0x000F }
- enum shutDownReasonCodes { shutdownReasonCodeNoError = 0x001F }
- enum hwErrorStatusCodes { errCodeNoErr = 0 }
- enum PowerMgrConf { Normal = 0, ApplicationControlled = 1, BatterySuspend = 2 }
- enum PowerMgrStatus { NoRequestsPending = 0, SuspendPending = 1, Shutdown-Pending = 2 }
- enum TouchScreenModeSettings { MOUSE_NEXT_BOOT = 0, TOUCH_NE-XT_BOOT = 1, MOUSE_NOW = 2, TOUCH_NOW = 3 }
- enum TSAdvancedSettingsParameter {
TS_RIGHT_CLICK_TIME = 0, TS_LOW_LEVEL = 1, TS_UNTOUCHLEV-EL = 2, TS_DEBOUNCE_TIME = 3,
TS_DEBOUNCE_TIMEOUT_TIME = 4, TS_DOUBLECLICK_MAX_CLIC-K_TIME = 5, TS_DOUBLE_CLICK_TIME = 6, TS_MAX_RIGHTCLICK_D-ISTANCE = 7,
TS_USE_DEJITTER = 8, TS_CALIBTATION_WIDTH = 9, TS_CALIBRAT-ION_MEASUREMENTS = 10, TS_RESTORE_DEFAULT_SETTINGS = 11,
TS_TCHAUTOCAL = 12 }
- enum CalibrationModeSettings {
MODE_UNKNOWN = 0, MODE_NORMAL = 1, MODE_CALIBRATION_-5P = 2, MODE_CALIBRATION_9P = 3,
MODE_CALIBRATION_13P = 4 }
- enum CalibrationConfigParam {
CONFIG_CALIBRATION_WITH = 0, CONFIG_CALIBRATION_MEASU-REMENTS = 1, CONFIG_5P_CALIBRATION_POINT_BORDER = 2, CO-NFIG_13P_CALIBRATION_POINT_BORDER = 3,
CONFIG_13P_CALIBRATION_TRANSITION_MIN = 4, CONFIG_13P_CA-LIBRATION_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 (ABOUTHAND-LE, 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 (ABOU-THANDLE, char ∗buff, int len)

- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV About_getMainHWversion (ABOUTHAND-LE, 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 (ABOUTHAND-LE, char ∗buff, int len)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV About_getNrOfETHConnections (ABOUTH-ANDLE, unsigned char ∗NrOfConnections)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV About_getNrOfCANConnections (ABOUT-HANDLE, unsigned char ∗NrOfConnections)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV About_getNrOfVideoConnections (ABOUT-HANDLE, unsigned char ∗NrOfConnections)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV About_getNrOfUSBConnections (ABOUTH-ANDLE, unsigned char ∗NrOfConnections)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV About_getNrOfSerialConnections (ABOUT-HANDLE, unsigned char ∗NrOfConnections)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV About_getNrOfDigIOConnections (ABOUT-HANDLE, unsigned char ∗NrOfConnections)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV About_getIsDisplayAvailable (ABOUTHAN-DLE, bool ∗available)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV About_getIsTouchScreenAvailable (ABOUT-HANDLE, bool ∗available)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV About_getDisplayResolution (ABOUTHAN-DLE, char ∗buff, int len)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV About_getAddOnPCBSerial (ABOUTHAN-DLE, char ∗buff, int len)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV About_getAddOnPCBArt (ABOUTHANDL-E, char ∗buff, int length)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV About_getAddOnManufacturingDate (ABO-UTHANDLE, char ∗buff, int len)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV About_getAddOnHWversion (ABOUTHAN-DLE, 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 (AUXVERSIO-NHANDLE, unsigned char ∗major, unsigned char ∗minor, unsigned char ∗release, unsigned char ∗build)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV AuxVersion_getCCAuxVersion (AUXVERS-IONHANDLE, unsigned char ∗major, unsigned char ∗minor, unsigned char ∗release, unsigned char ∗build)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV AuxVersion_getOSVersion (AUXVERSION-HANDLE, unsigned char ∗major, unsigned char ∗minor, unsigned char ∗release, unsigned char ∗build)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV AuxVersion_getCCAuxDrvVersion (AUXV-ERSIONHANDLE, 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 (BACKLIGHTHAN-DLE, unsigned char ∗intensity)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Backlight_setIntensity (BACKLIGHTHAN-DLE, unsigned char intensity)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Backlight_getStatus (BACKLIGHTHANDL-E, unsigned char ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Backlight_getHWStatus (BACKLIGHTHA-NDLE, bool ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Backlight_startAutomaticBL (BACKLIGHT-HANDLE)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Backlight_stopAutomaticBL (BACKLIGHT-HANDLE)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Backlight_getAutomaticBLStatus (BACKL-IGHTHANDLE, unsigned char ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Backlight_setAutomaticBLParams (BACKL-IGHTHANDLE, bool bSoftTransitions)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Backlight_getAutomaticBLParams (BACK-LIGHTHANDLE, bool ∗bSoftTransitions, double ∗k)

- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Backlight_setAutomaticBLFilter (BACKLI-GHTHANDLE, unsigned long averageWndSize, unsigned long rejectWndSize, unsigned long rejectDeltaInLux, LightSensorSamplingMode mode)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Backlight_getAutomaticBLFilter (BACKLI-GHTHANDLE, unsigned long ∗averageWndSize, unsigned long ∗rejectWnd-Size, unsigned long ∗rejectDeltaInLux, LightSensorSamplingMode ∗mode)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Backlight_getLedDimming (BACKLIGHTH-ANDLE, CCStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Backlight_setLedDimming (BACKLIGHTH-ANDLE, CCStatus status)
- EXTERN_C CCAUXDLL_API
BATTERYHANDLE
CCAUXDLL_CALLING_CONV GetBattery (void)
- EXTERN_C CCAUXDLL_API void
CCAUXDLL_CALLING_CONV Battery_release (BATTERYHANDLE)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Battery_isBatteryPresent (BATTERYHAND-LE, bool ∗batteryIsPresent)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Battery_getBatteryVoltageStatus (BATTER-YHANDLE, unsigned char ∗batteryVoltagePercent)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Battery_getBatteryChargingStatus (BATTE-RYHANDLE, ChargingStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Battery_getPowerSource (BATTERYHAND-LE, PowerSource ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Battery_getBatteryTemp (BATTERYHAND-LE, signed short ∗temperature)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Battery_getHwErrorStatus (BATTERYHAN-DLE, ErrorStatus ∗errorCode)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Battery_getTimer (BATTERYHANDLE, Battery-TimerType ∗times)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Battery_getMinMaxTemp (BATTERYHAN-DLE, signed short ∗minTemp, signed short ∗maxTemp)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Battery_getBatteryHWversion (BATTERY-HANDLE, char ∗buff, int len)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Battery_getBatterySwVersion (BATTERYH-ANDLE, unsigned short ∗major, unsigned short ∗minor, unsigned short ∗release, unsigned short ∗build)

- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Battery_getBatterySerial (BATTERYHAND-LE, char ∗buff, int len)
- EXTERN_C CCAUXDLL_API
  BUZZERHANDLE
  CCAUXDLL_CALLING_CONV GetBuzzer (void)
- EXTERN_C CCAUXDLL_API void
  CCAUXDLL_CALLING_CONV Buzzer_release (BUZZERHANDLE)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Buzzer_getFrequency (BUZZERHANDLE, unsigned short ∗frequency)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Buzzer_getVolume (BUZZERHANDLE, unsigned short ∗volume)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Buzzer_getTrigger (BUZZERHANDLE, bool ∗trigger)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Buzzer_setFrequency (BUZZERHANDLE, unsigned short frequency)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Buzzer_setVolume (BUZZERHANDLE, unsigned short volume)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Buzzer_setTrigger (BUZZERHANDLE, bool trigger)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Buzzer_buzze (BUZZERHANDLE, int time, bool blocking)
- EXTERN_C CCAUXDLL_API
  CANSETTINGHANDLE
  CCAUXDLL_CALLING_CONV GetCanSetting (void)
- EXTERN_C CCAUXDLL_API void
  CCAUXDLL_CALLING_CONV CanSetting_release (CANSETTINGHAND-LE)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV CanSetting_getBaudrate (CANSETTINGHA-NDLE, unsigned char net, unsigned short ∗baudrate)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV CanSetting_getFrameType (CANSETTING-HANDLE, unsigned char net, CanFrameType ∗frameType)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV CanSetting_setBaudrate (CANSETTINGHA-NDLE, unsigned char net, unsigned short baudrate)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV CanSetting_setFrameType (CANSETTING-HANDLE, 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 (CONFIGH-ANDLE, TriggerConf ∗config)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Config_getShortButtonPressAction (CONFI-GHANDLE, PowerAction ∗action)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Config_getLongButtonPressAction (CONFI-GHANDLE, PowerAction ∗action)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Config_getOnOffSigAction (CONFIGHAN-DLE, PowerAction ∗action)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Config_getFrontBtnTrigTime (CONFIGHA-NDLE, unsigned short ∗triggertime)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Config_getExtOnOffSigTrigTime (CONFIG-HANDLE, unsigned long ∗triggertime)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Config_getSuspendMaxTime (CONFIGHA-NDLE, unsigned short ∗maxTime)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Config_getCanStartupPowerConfig (CONFI-GHANDLE, CCStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Config_getVideoStartupPowerConfig (CON-FIGHANDLE, unsigned char ∗config)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Config_getExtFanStartupPowerConfig (CO-NFIGHANDLE, CCStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Config_getStartupVoltageConfig (CONFIG-HANDLE, double ∗voltage)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Config_getHeatingTempLimit (CONFIGHA-NDLE, signed short ∗temperature)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Config_getPowerOnStartup (CONFIGHAN-DLE, CCStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Config_setStartupTriggerConfig (CONFIGH-ANDLE, TriggerConf conf)

- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_setShortButtonPressAction (CONFIGHANDLE, PowerAction action)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_setLongButtonPressAction (CONFIGHANDLE, PowerAction action)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_setOnOffSigAction (CONFIGHANDLE, PowerAction action)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_setFrontBtnTrigTime (CONFIGHANDLE, unsigned short triggertime)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_setExtOnOffSigTrigTime (CONFIGHANDLE, unsigned long triggertime)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_setSuspendMaxTime (CONFIGHANDLE, unsigned short maxTime)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_setCanStartupPowerConfig (CONFIGHANDLE, CCStatus status)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_setVideoStartupPowerConfig (CONFIGHANDLE, unsigned char config)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_setExtFanStartupPowerConfig (CONFIGHANDLE, CCStatus status)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_setStartupVoltageConfig (CONFIGHANDLE, double voltage)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_setHeatingTempLimit (CONFIGHANDLE, signed short temperature)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_setPowerOnStartup (CONFIGHANDLE, CCStatus status)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_setRS485Enabled (CONFIGHANDLE, RS4XXPort port, bool enabled)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_getRS485Enabled (CONFIGHANDLE, RS4XXPort port, bool ∗enabled)
- EXTERN_C CCAUXDLL_API DIAGNOSTICHANDLE CCAUXDLL_CALLING_CONV GetDiagnostic (void)
- EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV Diagnostic_release (DIAGNOSTICHANDLE)

- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Diagnostic_getSSTemp (DIAGNOSTICHA-NDLE, signed short ∗temperature)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Diagnostic_getPCBTemp (DIAGNOSTICH-ANDLE, signed short ∗temperature)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Diagnostic_getPMTemp (DIAGNOSTICHA-NDLE, unsigned char index, signed short ∗temperature, JidaSensorType ∗jst)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Diagnostic_getStartupReason (DIAGNOST-ICHANDLE, unsigned short ∗reason)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Diagnostic_getShutDownReason (DIAGNO-STICHANDLE, unsigned short ∗reason)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Diagnostic_getHwErrorStatus (DIAGNOST-ICHANDLE, unsigned short ∗errorCode)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Diagnostic_getTimer (DIAGNOSTICHAND-LE, TimerType ∗times)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Diagnostic_getMinMaxTemp (DIAGNOSTI-CHANDLE, signed short ∗minTemp, signed short ∗maxTemp)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Diagnostic_getPowerCycles (DIAGNOSTIC-HANDLE, unsigned short ∗powerCycles)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Diagnostic_clearHwErrorStatus (DIAGNOS-TICHANDLE)
- 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 LIGHTSENSORHANDLE CCAUXDLL_CALLING_CONV GetLightsensor (void)

- EXTERN_C CCAUXDLL_API void
CCAUXDLL_CALLING_CONV Lightsensor_release (LIGHTSENSORHAN-DLE)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Lightsensor_getIlluminance (LIGHTSENSO-RHANDLE, unsigned short ∗value)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Lightsensor_getIlluminance2 (LIGHTSENS-ORHANDLE, unsigned short ∗value, unsigned char ∗ch0, unsigned char ∗ch1)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Lightsensor_getAverageIlluminance (LIGH-TSENSORHANDLE, unsigned short ∗value)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Lightsensor_startAverageCalc (LIGHTSEN-SORHANDLE, unsigned long averageWndSize, unsigned long rejectWndSize, unsigned long rejectDeltaInLux, LightSensorSamplingMode mode)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Lightsensor_stopAverageCalc (LIGHTSEN-SORHANDLE)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Lightsensor_getOperatingRange (LIGHTSE-NSORHANDLE, LightSensorOperationRange ∗range)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Lightsensor_setOperatingRange (LIGHTSE-NSORHANDLE, 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 (POWERHAND-LE, CCStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Power_getCanPowerStatus (POWERHAND-LE, CCStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Power_getVideoPowerStatus (POWERHAN-DLE, unsigned char ∗videoStatus)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Power_getExtFanPowerStatus (POWERHA-NDLE, CCStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Power_getButtonPowerTransitionStatus (PO-WERHANDLE, ButtonPowerTransitionStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Power_getVideoOCDStatus (POWERHAN-DLE, 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 POWERMGRHANDLE CCAUXDLL_CALLING_CONV GetPowerMgr (void)
- EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV PowerMgr_release (POWERMGRHANDLE)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV PowerMgr_registerControlledSuspendOrShutDown (POWERMGRHANDLE, PowerMgrConf conf)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV PowerMgr_getConfiguration (POWERMGRHANDLE, PowerMgrConf ∗conf)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV PowerMgr_getPowerMgrStatus (POWERMGRHANDLE, PowerMgrStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV PowerMgr_setAppReadyForSuspendOrShutdown (POWERMGRHANDLE)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV PowerMgr_hasResumed (POWERMGRHANDLE, bool ∗resumed)
- EXTERN_C CCAUXDLL_API SMARTHANDLE CCAUXDLL_CALLING_CONV GetSmart (void)
- EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV Smart_release (SMARTHANDLE)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Smart_getRemainingLifeTime (SMARTHANDLE, unsigned char ∗lifetimepercent)

- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Smart_getDeviceSerial (SMARTHANDLE, char ∗buff, int len)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Smart_getInitialTime (SMARTHANDLE, time-_t ∗time)
- EXTERN_C CCAUXDLL_API
TELEMATICSHANDLE
CCAUXDLL_CALLING_CONV GetTelematics (void)
- EXTERN_C CCAUXDLL_API void
CCAUXDLL_CALLING_CONV Telematics_release (TELEMATICSHANDL-E)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Telematics_getTelematicsAvailable (TELEM-ATICSHANDLE, CCStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Telematics_getGPRSPowerStatus (TELEM-ATICSHANDLE, CCStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Telematics_getGPRSStartUpPowerStatus (T-ELEMATICSHANDLE, CCStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Telematics_getWLANPowerStatus (TELEM-ATICSHANDLE, CCStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Telematics_getWLANStartUpPowerStatus (T-ELEMATICSHANDLE, CCStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Telematics_getBTPowerStatus (TELEMAT-ICSHANDLE, CCStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Telematics_getBTStartUpPowerStatus (TEL-EMATICSHANDLE, CCStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Telematics_getGPSPowerStatus (TELEMA-TICSHANDLE, CCStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Telematics_getGPSStartUpPowerStatus (TE-LEMATICSHANDLE, CCStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Telematics_getGPSAntennaStatus (TELEM-ATICSHANDLE, CCStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Telematics_setGPRSPowerStatus (TELEMA-TICSHANDLE, CCStatus status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Telematics_setGPRSStartUpPowerStatus (T-ELEMATICSHANDLE, 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_setRotation (VIDEOHANDLE, VideoRotation rotation)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Video_getRotation (VIDEOHANDLE, VideoRotation ∗rotation)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Video_setActiveChannel (VIDEOHANDLE, VideoChannel channel)

- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_getActiveChannel (VIDEOHANDLE, VideoChannel ∗channel)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_setColorKeys (VIDEOHANDLE, unsigned char rKey, unsigned char gKey, unsigned char bKey)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_getColorKeys (VIDEOHANDLE, unsigned char ∗rKey, unsigned char ∗gKey, unsigned char ∗bKey)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_setVideoArea (VIDEOHANDLE, unsigned short topLeftX, unsigned short topLeftY, unsigned short bottomRightX, unsigned short bottomRightY)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_getRawImage (VIDEOHANDLE, unsigned short ∗width, unsigned short ∗height, float ∗frameRate)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_getVideoArea (VIDEOHANDLE, unsigned short ∗topLeftX, unsigned short ∗topLeftY, unsigned short ∗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)

5.1.1 Typedef Documentation

5.1.1.1 typedef enum CrossControl::PowerMgrConf _PowerMgrConf

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

5.1.1.2 typedef enum CrossControl::PowerMgrStatus _PowerMgrStatus

5.1.1.3 typedef void∗ ABOUTHANDLE

5.1.1.4 typedef void∗ ADCHANDLE

5.1.1.5 typedef void∗ AUXVERSIONHANDLE

5.1.1.6 typedef void∗ BACKLIGHTHANDLE

5.1.1.7 typedef void∗ BATTERYHANDLE

5.1.1.8 typedef void∗ BUZZERHANDLE

5.1.1.9 typedef void∗ CANSETTINGHANDLE

5.1.1.10 typedef void∗ CONFIGHANDLE

5.1.1.11 typedef void∗ DIAGNOSTICHANDLE

5.1.1.12 typedef void∗ DIGIOHANDLE

5.1.1.13 typedef void∗ FIRMWAREUPGHANDLE

5.1.1.14 typedef void∗ FRONTLEDHANDLE

5.1.1.15 typedef void∗ LIGHTSENSORHANDLE

5.1.1.16 typedef void∗ POWERHANDLE

5.1.1.17 typedef void∗ POWERMGRHANDLE

5.1.1.18 typedef void∗ SMARTHANDLE

5.1.1.19 typedef void∗ TELEMATICSHANDLE

5.1.1.20 typedef void∗ TOUCHSCREENCALIBHANDLE

5.1.1.21 typedef void∗ TOUCHSCREENHANDLE

5.1.1.22 typedef struct version_info VersionType

5.1.1.23 typedef void∗ VIDEOHANDLE

5.1.2 Enumeration Type Documentation

5.1.2.1 enum ButtonPowerTransitionStatus

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

Enumerator

BPTS_No_Change No change

BPTS_ShutDown A shutdown has been initiated since the front panel button has been pressed longer than the set FrontBtnShutDownTrigTime

BPTS_Suspend Suspend mode has been initiated since the front panel button has been pressed (shortly) and suspend mode is enabled

BPTS_Restart Not currently in use

BPTS_BtnPressed The front panel button is currently pressed. It has not been released and it has not yet been held longer than FrontBtnShutDownTrigTime.

BPTS_BtnPressedLong The front panel button is currently pressed. It has not been released and it has been held longer than FrontBtnShutDownTrigTime.

BPTS_SignalOff The external on/off signal is low, but not yet long enough for the ExtOnOffSigSuspTrigTime.

5.1.2.2 enum CalibrationConfigParam

Touch screen calibration parameters

Enumerator

CONFIG_CALIBRATION_WITH

CONFIG_CALIBRATION_MEASUREMENTS Accepted error value when calibrating.

CONFIG_5P_CALIBRATION_POINT_BORDER Number of measurements to accept a calibration point.

CONFIG_13P_CALIBRATION_POINT_BORDER The number of pixels from the border where the 5 point calibration points should be located.

CONFIG_13P_CALIBRATION_TRANSITION_MIN The number of pixels from the border where the 13 point calibration points should be located.

CONFIG_13P_CALIBRATION_TRANSITION_MAX Min defines the transition area in number of pixels, where the two different calibrations are used.

5.1.2.3 enum CalibrationModeSettings

Touch screen calibration modes

Enumerator

MODE_UNKNOWN

MODE_NORMAL Unknown mode.

MODE_CALIBRATION_5P Normal operation mode.

MODE_CALIBRATION_9P Calibration with 5 points mode.

MODE_CALIBRATION_13P Calibration with 9 points mode.

5.1.2.4 enum CanFrameType

Can frame type settings

Enumerator

- ***FrameStandard***
- ***FrameExtended***
- ***FrameStandardExtended***

5.1.2.5 enum CCAuxColor

Enumeration of standard colors

Enumerator

- ***RED***
- ***GREEN*** RGB 0xF, 0x0, 0x0
- ***BLUE*** RGB 0x0, 0xF, 0x0
- ***CYAN*** RGB 0x0, 0x0, 0xF
- ***MAGENTA*** RGB 0x0, 0xF, 0xF
- ***YELLOW*** RGB 0xF, 0x0, 0xF
- ***UNDEFINED_COLOR*** RGB 0xF, 0xF, 0x0
  Returns if color is not a standard color

5.1.2.6 enum CCStatus

Enable/disable enumeration

Enumerator

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

5.1.2.7 enum ChargingStatus

Current charging status of the battery.

Enumerator

- ***ChargingStatus_NoCharge*** The battery is not being charged. System is running on battery power.
- ***ChargingStatus_Charging*** The battery is currently being charged
- ***ChargingStatus_FullyCharged*** The battery is fully chareged

ChargingStatus_TempLow The temperature is too low to allow the battery to be charged

ChargingStatus_TempHigh The temperature is too high to allow the battery to be charged

ChargingStatus_Unknown There was an error determining the charging status

5.1.2.8 enum DeInterlaceMode

Enumerator

DeInterlace_Even

DeInterlace_Odd Use only even rows from the interlaced input stream

DeInterlace_BOB Use only odd rows from the interlaced input stream

5.1.2.9 enum eErr

Error code enumeration

Enumerator

ERR_SUCCESS

ERR_OPEN_FAILED Success

ERR_NOT_SUPPORTED Open failed

ERR_UNKNOWN_FEATURE Not supported

ERR_DATATYPE_MISMATCH Unknown feature

ERR_CODE_NOT_EXIST Datatype mismatch

ERR_BUFFER_SIZE Code doesn't exist

ERR_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

5.1.2.10 enum ErrorStatus

Enumerator

ErrorStatus_NoError

ErrorStatus_ThermistorTempSensor

ErrorStatus_SecondaryTempSensor

ErrorStatus_ChargeFail

ErrorStatus_Overcurrent

ErrorStatus_Init

5.1.2.11 enum hwErrorStatusCodes

The error codes returned by getHWErrorStatus.

Enumerator

errCodeNoErr

5.1.2.12 enum JidaSensorType

Jida temperature sensor types

Enumerator

TEMP_CPU

TEMP_BOX
TEMP_ENV
TEMP_BOARD
TEMP_BACKPLANE
TEMP_CHIPSETS
TEMP_VIDEO
TEMP_OTHER

5.1.2.13 enum LightSensorOperationRange

Light sensor operation ranges.

Enumerator

RangeStandard
RangeExtended Light sensor operation range standard

5.1.2.14 enum LightSensorSamplingMode

Light sensor sampling modes.

Enumerator

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

5.1.2.15 enum OCDStatus

Overcurrent detection status.

Enumerator

OCD_OK Normal operation, no overcurrent condition detected
OCD_OC Overcurrent has been detected, power has therefore been turned off, but may be functioning again if the problem disappeared after re-test
OCD_POWER_OFF Overcurrent has been detected, power has been permanently turned off

5.1.2.16 enum PowerAction

Button and on/off signal actions.

Enumerator

NoAction No action taken

ActionSuspend The system enters suspend mode

ActionShutDown The system shuts down

5.1.2.17 enum PowerMgrConf

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

Enumerator

Normal Applications will not be able to delay suspend/shutdown requests. This is the normal configuration that is used when the PowerMgr class is not being used. Setting this configuration turns off the feature if it is in use.

ApplicationControlled Applications can delay suspend/shutdown requests.

BatterySuspend In this mode, the computer will automatically enter suspend mode when the unit starts running on battery power. Applications can delay suspend/shutdown requests. This mode is only applicable if the unit has an external battery. Using this configuration on a computer without an external battery will be the same as using the configuration ApplicationControlled.

5.1.2.18 enum PowerMgrStatus

Enumerator

NoRequestsPending No suspend or shutdown requests.

SuspendPending A suspend request is pending.

ShutdownPending A shutdown request is pending.

5.1.2.19 enum PowerSource

Current power source of the computer.

Enumerator

PowerSource_Battery

PowerSource_ExternalPower

5.1.2.20 enum RS4XXPort

Enumeration of RS4XX ports (1-4)

Enumerator

RS4XXPort1

RS4XXPort2

RS4XXPort3

RS4XXPort4

5.1.2.21 enum shutDownReasonCodes

The shutdown codes returned by getShutDownReason.

Enumerator

shutdownReasonCodeNoError

5.1.2.22 enum startupReasonCodes

The restart codes returned by getStartupReason.

Enumerator

startupReasonCodeUndefined

startupReasonCodeButtonPress Unknown startup reason.

startupReasonCodeExtCtrl The system was started by front panel button press

startupReasonCodeMPRestart The system was started by the external control signal

startupReasonCodePowerOnStartup The system was restarted by OS request

5.1.2.23 enum TouchScreenModeSettings

Touch screen USB profile settings

Enumerator

MOUSE_NEXT_BOOT

TOUCH_NEXT_BOOT Set the touch USB profile to mouse profile. Active upon the next boot.

MOUSE_NOW Set the touch USB profile to touch profile. Active upon the next boot.

TOUCH_NOW Immediately set the touch USB profile to mouse profile.

5.1.2.24 enum TriggerConf

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

Enumerator

Front_Button_Enabled Front button is enabled for startup and wake-up

OnOff_Signal_Enabled The external on/off signal is enabled for startup and wake-up

Both_Button_And_Signal_Enabled Both of the above are enabled

5.1.2.25 enum TSAdvancedSettingsParameter

Touch screen advanced settings parameters

Enumerator

TS_RIGHT_CLICK_TIME Right click time in ms, except for touch profile on XM platform

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

TS_UNTOUCHLEVEL A/D value where the screen is considered to be untouched.

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

TS_DEBOUNCE_TIMEOUT_TIME After debounce, an event will be ignored if after this time there are no valid measurements above TS_LOW_LEVEL. This time must be larger than TS_DEBOUNCE_TIME. Front uc 0.5.3.1 had the default value of 12ms, newer versions: 36ms.

TS_DOUBLECLICK_MAX_CLICK_TIME Parameter used for improving double click accuracy. A touch event this long or shorter is considered to be one of the clicks in a double click.

TS_DOUBLE_CLICK_TIME Parameter used for improving double click accuracy. Time allowed between double clicks. Used for double click improvement.

TS_MAX_RIGHTCLICK_DISTANCE Maximum distance allowed to move pointer and still consider the event a right click.

TS_USE_DEJITTER The dejitter function enables smoother pointer movement. Set to non-zero to enable the function or zero to disable it.

TS_CALIBTATION_WIDTH Accepted difference in measurement during calibration of a point.

TS_CALIBRATION_MEASUREMENTS Number of measurements needed to accept a calibration point.

TS_RESTORE_DEFAULT_SETTINGS Set to non-zero to restore all the above settings to their defaults. This parameter cannot be read and setting it to zero has no effect.

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

5.1.2.26 enum UpgradeAction

Upgrade Action enumeration

Enumerator

UPGRADE_INIT

UPGRADE_PREP_COM Initiating, checking for compatibility etc

UPGRADE_READING_FILE Preparing communication

UPGRADE_CONVERTING_FILE Opening and reading the supplied file

UPGRADE_FLASHING Converting the mcs format to binary format

UPGRADE_VERIFYING Flashing the file

UPGRADE_COMPLETE Verifying the programmed image

UPGRADE_COMPLETE_WITH_ERRORS Upgrade was finished
Upgrade finished prematurely, see errorCode for the reason of failure

5.1.2.27 enum VideoChannel

The available analog video channels

Enumerator

Analog_Channel_1

Analog_Channel_2

Analog_Channel_3

Analog_Channel_4

5.1.2.28 enum VideoRotation

Enumerator

RotNone

Rot90

Rot180

Rot270

5.1.2.29 enum videoStandard

Enumerator

STD_M_J_NTSC	
STD_B_D_G_H_I_N_PAL	(M,J) NTSC ITU-R BT.601
STD_M_PAL	(B, D, G, H, I, N) PAL ITU-R BT.601
STD_PAL	(M) PAL ITU-R BT.601
STD_NTSC	PAL-Nc ITU-R BT.601
STD_SECAM	NTSC 4.43 ITU-R BT.601

5.1.2.30 enum VoltageEnum

Voltage type enumeration

Enumerator

VOLTAGE_24VIN	
VOLTAGE_24V	< 24VIN
VOLTAGE_12V	< 24V
VOLTAGE_12VID	< 12V
VOLTAGE_5V	< 12VID
VOLTAGE_3V3	< 5V
VOLTAGE_VTFT	< 3.3V
VOLTAGE_5VSTB	< VTFT
VOLTAGE_1V9	< 5VSTB
VOLTAGE_1V8	< 1.9V
VOLTAGE_1V5	< 1.8V
VOLTAGE_1V2	< 1.5V
VOLTAGE_1V05	< 1.2V
VOLTAGE_1V0	< 1.05V
VOLTAGE_0V9	< 1.0V
VOLTAGE_VREF_INT	< 0.9V
VOLTAGE_24V_BACKUP	< SS internal VRef
VOLTAGE_2V5	< 24V backup capacitor
VOLTAGE_1V1	< 2.5V
VOLTAGE_1V3_PER	< 1.1V
VOLTAGE_1V3_VDDA	< 1.3V_PER

< 1.3V_VDDA

5.1.3 Function Documentation

5.1.3.1 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getAddOnHWversion (ABOUTHANDLE , char ∗ *buff*, int *len*)

Get Add on hardware version.

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

Example Usage:

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

5.1.3.2 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getAddOnManufacturingDate (ABOUTHANDLE , char ∗ *buff*, int *len*)

Get Add on manufacturing date.

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

Example Usage:

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

5.1.3.3 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getAddOnPCBArt (ABOUTHANDLE , char * *buff*, int *length*)

Get Add on PCB article number.

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

Example Usage:

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

5.1.3.4 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getAddOnPCBSerial (ABOUTHANDLE , char * *buff*, int *len*)

Get Add on PCB serial number.

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

Example Usage:

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

5.1.3.5 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getDisplayResolution (ABOUTHANDLE , char ∗ *buff,* int *len*)

Get display resolution.

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

Example Usage:

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

5.1.3.6 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getFrontPcbRev (ABOUTHANDLE , unsigned char ∗ *major,* unsigned char ∗ *minor*)

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

Supported Platform(s): XA, XS

Parameters

major	The major pcb revision.
minor	The minor pcb revision.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.7 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getIOExpanderValue (ABOUTHANDLE , unsigned short ∗ *value*)

Get Value for IO Expander

Supported Platform(s): XA, XS

Parameters

value	IO Expander value.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.8 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getIsAnybusMounted (ABOUTHANDLE , bool * *mounted*)

Get Anybus mounting status.

Supported Platform(s): XA, XS

Parameters

mounted	Is Anybus mounted?

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.9 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getIsBTMounted (ABOUTHANDLE , bool * *mounted*)

Get BlueTooth module mounting status.

Supported Platform(s): XM, XA, XS

Parameters

mounted	Is module mounted?

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.10 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getIsDisplayAvailable (ABOUTHANDLE , bool * *available*)

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

Supported Platform(s): XM, XA, XS

Parameters

available	Is display available?

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.11 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getIsGPRSMounted (ABOUTHANDLE , bool * *mounted*)

Get GPRS module mounting status.

Supported Platform(s): XM, XA, XS

Parameters

mounted	Is module mounted?

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.12 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getIsGPSMounted (ABOUTHANDLE , bool ∗ *mounted*)

Get GPS module mounting status.

Supported Platform(s): XM, XA, XS

Parameters

mounted	Is module mounted?

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.13 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getIsIOExpanderMounted (ABOUTHANDLE , bool ∗ *mounted*)

Get IO Expander mounting status.

Supported Platform(s): XA, XS

Parameters

mounted	Is IO Expander mounted?

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.14 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getIsTouchScreenAvailable (ABOUTHANDLE , bool ∗ *available*)

Get Display TouchScreen status.

Supported Platform(s): XM, XA, XS

Parameters

available	Is TouchScreen available?

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.15 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getIsWLANMounted (ABOUTHANDLE , bool ∗ *mounted*)

Get WLAN module mounting status.

Supported Platform(s): XM, XA, XS

Parameters

mounted	Is module mounted?

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.16 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getMainHWversion (ABOUTHANDLE , char ∗ *buff,* int *len*)

Get main hardware version (PCB revision).

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

Example Usage:

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

5.1.3.17 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getMainManufacturingDate (ABOUTHANDLE , char ∗ *buff,* int *len*)

Get main manufacturing date.

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

Example Usage:

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

5.1.3.18 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getMainPCBArt (ABOUTHANDLE , char ∗ *buff,* int *length*)

Get main PCB article number.

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

Example Usage:

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

5.1.3.19 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getMainPCBSerial (ABOUTHANDLE , char ∗ *buff*, int *len*)

Get main PCB serial number.

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

Example Usage:

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

5.1.3.20 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getMainProdArtNr (ABOUTHANDLE , char ∗ *buff*, int *len*)

Get main product article number.

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

Example Usage:

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

5.1.3.21 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getMainProdRev (ABOUTHANDLE , char ∗ *buff*, int *len*)

Get main product revision.

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

Example Usage:

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

5.1.3.22 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getNrOfCANConnections (ABOUTHANDLE , unsigned char ∗ *NrOfConnections*)

Get number of CAN connections present.

Supported Platform(s): XM, XA, XS

Parameters

NrOf-Connections	Returns the number of connections.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.23 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getNrOfDigIOConnections (ABOUTHANDLE , unsigned char ∗ *NrOfConnections*)

Get number of digital I/O connections present.

Supported Platform(s): XM, XA, XS

Parameters

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

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.24 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getNrOfETHConnections (ABOUTHANDLE , unsigned char ∗ *NrOfConnections*)

Get number of ethernet connections present.

Supported Platform(s): XM, XA, XS

Parameters

NrOf-Connections	Returns the number of connections.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.25 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getNrOfSerialConnections (ABOUTHANDLE , unsigned char ∗ *NrOfConnections*)

Get number of serial port (RS232) connections present.

Supported Platform(s): XM, XA, XS

Parameters

NrOf-Connections	Returns the number of connections.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.26 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getNrOfUSBConnections (ABOUTHANDLE , unsigned char ∗ *NrOfConnections*)

Get number of USB connections present.

Supported Platform(s): XM, XA, XS

Parameters

NrOf-Connections	Returns the number of connections.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.27 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getNrOfVideoConnections (ABOUTHANDLE , unsigned char ∗ *NrOfConnections*)

Get number of Video connections present.

Supported Platform(s): XM, XA, XS

Parameters

NrOf-Connections	Returns the number of connections.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.28 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_getUnitSerial (ABOUTHANDLE , char ∗ *buff,* int *len*)

Get unit serial number.

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

Example Usage:

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

5.1.3.29 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::About_hasOsBooted (ABOUTHANDLE , bool ∗ *bootComplete*)

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

Supported Platform(s): XA, XS

Parameters

boot-Complete	Is the OS fully booted?

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.30 EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV CrossControl::About_release (ABOUTHANDLE)

Delete the About object.

Supported Platform(s): XM, XA, XS

Returns

-

Example Usage:

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

list_about_information(pAbout);

About_release(pAbout);
```

5.1.3.31 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Adc_getVoltage (ADCHANDLE , VoltageEnum *selection,* double ∗ *value*)

Read measured voltage.

Supported Platform(s): XM, XA, XS

Parameters

selection	The type of voltage to get.
value	Voltage value in Volt.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.32 EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV CrossControl::Adc_release (ADCHANDLE)

Delete the ADC object.

Supported Platform(s): XM, XA, XS

Returns

-

Example Usage:

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

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


Adc_release(pAdc);
```

5.1.3.33 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::AuxVersion_getCCAuxDrvVersion (AUXVERSIONHANDLE , unsigned char ∗ *major,* unsigned char ∗ *minor,* unsigned char ∗ *release,* unsigned char ∗ *build*)

Get the CrossControl CCAux CCAuxDrv version. Can be used to check that the correct driver is loaded. The version should be the same as that of AuxVersion_getCCAux-Version (Win32).

Supported Platform(s): XM

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. See the enum eErr for details.

Example Usage:

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

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

5.1.3.34 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::AuxVersion_getCCAuxVersion (AUXVERSIONHANDLE , unsigned char ∗ *major*, unsigned char ∗ *minor*, unsigned char ∗ *release*, unsigned char ∗ *build*)

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

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

Example Usage:

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

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

5.1.3.35 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::AuxVersion_getFPGAVersion (AUXVERSIONHANDLE , unsigned char ∗ *major,* unsigned char ∗ *minor,* unsigned char ∗ *release,* unsigned char ∗ *build*)

Get the FPGA software version

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

Example Usage:

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

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

5.1.3.36 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::AuxVersion_getFrontVersion (AUXVERSIONHANDLE , unsigned char ∗ *major,* unsigned char ∗ *minor,* unsigned char ∗ *release,* unsigned char ∗ *build*)

Get the front microcontroller software version

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

Example Usage:

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

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

5.1.3.37 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::AuxVersion_getOSVersion (AUXVERSIONHANDLE , unsigned char ∗ *major*, unsigned char ∗ *minor*, unsigned char ∗ *release*, unsigned char ∗ *build*)

Get the CrossControl Operating System version.

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

Example Usage:

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

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

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

5.1.3.38 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::AuxVersion_getSSVersion (AUXVERSIONHANDLE , unsigned char * *major*, unsigned char * *minor*, unsigned char * *release*, unsigned char * *build*)

Get the System Supervisor software version

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

Example Usage:

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

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

5.1.3.39 EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV CrossControl::AuxVersion_release (AUXVERSIONHANDLE)

Delete the AuxVersion object.

Supported Platform(s): XM, XA, XS

Returns

-

Example Usage:

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

output_versions(pAuxVersion);


AuxVersion_release(pAuxVersion);
```

5.1.3.40 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Backlight_getAutomaticBLFilter (BACKLIGHTHANDLE , unsigned long ∗ *averageWndSize,* unsigned long ∗ *rejectWndSize,* unsigned long ∗ *rejectDeltaInLux,* LightSensorSamplingMode ∗ *mode*)

Get light sensor filter parameters for automatic backlight control.

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

5.1.3.41 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Backlight_getAutomaticBLParams (BACKLIGHTHANDLE , bool ∗ *bSoftTransitions,* double ∗ *k*)

Get parameters for automatic backlight control.

Supported Platform(s): XM, XA, XS

Parameters

bSoftTransitions	Soft transitions used?
k	K value.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.42 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Backlight_getAutomaticBLStatus (BACKLIGHTHANDLE , unsigned char ∗ *status*)

Get status from automatic backlight control.

Supported Platform(s): XM, XA, XS

Parameters

status	1=running, 0=stopped.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.43 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Backlight_getHWStatus (BACKLIGHTHANDLE , bool ∗ *status*)

Get backlight hardware status.

Parameters

status	Backlight controller status. true: All backlight drivers works ok, false: one or more backlight drivers are faulty.

Supported Platform(s): XM, XA, XS

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

```
bool backlightStatus = false;
err = Backlight_getHWStatus(pBacklight, &backlightStatus);
if(err == ERR_SUCCESS)
{
  if(backlightStatus)
    printf("Backlight hardware status: OK\n");
  else
    printf("Backlight hardware status: not OK, one or more backlight drivers are faulty\n");
}
else if(err == ERR_NOT_SUPPORTED)
{
  printf("Backlight_getHWStatus: Not supported!\n");
}
else
{
  printf("Error(%d) in function Backlight_getHWStatus: %s\n", err,
    GetErrorStringA(err));
}
```

5.1.3.44 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Backlight_getIntensity (BACKLIGHTHANDLE , unsigned char ∗ *intensity*)

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

Supported Platform(s): XM, XA, XS

Parameters

intensity	The current backlight intensity (1..255).

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

```
  err = Backlight_getIntensity(pBacklight, &value);

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

5.1.3.45 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Backlight_getLedDimming (BACKLIGHTHANDLE , CCStatus ∗ *status*)

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

Supported Platform(s): XM, XA, XS

Parameters

status	Enabled/Disabled

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.46 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Backlight_getStatus (BACKLIGHTHANDLE , unsigned char * *status*)

Get backlight controller status. Deprecated, use Backlight_getHWStatus instead.

Supported Platform(s): XM

Parameters

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. See the enum eErr for details.

Example Usage:

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

5.1.3.47 EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV CrossControl::Backlight_release (BACKLIGHTHANDLE)

Delete the backlight object.

Supported Platform(s): XM, XA, XS

Returns

-

Example Usage:

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

change_backlight(pBacklight);

Backlight_release(pBacklight);
```

5.1.3.48 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Backlight_setAutomaticBLFilter (BACKLIGHTHANDLE , unsigned long *averageWndSize*, unsigned long *rejectWndSize*, unsigned long *rejectDeltaInLux*, LightSensorSamplingMode *mode*)

Set light sensor filter parameters for automatic backlight control.

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

5.1.3.49 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Backlight_setAutomaticBLParams (BACKLIGHTHANDLE , bool *bSoftTransitions*)

Set parameters for automatic backlight control.

Supported Platform(s): XM, XA, XS

Parameters

bSoftTransitions	Use soft transitions?

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.50 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Backlight_setIntensity (BACKLIGHTHANDLE , unsigned char *intensity*)

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

Supported Platform(s): XM, XA, XS

Parameters

intensity	The backlight intensity to set (1..255).

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

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

5.1.3.51 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Backlight_setLedDimming (BACKLIGHTHANDLE , CCStatus *status*)

Enable/disable Led dimming. If enabled, the function automatically 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.

Supported Platform(s): XM, XA, XS

Parameters

status	Enabled/Disabled

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.52 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Backlight_startAutomaticBL (BACKLIGHTHANDLE)

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

Supported Platform(s): XM, XA, XS

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.53 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Backlight_stopAutomaticBL (BACKLIGHTHANDLE)

Stop automatic backlight control.

Supported Platform(s): XM, XA, XS

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.54 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Battery_getBatteryChargingStatus (BATTERYHANDLE , ChargingStatus ∗ *status*)

Get battery charging status.

Supported Platform(s): XM

Parameters

status	the current charging mode of the battery.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.55 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Battery_getBatteryHWversion (BATTERYHANDLE , char ∗ *buff,* int *len*)

Get battery hardware version (PCB revision).

Supported Platform(s): XM

Parameters

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. See the enum eErr for details.

Example Usage:

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

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

5.1.3.56 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Battery_getBatterySerial (BATTERYHANDLE , char ∗ *buff*, int *len*)

Get battery serial number.

Supported Platform(s): XM

Parameters

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 serial number is 10 characters plus terminating zero, in total 11 bytes in size.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.57 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Battery_getBatterySwVersion (BATTERYHANDLE , unsigned short ∗ *major*, unsigned short ∗ *minor*, unsigned short ∗ *release*, unsigned short ∗ *build*)

Get the battery software version

Supported Platform(s): XM

Parameters

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. See the enum eErr for details.

Example Usage:

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

5.1.3.58 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Battery_getBatteryTemp (BATTERYHANDLE , signed short * *temperature*)

Get battery temperature.

Supported Platform(s): XM

Parameters

temperature	PCB Temperature in degrees Celsius.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

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

5.1.3.59 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Battery_getBatteryVoltageStatus (BATTERYHANDLE , unsigned char ∗ *batteryVoltagePercent*)

Get battery voltage status.

Supported Platform(s): XM

Parameters

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

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.60 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Battery_getHwErrorStatus (BATTERYHANDLE , ErrorStatus ∗ *errorCode*)

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

Supported Platform(s): XM

Parameters

errorCode	Error code. Zero means no error.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

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

5.1.3.61 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Battery_getMinMaxTemp (BATTERYHANDLE , signed short ∗ *minTemp*, signed short ∗ *maxTemp*)

Get temperature interval of the battery.

Supported Platform(s): XM

Parameters

minTemp	Minimum measured temperature.
maxTemp	Maximum measured temperature.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.62 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Battery_getPowerSource (BATTERYHANDLE , PowerSource ∗ *status*)

Get the currently used power source.

Supported Platform(s): XM

Parameters

status	the current power source, external power or battery.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

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

5.1.3.63 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Battery_getTimer (BATTERYHANDLE , BatteryTimerType ∗ *times*)

Get battery diagnostic timer.

Supported Platform(s): XM

Parameters

times	Get a struct with the current diagnostic times.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.64 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Battery_isBatteryPresent (BATTERYHANDLE , bool ∗ *batteryIsPresent*)

Is an external battery connected?

Supported Platform(s): XM

Parameters

batteryIs-Present	true if a battery is connected, otherwise false.

Returns

-

Example Usage:

```
error = Battery_isBatteryPresent(pBattery, &bpresent);

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

5.1.3.65 EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV CrossControl::Battery_release (BATTERYHANDLE)

Delete the Battery object

Supported Platform(s): XM.

Returns

-

Example Usage:

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

readBatteryInfo(pBattery);

Battery_release(pBattery);
```

5.1.3.66 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Buzzer_buzze (BUZZERHANDLE , int *time*, bool *blocking*)

Buzzes for a specified time.

Supported Platform(s): XM, XA, XS

Parameters

time	Time (ms) to buzz.
blocking	Blocking or non-blocking function.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.67 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Buzzer_getFrequency (BUZZERHANDLE , unsigned short ∗ *frequency*)

Get buzzer frequency.

Supported Platform(s): XM, XA, XS

Parameters

frequency	Current frequency (700-10000 Hz).

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.68 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Buzzer_getTrigger (BUZZERHANDLE , bool ∗ *trigger*)

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

Supported Platform(s): XM, XA, XS

Parameters

trigger	Current trigger status.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.69 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Buzzer_getVolume (BUZZERHANDLE , unsigned short ∗ *volume*)

Get buzzer volume.

Supported Platform(s): XM, XA, XS

Parameters

volume	Current volume (0-51).

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.70 EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV CrossControl::Buzzer_release (BUZZERHANDLE)

Delete the Buzzer object.

Supported Platform(s): XM, XA, XS

Returns

-

Example Usage:

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

play_beeps(pBuzzer);

Buzzer_release(pBuzzer);
```

5.1.3.71 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Buzzer_setFrequency (BUZZERHANDLE , unsigned short *frequency*)

Set buzzer frequency.

Supported Platform(s): XM, XA, XS

Parameters

frequency	Frequency to set (700-10000 Hz).

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.72 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Buzzer_setTrigger (BUZZERHANDLE , bool *trigger*)

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

Supported Platform(s): XM, XA, XS

Parameters

trigger	Status to set.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.73 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Buzzer_setVolume (BUZZERHANDLE , unsigned short *volume*)

Set buzzer volume.

Supported Platform(s): XM, XA, XS

Parameters

volume	Volume to set (0-51).

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.74 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::CanSetting_getBaudrate (CANSETTINGHANDLE , unsigned char *net*, unsigned short ∗ *baudrate*)

Get Baud rate

Supported Platform(s): XM, XA, XS

Parameters

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

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.75 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::CanSetting_getFrameType (CANSETTINGHANDLE , unsigned char *net*, CanFrameType ∗ *frameType*)

Get frame type

Supported Platform(s): XM, XA, XS

Parameters

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

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.76 EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV CrossControl::CanSetting_release (CANSETTINGHANDLE)

Delete the CanSetting object.

Supported Platform(s): XM, XA, XS

Returns

-

Example Usage:

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

read_cansettings(pCanSetting);

CanSetting_release(pCanSetting);
```

5.1.3.77 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::CanSetting_setBaudrate (CANSETTINGHANDLE , unsigned char *net*, unsigned short *baudrate*)

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

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

5.1.3.78 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::CanSetting_setFrameType (CANSETTINGHANDLE , unsigned char *net*, CanFrameType *frameType*)

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

Supported Platform(s): XM, XA, XS

Parameters

net	CAN net (1-4).
frameType	CAN frameType

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.79 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_getCanStartupPowerConfig (CONFIGHANDLE , CCStatus ∗ *status*)

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

Supported Platform(s): XM, XA, XS

Parameters

status	Enabled/Disabled

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.80 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_getExtFanStartupPowerConfig (CONFIGHANDLE , CCStatus * *status*)

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

Supported Platform(s): XM

Parameters

status	Enabled/Disabled

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.81 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_getExtOnOffSigTrigTime (CONFIGHANDLE , unsigned long * *triggertime*)

Get external on/off signal trigger time.

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

5.1.3.82 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_getFrontBtnTrigTime (CONFIGHANDLE , unsigned short * *triggertime*)

Get front button trigger time for long press.

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

5.1.3.83 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_getHeatingTempLimit (CONFIGHANDLE , signed short ∗ *temperature*)

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

Supported Platform(s): XM, XA, XS

Parameters

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

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.84 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_getLongButtonPressAction (CONFIGHANDLE , PowerAction ∗ *action*)

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

Supported Platform(s): XM, XA, XS

Parameters

action	The configured action.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.85 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_getOnOffSigAction (CONFIGHANDLE , PowerAction ∗ *action*)

Get On/Off signal action. Gets the configured action for an On/Off signal event: NoAction, ActionSuspend or ActionShutDown. An On/Off signal event is determined by the ExtOnOffSigTrigTime.

Supported Platform(s): XM, XA, XS

Parameters

action	The configured action.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.86 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_getPowerOnStartup (CONFIGHANDLE , CCStatus ∗ *status*)

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

Supported Platform(s): XM, XA, XS

Parameters

status	Enabled/Disabled

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.87 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_getRS485Enabled (CONFIGHANDLE , RS4XXPort *port,* bool ∗ *enabled*)

Get RS485 mode configuration for RS4XX port.

Supported Platform(s): XA, XS

Parameters

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

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.88 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_getShortButtonPressAction (CONFIGHANDLE , PowerAction ∗ *action*)

Get short button press action. Gets the configured action for a short button press: NoAction, ActionSuspend or ActionShutDown.

Supported Platform(s): XM, XA, XS

Parameters

action	The configured action.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.89 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_getStartupTriggerConfig (CONFIGHANDLE , TriggerConf ∗ *config*)

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

Supported Platform(s): XM, XA, XS

Parameters

config	One of: Front_Button_Enabled, OnOff_Signal_Enabled or Both_Button_And_Signal_Enabled.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

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

5.1.3.90 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_getStartupVoltageConfig (CONFIGHANDLE , double ∗ *voltage*)

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

Supported Platform(s): XM

Parameters

voltage	The current voltage setting. (9V .. 28V)

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.91 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_getSuspendMaxTime (CONFIGHANDLE , unsigned short ∗ *maxTime*)

Get suspend mode maximum time.

Supported Platform(s): XM

Parameters

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. See the enum eErr for details.

5.1.3.92 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_getVideoStartupPowerConfig (CONFIGHANDLE , unsigned char * *config*)

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

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

5.1.3.93 EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV CrossControl::Config_release (CONFIGHANDLE)

Delete the Config object.

Supported Platform(s): XM, XA, XS

Returns

-

Example Usage:

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

conf_example(pConfig);

Config_release(pConfig);
```

5.1.3.94 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_setCanStartupPowerConfig (CONFIGHANDLE , CCStatus *status*)

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

Supported Platform(s): XM, XA, XS

Parameters

status	Enabled/Disabled

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.95 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_setExtFanStartupPowerConfig (CONFIGHANDLE , CCStatus *status*)

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

Supported Platform(s): XM

Parameters

status	Enabled/Disabled

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.96 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_setExtOnOffSigTrigTime (CONFIGHANDLE , unsigned long *triggertime*)

Set external on/off signal trigger time.

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

5.1.3.97 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_setFrontBtnTrigTime (CONFIGHANDLE , unsigned short *triggertime*)

Set front button trigger time for long press.

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

5.1.3.98 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_setHeatingTempLimit (CONFIGHANDLE , signed short *temperature*)

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

Supported Platform(s): XM, XA, XS

Parameters

temperature	The heating limit, in degrees Celsius (-25 to +5)

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.99 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_setLongButtonPressAction (CONFIGHANDLE , PowerAction *action*)

Set long button press action. Sets the configured action for a long button press: NoAction, ActionSuspend or ActionShutDown. A long button press is determined by the FrontBtnTrigTime.

Supported Platform(s): XM, XA, XS

Parameters

action	The action to set.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.100 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_setOnOffSigAction (CONFIGHANDLE , PowerAction *action*)

Set On/Off signal action. Sets the configured action for an On/Off signal event: NoAction, ActionSuspend or ActionShutDown. An On/Off signal event is determined by the ExtOnOffSigTrigTime.

Supported Platform(s): XM, XA, XS

Parameters

action	The action to set.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.101 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_setPowerOnStartup (CONFIGHANDLE , CCStatus *status*)

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

Supported Platform(s): XM, XA, XS

Parameters

status	Enabled/Disabled

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

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

Set RS485 mode enabled or disabled for RS4XX port.

Supported Platform(s): XA, XS

Parameters

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

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

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

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

Supported Platform(s): XM, XA, XS

Parameters

action	The action to set.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.104 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_setStartupTriggerConfig (CONFIGHANDLE , TriggerConf *conf*)

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

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

5.1.3.105 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_setStartupVoltageConfig (CONFIGHANDLE , double *voltage*)

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

Supported Platform(s): XM

Parameters

voltage	The voltage to set (9V .. 28V).

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.106 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_setSuspendMaxTime (CONFIGHANDLE , unsigned short *maxTime*)

Set suspend mode maximum time.

Supported Platform(s): XM

Parameters

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. See the enum eErr for details.

5.1.3.107 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_setVideoStartupPowerConfig (CONFIGHANDLE , unsigned char *config*)

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

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

5.1.3.108 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Diagnostic_clearHwErrorStatus (DIAGNOSTICHANDLE)

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

Supported Platform(s): XM, XA, XS

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.109 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Diagnostic_getHwErrorStatus (DIAGNOSTICHANDLE , unsigned short ∗ *errorCode*)

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

Supported Platform(s): XM, XA, XS

Parameters

errorCode	Error code. Zero means no error.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.110 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Diagnostic_getMinMaxTemp (DIAGNOSTICHANDLE , signed short ∗ *minTemp*, signed short ∗ *maxTemp*)

Get diagnostic temperature interval of the unit.

Supported Platform(s): XM, XA, XS

Parameters

minTemp	Minimum measured PCB temperature.
maxTemp	Maximum measured PCB temperature.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.111 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Diagnostic_getPCBTemp (DIAGNOSTICHANDLE , signed short ∗ *temperature*)

Get PCB temperature.

Supported Platform(s): XM, XA, XS

Parameters

temperature	PCB Temperature in degrees Celsius.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.112 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Diagnostic_getPMTemp (DIAGNOSTICHANDLE , unsigned char *index*, signed short ∗ *temperature*, JidaSensorType ∗ *jst*)

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

Parameters

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.

Supported Platform(s): XM

Parameters

temperature	Temperature in degrees Celsius.
jst	The type of sensor that is being read.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.113 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Diagnostic_getPowerCycles (DIAGNOSTICHANDLE , unsigned short ∗ *powerCycles*)

Get number of power cycles.

Supported Platform(s): XM, XA, XS

Parameters

powerCycles	Total number of power cycles.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.114 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Diagnostic_getShutDownReason (DIAGNOSTICHANDLE , unsigned short ∗ *reason*)

Get shutdown reason.

Supported Platform(s): XM

Parameters

reason	See DiagnosticCodes.h for shutdown codes.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.115 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Diagnostic_getSSTemp (DIAGNOSTICHANDLE , signed short ∗ *temperature*)

Get System Supervisor temperature.

Supported Platform(s): XM, XA, XS

Parameters

temperature	System Supervisor temperature in degrees Celsius.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.116 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Diagnostic_getStartupReason (DIAGNOSTICHANDLE , unsigned short ∗ *reason*)

Get startup reason.

Supported Platform(s): XM, XA, XS

Parameters

reason	See DiagnosticCodes.h for startup codes.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.117 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Diagnostic_getTimer (DIAGNOSTICHANDLE , TimerType ∗ *times*)

Get diagnostic timer.

Supported Platform(s): XM, XA, XS

Parameters

times	Get a struct with the current diagnostic times.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.118 EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV CrossControl::Diagnostic_release (DIAGNOSTICHANDLE)

Delete the Diagnostic object.

Supported Platform(s): XM, XA, XS

Returns

-

Example Usage:

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

diagnostic_example(pDiagnostic);

Diagnostic_release(pDiagnostic);
```

5.1.3.119 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::DigIO_getDigIO (DIGIOHANDLE , unsigned char ∗ *status*)

Get Digital inputs.

Supported Platform(s): XM, XA, XS

Parameters

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 zero.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.120 EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV CrossControl::DigIO_release (DIGIOHANDLE)

Delete the DigIO object.

Supported Platform(s): XM, XA, XS

Returns

-

Example Usage:

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

list_digital_inputs(pDigIO);

DigIO_release(pDigIO);
```

5.1.3.121 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::DigIO_setDigIO (DIGIOHANDLE , unsigned char *state*)

Set Digital outputs.

Supported Platform(s): XA, XS

Parameters

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. See the enum eErr for details.

Example Usage:

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

5.1.3.122 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FirmwareUpgrade_getUpgradeStatus (FIRMWAREUPGHANDLE , UpgradeStatus ∗ *status,* bool *blocking*)

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

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

5.1.3.123 EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV CrossControl::FirmwareUpgrade_release (FIRMWAREUPGHANDLE)

Delete the FirmwareUpgrade object.

Supported Platform(s): XM, XA, XS

Returns

-

Example Usage:

```
FirmwareUpgrade_release(pFirmwareUpgrade);
```

5.1.3.124 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FirmwareUpgrade_shutDown (FIRMWAREUPGHANDLE)

Shut down the operating system.

Supported Platform(s): XM, XA, XS

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.125 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FirmwareUpgrade_startFpgaUpgrade (FIRMWAREUPGHANDLE , const char * *filename*, bool *blocking*)

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

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

Example Usage:

```
cout << "Upgrading FPGA" << endl;

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

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

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

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

5.1.3.126 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FirmwareUpgrade_startFpgaVerification (FIRMWAREUPGHANDLE , const char ∗ *filename,* bool *blocking*)

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

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

Example Usage:

```
cout << "Upgrading FPGA" << endl;

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

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

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

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

5.1.3.127 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FirmwareUpgrade_startFrontUpgrade (FIRMWAREUPGHANDLE , const char ∗ *filename*, bool *blocking*)

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

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

Example Usage:

```
cout << "Upgrading front" << endl;

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

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

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

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

5.1.3.128 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FirmwareUpgrade_startFrontVerification (FIRMWAREUPGHANDLE , const char ∗ *filename,* bool *blocking*)

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

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

Example Usage:

```
cout << "Upgrading front" << endl;

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

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

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

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

5.1.3.129 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FirmwareUpgrade_startSSUpgrade (FIRMWAREUPGHANDLE , const char ∗ *filename,* bool *blocking*)

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

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

Example Usage:

```
cout << "Upgrading SS" << endl;

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

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

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

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

5.1.3.130 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FirmwareUpgrade_startSSVerification (FIRMWAREUPGHANDLE , const char ∗ *filename*, bool *blocking*)

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

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

Example Usage:

```
cout << "Upgrading SS" << endl;

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

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

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

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

5.1.3.131 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FrontLED_getColor (FRONTLEDHANDLE , unsigned char ∗ *red*, unsigned char ∗ *green*, unsigned char ∗ *blue*)

Get front LED color mix.

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

Example Usage:

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

5.1.3.132 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FrontLED_getEnabledDuringStartup (FRONTLEDHANDLE , CCStatus ∗ *status*)

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

Supported Platform(s): XM, XA, XS

Parameters

status	LED Enabled or Disabled during startup.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.133 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FrontLED_getIdleTime (FRONTLEDHANDLE , unsigned char ∗ *idleTime*)

Get front LED idle time.

Supported Platform(s): XM, XA, XS

Parameters

idleTime	Time in 100ms increments.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.134 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FrontLED_getNrOfPulses (FRONTLEDHANDLE , unsigned char ∗ *nrOfPulses*)

Get number of pulses during a blink sequence.

Supported Platform(s): XM, XA, XS

Parameters

nrOfPulses	Number of pulses.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.135 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FrontLED_getOffTime (FRONTLEDHANDLE , unsigned char * *offTime*)

Get front LED off time.

Supported Platform(s): XM, XA, XS

Parameters

offTime	Time in 10ms increments.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.136 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FrontLED_getOnTime (FRONTLEDHANDLE , unsigned char * *onTime*)

Get front LED on time.

Supported Platform(s): XM, XA, XS

Parameters

onTime	Time in 10ms increments. 0 = off

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.137 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FrontLED_getSignal (FRONTLEDHANDLE , double * *frequency*, unsigned char * *dutyCycle*)

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

Supported Platform(s): XM, XA, XS

Parameters

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

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.138 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FrontLED_getStandardColor (FRONTLEDHANDLE , CCAuxColor * *color*)

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

Supported Platform(s): XM, XA, XS

Parameters

color	Color from CCAuxColor enum.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.139 EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV CrossControl::FrontLED_release (FRONTLEDHANDLE)

Delete the FrontLED object.

Supported Platform(s): XM, XA, XS

Returns

-

Example Usage:

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

led_example(pFrontLED);

FrontLED_release(pFrontLED);
```

5.1.3.140 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FrontLED_setColor (FRONTLEDHANDLE , unsigned char *red*, unsigned char *green*, unsigned char *blue*)

Set front LED color mix.

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

Example Usage:

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

5.1.3.141 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FrontLED_setEnabledDuringStartup (FRONTLEDHANDLE , CCStatus *status*)

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

Supported Platform(s): XM, XA, XS

Parameters

status	Enable or Disable the LED during startup.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.142 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FrontLED_setIdleTime (FRONTLEDHANDLE , unsigned char *idleTime*)

Get front LED idle time.

Supported Platform(s): XM, XA, XS

Parameters

idleTime	Time in 100ms.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.143 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FrontLED_setNrOfPulses (FRONTLEDHANDLE , unsigned char *nrOfPulses*)

Set front LED number of pulses during a blink sequence.

Supported Platform(s): XM, XA, XS

Parameters

nrOfPulses	Number of pulses.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.144 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FrontLED_setOff (FRONTLEDHANDLE)

Set front LED off.

Supported Platform(s): XM, XA, XS

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.145 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FrontLED_setOffTime (FRONTLEDHANDLE , unsigned char *offTime*)

Set front LED off time.

Supported Platform(s): XM, XA, XS

Parameters

offTime	Time in 10ms increments.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.146 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FrontLED_setOnTime (FRONTLEDHANDLE , unsigned char *onTime*)

Set front LED on time.

Supported Platform(s): XM, XA, XS

Parameters

onTime	Time in 10ms increments. 0 = off

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.147 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FrontLED_setSignal (FRONTLEDHANDLE , double *frequency*, unsigned char *dutyCycle*)

Set front LED signal.

Supported Platform(s): XM, XA, XS

Parameters

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

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.148 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::FrontLED_setStandardColor (FRONTLEDHANDLE , CCAuxColor *color*)

Set one of the front LED standard colors.

Supported Platform(s): XM, XA, XS

Parameters

color	Color from CCAuxColor enum.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.149 EXTERN_C CCAUXDLL_API ABOUTHANDLE CCAUXDLL_CALLING_CONV CrossControl::GetAbout (void)

Factory function that creates instances of the About object.

Supported Platform(s): XM, XA, XS

Returns

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

Returns NULL if it fails to allocate memory.

Example Usage:

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

list_about_information(pAbout);

About_release(pAbout);
```

5.1.3.150 EXTERN_C CCAUXDLL_API ADCHANDLE CCAUXDLL_CALLING_CONV CrossControl::GetAdc (void)

Factory function that creates instances of the Adc object.

Supported Platform(s): XM, XA, XS

Returns

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

Returns NULL if it fails to allocate memory.

Example Usage:

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

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


Adc_release(pAdc);
```

5.1.3.151 EXTERN_C CCAUXDLL_API AUXVERSIONHANDLE CCAUXDLL_CALLING_CONV CrossControl::GetAuxVersion (void)

Factory function that creates instances of the AuxVersion object.

Supported Platform(s): XM, XA, XS

Returns

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

Example Usage:

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

output_versions(pAuxVersion);


AuxVersion_release(pAuxVersion);
```

5.1.3.152 EXTERN_C CCAUXDLL_API BACKLIGHTHANDLE CCAUXDLL_CALLING_CONV CrossControl::GetBacklight (void)

Factory function that creates instances of the Backlight object.

Supported Platform(s): XM, XA, XS

Returns

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

Example Usage:

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

change_backlight(pBacklight);

Backlight_release(pBacklight);
```

5.1.3.153 EXTERN_C CCAUXDLL_API BATTERYHANDLE CCAUXDLL_CALLING_CONV CrossControl::GetBattery (void)

Factory function that creates instances of the Battery object.

Supported Platform(s): XM

Returns

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

Example Usage:

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

readBatteryInfo(pBattery);

Battery_release(pBattery);
```

5.1.3.154 EXTERN_C CCAUXDLL_API BUZZERHANDLE CCAUXDLL_CALLING_CONV CrossControl::GetBuzzer (void)

Factory function that creates instances of the Buzzer object.

Supported Platform(s): XM, XA, XS

Returns

BUZZERHANDLE to an allocated Buzzer object. The returned handle needs to be deallocated using the Buzzer_release(BUZZERHANDLE) method when it's no longer needed. Returns NULL if it fails to allocate memory.

Example Usage:

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

play_beeps(pBuzzer);

Buzzer_release(pBuzzer);
```

5.1.3.155 EXTERN_C CCAUXDLL_API CANSETTINGHANDLE CCAUXDLL_CALLING_CONV CrossControl::GetCanSetting (void)

Factory function that creates instances of the CanSetting object.

Supported Platform(s): XM, XA, XS

Returns

CANSETTINGHANDLE to an allocated CanSetting object. The returned handle needs to be deallocated using the CanSetting_release(CANSETTINGHANDLE) method when it's no longer needed. Returns NULL if it fails to allocate memory.

Example Usage:

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

read_cansettings(pCanSetting);

CanSetting_release(pCanSetting);
```

5.1.3.156 EXTERN_C CCAUXDLL_API CONFIGHANDLE CCAUXDLL_CALLING_CONV CrossControl::GetConfig ( )

Video channel 4 config

Factory function that creates instances of the Config object.

Supported Platform(s): XM, XA, XS

Returns

CONFIGHANDLE to an allocated Config object. The returned handle needs to be deallocated using the Config_release(CONFIGHANDLE) method when it's no longer needed. Returns NULL if it fails to allocate memory.

Example Usage:

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

conf_example(pConfig);

Config_release(pConfig);
```

5.1.3.157 EXTERN_C CCAUXDLL_API DIAGNOSTICHANDLE CCAUXDLL_CALLING_CONV CrossControl::GetDiagnostic (void)

Factory function that creates instances of the Diagnostic object.

Supported Platform(s): XM, XA, XS

Returns

DIAGNOSTICHANDLE to an allocated Diagnostic object. The returned handle needs to be deallocated using the Diagnostic_release(DIAGNOSTICHANDLE) method when it's no longer needed. Returns NULL if it fails to allocate memory.

Example Usage:

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

diagnostic_example(pDiagnostic);

Diagnostic_release(pDiagnostic);
```

5.1.3.158 EXTERN_C CCAUXDLL_API DIGIOHANDLE CCAUXDLL_CALLING_CONV CrossControl::GetDigIO (void)

Factory function that creates instances of the DigIO object.

Supported Platform(s): XM, XA, XS

Returns

DIGIOHANDLE to an allocated DigIO object. The returned handle needs to be deallocated using the DigIO_release(DIGIOHANDLE) method when it's no longer needed. Returns NULL if it fails to allocate memory.

Example Usage:

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

list_digital_inputs(pDigIO);

DigIO_release(pDigIO);
```

5.1.3.159 EXTERN_C CCAUXDLL_API char const∗ CCAUXDLL_CALLING_CONV CrossControl::GetErrorStringA (eErr *errCode*)

Get a string description of an error code.

Parameters

errCode	An error code for which to get a string description.

Returns

String description of an error code.

5.1.3.160 EXTERN_C CCAUXDLL_API wchar_t const∗ CCAUXDLL_CALLING_CONV CrossControl::GetErrorStringW (eErr *errCode*)

Get a string description of an error code.

Parameters

errCode	An error code for which to get a string description.

Returns

String description of an error code.

5.1.3.161 EXTERN_C CCAUXDLL_API FIRMWAREUPGHANDLE CCAUXDLL_CALLING_CONV CrossControl::GetFirmwareUpgrade (void)

Factory function that creates instances of the Adc object.

Supported Platform(s): XM, XA, XS

Returns

FIRMWAREUPGHANDLE to an allocated FirmwareUpgrade object. The returned handle needs to be deallocated using the FirmwareUpgrade_release(FIRMWAREUPGHANDLE) method when it's no longer needed. Returns NULL if it fails to allocate memory.

Example Usage:

```
FIRMWAREUPGHANDLE pFirmwareUpgrade = GetFirmwareUpgrade();
assert(pFirmwareUpgrade != NULL);
```

5.1.3.162 EXTERN_C CCAUXDLL_API FRONTLEDHANDLE CCAUXDLL_CALLING_CONV CrossControl::GetFrontLED (void)

Factory function that creates instances of the FrontLED object.

Supported Platform(s): XM, XA, XS

Returns

FRONTLEDHANDLE to an allocated FrontLED object. The returned handle needs to be deallocated using the FrontLED_release(FRONTLEDHANDLE) method when it's no longer needed. Returns NULL if it fails to allocate memory.

Example Usage:

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

led_example(pFrontLED);

FrontLED_release(pFrontLED);
```

5.1.3.163 EXTERN_C CCAUXDLL_API char const∗ CCAUXDLL_CALLING_CONV CrossControl::GetHwErrorStatusStringA (unsigned short *errCode*)

Get a string description of an error code returned from getHwErrorStatus.

Parameters

errCode	An error code for which to get a string description.

Returns

String description of an error code.

5.1.3.164 EXTERN_C CCAUXDLL_API wchar_t const∗ CCAUXDLL_CALLING_CONV CrossControl::GetHwErrorStatusStringW (unsigned short *errCode*)

Get a string description of an error code returned from getHwErrorStatus.

Parameters

errCode	An error code for which to get a string description.

Returns

String description of an error code.

5.1.3.165 EXTERN_C CCAUXDLL_API LIGHTSENSORHANDLE CCAUXDLL_CALLING_CONV CrossControl::GetLightsensor (void)

Factory function that creates instances of the Lightsensor object.

Supported Platform(s): XM, XA, XS

Returns

LIGHTSENSORHANDLE to an allocated Lightsensor object. The returned handle needs to be deallocated using the Lightsensor_release(LIGHTSENSORHANDLE) method when it's no longer needed. Returns NULL if it fails to allocate memory.

Example Usage:

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

ls_example(pLightSensor);

Lightsensor_release(pLightSensor);
```

5.1.3.166 EXTERN_C CCAUXDLL_API POWERHANDLE CCAUXDLL_CALLING_CONV CrossControl::GetPower (void)

Factory function that creates instances of the Power object.

Supported Platform(s): XM, XA, XS

Returns

POWERHANDLE to an allocated Power object. The returned handle needs to be deallocated using the Power_release(POWERHANDLE) method when it's no longer needed. Returns NULL if it fails to allocate memory.

Example Usage:

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

power_example(pPower);

Power_release(pPower);
```

5.1.3.167 EXTERN_C CCAUXDLL_API POWERMGRHANDLE CCAUXDLL_CALLING_CONV CrossControl::GetPowerMgr (void)

Factory function that creates instances of the PowerMgr object.

Supported Platform(s): XM, XA, XS

Returns

POWERMGRHANDLE to an allocated PowerMgr structure. The returned handle needs to be deallocated using the PowerMgr::Release() method when it's no longer needed. Returns NULL if it fails to allocate memory.

Example Usage:

```
CrossControl::eErr err;
POWERMGRHANDLE pPowerMgr = ::GetPowerMgr();
BATTERYHANDLE pBattery = ::GetBattery();

assert(pPowerMgr);
assert(pBattery);

// Register a separate exit handler for the case where OS is initiating the shutdown. The Application
     must handle this case itself.
atexit(fnExit);

bool bBatt = false;
Battery_isBatteryPresent(pBattery, &bBatt);
if(bBatt)   // Ask user wich configuration to use...
  cout << "Choose configuration to use, 0 - Normal, 1 - Application Controlled , 2 - Battery Suspend"  <<
     endl;
else
  cout << "Choose configuration to use, 0 - Normal, 1 - Application Controlled"  << endl;

cin >> suspendConfiguration;
Battery_release(pBattery);

// Register that this application needs to delay suspend/shutdown
// This should be done as soon as possible.
// Then the app must poll getPowerMgrStatus() and allow the suspend/shutdown with
     setAppReadyForSuspendOrShutdown().
// Depending on application design, this might be best handled in a separate thread.
err = PowerMgr_registerControlledSuspendOrShutDown(pPowerMgr,
     (PowerMgrConf) suspendConfiguration);

cout << "suspendConfiguration " << suspendConfiguration << endl;

if(err == ERR_SUCCESS)
  cout << "Registered to powerMgr." << endl;
else
  cout << "Error(" << err << ") in function registerControlledSuspendOrShutDown: " <<
    GetErrorStringA(err) << endl;

test_powermgr(pPowerMgr);

PowerMgr_release(pPowerMgr);
```

5.1.3.168 EXTERN_C CCAUXDLL_API SMARTHANDLE CCAUXDLL_CALLING_CONV CrossControl::GetSmart (void)

Factory function that creates instances of the Smart object.

Supported Platform(s): XM, XA, XS

Returns

SMARTHANDLE to an allocated AuxVersion structure. The returned handle needs to be deallocated using the Smart::Release() method when it's no longer needed. Returns NULL if it fails to allocate memory.

Example Usage:

```
SMARTHANDLE pSmart = ::GetSmart();
assert(pSmart);

show_card_data(pSmart);

Smart_release(pSmart);
```

5.1.3.169 EXTERN_C CCAUXDLL_API char const∗ CCAUXDLL_CALLING_CONV CrossControl::GetStartupReasonStringA (unsigned short *code*)

Get a string description of a startup reason code returned from getStartupReason.

Parameters

code	A code for which to get a string description.

Returns

String description of a code.

5.1.3.170 EXTERN_C CCAUXDLL_API wchar_t const∗ CCAUXDLL_CALLING_CONV CrossControl::GetStartupReasonStringW (unsigned short *code*)

Get a string description of a startup reason code returned from getStartupReason.

Parameters

code	A code for which to get a string description.

Returns

String description of a code.

5.1.3.171 EXTERN_C CCAUXDLL_API TELEMATICSHANDLE CCAUXDLL_CALLING_CONV CrossControl::GetTelematics (void)

Factory function that creates instances of the Telematics object.

Supported Platform(s): XM, XA, XS

Returns

TELEMATICSHANDLE to an allocated Telematics object. The returned handle needs to be deallocated using the Telematics_release(TELEMATICSHANDLE) method when it's no longer needed. Returns NULL if it fails to allocate memory.

Example Usage:

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

telematics_example(pTelematics);

Telematics_release(pTelematics);
```

5.1.3.172 EXTERN_C CCAUXDLL_API TOUCHSCREENHANDLE CCAUXDLL_CALLING_CONV CrossControl::GetTouchScreen (void)

Factory function that creates instances of the TouchScreen object.

Supported Platform(s): XM, XA, XS

Returns

TOUCHSCREENHANDLE to an allocated TouchScreen object. The returned handle needs to be deallocated using the TouchScreen_release(TOUCHSCREE-NHANDLE) method when it's no longer needed. Returns NULL if it fails to allocate memory.

Example Usage:

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

touchscreen_example(pTouchScreen);

TouchScreen_release(pTouchScreen);
```

5.1.3.173 EXTERN_C CCAUXDLL_API TOUCHSCREENCALIBHANDLE CCAUXDLL_CALLING_CONV CrossControl::GetTouchScreenCalib (void)

Factory function that creates instances of the TouchScreenCalib object.

Returns

TOUCHSCREENCALIBHANDLE to an allocated TouchScreenCalib object. The returned handle needs to be deallocated using the TouchScreenCalib_release(T-OUCHSCREENCALIBHANDLE) method when it's no longer needed. Returns NULL if it fails to allocate memory.

5.1.3.174 EXTERN_C CCAUXDLL_API VIDEOHANDLE CCAUXDLL_CALLING_CONV CrossControl::GetVideo (void)

Factory function that creates instances of the Video object.

Supported Platform(s): XM, XA, XS

Returns

VIDEOHANDLE to an allocated Video object. The returned handle needs to be deallocated using the Video_release(VIDEOHANDLE) method when it's no longer needed. Returns NULL if it fails to allocate memory.

5.1.3.175 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Lightsensor_getAverageIlluminance (LIGHTSENSORHANDLE , unsigned short ∗ *value*)

Get average illuminance (light) value from light sensor.

Supported Platform(s): XM, XA, XS

Parameters

value	Illuminance value (Lux).

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.176 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Lightsensor_getIlluminance (LIGHTSENSORHANDLE , unsigned short ∗ *value*)

Get illuminance (light) value from light sensor.

Supported Platform(s): XM, XA, XS

Parameters

value	Illuminace value (Lux).

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.177 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Lightsensor_getIlluminance2 (LIGHTSENSORHANDLE , unsigned short ∗ *value,* unsigned char ∗ *ch0,* unsigned char ∗ *ch1*)

Get illuminance (light) value from light sensor. The parameters cho and ch1 are raw ADC values read from a TAOS TSL2550 lightsensor.

Supported Platform(s): XM, XA, XS

Parameters

value	Illuminance value (Lux).
ch0	Channel0 value.
ch1	Channel1 value.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.178 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Lightsensor_getOperatingRange (LIGHTSENSORHANDLE , LightSensorOperationRange ∗ *range*)

Get operating range. The light sensor can operate in two ranges. Standard and extended range. In standard range, the range is smaller but resolution higher. See the TSL2550 data sheet for more information.

Supported Platform(s): XM, XA, XS

Parameters

range	Operating range. RangeStandard or RangeExtended.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code.

5.1.3.179 EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV CrossControl::Lightsensor_release (LIGHTSENSORHANDLE)

Delete the Lightsensor object.

Supported Platform(s): XM, XA, XS

Returns

-

Example Usage:

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

ls_example(pLightSensor);

Lightsensor_release(pLightSensor);
```

5.1.3.180 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Lightsensor_setOperatingRange (LIGHTSENSORHANDLE , LightSensorOperationRange *range*)

Set operating range. The light sensor can operate in two ranges. Standard and extended range. In standard range, the range is smaller but resolution higher. See the TSL2550 data sheet for more information.

Supported Platform(s): XM, XA, XS

Parameters

range	Operating range to set. RangeStandard or RangeExtended.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.181 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Lightsensor_startAverageCalc (LIGHTSENSORHANDLE , unsigned long *averageWndSize*, unsigned long *rejectWndSize*, unsigned long *rejectDeltaInLux*, LightSensorSamplingMode *mode*)

Start average calculation.

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

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;
}
```

5.1.3.182 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Lightsensor_stopAverageCalc (LIGHTSENSORHANDLE)

Stop average calculation.

Supported Platform(s): XM, XA, XS

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

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

5.1.3.183 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Power_ackPowerRequest (POWERHANDLE)

Acknowledge a power request from the system supervisor. This is handled by the service/daemon and should normally not be used by applications unless the 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

Supported Platform(s): XM, XA, XS

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.184 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Power_getBLPowerStatus (POWERHANDLE , CCStatus ∗ *status*)

Get backlight power status.

Supported Platform(s): XM

Parameters

status	Backlight power status.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

```
err = Power_getBLPowerStatus(pPower, &status);
if(err == ERR_SUCCESS)
{
  cout << "Backlight power is " << ((status == Enabled)? "ON" : "OFF") << endl;
}
else
{
  cout << "Error(" << err << ") in function Power_getBLPowerStatus: " <<
    GetErrorStringA(err) << endl;
}
```

5.1.3.185 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Power_getButtonPowerTransitionStatus (POWERHANDLE , ButtonPowerTransitionStatus ∗ *status*)

Get the current status for front panel button and on/off signal.

Supported Platform(s): XM, XA, XS

Parameters

status	The current status. See the definition of ButtonPowerTransitionStatus for details.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.186 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Power_getCanOCDStatus (POWERHANDLE , OCDStatus ∗ *status*)

Get Can power overcurrent detection status. Find out if the Can power supervision has detected overcurrent, likely caused by short circuit problems. The overcurrent detection system will immediately turn of the power if such a condition occurs. After a short while, the system will test again, and if there still is overcurrent, Can power is turned off permanently until the unit is restarted.

Supported Platform(s): XA, XS

Parameters

status	The current overcurrent detection status

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

```
cout << "Checking overcurrent status... " << endl;
OCDStatus ocdstatus;
err = Power_getCanOCDStatus(pPower, &ocdstatus);
if(err == ERR_NOT_SUPPORTED)
{
  cout << "Not supported." << endl;
}
else if(err != ERR_SUCCESS)
{
  cout << "Error(" << err << ") in function Power_getCanOCDStatus: " <<
    GetErrorStringA(err) << endl;
}
else
{
  cout << "Power_getCanOCDStatus: Can OCD status is: ";
  switch(ocdstatus)
  {
  case OCD_OK: cout << "OCD_OK" << std::endl; break;
  case OCD_OC: cout << "OCD_OC" << std::endl; break;
  case OCD_POWER_OFF: cout << "OCD_POWER_OFF" << std::endl; break;
  default: cout << "ERROR" <<  std::endl; break;
  }
}
```

5.1.3.187 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Power_getCanPowerStatus (POWERHANDLE , CCStatus ∗ *status*)

Get can power status.

Supported Platform(s): XM, XA, XS

Parameters

status	Can power status.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.188 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Power_getExtFanPowerStatus (POWERHANDLE , CCStatus ∗ *status*)

Get external fan power status.

Supported Platform(s): XM

Parameters

status	Fan power status.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.189 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Power_getVideoOCDStatus (POWERHANDLE , OCDStatus ∗ *status*)

Get Video power overcurrent detection status. Find out if the video power supervision has detected overcurrent, likely caused by short circuit problems. The overcurrent detection system will immediately turn of the power if such a condition occurs. After a short while, the system will test again, and if there still is overcurrent, video power is turned off permanently until the unit is restarted.

Supported Platform(s): XA, XS

Parameters

status	The current overcurrent detection status

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

```
err = Power_getVideoOCDStatus(pPower, &ocdstatus);
if(err == ERR_NOT_SUPPORTED)
```

```
{
  /* Don't print anything */
}
else
  if(err != ERR_SUCCESS)
  {
    cout << "Error(" << err << ") in function Power_getVideoOCDStatus: " <<
    GetErrorStringA(err) << endl;
  }
  else
  {
    cout << "Power_getVideoOCDStatus: Video OCD status is: ";
    switch(ocdstatus)
    {
    case OCD_OK: cout << "OCD_OK" << std::endl; break;
    case OCD_OC: cout << "OCD_OC" << std::endl; break;
    case OCD_POWER_OFF: cout << "OCD_POWER_OFF" << std::endl; break;
    default: cout << "ERROR" <<  std::endl; break;
    }
  }
```

5.1.3.190 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Power_getVideoPowerStatus (POWERHANDLE , unsigned char ∗ *videoStatus*)

Get Video power status.

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

Example Usage:

```
err = Power_getVideoPowerStatus(pPower, &value);
if(err == ERR_SUCCESS)
{
  cout << "Video power status: " << endl;
  cout << "Video1: " << ((value & 0x01)? "ON" : "OFF") << endl;
  cout << "Video2: " << ((value & 0x02)? "ON" : "OFF") << endl;
  cout << "Video3: " << ((value & 0x04)? "ON" : "OFF") << endl;
  cout << "Video4: " << ((value & 0x08)? "ON" : "OFF") << endl;
}
else
{
  cout << "Error(" << err << ") in function Power_getVideoPowerStatus: " <<
    GetErrorStringA(err) << endl;
}
```

5.1.3.191 EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV CrossControl::Power_release (POWERHANDLE)

Delete the Power object.

Supported Platform(s): XM, XA, XS

Returns

-

Example Usage:

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

power_example(pPower);

Power_release(pPower);
```

5.1.3.192 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Power_setBLPowerStatus (POWERHANDLE , CCStatus *status*)

Set backlight power status.

Supported Platform(s): XM

Parameters

status	Backlight power status.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

```
cout << "Blinking backlight... " << endl;
cin.sync();
cout << endl << "Press Enter to to turn off the Backlight and then Enter to turn it on again..." << endl;
cin.get();
err = Power_setBLPowerStatus(pPower, Disabled);
cin.sync();
cin.get();
err = Power_setBLPowerStatus(pPower, Enabled);
if(err != ERR_SUCCESS)
{
  cout << "Error(" << err << ") in function Power_setBLPowerStatus: " <<
    GetErrorStringA(err) << endl;
}
```

5.1.3.193 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Power_setCanPowerStatus (POWERHANDLE , CCStatus *status*)

Set can power status.

Supported Platform(s): XM, XA, XS

Parameters

status	Can power status.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.194 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Power_setExtFanPowerStatus (POWERHANDLE , CCStatus *status*)

Set external fan power status.

Supported Platform(s): XM

Parameters

status	Fan power status.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.195 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Power_setVideoPowerStatus (POWERHANDLE , unsigned char *status*)

Set Video power status.

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

5.1.3.196 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::PowerMgr_getConfiguration (POWERMGRHANDLE , PowerMgrConf ∗ *conf*)

Get the configuration that is in use.

Supported Platform(s): XM, XA, XS

Parameters

conf	The configuration in use.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

```
    CrossControl::PowerMgrConf conf;
    err = PowerMgr_getConfiguration(pPowerMgr, &conf);
    if(err == ERR_SUCCESS)
    {
      switch (conf)
      {
      case Normal:
        cout << "PowerMgrConf is now: Normal" << endl; break;
      case ApplicationControlled:
        cout << "PowerMgrConf is now: ApplicationControlled" << endl; break;
      case BatterySuspend:
        cout << "PowerMgrConf is now: BatterySuspend" << endl; break;
      }
    }
    else
    {
      cout << "Error(" << err << ") in function getConfiguration: " <<
  GetErrorStringA(err) << endl;
    }
```

5.1.3.197 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::PowerMgr_getPowerMgrStatus (POWERMGRHANDLE , PowerMgrStatus ∗ *status*)

Get the current status of the PowerMgr. This functions should be called periodically, to detect when suspend or shutdown requests arrive.

Supported Platform(s): XM, XA, XS

Parameters

status	The current status.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

```
while(1)
{
  OSSleep(500);

  PowerMgrStatus status;
  err = PowerMgr_getPowerMgrStatus(pPowerMgr, &status);
  if(err == ERR_SUCCESS)
  {
    switch(status)
    {
    case NoRequestsPending: // Wait until a PowerMgr request arrives...
      break;

    case ShutdownPending:
      {
        // Shutdown by means of power button or on/off signal are caught here.
        os_shutdown = false;

        cout << "A shutdown request detected. App should now do what it needs to do before shutdown can
     be performed."  << endl;
        cout << "Press Enter when ready to shutdown... "  << endl;

        // Make sure to clear cin buffer before read
        std::cin.clear();
        std::cin.ignore(100,'\n');
        cin.get();
        cout << "Signalling that app is ready..."  << endl;
        err = PowerMgr_setAppReadyForSuspendOrShutdown(pPowerMgr)
    ;
        if(err != ERR_SUCCESS)
        {
          cout << "Error(" << err << ") in function setAppReadyForSuspendOrShutdown: " <<
    GetErrorStringA(err) << endl;
        }
        return; //exit test appp
      }
    case SuspendPending:
      {
        os_shutdown = false;

        cout << "A suspend request detected. App should now do what it needs to do before suspend can be
     performed."  << endl;
        cout << "Press Enter when ready to suspend... "  << endl;

        // Make sure to clear cin buffer before read
        std::cin.clear();
        std::cin.ignore(100,'\n');
        cin.get();
        cout << "Signalling that app is ready..."  << endl;
        err = PowerMgr_setAppReadyForSuspendOrShutdown(pPowerMgr)
    ;
        if(err != ERR_SUCCESS)
        {
          cout << "Error(" << err << ") in function setAppReadyForSuspendOrShutdown: " <<
    GetErrorStringA(err) << endl;
        }
      }
      break;

    default:
      cout << "Error: Invalid status returned from getPowerMgrStatus!"  << endl;
      break;
    }

    //Wait for resume after notifying that we are ready to suspend
    if(status == SuspendPending)
    {
      bool b = false;
      while(!b)
      {
        OSSleep(100);
        cout << "." << endl;

        err = PowerMgr_hasResumed(pPowerMgr, &b);
        if(err != ERR_SUCCESS)
        {
          cout << "Error(" << err << ") in function hasResumed: " <<
    GetErrorStringA(err) << endl;
        }
      }
      cout << "System is now resumed from suspend mode!"  << endl <<
        "Now we will soon re-register using the registerControlledSuspendOrShutDown function!"  << endl;

      // Expecting to get configuration Normal after resume from suspend

      CrossControl::PowerMgrConf conf;
      err = PowerMgr_getConfiguration(pPowerMgr, &conf);
      if(err == ERR_SUCCESS)
      {
        switch (conf)
        {
        case Normal:
          cout << "PowerMgrConf is now: Normal" << endl; break;
        case ApplicationControlled:
          cout << "PowerMgrConf is now: ApplicationControlled" << endl; break;
        case BatterySuspend:
          cout << "PowerMgrConf is now: BatterySuspend" << endl; break;
        }
      }
      else
      {
        cout << "Error(" << err << ") in function getConfiguration: " <<
    GetErrorStringA(err) << endl;
      }


      // Re-register, do this as soon as possible after resume/startup
      PowerMgr_registerControlledSuspendOrShutDown(pPowerMgr,
     setConfiguration);
      if(err == ERR_SUCCESS)
        cout << "Re-registered to powerMgr.  Ctrl-C to exit." << endl;
      else
        cout << "Error(" << err << ") in function registerControlledSuspendOrShutDown: " <<
    GetErrorStringA(err) << endl;
    }
  }
  else
  {
    cout << "Error(" << err << ") in function getPowerMgrStatus: " <<
    GetErrorStringA(err) << endl;
  }
}
```

5.1.3.198 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::PowerMgr_hasResumed (POWERMGRHANDLE , bool ∗ *resumed*)

This function can be used in a suspend-resume scenario. After the application has used setAppReadyForSuspendOrShutdown() to init the suspend, this function may be polled in order to detect when the system is up and running again. Calling this function before calling setAppReadyForSuspendOrShutdown will return resumed = true.

Supported Platform(s): XM, XA, XS

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

```
while(1)
{
  OSSleep(500);

  PowerMgrStatus status;
  err = PowerMgr_getPowerMgrStatus(pPowerMgr, &status);
  if(err == ERR_SUCCESS)
  {
    switch(status)
    {
    case NoRequestsPending: // Wait until a PowerMgr request arrives...
      break;

    case ShutdownPending:
      {
        // Shutdown by means of power button or on/off signal are caught here.
        os_shutdown = false;

        cout << "A shutdown request detected. App should now do what it needs to do before shutdown can
     be performed."  << endl;
        cout << "Press Enter when ready to shutdown... "  << endl;

        // Make sure to clear cin buffer before read
        std::cin.clear();
        std::cin.ignore(100,'\n');
        cin.get();
        cout << "Signalling that app is ready..."  << endl;
        err = PowerMgr_setAppReadyForSuspendOrShutdown(pPowerMgr)
    ;
        if(err != ERR_SUCCESS)
        {
          cout << "Error(" << err << ") in function setAppReadyForSuspendOrShutdown: " <<
    GetErrorStringA(err) << endl;
        }
        return; //exit test appp
      }
    case SuspendPending:
      {
        os_shutdown = false;

        cout << "A suspend request detected. App should now do what it needs to do before suspend can be
     performed."  << endl;
        cout << "Press Enter when ready to suspend... "  << endl;

        // Make sure to clear cin buffer before read
        std::cin.clear();
        std::cin.ignore(100,'\n');
        cin.get();
        cout << "Signalling that app is ready..."  << endl;
        err = PowerMgr_setAppReadyForSuspendOrShutdown(pPowerMgr)
    ;
        if(err != ERR_SUCCESS)
        {
          cout << "Error(" << err << ") in function setAppReadyForSuspendOrShutdown: " <<
    GetErrorStringA(err) << endl;
        }
      }
      break;

    default:
      cout << "Error: Invalid status returned from getPowerMgrStatus!"  << endl;
      break;
    }

    //Wait for resume after notifying that we are ready to suspend
    if(status == SuspendPending)
    {
      bool b = false;
      while(!b)
      {
        OSSleep(100);
        cout << "." << endl;

        err = PowerMgr_hasResumed(pPowerMgr, &b);
```

```
          if(err != ERR_SUCCESS)
          {
            cout << "Error(" << err << ") in function hasResumed: " <<
      GetErrorStringA(err) << endl;
          }
        }
        cout << "System is now resumed from suspend mode!"  << endl <<
          "Now we will soon re-register using the registerControlledSuspendOrShutDown function!"  << endl;

        // Expecting to get configuration Normal after resume from suspend

        CrossControl::PowerMgrConf conf;
        err = PowerMgr_getConfiguration(pPowerMgr, &conf);
        if(err == ERR_SUCCESS)
        {
          switch (conf)
          {
          case Normal:
            cout << "PowerMgrConf is now: Normal" << endl; break;
          case ApplicationControlled:
            cout << "PowerMgrConf is now: ApplicationControlled" << endl; break;
          case BatterySuspend:
            cout << "PowerMgrConf is now: BatterySuspend" << endl; break;
          }
        }
        else
        {
          cout << "Error(" << err << ") in function getConfiguration: " <<
      GetErrorStringA(err) << endl;
        }


        // Re-register, do this as soon as possible after resume/startup
        PowerMgr_registerControlledSuspendOrShutDown(pPowerMgr,
       setConfiguration);
        if(err == ERR_SUCCESS)
          cout << "Re-registered to powerMgr.  Ctrl-C to exit." << endl;
        else
          cout << "Error(" << err << ") in function registerControlledSuspendOrShutDown: " <<
      GetErrorStringA(err) << endl;
      }
    }
    else
    {
      cout << "Error(" << err << ") in function getPowerMgrStatus: " <<
      GetErrorStringA(err) << endl;
    }
  }
```

5.1.3.199 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::PowerMgr_registerControlledSuspendOrShutDown (POWERMGRHANDLE , PowerMgrConf *conf*)

Configure the PowerMgr. Call this function once initially to turn on the functionality.

Supported Platform(s): XM, XA, XS

Parameters

conf	The configuration to use.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

```
CrossControl::eErr err;
POWERMGRHANDLE pPowerMgr = ::GetPowerMgr();
BATTERYHANDLE pBattery = ::GetBattery();

assert(pPowerMgr);
assert(pBattery);

// Register a separate exit handler for the case where OS is initiating the shutdown. The Application
     must handle this case itself.
atexit(fnExit);

bool bBatt = false;
Battery_isBatteryPresent(pBattery, &bBatt);
if(bBatt)   // Ask user wich configuration to use...
  cout << "Choose configuration to use, 0 - Normal, 1 - Application Controlled , 2 - Battery Suspend"  <<
     endl;
else
  cout << "Choose configuration to use, 0 - Normal, 1 - Application Controlled"  << endl;

cin >> suspendConfiguration;
Battery_release(pBattery);

// Register that this application needs to delay suspend/shutdown
// This should be done as soon as possible.
// Then the app must poll getPowerMgrStatus() and allow the suspend/shutdown with
     setAppReadyForSuspendOrShutdown().
// Depending on application design, this might be best handled in a separate thread.
err = PowerMgr_registerControlledSuspendOrShutDown(pPowerMgr,
     (PowerMgrConf) suspendConfiguration);

cout << "suspendConfiguration " << suspendConfiguration << endl;

if(err == ERR_SUCCESS)
  cout << "Registered to powerMgr." << endl;
else
  cout << "Error(" << err << ") in function registerControlledSuspendOrShutDown: " <<
    GetErrorStringA(err) << endl;

test_powermgr(pPowerMgr);

PowerMgr_release(pPowerMgr);
```

5.1.3.200 EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV CrossControl::PowerMgr_release (POWERMGRHANDLE)

Delete the PowerMgr object.

Supported Platform(s): XM, XA, XS

Returns

-

Example Usage:

```
CrossControl::eErr err;
POWERMGRHANDLE pPowerMgr = ::GetPowerMgr();
BATTERYHANDLE pBattery = ::GetBattery();

assert(pPowerMgr);
assert(pBattery);
```

```
// Register a separate exit handler for the case where OS is initiating the shutdown. The Application
     must handle this case itself.
atexit(fnExit);

bool bBatt = false;
Battery_isBatteryPresent(pBattery, &bBatt);
if(bBatt)   // Ask user wich configuration to use...
  cout << "Choose configuration to use, 0 - Normal, 1 - Application Controlled , 2 - Battery Suspend"  <<
     endl;
else
  cout << "Choose configuration to use, 0 - Normal, 1 - Application Controlled"  << endl;

cin >> suspendConfiguration;
Battery_release(pBattery);

// Register that this application needs to delay suspend/shutdown
// This should be done as soon as possible.
// Then the app must poll getPowerMgrStatus() and allow the suspend/shutdown with
     setAppReadyForSuspendOrShutdown().
// Depending on application design, this might be best handled in a separate thread.
err = PowerMgr_registerControlledSuspendOrShutDown(pPowerMgr,
     (PowerMgrConf) suspendConfiguration);

cout << "suspendConfiguration " << suspendConfiguration << endl;

if(err == ERR_SUCCESS)
  cout << "Registered to powerMgr." << endl;
else
  cout << "Error(" << err << ") in function registerControlledSuspendOrShutDown: " <<
    GetErrorStringA(err) << endl;

test_powermgr(pPowerMgr);

PowerMgr_release(pPowerMgr);
```

5.1.3.201 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::PowerMgr_setAppReadyForSuspendOrShutdown (POWERMGRHANDLE)

Acknowledge that the application is ready for suspend/shutdown. Should be called after a request has been received in order to execute the request. The application must acknowledge a request within 20s from when it arrives.

Supported Platform(s): XM, XA, XS

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

```
while(1)
{
  OSSleep(500);

  PowerMgrStatus status;
  err = PowerMgr_getPowerMgrStatus(pPowerMgr, &status);
  if(err == ERR_SUCCESS)
  {
    switch(status)
    {
```

```
case NoRequestsPending: // Wait until a PowerMgr request arrives...
  break;

case ShutdownPending:
  {
    // Shutdown by means of power button or on/off signal are caught here.
    os_shutdown = false;

    cout << "A shutdown request detected. App should now do what it needs to do before shutdown can
 be performed."  << endl;
    cout << "Press Enter when ready to shutdown... "  << endl;

    // Make sure to clear cin buffer before read
    std::cin.clear();
    std::cin.ignore(100,'\n');
    cin.get();
    cout << "Signalling that app is ready..."  << endl;
    err = PowerMgr_setAppReadyForSuspendOrShutdown(pPowerMgr)
;
    if(err != ERR_SUCCESS)
    {
      cout << "Error(" << err << ") in function setAppReadyForSuspendOrShutdown: " <<
GetErrorStringA(err) << endl;
    }
    return; //exit test appp
  }
case SuspendPending:
  {
    os_shutdown = false;

    cout << "A suspend request detected. App should now do what it needs to do before suspend can be
 performed."  << endl;
    cout << "Press Enter when ready to suspend... "  << endl;

    // Make sure to clear cin buffer before read
    std::cin.clear();
    std::cin.ignore(100,'\n');
    cin.get();
    cout << "Signalling that app is ready..."  << endl;
    err = PowerMgr_setAppReadyForSuspendOrShutdown(pPowerMgr)
;
    if(err != ERR_SUCCESS)
    {
      cout << "Error(" << err << ") in function setAppReadyForSuspendOrShutdown: " <<
GetErrorStringA(err) << endl;
    }
  }
  break;

default:
  cout << "Error: Invalid status returned from getPowerMgrStatus!"  << endl;
  break;
}

//Wait for resume after notifying that we are ready to suspend
if(status == SuspendPending)
{
  bool b = false;
  while(!b)
  {
    OSSleep(100);
    cout << "." << endl;

    err = PowerMgr_hasResumed(pPowerMgr, &b);
    if(err != ERR_SUCCESS)
    {
      cout << "Error(" << err << ") in function hasResumed: " <<
GetErrorStringA(err) << endl;
    }
  }
  cout << "System is now resumed from suspend mode!"  << endl <<
    "Now we will soon re-register using the registerControlledSuspendOrShutDown function!"  << endl;

  // Expecting to get configuration Normal after resume from suspend
```

```
        CrossControl::PowerMgrConf conf;
        err = PowerMgr_getConfiguration(pPowerMgr, &conf);
        if(err == ERR_SUCCESS)
        {
          switch (conf)
          {
          case Normal:
            cout << "PowerMgrConf is now: Normal" << endl; break;
          case ApplicationControlled:
            cout << "PowerMgrConf is now: ApplicationControlled" << endl; break;
          case BatterySuspend:
            cout << "PowerMgrConf is now: BatterySuspend" << endl; break;
          }
        }
        else
        {
          cout << "Error(" << err << ") in function getConfiguration: " <<
      GetErrorStringA(err) << endl;
        }


        // Re-register, do this as soon as possible after resume/startup
        PowerMgr_registerControlledSuspendOrShutDown(pPowerMgr,
       setConfiguration);
        if(err == ERR_SUCCESS)
          cout << "Re-registered to powerMgr.  Ctrl-C to exit." << endl;
        else
          cout << "Error(" << err << ") in function registerControlledSuspendOrShutDown: " <<
      GetErrorStringA(err) << endl;
      }
    }
    else
    {
      cout << "Error(" << err << ") in function getPowerMgrStatus: " <<
      GetErrorStringA(err) << endl;
    }
  }
```

5.1.3.202 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Smart_getDeviceSerial (SMARTHANDLE , char ∗ *buff*, int *len*)

Get serial number of the secondary storage device.

Supported Platform(s): XM, XA, XS

Parameters

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. At least an 21 bytes buffer size must be used since the serial number can be 20 bytes + trailing zero.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

```
  char serial[21];
```

```
err = Smart_getDeviceSerial (pSmart, serial, sizeof(serial));
if (ERR_SUCCESS == err)
{
  cout << "Device serial number: " << serial << endl;
}
else
{
  cout << "Error(" << err << ") in function getDeviceSerial: " <<
    GetErrorStringA(err) << endl;
}
```

5.1.3.203 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Smart_getInitialTime (SMARTHANDLE , time_t ∗ *time*)

Get the date/time when the SMART monitoring began for this storage device. This time is either when the card first was used or when the system software was updated to support S.M.A.R.T. monitoring for the first time. Logging of time is based on the local time of the computer at the time of logging and may therefore not always be accurate.

Supported Platform(s): XM, XA, XS

Parameters

time	A 32bit time_t value representing the number of seconds elapsed since 00:00 hours, Jan 1, 1970 UTC.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

```
time_t initialTime;
struct tm * timeinfo;
err = Smart_getInitialTime (pSmart, &initialTime);
if (ERR_SUCCESS == err)
{
  cout << "Device was initially timestamped on: ";
  timeinfo = localtime (&initialTime);
  cout << asctime(timeinfo) << endl;
}
else
{
  cout << "Error(" << err << ") in function getInitialTime: " <<
    GetErrorStringA(err) << endl;
}
```

5.1.3.204 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Smart_getRemainingLifeTime (SMARTHANDLE , unsigned char ∗ *lifetimepercent*)

Get remaining lifetime of the secondary storage device.

Supported Platform(s): XM, XA, XS

Parameters

lifetimepercent	The expected remaining lifetime (0..100%).

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

```
unsigned char life;
err = Smart_getRemainingLifeTime (pSmart, &life);
if (ERR_SUCCESS == err)
{
  cout << "Estimated remaining lifetime: " << (int)life << "%" << endl;
}
else
{
  cout << "Error(" << err << ") in function getRemainingLifeTime: " <<
    GetErrorStringA(err) << endl;
}
```

5.1.3.205 EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV CrossControl::Smart_release (SMARTHANDLE)

Delete the Smart object.

Supported Platform(s): XM, XA, XS

Returns

-

Example Usage:

```
SMARTHANDLE pSmart = ::GetSmart();
assert(pSmart);

show_card_data(pSmart);

Smart_release(pSmart);
```

5.1.3.206 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Telematics_getBTPowerStatus (TELEMATICSHANDLE , CCStatus ∗ *status*)

Get Bluetooth power status.

Supported Platform(s): XM, XA, XS

Parameters

status	Bluetooth power status.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

```
err = Telematics_getBTPowerStatus(pTelematics, &status);
if(err == ERR_SUCCESS)
{
  cout << "Bluetooth power is " << ((status == Enabled)? "ON" : "OFF") << endl;
}
else if(err == ERR_TELEMATICS_BT_NOT_AVAILABLE)
{
  cout << "getBLPowerStatus: Bluetooth is not available on this platform" << endl;
}
else
{
  cout << "Error(" << err << ") in function getBLPowerStatus: " <<
    GetErrorStringA(err) << endl;
}
```

5.1.3.207 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Telematics_getBTStartUpPowerStatus (TELEMATICSHANDLE , CCStatus ∗ *status*)

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

Supported Platform(s): XM, XA, XS

Parameters

status	Bluetooth power status.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

```
err = Telematics_getBTStartUpPowerStatus(pTelematics, &status);
if(err == ERR_SUCCESS)
{
  cout << "Bluetooth power is " << ((status == Enabled)? "Enabled" : "Disabled") << " at start-up"
     << endl;
}
else if(err == ERR_TELEMATICS_BT_NOT_AVAILABLE)
{
  cout << "getBTStartUpPowerStatus: Bluetooth is not available on this platform" << endl;
}
else
{
  cout << "Error(" << err << ") in function getBTStartUpPowerStatus: " <<
    GetErrorStringA(err) << endl;
}
```

5.1.3.208 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Telematics_getGPRSPowerStatus (TELEMATICSHANDLE , CCStatus ∗ *status*)

Get GPRS power status.

Supported Platform(s): XM, XA, XS

Parameters

status	GPRS power status.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

```
  err = Telematics_getGPRSPowerStatus(pTelematics, &status);
  if(err == ERR_SUCCESS)
  {
    cout << "GSM/GPRS power is " << ((status == Enabled)? "ON" : "OFF") << endl;
  }
  else if(err == ERR_TELEMATICS_GPRS_NOT_AVAILABLE)
  {
    cout << "getGPRSPowerStatus: GSM/GPRS is not available on this platform" << endl;
  }
  else
  {
    cout << "Error(" << err << ") in function getGPRSPowerStatus: " <<
      GetErrorStringA(err) << endl;
  }
```

5.1.3.209 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Telematics_getGPRSStartUpPowerStatus (TELEMATICSHANDLE , CCStatus ∗ *status*)

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

Supported Platform(s): XM, XA, XS

Parameters

status	GPRS power status.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

```
err = Telematics_getGPRSStartUpPowerStatus(pTelematics, &status);
if(err == ERR_SUCCESS)
{
  cout << "GSM/GPRS power is " << ((status == Enabled)? "Enabled" : "Disabled") << " at start-up"
    << endl;
}
else if(err == ERR_TELEMATICS_GPRS_NOT_AVAILABLE)
{
  cout << "getGPRSStartUpPowerStatus: GSM/GPRS is not available on this platform" << endl;
}
else
{
  cout << "Error(" << err << ") in function getGPRSStartUpPowerStatus: " <<
    GetErrorStringA(err) << endl;
}
```

5.1.3.210 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Telematics_getGPSAntennaStatus (TELEMATICSHANDLE , CCStatus ∗ *status*)

Get GPS antenna status. Antenna open/short detection. The status is set to disabled if no antenna is present or a short is detected.

Supported Platform(s): XM, XA, XS

Parameters

status	GPS antenna power status.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

```
err = Telematics_getGPSAntennaStatus(pTelematics, &status);
if(err == ERR_SUCCESS)
{
  cout << "GPS antenna status: " << ((status == Enabled)? "OK" : "ERROR: Open connection or
     short-circuit") << endl;
}
else if(err == ERR_TELEMATICS_GPS_NOT_AVAILABLE)
{
  cout << "getGPSAntennaStatus: GPS is not available on this platform" << endl;
}
else
{
  cout << "Error(" << err << ") in function getGPSAntennaStatus: " <<
    GetErrorStringA(err) << endl;
}
```

5.1.3.211 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Telematics_getGPSPowerStatus (TELEMATICSHANDLE , CCStatus ∗ *status*)

Get GPS power status. Note that it can take some time after calling setGPSPowerStatus before the status is reported correctly.

Supported Platform(s): XM, XA, XS

Parameters

status	GPS power status.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

```
err = Telematics_getGPSPowerStatus(pTelematics, &status);
if(err == ERR_SUCCESS)
{
  cout << "GPS power is " << ((status == Enabled)? "ON" : "OFF") << endl;
}
else if(err == ERR_TELEMATICS_GPS_NOT_AVAILABLE)
{
  cout << "getGPSPowerStatus: GPS is not available on this platform" << endl;
}
else
{
  cout << "Error(" << err << ") in function getGPSPowerStatus: " <<
    GetErrorStringA(err) << endl;
}
```

5.1.3.212 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Telematics_getGPSStartUpPowerStatus (TELEMATICSHANDLE , CCStatus ∗ *status*)

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

Supported Platform(s): XM, XA, XS

Parameters

status	GPS power status.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

```
err = Telematics_getGPSStartUpPowerStatus(pTelematics, &status);
if(err == ERR_SUCCESS)
{
  cout << "GPS power is " << ((status == Enabled)? "Enabled" : "Disabled") << " at start-up" <<
    endl;
}
else if(err == ERR_TELEMATICS_GPS_NOT_AVAILABLE)
{
  cout << "getGPSStartUpPowerStatus: GPS is not available on this platform" << endl;
}
else
{
  cout << "Error(" << err << ") in function getGPSStartUpPowerStatus: " <<
    GetErrorStringA(err) << endl;
}
```

5.1.3.213 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Telematics_getTelematicsAvailable (TELEMATICSHANDLE , CCStatus ∗ *status*)

Is a telematics add-on card installed?

Supported Platform(s): XM, XA, XS

Parameters

status	Enabled if a telematics add-on card is installed, otherwise Disabled.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

```
err = Telematics_getTelematicsAvailable(pTelematics, &status);
if(err == ERR_SUCCESS)
{
  cout << "Telematics add-on board: " << ((status == Enabled)? "available" : "not available") <<
    endl;
  if(status == Disabled)
    return;
}
else
{
  cout << "Error(" << err << ") in function getTelematicsAvailable: " <<
    GetErrorStringA(err) << endl;
  return;
}
```

5.1.3.214 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Telematics_getWLANPowerStatus (TELEMATICSHANDLE , CCStatus ∗ *status*)

Get WLAN power status.

Supported Platform(s): XM, XA, XS

Parameters

status	WLAN power status.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

```
err = Telematics_getWLANPowerStatus(pTelematics, &status);
if(err == ERR_SUCCESS)
{
  cout << "WLAN power is " << ((status == Enabled)? "ON" : "OFF") << endl;
}
else if(err == ERR_TELEMATICS_WLAN_NOT_AVAILABLE)
{
  cout << "getWLANPowerStatus: WLAN is not available on this platform" << endl;
}
else
{
  cout << "Error(" << err << ") in function getWLANPowerStatus: " <<
    GetErrorStringA(err) << endl;
}
```

5.1.3.215 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Telematics_getWLANStartUpPowerStatus (TELEMATICSHANDLE , CCStatus ∗ *status*)

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

Supported Platform(s): XM, XA, XS

Parameters

status	WLAN power status.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

```
err = Telematics_getWLANStartUpPowerStatus(pTelematics, &status);
if(err == ERR_SUCCESS)
{
  cout << "WLAN power is " << ((status == Enabled)? "Enabled" : "Disabled") << " at start-up" <<
    endl;
}
else if(err == ERR_TELEMATICS_WLAN_NOT_AVAILABLE)
{
  cout << "getWLANStartUpPowerStatus: WLAN is not available on this platform" << endl;
}
else
{
  cout << "Error(" << err << ") in function getWLANStartUpPowerStatus: " <<
    GetErrorStringA(err) << endl;
}
```

5.1.3.216 EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV CrossControl::Telematics_release (TELEMATICSHANDLE)

Delete the Telematics object.

Supported Platform(s): XM, XA, XS

Returns

-

Example Usage:

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

telematics_example(pTelematics);

Telematics_release(pTelematics);
```

5.1.3.217 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Telematics_setBTPowerStatus (TELEMATICSHANDLE , CCStatus *status*)

Set Bluetooth power status.

Supported Platform(s): XM, XA, XS

Parameters

status	Bluetooth power status.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.218 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Telematics_setBTStartUpPowerStatus (TELEMATICSHANDLE , CCStatus *status*)

Set Bluetooth power status at startup and at resume from suspended mode.

Supported Platform(s): XM, XA, XS

Parameters

status	Bluetooth power status.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.219 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Telematics_setGPRSPowerStatus (TELEMATICSHANDLE , CCStatus *status*)

Set GPRS modem power status.

Supported Platform(s): XM, XA, XS

Parameters

status	GPRS modem power status.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.220 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Telematics_setGPRSStartUpPowerStatus (TELEMATICSHANDLE , CCStatus *status*)

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

Supported Platform(s): XM, XA, XS

Parameters

status	GPRS power status.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.221 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Telematics_setGPSPowerStatus (TELEMATICSHANDLE , CCStatus *status*)

Set GPS power status.

Supported Platform(s): XM, XA, XS

Parameters

status	GPS power status.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.222 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Telematics_setGPSStartUpPowerStatus (TELEMATICSHANDLE , CCStatus *status*)

Set GPS power status at startup and at resume from suspended mode.

Supported Platform(s): XM, XA, XS

Parameters

status	GPS power status.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.223 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Telematics_setWLANPowerStatus (TELEMATICSHANDLE , CCStatus *status*)

Set WLAN power status.

Supported Platform(s): XM, XA, XS

Parameters

status	WLAN power status.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.224 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Telematics_setWLANStartUpPowerStatus (TELEMATICSHANDLE , CCStatus *status*)

Set WLAN power status at startup and at resume from suspended mode.

Supported Platform(s): XM, XA, XS

Parameters

status	WLAN power status.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.225 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::TouchScreen_getAdvancedSetting (TOUCHSCREENHANDLE , TSAdvancedSettingsParameter *param*, unsigned short ∗ *data*)

Get advanced touch screen settings. See the description of TSAdvancedSettingsParameter for a description of the parameters.

Supported Platform(s): XM, XA, XS

Parameters

param	The setting to get.
data	The current data for the setting.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

```
err = TouchScreen_getAdvancedSetting(pTouchScreen,
    TS_DEBOUNCE_TIME, &debouncetime);
if(err == ERR_SUCCESS)
{
  cout << "Touchscreen debounce time is set to: " << (int)debouncetime  << " ms" << endl;
}
else
{
  cout << "Error(" << err << ") in function getAdvancedSetting: " <<
    GetErrorStringA(err) << endl;
}
```

5.1.3.226 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::TouchScreen_getMode (TOUCHSCREENHANDLE , TouchScreenModeSettings ∗ *config*)

Get Touch Screen mode. Gets the current mode of the USB profile.

Supported Platform(s): XM, XA, XS

Parameters

config	The current mode.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

```
  err = TouchScreen_getMode(pTouchScreen, &ts_mode);
  if(err == ERR_SUCCESS)
  {
    switch(ts_mode)
    {
    case MOUSE_NEXT_BOOT: cout << "USB profile is set to Mouse profile (active next boot)" <
      < endl; break;
    case TOUCH_NEXT_BOOT: cout << "USB profile is set to Touch profile (active next boot)" <
      < endl; break;
    case MOUSE_NOW: cout << "USB profile is set to Mouse profile" << endl; break;
    case TOUCH_NOW: cout << "USB profile is set to Touch profile" << endl; break;
    default: cout << "Error: invalid setting returned from getMode" << endl; break;
    }
  }
  else if (err == ERR_NOT_SUPPORTED) {
                cout << "Function TouchScreen_getMode() is not supported on this platform";
  }
  else
  {
    cout << "Error(" << err << ") in function getMode: " << GetErrorStringA(err) << endl;
  }
```

5.1.3.227 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::TouchScreen_getMouseRightClickTime (TOUCHSCREENHANDLE , unsigned short ∗ *time*)

Get mouse right click time. Applies only to the mouse profile. Use the OS settings for the touch profile.

Supported Platform(s): XM, XA, XS

Parameters

time	The right click time, in milliseconds.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

Example Usage:

```
err = TouchScreen_getMouseRightClickTime(pTouchScreen, &rightclicktime)
    ;
if(err == ERR_SUCCESS)
{
  cout << "Right click time is set to: " << (int)rightclicktime << " ms" << endl;
}
else
{
  cout << "Error(" << err << ") in function getMouseRightClickTime: " <<
    GetErrorStringA(err) << endl;
}
```

5.1.3.228 EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV CrossControl::TouchScreen_release (TOUCHSCREENHANDLE)

Delete the TouchScreen object.

Supported Platform(s): XM, XA, XS

Returns

-

Example Usage:

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

touchscreen_example(pTouchScreen);

TouchScreen_release(pTouchScreen);
```

5.1.3.229 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::TouchScreen_setAdvancedSetting (TOUCHSCREENHANDLE , TSAdvancedSettingsParameter *param,* unsigned short *data*)

Set advanced touch screen settings. See the description of TSAdvancedSettingsParameter for a description of the parameters.

Supported Platform(s): XM, XA, XS

Parameters

param	The setting to set.
data	The data value to set.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.230 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::TouchScreen_setMode (TOUCHSCREENHANDLE , TouchScreenModeSettings *config*)

Set Touch Screen mode. Sets the mode of the USB profile.

Supported Platform(s): XM, XA, XS

Parameters

config	The mode to set.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.231 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::TouchScreen_setMouseRightClickTime (TOUCHSCREENHANDLE , unsigned short *time*)

Set mouse right click time. Applies only to the mouse profile. Use the OS settings for the touch profile.

Supported Platform(s): XM, XA, XS

Parameters

time	The right click time, in milliseconds.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.232 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::TouchScreenCalib_checkCalibrationPointFinished (TOUCHSCREENCALIBHANDLE , bool * *finished*, unsigned char *pointNr*)

Check if a calibration point is finished

Parameters

finished	Is current point finished?
pointNr	Calibration point number (1 to total number of points)

Returns

error status. 0 = ERR_SUCCESS, otherwise error code.

5.1.3.233 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::TouchScreenCalib_getConfigParam (TOUCHSCREENCALIBHANDLE , CalibrationConfigParam *param*, unsigned short * *value*)

Get calibration config parameters

Parameters

param	Config parameter
value	Parameter value

Returns

error status. 0 = ERR_SUCCESS, otherwise error code.

5.1.3.234 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::TouchScreenCalib_getMode (TOUCHSCREENCALIBHANDLE , CalibrationModeSettings * *mode*)

Get mode of front controller.

Parameters

mode	Current calibration mode

Returns

error status. 0 = ERR_SUCCESS, otherwise error code.

5.1.3.235 EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV CrossControl::TouchScreenCalib_release (TOUCHSCREENCALIBHANDLE)

Delete the TouchScreenCalib object.

Returns

-

5.1.3.236 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::TouchScreenCalib_setCalibrationPoint (TOUCHSCREENCALIBHANDLE , unsigned char *pointNr*)

Set calibration point

Parameters

pointNr	Calibartion point number (1 to total number of points)

Returns

error status. 0 = ERR_SUCCESS, otherwise error code.

5.1.3.237 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::TouchScreenCalib_setConfigParam (TOUCHSCREENCALIBHANDLE , CalibrationConfigParam *param*, unsigned short *value*)

Set calibration config parameters

Parameters

param	Config parameter
value	parameter value

Returns

error status. 0 = ERR_SUCCESS, otherwise error code.

5.1.3.238 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::TouchScreenCalib_setMode (TOUCHSCREENCALIBHANDLE , CalibrationModeSettings *mode*)

Set mode of front controller.

Parameters

mode	Selected calibration mode

Returns

error status. 0 = ERR_SUCCESS, otherwise error code.

5.1.3.239 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_activateSnapshot (VIDEOHANDLE , bool *activate*)

To be able to take snapshot the snapshot function has to be active. After activation it takes 120ms before first snapshot can be taken. The Snapshot function can be active all the time. If power consumption and heat is an issue, snapshot may be turned off.

Supported Platform(s): XM (Windows)

Parameters

activate	Set to true if the snapshot function shall be active.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.240 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_createBitmap (VIDEOHANDLE , char ** *bmpBuffer,* unsigned long * *bmpBufSize,* const char * *rawImgBuffer,* unsigned long *rawImgBufSize,* bool *bInterlaced,* bool *bNTSCFormat*)

Create a bitmap from a raw image buffer. The bmp buffer is allocated in the function and has to be deallocated by the application.

Supported Platform(s): XM (Windows)

Parameters

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. See the enum eErr for details.

5.1.3.241 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_freeBmpBuffer (VIDEOHANDLE , char ∗ *bmpBuffer*)

Free the memory allocated for BMP buffer.

Supported Platform(s): XM (Windows)

Parameters

bmpBuffer	The bmp buffer to free.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.242 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_getActiveChannel (VIDEOHANDLE , VideoChannel ∗ *channel*)

Get the current video channel.

Supported Platform(s): XM, XA, XS

Parameters

channel	Enum defining available channels.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.243 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_getColorKeys (VIDEOHANDLE , unsigned char * *rKey*, unsigned char * *gKey*, unsigned char * *bKey*)

Get color key values. Note that the system uses 18 bit colors, so the two least significant bits are not used.

Supported Platform(s): XM

Parameters

rKey	Red value.
gKey	Green value.
bKey	Blue value.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.244 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_getCropping (VIDEOHANDLE , unsigned char * *top*, unsigned char * *left*, unsigned char * *bottom*, unsigned char * *right*)

Get Crop parameters.

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

5.1.3.245 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_getDecoderReg (VIDEOHANDLE , unsigned char *decoderRegister,* unsigned char ∗ *registerValue*)

Get Video decoder bus register. Advanced function for direct access to the video decoder TVP5150AM1 registers.

Supported Platform(s): XM, XA, XS

Parameters

decoder-Register	Decoder Register Address.
register-Value	register value.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.246 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_getDeInterlaceMode (VIDEOHANDLE , DeInterlaceMode ∗ *mode*)

Get the deinterlace mode used when decoding the interlaced video stream.

Supported Platform(s): XM

Parameters

mode	The current mode. See enum DeInterlaceMode for descriptions of the modes.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.247 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_getMirroring (VIDEOHANDLE , CCStatus ∗ *mode*)

Get the current mirroring mode of the video image.

Supported Platform(s): XM, XA, XS

Parameters

mode	The current mode. Enabled or Disabled.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.248 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_getRawImage (VIDEOHANDLE , unsigned short ∗ *width,* unsigned short ∗ *height,* float ∗ *frameRate*)

Get the raw image size of moving image before any scaling and frame rate. For snapshot the height is 4 row less.

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

5.1.3.249 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_getRotation (VIDEOHANDLE , VideoRotation ∗ *rotation*)

Get the current rotation of the video image.

Supported Platform(s): XA, XS

Parameters

rotation	Enum defining the current rotation.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.250 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_getScaling (VIDEOHANDLE , float ∗ *x*, float ∗ *y*)

Get Video Scaling (image size). If the deinterlace mode is set to DeInterlace_Even or DeInterlace_Odd, this function divides the actual vertical scaling by a factor of two, to get the same scaling factor as set with setScaling.

Supported Platform(s): XM

Parameters

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. See the enum eErr for details.

5.1.3.251 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_getStatus (VIDEOHANDLE , unsigned char ∗ *status*)

Video status byte.

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

5.1.3.252 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_getVideoArea (VIDEOHANDLE , unsigned short ∗ *topLeftX*, unsigned short ∗ *topLeftY*, unsigned short ∗ *bottomRigthX*, unsigned short ∗ *bottomRigthY*)

Get the area where video is shown.

Supported Platform(s): XM, XA, XS

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. See the enum eErr for details.

5.1.3.253 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_getVideoStandard (VIDEOHANDLE , videoStandard ∗ *standard*)

Get video standard. The video decoder auto detects the video standard of the source.

Supported Platform(s): XM, XA, XS

Parameters

standard	Video standard.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.254 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_init (VIDEOHANDLE , unsigned char *deviceNr*)

Initialize a video device. The video device will initially use the following settings: DeInterlace_BOB and mirroring disabled.

Supported Platform(s): XM, XA, XS

Parameters

deviceNr	Device to connect to (1,2). Select one of 2 devices to connect to.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.255 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_minimize (VIDEOHANDLE)

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

Supported Platform(s): XM, XA, XS

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.256 EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV CrossControl::Video_release (VIDEOHANDLE)

Delete the Video object.

Supported Platform(s): XM, XA, XS

Returns

-

5.1.3.257 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_restore (VIDEOHANDLE)

Restores the video area to the size it was before a minimize() call. Don't use restore if minimize has not been used first.

Supported Platform(s): XM, XA, XS

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.258 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_setActiveChannel (VIDEOHANDLE , VideoChannel *channel*)

Sets the active video channel.

Supported Platform(s): XM, XA, XS

Parameters

channel	Enum defining available channels.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.259 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_setColorKeys (VIDEOHANDLE , unsigned char *rKey*, unsigned char *gKey*, unsigned char *bKey*)

Set color keys. Writes RGB color key values. Note that the system uses 18 bit colors, so the two least significant bits are not used.

Supported Platform(s): XM

Parameters

rKey	Red key value.
gKey	Green key value.
bKey	Blue key value.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.260 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_setCropping (VIDEOHANDLE , unsigned char *top*, unsigned char *left*, unsigned char *bottom*, unsigned char *right*)

Crop video image. Note that the video chip manual says the following about horisontal cropping: The number of pixels of active video must be an even number. The parameters top and bottom are internally converted to an even number. This is due to the input video being interlaced, a pair of odd/even lines are allways cropped together. On XA/XS platforms, cropping from top/bottom on device 2 (channels 3 and 4) is not supported.

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

5.1.3.261 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_setDecoderReg (VIDEOHANDLE , unsigned char *decoderRegister,* unsigned char *registerValue*)

Set Video decoder bus register. Advanced function for direct access to the video decoder TVP5150AM1 registers.

Supported Platform(s): XM, XA, XS

Parameters

decoderRegister	Decoder Register Address.
registerValue	register value.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.262 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_setDeInterlaceMode (VIDEOHANDLE , DeInterlaceMode *mode*)

Set the deinterlace mode used when decoding the interlaced video stream.

Supported Platform(s): XM

Parameters

mode	The mode to set. See enum DeInterlaceMode for descriptions of the modes.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.263 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_setMirroring (VIDEOHANDLE , CCStatus *mode*)

Enable or disable mirroring of the video image.

Supported Platform(s): XM, XA, XS

Parameters

mode	The mode to set. Enabled or Disabled.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.264 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_setRotation (VIDEOHANDLE , VideoRotation *rotation*)

Set the current rotation of the video image.

Supported Platform(s): XA, XS

Parameters

rotation	Enum defining the rotation to set.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.265 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_setScaling (VIDEOHANDLE , float *x*, float *y*)

Set Video Scaling (image size). If the deinterlace mode is set to DeInterlace_Even or DeInterlace_Odd, this function multiplies the vertical scaling by a factor of two, to get the correct image proportions.

Supported Platform(s): XM

Parameters

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. See the enum eErr for details.

5.1.3.266 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_setVideoArea (VIDEOHANDLE , unsigned short *topLeftX*, unsigned short *topLeftY*, unsigned short *bottomRightX*, unsigned short *bottomRightY*)

Set the area where video is shown.

Supported Platform(s): XM, XA, XS

Parameters

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. See the enum eErr for details.

5.1.3.267 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_showFrame (VIDEOHANDLE)

Copy one frame from camera to the display.

Supported Platform(s): XA, XS

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.268 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_showVideo (VIDEOHANDLE , bool *show*)

Show or hide the video image. Note that it may take some time before the video is shown and correct input info can be read by getRawImage.

Supported Platform(s): XM, XA, XS

Parameters

show	True shows the video image.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.269 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_takeSnapshot (VIDEOHANDLE , const char * *path*, bool *bInterlaced*)

Takes a snapshot of the current video image and stores it to a bitmap file. This is a combination of takeSnapShotRaw, getVideoStandard and createBitMap and then storing of the bmpBuffer to file. To be able to take a snapshot, the snapshot function has to be active.

Supported Platform(s): XM (Windows)

Parameters

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. See the enum eErr for details.

5.1.3.270 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_takeSnapshotBmp (VIDEOHANDLE , char ** *bmpBuffer*, unsigned long * *bmpBufSize*, bool *bInterlaced*, bool *bNTSCFormat*)

Takes a snapshot of the current video image and return a data buffer with a bitmap image. The bmp buffer is allocated in the function and has to be deallocated with freeBmpBuffer() by the application. This is a combination of the function takeSnapShotRaw and createBitMap. To be able to take a snapshot, the snapshot function has to be active.

Supported Platform(s): XM (Windows)

Parameters

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.
bNTSC-Format	True if the video format in rawImageBuffer is NTSC format.

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.

5.1.3.271 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Video_takeSnapshotRaw (VIDEOHANDLE , char ∗ *rawImgBuffer,* unsigned long *rawImgBuffSize,* bool *bInterlaced*)

Takes a snapshot of the current video image and return raw image data. The size of the raw image is when interlaced = false 0x100 + line count ∗ row count ∗ 4. The size of the raw image is when interlaced = true 0x100 + line count ∗ row count ∗ 2. To be able to take a snapshot, the snapshot function has to be active. This function is blocking until a new frame is available from the decoder. An error will be returned if the decoder doesn't return any frames before a timeout.

Supported Platform(s): XM (Windows)

Parameters

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. See the enum eErr for details.

5.1.4 Variable Documentation

5.1.4.1 const unsigned char DigitalIn_1 = (1 << 0)

Bit defines for getDigIO

5.1.4.2 const unsigned char DigitalIn_2 = (1 << 1)

5.1.4.3 const unsigned char DigitalIn_3 = (1 << 2)

5.1.4.4 const unsigned char DigitalIn_4 = (1 << 3)

5.1.4.5 const unsigned char Video1Conf = (1 << 0)

Bit defines for getVideoStartupPowerConfig and setVideoStartupPowerConfig

5.1.4.6 const unsigned char Video2Conf = (1 $<<$ 1)

Video channel 1 config

5.1.4.7 const unsigned char Video3Conf = (1 $<<$ 2)

Video channel 2 config

5.1.4.8 const unsigned char Video4Conf = (1 $<<$ 3)

Video channel 3 config

Chapter 6

Data Structure Documentation

6.1 BatteryTimerType Struct Reference

```
#include <Battery.h>
```

Data Fields

- unsigned long TotRunTimeMain
- unsigned long TotRunTimeBattery
- unsigned long RunTime_m20
- unsigned long RunTime_m20_0
- unsigned long RunTime_0_40
- unsigned long RunTime_40_60
- unsigned long RunTime_60_70
- unsigned long RunTime_70_80
- unsigned long RunTime_Above80

6.1.1 Field Documentation

6.1.1.1 unsigned long RunTime_0_40

Total runtime in range 0 to -20 deg C (minutes)

6.1.1.2 unsigned long RunTime_40_60

Total runtime in range 0 to 40 deg C (minutes)

6.1.1.3 unsigned long RunTime_60_70

Total runtime in range 40 to 60 deg C (minutes)

6.1.1.4 unsigned long RunTime_70_80

Total runtime in range 60 to 70 deg C (minutes)

6.1.1.5 unsigned long RunTime_Above80

Total runtime in range 70 to 80 deg C (minutes)

6.1.1.6 unsigned long RunTime_m20

Total running time on battery power (minutes)

6.1.1.7 unsigned long RunTime_m20_0

Total runtime below -20 deg C (minutes)

6.1.1.8 unsigned long TotRunTimeBattery

Total running time on main power (minutes)

6.1.1.9 unsigned long TotRunTimeMain

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

- IncludeFiles/Battery.h

6.2 BuzzerSetup Struct Reference

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

Data Fields

- unsigned short frequency
- unsigned short volume

6.2.1 Field Documentation

6.2.1.1 unsigned short frequency

buzzer frequency

6.2.1.2 unsigned short volume

buzzer volume

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

- IncludeFiles/CCAuxTypes.h

6.3 FpgaLedTimingType Struct Reference

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

Data Fields

- unsigned char ledNbr
- unsigned char onTime
- unsigned char offTime
- unsigned char idleTime
- unsigned char nrOfPulses

6.3.1 Field Documentation

6.3.1.1 unsigned char idleTime

LED idle time in 100ms

6.3.1.2 unsigned char ledNbr

Number of LED

6.3.1.3 unsigned char nrOfPulses

Pulses per sequences

6.3.1.4 unsigned char offTime

LED off time in 10ms

6.3.1.5 unsigned char onTime

LED on time in 10ms

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

- IncludeFiles/CCAuxTypes.h

6.4 LedColorMixType Struct Reference

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

Data Fields

- unsigned char red
- unsigned char green
- unsigned char blue

6.4.1 Field Documentation

6.4.1.1 unsigned char blue

Blue color intensity 0-0x0F

6.4.1.2 unsigned char green

Green color intensity 0-0x0F

6.4.1.3 unsigned char red

Red color intensity 0-0x0F

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

- IncludeFiles/CCAuxTypes.h

6.5 LedTimingType Struct Reference

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

Data Fields

- unsigned char onTime
- unsigned char offTime
- unsigned char idleTime
- unsigned char nrOfPulses

6.5.1 Field Documentation

6.5.1.1 unsigned char idleTime

LED idle time in 100ms

6.5.1.2 unsigned char nrOfPulses

Pulses per sequences

6.5.1.3 unsigned char offTime

LED off time in 10ms

6.5.1.4 unsigned char onTime

LED on time in 10ms

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

- IncludeFiles/CCAuxTypes.h

6.6 received_video Struct Reference

`#include <CCAuxTypes.h>`

Data Fields

- unsigned short received_width
- unsigned short received_height
- unsigned char received_framerate

6.6.1 Field Documentation

6.6.1.1 unsigned char received_framerate

6.6.1.2 unsigned short received_height

6.6.1.3 unsigned short received_width

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

- IncludeFiles/CCAuxTypes.h

6.7 TimerType Struct Reference

`#include <CCAuxTypes.h>`

Data Fields

- unsigned long TotRunTime
- unsigned long TotSuspTime
- unsigned long TotHeatTime
- unsigned long RunTime40_60
- unsigned long RunTime60_70
- unsigned long RunTime70_80
- unsigned long Above80RunTime

6.7.1 Detailed Description

Diagnostic timer data

6.7.2 Field Documentation

6.7.2.1 unsigned long Above80RunTime

Total runtime in 70-80deg (minutes)

6.7.2.2 unsigned long RunTime40_60

Total heating time (minutes)

6.7.2.3 unsigned long RunTime60_70

Total runtime in 40-60deg (minutes)

6.7.2.4 unsigned long RunTime70_80

Total runtime in 60-70deg (minutes)

6.7.2.5 unsigned long TotHeatTime

Total suspend time (minutes)

6.7.2.6 unsigned long TotRunTime

6.7.2.7 unsigned long TotSuspTime

Total running time (minutes)

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

- IncludeFiles/CCAuxTypes.h

6.8 UpgradeStatus Struct Reference

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

Data Fields

- enum UpgradeAction currentAction
- unsigned char percent
- eErr errorCode

6.8.1 Detailed Description

Upgrade Status

6.8.2 Field Documentation

6.8.2.1 enum UpgradeAction currentAction

6.8.2.2 eErr errorCode

Represents the percentage of completion of the current action

6.8.2.3 unsigned char percent

The current action.

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

- IncludeFiles/CCAuxTypes.h

6.9 version_info Struct Reference

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

Data Fields

- unsigned char major
- unsigned char minor
- unsigned char release
- unsigned char build

6.9.1 Field Documentation

6.9.1.1 unsigned char build

version build number

6.9.1.2 unsigned char major

version major number

6.9.1.3 unsigned char minor

version minor number

6.9.1.4 unsigned char release

version release number

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

- IncludeFiles/CCAuxTypes.h

6.10 video_dec_command Struct Reference

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

Data Fields

- unsigned char decoder_register
- unsigned char register_value

6.10.1 Field Documentation

6.10.1.1 unsigned char decoder_register

6.10.1.2 unsigned char register_value

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

- IncludeFiles/CCAuxTypes.h

Chapter 7

File Documentation

7.1 IncludeFiles/About.h File Reference

Namespaces

- namespace CrossControl

Typedefs

- typedef void ∗ ABOUTHANDLE

Functions

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

- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV About_getMainHWversion (ABOUTHAND-LE, 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 (ABOUTHAND-LE, char ∗buff, int len)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV About_getNrOfETHConnections (ABOUTH-ANDLE, unsigned char ∗NrOfConnections)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV About_getNrOfCANConnections (ABOUT-HANDLE, unsigned char ∗NrOfConnections)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV About_getNrOfVideoConnections (ABOUT-HANDLE, unsigned char ∗NrOfConnections)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV About_getNrOfUSBConnections (ABOUTH-ANDLE, unsigned char ∗NrOfConnections)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV About_getNrOfSerialConnections (ABOUT-HANDLE, unsigned char ∗NrOfConnections)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV About_getNrOfDigIOConnections (ABOUT-HANDLE, unsigned char ∗NrOfConnections)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV About_getIsDisplayAvailable (ABOUTHAN-DLE, bool ∗available)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV About_getIsTouchScreenAvailable (ABOUT-HANDLE, bool ∗available)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV About_getDisplayResolution (ABOUTHAN-DLE, char ∗buff, int len)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV About_getAddOnPCBSerial (ABOUTHAN-DLE, char ∗buff, int len)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV About_getAddOnPCBArt (ABOUTHANDL-E, char ∗buff, int length)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV About_getAddOnManufacturingDate (ABO-UTHANDLE, char ∗buff, int len)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV About_getAddOnHWversion (ABOUTHAN-DLE, char ∗buff, int len)

- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV About_getIsWLANMounted (ABOUTHAN-DLE, bool ∗mounted)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV About_getIsGPSMounted (ABOUTHANDL-E, bool ∗mounted)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV About_getIsGPRSMounted (ABOUTHAND-LE, 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 (ABOUT-HANDLE, bool ∗mounted)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV About_getIOExpanderValue (ABOUTHAN-DLE, 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 (ABOUTHAN-DLE, bool ∗mounted)

7.2 IncludeFiles/Adc.h File Reference

Namespaces

- namespace CrossControl

Typedefs

- typedef void ∗ ADCHANDLE

Functions

- EXTERN_C CCAUXDLL_API
ADCHANDLE
CCAUXDLL_CALLING_CONV GetAdc (void)
- EXTERN_C CCAUXDLL_API void
CCAUXDLL_CALLING_CONV Adc_release (ADCHANDLE)

- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Adc_getVoltage (ADCHANDLE, VoltageEnum selection, double ∗value)

7.3 IncludeFiles/AuxVersion.h File Reference

Namespaces

- namespace CrossControl

Typedefs

- typedef void ∗ AUXVERSIONHANDLE

Functions

- EXTERN_C CCAUXDLL_API
  AUXVERSIONHANDLE
  CCAUXDLL_CALLING_CONV GetAuxVersion (void)
- EXTERN_C CCAUXDLL_API void
  CCAUXDLL_CALLING_CONV AuxVersion_release (AUXVERSIONHAND-LE)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV AuxVersion_getFPGAVersion (AUXVERSI-ONHANDLE, unsigned char ∗major, unsigned char ∗minor, unsigned char ∗release, unsigned char ∗build)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV AuxVersion_getSSVersion (AUXVERSION-HANDLE, unsigned char ∗major, unsigned char ∗minor, unsigned char ∗release, unsigned char ∗build)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV AuxVersion_getFrontVersion (AUXVERSIO-NHANDLE, unsigned char ∗major, unsigned char ∗minor, unsigned char ∗release, unsigned char ∗build)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV AuxVersion_getCCAuxVersion (AUXVERS-IONHANDLE, unsigned char ∗major, unsigned char ∗minor, unsigned char ∗release, unsigned char ∗build)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV AuxVersion_getOSVersion (AUXVERSION-HANDLE, unsigned char ∗major, unsigned char ∗minor, unsigned char ∗release, unsigned char ∗build)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV AuxVersion_getCCAuxDrvVersion (AUXV-ERSIONHANDLE, unsigned char ∗major, unsigned char ∗minor, unsigned char ∗release, unsigned char ∗build)

7.4 IncludeFiles/Backlight.h File Reference

Namespaces

- namespace CrossControl

Typedefs

- typedef void ∗ BACKLIGHTHANDLE

Functions

- EXTERN_C CCAUXDLL_API BACKLIGHTHANDLE CCAUXDLL_CALLING_CONV GetBacklight (void)
- EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV Backlight_release (BACKLIGHTHANDLE)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Backlight_getIntensity (BACKLIGHTHANDLE, unsigned char ∗intensity)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Backlight_setIntensity (BACKLIGHTHANDLE, unsigned char intensity)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Backlight_getStatus (BACKLIGHTHANDLE, unsigned char ∗status)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Backlight_getHWStatus (BACKLIGHTHANDLE, bool ∗status)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Backlight_startAutomaticBL (BACKLIGHTHANDLE)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Backlight_stopAutomaticBL (BACKLIGHTHANDLE)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Backlight_getAutomaticBLStatus (BACKLIGHTHANDLE, unsigned char ∗status)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Backlight_setAutomaticBLParams (BACKLIGHTHANDLE, bool bSoftTransitions)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Backlight_getAutomaticBLParams (BACKLIGHTHANDLE, bool ∗bSoftTransitions, double ∗k)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Backlight_setAutomaticBLFilter (BACKLIGHTHANDLE, unsigned long averageWndSize, unsigned long rejectWndSize, unsigned long rejectDeltaInLux, LightSensorSamplingMode mode)

- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Backlight_getAutomaticBLFilter (BACKLIGHTHANDLE, unsigned long ∗averageWndSize, unsigned long ∗rejectWndSize, unsigned long ∗rejectDeltaInLux, LightSensorSamplingMode ∗mode)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Backlight_getLedDimming (BACKLIGHTHANDLE, CCStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Backlight_setLedDimming (BACKLIGHTHANDLE, CCStatus status)

7.5 IncludeFiles/Battery.h File Reference

Data Structures

- struct BatteryTimerType

Namespaces

- namespace CrossControl

Typedefs

- typedef void ∗ BATTERYHANDLE

Enumerations

- enum ChargingStatus {
  ChargingStatus_NoCharge = 0, ChargingStatus_Charging = 1, ChargingStatus_FullyCharged = 2, ChargingStatus_TempLow = 3,
  ChargingStatus_TempHigh = 4, ChargingStatus_Unknown = 5 }
- enum PowerSource { PowerSource_Battery = 0, PowerSource_ExternalPower = 1 }
- enum ErrorStatus {
  ErrorStatus_NoError = 0, ErrorStatus_ThermistorTempSensor = 1, ErrorStatus_SecondaryTempSensor = 2, ErrorStatus_ChargeFail = 3,
  ErrorStatus_Overcurrent = 4, ErrorStatus_Init = 5 }

Functions

- EXTERN_C CCAUXDLL_API
  BATTERYHANDLE
  CCAUXDLL_CALLING_CONV GetBattery (void)
- EXTERN_C CCAUXDLL_API void
  CCAUXDLL_CALLING_CONV Battery_release (BATTERYHANDLE)

- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Battery_isBatteryPresent (BATTERYHANDLE, bool ∗batteryIsPresent)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Battery_getBatteryVoltageStatus (BATTERYHANDLE, unsigned char ∗batteryVoltagePercent)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Battery_getBatteryChargingStatus (BATTERYHANDLE, ChargingStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Battery_getPowerSource (BATTERYHANDLE, PowerSource ∗status)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Battery_getBatteryTemp (BATTERYHANDLE, signed short ∗temperature)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Battery_getHwErrorStatus (BATTERYHANDLE, ErrorStatus ∗errorCode)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Battery_getTimer (BATTERYHANDLE, BatteryTimerType ∗times)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Battery_getMinMaxTemp (BATTERYHANDLE, signed short ∗minTemp, signed short ∗maxTemp)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Battery_getBatteryHWversion (BATTERYHANDLE, char ∗buff, int len)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Battery_getBatterySwVersion (BATTERYHANDLE, unsigned short ∗major, unsigned short ∗minor, unsigned short ∗release, unsigned short ∗build)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Battery_getBatterySerial (BATTERYHANDLE, char ∗buff, int len)

7.6 IncludeFiles/Buzzer.h File Reference

Namespaces

- namespace CrossControl

Typedefs

- typedef void ∗ BUZZERHANDLE

Functions

- EXTERN_C CCAUXDLL_API BUZZERHANDLE CCAUXDLL_CALLING_CONV GetBuzzer (void)
- EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV Buzzer_release (BUZZERHANDLE)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Buzzer_getFrequency (BUZZERHANDLE, unsigned short ∗frequency)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Buzzer_getVolume (BUZZERHANDLE, unsigned short ∗volume)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Buzzer_getTrigger (BUZZERHANDLE, bool ∗trigger)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Buzzer_setFrequency (BUZZERHANDLE, unsigned short frequency)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Buzzer_setVolume (BUZZERHANDLE, unsigned short volume)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Buzzer_setTrigger (BUZZERHANDLE, bool trigger)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Buzzer_buzze (BUZZERHANDLE, int time, bool blocking)

7.7 IncludeFiles/CanSetting.h File Reference

Namespaces

- namespace CrossControl

Typedefs

- typedef void ∗ CANSETTINGHANDLE

Functions

- EXTERN_C CCAUXDLL_API CANSETTINGHANDLE CCAUXDLL_CALLING_CONV GetCanSetting (void)

- EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV CanSetting_release (CANSETTINGHANDLE)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CanSetting_getBaudrate (CANSETTINGHANDLE, unsigned char net, unsigned short ∗baudrate)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CanSetting_getFrameType (CANSETTINGHANDLE, unsigned char net, CanFrameType ∗frameType)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CanSetting_setBaudrate (CANSETTINGHANDLE, unsigned char net, unsigned short baudrate)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CanSetting_setFrameType (CANSETTINGHANDLE, unsigned char net, CanFrameType frameType)

7.8 IncludeFiles/CCAuxErrors.h File Reference

Namespaces

- namespace CrossControl

Functions

- EXTERN_C CCAUXDLL_API char const ∗CCAUXDLL_CALLING_CONV GetErrorStringA (eErr errCode)
- EXTERN_C CCAUXDLL_API wchar_t const ∗CCAUXDLL_CALLING_CONV GetErrorStringW (eErr errCode)

7.9 IncludeFiles/CCAuxTypes.h File Reference

Data Structures

- struct received_video
- struct video_dec_command
- struct version_info
- struct BuzzerSetup
- struct LedTimingType
- struct FpgaLedTimingType
- struct LedColorMixType
- struct TimerType
- struct UpgradeStatus

Namespaces

- namespace CrossControl

Typedefs

- typedef struct version_info VersionType

Enumerations

- enum VoltageEnum {
  VOLTAGE_24VIN = 0, VOLTAGE_24V, VOLTAGE_12V, VOLTAGE_12-VID,
  VOLTAGE_5V, VOLTAGE_3V3, VOLTAGE_VTFT, VOLTAGE_5VSTB,
  VOLTAGE_1V9, VOLTAGE_1V8, VOLTAGE_1V5, VOLTAGE_1V2,
  VOLTAGE_1V05, VOLTAGE_1V0, VOLTAGE_0V9, VOLTAGE_VREF_I-NT,
  VOLTAGE_24V_BACKUP, VOLTAGE_2V5, VOLTAGE_1V1, VOLTAGE-_1V3_PER,
  VOLTAGE_1V3_VDDA }
- enum LightSensorOperationRange { RangeStandard = 0, RangeExtended = 1 }
- enum LightSensorSamplingMode { SamplingModeStandard = 0, SamplingMode-Extended, SamplingModeAuto }
- enum CCStatus { Disabled = 0, Enabled = 1 }
- enum eErr {
  ERR_SUCCESS = 0, ERR_OPEN_FAILED = 1, ERR_NOT_SUPPORTED = 2, ERR_UNKNOWN_FEATURE = 3,
  ERR_DATATYPE_MISMATCH = 4, ERR_CODE_NOT_EXIST = 5, ERR_-BUFFER_SIZE = 6, ERR_IOCTRL_FAILED = 7,
  ERR_INVALID_DATA = 8, ERR_INVALID_PARAMETER = 9, ERR_CRE-ATE_THREAD = 10, ERR_IN_PROGRESS = 11,
  ERR_CHECKSUM = 12, ERR_INIT_FAILED = 13, ERR_VERIFY_FAILED = 14, ERR_DEVICE_READ_DATA_FAILED = 15,
  ERR_DEVICE_WRITE_DATA_FAILED = 16, ERR_COMMAND_FAILED = 17, ERR_EEPROM = 18, ERR_JIDA_TEMP = 19,
  ERR_AVERAGE_CALC_STARTED = 20, ERR_NOT_RUNNING = 21, ER-R_I2C_EXPANDER_READ_FAILED = 22, ERR_I2C_EXPANDER_WRITE-_FAILED = 23,
  ERR_I2C_EXPANDER_INIT_FAILED = 24, ERR_NEWER_SS_VERSION-_REQUIRED = 25, ERR_NEWER_FPGA_VERSION_REQUIRED = 26, ER-R_NEWER_FRONT_VERSION_REQUIRED = 27,
  ERR_TELEMATICS_GPRS_NOT_AVAILABLE = 28, ERR_TELEMATICS-_WLAN_NOT_AVAILABLE = 29, ERR_TELEMATICS_BT_NOT_AVAIL-ABLE = 30, ERR_TELEMATICS_GPS_NOT_AVAILABLE = 31,
  ERR_MEM_ALLOC_FAIL = 32, ERR_JOIN_THREAD = 33 }
- enum DeInterlaceMode { DeInterlace_Even = 0, DeInterlace_Odd = 1, DeInterlace-_BOB = 2 }

- enum VideoChannel { Analog_Channel_1 = 0, Analog_Channel_2 = 1, Analog-_Channel_3 = 2, Analog_Channel_4 = 3 }
- enum videoStandard {
  STD_M_J_NTSC = 0, STD_B_D_G_H_I_N_PAL = 1, STD_M_PAL = 2, ST-D_PAL = 3,
  STD_NTSC = 4, STD_SECAM = 5 }
- enum VideoRotation { RotNone = 0, Rot90, Rot180, Rot270 }
- enum CanFrameType { FrameStandard, FrameExtended, FrameStandardExtended }
- enum TriggerConf { Front_Button_Enabled = 1, OnOff_Signal_Enabled = 2, Both_Button_And_Signal_Enabled = 3 }
- enum PowerAction { NoAction = 0, ActionSuspend = 1, ActionShutDown = 2 }
- enum ButtonPowerTransitionStatus {
  BPTS_No_Change = 0, BPTS_ShutDown = 1, BPTS_Suspend = 2, BPTS_-Restart = 3,
  BPTS_BtnPressed = 4, BPTS_BtnPressedLong = 5, BPTS_SignalOff = 6 }
- enum OCDStatus { OCD_OK = 0, OCD_OC = 1, OCD_POWER_OFF = 2 }
- enum JidaSensorType {
  TEMP_CPU = 0, TEMP_BOX = 1, TEMP_ENV = 2, TEMP_BOARD = 3,
  TEMP_BACKPLANE = 4, TEMP_CHIPSETS = 5, TEMP_VIDEO = 6, TEM-P_OTHER = 7 }
- enum UpgradeAction {
  UPGRADE_INIT, UPGRADE_PREP_COM, UPGRADE_READING_FILE, U-PGRADE_CONVERTING_FILE,
  UPGRADE_FLASHING, UPGRADE_VERIFYING, UPGRADE_COMPLET-E, UPGRADE_COMPLETE_WITH_ERRORS }
- enum CCAuxColor {
  RED = 0, GREEN, BLUE, CYAN,
  MAGENTA, YELLOW, UNDEFINED_COLOR }
- enum RS4XXPort { RS4XXPort1 = 1, RS4XXPort2, RS4XXPort3, RS4XX-Port4 }

7.10 IncludeFiles/CCPlatform.h File Reference

7.11 IncludeFiles/Config.h File Reference

Namespaces

- namespace CrossControl

Typedefs

- typedef void ∗ CONFIGHANDLE

Functions

- EXTERN_C CCAUXDLL_API CONFIGHANDLE CCAUXDLL_CALLING_CONV GetConfig ()
- EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV Config_release (CONFIGHANDLE)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_getStartupTriggerConfig (CONFIGHANDLE, TriggerConf ∗config)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_getShortButtonPressAction (CONFIGHANDLE, PowerAction ∗action)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_getLongButtonPressAction (CONFIGHANDLE, PowerAction ∗action)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_getOnOffSigAction (CONFIGHANDLE, PowerAction ∗action)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_getFrontBtnTrigTime (CONFIGHANDLE, unsigned short ∗triggertime)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_getExtOnOffSigTrigTime (CONFIGHANDLE, unsigned long ∗triggertime)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_getSuspendMaxTime (CONFIGHANDLE, unsigned short ∗maxTime)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_getCanStartupPowerConfig (CONFIGHANDLE, CCStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_getVideoStartupPowerConfig (CONFIGHANDLE, unsigned char ∗config)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_getExtFanStartupPowerConfig (CONFIGHANDLE, CCStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_getStartupVoltageConfig (CONFIGHANDLE, double ∗voltage)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_getHeatingTempLimit (CONFIGHANDLE, signed short ∗temperature)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_getPowerOnStartup (CONFIGHANDLE, CCStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_setStartupTriggerConfig (CONFIGHANDLE, TriggerConf conf)

- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_setShortButtonPressAction (CONFIGHANDLE, PowerAction action)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_setLongButtonPressAction (CONFIGHANDLE, PowerAction action)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_setOnOffSigAction (CONFIGHANDLE, PowerAction action)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_setFrontBtnTrigTime (CONFIGHANDLE, unsigned short triggertime)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_setExtOnOffSigTrigTime (CONFIGHANDLE, unsigned long triggertime)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_setSuspendMaxTime (CONFIGHANDLE, unsigned short maxTime)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_setCanStartupPowerConfig (CONFIGHANDLE, CCStatus status)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_setVideoStartupPowerConfig (CONFIGHANDLE, unsigned char config)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_setExtFanStartupPowerConfig (CONFIGHANDLE, CCStatus status)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_setStartupVoltageConfig (CONFIGHANDLE, double voltage)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_setHeatingTempLimit (CONFIGHANDLE, signed short temperature)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_setPowerOnStartup (CONFIGHANDLE, CCStatus status)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_setRS485Enabled (CONFIGHANDLE, RS4XXPort port, bool enabled)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Config_getRS485Enabled (CONFIGHANDLE, RS4XXPort port, bool *enabled)

Variables

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

7.12 IncludeFiles/Diagnostic.h File Reference

Namespaces

- namespace CrossControl

Typedefs

- typedef void ∗ DIAGNOSTICHANDLE

Functions

- EXTERN_C CCAUXDLL_API
DIAGNOSTICHANDLE
CCAUXDLL_CALLING_CONV GetDiagnostic (void)
- EXTERN_C CCAUXDLL_API void
CCAUXDLL_CALLING_CONV Diagnostic_release (DIAGNOSTICHANDL-
E)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Diagnostic_getSSTemp (DIAGNOSTICHA-
NDLE, signed short ∗temperature)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Diagnostic_getPCBTemp (DIAGNOSTICH-
ANDLE, signed short ∗temperature)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Diagnostic_getPMTemp (DIAGNOSTICHA-
NDLE, unsigned char index, signed short ∗temperature, JidaSensorType ∗jst)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Diagnostic_getStartupReason (DIAGNOST-
ICHANDLE, unsigned short ∗reason)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Diagnostic_getShutDownReason (DIAGNO-
STICHANDLE, unsigned short ∗reason)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Diagnostic_getHwErrorStatus (DIAGNOST-
ICHANDLE, unsigned short ∗errorCode)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Diagnostic_getTimer (DIAGNOSTICHAND-
LE, TimerType ∗times)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Diagnostic_getMinMaxTemp (DIAGNOSTI-
CHANDLE, signed short ∗minTemp, signed short ∗maxTemp)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Diagnostic_getPowerCycles (DIAGNOSTIC-
HANDLE, unsigned short ∗powerCycles)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Diagnostic_clearHwErrorStatus (DIAGNOS-
TICHANDLE)

7.13 IncludeFiles/DiagnosticCodes.h File Reference

Namespaces

- namespace CrossControl

Enumerations

- enum startupReasonCodes {
  startupReasonCodeUndefined = 0x0000, startupReasonCodeButtonPress = 0x0055,
  startupReasonCodeExtCtrl = 0x00AA, startupReasonCodeMPRestart = 0x00F0,
  startupReasonCodePowerOnStartup = 0x000F }
- enum shutDownReasonCodes { shutdownReasonCodeNoError = 0x001F }
- enum hwErrorStatusCodes { errCodeNoErr = 0 }

Functions

- EXTERN_C CCAUXDLL_API char
  const ∗CCAUXDLL_CALLING_CONV GetHwErrorStatusStringA (unsigned
  short errCode)
- EXTERN_C CCAUXDLL_API wchar_t
  const ∗CCAUXDLL_CALLING_CONV GetHwErrorStatusStringW (unsigned
  short errCode)
- EXTERN_C CCAUXDLL_API char
  const ∗CCAUXDLL_CALLING_CONV GetStartupReasonStringA (unsigned short
  code)
- EXTERN_C CCAUXDLL_API wchar_t
  const ∗CCAUXDLL_CALLING_CONV GetStartupReasonStringW (unsigned
  short code)

7.14 IncludeFiles/DigIO.h File Reference

Namespaces

- namespace CrossControl

Typedefs

- typedef void ∗ DIGIOHANDLE

Functions

- EXTERN_C CCAUXDLL_API
  DIGIOHANDLE
  CCAUXDLL_CALLING_CONV GetDigIO (void)

- EXTERN_C CCAUXDLL_API void
  CCAUXDLL_CALLING_CONV DigIO_release (DIGIOHANDLE)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV DigIO_getDigIO (DIGIOHANDLE, unsigned char ∗status)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV DigIO_setDigIO (DIGIOHANDLE, unsigned char state)

Variables

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

7.15 IncludeFiles/FirmwareUpgrade.h File Reference

Namespaces

- namespace CrossControl

Typedefs

- typedef void ∗ FIRMWAREUPGHANDLE

Functions

- EXTERN_C CCAUXDLL_API
  FIRMWAREUPGHANDLE
  CCAUXDLL_CALLING_CONV GetFirmwareUpgrade (void)
- EXTERN_C CCAUXDLL_API void
  CCAUXDLL_CALLING_CONV FirmwareUpgrade_release (FIRMWAREUPGHANDLE)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV FirmwareUpgrade_startFpgaUpgrade (FIRMWAREUPGHANDLE, const char ∗filename, bool blocking)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV FirmwareUpgrade_startFpgaVerification (FIRMWAREUPGHANDLE, const char ∗filename, bool blocking)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV FirmwareUpgrade_startSSUpgrade (FIRMWAREUPGHANDLE, const char ∗filename, bool blocking)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV FirmwareUpgrade_startSSVerification (FIRMWAREUPGHANDLE, const char ∗filename, bool blocking)

- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV FirmwareUpgrade_startFrontUpgrade (FIRMWAREUPGHANDLE, const char ∗filename, bool blocking)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV FirmwareUpgrade_startFrontVerification (FIRMWAREUPGHANDLE, const char ∗filename, bool blocking)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV FirmwareUpgrade_getUpgradeStatus (FIRMWAREUPGHANDLE, UpgradeStatus ∗status, bool blocking)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV FirmwareUpgrade_shutDown (FIRMWAREUPGHANDLE)

7.16 IncludeFiles/FrontLED.h File Reference

Namespaces

- namespace CrossControl

Typedefs

- typedef void ∗ FRONTLEDHANDLE

Functions

- EXTERN_C CCAUXDLL_API FRONTLEDHANDLE CCAUXDLL_CALLING_CONV GetFrontLED (void)
- EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV FrontLED_release (FRONTLEDHANDLE)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV FrontLED_getSignal (FRONTLEDHANDLE, double ∗frequency, unsigned char ∗dutyCycle)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV FrontLED_getOnTime (FRONTLEDHANDLE, unsigned char ∗onTime)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV FrontLED_getOffTime (FRONTLEDHANDLE, unsigned char ∗offTime)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV FrontLED_getIdleTime (FRONTLEDHANDLE, unsigned char ∗idleTime)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV FrontLED_getNrOfPulses (FRONTLEDHANDLE, unsigned char ∗nrOfPulses)

- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV FrontLED_getColor (FRONTLEDHANDLE, unsigned char ∗red, unsigned char ∗green, unsigned char ∗blue)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV FrontLED_getStandardColor (FRONTLEDHANDLE, CCAuxColor ∗color)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV FrontLED_getEnabledDuringStartup (FRONTLEDHANDLE, CCStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV FrontLED_setSignal (FRONTLEDHANDLE, double frequency, unsigned char dutyCycle)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV FrontLED_setOnTime (FRONTLEDHANDLE, unsigned char onTime)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV FrontLED_setOffTime (FRONTLEDHANDLE, unsigned char offTime)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV FrontLED_setIdleTime (FRONTLEDHANDLE, unsigned char idleTime)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV FrontLED_setNrOfPulses (FRONTLEDHANDLE, unsigned char nrOfPulses)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV FrontLED_setColor (FRONTLEDHANDLE, unsigned char red, unsigned char green, unsigned char blue)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV FrontLED_setStandardColor (FRONTLEDHANDLE, CCAuxColor color)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV FrontLED_setOff (FRONTLEDHANDLE)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV FrontLED_setEnabledDuringStartup (FRONTLEDHANDLE, CCStatus status)

7.17 IncludeFiles/Lightsensor.h File Reference

Namespaces

- namespace CrossControl

Typedefs

- typedef void ∗ LIGHTSENSORHANDLE

Functions

- EXTERN_C CCAUXDLL_API
  LIGHTSENSORHANDLE
  CCAUXDLL_CALLING_CONV GetLightsensor (void)
- EXTERN_C CCAUXDLL_API void
  CCAUXDLL_CALLING_CONV Lightsensor_release (LIGHTSENSORHANDLE)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Lightsensor_getIlluminance (LIGHTSENSORHANDLE, unsigned short ∗value)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Lightsensor_getIlluminance2 (LIGHTSENSORHANDLE, unsigned short ∗value, unsigned char ∗ch0, unsigned char ∗ch1)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Lightsensor_getAverageIlluminance (LIGHTSENSORHANDLE, unsigned short ∗value)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Lightsensor_startAverageCalc (LIGHTSENSORHANDLE, unsigned long averageWndSize, unsigned long rejectWndSize, unsigned long rejectDeltaInLux, LightSensorSamplingMode mode)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Lightsensor_stopAverageCalc (LIGHTSENSORHANDLE)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Lightsensor_getOperatingRange (LIGHTSENSORHANDLE, LightSensorOperationRange ∗range)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Lightsensor_setOperatingRange (LIGHTSENSORHANDLE, LightSensorOperationRange range)

7.18 IncludeFiles/Power.h File Reference

Namespaces

- namespace CrossControl

Typedefs

- typedef void ∗ POWERHANDLE

Functions

- EXTERN_C CCAUXDLL_API
  POWERHANDLE
  CCAUXDLL_CALLING_CONV GetPower (void)

- EXTERN_C CCAUXDLL_API void
CCAUXDLL_CALLING_CONV Power_release (POWERHANDLE)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Power_getBLPowerStatus (POWERHANDLE, CCStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Power_getCanPowerStatus (POWERHANDLE, CCStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Power_getVideoPowerStatus (POWERHANDLE, unsigned char ∗videoStatus)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Power_getExtFanPowerStatus (POWERHANDLE, CCStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Power_getButtonPowerTransitionStatus (POWERHANDLE, ButtonPowerTransitionStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Power_getVideoOCDStatus (POWERHANDLE, OCDStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Power_getCanOCDStatus (POWERHANDLE, OCDStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Power_setBLPowerStatus (POWERHANDLE, CCStatus status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Power_setCanPowerStatus (POWERHANDLE, CCStatus status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Power_setVideoPowerStatus (POWERHANDLE, unsigned char status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Power_setExtFanPowerStatus (POWERHANDLE, CCStatus status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Power_ackPowerRequest (POWERHANDLE)

7.19 IncludeFiles/PowerMgr.h File Reference

Namespaces

- namespace CrossControl

Typedefs

- typedef enum
  CrossControl::PowerMgrConf _PowerMgrConf
- typedef enum
  CrossControl::PowerMgrStatus _PowerMgrStatus
- typedef void ∗ POWERMGRHANDLE

Enumerations

- enum PowerMgrConf { Normal = 0, ApplicationControlled = 1, BatterySuspend = 2 }
- enum PowerMgrStatus { NoRequestsPending = 0, SuspendPending = 1, Shutdown-Pending = 2 }

Functions

- EXTERN_C CCAUXDLL_API
  POWERMGRHANDLE
  CCAUXDLL_CALLING_CONV GetPowerMgr (void)
- EXTERN_C CCAUXDLL_API void
  CCAUXDLL_CALLING_CONV PowerMgr_release (POWERMGRHANDL-E)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV PowerMgr_registerControlledSuspendOrShut-Down (POWERMGRHANDLE, PowerMgrConf conf)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV PowerMgr_getConfiguration (POWERMGR-HANDLE, PowerMgrConf ∗conf)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV PowerMgr_getPowerMgrStatus (POWERM-GRHANDLE, PowerMgrStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV PowerMgr_setAppReadyForSuspendOrShutdown (POWERMGRHANDLE)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV PowerMgr_hasResumed (POWERMGRHA-NDLE, bool ∗resumed)

7.20 IncludeFiles/Releasenotes.dox File Reference

7.21 IncludeFiles/Smart.h File Reference

Namespaces

- namespace CrossControl

Typedefs

- typedef void ∗ SMARTHANDLE

Functions

- EXTERN_C CCAUXDLL_API
SMARTHANDLE
CCAUXDLL_CALLING_CONV GetSmart (void)
- EXTERN_C CCAUXDLL_API void
CCAUXDLL_CALLING_CONV Smart_release (SMARTHANDLE)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Smart_getRemainingLifeTime (SMARTHA-NDLE, unsigned char ∗lifetimepercent)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Smart_getDeviceSerial (SMARTHANDLE, char ∗buff, int len)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Smart_getInitialTime (SMARTHANDLE, time-_t ∗time)

7.22 IncludeFiles/Telematics.h File Reference

Namespaces

- namespace CrossControl

Typedefs

- typedef void ∗ TELEMATICSHANDLE

Functions

- EXTERN_C CCAUXDLL_API
TELEMATICSHANDLE
CCAUXDLL_CALLING_CONV GetTelematics (void)
- EXTERN_C CCAUXDLL_API void
CCAUXDLL_CALLING_CONV Telematics_release (TELEMATICSHANDL-E)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Telematics_getTelematicsAvailable (TELEM-ATICSHANDLE, CCStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Telematics_getGPRSPowerStatus (TELEM-ATICSHANDLE, CCStatus ∗status)

- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Telematics_getGPRSStartUpPowerStatus (TELEMATICSHANDLE, CCStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Telematics_getWLANPowerStatus (TELEMATICSHANDLE, CCStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Telematics_getWLANStartUpPowerStatus (TELEMATICSHANDLE, CCStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Telematics_getBTPowerStatus (TELEMATICSHANDLE, CCStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Telematics_getBTStartUpPowerStatus (TELEMATICSHANDLE, CCStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Telematics_getGPSPowerStatus (TELEMATICSHANDLE, CCStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Telematics_getGPSStartUpPowerStatus (TELEMATICSHANDLE, CCStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Telematics_getGPSAntennaStatus (TELEMATICSHANDLE, CCStatus ∗status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Telematics_setGPRSPowerStatus (TELEMATICSHANDLE, CCStatus status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Telematics_setGPRSStartUpPowerStatus (TELEMATICSHANDLE, CCStatus status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Telematics_setWLANPowerStatus (TELEMATICSHANDLE, CCStatus status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Telematics_setWLANStartUpPowerStatus (TELEMATICSHANDLE, CCStatus status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Telematics_setBTPowerStatus (TELEMATICSHANDLE, CCStatus status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Telematics_setBTStartUpPowerStatus (TELEMATICSHANDLE, CCStatus status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Telematics_setGPSPowerStatus (TELEMATICSHANDLE, CCStatus status)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Telematics_setGPSStartUpPowerStatus (TELEMATICSHANDLE, CCStatus status)

7.23 IncludeFiles/TouchScreen.h File Reference

Namespaces

- namespace CrossControl

Typedefs

- typedef void ∗ TOUCHSCREENHANDLE

Enumerations

- enum TouchScreenModeSettings { MOUSE_NEXT_BOOT = 0, TOUCH_NE-XT_BOOT = 1, MOUSE_NOW = 2, TOUCH_NOW = 3 }
- enum TSAdvancedSettingsParameter {
  TS_RIGHT_CLICK_TIME = 0, TS_LOW_LEVEL = 1, TS_UNTOUCHLEV-EL = 2, TS_DEBOUNCE_TIME = 3,
  TS_DEBOUNCE_TIMEOUT_TIME = 4, TS_DOUBLECLICK_MAX_CLIC-K_TIME = 5, TS_DOUBLE_CLICK_TIME = 6, TS_MAX_RIGHTCLICK_D-ISTANCE = 7,
  TS_USE_DEJITTER = 8, TS_CALIBTATION_WIDTH = 9, TS_CALIBRAT-ION_MEASUREMENTS = 10, TS_RESTORE_DEFAULT_SETTINGS = 11,
  TS_TCHAUTOCAL = 12 }

Functions

- EXTERN_C CCAUXDLL_API
  TOUCHSCREENHANDLE
  CCAUXDLL_CALLING_CONV GetTouchScreen (void)
- EXTERN_C CCAUXDLL_API void
  CCAUXDLL_CALLING_CONV TouchScreen_release (TOUCHSCREENHA-NDLE)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV TouchScreen_getMode (TOUCHSCREENH-ANDLE, TouchScreenModeSettings ∗config)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV TouchScreen_getMouseRightClickTime (TO-UCHSCREENHANDLE, unsigned short ∗time)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV TouchScreen_setMode (TOUCHSCREENH-ANDLE, TouchScreenModeSettings config)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV TouchScreen_setMouseRightClickTime (TO-UCHSCREENHANDLE, unsigned short time)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV TouchScreen_setAdvancedSetting (TOUCH-SCREENHANDLE, TSAdvancedSettingsParameter param, unsigned short data)

- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV TouchScreen_getAdvancedSetting (TOUCH-SCREENHANDLE, TSAdvancedSettingsParameter param, unsigned short ∗data)

7.24 IncludeFiles/TouchScreenCalib.h File Reference

Namespaces

- namespace CrossControl

Typedefs

- typedef void ∗ TOUCHSCREENCALIBHANDLE

Enumerations

- enum CalibrationModeSettings {
MODE_UNKNOWN = 0, MODE_NORMAL = 1, MODE_CALIBRATION_-5P = 2, MODE_CALIBRATION_9P = 3,
MODE_CALIBRATION_13P = 4 }
- enum CalibrationConfigParam {
CONFIG_CALIBRATION_WITH = 0, CONFIG_CALIBRATION_MEASU-REMENTS = 1, CONFIG_5P_CALIBRATION_POINT_BORDER = 2, CO-NFIG_13P_CALIBRATION_POINT_BORDER = 3,
CONFIG_13P_CALIBRATION_TRANSITION_MIN = 4, CONFIG_13P_CA-LIBRATION_TRANSITION_MAX = 5 }

Functions

- EXTERN_C CCAUXDLL_API
TOUCHSCREENCALIBHANDLE
CCAUXDLL_CALLING_CONV GetTouchScreenCalib (void)
- EXTERN_C CCAUXDLL_API void
CCAUXDLL_CALLING_CONV TouchScreenCalib_release (TOUCHSCREE-NCALIBHANDLE)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV TouchScreenCalib_setMode (TOUCHSCRE-ENCALIBHANDLE, CalibrationModeSettings mode)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV TouchScreenCalib_getMode (TOUCHSCRE-ENCALIBHANDLE, CalibrationModeSettings ∗mode)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV TouchScreenCalib_setCalibrationPoint (TO-UCHSCREENCALIBHANDLE, 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 (TOUC-
  HSCREENCALIBHANDLE, CalibrationConfigParam param, unsigned short ∗value)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV TouchScreenCalib_setConfigParam (TOUC-
  HSCREENCALIBHANDLE, CalibrationConfigParam param, unsigned short value)

7.25 IncludeFiles/Video.h File Reference

Namespaces

- namespace CrossControl

Typedefs

- typedef void ∗ VIDEOHANDLE

Functions

- EXTERN_C CCAUXDLL_API
  VIDEOHANDLE
  CCAUXDLL_CALLING_CONV GetVideo (void)
- EXTERN_C CCAUXDLL_API void
  CCAUXDLL_CALLING_CONV Video_release (VIDEOHANDLE)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_init (VIDEOHANDLE, unsigned char
  deviceNr)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_showVideo (VIDEOHANDLE, bool show)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_setDeInterlaceMode (VIDEOHAND-
  LE, DeInterlaceMode mode)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_getDeInterlaceMode (VIDEOHAND-
  LE, DeInterlaceMode ∗mode)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_setMirroring (VIDEOHANDLE, CC-
  Status mode)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_getMirroring (VIDEOHANDLE, CC-
  Status ∗mode)

- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_setRotation (VIDEOHANDLE, VideoRotation rotation)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_getRotation (VIDEOHANDLE, VideoRotation ∗rotation)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_setActiveChannel (VIDEOHANDLE, VideoChannel channel)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_getActiveChannel (VIDEOHANDLE, VideoChannel ∗channel)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_setColorKeys (VIDEOHANDLE, unsigned char rKey, unsigned char gKey, unsigned char bKey)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_getColorKeys (VIDEOHANDLE, unsigned char ∗rKey, unsigned char ∗gKey, unsigned char ∗bKey)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_setVideoArea (VIDEOHANDLE, unsigned short topLeftX, unsigned short topLeftY, unsigned short bottomRightX, unsigned short bottomRightY)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_getRawImage (VIDEOHANDLE, unsigned short ∗width, unsigned short ∗height, float ∗frameRate)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_getVideoArea (VIDEOHANDLE, unsigned short ∗topLeftX, unsigned short ∗topLeftY, unsigned short ∗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 bNTSC-Format)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Video_createBitmap (VIDEOHANDLE, char ∗∗bmpBuffer, unsigned long ∗bmpBufSize, const char ∗rawImgBuffer, unsigned long rawImgBufSize, bool bInterlaced, bool bNTSCFormat)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Video_freeBmpBuffer (VIDEOHANDLE, char ∗bmpBuffer)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Video_minimize (VIDEOHANDLE)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Video_restore (VIDEOHANDLE)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Video_setDecoderReg (VIDEOHANDLE, unsigned char decoderRegister, unsigned char registerValue)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Video_getDecoderReg (VIDEOHANDLE, unsigned char decoderRegister, unsigned char ∗registerValue)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Video_setCropping (VIDEOHANDLE, unsigned char top, unsigned char left, unsigned char bottom, unsigned char right)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Video_getCropping (VIDEOHANDLE, unsigned char ∗top, unsigned char ∗left, unsigned char ∗bottom, unsigned char ∗right)
- EXTERN_C CCAUXDLL_API eErr
CCAUXDLL_CALLING_CONV Video_showFrame (VIDEOHANDLE)

Index