CCAux
2.8.3.0
Wed Apr 06 2016
## Contents

1 Main Page
   1.1 Introduction ................................................. 1
   1.2 Changelog .................................................. 1
      1.2.1 Version 2.8.3.0 - VC/VA Linux platforms .............. 1
      1.2.2 Version 2.7.4.0 - VC Linux platform ................. 2
      1.2.3 Version 2.7.3.0 - VC Linux platform ................. 2
      1.2.4 Version 2.7.2.0 - XM/XL Windows x86, x64 platform, VC Linux platform .............................................. 2
      1.2.5 Version 2.7.1.0 - XM/XL Windows x86, x64 platform, VC Linux platform .............................................. 2
      1.2.6 Version 2.7.0.0 - XM/XL Windows x86, x64 platform, VC Linux platform .............................................. 2
      1.2.7 Version 2.6.2.0 - XM/XL Windows x86, x64 platform ... 3
      1.2.8 Version 2.6.1.0 - XM/XL Windows x86, x64 platform and X-A/XS Linux platform ............................................ 3
      1.2.9 Version 2.5.0.0 - XM/XL x86 platform ................. 4
      1.2.10 Version 2.4.7.0 - XM Linux platform ................. 5
      1.2.11 Version 2.4.6.0 - XA/XS platform ............... 5
      1.2.12 Version 2.4.2.0 - XA/XS platform ............... 5
      1.2.13 Version 2.4.0.0 - XA/XS, XM platforms .......... 5
      1.2.14 Version 2.3.0.0 - XA/XS platform ............... 7
      1.2.15 Version 2.2.0.0 - XA/XS platform ............... 7
      1.2.16 Version 2.1.0.0 - XA/XS platform ............... 8
      1.2.17 Version 2.0.0.0 - XA/XS platform ............... 8
   1.3 Known Issues .............................................. 9

2 Namespace Index .............................................. 10
   2.1 Namespace List ............................................. 10

3 Data Structure Index ........................................ 11
   3.1 Data Structures ........................................... 11

4 File Index ..................................................... 12
   4.1 File List .................................................. 12

5 Namespace Documentation .................................... 13
   5.1 CrossControl Namespace Reference ....................... 13
      5.1.1 Typedef Documentation .................................. 36
          5.1.1.1 _PowerMgrConf .................................. 36
5.1.1.2 _PowerMgrStatus ........................................ 36
5.1.1.3 ABOUTHANDLE ........................................ 36
5.1.1.4 ADCHANDLE ........................................ 36
5.1.1.5 AUXVERSIONHANDLE .................................. 36
5.1.1.6 BACKLIGHTHANDLE ................................... 36
5.1.1.7 BATTERYHANDLE ...................................... 36
5.1.1.8 BUZZERHANDLE ....................................... 36
5.1.1.9 CANSETTINGHANDLE .................................. 37
5.1.1.10 CFGINHANDLE ........................................ 37
5.1.1.11 CONFIGHANDLE ...................................... 37
5.1.1.12 DIAGNOSTICHANDLE .................................. 37
5.1.1.13 DIGIOHANDLE ........................................ 37
5.1.1.14 FIRMWAREUPDATEHANDLE .............................. 37
5.1.1.15 FRONTLEDHANDLE .................................... 37
5.1.1.16 LIGHTSENSORHANDLE ................................ 37
5.1.1.17 POWERHANDLE ......................................... 37
5.1.1.18 POWERMGRHANDLE .................................... 37
5.1.1.19 PWMOUTHANDLE ....................................... 37
5.1.1.20 SMARTHANDLE ......................................... 37
5.1.1.21 TELEMATICSHANDLE ................................... 37
5.1.1.22 TOUCHSCREENCALIBHANDLE ........................... 37
5.1.1.23 TOUCHSCREENHANDLE ................................ 37
5.1.1.24 VersionType .......................................... 37
5.1.1.25 VIDEOHANDLE ......................................... 37
5.1.2 Enumeration Type Documentation .......................... 37
5.1.2.1 ButtonConfigEnum ...................................... 37
5.1.2.2 ButtonPowerTransitionStatus ......................... 38
5.1.2.3 CalibrationConfigParam ............................... 38
5.1.2.4 CalibrationModeSettings .............................. 39
5.1.2.5 CanFrameType ........................................ 39
5.1.2.6 CCAuxColor ........................................... 39
5.1.2.7 CCStatus .............................................. 40
5.1.2.8 CfgInModeEnum ........................................ 40
5.1.2.9 ChargingStatus ........................................ 41
5.1.2.10 ConfigOnOffTriggerMode .............................. 42
5.1.2.11 DeInterlaceMode ...................................... 42
5.1.2.12 eErr .................................................. 42
5.1.2.13 ErrorStatus ......................................... 43
5.1.2.14 hwErrorStatusCodes ................................ 43
5.1.2.15 JidaSensorType ....................................... 44
5.1.2.16 LightSensorOperationRange ......................... 44
5.1.2.17 LightSensorSamplingMode ............................ 44
5.1.2.18 OCDStatus ............................................ 44
5.1.2.19 PowerAction ......................................... 45
5.1.2.20 PowerMgrConf ....................................... 45
5.1.2.21 PowerMgrStatus ...................................... 45
5.1.2.22 PowerSource ......................................... 45
5.1.2.23 RS4XXPort .......................................... 46
5.1.2.24 shutDownReasonCodes ............................... 46
5.1.2.25 startupReasonCodes ................................ 46
### CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1.2.26</td>
<td>TouchScreenModeSettings</td>
<td>46</td>
</tr>
<tr>
<td>5.1.2.27</td>
<td>TriggerConf</td>
<td>47</td>
</tr>
<tr>
<td>5.1.2.28</td>
<td>TSAdvancedSettingsParameter</td>
<td>48</td>
</tr>
<tr>
<td>5.1.2.29</td>
<td>UpgradeAction</td>
<td>49</td>
</tr>
<tr>
<td>5.1.2.30</td>
<td>VideoChannel</td>
<td>49</td>
</tr>
<tr>
<td>5.1.2.31</td>
<td>VideoRotation</td>
<td>49</td>
</tr>
<tr>
<td>5.1.2.32</td>
<td>videoStandard</td>
<td>50</td>
</tr>
<tr>
<td>5.1.2.33</td>
<td>VoltageEnum</td>
<td>50</td>
</tr>
<tr>
<td>5.1.3.1</td>
<td>About_getAddOnHWversion</td>
<td>51</td>
</tr>
<tr>
<td>5.1.3.2</td>
<td>About_getAddOnManufacturingDate</td>
<td>51</td>
</tr>
<tr>
<td>5.1.3.3</td>
<td>About_getAddOnPCBArt</td>
<td>52</td>
</tr>
<tr>
<td>5.1.3.4</td>
<td>About_getAddOnPCBSerial</td>
<td>52</td>
</tr>
<tr>
<td>5.1.3.5</td>
<td>About_getDisplayResolution</td>
<td>53</td>
</tr>
<tr>
<td>5.1.3.6</td>
<td>About_getFrontPcbRev</td>
<td>53</td>
</tr>
<tr>
<td>5.1.3.7</td>
<td>About_getIOExpanderValue</td>
<td>53</td>
</tr>
<tr>
<td>5.1.3.8</td>
<td>About_getIsAnybus Mounted</td>
<td>54</td>
</tr>
<tr>
<td>5.1.3.9</td>
<td>About_getIsBTMounted</td>
<td>54</td>
</tr>
<tr>
<td>5.1.3.10</td>
<td>About_getIsDisplayAvailable</td>
<td>55</td>
</tr>
<tr>
<td>5.1.3.11</td>
<td>About_getIsGPRSMounted</td>
<td>55</td>
</tr>
<tr>
<td>5.1.3.12</td>
<td>About_getIsGPSMounted</td>
<td>56</td>
</tr>
<tr>
<td>5.1.3.13</td>
<td>About_getIsIOExpanderMounted</td>
<td>56</td>
</tr>
<tr>
<td>5.1.3.14</td>
<td>About_getIsTouchScreenAvailable</td>
<td>56</td>
</tr>
<tr>
<td>5.1.3.15</td>
<td>About_getIsWLANMounted</td>
<td>57</td>
</tr>
<tr>
<td>5.1.3.16</td>
<td>About_getMainHWversion</td>
<td>57</td>
</tr>
<tr>
<td>5.1.3.17</td>
<td>About_getMainManufacturingDate</td>
<td>58</td>
</tr>
<tr>
<td>5.1.3.18</td>
<td>About_getMainPCBArt</td>
<td>58</td>
</tr>
<tr>
<td>5.1.3.19</td>
<td>About_getMainPCBSerial</td>
<td>59</td>
</tr>
<tr>
<td>5.1.3.20</td>
<td>About_getMainProdArtNr</td>
<td>59</td>
</tr>
<tr>
<td>5.1.3.21</td>
<td>About_getMainProdRev</td>
<td>60</td>
</tr>
<tr>
<td>5.1.3.22</td>
<td>About_getNrOfButtons</td>
<td>60</td>
</tr>
<tr>
<td>5.1.3.23</td>
<td>About_getNrOfCANConnections</td>
<td>61</td>
</tr>
<tr>
<td>5.1.3.24</td>
<td>About_getNrOfCIGInConnections</td>
<td>61</td>
</tr>
<tr>
<td>5.1.3.25</td>
<td>About_getNrOfDigIOConnections</td>
<td>62</td>
</tr>
<tr>
<td>5.1.3.26</td>
<td>About_getNrOfETHConnections</td>
<td>62</td>
</tr>
<tr>
<td>5.1.3.27</td>
<td>About_getNrOfPWMOutConnections</td>
<td>63</td>
</tr>
<tr>
<td>5.1.3.28</td>
<td>About_getNrOfSerialConnections</td>
<td>63</td>
</tr>
<tr>
<td>5.1.3.29</td>
<td>About_getNrOfUSBConnections</td>
<td>64</td>
</tr>
<tr>
<td>5.1.3.30</td>
<td>About_getNrOfVideoConnections</td>
<td>64</td>
</tr>
<tr>
<td>5.1.3.31</td>
<td>About_getUnitSerial</td>
<td>65</td>
</tr>
<tr>
<td>5.1.3.32</td>
<td>About_getUserEepromData</td>
<td>65</td>
</tr>
<tr>
<td>5.1.3.33</td>
<td>About_hasOsBooted</td>
<td>66</td>
</tr>
<tr>
<td>5.1.3.34</td>
<td>About_release</td>
<td>66</td>
</tr>
<tr>
<td>5.1.3.35</td>
<td>About_setUserEepromData</td>
<td>67</td>
</tr>
<tr>
<td>5.1.3.36</td>
<td>Adc_getVoltage</td>
<td>67</td>
</tr>
<tr>
<td>5.1.3.37</td>
<td>Adc_release</td>
<td>68</td>
</tr>
<tr>
<td>5.1.3.38</td>
<td>AuxVersion_getCCAuxDrvVersion</td>
<td>68</td>
</tr>
<tr>
<td>5.1.3.39</td>
<td>AuxVersion_getCCAuxVersion</td>
<td>69</td>
</tr>
<tr>
<td>5.1.3.40</td>
<td>AuxVersion_getFPGAVersion</td>
<td>70</td>
</tr>
<tr>
<td>5.1.3.41</td>
<td>AuxVersion_getFrontVersion</td>
<td>71</td>
</tr>
<tr>
<td>Function Name</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------------------------------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>CfgIn_setMinFrequencyThreshold</td>
<td>99</td>
<td></td>
</tr>
<tr>
<td>Config_getButtonFunction</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>Config_getCanStartupPowerConfig</td>
<td>101</td>
<td></td>
</tr>
<tr>
<td>Config_getExtFanStartupPowerConfig</td>
<td>101</td>
<td></td>
</tr>
<tr>
<td>Config_getExtOnOffSigTrigTime</td>
<td>102</td>
<td></td>
</tr>
<tr>
<td>Config_getFrontBtnTrigTime</td>
<td>102</td>
<td></td>
</tr>
<tr>
<td>Config_getHeatingTempLimit</td>
<td>102</td>
<td></td>
</tr>
<tr>
<td>Config_getLongButtonPressAction</td>
<td>103</td>
<td></td>
</tr>
<tr>
<td>Config_getOnOffSigAction</td>
<td>103</td>
<td></td>
</tr>
<tr>
<td>Config_getOnOffSignalState</td>
<td>104</td>
<td></td>
</tr>
<tr>
<td>Config_getOnOffTriggerMode</td>
<td>104</td>
<td></td>
</tr>
<tr>
<td>Config_getPowerOnStartup</td>
<td>104</td>
<td></td>
</tr>
<tr>
<td>Config_getRS485Enabled</td>
<td>105</td>
<td></td>
</tr>
<tr>
<td>Config_getShortButtonPressAction</td>
<td>105</td>
<td></td>
</tr>
<tr>
<td>Config_getStartupTriggerConfig</td>
<td>105</td>
<td></td>
</tr>
<tr>
<td>Config_getStartupVoltageConfig</td>
<td>106</td>
<td></td>
</tr>
<tr>
<td>Config_getSuspendMaxTime</td>
<td>107</td>
<td></td>
</tr>
<tr>
<td>Config_getVideoStartupPowerConfig</td>
<td>107</td>
<td></td>
</tr>
<tr>
<td>Config_release</td>
<td>108</td>
<td></td>
</tr>
<tr>
<td>Config_setButtonFunction</td>
<td>108</td>
<td></td>
</tr>
<tr>
<td>Config_setCanStartupPowerConfig</td>
<td>108</td>
<td></td>
</tr>
<tr>
<td>Config_setExtFanStartupPowerConfig</td>
<td>109</td>
<td></td>
</tr>
<tr>
<td>Config_setExtOnOffSigTrigTime</td>
<td>109</td>
<td></td>
</tr>
<tr>
<td>Config_setFrontBtnTrigTime</td>
<td>110</td>
<td></td>
</tr>
<tr>
<td>Config_setHeatingTempLimit</td>
<td>110</td>
<td></td>
</tr>
<tr>
<td>Config_setLongButtonPressAction</td>
<td>110</td>
<td></td>
</tr>
<tr>
<td>Config_setOnOffSigAction</td>
<td>111</td>
<td></td>
</tr>
<tr>
<td>Config_setOnOffTriggerMode</td>
<td>111</td>
<td></td>
</tr>
<tr>
<td>Config_setPowerOnStartup</td>
<td>112</td>
<td></td>
</tr>
<tr>
<td>Config_setRS485Enabled</td>
<td>112</td>
<td></td>
</tr>
<tr>
<td>Config_setShortButtonPressAction</td>
<td>112</td>
<td></td>
</tr>
<tr>
<td>Config_setStartupTriggerConfig</td>
<td>113</td>
<td></td>
</tr>
<tr>
<td>Config_setStartupVoltageConfig</td>
<td>113</td>
<td></td>
</tr>
<tr>
<td>Config_setSuspendMaxTime</td>
<td>114</td>
<td></td>
</tr>
<tr>
<td>Config_setVideoStartupPowerConfig</td>
<td>114</td>
<td></td>
</tr>
<tr>
<td>Diagnostic_clearHwErrorStatus</td>
<td>115</td>
<td></td>
</tr>
<tr>
<td>Diagnostic_getHwErrorStatus</td>
<td>115</td>
<td></td>
</tr>
<tr>
<td>Diagnostic_getMinMaxTemp</td>
<td>115</td>
<td></td>
</tr>
<tr>
<td>Diagnostic_getPCBTemp</td>
<td>116</td>
<td></td>
</tr>
<tr>
<td>Diagnostic_getPMTemp</td>
<td>116</td>
<td></td>
</tr>
<tr>
<td>Diagnostic_getPMTemp</td>
<td>116</td>
<td></td>
</tr>
<tr>
<td>Diagnostic_getPowerCycles</td>
<td>117</td>
<td></td>
</tr>
<tr>
<td>Diagnostic_getShutdownReason</td>
<td>117</td>
<td></td>
</tr>
<tr>
<td>Diagnostic_getSSTemp</td>
<td>117</td>
<td></td>
</tr>
<tr>
<td>Diagnostic_getStartupReason</td>
<td>118</td>
<td></td>
</tr>
<tr>
<td>Diagnostic_getTimer</td>
<td>118</td>
<td></td>
</tr>
<tr>
<td>Diagnostic_release</td>
<td>119</td>
<td></td>
</tr>
<tr>
<td>DigIO_getDigIO</td>
<td>119</td>
<td></td>
</tr>
<tr>
<td>DigIO_release</td>
<td>120</td>
<td></td>
</tr>
<tr>
<td>DigIO_setDigIO</td>
<td>120</td>
<td></td>
</tr>
<tr>
<td>FirmwareUpgrade_getUpgradeStatus</td>
<td>121</td>
<td></td>
</tr>
<tr>
<td>Method Name</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>5.1.3.242 Telematics_getGPSPowerStatus</td>
<td>178</td>
<td></td>
</tr>
<tr>
<td>5.1.3.243 Telematics_getGPRSSStartUpPowerStatus</td>
<td>179</td>
<td></td>
</tr>
<tr>
<td>5.1.3.244 Telematics_getTelematicsAvailable</td>
<td>180</td>
<td></td>
</tr>
<tr>
<td>5.1.3.245 Telematics_getWLANPowerStatus</td>
<td>180</td>
<td></td>
</tr>
<tr>
<td>5.1.3.246 Telematics_getWLANStartUpPowerStatus</td>
<td>181</td>
<td></td>
</tr>
<tr>
<td>5.1.3.247 Telematics_release</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.1.3.248 Telematics_setBTPowerStatus</td>
<td>182</td>
<td></td>
</tr>
<tr>
<td>5.1.3.249 Telematics_setBTStartUpPowerStatus</td>
<td>182</td>
<td></td>
</tr>
<tr>
<td>5.1.3.250 Telematics_setGPRSPowerStatus</td>
<td>183</td>
<td></td>
</tr>
<tr>
<td>5.1.3.251 Telematics_setGPRSSStartUpPowerStatus</td>
<td>183</td>
<td></td>
</tr>
<tr>
<td>5.1.3.252 Telematics_setGPSPowerStatus</td>
<td>183</td>
<td></td>
</tr>
<tr>
<td>5.1.3.253 Telematics_setGPRSSStartUpPowerStatus</td>
<td>184</td>
<td></td>
</tr>
<tr>
<td>5.1.3.254 Telematics_setWLANPowerStatus</td>
<td>184</td>
<td></td>
</tr>
<tr>
<td>5.1.3.255 Telematics_setWLANStartUpPowerStatus</td>
<td>184</td>
<td></td>
</tr>
<tr>
<td>5.1.3.256 TouchScreen_getAdvancedSetting</td>
<td>185</td>
<td></td>
</tr>
<tr>
<td>5.1.3.257 TouchScreen_getMode</td>
<td>185</td>
<td></td>
</tr>
<tr>
<td>5.1.3.258 TouchScreen_getMouseRightClickTime</td>
<td>186</td>
<td></td>
</tr>
<tr>
<td>5.1.3.259 TouchScreen_release</td>
<td>187</td>
<td></td>
</tr>
<tr>
<td>5.1.3.260 TouchScreen_setAdvancedSetting</td>
<td>187</td>
<td></td>
</tr>
<tr>
<td>5.1.3.261 TouchScreen_setMode</td>
<td>187</td>
<td></td>
</tr>
<tr>
<td>5.1.3.262 TouchScreen_setMouseRightClickTime</td>
<td>188</td>
<td></td>
</tr>
<tr>
<td>5.1.3.263 TouchScreenCalib_autoSensorCalib</td>
<td>188</td>
<td></td>
</tr>
<tr>
<td>5.1.3.264 TouchScreenCalib_checkCalibrationPointFinished</td>
<td>188</td>
<td></td>
</tr>
<tr>
<td>5.1.3.265 TouchScreenCalib_getConfigParam</td>
<td>189</td>
<td></td>
</tr>
<tr>
<td>5.1.3.266 TouchScreenCalib_getMode</td>
<td>189</td>
<td></td>
</tr>
<tr>
<td>5.1.3.267 TouchScreenCalib_release</td>
<td>189</td>
<td></td>
</tr>
<tr>
<td>5.1.3.268 TouchScreenCalib_setCalibrationPoint</td>
<td>190</td>
<td></td>
</tr>
<tr>
<td>5.1.3.269 TouchScreenCalib_setConfigParam</td>
<td>190</td>
<td></td>
</tr>
<tr>
<td>5.1.3.270 TouchScreenCalib_setMode</td>
<td>190</td>
<td></td>
</tr>
<tr>
<td>5.1.3.271 Video_activateSnapshot</td>
<td>191</td>
<td></td>
</tr>
<tr>
<td>5.1.3.272 Video_createBitmap</td>
<td>191</td>
<td></td>
</tr>
<tr>
<td>5.1.3.273 Video_freeBmpBuffer</td>
<td>192</td>
<td></td>
</tr>
<tr>
<td>5.1.3.274 Video_getActiveChannel</td>
<td>192</td>
<td></td>
</tr>
<tr>
<td>5.1.3.275 Video_getColorKeys</td>
<td>192</td>
<td></td>
</tr>
<tr>
<td>5.1.3.276 Video_getCropping</td>
<td>193</td>
<td></td>
</tr>
<tr>
<td>5.1.3.277 Video_getDecoderReg</td>
<td>193</td>
<td></td>
</tr>
<tr>
<td>5.1.3.278 Video_getDeInterlaceMode</td>
<td>194</td>
<td></td>
</tr>
<tr>
<td>5.1.3.279 Video_getGraphicsOverlay</td>
<td>194</td>
<td></td>
</tr>
<tr>
<td>5.1.3.280 Video_getMirroring</td>
<td>194</td>
<td></td>
</tr>
<tr>
<td>5.1.3.281 Video_getRawImage</td>
<td>195</td>
<td></td>
</tr>
<tr>
<td>5.1.3.282 Video_getRotation</td>
<td>195</td>
<td></td>
</tr>
<tr>
<td>5.1.3.283 Video_getScaling</td>
<td>195</td>
<td></td>
</tr>
<tr>
<td>5.1.3.284 Video_getStatus</td>
<td>196</td>
<td></td>
</tr>
<tr>
<td>5.1.3.285 Video_getVideoArea</td>
<td>196</td>
<td></td>
</tr>
<tr>
<td>5.1.3.286 Video_getVideoStandard</td>
<td>197</td>
<td></td>
</tr>
<tr>
<td>5.1.3.287 Video_init</td>
<td>197</td>
<td></td>
</tr>
<tr>
<td>5.1.3.288 Video_minimize</td>
<td>197</td>
<td></td>
</tr>
<tr>
<td>5.1.3.289 Video_release</td>
<td>198</td>
<td></td>
</tr>
<tr>
<td>5.1.3.290 Video_restore</td>
<td>198</td>
<td></td>
</tr>
<tr>
<td>5.1.3.291 Video_setActiveChannel</td>
<td>198</td>
<td></td>
</tr>
</tbody>
</table>
5.1.3.292 Video_setColorKeys .................................. 198
5.1.3.293 Video_setCropping .................................. 199
5.1.3.294 Video_setDecoderReg ................................ 199
5.1.3.295 Video_setDeInterlaceMode ........................... 200
5.1.3.296 Video_setGraphicsOverlay ............................ 200
5.1.3.297 Video_setMirroring .................................. 200
5.1.3.298 Video_setRotation .................................. 201
5.1.3.299 Video_setScaling .................................. 201
5.1.3.300 Video_setVideoArea ................................ 202
5.1.3.301 Video_showFrame .................................. 202
5.1.3.302 Video_showVideo .................................. 202
5.1.3.303 Video_takeSnapshot ................................ 203
5.1.3.304 Video_takeSnapshotBmp .............................. 203
5.1.3.305 Video_takeSnapshotRaw .............................. 204

5.1.4 Variable Documentation .................................. 204
5.1.4.1 DigitalIn_1 .......................................... 204
5.1.4.2 DigitalIn_2 .......................................... 204
5.1.4.3 DigitalIn_3 .......................................... 204
5.1.4.4 DigitalIn_4 .......................................... 204
5.1.4.5 Video1Conf .......................................... 204
5.1.4.6 Video2Conf .......................................... 205
5.1.4.7 Video3Conf .......................................... 205
5.1.4.8 Video4Conf .......................................... 205

6 Data Structure Documentation ................................ 206
6.1 BatteryTimerType Struct Reference ......................... 206
6.1.1 Field Documentation ................................... 206
6.1.1.1 RunTime_0_40 ....................................... 206
6.1.1.2 RunTime_40_60 ....................................... 206
6.1.1.3 RunTime_60_70 ....................................... 206
6.1.1.4 RunTime_70_80 ....................................... 207
6.1.1.5 RunTime_Above80 ................................... 207
6.1.1.6 RunTime_m20 ......................................... 207
6.1.1.7 RunTime_m20_0 ....................................... 207
6.1.1.8 TotRunTimeBattery .................................. 207
6.1.1.9 TotRunTimeMain ..................................... 207

6.2 BuzzerSetup Struct Reference .............................. 207
6.2.1 Field Documentation ................................... 207
6.2.1.1 frequency ........................................... 207
6.2.1.2 volume .............................................. 208

6.3 FpgaLedTimingType Struct Reference ....................... 208
6.3.1 Field Documentation ................................... 208
6.3.1.1 idleTime .......................................... 208
6.3.1.2 ledNbr .............................................. 208
6.3.1.3 nrOfPulses ......................................... 208
6.3.1.4 offTime .......................................... 208
6.3.1.5 onTime ............................................. 208

6.4 LedColorMixType Struct Reference ......................... 209
6.4.1 Field Documentation ................................... 209
6.4.1.1 blue .............................................. 209
## CONTENTS

| 7.10 | IncludeFiles/CCPlatform.h File Reference | 225 |
| 7.11 | IncludeFiles/CfgIn.h File Reference | 225 |
| 7.12 | IncludeFiles/Config.h File Reference | 226 |
| 7.13 | IncludeFiles/Diagnostic.h File Reference | 229 |
| 7.14 | IncludeFiles/DiagnosticCodes.h File Reference | 230 |
| 7.15 | IncludeFiles/DigIO.h File Reference | 231 |
| 7.16 | IncludeFiles/FirmwareUpgrade.h File Reference | 231 |
| 7.17 | IncludeFiles/FrontLED.h File Reference | 232 |
| 7.18 | IncludeFiles/Lightsensor.h File Reference | 234 |
| 7.19 | IncludeFiles/Power.h File Reference | 235 |
| 7.20 | IncludeFiles/PowerMgr.h File Reference | 236 |
| 7.21 | IncludeFiles/PWMOut.h File Reference | 237 |
| 7.22 | IncludeFiles/Releasesnotes.dox File Reference | 238 |
| 7.23 | IncludeFiles/Smart.h File Reference | 238 |
| 7.24 | IncludeFiles/Telematics.h File Reference | 239 |
| 7.25 | IncludeFiles/TouchScreen.h File Reference | 240 |
| 7.26 | IncludeFiles/TouchScreenCalib.h File Reference | 241 |
| 7.27 | IncludeFiles/Video.h File Reference | 242 |

## Index

| 245 |
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, VC and XM products in all variations. For the XM and XL platforms, Windows XP, Windows 7 and 8 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 2.8.3.0 - VC/VA Linux platforms

- Support for VA platform.
- Code corrections based on static analysis which should lead to improved reliability.
- Config_[set|get]OnOffSignalState API functions added.
1.2 Changelog

- Config_{set|get}KeySwitchTriggerMode API functions added.
- Using modified stm32flash for stm32 flashing in Linux.
- The function Telematics_getGPSAntennaStatus is not supported in the XA/XS platform.
- The function Telematics_getGPSAntennaStatus is only supported on revision A XM AI add-on boards.

1.2.2 Version 2.7.4.0 - VC Linux platform

- Bugfixes and documentation updates.

1.2.3 Version 2.7.3.0 - VC Linux platform

- Added functions CfgIn_getMinFrequencyThreshold and CfgIn_setMinFrequencyThreshold.
- Added restrictions on the usage of start-up triggers in combination with button configurations.

1.2.4 Version 2.7.2.0 - XM/XL Windows x86, x64 platform, VC Linux platform

- XM/XL, Windows: Fixed a bug introduced in 2.7.0.0 where the light sensor data could not be read on XM/XL.
- CCSettingsConsole: Fixed an issue where some commands did not work in Windows.

1.2.5 Version 2.7.1.0 - XM/XL Windows x86, x64 platform, VC Linux platform

- ccvideo: Fixed an issue where channels were not displayed correctly in the menu.

1.2.6 Version 2.7.0.0 - XM/XL Windows x86, x64 platform, VC Linux platform

- VC: Support for the VC platform (Linux).
- XM/XL: Support for the XM 2.0 platform (Windows/Linux).
- Added the following classes/functions for the VC platform:
  - Class CfgIn - Functions for managing configurable inputs
  - Class PWMOut - Functions for managing PWM outputs
  - About_getNrOfCfgInConnections
  - About_getNrOfPWMOutConnections
  - About_getNrOfButtons
1.2 Changelog

- About_getNrOfButtons
- Config_getButtonFunction
- Config_setButtonFunction

• Added the following functions for all platforms:
  - About_getUserEepromData
  - About_setUserEepromData

• Known issues:
  - XA/XS: Same as 2.4.7.0 release
  - XM/XL: Same as 2.5.0.0 release
  - VC: -

1.2.7 Version 2.6.2.0 - XM/XL Windows x86, x64 platform

• XM/XL: Fix for an issue with the function Video_getActiveChannel in x86 API on x64 OS.
• XM/XL: Support for Power_getCanOCDStatus and Power_getVideoOCDStatus with SS v1.2.0.0 or later.
• XM/XL: Support for optional integrated WLAN on CCpilot XL4.
• XM/XL: CCsettings: Improved Telematic GUI when not all interfaces are available.
• XM/XL: SnbService: Improved unit type descriptions: "CCpilot XM" instead of just "XM".
• XM/XL: CCsettings, CCvideo and TouchCalibrator: QT x86 libraries updated to v4.8.5.

• Known issues:
  - XA/XS: Same as 2.4.7.0 release
  - XM/XL: Same as 2.5.0.0 release

1.2.8 Version 2.6.1.0 - XM/XL Windows x86, x64 platform and XA/XS Linux platform

• XA/XS: Functions added: Video_getGraphicsOverlay and Video_setGraphicsOverlay.
• XM/XL: 64-bit support. Both x86 and x64 versions of the API can be installed at the same time on x64 systems.
• XM/XL: SnbService is now a selectable component in the installer.
1.2 Changelog


- Known issues:
  - XA/XS: Same as 2.4.7.0 release
  - XM/XL: Same as 2.5.0.0 release

1.2.9 Version 2.5.0.0 - XM/XL x86 platform

- CCAux2 API: Support for the XL platform. The XL platform is almost identical to the XM platform in terms of API support.

- CCAux2 API: Added SMART support for a second card used in XL (new functions Smart_getRemainingLifeTime2, Smart_getDeviceSerial2 and Smart_getInitialTime2).

- CCAux2 API: Bugfix for crash when incorrect filename was supplied to the functions FirmwareUpgrade_startFpgaUpgrade and FirmwareUpgrade_startFpgaVerification.

- CCvideo: Fixed a bug where selecting video 3 and 4 both selected video 3. The bug was only present in CCvideo v2.4.0.0 for XM and not in previous versions.

- CCvideo,CCAuxDrv: On the XL platform, video channel 3 and 4 are not available on both devices as on XM. Instead ch1 and ch2 can be selected for both devices.

  Only one channel can be shown at the same time per device and a device is on the XL platform equal to a physical connector.

- CCAux2CS: Added support for SMART interface for the C# dll

- CCAux2CS: Rewrote the following functions and changed their declaration to use System.String as output. The old overloads now return ERR_NOT_SUPPORTED:
  - About_getMainPCBSerial
  - About_getUnitSerial
  - About_getMainPCBArt
  - About_getMainManufacturingDate
  - About_getMainHWversion
  - About_getMainProdRev
  - About_getMainProdArtNr
  - About_getAddOnPCBSerial
  - About_getAddOnPCBArt
  - About_getAddOnManufacturingDate
  - About_getAddOnHWversion
  - FirmwareUpgrade_startFpgaUpgrade
  - FirmwareUpgrade_startFpgaVerification
1.2 Changelog

FirmwareUpgrade_startSSUpgrade
FirmwareUpgrade_startSSVerification
FirmwareUpgrade_startFrontUpgrade
FirmwareUpgrade_startFrontVerification
Video_takeSnapshot

• Known issues:
  – Some API functions are missing from ccsettingsconsole and CCAux2CS.

1.2.10 Version 2.4.7.0 - XM Linux platform

• XM: Improved fault-handling in function registerControlledSuspendOrShutDown()
• Known issues:
  – Same as 2.4.6.0 release

1.2.11 Version 2.4.6.0 - XA/XS platform

• XA/XS: Improve initialization of video channels 3/4
• XA/XS: Prevent scrolling when changing between video channels 3/4
• Calling Buzzer_buzzze no longer leaks memory
• Known issues:
  – Same as 2.4.0.0 release (minus Buzzer_buzzze memory leak)

1.2.12 Version 2.4.2.0 - XA/XS platform

• XA/XS: Config_get/setRS485Mode now uses settings file for intermediate storage
• Known issues:
  – Same as 2.4.0.0 release

1.2.13 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 compatibility. 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

• Known issues:
  
  – XA/XS: 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.

  – XA/XS: CCSettings - Advanced: After Firmware update, the shutdown button does not work. Workaround: Turn off power to the device.

  – Some info/functions are missing from ccsettingsconsole

  – XA/XS: About_hasOsBooted can return true even when not all drivers have not been loaded (API)

  – XA/XS: Calling Buzzer_buzzze in non-blocking mode leaks memory
1.2 Changelog

1.2.14 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

1.2.15 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.
  - Some info/functions are missing from ccsettingsconsole
  - About_hasOsBooted can return true even when not all drivers have not been loaded (API)
  - Calling Buzzer_buzze in non-blocking mode leaks memory
  - cctv: Rightclick (long press) menu not appearing consistently
  - 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
  - cctv: Selecting "Mirror image" does not have an effect
1.2 Changelog

1.2.16 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

1.2.17 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 compatibility 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/Font 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: 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:

CrossControl  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

- BatteryTimerType ........................................ 206
- BuzzerSetup .............................................. 207
- FpgaLedTimingType ..................................... 208
- LedColorMixType ........................................ 209
- LedTimingType ........................................... 209
- received_video ........................................... 210
- TimerType ................................................. 210
- UpgradeStatus ............................................ 212
- version_info ............................................. 212
- video_dec_command ................................... 213
Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

- IncludeFiles/About.h ........................................ 214
- IncludeFiles/Adc.h ........................................... 216
- IncludeFiles/AuxVersion.h .................................. 217
- IncludeFiles/Backlight.h .................................... 218
- IncludeFiles/Battery.h ....................................... 219
- IncludeFiles/Buzzer.h ....................................... 221
- IncludeFiles/CanSetting.h ................................... 222
- IncludeFiles/CCAuxErrors.h .................................. 222
- IncludeFiles/CCAuxTypes.h ................................... 223
- IncludeFiles/CCPlatform.h ................................... 225
- IncludeFiles/CfgIn.h ........................................ 225
- IncludeFiles/Config.h ....................................... 226
- IncludeFiles/Diagnostic.h .................................... 229
- IncludeFiles/DiagnosticCodes.h .............................. 230
- IncludeFiles/DigIO.h ......................................... 231
- IncludeFiles/FirmwareUpgrade.h .............................. 231
- IncludeFiles/FrontLED.h ..................................... 232
- IncludeFiles/Lightsensor.h ................................... 234
- IncludeFiles/Power.h ......................................... 235
- IncludeFiles/PowerMgr.h ...................................... 236
- IncludeFiles/PWMOut.h ....................................... 237
- IncludeFiles/Smart.h ......................................... 238
- IncludeFiles/Telematics.h .................................... 239
- IncludeFiles/TouchScreen.h ................................... 240
- IncludeFiles/TouchScreenCalib.h ............................. 241
- IncludeFiles/Video.h ......................................... 242
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 * ABOUHANDLE
- typedef void * ADCHANDLE
- typedef void * AUXVERSIONHANDLE
- typedef void * BACKLIGHHANDLE
- typedef void * BATTERYHANDLE
- typedef void * BUZZERHANDLE
- typedef void * CANSETTINGHANDLE
- typedef struct version_info VersionType
- typedef void * CFGINHANDLE
- 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 *PWMOUTHANDLE
• 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_12VID,
  VOLTAGE_5V, VOLTAGE_3V3, VOLTAGE_VTFT, VOLTAGE_5VSTB,
  VOLTAGE_1V9, VOLTAGE_1V8, VOLTAGE_1V5, VOLTAGE_1V2,
  VOLTAGE_1V05, VOLTAGE_1V0, VOLTAGE_0V9, VOLTAGE_VREF_INT,
  VOLTAGE_24V_BACKUP, VOLTAGE_2V5, VOLTAGE_1V1, VOLTAGE_1V3_PER,
  VOLTAGE_1V3_VDDA, VOLTAGE_3V3STBY, VOLTAGE_VPMIC, VOLTAGE_VMAIN }
• enum LightSensorOperationRange { RangeStandard = 0, RangeExtended = 1 }
• enum LightSensorSamplingMode { SamplingModeStandard = 0, SamplingModeExtended, SamplingModeAuto }
• enum CCStatus { Disabled = 0, Enabled = 1 }
• enum eErr {
  ERR_SUCCESS = 0, ERR_OPEN_FAILED = 1, ERR_NOT_SUPPORTED = 2, ERR_UNKNOWN_FEATURE = 3,
  ERR_DATATYPE_MISMATCH = 4, ERR_CODE_NOT_EXIST = 5, ERR_BUFFER_SIZE = 6, ERR_IOCTL_FAILED = 7,
  ERR_INVALID_DATA = 8, ERR_INVALID_PARAMETER = 9, ERR_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, ERR_INVALID_D_STARTUP_TRIGGER = 34

• 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_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,
CAN_Button_Activity = 5, CAN_OnOff_Activity = 6, CAN_Button_OnOff_Activity = 7,
CI_Button_Activity = 9, CI_OnOff_Activity = 10, CI_Button_OnOff_Activity = 11,
CI_CAN_Button_Activity = 13, CI_CAN_OnOff_Activity = 14, All_Events = 15,
Last_trigger_conf }
• 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 CfgInModeEnum {
    CFGIN_NOT_IN_USE = 0, CFGIN_HI_SWITCH, CFGIN_LOW_SWITCH,
    CFGIN_VOLTAGE_2V5, CFGIN_VOLTAGE_5V, CFGIN_VOLTAGE_10V,
    CFGIN_VOLTAGE_32V, CFGIN_DIGITAL_PD_5V, CFGIN_RESISTANCE,
    CFGIN_FREQ_FLOATING, CFGIN_FREQ_PULLUP,
    CFGIN_FREQ_PULLDOWN, CFGIN_VOLTAGE_500, CFGIN_CURRENT_4_20,
    CFGIN_CURRENT_10V, CFGIN_CURRENT_32V, CFGIN_DIGITAL_PD_5V,
    CFGIN_DIGITAL_PD_10V, CFGIN_DIGITAL_PD_32V
}

• enum ButtonConfigEnum {
    BUTTON_ONLY_MP_ACTION = 0x00, BUTTON_AS_STARTUP_TRAIG =
    0x02, BUTTON_AS_ACTION_TRIG = 0x04, BUTTON_AS_ACTION_STARTUP_TRIG = 0x06,
    BUTTON_AS_BACKLIGHT_DECREASE = 0x08, BUTTON_AS_BACKLIGHT_INCREASE = 0x0C,
    BUTTON_AS_BACKLIGHT_INCREASE_STARTUP_TRIG = 0x0E
}

• enum ConfigOnOffTriggerMode { CONFIG_ONOFF_EDGE_TRIGGER = 0,
    CONFIG_ONOFF_LEVEL_TRIGGER }

• enum startupReasonCodes {
    startupReasonCodeUndefined = 0x0000, startupReasonCodeButtonPress = 0x0055,
    startupReasonCodeExtCtrl = 0x00AA, startupReasonCodeMPRestart = 0x00F0,
    startupReasonCodePowerOnStartup = 0x000F, startupReasonCodeCanActivity =
    0x003c, startupReasonCodeCIActivity = 0x00c3, startupReasonAlwaysStart =
    0x00e1, startupReasonUnknownTrigger = 0x001e
}

• enum shutDownReasonCodes { shutdownReasonCodeNoError = 0x001F }

• enum hwErrorStatusCodes { enCodeNoErr = 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_NEXT_BOOT = 1,
    MOUSE_NOW = 2, TOUCH_NOW = 3 }

• enum TSAdvancedSettingsParameter {
    TS_RIGHT_CLICK_TIME = 0, TS_LOW_LEVEL = 1, TS_UNTOUCHLEVEL=
5.1 CrossControl Namespace Reference

EL = 2, TS_DEBOUNCE_TIME = 3,
TS_DEBOUNCE_TIMEOUT_TIME = 4, TS_DOUBLECLICK_MAX_CLICK_TIME = 5, TS_DOUBLE_CLICK_TIME = 6, TS_MAX_RIGHTCLICK_DISTANCE = 7,
TS_USE_DEJITTER = 8, TS_CALIBRATION_WIDTH = 9, TS_CALIBRATION_MEASUREMENTS = 10, TS_RESTORE_DEFAULT_SETTINGS = 11,
TS_TCHAUTOCAL = 12

• enum CalibrationModeSettings {
    MODE_UNKNOWN = 0, MODE_NORMAL = 1, MODE_CALIBRATION_5P = 2, MODE_CALIBRATION_9P = 3,
    MODE_CALIBRATION_13P = 4
}

• enum CalibrationConfigParam {
    CONFIG_CALIBRATION_WITH = 0, CONFIG_CALIBRATION_MEASUREMENTS = 1, CONFIG_5P_CALIBRATION_POINT_BORDER = 2, CONFIG_13P_CALIBRATION_POINT_BORDER = 3,
    CONFIG_13P_CALIBRATION_TRANSITION_MIN = 4, CONFIG_13P_CALIBRATION_TRANSITION_MAX = 5
}

Functions

• EXTERN_C CCAUXDLL_API
  ABOUTHANDLE
  CCAUXDLL_CALLING_CONV GetAbout (void)

• EXTERN_C CCAUXDLL_API void
  CCAUXDLL_CALLING_CONV About_release (ABOUTHANDLE)

• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getMainPCBSerial (ABOUTHANDLE, char *buff, int len)

• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getUnitSerial (ABOUTHANDLE, char *buff, int len)

• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getMainPCBArt (ABOUTHANDLE, char *buff, int length)

• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getMainManufacturingDate (ABOUTHANDLE, char *buff, int len)

• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getMainHWversion (ABOUTHANDLE, char *buff, int len)

• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getMainProdRev (ABOUTHANDLE, char *buff, int len)

• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getMainProdArtNr (ABOUTHANDLE, char *buff, int len)

• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getNrOfETHConnections (ABOUTHANDLE, unsigned char *NrOfConnections)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getNrOfCANConnections (ABOUTHANDLE, unsigned char *NrOfConnections)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getNrOfVideoConnections (ABOUTHANDLE, unsigned char *NrOfConnections)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getNrOfUSBConnections (ABOUTHANDLE, unsigned char *NrOfConnections)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getNrOfSerialConnections (ABOUTHANDLE, unsigned char *NrOfConnections)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getNrOfDigIOConnections (ABOUTHANDLE, unsigned char *NrOfConnections)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getIsDisplayAvailable (ABOUTHANDLE, bool *available)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getIsTouchScreenAvailable (ABOUTHANDLE, bool *available)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getDisplayResolution (ABOUTHANDLE, char *buff, int len)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getAddOnPCBSerial (ABOUTHANDLE, char *buff, int len)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getAddOnPCBArt (ABOUTHANDLE, char *buff, int len)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getAddOnManufacturingDate (ABOUTHANDLE, char *buff, int len)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getAddOnHWversion (ABOUTHANDLE, char *buff, int len)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getIsWLANMounted (ABOUTHANDLE, bool *mounted)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getIsGPSMounted (ABOUTHANDLE, bool *mounted)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getIsGPRSMounted (ABOUTHANDLE, bool *mounted)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getIsBTMounted (ABOUTHANDLE, bool *mounted)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getFrontPcbRev (ABOUTHANDLE,
  unsigned char *major, unsigned char *minor)

• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getIsIOExpanderMounted (ABOUTHANDLE,
  bool *mounted)

• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getIOExpanderValue (ABOUTHANDLE,
  unsigned short *value)

• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_hasOsBooted (ABOUTHANDLE, bool
  *bootComplete)

• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getIsAnybusMounted (ABOUTHANDLE,
  bool *mounted)

• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getNrOfCfgInConnections (ABOUTHANDLE,
  unsigned char *NrOfConnections)

• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getNrOfPWMOutConnections (ABOUTHANDLE,
  unsigned char *NrOfConnections)

• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getNrOfButtons (ABOUTHANDLE, int
  *numbuttons)

• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getUserEepromData (ABOUTHANDLE,
  char *buff, unsigned short length)

• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_setUserEepromData (ABOUTHANDLE,
  unsigned short startpos, const char *buff, unsigned short length)

• EXTERN_C CCAUXDLL_API
  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
  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)
5.1 CrossControl Namespace Reference

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

- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV AuxVersion_getFrontVersion (AUXVERSIONHANDLE, unsigned char *major, unsigned char *minor, unsigned char *release, unsigned char *build)

- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV AuxVersion_getCCAuxVersion (AUXVERSIONHANDLE, unsigned char *major, unsigned char *minor, unsigned char *release, unsigned char *build)

- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV AuxVersion_getOSVersion (AUXVERSIONHANDLE, unsigned char *major, unsigned char *minor, unsigned char *release, unsigned char *build)

- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV AuxVersion_getCCAuxDrvVersion (AUXVERSIONHANDLE, unsigned char *major, unsigned char *minor, unsigned char *release, unsigned char *build)

- EXTERN_C CCAUXDLL_API BACKLIGHTHANDLE
  CCAUXDLL_CALLING_CONV GetBacklight (void)

- EXTERN_C CCAUXDLL_API void
  CCAUXDLL_CALLING_CONV Backlight_release (BACKLIGHTHANDLE)

- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Backlight_getIntensity (BACKLIGHTHANDLE, unsigned char *intensity)

- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Backlight_setIntensity (BACKLIGHTHANDLE, unsigned char intensity)

- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Backlight_getStatus (BACKLIGHTHANDLE, unsigned char *status)

- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Backlight_getHWStatus (BACKLIGHTHANDLE, bool *status)

- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Backlight_startAutomaticBL (BACKLIGHTHANDLE)

- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Backlight_stopAutomaticBL (BACKLIGHTHANDLE)

- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Backlight_getAutomaticBLStatus (BACKLIGHTHANDLE, unsigned char *status)

- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Backlight_setAutomaticBLParams (BACKLIGHTHANDLE, bool bSoftTransitions)
5.1 CrossControl Namespace Reference

- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Backlight_getAutomaticBLParams (BACKLIGHTHANDLE, bool *bSoftTransitions, double *k)

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

- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Backlight_getAutomaticBLFilter (BACKLIGHTHANDLE, unsigned long *averageWndSize, unsigned long *rejectWndSize, unsigned long *rejectDeltaInLux, LightSensorSamplingMode *mode)

- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Backlight_getLedDimming (BACKLIGHTHANDLE, CCStatus *status)

- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Backlight_setLedDimming (BACKLIGHTHANDLE, CCStatus status)

- EXTERN_C CCAUXDLL_API BATTERYHANDLE
  CCAUXDLL_CALLING_CONV GetBattery (void)

- EXTERN_C CCAUXDLL_API void
  CCAUXDLL_CALLING_CONV Battery_release (BATTERYHANDLE)

- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Battery_isBatteryPresent (BATTERYHANDLE, bool *batteryIsPresent)

- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Battery_getBatteryVoltageStatus (BATTERYHANDLE, unsigned char *batteryVoltagePercent)

- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Battery_getBatteryChargingStatus (BATTERYHANDLE, ChargingStatus *status)

- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Battery_getPowerSource (BATTERYHANDLE, PowerSource *status)

- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Battery_getBatteryTemp (BATTERYHANDLE, signed short *temperature)

- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Battery_getHwErrorStatus (BATTERYHANDLE, ErrorStatus *errorCode)

- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Battery_getTimer (BATTERYHANDLE, 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)
• 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_buzz (BUZZERHANDLE, int time, bool blocking)
• EXTERN_C CCAUXDLL_API
  CANSETTINGHANDLE
  CCAUXDLL_CALLING_CONV GetCanSetting (void)
• EXTERN_C CCAUXDLL_API void
  CCAUXDLL_CALLING_CONV CanSetting_release (CANSETTINGHANDLE)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV CanSetting_getBaudrate (CANSETTINGHANDLE, unsigned char net, unsigned short *baudrate)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV CanSetting_getFrameType (CANSETTINGHANDLE, unsigned char net, CanFrameType *frameType)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV CanSetting_setBaudrate (CANSETTINGHANDLE, unsigned char net, unsigned short baudrate)
5.1 CrossControl Namespace Reference

- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV CanSetting_setFrameType (CANSETTINGHANDLE, unsigned char net, CanFrameType frameType)
- EXTERN_C CCAUXDLL_API char
  const *CCAUXDLL_CALLING_CONV GetErrorStringA (eErr errCode)
- EXTERN_C CCAUXDLL_API wchar_t
  const *CCAUXDLL_CALLING_CONV GetErrorStringW (eErr errCode)
- EXTERN_C CCAUXDLL_API CFGINHANDLE
  CCAUXDLL_CALLING_CONV GetCfgIn (void)
- EXTERN_C CCAUXDLL_API void
  CCAUXDLL_CALLING_CONV CfgIn_release (CFGINHANDLE)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV CfgIn_setCfgInMode (CFGINHANDLE, unsigned char channel, CfgInModeEnum set_mode)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV CfgIn_getCfgInMode (CFGINHANDLE, unsigned char channel, CfgInModeEnum *get_mode)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV CfgIn_getValue (CFGINHANDLE, unsigned char channel, unsigned short *sample_value)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV CfgIn_getPwmValue (CFGINHANDLE, unsigned char channel, float *frequency, unsigned char *duty_cycle)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV CfgIn_getFrequencyValue (CFGINHANDLE, unsigned char channel, float *frequency)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV CfgIn_getMinFrequencyThreshold (CFGINHANDLE, unsigned char channel, float *frequency)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV CfgIn_setMinFrequencyThreshold (CFGINHANDLE, unsigned char channel, float frequency)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV CfgIn_setFrequencyFilterLevel (CFGINHANDLE, unsigned char level)
- 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_getButtonFunction (CONFIGHANDLE, unsigned char button_number, ButtonConfigEnum *button_config)
• 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_setButtonFunction (CONFIGHANDLE, unsigned char button_number, ButtonConfigEnum button_config)
• 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 eErr
  CCAUXDLL_CALLING_CONV Config_setOnOffTriggerMode (CONFIGHANDLE, ConfigOnOffTriggerMode mode)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Config_getOnOffTriggerMode (CONFIGHANDLE, ConfigOnOffTriggerMode *mode)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Config_getOnOffSignalState (CONFIGHANDLE, CCStatus *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_getSSSTemp (DIAGNOSTICHANDLE, signed short *temperature)
• EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Diagnostic_getPCBTemp (DIAGNOSTICHANDLE, signed short *temperature)
• EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Diagnostic_getPMTemp (DIAGNOSTICHANDLE, unsigned char index, signed short *temperature, JidaSensorType *jst)
• EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Diagnostic_getStartupReason (DIAGNOSTICHANDLE, unsigned short *reason)
• EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Diagnostic_getShutDownReason (DIAGNOSTICHANDLE, unsigned short *reason)
• EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Diagnostic_getHwErrorStatus (DIAGNOSTICHANDLE, unsigned short *errorCode)
• EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Diagnostic_getTimer (DIAGNOSTICHANDLE, TimerType *times)
• EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Diagnostic_getMinMaxTemp (DIAGNOSTICHANDLE, signed short *minTemp, signed short *maxTemp)
• EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Diagnostic_getPowerCycles (DIAGNOSTICHANDLE, unsigned short *powerCycles)
• EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Diagnostic_clearHwErrorStatus (DIAGNOSTICHANDLE)
• EXTERN_C CCAUXDLL_API char const *CCAUXDLL_CALLING_CONV GetHwErrorStatusStringA (unsigned short errCode)
• EXTERN_C CCAUXDLL_API wchar_t const *CCAUXDLL_CALLING_CONV GetHwErrorStatusStringW (unsigned short errCode)
• EXTERN_C CCAUXDLL_API char const *CCAUXDLL_CALLING_CONV GetStartupReasonStringA (unsigned short code)
• EXTERN_C CCAUXDLL_API wchar_t const *CCAUXDLL_CALLING_CONV GetStartupReasonStringW (unsigned short code)
• EXTERN_C CCAUXDLL_API DIGIOHANDLE CCAUXDLL_CALLING_CONV GetDigIO (void)
5.1 CrossControl Namespace Reference

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

- 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)
5.1 CrossControl Namespace Reference

- EXTERN_C CCAUXDLL_API void
  CCAUXDLL_CALLING_CONV Lightsensor_release (LIGHTSENSORHANDLE)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Lightsensor_getIlluminance (LIGHTSENSORHANDLE, unsigned short *value)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Lightsensor_getIlluminance2 (LIGHTSENSORHANDLE, unsigned short *value, unsigned char *ch0, unsigned char *ch1)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Lightsensor_getAverageIlluminance (LIGHTSENSORHANDLE, unsigned short *value)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Lightsensor_startAverageCalc (LIGHTSENSORHANDLE, unsigned long averageWndSize, unsigned long rejectWndSize, unsigned long rejectDeltaInLux, LightSensorSamplingMode mode)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Lightsensor_stopAverageCalc (LIGHTSENSORHANDLE)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Lightsensor_getOperatingRange (LIGHTSENSORHANDLE, LightSensorOperationRange *range)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Lightsensor_setOperatingRange (LIGHTSENSORHANDLE, LightSensorOperationRange range)
- EXTERN_C CCAUXDLL_API
  POWERHANDLE
  CCAUXDLL_CALLING_CONV GetPower (void)
- EXTERN_C CCAUXDLL_API void
  CCAUXDLL_CALLING_CONV Power_release (POWERHANDLE)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Power_getBLPowerStatus (POWERHANDLE, CCStatus *status)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Power_getCanPowerStatus (POWERHANDLE, CCStatus *status)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Power_getVideoPowerStatus (POWERHANDLE, unsigned char *videoStatus)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Power_getExtFanPowerStatus (POWERHANDLE, CCStatus *status)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Power_getButtonPowerTransitionStatus (POWERHANDLE, ButtonPowerTransitionStatus *status)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Power_getVideoOCDStatus (POWERHANDLE, OCDStatus *status)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Power_getCanOCDStatus (POWERHANDL-
  E, OCDStatus *status)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Power_setBLPowerStatus (POWERHANDL-
  E, CCStatus status)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Power_setCanPowerStatus (POWERHANDL-
  E, CCStatus status)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Power_setVideoPowerStatus (POWERHANDL-
  E, unsigned char status)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Power_setExtFanPowerStatus (POWERHANDL-
  E, CCStatus status)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Power_ackPowerRequest (POWERHANDL-
  E)
• 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_registerControlledSuspendOrShut-
  Down (POWERMGRHANDLE, PowerMgrConf conf)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Power_mgr_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
  PWMOUTHANDLE
  CCAUXDLL_CALLING_CONV GetPWMOut (void)
• EXTERN_C CCAUXDLL_API void
  CCAUXDLL_CALLING_CONV PWMOut_release (PWMOUTHANDLE)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV PWMOut_setPWMOutputChannelDutyCycle (PWMOUTHANDLE, unsigned char channel, unsigned char duty_cycle)
5.1 CrossControl Namespace Reference

- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV PWMOut_setPWMOutputChannelFrequency
  (PWMOUTHANDLE, unsigned char channel, float frequency)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV PWMOut_getPWMOutputChannelDutyCycle
  (PWMOUTHANDLE, unsigned char channel, unsigned char *duty_cycle)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV PWMOut_getPWMOutputChannelFrequency
  (PWMOUTHANDLE, unsigned char channel, float *frequency)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV PWMOut_getPWMOutputStatus (PWMOUTHANDLE, unsigned char *status)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV PWMOut_setPWMOutOff (PWMOUTHANDLE, unsigned char channel)
- 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_getRemainingLifeTime2 (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_getDeviceSerial2 (SMARTHANDLE, char *buff, int len)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Smart_getInitialTime (SMARTHANDLE, time_t *time)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Smart_getInitialTime2 (SMARTHANDLE, time_t *time)
- EXTERN_C CCAUXDLL_API TELEMATICSHANDLE
  CCAUXDLL_CALLING_CONV GetTelematics (void)
- EXTERN_C CCAUXDLL_API void
  CCAUXDLL_CALLING_CONV Telematics_release (TELEMATICSHANDLE)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Telematics_getTelematicsAvailable (TELEMATICSHANDLE, CCStatus *status)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Telematics_getGPRSStatus (TELEMATICSHANDLE, CCStatus *status)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Telematics_getGPRSStartUpPowerStatus (TELEMATICSHANDLE, CCStatus *status)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Telematics_getWLANPowerStatus (TELEMATICSHANDLE, CCStatus *status)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Telematics_getWLANStartUpPowerStatus (TELEMATICSHANDLE, CCStatus *status)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Telematics_getBTPowerStatus (TELEMATICSHANDLE, CCStatus *status)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Telematics_getBTStartUpPowerStatus (TELEMATICSHANDLE, CCStatus *status)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Telematics_getGPSPowerStatus (TELEMATICSHANDLE, CCStatus *status)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Telematics_getGPSStartUpPowerStatus (TELEMATICSHANDLE, CCStatus *status)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Telematics_getGPSAntennaStatus (TELEMATICSHANDLE, CCStatus *status)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Telematics_setGPRSStatus (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)
5.1 CrossControl Namespace Reference

- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Telematics_setGPSStartUpPowerStatus (TELEMETRICSHANDLE, 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 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)
5.1 CrossControl Namespace Reference

- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV TouchScreenCalib_setConfigParam (TOUCHSCREENCALIBHANDLE, CalibrationConfigParam param, unsigned short value)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV TouchScreenCalib_autoSensorCalib (TOUCHSCREENCALIBHANDLE)
- 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)
5.1 CrossControl Namespace Reference

- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_getRawImage (VIDEOHANDLE, unsigned short *width, unsigned short *height, float *frameRate)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_getVideoArea (VIDEOHANDLE, unsigned short *topLeftX, unsigned short *topLeftY, unsigned short *bottomRightX, unsigned short *bottomRightY)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_getVideoStandard (VIDEOHANDLE, videoStandard *standard)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_getStatus (VIDEOHANDLE, unsigned char *status)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_setScaling (VIDEOHANDLE, float x, float y)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_getScaling (VIDEOHANDLE, float *x, float *y)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_activateSnapshot (VIDEOHANDLE, bool activate)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_takeSnapshot (VIDEOHANDLE, const char *path, bool bInterlaced)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_takeSnapshotRaw (VIDEOHANDLE, char *rawImgBuffer, unsigned long rawImgBufSize, bool bInterlaced)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_takeSnapshotBmp (VIDEOHANDLE, char **bmpBuffer, unsigned long *bmpBufSize, bool bInterlaced, bool bNTSCFormat)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_createBitmap (VIDEOHANDLE, char **bmpBuffer, unsigned long *bmpBufSize, const char *rawImgBuffer, unsigned long rawImgBufSize, bool bInterlaced, bool bNTSCFormat)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_freeBmpBuffer (VIDEOHANDLE, char *bmpBuffer)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_minimize (VIDEOHANDLE)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_restore (VIDEOHANDLE)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_setDecoderReg (VIDEOHANDLE, unsigned char decoderRegister, unsigned char *registerValue)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_getDecoderReg (VIDEOHANDLE, unsigned char decoderRegister, unsigned char *registerValue)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_setCropping (VIDEOHANDLE, unsigned char top, unsigned char left, unsigned char bottom, unsigned char right)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_getCropping (VIDEOHANDLE, unsigned char *top, unsigned char *left, unsigned char *bottom, unsigned char *right)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_showFrame (VIDEOHANDLE)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_setGraphicsOverlay (VIDEOHANDLE, CCStatus mode)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_getGraphicsOverlay (VIDEOHANDLE, CCStatus *mode)

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* BACKLIGHHANDLE

5.1.1.7 typedef void* BATTERYHANDLE

5.1.1.8 typedef void* BUZZERHANDLE
5.1 CrossControl Namespace Reference

5.1.1.9 typedef void* CANSETTINGHANDLE
5.1.1.10 typedef void* CFGINHANDLE
5.1.1.11 typedef void* CONFIGHANDLE
5.1.1.12 typedef void* DIAGNOSTICHANDLE
5.1.1.13 typedef void* DIGIOHANDLE
5.1.1.14 typedef void* FIRMWAREUPGHANDLE
5.1.1.15 typedef void* FRONTLEDHANDLE
5.1.1.16 typedef void* LIGHTSENSORHANDLE
5.1.1.17 typedef void* POWERHANDLE
5.1.1.18 typedef void* POWERMGRHANDLE
5.1.1.19 typedef void* PWMOUTHANDLE
5.1.1.20 typedef void* SMARTHANDLE
5.1.1.21 typedef void* TELEMATICSHANDLE
5.1.1.22 typedef void* TOUCHSCREENCALIBHANDLE
5.1.1.23 typedef void* TOUCHSCREENHANDLE
5.1.1.24 typedef struct version_info VersionType
5.1.1.25 typedef void* VIDEOHANDLE

5.1.2 Enumeration Type Documentation

5.1.2.1 enum ButtonConfigEnum

Enumeration of Button Configuration (bitfield representation)

Enumerator

**BUTTON_ONLY_MP_ACTION**

**BUTTON_AS_STARTUP_TRIG**  Buttons are only read by Main Processor, i.e. normal button action which is handled in application space

**BUTTON_AS_ACTION_TRIG**  Set button to be used as startup trigger, in addition to MP application event

**BUTTON_AS_ACTION_STARTUP_TRIG**  Set button to trigger suspend, shutdown or hard shutdown actions
5.1 CrossControl Namespace Reference

BUTTON_AS_BACKLIGHT_DECREASE  Set button to trigger startup, suspend, shutdown or hard shutdown actions

BUTTON_AS_BACKLIGHT_DECR_STARTUP_TRIG  Set button to act as backlight decrease button

BUTTON_AS_BACKLIGHT_INCREASE  Set button to act as backlight decrease and startup trigger

BUTTON_AS_BACKLIGHT_INCR_STARTUP_TRIG  Set button to act as backlight increase button

5.1.2.2 enum ButtonPowerTransitionStatus

Current status for front panel button and on/off signal. If any of them generate a suspend or shutdown event, it may also be read, briefly. When the button/signal is released, typically BPTS_Suspend or BPTS_ShutDown follows. Note: Do not rely on getting BPTS_ShutDown or BPTS_Suspend from user applications. The system is likely to start shutting down before the status can be read. Instead, use the PowerMgr class for handling system shutdown/suspend events.

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.

5.1.2.3 enum CalibrationConfigParam

Touch screen calibration parameters

Enumerator

CONFIG_CALIBRATION_WITH

CONFIG_CALIBRATION_MEASUREMENTS  Accepted error value when calibrating.
5.1 CrossControl Namespace Reference

*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.

5.1.2.4 enum CalibrationModeSettings

Touch screen calibration modes

Enumerator

*MODE_UNKNOWN*  Unknown mode.

*MODE_NORMAL*  Normal operation mode.

*MODE_CALIBRATION_5P*  Calibration with 5 points mode.

*MODE_CALIBRATION_9P*  Calibration with 9 points mode.

5.1.2.5 enum CanFrameType

Can frame type settings

Enumerator

*FrameStandard*  

*FrameExtended*  

*FrameStandardExtended*  

5.1.2.6 enum CCAuxColor

Enumeration of standard colors

Enumerator

*RED*  RGB 0xF, 0x0, 0x0

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

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

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

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

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

*UNDEFINED_COLOR*  RGB 0xF, 0xF, 0x0

Returns if color is not a standard color
5.1 CrossControl Namespace Reference

5.1.2.7 enum CCStatus

Enable/disable enumeration

Enum

*Disabled*
The setting is disabled or turned off

*Enabled*

5.1.2.8 enum CfgInModeEnum

Enumeration of ConfigurableInput modes

Enum

*CFGIN_NOT_IN_USE*
Disable configurable input measurement

*CFGIN_HI_SWITCH*
Read digital input value with CfgIn_getValue

*CFGIN_LOW_SWITCH*
Read digital input value with CfgIn_getValue

*CFGIN_VOLTAGE_2V5*
Read digital input value with CfgIn_getValue

*CFGIN_VOLTAGE_5V*
Read voltage input value with CfgIn_getValue, 2.5V range

*CFGIN_RESISTANCE*
Read voltage input value with CfgIn_getValue, 5V range

*CFGIN_FREQ_FLOATING*
Read resistance input value with CfgIn_getValue, ports 1-4 only

*CFGIN_FREQ_PULLUP*
Read frequency value with CfgIn_getPwmValue

*CFGIN_FREQ_PULLDOWN*
Read frequency value with CfgIn_getPwmValue

*CFGIN_RESISTANCE_500*
Read frequency value with CfgIn_getPwmValue

*CFGIN_CURRENT_4_20*
Read resistance input value with CfgIn_getValue, 0-500Ohm range, VA only, ports 1-4 only

*CFGIN_VOLTAGE_10V*
Read current input value with CfgIn_getValuerange 4-20 mA, VA only.

*CFGIN_VOLTAGE_32V*
Read voltage input value with CfgIn_getValue, 10V range, VA only

*CFGIN_DIGITAL_PD_5V*
Read voltage input value with CfgIn_getValue, 32V range, VA only

*CFGIN_DIGITAL_PD_10V*
Read digital input value with CfgIn_getValue, pull-down, 5V range, 2.5V threshold, VA only

*CFGIN_DIGITAL_PD_32V*
Read digital input value with CfgIn_getValue, pull-down, 10V range, 5V threshold, VA only

*CFGIN_DIGITAL_F_5V*
Read digital input value with CfgIn_getValue, pull-down, 32V range, 10V threshold, VA only

*CFGIN_DIGITAL_F_10V*
Read digital input value with CfgIn_getValue, floating, 5V range, 2.5V threshold, VA only
5.1 CrossControl Namespace Reference

**CFGIN_DIGITAL_F_32V**  Read digital input value with CfgIn_getValue, floating, 10V range, 5V threshold, VA only

**CFGIN_DIGITAL_PU_5V**  Read digital input value with CfgIn_getValue, floating, 32V range, 10V threshold, VA only

**CFGIN_DIGITAL_PU_10V**  Read digital input value with CfgIn_getValue, pull-up, 5V range, 2.5V threshold, VA only, ports 5-8 only

**CFGIN_DIGITAL_PU_32V**  Read digital input value with CfgIn_getValue, pull-up, 10V range, 5V threshold, VA only, ports 5-8 only

**CFGIN_FREQ_PD_5V**  Read digital input value with CfgIn_getValue, pull-up, 32V range, 10V threshold, VA only, ports 5-8 only

**CFGIN_FREQ_PD_10V**  Read frequency value with CfgIn_getFrequencyValue, pull-down, 5V range, 2.5V threshold, VA only

**CFGIN_FREQ_PD_32V**  Read frequency value with CfgIn_getFrequencyValue, pull-down, 10V range, 5V threshold, VA only

**CFGIN_FREQ_F_5V**  Read frequency value with CfgIn_getFrequencyValue, pull-down, 32V range, 10V threshold, VA only

**CFGIN_FREQ_F_10V**  Read frequency value with CfgIn_getFrequencyValue, floating, 5V range, 2.5V threshold, VA only

**CFGIN_FREQ_F_32V**  Read frequency value with CfgIn_getFrequencyValue, floating, 10V range, 5V threshold, VA only

**CFGIN_FREQ_PU_5V**  Read frequency value with CfgIn_getFrequencyValue, floating, 32V range, 10V threshold, VA only

**CFGIN_FREQ_PU_10V**  Read frequency value with CfgIn_getFrequencyValue, pull-up, 5V range, 2.5V threshold, VA only, ports 5-8 only

**CFGIN_FREQ_PU_32V**  Read frequency value with CfgIn_getFrequencyValue, pull-up, 10V range, 5V threshold, VA only, ports 5-8 only

5.1.2.9 enum ChargingStatus

Current charging status of the battery.

**Enumerator**

- **ChargingStatus_NoCharge**  The battery is not being charged. System is running on battery power.
- **ChargingStatus_Charging**  The battery is currently being charged
- **ChargingStatus_FullyCharged**  The battery is fully charged
- **ChargingStatus_TempLow**  The temperature is too low to allow the battery to be charged
- **ChargingStatus_TempHigh**  The temperature is too high to allow the battery to be charged
- **ChargingStatus_Unknown**  There was an error determining the charging status
5.1 CrossControl Namespace Reference

5.1.2.10 enum ConfigOnOffTriggerMode

Enumeration of OnOff/Ignition/KeySwitch signal trigger modes

Enumerator

- **CONFIG_ONOFF_EDGE_TRIGGER**
- **CONFIG_ONOFF_LEVEL_TRIGGER** OnOff/Ignition/KeySwitch signal trigger on signal edge (default)
- OnOff/Ignition/KeySwitch signal trigger on signal level

5.1.2.11 enum DeInterlaceMode

The available deinterlace modes

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

5.1.2.12 enum eErr

Error code enumeration

Enumerator

- **ERR_SUCCESS**
- **ERR_OPEN_FAILED** Success
- **ERR_NOT_SUPPORTED** Open failed
- **ERR_UNKNOWN_FEATURE** Not supported
- **ERR_DATATYPE_MISMATCH** Unknown feature
- **ERR_CODE_NOT_EXIST** Datatype mismatch
- **ERR_BUFFER_SIZE** Code doesn’t exist
- **ERR_IOCTL_FAILED** Buffer size error
- **ERR_INVALID_DATA** IoCtrl operation failed
- **ERR_INVALID_PARAMETER** Invalid data
- **ERR_CREATE_THREAD** Invalid parameter
- **ERR_IN_PROGRESS** Failed to create thread
- **ERR_CHECKSUM** Operation in progress
- **ERR_INIT_FAILED** Checksum error
- **ERR_VERIFY_FAILED** Initialization failed
- **ERR_DEVICE_READ_DATA_FAILED** Failed to verify
5.1 CrossControl Namespace Reference

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
ERR_INVALID_STARTUP_TRIGGER Failed to join thread

5.1.2.13 enum ErrorStatus

Error status.

Enumerator

ErrorStatus_NoError
ErrorStatus_ThermistorTempSensor
ErrorStatus_SecondaryTempSensor
ErrorStatus_ChargeFail
ErrorStatus_Overcurrent
ErrorStatus_Init

5.1.2.14 enum hwErrorStatusCodes

The error codes returned by getHWErrorStatus.

Enumerator

errCodeNoErr
5.1 CrossControl Namespace Reference

5.1.2.15 enum JidaSensorType

Jida temperature sensor types

Enumerator

    TEMP_CPU
    TEMP_BOX
    TEMP_ENV
    TEMP_BOARD
    TEMP_BACKPLANE
    TEMP_CHIPSETS
    TEMP_VIDEO
    TEMP_OTHER

5.1.2.16 enum LightSensorOperationRange

Light sensor operation ranges.

Enumerator

    RangeStandard
    RangeExtended  Light sensor operation range standard

5.1.2.17 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.

5.1.2.18 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 CrossControl Namespace Reference

5.1.2.19 enum PowerAction

Button and on/off signal actions.

Enumerator

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

5.1.2.20 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.

5.1.2.21 enum PowerMgrStatus

Enumerator

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

5.1.2.22 enum PowerSource

Current power source of the computer.

Enumerator

- **PowerSource_Battery**
- **PowerSource_ExternalPower**
5.1 CrossControl Namespace Reference

5.1.2.23 enum RS4XXPort

Enumeration of RS4XX ports (1-4)

Enumerator

   RS4XXPort1
   RS4XXPort2
   RS4XXPort3
   RS4XXPort4

5.1.2.24 enum shutDownReasonCodes

The shutdown codes returned by getShutDownReason.

Enumerator

   shutdownReasonCodeNoError

5.1.2.25 enum startupReasonCodes

The restart codes returned by getStartupReason.

Enumerator

   startupReasonCodeUndefined
   startupReasonCodeExtCtrl      The system was started by front panel button press
   startupReasonCodeMPRestart   The system was started by the external control signal
   startupReasonCodePowerOnStart The system was restarted by OS request
   startupReasonCodeCanActivity  The system was started due to the PowerOn-Startup setting
   startupReasonCodeCIAActivity  The system was started due to activity on the Can bus (CCpilot VC family)
   startupReasonAlwaysStart     The system was started due to activity on the configurable input signals (CCpilot VC family)
   startupReasonUnknownTrigger The system was prevented to shutdown, since it is not allowed on this unit type.

5.1.2.26 enum TouchScreenModeSettings

Touch screen USB profile settings
5.1 CrossControl Namespace Reference

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.

5.1.2.27 enum TriggerConf

Trigger configuration enumeration. Valid settings for enabling of front button and external on/off signal. For platforms XM, XL and XA platforms, front button and on/off (ignition) signal can be configured.

For the VC platform, CI state activity and Can data reception can also be used as wakeup sources from suspend mode. bit 0 - enable wakeup by front button (from OFF and suspend mode) bit 1 - enable wakeup by on/off (ignition) signal (from OFF and suspend mode) bit 2 - enable wakeup by CAN activity (from suspend mode, VC only) bit 3 - enable wakeup by CI (Configurable input) state change (from suspend mode, VC only)

Note that there must always be a way to start the unit from shutdown mode. Therefore, at least one of the following must be true:

• Front button enabled as start-up trigger AND (CCpilot VC) at least one button configured as start-up trigger
• External on/off (ignition) signal configured as start-up trigger.

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
CAN_Button_Activity VC platform, wake up on CAN and Buttons
CAN_OnOff_Activity VC platform, wake up on CAN and On/Off/Ignition signal
CAN_Button_OnOff_Activity VC platform, wake up on CAN, Buttons and On/Off/Ignition signal
CI_Button_Activity VC platform, wake up on CI and Button State Change
CI_OnOff_Activity VC platform, wake up on CI and On/Off/Ignition signal State Change
CI_Button_OnOff_Activity VC platform, wake up on CI, Button and On/Off/Ignition signal State Change
CI_CAN_Button_Activity VC platform, wake up on CI, CAN and Button State Change
5.1 CrossControl Namespace Reference

CI_CAN_OnOff_Activity  VC platform, wake up on CI, CAN and On/Off/Ignition signal State Change

All_Events  VC platform, wake up on all events

Last_trigger_conf

5.1.2.28  enum TSAvancedSettingsParameter

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.

TSDebeounce_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.

TSDebeounce_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 TSDebeounce_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_CALIBRATION_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.

5.1.2.29 enum UpgradeAction

Upgrade Action enumeration

Enumerator

- **UPGRADE_INIT**  Initiating, checking for compatibility etc
- **UPGRADE_PREP_COM**  Preparing communication
- **UPGRADE_READING_FILE**  Opening and reading the supplied file
- **UPGRADE_CONVERTING_FILE**  Converting the mcs format to binary format
- **UPGRADE_FLASHING**  Flashing the file
- **UPGRADE_VERIFYING**  Verifying the file
- **UPGRADE_COMPLETE**  Verifying the programmed image
- **UPGRADE_COMPLETE_WITH_ERRORS**  Upgrade finished prematurely, see errorCode for the reason of failure

5.1.2.30 enum VideoChannel

The available analog video channels

Enumerator

- **Analog_Channel_1**
- **Analog_Channel_2**
- **Analog_Channel_3**
- **Analog_Channel_4**

5.1.2.31 enum VideoRotation

Enumerator

- **RotNone**
- **Rot90**
- **Rot180**
- **Rot270**
5.1 CrossControl Namespace Reference

5.1.2.32 enum videoStandard

Enumerator

\[\text{STD} \_ \text{M,J NTSC} \]
\[\text{STD} \_ \text{B,D,G,H,I,N PAL} \ (\text{M,J}) \text{ NTSC ITU-R BT.601} \]
\[\text{STD} \_ \text{M,PAL} \ (\text{B,D,G,H,I,N}) \text{ PAL ITU-R BT.601} \]
\[\text{STD} \_ \text{PAL} \ (\text{M}) \text{ PAL ITU-R BT.601} \]
\[\text{STD} \_ \text{NTSC} \text{ PAL-Nc ITU-R BT.601} \]
\[\text{STD} \_ \text{SECAM} \text{ NTSC 4.43 ITU-R BT.601} \]

5.1.2.33 enum VoltageEnum

Voltage type enumeration

Enumerator

\[\text{VOLTAGE} \_ \text{24VIN} \]
\[\text{VOLTAGE} \_ \text{24V} < 24\text{VIN} \]
\[\text{VOLTAGE} \_ \text{12V} < 24\text{V} \]
\[\text{VOLTAGE} \_ \text{12VID} < 12\text{V} \]
\[\text{VOLTAGE} \_ \text{5V} < 12\text{VID} \]
\[\text{VOLTAGE} \_ \text{3V3} < 5\text{V} \]
\[\text{VOLTAGE} \_ \text{VTFT} < 3.3\text{V} \]
\[\text{VOLTAGE} \_ \text{5VSTB} < \text{VTFT} \]
\[\text{VOLTAGE} \_ \text{IV9} < 5\text{VSTB} \]
\[\text{VOLTAGE} \_ \text{IV8} < 1.9\text{V} \]
\[\text{VOLTAGE} \_ \text{IV5} < 1.8\text{V} \]
\[\text{VOLTAGE} \_ \text{IV2} < 1.5\text{V} \]
\[\text{VOLTAGE} \_ \text{IV05} < 1.2\text{V} \]
\[\text{VOLTAGE} \_ \text{IV0} < 1.05\text{V} \]
\[\text{VOLTAGE} \_ \text{IV9} < 1.0\text{V} \]
\[\text{VOLTAGE} \_ \text{VREF_INT} < 0.9\text{V} \]
\[\text{VOLTAGE} \_ \text{24V_BACKUP} < \text{SS internal VRef} \]
\[\text{VOLTAGE} \_ \text{2V5} < 24\text{V backup capacitor} \]
\[\text{VOLTAGE} \_ \text{IV1} < 2.5\text{V} \]
\[\text{VOLTAGE} \_ \text{IV3_PER} < 1.1\text{V} \]
\[\text{VOLTAGE} \_ \text{IV3_VDDA} < 1.3\text{V_PER} \]
\[\text{VOLTAGE} \_ \text{3VSTBY} < 1.3\text{V_VDDA} \]
\[\text{VOLTAGE} \_ \text{VPMIC} < 3.3\text{V STBY VC} \]
\[\text{VOLTAGE} \_ \text{VMAIN} < \text{V PMIC VC} \]
\[< \text{V MAIN VC} \]
5.1 CrossControl Namespace Reference

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): XL, XM, XS, XA

Parameters

<table>
<thead>
<tr>
<th>buff</th>
<th>Text output buffer.</th>
</tr>
</thead>
<tbody>
<tr>
<td>len</td>
<td>Maximum length of the output buffer. If the actual length of the data is greater, an error will be returned.</td>
</tr>
</tbody>
</table>

Returns

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

Example Usage:

```c
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): XL, XM, XS, XA

Parameters

<table>
<thead>
<tr>
<th>buff</th>
<th>Text output buffer.</th>
</tr>
</thead>
<tbody>
<tr>
<td>len</td>
<td>Maximum length of the output buffer. If the actual length of the data is greater, an error will be returned.</td>
</tr>
</tbody>
</table>

Returns

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

Example Usage:

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

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): XL, XM, XS, XA

Parameters

<table>
<thead>
<tr>
<th>buff</th>
<th>Text output buffer.</th>
</tr>
</thead>
<tbody>
<tr>
<td>length</td>
<td>Maximum length of the output buffer. If the actual length of the data is greater, an error will be returned.</td>
</tr>
</tbody>
</table>

Returns

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

Example Usage:

```c
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): XL, XM, XS, XA

Parameters

<table>
<thead>
<tr>
<th>buff</th>
<th>Text output buffer.</th>
</tr>
</thead>
<tbody>
<tr>
<td>len</td>
<td>Maximum length of the output buffer. If the actual length of the data is greater, an error will be returned.</td>
</tr>
</tbody>
</table>

Returns

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

Example Usage:

```c
err = About_getAddOnPCBSerial (pAbout, buffer, buffer_len);
if (CrossControl::ERR_SUCCESS == err)
  cout << "Add on PCB serial number: " << buffer << endl;
```
Get display resolution.
Supported Platform(s): XL, XM, XS,XA, VC

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>buff</code></td>
<td>Text output buffer.</td>
</tr>
<tr>
<td><code>len</code></td>
<td>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 &quot;1024x768&quot;</td>
</tr>
</tbody>
</table>

Returns

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

Example Usage:

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

Get the front hardware pcb revision in the format major.minor (e.g. 1.1).
Supported Platform(s): XA, XS

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>major</code></td>
<td>The major pcb revision.</td>
</tr>
<tr>
<td><code>minor</code></td>
<td>The minor pcb revision.</td>
</tr>
</tbody>
</table>

Returns

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

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:

```cpp
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): XL, XM, XS, XA

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): XL, XM, XS, XA, VC

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): XL, XM, XS, XA

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 CrossControl Namespace Reference

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): XL, XM, XS, XA

Parameters

| mounted | Is module mounted? |

Returns

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

Example Usage:

```cpp
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:

```cpp
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.
5.1 CrossControl Namespace Reference

Supported Platform(s): XL, XM, XS, XA, VC

Parameters

| available | Is TouchScreen available? |

Returns

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

Example Usage:

```c
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): XL, XM, XS, XA

Parameters

| mounted | Is module mounted? |

Returns

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

Example Usage:

```c
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): XL, XM, XS, XA, VC
5.1 CrossControl Namespace Reference

Parameters

<table>
<thead>
<tr>
<th>buff</th>
<th>Text output buffer.</th>
</tr>
</thead>
<tbody>
<tr>
<td>len</td>
<td>Maximum length of the output buffer. If the actual length of the data is greater, an error will be returned.</td>
</tr>
</tbody>
</table>

Returns

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

Example Usage:

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

5.1.3.17 EXTERN_C CCAUX_DLL API eErr CCAUX_DLL_CALLING_CONV

CrossControl::About_getMainManufacturingDate(ABOUTHANDLE, char * buff, int len)

Get main manufacturing date.

Supported Platform(s): XL, XM, XS, XA, VC

Parameters

<table>
<thead>
<tr>
<th>buff</th>
<th>Text output buffer.</th>
</tr>
</thead>
<tbody>
<tr>
<td>len</td>
<td>Maximum length of the output buffer. If the actual length of the data is greater, an error will be returned.</td>
</tr>
</tbody>
</table>

Returns

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

Example Usage:

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

5.1.3.18 EXTERN_C CCAUX_DLL_API eErr CCAUX_DLL_CALLING_CONV

CrossControl::About_getMainPCBArt(ABOUTHANDLE, char * buff, int length)

Get main PCB article number.

Supported Platform(s): XL, XM, XS, XA, VC
5.1 CrossControl Namespace Reference

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>buff</code></td>
<td>Text output buffer.</td>
</tr>
<tr>
<td><code>length</code></td>
<td>Maximum length of the output buffer. If the actual length of the data is greater, an error will be returned.</td>
</tr>
</tbody>
</table>

Returns

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

Example Usage:

```cpp
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): XL, XM, XS, XA, VC

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>buff</code></td>
<td>Text output buffer.</td>
</tr>
<tr>
<td><code>len</code></td>
<td>Maximum length of the output buffer. If the actual length of the data is greater, an error will be returned.</td>
</tr>
</tbody>
</table>

Returns

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

Example Usage:

```cpp
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): XL, XM, XS, XA, VC

Parameters
5.1 CrossControl Namespace Reference

<table>
<thead>
<tr>
<th>buff</th>
<th>Text output buffer.</th>
</tr>
</thead>
<tbody>
<tr>
<td>len</td>
<td>Maximum length of the output buffer. If the actual length of the data is greater, an error will be returned.</td>
</tr>
</tbody>
</table>

Returns

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

Example Usage:

```c
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): XL, XM, XS, XA, VC

Parameters

<table>
<thead>
<tr>
<th>buff</th>
<th>Text output buffer.</th>
</tr>
</thead>
<tbody>
<tr>
<td>len</td>
<td>Maximum length of the output buffer. If the actual length of the data is greater, an error will be returned.</td>
</tr>
</tbody>
</table>

Returns

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

Example Usage:

```c
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_getNrOfButtons ( ABOUTHANDLE , int * numbuttons )`

Get number of configurable buttons.

Supported Platform(s): VC, VA

Parameters

60
5.1 CrossControl Namespace Reference

| numbuttons | Number of configurable buttons. |

Returns

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

Example Usage:

```c
int nrOfButtons;
err = About_getNrOfButtons (pAbout, &nrOfButtons);
if (CrossControl::ERR_SUCCESS == err)
    cout << "Nr of configurable buttons: " << (int)nrOfButtons << endl;
else if (CrossControl::ERR_NOT_SUPPORTED == err)
    cout << "About_getNrOfButtons: Not supported" << endl;
```

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

Get number of CAN connections present.
Supported Platform(s): XL, XM, XS, XA, VC

Parameters

| NrOfConnections | Returns the number of connections. |

Returns

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

Example Usage:

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

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

Get number of configurable input connections present.
Supported Platform(s): VC
### 5.1 CrossControl Namespace Reference

#### 5.1.3.25 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV

```c
CrossControl::About::getNrOfETHConnections (ABOUTHANDLE, unsigned char * NrOfConnections)
```

Get number of ethernet connections present.

Supported Platform(s): XL, XM, XS, XA, VC

#### Parameters

<table>
<thead>
<tr>
<th>NrOfConnections</th>
<th>Returns the number of input or input/output connections.</th>
</tr>
</thead>
</table>

#### Returns

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

Example Usage:

```c
unsigned char nrOfDigIOConnections;
err = About_getNrOfDigIOConnections (pAbout, &nrOfDigIOConnections);
if (CrossControl::ERR_SUCCESS == err)
    cout << "Nr of digital I/O connections: (int)nrOfDigIOConnections << endl;
else if (CrossControl::ERR_NOT_SUPPORTED == err)
    cout << "About_getNrOfDigIOConnections: Not supported" << endl;
```

---

#### 5.1.3.26 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV

```c
CrossControl::About::getNrOfETHConnections (ABOUTHANDLE, unsigned char * NrOfConnections)
```

Get number of ethernet connections present.

Supported Platform(s): XL, XM, XS, XA, VC

#### Parameters

<table>
<thead>
<tr>
<th>NrOfConnections</th>
<th>Returns the number of input or input/output connections.</th>
</tr>
</thead>
</table>

#### Returns

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

Example Usage:

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

Parameters

| NrOfConnections | Returns the number of connections. |

Returns

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

Example Usage:

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

5.1.3.27 EXTERN C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::About_getNrOfPWMOutConnections ( ABOUTHANDLE, unsigned char * NrOfConnections )

Get number of PWM Output connections present.

Supported Platform(s): VC

Parameters

| NrOfConnections | Returns the number of outputs. |

Returns

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

Example Usage:

```c
unsigned char nrOfPwmOut;
err = About_getNrOfPWMOutConnections (pAbout, &nrOfPwmOut);
if (CrossControl::ERR_SUCCESS == err)
cout << "Nr of PWM outputs: " << (int)nrOfPwmOut << endl;
else if (CrossControl::ERR_NOT_SUPPORTED == err)
cout << "About_getNrOfPWMOutConnections: Not supported" << endl;
```

5.1.3.28 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): XL, XM, XS, XA, VC
5.1 CrossControl Namespace Reference

Parameters

| NrOfConnections | Returns the number of connections. |

Returns

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

Example Usage:

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

5.1.3.29 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV

CrossControl::About_getNrOfUSBConnections ( ABOUTHANDLE , unsigned char * NrOfConnections )

Get number of USB connections present.

Supported Platform(s): XL, XM, XS, XA, VC

Parameters

| NrOfConnections | Returns the number of connections. |

Returns

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

Example Usage:

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

5.1.3.30 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV

CrossControl::About_getNrOfVideoConnections ( ABOUTHANDLE , unsigned char * NrOfConnections )

Get number of Video connections present.

Supported Platform(s): XL, XM, XS, XA, VC
5.1 CrossControl Namespace Reference

Parameters

<table>
<thead>
<tr>
<th>NrOfConnections</th>
<th>Returns the number of connections.</th>
</tr>
</thead>
</table>

Returns

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

Example Usage:

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

5.1.3.31 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV

CrossControl::About_getUnitSerial ( ABOUTHANDLE, char * buff, int len )

Get unit serial number.
Supported Platform(s): XL, XM, XS, XA, VC

Parameters

<table>
<thead>
<tr>
<th>buff</th>
<th>Text output buffer.</th>
</tr>
</thead>
<tbody>
<tr>
<td>len</td>
<td>Maximum length of the output buffer. If the actual length of the data is greater, an error will be returned.</td>
</tr>
</tbody>
</table>

Returns

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

Example Usage:

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

5.1.3.32 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV

CrossControl::About_getUserEepromData ( ABOUTHANDLE, char * buff, unsigned short length )

Get User Eeprom data. The user eeprom holds 4096 bytes of data which are fully accessible. Data is always read from position 0.
Supported Platform(s): XL, XM, XS, XA
Parameters

<table>
<thead>
<tr>
<th>buff</th>
<th>data buffer.</th>
</tr>
</thead>
<tbody>
<tr>
<td>length</td>
<td>data buffer length or number of data bytes to read.</td>
</tr>
</tbody>
</table>

Returns

error status. 0 = ERR_SUCCESS, otherwise error code.

5.1.3.33 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;

5.1.3.34 EXTERN C CCAUXDLL_API void CCAUXDLL_CALLING_CONV
CrossControl::About_release ( ABOUTHANDLE )

Delete the About object.

Supported Platform(s): XL, XM, XS, XA, VC

Returns

-

Example Usage:

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

5.1.3.35 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::About_setUserEepromData (ABOUTHANDLE, unsigned short startpos, const char *buff, unsigned short length )

Set User Eeprom data. The user eeprom holds 4096 bytes of data which are fully accessible.
Supported Platform(s): XL, XM, XS, XA

Parameters

<table>
<thead>
<tr>
<th>startpos</th>
<th>eeprom write start position.</th>
</tr>
</thead>
<tbody>
<tr>
<td>buff</td>
<td>data buffer.</td>
</tr>
<tr>
<td>length</td>
<td>buffer length to write.</td>
</tr>
</tbody>
</table>

Returns

error status. 0 = ERR_SUCCESS, otherwise error code.

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

Read measured voltage.
Supported Platform(s): XL, XM, XS, XA, VC

Parameters

<table>
<thead>
<tr>
<th>selection</th>
<th>The type of voltage to get.</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Voltage value in Volt. Can be undefined if return value is error code. Not all values are supported on all platforms, ERR_NOT_SUPPORTED will indicate that.</td>
</tr>
</tbody>
</table>

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)
5.1 CrossControl Namespace Reference

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;
}

5.1.3.37 EXTERN C CCAUXDLL_API void CCAUXDLL_CALLING_CONV
CrossControl::Adc_release ( ADCHANDLE )

Delete the ADC object.

Supported Platform(s): XL, XM, XS, XA, VC

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, "VFT", CrossControl::VOLTAGE_VFT);
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);
output_voltage (pAdc, "3V1_STBY", CrossControl::VOLTAGE_3V1STBY);
output_voltage (pAdc, "VMIC", CrossControl::VOLTAGE_VMIC);
output_voltage (pAdc, "VMAIN", CrossControl::VOLTAGE_VMAIN);
Adc_release(pAdc);

5.1.3.38 EXTERN C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::AuxVersionGetCCAuxDrvVersion ( 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.
5.1 CrossControl Namespace Reference

Supported Platform(s): XL, XM

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>major</td>
<td>Major version number</td>
</tr>
<tr>
<td>minor</td>
<td>Minor version number</td>
</tr>
<tr>
<td>release</td>
<td>Release version number</td>
</tr>
<tr>
<td>build</td>
<td>Build version number</td>
</tr>
</tbody>
</table>

Returns

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

Example Usage:

```c
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;
```

5.1.3.39 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): XL, XM, XS, XA, VC

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>major</td>
<td>Major version number</td>
</tr>
<tr>
<td>minor</td>
<td>Minor version number</td>
</tr>
<tr>
<td>release</td>
<td>Release version number</td>
</tr>
<tr>
<td>build</td>
<td>Build version number</td>
</tr>
</tbody>
</table>

Returns

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

```c
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;
```

5.1.3.40 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): XL, XM, XS, XA

<table>
<thead>
<tr>
<th></th>
<th>Major version number</th>
</tr>
</thead>
<tbody>
<tr>
<td>major</td>
<td>Minor version number</td>
</tr>
<tr>
<td>minor</td>
<td>Release version number</td>
</tr>
<tr>
<td>release</td>
<td>Build version number</td>
</tr>
</tbody>
</table>

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

Example Usage:

```c
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 CrossControl Namespace Reference

5.1.3.41 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): XL, XM, XS, XA

Parameters

<table>
<thead>
<tr>
<th>major</th>
<th>Minor version number</th>
</tr>
</thead>
<tbody>
<tr>
<td>minor</td>
<td>Minor version number</td>
</tr>
<tr>
<td>release</td>
<td>Release version number</td>
</tr>
<tr>
<td>build</td>
<td>Build version number</td>
</tr>
</tbody>
</table>

Returns

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

Example Usage:

```c
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;
```

5.1.3.42 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): XL, XM, XS, XA, VC

Parameters

<table>
<thead>
<tr>
<th>major</th>
<th>Major version number</th>
</tr>
</thead>
<tbody>
<tr>
<td>minor</td>
<td>Minor version number</td>
</tr>
<tr>
<td>release</td>
<td>Release version number</td>
</tr>
<tr>
<td>build</td>
<td>Build version number</td>
</tr>
</tbody>
</table>
5.1 CrossControl Namespace Reference

Returns

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

Example Usage:

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

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

5.1.3.43 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): XL, XM, XS,XA, VC

Parameters

<table>
<thead>
<tr>
<th>major</th>
<th>Minor version number</th>
</tr>
</thead>
<tbody>
<tr>
<td>minor</td>
<td>Minor version number</td>
</tr>
<tr>
<td>release</td>
<td>Release version number</td>
</tr>
<tr>
<td>build</td>
<td>Build version number</td>
</tr>
</tbody>
</table>

Returns

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

Example Usage:

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

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

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

Delete the AuxVersion object.
Supported Platform(s): XL, XM, XS, XA, VC

Returns

Example Usage:

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

5.1.3.45  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): XL, XM, XS, XA, VC

Parameters

<table>
<thead>
<tr>
<th>Average Window Size</th>
<th>The average window size in nr of samples.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reject Window Size</td>
<td>The reject window size in nr of samples.</td>
</tr>
<tr>
<td>Reject Delta In Lux</td>
<td>The reject delta in lux.</td>
</tr>
<tr>
<td>Mode</td>
<td>The configured sampling mode.</td>
</tr>
</tbody>
</table>

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.
5.1 CrossControl Namespace Reference

5.1.3.46 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): XL, XM, XS, XA, VC

Parameters

<table>
<thead>
<tr>
<th>bSoftTransitions</th>
<th>Soft transitions used?</th>
</tr>
</thead>
<tbody>
<tr>
<td>k</td>
<td>K value.</td>
</tr>
</tbody>
</table>

Returns

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

5.1.3.47 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::Backlight_getAutomaticBLStatus ( BACKLIGHThANDLE , unsigned char * status )

Get status from automatic backlight control.
Supported Platform(s): XL, XM, XS, XA, VC

Parameters

| status | 1=running, 0=stopped. |

Returns

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

5.1.3.48 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): XL, XM, XS, XA
5.1 CrossControl Namespace Reference

**Returns**

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

**Example Usage:**

```c
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));
}
```

5.1.3.49 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):** XL, XM, XS, XA, VC

**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:**

```c
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 CrossControl Namespace Reference

5.1.3.50 EXTERN C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::Backlight_getLedDimming ( BACKLIGHTHANDLE, CCStatus ∗ status )

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

Supported Platform(s): XL, XM, XS, XA, VC

Parameters

| status | Enabled/Disabled |

Returns

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

5.1.3.51 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): XL, XM

Parameters


Returns

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

Example Usage:

```c
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
```
5.1 CrossControl Namespace Reference

```c
printf("Error(%d) in function Backlight_getStatus: %s\n", err,
       GetErrorStringA(err));
```

### 5.1.3.52 EXTERN C CCAUXDLL_API void CCAUXDLL_CALLING_CONV
CrossControl::Backlight_release ( BACKLIGHTHANDLE )

Delete the backlight object.

**Supported Platform(s):** XL, XM, XS, XA, VC

**Returns**
- 

**Example Usage:**

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

### 5.1.3.53 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):** XL, XM, XS, XA, VC

**Parameters**

<table>
<thead>
<tr>
<th>parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>averageWndSize</td>
<td>The average window size in nr of samples.</td>
</tr>
<tr>
<td>rejectWndSize</td>
<td>The reject window size in nr of samples.</td>
</tr>
<tr>
<td>rejectDeltaInLux</td>
<td>The reject delta in lux.</td>
</tr>
<tr>
<td>mode</td>
<td>The configured sampling mode.</td>
</tr>
</tbody>
</table>

**Returns**

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.
5.1 CrossControl Namespace Reference

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

Set parameters for automatic backlight control.
Supported Platform(s): XL, XM, XS, XA, VC

Parameters

<table>
<thead>
<tr>
<th>bSoftTransitions</th>
<th>Use soft transitions?</th>
</tr>
</thead>
</table>

Returns

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

5.1.3.55 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): XL, XM, XS, XA, VC

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:

```c
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 CrossControl Namespace Reference

5.1.3.56 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): XL, XM, XS, XA, VC

Parameters

| status | Enabled/Disabled |

Returns

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

5.1.3.57 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): XL, XM, XS, XA, VC

Returns

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

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

Stop automatic backlight control.

Supported Platform(s): XL, XM, XS, XA, VC

Returns

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

5.1.3.59 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:

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

5.1.3.60 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV

Get battery hardware version (PCB revision).

Supported Platform(s): XM

Parameters

<table>
<thead>
<tr>
<th>buff</th>
<th>Text output buffer.</th>
</tr>
</thead>
<tbody>
<tr>
<td>len</td>
<td>Maximum length of the output buffer. If the actual length of the data is greater, an error will be returned.</td>
</tr>
</tbody>
</table>
5.1 CrossControl Namespace Reference

Returns

error status. \(0 = \text{ERR\_SUCCESS}\), otherwise error code. See the enum eErr for details.

Example Usage:

```c
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;
```

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

<table>
<thead>
<tr>
<th>buff</th>
<th>Text output buffer.</th>
</tr>
</thead>
<tbody>
<tr>
<td>len</td>
<td>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.</td>
</tr>
</tbody>
</table>

Returns

error status. \(0 = \text{ERR\_SUCCESS}\), otherwise error code. See the enum eErr for details.

Example Usage:

```c
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 CrossControl Namespace Reference

5.1.3.62 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:

```c
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;
}
```

5.1.3.63 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. |
5.1 CrossControl Namespace Reference

Returns

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

Example Usage:

```c
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;
}
```

5.1.3.64  EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV

CrossControl::Battery_getBatteryVoltageStatus ( BATTERYHANDLE , unsigned char * batteryVoltagePercent )

Get battery voltage status.

Supported Platform(s): XM

Parameters

<table>
<thead>
<tr>
<th>battery-Voltage-Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>the current voltage level of the battery, in percent [0..100].</td>
</tr>
</tbody>
</table>

Returns

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

Example Usage:

```c
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;
}
```
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


Returns

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

Example Usage:

```cpp
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;
    }
```
5.1.3.66 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:

```c
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;
}
```

5.1.3.67 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:

```c
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;
}

5.1.3.68  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;
cout << "Total run time on battery power=" << times.TotRunTimeBattery*60 << " min(s)" << std::endl;
cout << "Total run time below -20C=" << times.RunTime_m20 << " min(s)" << std::endl;
cout << "Run time from -20 to 0C=" << times.RunTime_m20_0 << " min(s)" << std::endl;
cout << "Total run time 0-40C=" << times.RunTime_0_40 << " min(s)" << std::endl;
cout << "Total run time 40-60C=" << times.RunTime_40_60 << " min(s)" << std::endl;
cout << "Total run time 60-70C=" << times.RunTime_60_70 << " min(s)" << std::endl;
cout << "Total run time 70-80C=" << times.RunTime_70_80 << " min(s)" << std::endl;
cout << "Total run time above 80C=" << times.RunTime_Above80 << " min(s)" << std::endl;
}
5.1 CrossControl Namespace Reference

5.1.3.69 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::BatteryIsBatteryPresent ( BATTERYHANDLE, bool * batteryIsPresent )

Is an external battery connected?
Supported Platform(s): XM

Parameters

| batteryIsPresent | true if a battery is connected, otherwise false. |

Returns

- 

Example Usage:

```c++
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;
```

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

Delete the Battery object
Supported Platform(s): XM.

Returns

- 

Example Usage:

```c++
BATTERYHANDLE pBattery = ::GetBattery();
assert(pBattery);
readBatteryInfo(pBattery);
Battery_release(pBattery);
```
5.1 CrossControl Namespace Reference

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

Buzzes for a specified time.
Supported Platform(s): XL, XM, XS, XA, VC

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>Time (ms) to buzz.</td>
</tr>
<tr>
<td>blocking</td>
<td>Blocking or non-blocking function.</td>
</tr>
</tbody>
</table>

Returns

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

Example Usage:

```c
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_getFrequency ( BUZZERHANDLE , unsigned short * frequency )

Get buzzer frequency.
Supported Platform(s): XL, XM, XS, XA, VC

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>frequency</td>
<td>Current frequency (700-10000 Hz).</td>
</tr>
</tbody>
</table>

Returns

eError 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_getTrigger ( BUZZERHANDLE , bool * trigger )

Get buzzer trigger. The Buzzer is enabled when the trigger is enabled.
5.1 CrossControl Namespace Reference

Supported Platform(s): XL, XM, XS, XA, VC

Parameters

| trigger | Current trigger status. |

Returns

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

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

Get buzzer volume.

Supported Platform(s): XL, XM, XS, XA, VC

Parameters

| volume | Current volume (0-51). |

Returns

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

Example Usage:

```c
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;
```

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

Delete the Buzzer object.

Supported Platform(s): XL, XM, XS, XA, VC

Returns

Example Usage:
BUZZERHANDLE pBuzzer = ::GetBuzzer();
assert(pBuzzer);

play_beeps(pBuzzer);
Buzzer_release(pBuzzer);

5.1.3.76 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::Buzzer_setFrequency ( BUZZERHANDLE , unsigned short frequency )
Set buzzer frequency.
Supported Platform(s): XL, XM, XS, XA, VC
Parameters
<table>
<thead>
<tr>
<th>frequency</th>
<th>Frequency to set (700-10000 Hz).</th>
</tr>
</thead>
</table>
Returns
error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.
Example Usage:

```c
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;
    }
}
```

5.1.3.77 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): XL, XM, XS, XA, VC
Parameters
| trigger | Status to set. |
5.1 CrossControl Namespace Reference

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::Buzzer_setVolume ( BUZERHANDLE , unsigned short volume )

Set buzzer volume.
Supported Platform(s): XL, XM, XS, XA, VC

Parameters

| volume | Volume to set (0-51). |

Returns

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

Example Usage:

```c
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;
}
```

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

Get baud rate
Supported Platform(s): XL, XM, XS, XA, VC

Parameters

<table>
<thead>
<tr>
<th>net</th>
<th>CAN net (1-4) to get settings for.</th>
</tr>
</thead>
<tbody>
<tr>
<td>baudrate</td>
<td>CAN baud rate (kbit/s).</td>
</tr>
</tbody>
</table>

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;
}

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

Get frame type
Supported Platform(s): XL, XM

Parameters

<table>
<thead>
<tr>
<th>net</th>
<th>CAN net (1-4) to get settings for.</th>
</tr>
</thead>
<tbody>
<tr>
<td>frameType</td>
<td>CAN frame type</td>
</tr>
</tbody>
</table>

Returns

erorr 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;
}

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

Delete the CanSetting object.
Supported Platform(s): XL, XM, XS, XA, VC

Returns

Example Usage:

CANSETTINGHANDLE pCanSetting = ::GetCanSetting();
assert(pCanSetting);
read_cansettings(pCanSetting);
CanSetting_release(pCanSetting);
5.1 CrossControl Namespace Reference

5.1.3.82 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): XL, XM, XS, XA, VC

Parameters

<table>
<thead>
<tr>
<th>net</th>
<th>CAN net (1-4).</th>
</tr>
</thead>
<tbody>
<tr>
<td>baudrate</td>
<td>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.</td>
</tr>
</tbody>
</table>

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::CanSetting_setFrameType ( CANSETTINGHANDLE , unsigned char net,
CanFrameType frameType )

Set frame type. The changes will take effect after a restart.
Supported Platform(s): XL, XM

Parameters

<table>
<thead>
<tr>
<th>net</th>
<th>CAN net (1-4).</th>
</tr>
</thead>
<tbody>
<tr>
<td>frameType</td>
<td>CAN frameType</td>
</tr>
</tbody>
</table>

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::CfgIn_getCfgInMode ( CFGINHANDLE , unsigned char channel,
CfgInModeEnum * get_mode )

Get Configurable Input mode
Supported Platform(s): VC, VA
5.1 CrossControl Namespace Reference

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>channel</code></td>
<td>Which configurable input channel to use, 1-2 (VC) or (1-8) VA, corresponding to physical input channel</td>
</tr>
<tr>
<td><code>get_mode</code></td>
<td>Storage container for retrieved mode Configurable input can be set to different measurement modes, this reads the setting back</td>
</tr>
</tbody>
</table>

Returns

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

Example Usage:

```c
err = CfgIn_getCfgInMode(pCfgIn, channel, &get_mode);
if (err != ERR_SUCCESS)
{
    cout << "CfgIn_getCfgInMode: " << GetErrorStringA(err) << std::endl;
}
else
{
    switch(get_mode)
    {
    case CFGIN_NOT_IN_USE: cout << "CfgIn_getCfgInMode (" << (int)channel << "): CFGIN_NOT_IN_USE" << std::endl; break;
    case CFGIN_HI_SWITCH: cout << "CfgIn_getCfgInMode (" << (int)channel << "): CFGIN_HI_SWITCH" << std::endl; break;
    case CFGIN_LOW_SWITCH: cout << "CfgIn_getCfgInMode (" << (int)channel << "): CFGIN_LOW_SWITCH" << std::endl; break;
    case CFGIN_VOLTAGE_2V5: cout << "CfgIn_getCfgInMode (" << (int)channel << "): CFGIN_VOLTAGE_2V5" << std::endl; break;
    case CFGIN_VOLTAGE_5V: cout << "CfgIn_getCfgInMode (" << (int)channel << "): CFGIN_VOLTAGE_5V" << std::endl; break;
    case CFGIN_VOLTAGE_32V: cout << "CfgIn_getCfgInMode (" << (int)channel << "): CFGIN_VOLTAGE_32V" << std::endl; break;
    case CFGIN_VOLTAGE_5V: cout << "CfgIn_getCfgInMode (" << (int)channel << "): CFGIN_VOLTAGE_5V" << std::endl; break;
    case CFGIN_VOLTAGE_10V: cout << "CfgIn_getCfgInMode (" << (int)channel << "): CFGIN_VOLTAGE_10V" << std::endl; break;
    case CFGIN_VOLTAGE_32V: cout << "CfgIn_getCfgInMode (" << (int)channel << "): CFGIN_VOLTAGE_32V" << std::endl; break;
    case CFGIN_VOLTAGE_32V: cout << "CfgIn_getCfgInMode (" << (int)channel << "): CFGIN_VOLTAGE_32V" << std::endl; break;
    case CFGIN_VOLTAGE_32V: cout << "CfgIn_getCfgInMode (" << (int)channel << "): CFGIN_VOLTAGE_32V" << std::endl; break;
    default: cout << "CfgIn_getCfgInMode (" << (int)channel << "): Unknown mode" << std::endl; break;
    }
}
```

5.1.3.85 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV

CrossControl::CfgIn_getFrequencyValue ( CFGINHANDLE, unsigned char channel, float *frequency )

Read the sampled frequency value from configurable input, when in modes other than frequency mode:

For ports 1-4, time base is 72 MHz ± 100 ppm (more accurate) For ports 5-8, time base is 60 kHz ± 100 ppm (less accurate) Input range is 0 Hz – 15 kHz. See technical manual for more details.

For all ports 1-8: CFGIN_FREQ_PD_5V - sample_value in Hz CFGIN_FREQ_PD_10V - sample_value in Hz CFGIN_FREQ_PD_32V - sample_value in Hz CFGIN_FREQ_F_5V - sample_value in Hz CFGIN_FREQ_F_10V - sample_value in Hz CFGIN_FREQ_F_32V - sample_value in Hz
5.1 CrossControl Namespace Reference

For ports 5-8 only: CFGIN_FREQ_PU_5V - sample_value in Hz CFGIN_FREQ_PU_10V - sample_value in Hz CFGIN_FREQ_PU_32V - sample_value in Hz

Supported Platform(s): VA

Parameters

| channel | Which configurable input channel to use, 1 through 8, corresponding to physical input channel |
| frequency | Read signal frequency in Hz; signal resolution and range depending on mode and port |

Returns

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

Example Usage:

5.1.3.86 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
   CrossControl::CfgIn_getMinFrequencyThreshold ( CFGINHANDLE , unsigned char channel , float ∗ frequency )

Read the configured minimum frequency threshold for configurable input, when in frequency measurement mode (CFGIN_FREQ_FLOATING, CFGIN_FREQ_PULLUP, CFGIN_FREQ_PULLDOWN for VC, and CFGIN_FREQ_PD_5V, CFGIN_FREQ_PD_10V, CFGIN_FREQ_PD_32V, CFGIN_FREQ_F_5V, CFGIN_FREQ_F_10V, CFGIN_FREQ_F_32V, CFGIN_FREQ_PULLUP_5V, CFGIN_FREQ_PULLUP_10V, CFGIN_FREQ_PULLUP_32V for VA). The frequency threshold is set to 1Hz at device start-up. Use the frequency threshold to set up how fast to detect a frequency change or a static signal. If you know the frequency range of the measured signal - set the threshold slightly lower than this. That way, a change from pulses to a static signal is detected as fast as possible. If the frequency threshold is set to e.g. 0.1Hz, it can take up to 10 seconds before a change in frequency is detected - also depending on the actual frequency of the signal. For VC, when the measured signal is slower than the frequency threshold, CfgIn_getPwmValue will return frequency 0Hz, duty cycle 0 or 100%. For VA, when the measured signal is slower than the frequency threshold, CfgIn_getFrequencyValue will return frequency 0 Hz.

Supported Platform(s): VC, VA

Parameters

| channel | Which configurable input channel to use, 1-2 (VC) or 1-8 (VA), corresponding to physical input channel |
| frequency | Minimum frequency threshold, 0.0 - 50000.0 Hz for VC, 0 - 15000 Hz for VA. |
5.1 CrossControl Namespace Reference

Returns

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

Example Usage:

```c
float frequency_threshold;
channel = 2;
err = CfgIn_getMinFrequencyThreshold(pCfgIn, channel, &frequency_threshold);
if (err != ERR_SUCCESS)
{
    cout << "CfgIn_getMinFrequencyThreshold: " << GetErrorStringA(err) << std::endl;
}
else
{
    cout << "CfgIn_getMinFrequencyThreshold: channel 2: " << std::fixed << frequency_threshold << "Hz" << std::endl;
}
```

5.1.3.87 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::CfgIn_getPwmValue ( CFGINHANDLE , unsigned char channel, float * frequency, unsigned char * duty_cycle )

Read the sampled value from configurable input, when in frequency measurement mode (CFGIN_FREQ_FLOATING, CFGIN_FREQ_PULLUP, CFGIN_FREQ_PULLDOWN).

Supported Platform(s): VC

Parameters

<table>
<thead>
<tr>
<th>channel</th>
<th>Which configurable input channel to use, 1 or 2, corresponding to physical input channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>frequency</td>
<td>Read signal frequency, 0.0 - 50000.0 Hz</td>
</tr>
<tr>
<td>duty_cycle</td>
<td>Read signal duty cycle, 0-100%</td>
</tr>
</tbody>
</table>

Returns

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

Example Usage:

```c
float frequency;
unsigned char duty_cycle;
err = CfgIn_getPwmValue(pCfgIn, 2, &frequency, &duty_cycle);
if (err != ERR_SUCCESS)
{
    cout << "CfgIn_getPwmValue: " << GetErrorStringA(err) << std::endl;
}
else
{
    cout << "CfgIn_getPwmValue: channel 2 PWM measurement: " << std::fixed << frequency << "Hz, " << (int) duty_cycle << "% duty cycle" << std::endl;
}
```
CrossControl::CfgIn_getValue ( CFGINHANDLE, unsigned char channel, unsigned short *sample_value )

Read the sampled value from configurable input, when in modes other than frequency mode:

For VC platform:
- CFGIN_HI_SWITCH - sample_value is 0-1
- CFGIN_LOW_SWITCH - sample_value is 0-1
- CFGIN_VOLTAGE_2V5 - sample_value is 0-30000 (0.1mV steps)
- CFGIN_VOLTAGE_5V - sample_value is 0-60000 (0.1mV steps)
- CFGIN_RESISTANCE - sample_value is 0-65535 Ohm

For VA platform, all ports 1-8:
- CFGIN_CURRENT_4_20 - sample_value in µA, accuracy ± 0.5% ± 50 µA (typical)
- CFGIN_VOLTAGE_5V - sample_value in mV, accuracy ± 0.5% ± 5 mV (typical)
- CFGIN_VOLTAGE_10V - sample_value in mV, accuracy ± 0.5% ± 10 mV (typical)
- CFGIN_VOLTAGE_32V - sample_value in mV, accuracy ± 0.5% ± 32 mV (typical)

For VA platform, ports 1-4 only:
- CFGIN_RESISTANCE - sample_value in Ohm, accuracy ± 0.5% ± 5 Ohm (typical)
- CFGIN_RESISTANCE_500 - sample_value in 0.1 Ohm/bit, accuracy ± 0.5% ± 0.5 Ohm (typical)

For VA platform, ports 5-8 only:
- CFGIN_DIGITAL_PU_5V - sample_value is 0-1
- CFGIN_DIGITAL_PU_10V - sample_value is 0-1
- CFGIN_DIGITAL_PU_32V - sample_value is 0-1

Supported Platform(s): VC, VA

Parameters

<table>
<thead>
<tr>
<th>channel</th>
<th>Which configurable input channel to use, 1-2 (VC) or 1-8 (VA), corresponding to physical input channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>sample_value</td>
<td>Read value which is relevant to actual mode setting. The actual value is dependent on the mode setting</td>
</tr>
</tbody>
</table>

Returns

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

Example Usage:

```c
unsigned short value;
err = CfgIn_getValue(pCfgIn, 1, &value);
if (err != ERR_SUCCESS)
{
    cout << "CfgIn_getValue: " << GetErrorStringA(err) << std::endl;
}
else
{
    cout << "CfgIn_getValue: channel 1 2V5 voltage measurement: " << (int)value << "mV" << std::endl;
}
```
5.1.3.89  EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV

CrossControl::CfgIn_release ( CFGINHANDLE )

Delete the CfgIn object.

Supported Platform(s): VC, VA

Returns

- 

Example Usage:

```c
CFGINHANDLE pCfgIn = ::GetCfgIn();
assert(pCfgIn);

CfgIn_example(pCfgIn);

CfgIn_release(pCfgIn);
```

5.1.3.90  EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV

CrossControl::CfgIn_setCfgInMode ( CFGINHANDLE, unsigned char channel, CfgInModeEnum set_mode )

Set Configurable Input mode

Supported Platform(s): VC, VA

Parameters

<table>
<thead>
<tr>
<th>channel</th>
<th>Which configurable input channel to use, 1-2 (VC) or 1-8 (VA), corresponding to physical input channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>set_mode</td>
<td>Which mode to set Configurable input can be set to different measurement modes. See CfgInModeEnum for a description of which platform and input combinations are possible.</td>
</tr>
</tbody>
</table>

Returns

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

Example Usage:

```c
e = CfgIn_setCfgInMode(pCfgIn, channel, CFGIN_VOLTAGE_2V5);
if (e != ERR_SUCCESS)
{
    cout << "CfgIn_setCfgInMode: " << GetErrorStringA(e) << std::endl;
}
else
{
    cout << "CfgIn_setCfgInMode: channel 1 mode set to CFGIN_VOLTAGE_2V5" << std::endl;
}
```

5.1 CrossControl Namespace Reference

5.1.3.91 EXTERN C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
   CrossControl::CfgIn_setFrequencyFilterLevel ( CFGINHANDLE , unsigned char level )

Sets the weight of the old sample value in frequency measurements as a percentage.
The sampled frequency is filtered with a moving average. A large weight increases
the filter level and gives better accuracy in high frequency measurements, but decreases
the speed of which changes in the input frequency can be detected.

Supported Platform(s): VA

Parameters

| level | Weight of the old sample value as a percentage (0-99) |

Returns

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

Example Usage:

5.1.3.92 EXTERN C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
   CrossControl::CfgIn_setMinFrequencyThreshold ( CFGINHANDLE , unsigned char channel, float frequency )

Set the minimum frequency threshold for configurable input, when in frequency mea-
surement mode (CFGIN_FREQ_FLOATING, CFGIN_FREQ_PULLUP, CFGIN_FREQ_ PULLDOWN for VC, and CFGIN_FREQ_PD_5V, CFGIN_FREQ_PD_10V, CFGIN_FREQ_PD_32V, CFGIN_FREQ_F_5V, CFGIN_FREQ_F_10V, CFGIN_FREQ_F_32V, CFGIN_FREQ_PU_5V, CFGIN_FREQ_PU_10V, CFGIN_FREQ_PU_32V
for VA). The frequency threshold is set to 1Hz at device start-up. Use the frequency
threshold to set up how fast to detect a frequency change or a static signal. If you know
the frequency range of the measured signal - set the threshold slightly lower than this.
That way, a change from pulses to a static signal is detected as fast as possible. If the
frequency threshold is set to e.g. 0.1Hz, it can take up to 10 seconds before a change in
frequency is detected - also depending on the actual frequency of the signal. For VC,
when the measured signal is slower than the frequency threshold, CfgIn_getPwmValue
will return frequency 0Hz, duty cycle 0 or 100%. For VA, when the measured signal is
slower than the frequency threshold, CfgIn_getFrequencyValue will return frequency
0 Hz.

Supported Platform(s): VC, VA

Parameters

| channel | Which configurable input channel to use, 1-2 (VC) or 1-8 (VA), corres-
| frequency | Minimum frequency threshold, 0.0 - 50000.0 Hz for VC, 0-15000 Hz
          | for VA. |
5.1 CrossControl Namespace Reference

Returns

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

Example Usage:

```cpp
class CrossControl: :CfgIn_setMinFrequencyThreshold(CfgIn, channel, 50.0);
if (err != ERR_SUCCESS)
    cout << "CfgIn_setMinFrequencyThreshold: " << GetErrorStringA(err) << std::endl;
else
    cout << "CfgIn_setMinFrequencyThreshold: channel 2 minimum frequency threshold set to 50.0Hz" <<
        std::endl;
```

5.1.3.93 EXTERN_CCXaulx_DLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::Config_getButtonFunction ( CONFIGHANDLE, unsigned char
button_number, ButtonConfigEnum button_config )

Get Button Function Configuration

Supported Platform(s): VC

Parameters

| button_number | Which button to configure (1-MAX_BUTTONS) |
| button_config | Bitfield for button configuration, see enum ButtonConfigEnum for de-
|               | tails.                                     |

Returns

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

Example Usage:

```cpp
CrossControl::ButtonConfigEnum btnconf;
CrossControl::eErr error;
for (int i = 1; i < 9; i++)
    if (error != ERR_SUCCESS)
        cout << "Error( " << error << ") in function Config_getButtonFunction: " << GetErrorStringA(error) << std::endl;
    else
        cout << "Button " << (int)i << " is set to: ";
        switch (btnconf)
        {
            case BUTTON_ONLY_MP_ACTION: cout << "Application only" << std::endl; break;
            case BUTTON_AS_STARTUP_TRIG: cout << "Startup trigger" << std::endl; break;
```
case BUTTON_AS_ACTION_TRIG: cout << "Action trigger" << std::endl; break;
case BUTTON_AS_ACTION_STARTUP_TRIG: cout << "Action and Startup trigger" << std::endl; break;
case BUTTON_AS_BACKLIGHT_DECREASE: cout << "Backlight decrease" << std::endl; break;
case BUTTON_AS_BACKLIGHT_DECR_STARTUP_TRIG: cout << "Backlight decrease and Startup trigger" << std::endl; break;
case BUTTON_AS_BACKLIGHT_INCREASE: cout << "Backlight increase" << std::endl; break;
case BUTTON_AS_BACKLIGHT_INCR_STARTUP_TRIG: cout << "Backlight increase and Startup trigger" << std::endl; break;
default: cout << "Invalid value" << std::endl; break;
}

5.1.3.94 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): XL, XM, XS, XA

Parameters

<table>
<thead>
<tr>
<th>status</th>
<th>Enabled/Disabled</th>
</tr>
</thead>
</table>

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::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): XL, XM

Parameters

<table>
<thead>
<tr>
<th>status</th>
<th>Enabled/Disabled</th>
</tr>
</thead>
</table>

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.
5.1 CrossControl Namespace Reference

5.1.3.96 EXTERN C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::Config_getExtOnOffSigTrigTime ( CONFIGHANDLE , unsigned long ∗ trigertime )

Get external on/off signal trigger time.
Supported Platform(s): XL, XM, XS, XA, VC

Parameters

| trigertime | 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_getFrontBtnTrigTime ( CONFIGHANDLE , unsigned short ∗ trigertime )

Get front button trigger time for long press.
Supported Platform(s): XL, XM, XS, XA, VC

Parameters

| trigertime | 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_getHeatingTempLimit ( CONFIGHANDLE , signed short ∗ temperature )

Get the temperature 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): XL, XM, XS, XA, VC
5.1 CrossControl Namespace Reference

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.

5.1.3.99 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): XL, XM, XS, XA, VC

Parameters

| action | The configured action |

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_getOnOffSigAction ( CONFIGHANDLE, PowerAction * action )


Supported Platform(s): XL, XM, XS, XA, VC

Parameters

| action | The configured action |

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::getOnOffSignalState ( CONFIGHANDLE, CCStatus * enabled )

Get OnOff signal state
Supported Platform(s): XA, XS, VC, VA, VIT

Parameters

| enabled | Is OnOff signal 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::getOnOffTriggerMode ( CONFIGHANDLE, ConfigOnOffTriggerMode * mode )

Get OnOff/Ignition/KeySwitch signal trigger mode.
Supported Platform(s): XA, XS, VC, VA, VIT

Parameters

| mode | Signal trigger mode. See ConfigOnOffTriggerMode for details |

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::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): XL, XM, XS, XA, VC

Parameters

| status | Enabled/Disabled |
Returns

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

5.1.3.104 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

<table>
<thead>
<tr>
<th>port</th>
<th>RS4XX port (RS4XXPort1-4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>enabled</td>
<td>Is the RS485 port enabled (true/false)</td>
</tr>
</tbody>
</table>

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.getShortButtonPressAction ( CONFIGHANDLE , PowerAction * action )

Supported Platform(s): XL, XM, XS, XA, VC

Parameters

| action | The configured action. |

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.getStartupTriggerConfig ( CONFIGHANDLE , TriggerConf * config )

Get Start-up trigger configuration. Is the front button and/or the external on/off (ignition) signal enabled as triggers for startup and wake up from suspended mode? VC
platform: CI state change and Can activity also available as wakeup triggers from suspend mode. See enum TriggerConf for more details.

Supported Platform(s): XL, XM, XS, XA, VC

Parameters

| config | See enum TriggerConf. |

Returns

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

Example Usage:

```c
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;
        case CAN_Button_Activity: cout << "Wake up on CAN and Buttons" << endl; break;
        case CAN_OnOff_Activity: cout << "Wake up on CAN and On/Off/Ignition signal" << endl; break;
        case CAN_Button_OnOff_Activity: cout << "Wake up on CAN, Buttons and On/Off/Ignition signal" << endl; break;
        case CI_Button_Activity: cout << "Wake up on CI and Button State Change" << endl; break;
        case CI_OnOff_Activity: cout << "Wake up on CI and OnOff Signal State Change" << endl; break;
        case CI_Button_OnOff_Activity: cout << "Wake up on CI, Button and OnOff Signal State Change" << endl; break;
        case CI_CAN_Button_Activity: cout << "Wake up on CI, CAN and Button State Change" << endl; break;
        case CI_CAN_OnOff_Activity: cout << "Wake up on CI, CAN and OnOff Signal State Change" << endl; break;
        case CI_CAN_Button_OnOff_Activity: cout << "Wake up on CI, CAN, Button and OnOff Signal State Change" << endl; break;
        case CI_CAN_ALL_EVENTS: cout << "Wake up on all events" << endl; break;
        default: cout << "Error - Undefined StartupTrigger" << endl; break;
    }
}
else
{
    cout << "Error(" << err << ") in function getStartupTriggerConfig: " << GetErrStringA(err) << endl;
}
```

5.1.3.107 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV

CrossControl::Config_getStartupVoltageConfig ( CONFIGHANDLE, double * voltag e )

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): XL, XM
Parameters

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

Returns

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

5.1.3.108 EXTERN_C CCAUXDLL_API eErr CCAUXDLL\_CALLING\_CONV
CrossControl::Config\_getSuspendMaxTime ( CONFIGHANDLE , unsigned short * maxTime )

Get suspend mode maximum time.
Supported Platform(s): XL, XM, VC

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. $0 = \text{ERR\_SUCCESS}$, otherwise error code. See the enum eErr for details.

5.1.3.109 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): XL, XM, XS, XA

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. $0 = \text{ERR\_SUCCESS}$, otherwise error code. See the enum eErr for details.
5.1 CrossControl Namespace Reference

5.1.3.110 EXTERN C CCAUXDLL_API void CCAUXDLL_CALLING_CONV CrossControl::Config_release ( CONFIGHANDLE )

Delete the Config object.
Supported Platform(s): XL, XM, XS, XA, VC

Returns

Example Usage:

```c
CONFIGHANDLE pConfig = :GetConfig();
assert(pConfig);
conf_example(pConfig);
Config_release(pConfig);
```

5.1.3.111 EXTERN C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Config_setButtonFunction ( CONFIGHANDLE , unsigned char button_number, ButtonConfigEnum button_config )

Set button function configuration
Supported Platform(s): VC

Parameters

<table>
<thead>
<tr>
<th>button_number</th>
<th>Which button to configure (1-MAX_BUTTONS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>button_config</td>
<td>Bitfield for button configuration, see enum ButtonConfigEnum for details.</td>
</tr>
</tbody>
</table>

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::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): XL, XM, XS, XA
5.1 CrossControl Namespace Reference

Parameters

| status   | Enabled/Disabled |

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::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): XL, XM

Parameters

| status   | Enabled/Disabled |

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::Config_setExtOnOffSigTrigTime ( CONFIGHANDLE, unsigned long triggertime )

Set external on/off signal trigger time.

Supported Platform(s): XL, XM, XS, XA, VC

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 CrossControl Namespace Reference

5.1.3.115  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): XL, XM, XS, XA, VC

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.

5.1.3.116  EXTERN C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
            CrossControl::Config_setHeatingTempLimit ( CONFIGHANDLE, signed short
            temperature )

Set the temperature limit for heating. When temperature is below this limit, the sys-
tem 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): XL, XM, XS, XA, VC

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.

5.1.3.117  EXTERN C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
            CrossControl::Config_setLongButtonPressAction ( CONFIGHANDLE, PowerAction
            action )

Set long button press action. Sets the configured action for a long button press: No-
Action, ActionSuspend or ActionShutDown. A long button press is determined by the
FrontBtnTrigTime.
Supported Platform(s): XL, XM, XS, XA, VC
5.1 CrossControl Namespace Reference

Parameters

| action | The action to set. |

Returns

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

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

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

Supported Platform(s): XL, XM, XS, XA, VC

Parameters

| action | The action to set. |

Returns

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

5.1.3.119 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::Config_setOnOffTriggerMode ( CONFIGHANDLE, ConfigOnOffTriggerMode mode )

Set OnOff/Ignition/KeySwitch signal trigger mode.

Supported Platform(s): XA, XS, VC, VA, VIT

Parameters

| mode | Signal trigger mode. See ConfigOnOffTriggerMode for details |
5.1 CrossControl Namespace Reference

Returns

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

5.1.3.120 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): XL, XM, XS, XA, VC

Parameters

| status | Enabled/Disabled |

Returns

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

5.1.3.121 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

<table>
<thead>
<tr>
<th>port</th>
<th>RS4XX port (RS4XXPort1-4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>enabled</td>
<td>RS485 enabled (true/false)</td>
</tr>
</tbody>
</table>

Returns

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

5.1.3.122 EXTERN C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::Config_setShortButtonPressAction ( CONFIGHANDLE , PowerAction action )

Supported Platform(s): XL, XM, XS, XA, VC

Parameters

| action | The action to set. |

Returns

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

Example Usage:

```c
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;
```

5.1.3.123 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 (ignition) signal be enabled as triggers for startup and wake up from suspended mode? VC platform: CI state change and Can activity also available as wakeup triggers from suspend mode. See enum TriggerConf for more details.

Supported Platform(s): XL, XM, XS, XA, VC

Parameters

| conf | See enum TriggerConf. |

Returns

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

5.1.3.124 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): XL, XM
Parameters

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

Returns

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

5.1.3.125 EXTERN C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::Config_setSuspendMaxTime ( CONFIGHANDLE, unsigned short maxTime )

Set suspend mode maximum time.
Supported Platform(s): XL, XM, VC

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.

5.1.3.126 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): XL, XM, XS, XA

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.
5.1.3.127 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): XL, XM, XS, XA, VC

Returns

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

5.1.3.128 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): XL, XM, XS, XA, VC

Parameters

|-----------|---------------------------------|

Returns

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

5.1.3.129 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): XL, XM, XS, XA, VC

Parameters

<table>
<thead>
<tr>
<th>minTemp</th>
<th>Minimum measured PCB temperature.</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxTemp</td>
<td>Maximum measured PCB temperature.</td>
</tr>
</tbody>
</table>
5.1 CrossControl Namespace Reference

Returns

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

Example Usage:

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

5.1.3.130 EXTERN C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::Diagnostic_getPCBTemp ( DIAGNOSTICHANDLE , signed short *
temperature )

Get PCB temperature.

Supported Platform(s): XL, XM, XS, XA, VC

Parameters

| temperature | PCB Temperature in degrees Celsius. |

Returns

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

5.1.3.131 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 or Congatec CGOS API. These API’s also has a number of other functions, please
see the JIDA/CGOS documentation for how to use them separately.

Parameters

| index | Zero-based index of the temperature sensor. Different boards may have
different number of sensors. CCpilot XM and XL currently has 2 sens-
sors, board and cpu. An error is returned if the index is not supported.
CCpilot XM 2.0 supports only one sensor, CPU temperature. |

Supported Platform(s): XL, XM

Parameters

| temperature | Temperature in degrees Celsius. |
| jst | The type of sensor that is being read. |
5.1 CrossControl Namespace Reference

Returns

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

5.1.3.132 EXTERN C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::Diagnostic::getPowerCycles ( DIAGNOSTICHANDLE , unsigned short * powerCycles )

Get number of power cycles.
Supported Platform(s): XL, XM, XS, XA, VC

Parameters

| powerCycles | Total number of power cycles. |

Returns

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

5.1.3.133 EXTERN C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::Diagnostic::getShutDownReason ( DIAGNOSTICHANDLE , unsigned short * reason )

Get shutdown reason.
Supported Platform(s): XL, XM

Parameters

| reason | See DiagnosticCodes.h for shutdown codes. |

Returns

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

5.1.3.134 EXTERN C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::Diagnostic::getSSTemp ( DIAGNOSTICHANDLE , signed short * temperature )

Get System Supervisor temperature.
Supported Platform(s): XL, XM, XS, XA, VC
5.1 CrossControl Namespace Reference

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:

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

5.1.3.135 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV

CrossControl::Diagnostic_getStartupReason ( DIAGNOSTICHANDLE , unsigned short * reason )

Get startup reason.

Supported Platform(s): XL, XM, XS, XA, VC

Parameters

| reason | See DiagnosticCodes.h for startup codes. |

Returns

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

5.1.3.136 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV

CrossControl::Diagnostic_getTimer ( DIAGNOSTICHANDLE , TimerType * times )

Get diagnostic timer.

Supported Platform(s): XL, XM, XS, XA, VC

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);

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

Delete the Diagnostic object.
Supported Platform(s): XL, XM, XS, XA, VC

Returns
-

Example Usage:

DIAGNOSTICHANDLE pDiagnostic = ::GetDiagnostic();
assert(pDiagnostic);
diagnostic_example(pDiagnostic);
Diagnostic_release(pDiagnostic);

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

Get Digital inputs.
Supported Platform(s): XL, XM, XS, XA

Parameters

|--------|

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;
}

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

Delete the DigIO object.

Supported Platform(s): XL, XM, XS, XA

Returns
-

Example Usage:

DIGIOHANDLE pDigIO = ::GetDigIO();
assert(pDigIO);
list_digital_inputs(pDigIO);
DigIO_release(pDigIO);

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

Set Digital outputs.

Supported Platform(s): XA, XS

Parameters

|-------|--------------------------------------------------------------------------------------------------|

Returns

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

Example Usage:
5.1 CrossControl Namespace Reference

```cpp
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;
```

5.1.3.141 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): XL, XM, XS, XA, VC

### Parameters

<table>
<thead>
<tr>
<th>status</th>
<th>The current status of the upgrade operation.</th>
</tr>
</thead>
<tbody>
<tr>
<td>blocking</td>
<td>Whether or not the function should wait until a new status event has</td>
</tr>
<tr>
<td></td>
<td>been reported. If blocking is set to false, the function will return im-</td>
</tr>
<tr>
<td></td>
<td>mediately with the current status.</td>
</tr>
</tbody>
</table>

### Returns

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

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

Delete the FirmwareUpgrade object.

Supported Platform(s): XL, XM, XS, XA, VC

### Returns

- Example Usage:

```cpp
FirmwareUpgrade_release (pFirmwareUpgrade);
```

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

Shut down the operating system.

Supported Platform(s): XL, XM, XS, XA, VC
5.1 CrossControl Namespace Reference

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::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): XL, XM, XS, XA

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;
}

5.1.3.145 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): XL, XM, XS, XA

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_startFpgaVerification(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;
    }

5.1.3.146 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): XL, XM, XS, XA

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;
    }
}

5.1.3.147  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): XL, XM, XS, XA

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
</tr>
<tr>
<td>blocking</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

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())
}
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;
    }
}

5.1.3.148 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): XL, XM, XS, XA, VC

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
<td>Path and filename to the .hex file to program.</td>
</tr>
<tr>
<td>blocking</td>
<td>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.</td>
</tr>
</tbody>
</table>

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;
    }
}

5.1.3.149  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): XL, XM, XS, XA, VC

| 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:

```c
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;
    }
}
```

### 5.1.3.150 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::FrontLEDgetColor ( FRONTLEDHANDLE , unsigned char * red,
                                unsigned char * green, unsigned char * blue )

Get front LED color mix.

Supported Platform(s): XL, XM, XS, XA, VC On the VC platform - the blue parameter
gets the button backlight intensity (0-15)

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>Red color intensity 0-0x0F.</td>
</tr>
<tr>
<td>green</td>
<td>Green color intensity 0-0x0F.</td>
</tr>
<tr>
<td>blue</td>
<td>Blue color intensity 0-0x0F.</td>
</tr>
</tbody>
</table>

**Returns**

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

**Example Usage:**

```c
```
\texttt{err = FrontLED\_getColor(pFrontLED, \&red, \&green, \&blue);}
if (err != ERR\_SUCCESS)
{
    std::cout << "Error(\" << err << ") in function getColor: " << GetErrorStringA(err) << std::endl;
}

5.1.3.151 \texttt{EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV}
\texttt{CrossControl::FrontLED\_getEnabledDuringStartup ( FRONTLED\_HANDLE , 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): XL, XM, XS,XA, VC

Parameters

| status | LED Enabled or Disabled during startup. |

Returns

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

5.1.3.152 \texttt{EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV}
\texttt{CrossControl::FrontLED\_getIdleTime ( FRONTLED\_HANDLE , unsigned char \* idleTime )}

Get front LED idle time.

Supported Platform(s): XL, XM, XS, XA, VC

Parameters

| idleTime | Time in 100ms increments. |

Returns

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

5.1.3.153 \texttt{EXTERN\_C CCAUXDLL\_API eErr CCAUXDLL\_CALLING\_CONV}
\texttt{CrossControl::FrontLED\_getNrOfPulses ( FRONTLED\_HANDLE , unsigned char \* nrOfPulses )}

Get number of pulses during a blink sequence.

Supported Platform(s): XL, XM, XS, XA, VC
5.1 CrossControl Namespace Reference

Parameters

- nrOfPulses | Number of pulses.

Returns

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

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

Get front LED off time.
Supported Platform(s): XL, XM, XS, XA, VC

Parameters

- offTime | Time in 10ms increments.

Returns

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

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

Get front LED on time.
Supported Platform(s): XL, XM, XS, XA, VC

Parameters

- onTime | Time in 10ms increments. 0 = off
5.1 CrossControl Namespace Reference

Returns

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

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

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

Supported Platform(s): XL, XM, XS, XA, VC

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:

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

5.1.3.157 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 pre-defined colors, UNDEFINED_COLOR will be returned. It is not recommended to use this function on the VC platform.

Supported Platform(s): XL, XM, XS, XA, VC

Parameters

| color | Color from CCAuxColor enum. |

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.
5.1 CrossControl Namespace Reference

5.1.3.158 EXTERN C CCAUXDLL_API void CCAUXDLL_CALLING_CONV
CrossControl::FrontLED_release ( FRONTLEDHANDLE )

Delete the FrontLED object.
Supported Platform(s): XL, XM, XS, XA, VC

Returns
-

Example Usage:

```c
FRONTLEDHANDLE pFrontLED = ::GetFrontLED();
assert(pFrontLED);
led_example(pFrontLED);
FrontLED_release(pFrontLED);
```

5.1.3.159 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): XL, XM, XS, XA, VC On the VC platform - use the blue parameter to set the button backlight intensity (0-15)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>Red color intensity 0-0x0F.</td>
</tr>
<tr>
<td>green</td>
<td>Green color intensity 0-0x0F.</td>
</tr>
<tr>
<td>blue</td>
<td>Blue color intensity 0-0x0F.</td>
</tr>
</tbody>
</table>

Returns

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

Example Usage:

```c
err = FrontLED_setColor(pFrontLED, red, green, blue);
if (err != ERR_SUCCESS)
{
    cout << "Error(" << err << ") in function setColor: " << GetLastErrorA(err) << endl;
}
```
5.1 CrossControl Namespace Reference

5.1.3.160 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): XL, XM, XS, XA, VC

Parameters

| status | Enable or Disable the LED during startup. |

Returns

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

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

Get front LED idle time.
Supported Platform(s): XL, XM, XS, XA, VC

Parameters

| idleTime | Time in 100ms. |

Returns

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

5.1.3.162 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): XL, XM, XS, XA, VC

Parameters

| nrOfPulses | Number of pulses. |
5.1 CrossControl Namespace Reference

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

Set front LED off.

Supported Platform(s): XL, XM, XS, XA, VC

Returns

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

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

Set front LED off time.

Supported Platform(s): XL, XM, XS, XA, VC

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;

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

Set front LED on time.

Supported Platform(s): XL, XM, XS, XA, VC
5.1 CrossControl Namespace Reference

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onTime</td>
<td>Time in 10ms increments. 0 = off</td>
</tr>
</tbody>
</table>

Returns

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

Example Usage:

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

5.1.3.166 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV

CrossControl::FrontLED_setSignal ( FRONTLEDHANDLE, double frequency,
unsigned char dutyCycle )

Set front LED signal.

Supported Platform(s): XL, XM, XS, XA, VC

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>frequency</td>
<td>LED blink frequency (0.2-50 Hz).</td>
</tr>
<tr>
<td>dutyCycle</td>
<td>LED on duty cycle (0-100%).</td>
</tr>
</tbody>
</table>

Returns

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

Note: The hardware cannot be set to have an on or off time of the LED that’s longer than 2.55s (255*10ms) Hence there are limitations in this function when using frequencies slower than 0.39Hz. At 0.38Hz, the valid duty cycle range is [3 - 97]. At 0.30Hz, the valid duty cycle range is [24 - 76]. At 0.20Hz, the valid duty cycle range is [49 - 51]. At 0.19Hz and slower, the behavior is undefined for all duty cycles, so this is not allowed to be set. The behavior is undefined outside these ranges but setting 0% or 100% duty cycle will always work, regardless of the frequency. If you need to blink in an unsupported range, it can be done with a software timer instead.

Example Usage:

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

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

Set one of the front LED standard colors. It is not recommended to use this function on the VC platform.

Supported Platform(s): XL, XM, XS, XA, VC

Parameters

| color | Color from CCAuxColor enum. |

Returns

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

Example Usage:

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

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

Factory function that creates instances of the About object.

Supported Platform(s): XL, XM, XS, XA, VC

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:

```c
ABOUTHANDLE pAbout = ::GetAbout();
assert(pAbout);
list_about_information(pAbout);
About_release(pAbout);
```
5.1 CrossControl Namespace Reference

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

Factory function that creates instances of the Adc object.

Supported Platform(s): XL, XM, XS, XA, VC

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:

```cpp
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, "VFT", CrossControl::VOLTAGE_VFT);
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, "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, "1V3_PER", CrossControl::VOLTAGE_1V3_PER);
output_voltage (pAdc, "1V3_VDDA", CrossControl::VOLTAGE_1V3_VDDA);
output_voltage (pAdc, "3V1 STBY", CrossControl::VOLTAGE_3V1STBY);
output_voltage (pAdc, "VPMIC", CrossControl::VOLTAGE_VPMIC);
output_voltage (pAdc, "VMAIN", CrossControl::VOLTAGE_VMAIN);

Adc_release(pAdc);
```

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

Factory function that creates instances of the AuxVersion object.

Supported Platform(s): XL, XM, XS, XA, VC

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:

```cpp
AUXVERSIONHANDLE pAux = ::GetAuxVersion();
assert(pAux);

output_voltage (pAux, "24VIN", CrossControl::VOLTAGE_24VIN);
output_voltage (pAux, "24V", CrossControl::VOLTAGE_24V);
output_voltage (pAux, "12V", CrossControl::VOLTAGE_12V);
output_voltage (pAux, "12VID", CrossControl::VOLTAGE_12VID);
output_voltage (pAux, "5V", CrossControl::VOLTAGE_5V);
output_voltage (pAux, "3V3", CrossControl::VOLTAGE_3V3);
output_voltage (pAux, "VFT", CrossControl::VOLTAGE_VFT);
output_voltage (pAux, "5VSTB", CrossControl::VOLTAGE_5VSTB);
output_voltage (pAux, "1V9", CrossControl::VOLTAGE_1V9);
output_voltage (pAux, "1V8", CrossControl::VOLTAGE_1V8);
output_voltage (pAux, "1V5", CrossControl::VOLTAGE_1V5);
output_voltage (pAux, "1V2", CrossControl::VOLTAGE_1V2);
output_voltage (pAux, "1V05", CrossControl::VOLTAGE_1V05);
output_voltage (pAux, "1V0", CrossControl::VOLTAGE_1V0);
output_voltage (pAux, "VREF_INT", CrossControl::VOLTAGE_VREF_INT);
output_voltage (pAux, "24V_BACKUP", CrossControl::VOLTAGE_24V_BACKUP);
output_voltage (pAux, "2V5", CrossControl::VOLTAGE_2V5);
output_voltage (pAux, "1V3_PER", CrossControl::VOLTAGE_1V3_PER);
output_voltage (pAux, "1V3_VDDA", CrossControl::VOLTAGE_1V3_VDDA);
output_voltage (pAux, "3V1 STBY", CrossControl::VOLTAGE_3V1STBY);
output_voltage (pAux, "VPMIC", CrossControl::VOLTAGE_VPMIC);
output_voltage (pAux, "VMAIN", CrossControl::VOLTAGE_VMAIN);

AuxVersion_release(pAux);
```
AUXVERSIONHANDLE pAuxVersion = ::GetAuxVersion();
assert (pAuxVersion);
output_versions(pAuxVersion);
AuxVersion_release(pAuxVersion);

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

Factory function that creates instances of the Backlight object.

Supported Platform(s): XL, XM, XS, XA, VC

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);

5.1.3.172 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 CrossControl Namespace Reference

5.1.3.173 EXTERN C CCAUXDLL_API BUZZERHANDLE CCAUXDLL_CALLING_CONV
CrossControl::GetBuzzer ( void )

Factory function that creates instances of the Buzzer object.
Supported Platform(s): XL, XM, XS, XA, VC

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);

5.1.3.174 EXTERN C CCAUXDLL_API CANSETTINGHANDLE CCAUXDLL_CALLING_CONV CrossControl::GetCanSetting ( void )

Factory function that creates instances of the CanSetting object.
Supported Platform(s): XL, XM, XS, XA, VC

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);

5.1.3.175 EXTERN C CCAUXDLL_API CFGINHANDLE CCAUXDLL_CALLING_CONV
CrossControl::GetCfgIn ( void )

Factory function that creates instances of the CfgIn object.
Supported Platform(s): VC, VA
5.1 CrossControl Namespace Reference

Returns

CFGINHANDLE to an allocated CfgIn object. The returned handle needs to be deallocated using the _CfgIn_release(CFGINHANDLE)_ method when it’s no longer needed. Returns NULL if it fails to allocate memory.

Example Usage:

```c
CFGINHANDLE pCfgIn = ::GetCfgIn();
assert(pCfgIn);
CfgIn_example(pCfgIn);
CfgIn_release(pCfgIn);
```

5.1.3.176 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): XL, XM, XS, XA, VC

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:

```c
CONFIGHANDLE pConfig = ::GetConfig();
assert(pConfig);
conf_example(pConfig);
Config_release(pConfig);
```

5.1.3.177 EXTERN_C CCAUXDLL_API DIAGNOSTICHANDLE CCAUXDLL_CALLING_CONV
crossControl::GetDiagnostic ( void )

Factory function that creates instances of the Diagnostic object.

Supported Platform(s): XL, XM, XS, XA, VC

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);

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

Factory function that creates instances of the DigIO object.

Supported Platform(s): XL, XM, XS, XA

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);

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

to get a string description.

Get a string description of an error code.

Supported Platform(s): XL, XM, XS, XA

Parameters

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

Returns
String description of an error code.

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

Get a string description of an error code.

Supported Platform(s): XL, XM, XS, XA
5.1 CrossControl Namespace Reference

Parameters

| errCode | An error code for which |

Returns

String description of an error code.

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

Factory function that creates instances of the FirmwareUpgrade object.

Supported Platform(s): XL, XM, XS, XA, VC

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:

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

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

Factory function that creates instances of the FrontLED object.

Supported Platform(s): XL, XM, XS, XA, VC

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:

```c
FRONTLEDHANDLE pFrontLED = ::GetFrontLED();
assert(pFrontLED);
led_example(pFrontLED);
FrontLED_release(pFrontLED);
```
5.1 CrossControl Namespace Reference

5.1.3.183 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.

5.1.3.184 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.

5.1.3.185 EXTERN C CCAUXDLL_API LIGHTSENSORHANDLE CCAUXDLL_CALLING_CONV CrossControl::GetLightsensor ( void )

Factory function that creates instances of the Lightsensor object.

Supported Platform(s): XL, XM, XS, XA, VC

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:

```c
LIGHTSENSORHANDLE pLightSensor = ::GetLightsensor();
assert(pLightSensor);
ls_example(pLightSensor);
Lightsensor_release(pLightSensor);
```
5.1.3.186 EXTERN_C CCAUXDLL_API POWERHANDLE CCAUXDLL_CALLING_CONV
CrossControl::GetPower ( void )

Factory function that creates instances of the Power object.
Supported Platform(s): XL, XM, XS, XA, VC

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:

```c
POWERHANDLE pPower = ::GetPower();
assert(pPower);
power_example(pPower);
Power_release(pPower);
```

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

Factory function that creates instances of the PowerMgr object.
Supported Platform(s): XL, XM, XS, XA, VC

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:

```c
CrossControl::eErr err;
POWERMGRHANDLE pPowerMgr = ::GetPowerMgr();
BATTERYNHANDLE 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(atnExit);
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);

5.1.3.188 EXTERN_C CCAUXDLL_API PWOUTHANDLE CCAUXDLL_CALLING_CONV
    CrossControl::GetPWMOut ( void )

Factory function that creates instances of the PWOut object.

Supported Platform(s): VC

Returns

PWOUTHANDLE to an allocated PWOut object. The returned handle needs
be deallocated using the PWOut_release(PWOUTHANDLE) method when
it’s no longer needed. Returns NULL if it fails to allocate memory.

Example Usage:

PWOUTHANDLE pPwmOut = ::GetPWMOut();
assert(pPwmOut);
pwmout_example(pPwmOut);
PWOut_release(pPwmOut);

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

Factory function that creates instances of the Smart object.

Supported Platform(s): XL, XM

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);

5.1.3.190 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.

5.1.3.191 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.

5.1.3.192 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);

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

Factory function that creates instances of the TouchScreen object.
Supported Platform(s): XL, XM, XS, XA

Returns

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

Example Usage:

TOUCHSCREENHANDLE pTouchScreen = ::GetTouchScreen();
assert(pTouchScreen);
touchscreen_example(pTouchScreen);
TouchScreen_release(pTouchScreen);

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

Factory function that creates instances of the TouchScreenCalib object.
Supported Platform(s): XL, XM, XS, XA

Returns

TOUCHSCREENCALIBHANDLE to an allocated TouchScreenCalib object. The returned handle needs to be deallocated using the TouchScreenCalib_release(TOUCHSCREENCALIBHANDLE) method when it’s no longer needed. Returns NULL if it fails to allocate memory.

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

Factory function that creates instances of the Video object.
Supported Platform(s): XL, XM, XS, XA, VC
5.1 CrossControl Namespace Reference

5.1.3.196 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): XL, XM, XS, XA, VC

Parameters

| value | Illuminance value (Lux). |

Returns

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

Example Usage:

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

5.1.3.197 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): XL, XM, XS, XA, VC

Parameters

| value | Illuminance value (Lux). |

Returns

e 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;
}

5.1.3.198 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): XL, XM, XS, XA, VC

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
</tr>
<tr>
<td>ch0</td>
</tr>
<tr>
<td>ch1</td>
</tr>
</tbody>
</table>

Returns

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

5.1.3.199 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. On the VC platform, the ranges correspond to 1000
and 4000 lux maximum value.

Supported Platform(s): XL, XM, XS, XA, VC

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>range</td>
</tr>
</tbody>
</table>
5.1 CrossControl Namespace Reference

Returns

error status. 0 = ERR_SUCCESS, otherwise error code.

5.1.3.200 EXTERN C CCAUXDLL_API void CCAUXDLL_CALLING_CONV
CrossControl::Lightsensor_release ( LIGHTSENSORHANDLE )

Delete the Lightsensor object.
Supported Platform(s): XL, XM, XS, XA, VC

Returns

-

Example Usage:

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

5.1.3.201 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. On the VC platform, the ranges correspond to 1000 and 4000 lux maximum value.

Supported Platform(s): XL, XM, XS, XA, VC

Parameters

| range | Operating range to set. RangeStandard or RangeExtended. |

---

150
5.1 CrossControl Namespace Reference

Returns

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

5.1.3.202 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. The average calculation works by calculating the average from a number of consecutive samples, the average window size. The reject window is used to discard sudden changes or single extreme values of the measurement. If the difference of the maximum value and the minimum value in the number of samples in the reject delta window is larger than the reject delta, those samples are discarded.

Supported Platform(s): XL, XM, XS, XA, VC

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>averageWndSize</td>
<td>The average window size in nr of samples.</td>
</tr>
<tr>
<td>rejectWndSize</td>
<td>The reject window size in nr of samples.</td>
</tr>
<tr>
<td>rejectDeltaInLux</td>
<td>The reject delta in lux.</td>
</tr>
<tr>
<td>mode</td>
<td>The configured sampling mode.</td>
</tr>
</tbody>
</table>

Returns

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

Example Usage:

```c++
// Start the average calculation background function
// This cannot be used if the automatic backlight function is running.
err = Lightsensor_startAverageCalc(pLightSensor, 5, 5, 50, SamplingModeAuto);
if (err == ERR_AVERAGE_CALC_STARTED)
    cout << "Error(" << err << ") in function startAverageCalc: "
    << GetErrorStringA(err) << endl;
else if (err != ERR_SUCCESS)
    cout << "Error(" << err << ") in function startAverageCalc: "
    << GetErrorStringA(err) << endl;
```
5.1 CrossControl Namespace Reference

5.1.3.203 EXTERN C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::Lightsensor_stopAverageCalc ( LIGHTSENSORHANDLE )

Stop average calculation.
Supported Platform(s): XL, XM, XS, XA, VC

Returns

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

Example Usage:

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

5.1.3.204 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): XL, XM, XS, XA, VC

Returns

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

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

Get backlight power status.
Supported Platform(s): XL, XM

Parameters

| status | Backlight power status. |
5.1 CrossControl Namespace Reference

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;

5.1.3.206 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): XL, XM, XS, XA, VC

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.

5.1.3.207 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 off the power if such a condition occurs. If the overcurrent remains, Can power is turned off permanently until the unit is restarted. Up to 5 consecutive over-current conditions needed until power is turned off completely. If application software turns off and on the power, the failure counter will be reset.

Supported Platform(s): XL, XM, XS, XA

Parameters

| status         | The current overcurrent detection status |
5.1 CrossControl Namespace Reference

Returns

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

Example Usage:

```cpp
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;
    }
```

5.1.3.208 EXTERN C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV CrossControl::Power_getCanPowerStatus ( POWERHANDLE, CCStatus * status )

Get can power status.

Supported Platform(s): XL, XM, XS, XA

Parameters

| status | Can power status. |

Returns

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

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

Get external fan power status.

Supported Platform(s): XL, XM
Parameters

| status | Fan power status. |

Returns

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

5.1.3.210 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 off the power if such a condition occurs. If the overcurrent remains, video power is turned off permanently until the unit is restarted. Up to 5 consecutive over-current conditions needed until power is turned off completely. If application software turns off and on the power, the failure counter will be reset.

Supported Platform(s): XL, XM, XS, XA

Parameters

| status | The current overcurrent detection status |

Returns

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

Example Usage:

```c
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;
        } // switch
```
5.1 CrossControl Namespace Reference

5.1.3.211 EXTERN C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::Power_getVideoPowerStatus ( POWERHANDLE, unsigned char * videoStatus )

Get Video power status.
Supported Platform(s): XL, XM, XS, XA

Parameters


Returns

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

Example Usage:

```cpp
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;
}
```

5.1.3.212 EXTERN C CCAUXDLL_API void CCAUXDLL_CALLING_CONV
CrossControl::Power_release ( POWERHANDLE )

Delete the Power object.
Supported Platform(s): XL, XM, XS, XA, VC

Returns

-

Example Usage:

```cpp
POWERHANDLE pPower = ::GetPower();
assert(pPower);
power_example(pPower);
Power_release(pPower);
```
5.1 CrossControl Namespace Reference

5.1.3.213 EXTERN C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::Power_setBLPowerStatus ( POWERHANDLE, CCStatus status )

Set backlight power status.
Supported Platform(s): XL, XM

Parameters

| status | Backlight power status. |

Returns

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

Example Usage:

```cpp
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;
}
```

5.1.3.214 EXTERN C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::Power_setCanPowerStatus ( POWERHANDLE, CCStatus status )

Set can power status.
Supported Platform(s): XL, XM, XS, XA

Parameters

| status | Can power status. |

Returns

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

5.1.3.215 EXTERN C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::Power_setExtFanPowerStatus ( POWERHANDLE, CCStatus status )

Set external fan power status.
Supported Platform(s): XL, XM

Parameters

| status | Fan power status. |

Returns

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

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

Set Video power status.
Supported Platform(s): XL, XM, XS, XA

Parameters


Returns

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

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

Get the configuration that is in use.
Supported Platform(s): XL, XM, XS, XA, VC

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;
}

5.1.3.218 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONVCrossControl::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): XL, XM, XS, XA, VC

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 app
}
case SuspendPending:
{
os_shutdown = false;
    cout << "A suspend request detected. App should now do what it needs to do before suspend can be performed." << endl;
    cout << "Press Enter when ready to suspend... " << endl;
    // Make sure to clear cin buffer before read
    std::cin.clear();
    std::cin.ignore(100, '\n');
cin.get();
cout << "Signalling that app is ready..." << endl;
err = PowerMgr_setAppReadyForSuspendOrShutdown(pPowerMgr);
if (err != ERR_SUCCESS)
{
    cout << "Error(" << err << ") in function setAppReadyForSuspendOrShutdown: " << GetErrorStringA(err) << endl;
}
break;
}
default:
    cout << "Error: Invalid status returned from getPowerMgrStatus!" << endl;
break;
}
//Wait for resume after notifying that we are ready to suspend
if (status == SuspendPending)
{
    bool b = false;
    while(!b)
    {
        OSSleep(100);
cout << "." << endl;

err = PowerMgr_hasResumed(pPowerMgr, &b);
if (err != ERR_SUCCESS)
{
    cout << "Error(" << err << ") in function hasResumed: " << GetErrorStringA(err) << endl;
}
cout << "System is now resumed from suspend mode!" << endl <<
"Now we will soon re-register using the registerControlledSuspendOrShutDown function!" << endl;
// Expecting to get configuration Normal after resume from suspend
CrossControl::PowerMgrConf conf;
err = PowerMgr_getConfiguration(pPowerMgr, &conf);
if (err == ERR_SUCCESS)
{
    switch (conf)
    {
    case Normal:
        cout << "PowerMgrConf is now: Normal" << endl; break;
    case ApplicationControlled:
        cout << "PowerMgrConf is now: ApplicationControlled" << endl; break;
    }
5.1 CrossControl Namespace Reference

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;
}

5.1.3.219 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): XL, XM, XS, XA, VC

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;
            break;
        }
cout << "Press Enter when ready to shutdown..." << endl;

// Make sure to clear cin buffer before read
std::cin.clear();
std::cin.ignore(150,'\n');
cin.get();
cout << "Signalling that app is ready..." << endl;
err = PowerMgr_setAppReadyForSuspendOrShutdown(pPowerMgr);

if (err != ERR_SUCCESS)
{
    cout << "Error(" << err << ") in function setAppReadyForSuspendOrShutdown: " << endl;
    return; //exit test app
}

case SuspendPending:
{
o_s_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(150,'\n');
cin.get();
cout << "Signalling that app is ready..." << endl;
err = PowerMgr_setAppReadyForSuspendOrShutdown(pPowerMgr);

if (err != ERR_SUCCESS)
{
    cout << "Error(" << err << ") in function setAppReadyForSuspendOrShutdown: " << 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: " << endl;
            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, 
    getConfiguration());
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;
}

5.1.3.220 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): XL, XM, XS, XA, VC
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" <<

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);

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

Delete the PowerMgr object.

Supported Platform(s): XL, XM, XS, XA, VC

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 which 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: " << endl;
    GetErrorStringA(err) << endl;
test_powermgr(pPowerMgr);
PowerMgr_release(pPowerMgr);

5.1.3.222 EXTERN C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::PowerMgr_setAppReadyForSuspendOrShutDown (POWERMGRHANDLE)

Acknowledgment 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): XL, XM, XS, XA, VC

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(ISO, '\n');
            cin.get();
            cout << "Signalling that app is ready..." << endl;
            err = PowerMgr_setAppReadyForSuspendOrShutDown(pPowerMgr)
if (err != ERR_SUCCESS)
{
    cout << "Error( " << err << " ) in function setAppReadyForSuspendOrShutdown: " << GetErrorStringA(err) << endl;
    return; // exit test app
}
case SuspendPending:
{
    os_shutdown = false;
    cout << "A suspend request detected. App should now do what it needs to do before suspend can be performed." << endl;
    cout << "Press Enter when ready to suspend... " << endl;
    // Make sure to clear cin buffer before read
    std::cin.clear();
    std::cin.ignore(100, '\n');
    cin.get();
    cout << "Signalling that app is ready..." << endl;
    err = PowerMgr_setAppReadyForSuspendOrShutdown(pPowerMgr);
    if (err != ERR_SUCCESS)
    {
        cout << "Error( " << err << " ) in function setAppReadyForSuspendOrShutdown: " << GetErrorStringA(err) << endl;
    }
    break;

default:
    cout << "Error: Invalid status returned from getPowerMgrStatus!" << endl;
    break;
}
// Wait for resume after notifying that we are ready to suspend
if (status == SuspendPending)
{
    bool b = false;
    while(!b)
    {
        OSSleep(100);
        cout << "." << endl;
        err = PowerMgr_hasResumed(pPowerMgr, &b);
        if (err != ERR_SUCCESS)
        {
            cout << "Error( " << err << " ) in function hasResumed: " << GetErrorStringA(err) << endl;
        }
        cout << "System is now resumed from suspend mode!" << endl;
        "Now we will soon re-register using the registerControlledSuspendOrShutDown function!" << endl;
    }
    // Expecting to get configuration Normal after resume from suspend

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

5.1.3.223 EXTERN C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::PWMOut_getPWMOutputChannelDutyCycle ( PWMOUTHANDLE,
unsigned char channel, unsigned char * duty_cycle )

Get PWM Output channel duty cycle

Supported Platform(s): VC

<table>
<thead>
<tr>
<th>channel</th>
<th>Which channel to get value from There are two output channels, 1 or 2.</th>
</tr>
</thead>
<tbody>
<tr>
<td>duty_cycle</td>
<td>The read back duty value</td>
</tr>
</tbody>
</table>

Returns

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

Example Usage:

unsigned char duty;
err = PWMOut_getPWMOutputChannelDutyCycle(pPwmOut, 1, &duty);
if (err != ERR_SUCCESS)
{
  cout << "PWMOut_getPWMOutputChannelDutyCycle: " << GetErrorStringA(err) << std::endl;
}
else
{
  cout << "PWMOut_getPWMOutputChannelDutyCycle channel 1: " << (int)duty << "% duty cycle" << std::endl;
}

5.1.3.224 EXTERN C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::PWMOut_getPWMOutputChannelFrequency ( PWMOUTHANDLE,
unsigned char channel, float * frequency )

Get PWM Output frequency for a channel
5.1 CrossControl Namespace Reference

Supported Platform(s): VC

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>channel</td>
<td>Which channel to set. There are two output channels, 1 or 2.</td>
</tr>
<tr>
<td>frequency</td>
<td>0.0 - 5000.0 Hz frequency value</td>
</tr>
</tbody>
</table>

Returns

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

Example Usage:

```c
float frequency;
err = PWMOut_getPWMOutputChannelFrequency(pPwmOut, 1, &frequency);
if (err != ERR_SUCCESS)
{
    cout << "PWMOut_getPWMOutputChannelFrequency: " << GetErrorStringA(err) << std::endl;
}
else
{
    cout << "PWMOut_getPWMOutputChannelFrequency channel 1: " << std::fixed << frequency << "Hz" << std::endl;
}
```

5.1.3.225 EXTERN C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV

CrossControl::PWMOut_getPWMOutputStatus ( PWMOUTHANDLE , unsigned char * status )

Get PWM Output status

Supported Platform(s): VC

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>Read back status value. Bit 0 represents PWM Output channel 1. Bit 1 represents PWM Output channel 2. If bit is set, it means unconnected, short to ground or over temperature detected. The output will be turned off when the error occurs. The error status remains until the output is turned on successfully.</td>
</tr>
</tbody>
</table>

Returns

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

Example Usage:

```c
unsigned char status;
err = PWMOut_getPWMOutputStatus(pPwmOut, &status);
if (err != ERR_SUCCESS)
{
    // Handle error
}
```
cout << "PWMOut_getPWMOutputStatus: " << GetErrorStringA(err) << std::endl;
} else {
    if (status & 0x01) 
        cout << "PWMOut_getPWMOutputStatus: Status Not OK for channel 1" << std::endl;
    if (status & 0x02) 
        cout << "PWMOut_getPWMOutputStatus: Status Not OK for channel 2" << std::endl;
    if ((status & 0x03) == 0) 
        cout << "PWMOut_getPWMOutputStatus: Status OK for both channels" << std::endl;
}

5.1.3.226 EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV
CrossControl::PWMOut_release ( PWMOUTHANDLE )

Delete the PWMOut object.

Supported Platform(s): VC

Returns
-

Example Usage:

PWMOUTHANDLE pPwmOut = ::GetPWMOut();
assert(pPwmOut);
pwmout_example(pPwmOut);
PWMOut_release(pPwmOut);

5.1.3.227 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::PWMOut_setPWMOutOff ( PWMOUTHANDLE, unsigned char channel )

Turn off a PWM Output channel. This function sets both frequency and duty cycle to 0.

Supported Platform(s): VC

Parameters

| channel | Which channel to set |

Returns

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

Example Usage:

err = PWMOut_setPWMOutOff(pPwmOut, 1);
if (err != ERR_SUCCESS)
{
    cout << "PWMOut_setPWMOutOff: " << GetErrorStringA(err) << std::endl;
} else
{
    cout << "PWMOut_setPWMOutOff channel 1 turned off" << std::endl;
}

5.1.3.228 EXTERN C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::PWMOut_setPWMOutputChannelDutyCycle ( PWMOUTHANDLE ,
unsigned char channel , unsigned char duty_cycle )

Set PWM Output Duty cycle for a channel
Supported Platform(s): VC

Parameters

| channel | Which channel to set | There are two output channels, 1 or 2. |
| duty_cycle | Which duty cycle (0-100 %) to use |

Returns

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

Example Usage:

err = PWMOut_setPWMOutputChannelDutyCycle(pPwmOut, 1, 50);
if (err != ERR_SUCCESS)
{
    cout << "setPWMOutputChannelDutyCycle: " << GetErrorStringA(err) << std::endl;
} else
{
    cout << "setPWMOutputChannelDutyCycle: channel 1 set to 50% duty cycle" << std::endl;
}

5.1.3.229 EXTERN C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::PWMOut_setPWMOutputChannelFrequency ( PWMOUTHANDLE ,
unsigned char channel , float frequency )

Set PWM Output frequency for a channel
Supported Platform(s): VC

Parameters

| channel | Which channel to set | There are two output channels, 1 or 2. |
| frequency | 0.0 - 5000.0 Hz frequency value |
5.1 CrossControl Namespace Reference

Returns

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

Example Usage:

```c
err = PWMOut_setPWMOutputChannelFrequency(pPwmOut, 1, (float)100.0);
if (err != ERR_SUCCESS)
{
    cout << "PWMOut_setPWMOutputChannelFrequency: " << GetErrorStringA(err) << std::endl;
}
else
{
    cout << "PWMOut_setPWMOutputChannelFrequency: channel 1 set to 100Hz" << std::endl;
}
```

5.1.3.230 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): XL, XM

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>buff</td>
<td>Text output buffer.</td>
</tr>
<tr>
<td>len</td>
<td>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.</td>
</tr>
</tbody>
</table>

Returns

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

Example Usage:

```c
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;
}
```

5.1.3.231 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::Smart_getDeviceSerial2 ( SMARTHANDLE , char * buff , int len )

Get serial number of the second secondary storage device. Use this function to access the second card if the the device uses two cards.
5.1 CrossControl Namespace Reference

Supported Platform(s): XL

### Parameters

<table>
<thead>
<tr>
<th>buff</th>
<th>Text output buffer.</th>
</tr>
</thead>
<tbody>
<tr>
<td>len</td>
<td>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.</td>
</tr>
</tbody>
</table>

### Returns

error status. 0 = ERR_SUCCESS, otherwise error code. ERR_CODE_NOT_EXIST if only one card is available on XL platform. See the enum eErr for details.

#### Example Usage:

```cpp
char serial[21];
err = Smart_getDeviceSerial2 (pSmart, serial, sizeof(serial));
if (ERR_SUCCESS == err)
    cout << "Device serial number: " << serial << endl;
else if (ERR_NOT_SUPPORTED == err)
    cout << "Smart_getDeviceSerial2 is not supported on this platform" << endl;
else
    cout << "Error(" << err << ") in function getDeviceSerial: " << GetErrorStringA(err) << endl;
```

5.1.3.232 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): XL, XM

### 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;
}

5.1.3.233 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::Smart_getInitialTime2 ( 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.

Use this function to access the second card if the device uses two cards.

Supported Platform(s): XL

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. ERR_CODE_NOT_EXIST if only one card is available on XL platform. See the enum eErr for details.

Example Usage:

time_t initialTime;
struct tm * timeinfo;
err = Smart_getInitialTime2 (pSmart, &initialTime);
if (ERR_SUCCESS == err)
{
    cout << "Device was initially timestamped on: ";
    timeinfo = localtime (&initialTime);
    cout << asctime(timeinfo) << endl;
}
else if (ERR_NOT_SUPPORTED == err)
{
    cout << "Smart_getInitialTime2 is not supported on this platform" << endl;
}
else
{
    cout << "Error(" << err << ") in function getInitialTime: " <<
        GetErrorStringA(err) << endl;
}
Get remaining lifetime of the secondary storage device.  
Supported Platform(s): XL, XM

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:

```c
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;
```

Get remaining lifetime of the second secondary storage device. Use this function to access the second card if the device uses two cards.  
Supported Platform(s): XL

Parameters

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

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. ERR_CODE_NOT_EXIST if only one card is available on XL platform. See the enum eErr for details.

Example Usage:
unsigned char life;
err = Smart_getRemainingLifeTime2 (pSmart, &life);
if (ERR_SUCCESS == err)
  cout << "Estimated remaining lifetime: " << (int)life << '%' << endl;
else if (ERR_NOT_SUPPORTED == err)
  cout << "Smart_getRemainingLifeTime2 is not supported on this platform" << endl;
else
  cout << "Error(" << err << ") in function getRemainingLifeTime: " <<
    GetErrorStringA(err) << endl;

5.1.3.236 EXTERN C CCAUXDLL_API void CCAUXDLL_CALLING_CONV
CrossControl::Smart_release ( SMARTHANDLE )
Delete the Smart object.
Supported Platform(s): XL, XM

Returns
-

Example Usage:

SMARTHANDLE pSmart = ::GetSmart();
assert(pSmart);
show_card_data(pSmart);
Smart_release(pSmart);

5.1.3.237 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:
5.1 CrossControl Namespace Reference

```c
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;
```

### 5.1.3.238 EXTERN C CCAUXDLL 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:

```c
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;
```

### 5.1.3.239 EXTERN C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV

CrossControl::Telematics_getGPRSPowerStatus ( TELEMATICSHANDLE, CCStatus * status )

Get GPRS power status.

Supported Platform(s): XM, XA, XS
5.1 CrossControl Namespace Reference

Parameters

| status | GPRS power status. |

Returns

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

Example Usage:

```cpp
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;
}
```

5.1.3.240 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:

```cpp
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;

5.1.3.241 EXTERN C CCAUXDLL eErr CCAUXDLL_CALLING_CONV
CrossControl::Telematics_getGPSAntennaStatus ( TELEMATICS_HANDLE , 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. Note, This function is only supported on revision A Telematic Addon Cards (produced before 2015-09).

Supported Platform(s): XM

Parameters

| status | GPS antenna power status. |

Returns

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

Example Usage:

```cpp
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;
```

5.1.3.242 EXTERN C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::Telematics_getGPSPowerStatus ( TELEMATICS_HANDLE , 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. |
5.1 CrossControl Namespace Reference

Returns

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

Example Usage:

```cpp
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;
```

5.1.3.243 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:

```cpp
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 CrossControl Namespace Reference

5.1.3.244 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

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

Example Usage:

```c
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;
}
```

5.1.3.245 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

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

Example Usage:

```c
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;
}

5.1.3.246 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;
}

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

Delete the Telematics object.
Supported Platform(s): XM, XA, XS
Example Usage:

```c
TELEMATICSHANDLE pTelematics = ::GetTelematics();
assert(pTelematics);
telematics_example(pTelematics);
Telematics_release(pTelematics);
```

### 5.1.3.248 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.

### 5.1.3.249 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.
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.

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.

Set GPS power status.
Supported Platform(s): XM, XA, XS

**Parameters**

| status | GPS power status. |
5.1 CrossControl Namespace Reference

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::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.

5.1.3.254 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.

5.1.3.255 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. \(0 = \text{ERR\_SUCCESS}\), otherwise error code. See the enum eErr for details.

5.1.3.256 EXTERN \text{C} \text{CCAUXDLL}\_\text{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): XL, XM, XS, XA

Parameters

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

Returns

error status. \(0 = \text{ERR\_SUCCESS}\), otherwise error code. See the enum eErr for details.

Example Usage:

```c
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;
    }
```

5.1.3.257 EXTERN \text{C} \text{CCAUXDLL}\_\text{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): XL, XM, XS, XA

Parameters

| config | The current mode. |
5.1 CrossControl Namespace Reference

Returns

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

Example Usage:

```c
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" << endl;
}
else
{
    cout << "Error(" << err << ") in function getMode: " << GetErrorStringA(err) << endl;
}
```

5.1.3.258 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): XL, XM, XS,XA

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:

```c
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;
}
```
Delete the TouchScreen object.

Supported Platform(s): XL, XM, XS, XA

Example Usage:

```c
TOUCHSCREENHANDLE pTouchScreen = ::GetTouchScreen();
assert(pTouchScreen);
touchscreen_example(pTouchScreen);
TouchScreen_release(pTouchScreen);
```

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

Supported Platform(s): XL, XM, XS, XA

<table>
<thead>
<tr>
<th>param</th>
<th>The setting to set.</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>The data value to set.</td>
</tr>
</tbody>
</table>

Returns

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

Set Touch Screen mode. Sets the mode of the USB profile.

Supported Platform(s): XL, XM, XS, XA
5.1 CrossControl Namespace Reference

Parameters

| config  | The mode to set. |

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::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): XL, XM, XS, XA

Parameters

| time  | The right click time, in milliseconds. |

Returns

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

5.1.3.263 EXTERN C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::TouchScreenCalib_autoSensorCalib ( TOUCHSCREENCALIBHANDLE )

Perform automatic sensor calibration
Supported Platform(s): VA

Returns

error status. 0 = ERR_SUCCESS, otherwise error code.

5.1.3.264 EXTERN C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::TouchScreenCalib_checkCalibrationPointFinished ( TOUCHSCREENCALIBHANDLE ,
bool * finished , unsigned char pointNr )

Check if a calibration point is finished
Supported Platform(s): XL, XM, XS, XA

Parameters

| finished  | Is current point finished? |
| pointNr  | Calibration point number (1 to total number of points) |
5.1 CrossControl Namespace Reference

Returns

error status. 0 = ERR_SUCCESS, otherwise error code.

5.1.3.265 EXTERN_C CCAUX_DLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::TouchScreenCalib getConfigParam ( TOUCHSCREENCALIBHANDLE , CalibrationConfigParam param, unsigned short * value )

Get calibration config parameters

Supported Platform(s): XL, XM, XS, XA

Parameters

<table>
<thead>
<tr>
<th>param</th>
<th>Config parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Parameter value</td>
</tr>
</tbody>
</table>

Returns

error status. 0 = ERR_SUCCESS, otherwise error code.

5.1.3.266 EXTERN_C CCAUX_DLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::TouchScreenCalib getMode ( TOUCHSCREENCALIBHANDLE , CalibrationModeSettings * mode )

Get mode of front controller.

Supported Platform(s): XL, XM, XS, XA

Parameters

| mode | Current calibration mode |

Returns

error status. 0 = ERR_SUCCESS, otherwise error code.

5.1.3.267 EXTERN_C CCAUX_DLL_API void CCAUXDLL_CALLING_CONV
CrossControl::TouchScreenCalib release ( TOUCHSCREENCALIBHANDLE )

Delete the TouchScreenCalib object.

Supported Platform(s): XL, XM, XS, XA

Returns

-
5.1 CrossControl Namespace Reference

5.1.3.268 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::TouchScreenCalib_setCalibrationPoint ( TOUCHSCREENCALIBHANDLE , unsigned char pointNr )

Set calibration point
Supported Platform(s): XL, XM, XS, XA

Parameters

| pointNr | Calibration point number (1 to total number of points) |

Returns

error status. 0 = ERR_SUCCESS, otherwise error code.

5.1.3.269 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::TouchScreenCalib_setConfigParam ( TOUCHSCREENCALIBHANDLE , CalibrationConfigParam param, unsigned short value )

Set calibration config parameters
Supported Platform(s): XL, XM, XS, XA

Parameters

| param | Config parameter |
| value | parameter value |

Returns

error status. 0 = ERR_SUCCESS, otherwise error code.

5.1.3.270 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::TouchScreenCalib_setMode ( TOUCHSCREENCALIBHANDLE , CalibrationModeSettings mode )

Set mode of front controller.
Supported Platform(s): XL, XM, XS, XA

Parameters

| mode | Selected calibration mode |

Returns

error status. 0 = ERR_SUCCESS, otherwise error code.
5.1 CrossControl Namespace Reference

5.1.3.271 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): XL, 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.

5.1.3.272 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): XL, XM (Windows)

Parameters

<table>
<thead>
<tr>
<th>bmpBuffer</th>
<th>Bitmap ram buffer allocated by the API, has to be deallocated with freeBmpBuffer() by the application.</th>
</tr>
</thead>
<tbody>
<tr>
<td>bmpBufSize</td>
<td>Size of the returned bitmap buffer.</td>
</tr>
<tr>
<td>rawImgBuffer</td>
<td>Raw image buffer from takeSnapshotRaw.</td>
</tr>
<tr>
<td>rawImgBufSize</td>
<td>Size of the raw image buffer.</td>
</tr>
<tr>
<td>bInterlaced</td>
<td>Interlaced, if true the bitmap only contains every second line in the image, to save bandwidth.</td>
</tr>
<tr>
<td>bNTSCFormat</td>
<td>True if the video format in rawImageBuffer is NTSC format.</td>
</tr>
</tbody>
</table>

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for
details.
5.1 CrossControl Namespace Reference

5.1.3.273  EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::Video_freeBmpBuffer ( VIDEOHANDLE , char * bmpBuffer )

Free the memory allocated for BMP buffer.
Supported Platform(s): XL, XM (Windows)

Parameters

| bmpBuffer | The bmp buffer to free. |

Returns

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

5.1.3.274  EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::Video_getActiveChannel ( VIDEOHANDLE , VideoChannel * channel )

Get the current video channel.
Supported Platform(s): XL, XM, XS, XA, VC

Parameters

| channel | Enum defining available channels. (VC platform has only 1 channel, Analog_Channel_1) |

Returns

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

5.1.3.275  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): XL, XM

Parameters

| rKey | Red value. |
| gKey | Green value. |
| bKey | Blue value. |
5.1 CrossControl Namespace Reference

Returns

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

5.1.3.276 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): XL, XM, XS, XA, VC

Parameters

<table>
<thead>
<tr>
<th>top</th>
<th>Crop top (lines).</th>
</tr>
</thead>
<tbody>
<tr>
<td>left</td>
<td>Crop left (lines).</td>
</tr>
<tr>
<td>bottom</td>
<td>Crop bottom (lines).</td>
</tr>
<tr>
<td>right</td>
<td>Crop right (lines).</td>
</tr>
</tbody>
</table>

Returns

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

5.1.3.277 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): XL, XM, XS, XA, VC

Parameters

<table>
<thead>
<tr>
<th>decoderRegister</th>
<th>Decoder Register Address.</th>
</tr>
</thead>
<tbody>
<tr>
<td>registerValue</td>
<td>register value.</td>
</tr>
</tbody>
</table>

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.
5.1 CrossControl Namespace Reference

5.1.3.278 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): XL, 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.

5.1.3.279 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
    CrossControl::Video::getGraphicsOverlay ( VIDEOHANDLE , CCStatus * mode )

Get the current graphics overlaying mode.
Supported Platform(s): XA, XS, VC

Parameters

| mode | Overlay enable mode |

Returns

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

5.1.3.280 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): XL, XM, XS, XA, VC

Parameters

| mode | The current mode. Enabled or Disabled. |

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.
5.1.3.281 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): XL, XM, XS, XA, VC

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>width</td>
</tr>
<tr>
<td>height</td>
</tr>
<tr>
<td>frameRate</td>
</tr>
</tbody>
</table>

Returns

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

5.1.3.282 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, VC

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>rotation</td>
</tr>
</tbody>
</table>

Returns

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

5.1.3.283 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): XL, XM

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
</tr>
</tbody>
</table>
| y          | Vertical scaling (0.25-4 DeInterlace_BOB) (0.125-2 DeInterlace_-
Even, DeInterlace_Odd). |
5.1 CrossControl Namespace Reference

Returns

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

5.1.3.284 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::Video::getStatus (_VIDEOHANDLE, unsigned char * status )

Video status byte.

Supported Platform(s): XL, XM, XS, XA, VC

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.

5.1.3.285 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): XL, XM, XS, XA, VC

Parameters

| topLeftX | Top left X coordinate on screen. |
| topLeftY | Top left Y coordinate on screen. |
| bottomRigthX | Bottom right X coordinate on screen. |
| bottomRigthY | Bottom right Y coordinate on screen. |

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.
5.1 CrossControl Namespace Reference

5.1.3.286 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): XL, XM, XS, XA, VC

Parameters

| standard | Video standard |

Returns

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

5.1.3.287 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): XL, XM, XS, XA, VC

Parameters

| deviceNr | Device to connect to (1,2). Select one of 2 devices to connect to. (VC platform has only 1 device) |

Returns

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

5.1.3.288 EXTERN C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::Video.minimize ( VIDEOHANDLE )

Minimizes the video area. Restore with restore() call.
Supported Platform(s): XL, XM, XS, XA, VC

Returns

error status. 0 = ERR_SUCCESS, otherwise error code. See the enum eErr for details.
5.1 CrossControl Namespace Reference

5.1.3.289 EXTERN C CCAUXDLL_API void CCAUXDLL_CALLING_CONV
CrossControl::Video_release ( VIDEOHANDLE )

Delete the Video object.
Supported Platform(s): XL, XM, XS, XA, VC

Returns -

5.1.3.290 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): XL, XM, XS, XA, VC

Returns

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

5.1.3.291 EXTERN C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::Video_setActiveChannel ( VIDEOHANDLE, VideoChannel channel )

Sets the active video channel.
Supported Platform(s): XL, XM, XS, XA, VC

Parameters

| channel | Enum defining available channels. (VC platform has only 1 channel, Analog_Channel_1) |

Returns

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

5.1.3.292 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): XL, XM
5.1 CrossControl Namespace Reference

Parameters

<table>
<thead>
<tr>
<th>rKey</th>
<th>Red key value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>gKey</td>
<td>Green key value.</td>
</tr>
<tr>
<td>bKey</td>
<td>Blue key value.</td>
</tr>
</tbody>
</table>

Returns

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

5.1.3.293 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::Video_setCropping ( VIDEOHANDLE , unsigned char top , unsigned char left , unsigned char bottom , unsigned char right )

Crop video image. Note that the video chip manual says the following about horizontal cropping: The number of pixels of active video must be an even number. The parameters top and bottom are internally converted to an even number. This is due to the input video being interlaced, a pair of odd/even lines are always cropped together. On XA/XS platforms, cropping from top/bottom on device 2 (channels 3 and 4) is not supported.

Supported Platform(s): XL, XM, XS, XA, VC

Parameters

<table>
<thead>
<tr>
<th>top</th>
<th>Crop top (0-255 lines).</th>
</tr>
</thead>
<tbody>
<tr>
<td>left</td>
<td>Crop left (0-127 lines).</td>
</tr>
<tr>
<td>bottom</td>
<td>Crop bottom (0-255 lines).</td>
</tr>
<tr>
<td>right</td>
<td>Crop right (0-127 lines).</td>
</tr>
</tbody>
</table>

Returns

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

5.1.3.294 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): XL, XM, XS, XA, VC

Parameters

| decoderRegister | Decoder Register Address.     |

199
5.1 CrossControl Namespace Reference

<table>
<thead>
<tr>
<th>register-Value</th>
<th>register value.</th>
</tr>
</thead>
</table>

Returns

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

5.1.3.295 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): XL, XM

Parameters

<table>
<thead>
<tr>
<th>mode</th>
<th>The mode to set. See enum DeInterlaceMode for descriptions of the modes.</th>
</tr>
</thead>
</table>

Returns

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

5.1.3.296 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::Video_setGraphicsOverlay ( VIDEOHANDLE, CCStatus mode )

Enable or disable overlaying of graphics on top of video.
Supported Platform(s): XA, XS, VC

Parameters

<table>
<thead>
<tr>
<th>mode</th>
<th>Overlay enable mode</th>
</tr>
</thead>
</table>

Returns

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

5.1.3.297 EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV
CrossControl::Video_setMirroring ( VIDEOHANDLE, CCStatus mode )

Enable or disable mirroring of the video image.
5.1 CrossControl Namespace Reference

Supported Platform(s): XL, XM, XS, XA, VC

Parameters

| mode | The mode to set. Enabled or Disabled. |

Returns

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

5.1.3.298 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, VC

Parameters

| rotation | Enum defining the rotation to set. |

Returns

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

5.1.3.299 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): XL, 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 CrossControl Namespace Reference

5.1.3.300  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): XL, XM, XS, XA, VC

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>topLeftX</td>
<td>Top left X coordinate on screen.</td>
</tr>
<tr>
<td>topLeftY</td>
<td>Top left Y coordinate on screen.</td>
</tr>
<tr>
<td>bottomRightX</td>
<td>Bottom right X coordinate on screen.</td>
</tr>
<tr>
<td>bottomRightY</td>
<td>Bottom right Y coordinate on screen.</td>
</tr>
</tbody>
</table>

Returns

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

5.1.3.301  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, VC

Returns

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

5.1.3.302  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): XL, XM, XS, XA, VC

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>show</td>
<td>True shows the video image.</td>
</tr>
</tbody>
</table>
5.1 CrossControl Namespace Reference

Returns

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

5.1.3.303 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): XL, XM (Windows)

Parameters

<table>
<thead>
<tr>
<th>path</th>
<th>The file path to where the image should be stored.</th>
</tr>
</thead>
<tbody>
<tr>
<td>bInterlaced</td>
<td>If true the bitmap only contains every second line in the image, to save bandwidth.</td>
</tr>
</tbody>
</table>

Returns

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

5.1.3.304 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): XL, XM (Windows)

Parameters

<table>
<thead>
<tr>
<th>bmpBuffer</th>
<th>Bitmap ram buffer allocated by the API, has to be deallocated with freeBmpBuffer() by the application.</th>
</tr>
</thead>
<tbody>
<tr>
<td>bmpBufSize</td>
<td>Size of the returned bitmap buffer.</td>
</tr>
<tr>
<td>bInterlaced</td>
<td>If true the bitmap only contains every second line in the image, to save bandwidth.</td>
</tr>
<tr>
<td>bNTSC-Format</td>
<td>True if the video format in rawImageBuffer is NTSC format.</td>
</tr>
</tbody>
</table>
5.1 CrossControl Namespace Reference

Returns

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

5.1.3.305 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): XL, XM (Windows)

Parameters

<table>
<thead>
<tr>
<th>rawImgBuffer</th>
<th>Buffer for image to be stored in.</th>
</tr>
</thead>
<tbody>
<tr>
<td>rawImgBuffSize</td>
<td>Size of the buffer.</td>
</tr>
<tr>
<td>bInterlaced</td>
<td>If true the bitmap only contains every second line in the image, to save bandwidth.</td>
</tr>
</tbody>
</table>

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 CrossControl Namespace Reference

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)
6.2 BuzzerSetup Struct Reference

6.1.4 unsigned long RunTime_70_80
Total runtime in range 60 to 70 deg C (minutes)

6.1.5 unsigned long RunTime_Above80
Total runtime in range 70 to 80 deg C (minutes)

6.1.6 unsigned long RunTime_m20
Total running time on battery power (minutes)

6.1.7 unsigned long RunTime_m20_0
Total runtime below -20 deg C (minutes)

6.1.8 unsigned long TotRunTimeBattery
Total running time on main power (minutes)

6.1.9 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
6.6 received_video Struct Reference

6.5.2 unsigned char nrOfPulses
Pulses per sequences

6.5.3 unsigned char offTime
LED off time in 10ms

6.5.4 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>
6.7 TimerType Struct Reference

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
Total running time (minutes)

6.7.2.7 unsigned long TotSuspTime
The documentation for this struct was generated from the following file:

- IncludeFiles/CCAuxTypes.h
# 6.8 UpgradeStatus Struct Reference

```c
#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

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

**Data Fields**

- `unsigned char major`
- `unsigned char minor`
- `unsigned char release`
- `unsigned char build`
6.10 video_dec_command Struct Reference

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)
7.1 IncludeFiles/About.h File Reference

- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getMainHWversion (ABOUTHANDLE, char *buff, int len)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getMainProdRev (ABOUTHANDLE, char *buff, int len)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getMainProdArtNr (ABOUTHANDLE, char *buff, int len)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getNrOfETHConnections (ABOUTHANDLE, unsigned char *NrOfConnections)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getNrOfCANConnections (ABOUTHANDLE, unsigned char *NrOfConnections)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getNrOfVideoConnections (ABOUTHANDLE, unsigned char *NrOfConnections)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getNrOfUSBConnections (ABOUTHANDLE, unsigned char *NrOfConnections)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getNrOfSerialConnections (ABOUTHANDLE, unsigned char *NrOfConnections)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getNrOfDigIOConnections (ABOUTHANDLE, unsigned char *NrOfConnections)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getIsDisplayAvailable (ABOUTHANDLE, bool *available)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getIsTouchScreenAvailable (ABOUTHANDLE, bool *available)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getDisplayResolution (ABOUTHANDLE, char *buff, int len)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getAddOnPCBSerial (ABOUTHANDLE, char *buff, int len)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getAddOnPCBArt (ABOUTHANDLE, char *buff, int length)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getAddOnManufacturingDate (ABOUTHANDLE, char *buff, int len)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getAddOnHWversion (ABOUTHANDLE, char *buff, int len)
7.2 IncludeFiles/Adc.h File Reference

- 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 eErr
  CCAUXDLL_CALLING_CONV About_getNrOfCfgInConnections (ABOUTHANDLE, unsigned char *NrOfConnections)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getNrOfPWMOutConnections (ABOUTHANDLE, unsigned char *NrOfConnections)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getNrOfButtons (ABOUTHANDLE, int *numbuttons)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_getUserEepromData (ABOUTHANDLE, char *buff, unsigned short length)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV About_setUserEepromData (ABOUTHANDLE, unsigned short startpos, const char *buff, unsigned short length)

### Namespaces

- namespace CrossControl
7.3 IncludeFiles/AuxVersion.h File Reference

Typedefs

- typedef void *ADCHANDLE

Functions

- EXTERN_C CCAUXDLL_API ADCHANDLE CCAUXDLL_CALLING_CONV GetAdc (void)
- EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV Adc_release (ADCHANDLE)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Adc_getVoltage (ADCHANDLE, VoltageEnum selection, double *value)

7.3 IncludeFiles/AuxVersion.h File Reference

Namespaces

- namespace CrossControl

Typedefs

- typedef void *AUXVERSIONHANDLE

Functions

- EXTERN_C CCAUXDLL_API AUXVERSIONHANDLE CCAUXDLL_CALLING_CONV GetAuxVersion (void)
- EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV AuxVersion_release (AUXVERSIONHANDLE)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV AuxVersion_getFPGAVersion (AUXVERSIONHANDLE, unsigned char *major, unsigned char *minor, unsigned char *release, unsigned char *build)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV AuxVersion_getSSVersion (AUXVERSIONHANDLE, unsigned char *major, unsigned char *minor, unsigned char *release, unsigned char *build)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV AuxVersion_getFrontVersion (AUXVERSIONHANDLE, unsigned char *major, unsigned char *minor, unsigned char *release, unsigned char *build)
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)
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-
7.5 IncludeFiles/Battery.h File Reference

• 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, Battery_TimerType *times)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Battery_getMinMaxTemp (BATTERYHANDLE, signed short *minTemp, signed short *maxTemp)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Battery_getBatteryHWversion (BATTERYHANDLE, char *buff, int len)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Battery_getBatterySwVersion (BATTERYHANDLE, unsigned short *major, unsigned short *minor, unsigned short *release, unsigned short *build)
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
  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)
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_12VID,
  VOLTAGE_5V, VOLTAGE_3V3, VOLTAGE_VTFT, VOLTAGE_5VSTB,
  VOLTAGE_1V9, VOLTAGE_1V8, VOLTAGE_1V5, VOLTAGE_1V2,
  VOLTAGE_1V05, VOLTAGE_1V0, VOLTAGE_0V9, VOLTAGE_VREF_INT,
  VOLTAGE_24V_BACKUP, VOLTAGE_2V5, VOLTAGE_1V1, VOLTAGE_1V3_PER,
  VOLTAGE_1V3_VDDA, VOLTAGE_3V3STBY, VOLTAGE_VPMIC, VOLTAGE_VMAIN
}
- enum LightSensorOperationRange {
  RangeStandard = 0, RangeExtended = 1
}
- enum LightSensorSamplingMode {
  SamplingModeStandard = 0, SamplingModeExtended, SamplingModeAuto
}
- enum CCStatus {
  Disabled = 0, Enabled = 1
}
- enum eErr {
  ERR_SUCCESS = 0, ERR_OPEN_FAILED = 1, ERR_NOT_SUPPORTED = 2,
  ERR_UNKNOWN_FEATURE = 3,
  ERR_DATATYPE_MISMATCH = 4, ERR_CODE_NOT_EXIST = 5, ERR_BUFFER_SIZE = 6,
  ERR_IOCTL_FAILED = 7, ERR_INVALID_DATA = 8, ERR_INVALID_PARAMETER = 9,
  ERR_CREATE_THREAD = 10, ERR_IN_PROGRESS = 11,
  ERR_CHECKSUM = 12, ERR_INIT_FAILED = 13, ERR_VERIFY_FAILED
}
ERR_DEVICE_READ_DATA_FAILED = 14, 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, ERR_INVALID_D_STARTUP_TRIGGER = 34

- 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, CAN_Button_Activity = 5, CAN_OnOff_Activity = 6, CAN_Button_OnOff_Activity = 7, CI_Button_Activity = 9, CI_OnOff_Activity = 10, CI_Button_OnOff_Activity = 11, CI_CAN_Button_Activity = 13, CI_CAN_OnOff_Activity = 14, All_Events = 15, Last_trigger_conf }
- 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 }
7.10 IncludeFiles/CCPlatform.h File Reference

7.11 IncludeFiles/CfgIn.h File Reference

Namespaces

- namespace CrossControl

Typedefs

- typedef void * CFGINHANDLE
Functions

- **EXTERN_C CCAUXDLL_API**
  CFGINHANDLE
  CCAUXDLL_CALLING_CONV GetCfgIn (void)

- **EXTERN_C CCAUXDLL_API**
  void
  CCAUXDLL_CALLING_CONV CfgIn_release (CFGINHANDLE)

- **EXTERN_C CCAUXDLL_API** eErr
  CCAUXDLL_CALLING_CONV CfgIn_setCfgInMode (CFGINHANDLE, unsigned char channel, CfgInModeEnum set_mode)

- **EXTERN_C CCAUXDLL_API** eErr
  CCAUXDLL_CALLING_CONV CfgIn_getCfgInMode (CFGINHANDLE, unsigned char channel, CfgInModeEnum *get_mode)

- **EXTERN_C CCAUXDLL_API** eErr
  CCAUXDLL_CALLING_CONV CfgIn_getValue (CFGINHANDLE, unsigned char channel, unsigned short *sample_value)

- **EXTERN_C CCAUXDLL_API** eErr
  CCAUXDLL_CALLING_CONV CfgIn_getPwmValue (CFGINHANDLE, unsigned char channel, float *frequency, unsigned char *duty_cycle)

- **EXTERN_C CCAUXDLL_API** eErr
  CCAUXDLL_CALLING_CONV CfgIn_getFrequencyValue (CFGINHANDLE, unsigned char channel, float *frequency)

- **EXTERN_C CCAUXDLL_API** eErr
  CCAUXDLL_CALLING_CONV CfgIn_getMinFrequencyThreshold (CFGINHANDLE, unsigned char channel, float *frequency)

- **EXTERN_C CCAUXDLL_API** eErr
  CCAUXDLL_CALLING_CONV CfgIn_setMinFrequencyThreshold (CFGINHANDLE, unsigned char channel, float frequency)

- **EXTERN_C CCAUXDLL_API** eErr
  CCAUXDLL_CALLING_CONV CfgIn_setFrequencyFilterLevel (CFGINHANDLE, unsigned char level)

Namespaces

- namespace CrossControl

Typedefs

- typedef void * CONFIGHANDLE

Functions

- **EXTERN_C CCAUXDLL_API**
  CONFIGHANDLE
  CCAUXDLL_CALLING_CONV GetConfig ()
7.12 IncludeFiles/Config.h File Reference

- 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_getButtonFunction (CONFIGHANDLE, unsigned char button_number, ButtonConfigEnum *button_config)
- 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)
7.12 IncludeFiles/Config.h File Reference

- `EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Config_setShortButtonPressAction (HANDLE, PowerAction action)`
- `EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Config_setLongButtonPressAction (HANDLE, PowerAction action)`
- `EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Config_setOnOffSigAction (HANDLE, PowerAction action)`
- `EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Config_setFrontBtnTrigTime (HANDLE, unsigned short triggertime)`
- `EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Config_setExtOnOffSigTrigTime (HANDLE, unsigned long triggertime)`
- `EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Config_setButtonFunction (HANDLE, unsigned char button_number, ButtonConfigEnum button_config)`
- `EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Config_setSuspendMaxTime (HANDLE, unsigned short maxTime)`
- `EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Config_setCanStartupPowerConfig (HANDLE, CCStatus status)`
- `EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Config_setVideoStartupPowerConfig (HANDLE, unsigned char config)`
- `EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Config_setExtFanStartupPowerConfig (HANDLE, CCStatus status)`
- `EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Config_setStartupVoltageConfig (HANDLE, double voltage)`
- `EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Config_setHeatingTempLimit (HANDLE, signed short temperature)`
- `EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Config_setPowerOnStartup (HANDLE, CCStatus status)`
- `EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Config_setRS485Enabled (HANDLE, RS4XXPort port, bool enabled)`
- `EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Config_getRS485Enabled (HANDLE, RS4XXPort port, bool *enabled)`
- `EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Config_setOnOffTriggerMode (HANDLE, ConfigOnOffTriggerMode mode)`
7.13 IncludeFiles/Diagnostic.h File Reference

- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Config_getOnOffTriggerMode (CONFIGNHANDLE, ConfigOnOffTriggerMode *mode)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Config_getOnOffSignalState (CONFIGNHANDLE, CCStatus *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)

7.13 IncludeFiles/Diagnostic.h File Reference

Namespaces

- namespace CrossControl

Typedefs

- typedef void *DIAGNOSTICHANDLE

Functions

- EXTERN_C CCAUXDLL_API DIAGNOSTICHANDLE
  CCAUXDLL_CALLING_CONV GetDiagnostic (void)
- EXTERN_C CCAUXDLL_API void
  CCAUXDLL_CALLING_CONV Diagnostic_release (DIAGNOSTICHANDLE)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Diagnostic_getSSTemp (DIAGNOSTICHANDLE, signed short *temperature)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Diagnostic_getPCBTemp (DIAGNOSTICHANDLE, signed short *temperature)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Diagnostic_getPMTemp (DIAGNOSTICHANDLE, unsigned char index, signed short *temperature, JidaSensorType *jst)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Diagnostic_getStartupReason (DIAGNOSTICHANDLE, unsigned short *reason)
7.14 IncludeFiles/DiagnosticCodes.h File Reference

Namespaces

- namespace CrossControl

Enumerations

- enum startupReasonCodes {
  startupReasonCodeUndefined = 0x0000, startupReasonCodeButtonPress = 0x0055,
  startupReasonCodeExtCtrl = 0x00AA, startupReasonCodeMPRestart = 0x00F0,
  startupReasonCodePowerOnStartup = 0x000F, startupReasonCodeCanActivity
  = 0x003c, startupReasonCodeCIActivity = 0x00c3, startupReasonAlwaysStart
  = 0x00e1,
  startupReasonUnknownTrigger = 0x001e
}
- 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)
7.15 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)

7.16 IncludeFiles/FirmwareUpgrade.h File Reference

Namespaces

• namespace CrossControl
7.17 IncludeFiles/FrontLED.h File Reference

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)

7.17 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_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)
7.18 IncludeFiles/Lightsensor.h File Reference

Namespaces

- namespace CrossControl

Typedefs

- typedef void * LIGHTSENSORHANDLE

Functions

- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV FrontLED_setStandardColor (FRONTLEDHANDLE, CCAuxColor color)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV FrontLED_setOff (FRONTLEDHANDLE)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV FrontLED_setEnabledDuringStartup (FRONTLEDHANDLE, CCStatus status)

- EXTERN_C CCAUXDLL_API LIGHTSENSORHANDLE CCAUXDLL_CALLING_CONV GetLightsensor (void)
- EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV Lightsensor_release (LIGHTSENSORHANDLE)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Lightsensor_getIlluminance (LIGHTSENSORHANDLE, unsigned short *value)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Lightsensor_getIlluminance2 (LIGHTSENSORHANDLE, unsigned short *value, unsigned char *ch0, unsigned char *ch1)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Lightsensor_getAverageIlluminance (LIGHTSENSORHANDLE, unsigned short *value)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Lightsensor_startAverageCalc (LIGHTSENSORHANDLE, unsigned long averageWndSize, unsigned long rejectWndSize, unsigned long rejectDeltaInLux, LightSensorSamplingMode mode)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Lightsensor_stopAverageCalc (LIGHTSENSORHANDLE)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Lightsensor_getOperatingRange (LIGHTSENSORHANDLE, LightSensorOperationRange *range)
7.19 IncludeFiles/Power.h File Reference

Namespaces

• namespace CrossControl

Typedefs

• typedef void ∗POWERHANDLE

Functions

• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Lightsensor_setOperatingRange (LIGHTSENSORHANDLE, LightSensorOperationRange range)

• EXTERN_C CCAUXDLL_API void
  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)
7.20 IncludeFiles/PowerMgr.h File Reference

Namespaces

• namespace CrossControl

Typedefs

• typedef enum
  CrossControl::PowerMgrConf _PowerMgrConf
• typedef enum
  CrossControl::PowerMgrStatus _PowerMgrStatus
• typedef void ∗ POWERMGRHANDLE

Enumerations

• enum PowerMgrConf { Normal = 0, ApplicationControlled = 1, BatterySuspend = 2 }
• enum PowerMgrStatus { NoRequestsPending = 0, SuspendPending = 1, ShutdownPending = 2 }

Functions

• EXTERN_C CCAUXDLL_API POWERMGRHANDLE
  CCAUXDLL_CALLING_CONV GetPowerMgr (void)
• EXTERN_C CCAUXDLL_API void
  CCAUXDLL_CALLING_CONV PowerMgr_release (POWERMGRHANDLE)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV PowerMgr_registerControlledSuspendOrShutdown (POWERMGRHANDLE, PowerMgrConf conf)
7.21 IncludeFiles/PWMOut.h File Reference

**Namespaces**

- namespace CrossControl

**Typedefs**

- typedef void *PWMOUTHANDLE

**Functions**

- EXTERN_C CCAUXDLL_API PWMOUTHANDLE CCAUXDLL_CALLING_CONV GetPWMOut (void)
- EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV PWMOut_release (PWMOUTHANDLE)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV PWMOut_setPWMOutputChannelDutyCycle (PWMOUTHANDLE, unsigned char channel, unsigned char duty_cycle)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV PWMOut_setPWMOutputChannelFrequency (PWMOUTHANDLE, unsigned char channel, float frequency)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV PWMOut_getPWMOutputChannelDutyCycle (PWMOUTHANDLE, unsigned char channel, unsigned char *duty_cycle)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV PWMOut_getPWMOutputChannelFrequency (PWMOUTHANDLE, unsigned char channel, float *frequency)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV PWMOut_getPWMOutputStatus (PWMOUTHANDLE, unsigned char *status)
7.22 IncludeFiles/Releasenotes.dox File Reference

• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV PWMOut_setPWMOOutOff (PWMOUTHANDLE, unsigned char channel)

7.23 IncludeFiles/Smart.h File Reference

Namespaces

• namespace CrossControl

Typedefs

• typedef void ∗SMARTHANDLE

Functions

• EXTERN_C CCAUXDLL_API SMARTHANDLE
  CCAUXDLL_CALLING_CONV GetSmart (void)
• EXTERN_C CCAUXDLL_API void
  CCAUXDLL_CALLING_CONV Smart_release (SMARTHANDLE)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Smart_getRemainingLifeTime (SMARTHANDLE, unsigned char ∗lifetimepercent)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Smart_getRemainingLifeTime2 (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_getDeviceSerial2 (SMARTHANDLE, char ∗buff, int len)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Smart_getInitialTime (SMARTHANDLE, time_t ∗time)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Smart_getInitialTime2 (SMARTHANDLE, time_t ∗time)
Namespaces

- namespace CrossControl

Typedefs

- typedef void * TELEMATICS_HANDLE

Functions

- EXTERN_C CCAUXDLL_API TELEMATICS_HANDLE CCAUXDLL_CALLING_CONV GetTelematics (void)
- EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV Telematics_release (TELEMATICS_HANDLE)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Telematics_getTelematicsAvailable (TELEMATICS_HANDLE, CCStatus *status)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Telematics_getGPRS(startUpPowerStatus (TELEMATICS_HANDLE, CCStatus *status)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Telematics_getGPRS(startUpPowerStatus (TELEMATICS_HANDLE, CCStatus *status)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Telematics_getWLANPowerStatus (TELEMATICS_HANDLE, CCStatus *status)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Telematics_getWLAN(startUpPowerStatus (TELEMATICS_HANDLE, CCStatus *status)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Telematics_getBTPowerStatus (TELEMATICS_HANDLE, CCStatus *status)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Telematics_getBT(startUpPowerStatus (TELEMATICS_HANDLE, CCStatus *status)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Telematics_getGPSPowerStatus (TELEMATICS_HANDLE, CCStatus *status)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Telematics_getGPS(startUpPowerStatus (TELEMATICS_HANDLE, CCStatus *status)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Telematics_getGPSAntennaStatus (TELEMATICS_HANDLE, CCStatus *status)
7.25 IncludeFiles/TouchScreen.h File Reference

Namespaces

• namespace CrossControl

Typedefs

• typedef void * TOUCHSCREENHANDLE

Enumerations

• enum TouchScreenModeSettings { MOUSE_NEXT_BOOT = 0, TOUCH_NEXT_BOOT = 1, MOUSE_NOW = 2, TOUCH_NOW = 3 }

• enum TSAdvancedSettingsParameter { TS_RIGHT_CLICK_TIME = 0, TS_LOW_LEVEL = 1, TS_UNTOUCHLEVEL = 2, TS_DEBOUNCE_TIME = 3, TS_DEBOUNCE_TIMEOUT_TIME = 4, TS_DOUBLECLICK_MAX_CLICK_TIME = 5, TS_DOUBLE_CLICK_TIME = 6, TS_MAX_RIGHTCLICK_DISTANCE = 7, TS_USE_DEJITTER = 8, TS_CALIBTATION_WIDTH = 9, TS_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 (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)

7.26 IncludeFiles/TouchScreenCalib.h File Reference

Namespaces

• namespace CrossControl

Typedefs

• typedef void * TOUCHSCREENCALIBHANDLE

Enumerations

• enum CalibrationModeSettings {
  MODE_UNKNOWN = 0, MODE_NORMAL = 1, MODE_CALIBRATION_5P = 2, MODE_CALIBRATION_9P = 3, 
  MODE_CALIBRATION_13P = 4 }
enum CalibrationConfigParam {
  CONFIG_CALIBRATION_WITH = 0, CONFIG_CALIBRATION_MEASUREMENTS = 1,
  CONFIG_5P_CALIBRATION_POINT_BORDER = 2, CONFIG_13P_CALIBRATION_POINT_BORDER = 3,
  CONFIG_13P_CALIBRATION_TRANSITION_MIN = 4, CONFIG_13P_CALIBRATION_TRANSITION_MAX = 5
}

Functions

- EXTERN_C CCAUXDLL_API TOUCHSCREENCALIBHANDLE
  CCAUXDLL_CALLING_CONV GetTouchScreenCalib (void)
- EXTERN_C CCAUXDLL_API void
  CCAUXDLL_CALLING_CONV TouchScreenCalib_release (TOUCHSCREENCALIBHANDLE)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV TouchScreenCalib_setMode (TOUCHSCREENCALIBHANDLE, CalibrationModeSettings mode)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV TouchScreenCalib_getMode (TOUCHSCREENCALIBHANDLE, CalibrationModeSettings *mode)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV TouchScreenCalib_setCalibrationPoint (TOUCHSCREENCALIBHANDLE, unsigned char pointNr)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV TouchScreenCalib_checkCalibrationPointFinished (TOUCHSCREENCALIBHANDLE, bool *finished, unsigned char pointNr)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV TouchScreenCalib_getConfigParam (TOUCHSCREENCALIBHANDLE, CalibrationConfigParam param, unsigned short *value)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV TouchScreenCalib_setConfigParam (TOUCHSCREENCALIBHANDLE, CalibrationConfigParam param, unsigned short value)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV TouchScreenCalib_autoSensorCalib (TOUCHSCREENCALIBHANDLE)

7.27 IncludeFiles/Video.h File Reference

Namespaces

- namespace CrossControl

Typedefs

- typedef void * VIDEOHANDLE
Functions

- EXTERN_C CCAUXDLL_API VIDEOHANDLE CCAUXDLL_CALLING_CONV GetVideo (void)
- EXTERN_C CCAUXDLL_API void CCAUXDLL_CALLING_CONV Video_release (VIDEOHANDLE)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_init (VIDEOHANDLE, unsigned char deviceNr)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_showVideo (VIDEOHANDLE, bool show)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_setDeInterlaceMode (VIDEOHANDLE, DeInterlaceMode mode)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_getDeInterlaceMode (VIDEOHANDLE, DeInterlaceMode *mode)
- EXTERN_C CCAUXDLL_API eErr CCAUXDLL_CALLING_CONV Video_setMirroring (VIDEOHANDLE, 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 *bottomRightX, unsigned short *bottomRightY)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_getVideoStandard (VIDEOHANDLE, videoStandard *standard)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_getStatus (VIDEOHANDLE, unsigned char *status)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_setScaling (VIDEOHANDLE, float x, float y)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_getScaling (VIDEOHANDLE, float *x, float *y)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_activateSnapshot (VIDEOHANDLE, bool activate)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_takeSnapshot (VIDEOHANDLE, const char *path, bool bInterlaced)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_takeSnapshotRaw (VIDEOHANDLE, char *rawImgBuffer, unsigned long rawImgBuffSize, bool bInterlaced)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_takeSnapshotBmp (VIDEOHANDLE, char **bmpBuffer, unsigned long *bmpBufSize, bool bInterlaced, bool bNTSCFormat)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_createBitmap (VIDEOHANDLE, char **bmpBuffer, unsigned long *bmpBufSize, const char *rawImgBuffer, unsigned long rawImgBufSize, bool bInterlaced, bool bNTSCFormat)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_freeBmpBuffer (VIDEOHANDLE, char *bmpBuffer)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_minimize (VIDEOHANDLE)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_restore (VIDEOHANDLE)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_setDecoderReg (VIDEOHANDLE, unsigned char decoderRegister, unsigned char registerValue)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_getDecoderReg (VIDEOHANDLE, unsigned char decoderRegister, unsigned char *registerValue)
• EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_setCropping (VIDEOHANDLE, unsigned char top, unsigned char left, unsigned char bottom, unsigned char right)
7.27 IncludeFiles/Video.h File Reference

- 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)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_setGraphicsOverlay (VIDEOHANDLE, CCStatus mode)
- EXTERN_C CCAUXDLL_API eErr
  CCAUXDLL_CALLING_CONV Video_getGraphicsOverlay (VIDEOHANDLE, CCStatus *mode)