

# multichannel\* systems

McsUsbNet.dll

Version 5.0.12

---

Multi Channel Systems MCS GmbH  
Aspenhastrasse 21  
72770 Reutlingen  
Germany  
Fon +49-71 21-90 92 5 - 0  
Fax +49-71 21-90 92 5 -11  
[info@multichannelsystems.com](mailto:info@multichannelsystems.com)  
[www.multichannelsystems.com](http://www.multichannelsystems.com)

Generated by Doxygen 1.8.18

<b>1 McsUsbNet.dll for MCS USB devices</b>	<b>1</b>
1.1 Introduction	1
1.2 System requirements	2
1.3 Connecting to an MCS device	2
<b>2 Device Classes</b>	<b>2</b>
2.1 The MCS FluidControl Device	2
2.1.1 Introduction	2
2.1.2 Access to the FluidControl device	3
2.2 MCS-USB-Sw2to64 device	3
<b>3 Function Classes</b>	<b>4</b>
<b>4 Data ACQuisition (DACQ) Devices</b>	<b>5</b>
<b>5 The MCS Robo Device</b>	<b>6</b>
5.1 Introduction	6
<b>6 STG200x &amp; STG400x STimulus Generator</b>	<b>6</b>
6.1 Introduction	6
6.2 Download mode	6
6.2.1 Memory Layout and Trigger Setup	7
6.3 Streaming mode	8
6.3.1 Memory Layout and Trigger Setup	9
<b>7 Namespace Index</b>	<b>11</b>
7.1 Namespace List	11
<b>8 Hierarchical Index</b>	<b>11</b>
8.1 Class Hierarchy	11
<b>9 Class Index</b>	<b>16</b>
9.1 Class List	16
<b>10 Namespace Documentation</b>	<b>22</b>
10.1 Mcs Namespace Reference	22
10.2 Mcs::Usb Namespace Reference	22
10.2.1 Enumeration Type Documentation	26
10.2.2 Function Documentation	27
<b>11 Class Documentation</b>	<b>29</b>
11.1 CW2100_FunctionNet::AudioChannelsNet Struct Reference	29
11.1.1 Member Data Documentation	29
11.2 BatteryState Class Reference	29
11.2.1 Property Documentation	29
11.3 BesselFilterHighPassNet Class Reference	30

11.3.1 Constructor & Destructor Documentation	30
11.4 BesselFilterLowPassNet Class Reference	30
11.4.1 Constructor & Destructor Documentation	31
11.5 ButterworthFilterHighPassNet Class Reference	31
11.5.1 Constructor & Destructor Documentation	31
11.6 ButterworthFilterLowPassNet Class Reference	32
11.6.1 Constructor & Destructor Documentation	32
11.7 CChannelTestDeviceNet Class Reference	32
11.7.1 Constructor & Destructor Documentation	33
11.7.2 Member Function Documentation	33
11.8 CCMOSMea_FunctionNet Class Reference	33
11.8.1 Constructor & Destructor Documentation	35
11.8.2 Member Function Documentation	36
11.9 CCMOSMeaDeviceNet Class Reference	44
11.9.1 Constructor & Destructor Documentation	45
11.9.2 Member Function Documentation	45
11.9.3 Property Documentation	47
11.10 CCreateFilterNet Class Reference	47
11.10.1 Constructor & Destructor Documentation	48
11.10.2 Member Function Documentation	48
11.10.3 Property Documentation	49
11.11 CDacCalibrationFunctionNet Class Reference	49
11.11.1 Detailed Description	50
11.11.2 Constructor & Destructor Documentation	50
11.11.3 Member Function Documentation	50
11.12 CDacqGroupChannelGenericSelectionNet Class Reference	51
11.12.1 Constructor & Destructor Documentation	51
11.13 CDacqGroupChannelSelectionNet Class Reference	52
11.13.1 Constructor & Destructor Documentation	52
11.14 CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet > Class Template Reference	52
11.14.1 Constructor & Destructor Documentation	53
11.14.2 Member Function Documentation	53
11.15 CDeviceGroupChannelInfoGenericNet Class Reference	55
11.15.1 Constructor & Destructor Documentation	55
11.16 CDeviceGroupChannelInfoNet Class Reference	55
11.16.1 Constructor & Destructor Documentation	56
11.17 CDeviceGroupChannelInfoSCUNet Class Reference	56
11.17.1 Constructor & Destructor Documentation	56
11.18 CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumTemplateNet > Class Template Reference	57
11.18.1 Constructor & Destructor Documentation	57
11.18.2 Member Data Documentation	57

11.19 CDeviceGroupChannelInfoW2100Net Class Reference . . . . .	57
11.19.1 Constructor & Destructor Documentation . . . . .	58
11.20 CDigOutStimulatorFunctionNet Class Reference . . . . .	58
11.20.1 Detailed Description . . . . .	59
11.20.2 Constructor & Destructor Documentation . . . . .	59
11.20.3 Member Function Documentation . . . . .	59
11.21 CEncapsulatorDeviceNet Class Reference . . . . .	62
11.21.1 Detailed Description . . . . .	62
11.21.2 Constructor & Destructor Documentation . . . . .	63
11.21.3 Member Function Documentation . . . . .	63
11.22 CExternDTesterDeviceNet Class Reference . . . . .	63
11.22.1 Detailed Description . . . . .	63
11.22.2 Constructor & Destructor Documentation . . . . .	64
11.22.3 Member Function Documentation . . . . .	64
11.23 CFilterCoefficientsNet Class Reference . . . . .	65
11.23.1 Constructor & Destructor Documentation . . . . .	65
11.23.2 Member Function Documentation . . . . .	66
11.23.3 Property Documentation . . . . .	67
11.24 CFilterConfigurationNet Class Reference . . . . .	68
11.24.1 Constructor & Destructor Documentation . . . . .	68
11.24.2 Member Function Documentation . . . . .	69
11.25 CFilterConfigurationRegisterNet Class Reference . . . . .	70
11.25.1 Constructor & Destructor Documentation . . . . .	70
11.25.2 Member Function Documentation . . . . .	70
11.26 CFilterPropertyNet Class Reference . . . . .	71
11.26.1 Constructor & Destructor Documentation . . . . .	72
11.26.2 Member Function Documentation . . . . .	72
11.26.3 Property Documentation . . . . .	72
11.27 CFluidControlDeviceNet Class Reference . . . . .	73
11.27.1 Detailed Description . . . . .	74
11.27.2 Constructor & Destructor Documentation . . . . .	74
11.27.3 Member Function Documentation . . . . .	75
11.27.4 Property Documentation . . . . .	80
11.28 CFYIDeviceNet Class Reference . . . . .	80
11.28.1 Detailed Description . . . . .	80
11.28.2 Constructor & Destructor Documentation . . . . .	80
11.28.3 Property Documentation . . . . .	81
11.29 CGenericDevelopDeviceNet Class Reference . . . . .	81
11.29.1 Detailed Description . . . . .	88
11.29.2 Constructor & Destructor Documentation . . . . .	88
11.29.3 Member Function Documentation . . . . .	88
11.30 CGilsonDeviceNet Class Reference . . . . .	98

---

11.30.1 Detailed Description . . . . .	99
11.30.2 Constructor & Destructor Documentation . . . . .	99
11.30.3 Member Function Documentation . . . . .	99
11.30.4 Member Data Documentation . . . . .	100
11.31 CHiClampDeviceNet Class Reference . . . . .	100
11.31.1 Detailed Description . . . . .	101
11.31.2 Constructor & Destructor Documentation . . . . .	101
11.31.3 Property Documentation . . . . .	101
11.32 CHLADacqNet Class Reference . . . . .	101
11.32.1 Constructor & Destructor Documentation . . . . .	102
11.33 CHLADeviceNet Class Reference . . . . .	102
11.33.1 Detailed Description . . . . .	102
11.33.2 Constructor & Destructor Documentation . . . . .	102
11.33.3 Property Documentation . . . . .	103
11.34 CMcsUsbDacqNet::CHWInfo Class Reference . . . . .	103
11.34.1 Detailed Description . . . . .	103
11.34.2 Constructor & Destructor Documentation . . . . .	103
11.34.3 Member Function Documentation . . . . .	104
11.35 CIntanMea_FunctionNet Class Reference . . . . .	105
11.35.1 Constructor & Destructor Documentation . . . . .	106
11.35.2 Member Function Documentation . . . . .	106
11.36 CInterfaceboardFunctionNet Class Reference . . . . .	107
11.36.1 Detailed Description . . . . .	108
11.36.2 Constructor & Destructor Documentation . . . . .	108
11.36.3 Member Function Documentation . . . . .	109
11.37 CLIH3DeviceNet Class Reference . . . . .	109
11.37.1 Detailed Description . . . . .	111
11.37.2 Constructor & Destructor Documentation . . . . .	111
11.37.3 Member Function Documentation . . . . .	111
11.37.4 Property Documentation . . . . .	116
11.38 CMcsBus_AxisParametersNet Class Reference . . . . .	116
11.38.1 Constructor & Destructor Documentation . . . . .	117
11.38.2 Member Function Documentation . . . . .	117
11.39 CMcsBus_ExtensionNet Class Reference . . . . .	118
11.39.1 Constructor & Destructor Documentation . . . . .	118
11.39.2 Member Function Documentation . . . . .	119
11.40 CMcsBus_FYIExtensionNet Class Reference . . . . .	119
11.40.1 Constructor & Destructor Documentation . . . . .	119
11.40.2 Member Function Documentation . . . . .	120
11.41 CMcsBus_MotorControlNet Class Reference . . . . .	121
11.41.1 Constructor & Destructor Documentation . . . . .	124
11.41.2 Member Function Documentation . . . . .	124

---

11.42 CMcsBus_SensorNet Class Reference . . . . .	138
11.42.1 Constructor & Destructor Documentation . . . . .	140
11.42.2 Member Function Documentation . . . . .	140
11.43 CMcsBus_TempSensorNet Class Reference . . . . .	148
11.43.1 Constructor & Destructor Documentation . . . . .	149
11.43.2 Member Function Documentation . . . . .	149
11.44 CMcsBus_VoltageModeNet Class Reference . . . . .	150
11.44.1 Constructor & Destructor Documentation . . . . .	151
11.44.2 Member Function Documentation . . . . .	152
11.45 CMcsBusNet Class Reference . . . . .	155
11.45.1 Constructor & Destructor Documentation . . . . .	155
11.45.2 Member Function Documentation . . . . .	156
11.46 CMcsUsbDacqNet Class Reference . . . . .	159
11.46.1 Detailed Description . . . . .	164
11.46.2 Constructor & Destructor Documentation . . . . .	165
11.46.3 Member Function Documentation . . . . .	165
11.46.4 Member Data Documentation . . . . .	206
11.46.5 Property Documentation . . . . .	206
11.46.6 Event Documentation . . . . .	207
11.47 CMcsUsbDeviceStatePushFunctionNet Class Reference . . . . .	207
11.47.1 Constructor & Destructor Documentation . . . . .	207
11.47.2 Member Function Documentation . . . . .	208
11.47.3 Event Documentation . . . . .	208
11.48 CMcsUsbDeviceStatePushNet Class Reference . . . . .	208
11.48.1 Constructor & Destructor Documentation . . . . .	208
11.48.2 Member Function Documentation . . . . .	209
11.48.3 Event Documentation . . . . .	209
11.49 CMcsUsbFactoryNet Class Reference . . . . .	209
11.49.1 Constructor & Destructor Documentation . . . . .	211
11.49.2 Member Function Documentation . . . . .	211
11.49.3 Member Data Documentation . . . . .	217
11.50 CMcsUsbFunctionNet Class Reference . . . . .	217
11.50.1 Constructor & Destructor Documentation . . . . .	218
11.50.2 Member Function Documentation . . . . .	218
11.50.3 Member Data Documentation . . . . .	219
11.51 CMcsUsbFunctionPointerContainer Class Reference . . . . .	219
11.52 CMcsUsbListEntryNet Class Reference . . . . .	219
11.52.1 Detailed Description . . . . .	220
11.52.2 Constructor & Destructor Documentation . . . . .	220
11.52.3 Member Function Documentation . . . . .	220
11.52.4 Property Documentation . . . . .	222
11.53 CMcsUsbListNet Class Reference . . . . .	223

---

11.53.1 Detailed Description . . . . .	224
11.53.2 Constructor & Destructor Documentation . . . . .	224
11.53.3 Member Function Documentation . . . . .	224
11.53.4 Property Documentation . . . . .	225
11.53.5 Event Documentation . . . . .	225
11.54 CMcsUsbNet Class Reference . . . . .	226
11.54.1 Detailed Description . . . . .	230
11.54.2 Constructor & Destructor Documentation . . . . .	230
11.54.3 Member Function Documentation . . . . .	230
11.54.4 Member Data Documentation . . . . .	242
11.54.5 Property Documentation . . . . .	247
11.55 CMcsUsbPointerContainer Class Reference . . . . .	247
11.56 CMEA2100x256FunctionNet Class Reference . . . . .	247
11.56.1 Detailed Description . . . . .	247
11.56.2 Constructor & Destructor Documentation . . . . .	247
11.56.3 Member Function Documentation . . . . .	248
11.57 CMeaAudioFunctionNet Class Reference . . . . .	249
11.57.1 Constructor & Destructor Documentation . . . . .	249
11.57.2 Member Function Documentation . . . . .	250
11.58 CMeaCleanDeviceNet Class Reference . . . . .	252
11.58.1 Detailed Description . . . . .	253
11.58.2 Constructor & Destructor Documentation . . . . .	253
11.58.3 Member Function Documentation . . . . .	254
11.59 CMeaCoatDeviceNet Class Reference . . . . .	256
11.59.1 Detailed Description . . . . .	257
11.59.2 Constructor & Destructor Documentation . . . . .	258
11.59.3 Member Function Documentation . . . . .	258
11.60 CMeaDeviceNet Class Reference . . . . .	262
11.60.1 Detailed Description . . . . .	263
11.60.2 Constructor & Destructor Documentation . . . . .	263
11.60.3 Member Function Documentation . . . . .	264
11.60.4 Property Documentation . . . . .	269
11.61 CMeaDigitalDataFunctionNet Class Reference . . . . .	270
11.61.1 Constructor & Destructor Documentation . . . . .	270
11.61.2 Member Function Documentation . . . . .	271
11.62 CMeaFeedbackFunctionNet Class Reference . . . . .	272
11.62.1 Constructor & Destructor Documentation . . . . .	273
11.62.2 Member Function Documentation . . . . .	273
11.63 CMeaImpedanceDeviceNet Class Reference . . . . .	276
11.63.1 Constructor & Destructor Documentation . . . . .	276
11.63.2 Member Function Documentation . . . . .	277
11.64 CMeasureTableDeviceNet Class Reference . . . . .	277

11.64.1 Detailed Description . . . . .	278
11.64.2 Constructor & Destructor Documentation . . . . .	278
11.64.3 Property Documentation . . . . .	278
11.65 CMeaSwitchDeviceNet Class Reference . . . . .	278
11.65.1 Detailed Description . . . . .	279
11.65.2 Constructor & Destructor Documentation . . . . .	279
11.65.3 Member Function Documentation . . . . .	279
11.66 CMeaUSBDeviceNet Class Reference . . . . .	280
11.66.1 Detailed Description . . . . .	281
11.66.2 Constructor & Destructor Documentation . . . . .	281
11.67 CMeFunctionNet Class Reference . . . . .	281
11.67.1 Detailed Description . . . . .	282
11.67.2 Constructor & Destructor Documentation . . . . .	282
11.67.3 Member Function Documentation . . . . .	282
11.68 CMultiBatteryChargerDeviceNet Class Reference . . . . .	283
11.68.1 Detailed Description . . . . .	284
11.68.2 Constructor & Destructor Documentation . . . . .	284
11.68.3 Member Function Documentation . . . . .	284
11.69 CMultiwellCallbackFunctionNet Class Reference . . . . .	290
11.69.1 Detailed Description . . . . .	291
11.69.2 Constructor & Destructor Documentation . . . . .	291
11.69.3 Member Function Documentation . . . . .	291
11.69.4 Event Documentation . . . . .	293
11.70 CMultiwellDeviceNet Class Reference . . . . .	293
11.70.1 Detailed Description . . . . .	294
11.70.2 Constructor & Destructor Documentation . . . . .	295
11.70.3 Member Function Documentation . . . . .	295
11.71 CMultiwellOptoStimFunctionNet Class Reference . . . . .	299
11.71.1 Detailed Description . . . . .	299
11.71.2 Constructor & Destructor Documentation . . . . .	299
11.71.3 Member Function Documentation . . . . .	300
11.72 CNF_GenDeviceNet Class Reference . . . . .	304
11.72.1 Constructor & Destructor Documentation . . . . .	304
11.72.2 Member Function Documentation . . . . .	304
11.73 COctoPotDeviceNet Class Reference . . . . .	304
11.73.1 Constructor & Destructor Documentation . . . . .	305
11.73.2 Member Function Documentation . . . . .	305
11.74 COkuvisionStimulatorDeviceNet Class Reference . . . . .	308
11.74.1 Constructor & Destructor Documentation . . . . .	309
11.74.2 Member Function Documentation . . . . .	309
11.75 CPatchServerDeviceNet Class Reference . . . . .	312
11.75.1 Detailed Description . . . . .	312



---

11.75.2 Constructor & Destructor Documentation	312
11.75.3 Property Documentation	312
11.76 CPathIdentDeviceNet Class Reference	313
11.76.1 Constructor & Destructor Documentation	313
11.76.2 Member Function Documentation	313
11.77 CPedoterDeviceNet Class Reference	314
11.77.1 Detailed Description	314
11.77.2 Constructor & Destructor Documentation	314
11.77.3 Member Function Documentation	314
11.78 CPeristalticPumpDeviceNet Class Reference	315
11.78.1 Detailed Description	315
11.78.2 Constructor & Destructor Documentation	316
11.78.3 Property Documentation	316
11.79 CPgaDeviceNet Class Reference	316
11.79.1 Constructor & Destructor Documentation	317
11.79.2 Member Function Documentation	317
11.80 CPositionIIDeviceNet Class Reference	318
11.80.1 Detailed Description	319
11.80.2 Constructor & Destructor Documentation	319
11.80.3 Member Function Documentation	319
11.80.4 Property Documentation	321
11.81 CPositionImpDeviceNet Class Reference	321
11.81.1 Detailed Description	322
11.81.2 Constructor & Destructor Documentation	322
11.81.3 Member Function Documentation	323
11.82 CPPCDeviceNet Class Reference	324
11.82.1 Constructor & Destructor Documentation	325
11.82.2 Property Documentation	325
11.83 CPPCFunctionNet Class Reference	326
11.83.1 Detailed Description	327
11.83.2 Constructor & Destructor Documentation	327
11.83.3 Member Function Documentation	327
11.84 CPPS_DeviceNet Class Reference	332
11.84.1 Constructor & Destructor Documentation	333
11.84.2 Property Documentation	333
11.85 CPPS_FunctionNet Class Reference	333
11.85.1 Constructor & Destructor Documentation	334
11.85.2 Member Function Documentation	335
11.86 CPPSDeviceNet Class Reference	338
11.86.1 Detailed Description	338
11.86.2 Constructor & Destructor Documentation	338
11.87 CProgramPressureCurveNet Class Reference	338

---

11.87.1 Detailed Description . . . . .	339
11.87.2 Constructor & Destructor Documentation . . . . .	339
11.87.3 Member Function Documentation . . . . .	339
11.88 CPulseGeneratorFunctionNet Class Reference . . . . .	340
11.88.1 Detailed Description . . . . .	341
11.88.2 Constructor & Destructor Documentation . . . . .	341
11.88.3 Member Function Documentation . . . . .	341
11.89 CRadioControlledDevicesNet Class Reference . . . . .	343
11.89.1 Constructor & Destructor Documentation . . . . .	344
11.89.2 Member Function Documentation . . . . .	344
11.90 CCMOSMeaDeviceNet::CRegionOfInterestRect Class Reference . . . . .	345
11.90.1 Constructor & Destructor Documentation . . . . .	345
11.90.2 Member Function Documentation . . . . .	345
11.90.3 Member Data Documentation . . . . .	345
11.91 CRetinaLedDeviceNet Class Reference . . . . .	346
11.91.1 Constructor & Destructor Documentation . . . . .	346
11.91.2 Member Function Documentation . . . . .	346
11.92 CRFFFunctionNet Class Reference . . . . .	348
11.92.1 Detailed Description . . . . .	348
11.92.2 Constructor & Destructor Documentation . . . . .	349
11.92.3 Member Function Documentation . . . . .	349
11.93 CRobo_FYIProgram_FunctionNet Class Reference . . . . .	352
11.93.1 Constructor & Destructor Documentation . . . . .	353
11.93.2 Member Function Documentation . . . . .	353
11.94 CRobo_FYITemp_FunctionNet Class Reference . . . . .	354
11.94.1 Constructor & Destructor Documentation . . . . .	354
11.94.2 Member Function Documentation . . . . .	355
11.95 CRoboDacqNet Class Reference . . . . .	356
11.95.1 Constructor & Destructor Documentation . . . . .	358
11.95.2 Member Function Documentation . . . . .	358
11.96 CRoboDeviceNet Class Reference . . . . .	369
11.96.1 Detailed Description . . . . .	372
11.96.2 Constructor & Destructor Documentation . . . . .	372
11.96.3 Member Function Documentation . . . . .	373
11.96.4 Member Data Documentation . . . . .	379
11.96.5 Property Documentation . . . . .	383
11.96.6 Event Documentation . . . . .	383
11.97 CRoboFluidDeviceNet Class Reference . . . . .	383
11.97.1 Constructor & Destructor Documentation . . . . .	384
11.97.2 Member Function Documentation . . . . .	384
11.97.3 Member Data Documentation . . . . .	386
11.97.4 Property Documentation . . . . .	386

11.98 CRoboInjectDeviceNet Class Reference . . . . .	387
11.98.1 Detailed Description . . . . .	387
11.98.2 Constructor & Destructor Documentation . . . . .	387
11.99 CRoboocyte2DeviceNet Class Reference . . . . .	387
11.99.1 Detailed Description . . . . .	388
11.99.2 Constructor & Destructor Documentation . . . . .	388
11.99.3 Member Function Documentation . . . . .	388
11.100 CRoboStatorDeviceNet Class Reference . . . . .	389
11.100.1 Constructor & Destructor Documentation . . . . .	390
11.100.2 Member Function Documentation . . . . .	390
11.100.3 Property Documentation . . . . .	395
11.101 CSafeISDeviceNet Class Reference . . . . .	395
11.101.1 Detailed Description . . . . .	396
11.101.2 Constructor & Destructor Documentation . . . . .	396
11.101.3 Member Function Documentation . . . . .	396
11.101.4 Property Documentation . . . . .	397
11.102 CSCUDacqGroupChannelSelectionNet Class Reference . . . . .	398
11.102.1 Constructor & Destructor Documentation . . . . .	398
11.103 CSCUFunctionNet Class Reference . . . . .	399
11.103.1 Detailed Description . . . . .	401
11.103.2 Constructor & Destructor Documentation . . . . .	401
11.103.3 Member Function Documentation . . . . .	401
11.103.4 Event Documentation . . . . .	414
11.104 CSerialPortNet Class Reference . . . . .	414
11.104.1 Constructor & Destructor Documentation . . . . .	414
11.104.2 Member Function Documentation . . . . .	415
11.105 CStg200xBasicNet Class Reference . . . . .	416
11.105.1 Detailed Description . . . . .	420
11.105.2 Constructor & Destructor Documentation . . . . .	420
11.105.3 Member Function Documentation . . . . .	420
11.106 CStg200xDownloadBasicNet Class Reference . . . . .	446
11.106.1 Detailed Description . . . . .	448
11.106.2 Member Function Documentation . . . . .	448
11.106.3 Property Documentation . . . . .	455
11.107 CStg200xDownloadNet Class Reference . . . . .	455
11.107.1 Detailed Description . . . . .	456
11.107.2 Constructor & Destructor Documentation . . . . .	456
11.107.3 Member Function Documentation . . . . .	457
11.107.4 Event Documentation . . . . .	461
11.108 CStimulusFunctionNet Class Reference . . . . .	462
11.108.1 Constructor & Destructor Documentation . . . . .	463
11.108.2 Member Function Documentation . . . . .	464

11.108.3 Event Documentation . . . . .	473
11.109 CSw2to64DeviceNet Class Reference . . . . .	473
11.109.1 Detailed Description . . . . .	474
11.109.2 Constructor & Destructor Documentation . . . . .	474
11.109.3 Member Function Documentation . . . . .	474
11.110 CTcxDeviceNet Class Reference . . . . .	476
11.110.1 Detailed Description . . . . .	478
11.110.2 Constructor & Destructor Documentation . . . . .	478
11.110.3 Member Function Documentation . . . . .	478
11.111 CTEERFunctionNet Class Reference . . . . .	489
11.111.1 Detailed Description . . . . .	491
11.111.2 Constructor & Destructor Documentation . . . . .	491
11.111.3 Member Function Documentation . . . . .	491
11.112 CTEERMachineDeviceNet Class Reference . . . . .	499
11.112.1 Constructor & Destructor Documentation . . . . .	499
11.112.2 Property Documentation . . . . .	499
11.113 CUsbExceptionNet Class Reference . . . . .	500
11.113.1 Detailed Description . . . . .	500
11.113.2 Constructor & Destructor Documentation . . . . .	500
11.113.3 Property Documentation . . . . .	501
11.114 CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet Class Reference . . . . .	501
11.114.1 Constructor & Destructor Documentation . . . . .	501
11.114.2 Member Data Documentation . . . . .	501
11.115 CW2100_FunctionNet Class Reference . . . . .	501
11.115.1 Constructor & Destructor Documentation . . . . .	503
11.115.2 Member Function Documentation . . . . .	503
11.115.3 Property Documentation . . . . .	508
11.116 CW2100_StimulatorFunctionNet Class Reference . . . . .	509
11.116.1 Constructor & Destructor Documentation . . . . .	510
11.116.2 Member Function Documentation . . . . .	510
11.116.3 Member Data Documentation . . . . .	515
11.116.4 Event Documentation . . . . .	515
11.117 CW2100DacqGroupChannelSelectionNet Class Reference . . . . .	515
11.117.1 Constructor & Destructor Documentation . . . . .	516
11.118 CWarnerUssingDeviceNet Class Reference . . . . .	516
11.118.1 Detailed Description . . . . .	516
11.118.2 Constructor & Destructor Documentation . . . . .	517
11.118.3 Property Documentation . . . . .	517
11.119 CWarnerUssingFunctionNet Class Reference . . . . .	517
11.119.1 Detailed Description . . . . .	519
11.119.2 Constructor & Destructor Documentation . . . . .	519
11.119.3 Member Function Documentation . . . . .	520

11.120 CWarnerValveControllerDeviceNet Class Reference . . . . .	534
11.120.1 Detailed Description . . . . .	538
11.120.2 Constructor & Destructor Documentation . . . . .	538
11.120.3 Member Function Documentation . . . . .	538
11.120.4 Event Documentation . . . . .	554
11.121 CWarnerValveControllerDeviceTesterFunctionNet Class Reference . . . . .	557
11.121.1 Detailed Description . . . . .	558
11.121.2 Constructor & Destructor Documentation . . . . .	558
11.121.3 Member Function Documentation . . . . .	558
11.122 CWClassicFunctionNet Class Reference . . . . .	560
11.122.1 Constructor & Destructor Documentation . . . . .	561
11.122.2 Member Function Documentation . . . . .	561
11.123 CWirelessBaseFunctionNet Class Reference . . . . .	565
11.123.1 Constructor & Destructor Documentation . . . . .	565
11.123.2 Member Function Documentation . . . . .	565
11.124 DeviceIdNet Struct Reference . . . . .	565
11.124.1 Detailed Description . . . . .	566
11.124.2 Constructor & Destructor Documentation . . . . .	566
11.124.3 Member Function Documentation . . . . .	566
11.124.4 Member Data Documentation . . . . .	567
11.125 DigitalSource< digitalsourceenum > Class Template Reference . . . . .	567
11.125.1 Constructor & Destructor Documentation . . . . .	567
11.125.2 Member Function Documentation . . . . .	567
11.125.3 Property Documentation . . . . .	568
11.126 DriverVersionNet Class Reference . . . . .	568
11.126.1 Detailed Description . . . . .	569
11.126.2 Constructor & Destructor Documentation . . . . .	569
11.126.3 Member Function Documentation . . . . .	569
11.127 FirmwareDestinationNames Class Reference . . . . .	573
11.127.1 Member Data Documentation . . . . .	574
11.128 HeadStageIDType Class Reference . . . . .	577
11.128.1 Member Enumeration Documentation . . . . .	577
11.128.2 Constructor & Destructor Documentation . . . . .	578
11.128.3 Member Function Documentation . . . . .	578
11.128.4 Property Documentation . . . . .	578
11.129 HeadstageIDTypeObject Class Reference . . . . .	580
11.129.1 Constructor & Destructor Documentation . . . . .	580
11.129.2 Member Function Documentation . . . . .	580
11.129.3 Member Data Documentation . . . . .	580
11.129.4 Property Documentation . . . . .	581
11.130 HeadStageIDTypeState Class Reference . . . . .	581
11.130.1 Property Documentation . . . . .	581

11.131 mkfilterNet Class Reference . . . . .	582
11.131.1 Member Function Documentation . . . . .	582
11.132 CRoboDeviceNet::RoboMainLowLevelCommands Class Reference . . . . .	585
11.132.1 Member Function Documentation . . . . .	586
11.133 CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands Class Reference . . . . .	592
11.133.1 Member Function Documentation . . . . .	592
11.134 CMeaAudioFunctionNet::s_setaudionet Struct Reference . . . . .	592
11.134.1 Member Data Documentation . . . . .	592
11.135 CStimulusFunctionNet::SidebandData Class Reference . . . . .	592
11.135.1 Constructor & Destructor Documentation . . . . .	593
11.135.2 Property Documentation . . . . .	593
11.136 StgStatusNet Class Reference . . . . .	593
11.136.1 Member Function Documentation . . . . .	594
11.136.2 Member Data Documentation . . . . .	594
11.137 CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData Class Reference . . . . .	594
11.137.1 Constructor & Destructor Documentation . . . . .	594
11.137.2 Property Documentation . . . . .	595
11.138 usbSetupPacket_t Class Reference . . . . .	595
11.138.1 Member Data Documentation . . . . .	596
11.139 W2100_StimulusParametersNet Struct Reference . . . . .	596
11.139.1 Member Data Documentation . . . . .	596
<b>Index</b>	<b>598</b>

## 1 McsUsbNet.dll for MCS USB devices

### 1.1 Introduction

This DLL provides the .NET interface to MCS devices

The most important options are accessing our stimulator and data acquisition devices:

- [STG200x & STG400x STimulus Generator](#)
- [Data ACQuisition \(DACQ\) Devices](#)

See here for a list of our other devices: [Device Classes](#).

And here for a list of function classes addressing groups of features that might be shared between different devices: [Function Classes](#).

## 1.2 System requirements

The DLL can be used with any .NET compatible language.

The DLL needs the **.NET Framework 4.7.2**.

It requires the **Microsoft Visual C++ Redistributable for Visual Studio 2019** to be installed.

It also requires the **USB driver** to be installed.

The simplest way to achieve this is to install the latest **Multi Channel Experimenter** setup (will install 64bit redistributable).

All examples assume that the [Mcs.Usb](#) namespace is loaded:

```
using namespace Mcs.Usb;
```

Include the file `McsUsbNet.dll` into the references of your project.

## 1.3 Connecting to an MCS device

A connection to a DAQ device is established by [Mcs.Usb.CMcsUsbNet.Connect](#). When this function is called without argument, the first DAQ device found on the USB bus is used:

```
CMcsUsbNet device = new CMcsUsbNet();  
device.Connect();
```

When more than one DAQ device of the specific type is connected, you can use the [Mcs.Usb.CMcsUsbListNet](#) class to get a list of available devices:

```
CMcsUsbListNet usblast = new CMcsUsbListNet(DeviceEnumNet.MCS_DEVICE_USB);  
var entry = usblast.GetUsbListEntry((uint)0);  
CMcsUsbNet device = new CMcsUsbNet();  
device.Connect(entry);
```

After you are finished with the device, you can disconnect the device object from the device by:

```
device.Disconnect();
```

## 2 Device Classes

- For FluidControl device see [MCS FluidControl](#)
- For SW2TO64 device see [MCS-USB-Sw2to64](#)
- For TCx device see [Mcs.Usb.CTcxDeviceNet](#)

### 2.1 The MCS FluidControl Device

#### 2.1.1 Introduction

The FluidControl Device can control up to 24 valves. The nominal voltage is 24V.

8 TTL level digital output ports are available and 8 TTL inputs can be read in.

The device has 8 ADC inputs with a range from 0V to 3.3V.

### 2.1.2 Access to the FluidControl device

For connecting to a FluidControl device see [Connecting to an MCS device](#).\*

```
CFluidControlDevice* m_dacq;
m_fluidcontrol = new CFluidControlDevice;
status = m_fluidcontrol->Connect();
```

The valves are controlled with the CFluidControlDevice::SetValve call. The argument given is a bit pattern of all valves which should be open.

The digital outputs can be controlled with the CFluidControlDevice::SetDigout call. Again, a bit pattern of all digital output pins which should be set to a logic high level is given as an argument.

The current state of the valves and the digital outputs can be read back with the CFluidControlDevice::GetValve and CFluidControlDevice::GetDigout

The command to read an ADC-Channel is CFluidControlDevice::GetAdc. Here the channelnummer which should be read in is given as an argument and the return value is the current Adc level.

The state of the digital inputs is read with the CFluidControlDevice::GetDigin call. Here the return value is the bit pattern of the digital inputs.

The connection to the device is closed with the CFluidControlDevice::Disconnect call.

## 2.2 MCS-USB-Sw2to64 device

The class [Mcs.Usb.CSw2to64DeviceNet](#) controls the setting of the switches in the MCS-USB-Sw2to64 device.

First construct an object of the class:

```
CSw2to64DeviceNet device = new CSw2to64DeviceNet();
```

For connecting to an MCS-USB-Sw2to64 device see [Connecting to an MCS device](#).

To get the number of channels the device handles:

```
int number = device.GetNumber();
```

Set all channel switches at once:

```
byte z = 1;
byte[] pattern = new byte[number];
for(int i = 0; i < number; i++)
{
    pattern[i] = z; // pattern you want to switch this channel to
}
device.SetChannels(pattern);
```

Get all channel switches at once:

```
byte[] pattern = device.GetChannels();
```

Set one channel switch:

```
ushort index = 10;
byte pattern = 1;
device.SetChannel(index, pattern)
```

Get one channel switch:

```
ushort index = 10;
byte pattern = device.GetChannel(index);
```



## 3 Function Classes

- [Mcs.Usb.CCMOSMea\\_FunctionNet](#)
- [Mcs.Usb.CDacCalibrationFunctionNet](#)
- [Mcs.Usb.CDigOutStimulatorFunctionNet](#)
- [Mcs.Usb.CIntanMea\\_FunctionNet](#)
- [Mcs.Usb.CInterfaceboardFunctionNet](#)
- [Mcs.Usb.CMcsBus\\_MotorControlNet](#)
- [Mcs.Usb.CMcsBus\\_VoltageModeNet](#)
- [Mcs.Usb.CMcsBus\\_AxisParametersNet](#)
- [Mcs.Usb.CMcsBus\\_SensorNet](#)
- [Mcs.Usb.CMcsBus\\_TempSensorNet](#)
- [Mcs.Usb.CMcsBus\\_ExtensionNet](#)
- [Mcs.Usb.CMcsBus\\_FYIExtensionNet](#)
- [Mcs.Usb.CMcsUsbDeviceStatePushFunctionNet](#)
- [Mcs.Usb.CMEA2100x256FunctionNet](#)
- [Mcs.Usb.CMeaAudioFunctionNet](#)
- [Mcs.Usb.CMeaDigitalDataFunctionNet](#)
- [Mcs.Usb.CMeaFeedbackFunctionNet](#)
- [Mcs.Usb.CMeFunctionNet](#)
- [Mcs.Usb.CMultiwellCallbackFunctionNet](#)
- [Mcs.Usb.CMultiwellOptoStimFunctionNet](#)
- [Mcs.Usb.CPPCFunctionNet](#)
- [Mcs.Usb.CPPS\\_FunctionNet](#)
- [Mcs.Usb.CPPS\\_FunctionNet](#)
- [Mcs.Usb.CPulseGeneratorFunctionNet](#)
- [Mcs.Usb.CRFFunctionNet](#)
- [Mcs.Usb.CRobo\\_FYITemp\\_FunctionNet](#)
- [Mcs.Usb.CRobo\\_FYIProgram\\_FunctionNet](#)
- [Mcs.Usb.CRobo\\_FYITemp\\_FunctionNet](#)
- [Mcs.Usb.CRobo\\_FYIProgram\\_FunctionNet](#)
- [Mcs.Usb.CSCUFunctionNet](#)
- [Mcs.Usb.CStimulusFunctionNet](#)
- [Mcs.Usb.CTEERFunctionNet](#)
- [Mcs.Usb.CW2100\\_FunctionNet](#)
- [Mcs.Usb.CW2100\\_StimulatorFunctionNet](#)

- [Mcs.Usb.CWarnerUssingFunctionNet](#)
- [Mcs.Usb.CWarnerValveControllerDeviceTesterFunctionNet](#)
- [Mcs.Usb.CWClassicFunctionNet](#)
- [Mcs.Usb.CWirelessBaseFunctionNet](#)

## 4 Data ACQuisition (DACQ) Devices

There are different device types of (MEA) data acquisition (DACQ) devices. All of them are supported by this class.

This library does **not** support the writing of the MCD (MC\_Rack), MSRD (Multi Channel Experimenter) or HDF5 file format!

The class [Mcs.Usb.CMeaDeviceNet](#) is the base class for DACQ devices.

The base class [Mcs.Usb.CMeaDeviceNet](#) constructs actually the underlying classes for USB-MEA devices ([Mcs.Usb.CMeaUSBDeviceNet](#)).

```
CMeaDeviceNet device = new CMeaDeviceNet(McsBusTypeEnumNet.MCS_USB_BUS, OnChannelData, OnError);
```

For connecting to a DACQ device see [Connecting to an MCS device](#).

Get the number of available analog hardware channels and set the number of channels to the maximum.

```
int hwchannels;
device.HWInfo().GetNumberOfHWADCCChannels(out hwchannels);
device.SetNumberOfChannels(hwchannels);
int samplingrate = 1000;
device.SetSamplerate(samplingrate, 1, 0);
device.EnableDigitalIn(true, 0);
```

Get the layout to know how the data look like that you receive

```
int ana, digi, che, tim, block;
device.GetChannelLayout(out ana, out digi, out che, out tim, out block);
```

For the [Mcs.Usb.OnChannelData](#) callback function you have to provide a definition of the channels you want to receive.

```
bool[] selChannels = new bool[block];
for (int i = 0; i < block; i++)
{
    selChannels[i] = true; // With true channel i is selected
    // selChannels[i] = false; // With false the channel i is deselected
}
channelblocksize = samplingrate / 10;
// queue size and threshold should be selected carefully
device.SetSelectedChannels(selChannels, 10 * channelblocksize, channelblocksize);
```

The [Mcs.Usb.OnChannelData](#) callback function gets a callback for each channelblock that is defined. In this example a callback for each channel.

```
void OnChannelData(CMcsUsbDacqNet d, int cbHandle, int numSamples)
{
    int size_ret;
    ushort[] channeldata = device.ChannelBlock_ReadFramesUI16(CbHandle, numSamples, out size_ret);
}
void OnError(String msg, int info)
{
    MessageBox.Show("Mea Device Error: " + msg);
}
```

see MEA\_Recording in the Examples directory.

## 5 The MCS Robo Device

### 5.1 Introduction

Up to now two MCS devices exist that base on the Robo platform.

- The MCS Roboinject device is controlled by the [Mcs.Usb.CRoboInjectDeviceNet](#) class.
- The MCS Roboocyte2 device is controlled by the [Mcs.Usb.CRoboocyte2DeviceNet](#) class.

Both classes are derived from [Mcs.Usb.CRoboDeviceNet](#)

## 6 STG200x & STG400x STimulus Generator

### 6.1 Introduction

The STG200x & STG400x Series Stimulus Generators have two distinct modes of operation, the [Download mode](#) and the [Streaming mode](#).

### 6.2 Download mode

The Download mode is the "classic" mode of operation, as used by the MC Stimulus software. In this mode, one or multiple waveforms are defined in PC memory and downloaded to the STG. The waveforms are stored in STG device onboard memory and can be sent to the analog and sync outputs once or multiple times. The STG can operate independantly from the PC (without computer connection) after the download. Output is triggered either by the front panel start/stop button, the digital trigger inputs or under software control.

In the Download mode, there are up to eight independent triggers available (depending on the device). The user can assign each of the analog outputs and sync (digital) outputs to any of the triggers.

The analog output waveform is stored sample by sample in the STG memory. To reduce memory usage, this data can be compressed: whenever a given output value is to be held for more than one sample period, it has only to be given once. The user can define the number of sample periods for that a pattern should remain active. Compression is done for each channel independantly of the others, thus the algorithm to compress the data is very easy to implement.

A new feature of the Download mode is the segmentation of the STG memory. The onboard memory can be devided into up to 100 segments. Each segment can hold its own waveform pattern. Under software control, the user can switch between the defined segments within milliseconds. Another option is to use the four trigger inputs to select between four predefined segments. This option is accessible from the MC\_Stimulus Software as the "Multi-File mode", and can start each of up to four defined waveforms within microseconds. This feature allows a predefined flexible response (feedback) to recorded data.

[Mcs.Usb.CStg200xDownloadNet](#) is the class for using the STG in download mode.

### 6.2.1 Memory Layout and Trigger Setup

The class to be used for the Download mode is [Mcs.Usb.CStg200xDownloadNet](#), which is derived from [Mcs.Usb.CStg200xBasicNet](#). You can add a poll handler delegate ([Mcs.Usb.OnStg200xPollStatus](#)) to the constructor [Mcs.Usb.CStg200xDownloadNet](#).

For connecting to an STG see [Connecting to an MCS device](#).

To use the Download mode, the memory layout of the STG200x can be set up, if the default is not sufficient. The total amount of memory available in the STG is obtained by the [Mcs.Usb.CStg200xDownloadNet.GetTotalMemory](#) call. With [Mcs.Usb.CStg200xDownloadNet.SendSegmentDefine](#) the segment sizes are assigned.

```
uint32_t memory = device.GetTotalMemory(); // obtain total memory available
uint[] segmentmemory = new uint[2];      // each segments has half of total memory
segmentmemory[0] = memory / 2;
segmentmemory[1] = memory / 2;
device.SendSegmentDefine(segmentmemory); // setup the STG
```

Next, for each segment, one has to assign the amount of memory to be used for each channel and sync output. This is done by [Mcs.Usb.CStg200xDownloadBasicNet.SetCapacity](#). Its arguments contain a list of memory sizes, with one entry per channel and one entry per sync output. Again, the total memory assigned to the channels and sync outputs must not exceed the memory assigned to the segment.

```
uint32_t nchannels = device.GetNumberOfAnalogChannels();
uint32_t nsync = device.GetNumberOfSyncoutChannels();
uint[] channel_cap = new uint[nchannels];
uint[] syncout_cap = new uint[nsync];
for (int i = 0; i < 2; i++) // for each segment
{
    device.SendSegmentSelect((uint32_t)i); // switch to segment
    uint32_t segment_mem = device.GetMemory(); // get memory available in this segment
    for(int j = 0; j < nchannels; j++)
    {
        channel_cap[j] = segment_mem/(nchannels+nsync); // devide memory amount to all channels
    }
    for(int j = 0; j < nsync; j++)
    {
        syncout_cap[j] = segment_mem/(nchannels+nsync); // and all sync outs.
    }
    device.SetCapacity(channel_cap, syncout_cap); // define memory for current segment
}
```

Before the STG can start, the trigger has to be configured. This is done by the [Mcs.Usb.CStg200xDownloadNet.SetupTrigger](#) call. Its arguments are a list of channelmaps, syncoutmaps and repeats, one for each of the four available triggers. channelmap is a bitmap, each bit representing one of the available channels. To assign channel 1 and syncout 1 to trigger 1 and channel 3 to trigger 2 use:

```
uint32_t TriggerInputs = device.GetNumberOfTriggerInputs();
uint[] channelmap = new uint[TriggerInputs];
uint[] syncoutmap = new uint[TriggerInputs];
uint[] repeat = new uint[TriggerInputs];
for (int i = 0; i < TriggerInputs; i++)
{
    channelmap[i] = 0;
    syncoutmap[i] = 0;
    repeat[i] = 0;
}
// Trigger 0
channelmap[0] = 1; // Channel 1
syncoutmap[0] = 1; // Syncout 1
repeat[0] = 0; // forever
// Trigger 1
channelmap[1] = 4; // Channel 3
device.SetupTrigger(channelmap, syncoutmap, repeat);
```

For the STG400x series you have to set the output mode of the channels. [Mcs.Usb.CStg200xDownloadNet.SetVoltageMode](#) interprets the values as voltages. [Mcs.Usb.CStg200xDownloadNet.SetCurrentMode](#) as currents.

```
// Only meaningful for STG400x
device.SetVoltageMode();
```

For each segment, data can be sent to each of the defined channels and sync outputs using the [Mcs.Usb.CStg200xDownloadNet.SendChannelData](#) and [Mcs.Usb.CStg200xDownloadNet.SendSyncData](#) calls. channeldata and syncdata are a list of analog and digital samples as a list of two byte values (unsigned short). Multiple calls to [Mcs.Usb.CStg200xDownloadNet.SendChannelData](#) and [Mcs.Usb.CStg200xDownloadNet.SendSyncData](#) to the same channel append data to that channel.

If the Multi-File mode of the STG is enabled using the [Mcs.Usb.CStg200xDownloadNet.EnableMultiFileMode](#) call, the four trigger inputs are used to switch between four segments. A hardware trigger signal (TTL) on trigger input 1 selects the first segment and starts all pulses in this segment. Thus with the Multi-File mode, one can predefine four stimulus patterns and switch between them without a connection to the PC.

The STG200x series has an analog resolution of 13 bits, thus the analog data contains the information in bits 0 to 12 of each sample. Bits 13 to 15 have to be 0.

```
int DACResolution = device.GetDACResolution();
// Data for Channel 0
{
    device.ClearChannelData(0);
    double factor = 0.1;
    const int l = 1000;
    ushort[] pData = new ushort[l];
    UInt64_t[] tData = new UInt64_t[l];
    for (int i = 0; i < l; i++)
    {
        // calculate Sin-Wave
        double sin = factor * (Math.Pow(2, DACResolution - 1) - 1.0) *
            Math.Sin(2.0 * (double)i * Math.PI / (double)l);
        // calculate sign
        pData[i] = sin >= 0 ? (ushort)sin : (ushort)((int)Math.Abs(sin) +
            (int)Math.Pow(2, DACResolution - 1));
        tData[i] = (UInt64_t)20; // duration in µs
    }
    device.SendChannelData(0, pData, tData);
}
// Data for Channel 3
{
    device.ClearChannelData(2);
    double factor = 0.1;
    const int l = 700;
    // without compression
    ushort[] pData = new ushort[l];
    UInt64_t[] tData = new UInt64_t[l];
    for (int i = 0; i < l; i++)
    {
        // calculate Sin-Wave
        double sin = factor * (Math.Pow(2, DACResolution - 1) - 1.0) *
            Math.Sin(2.0 * (double)i * Math.PI / (double)l);
        // calculate sign
        pData[i] = sin >= 0 ? (ushort)sin : (ushort)((int)Math.Abs(sin) +
            (int)Math.Pow(2, DACResolution - 1));
        tData[i] = (UInt64_t)20; // duration in µs
    }
    device.SendChannelData(2, pData, tData);
}
// Data for Sync 0
{
    device.ClearSyncData(0);
    ushort[] pData = new ushort[1000];
    UInt64_t[] tData = new UInt64_t[1000];
    for (int i = 0; i < 1000; i++)
    {
        pData[i] = (ushort)(i&1);
        tData[i] = 20;
    }
    device.SendSyncData(0, pData, tData);
}
```

Start the trigger by pushing the front button or by software

```
// Start Trigger 1 and 2
device.SendStart(1 + 2); // Trigger 1 und 2
```

see the StgDownloadExampleNet in the example directory.

## 6.3 Streaming mode

The other mode of operation is the Streaming mode. Here the analog output is sent to the STG device in "real time". The PC has to be connected to the STG all the time. The data that is sent to the analog output is downloaded from the PC to the STG on the fly.

The Streaming mode is useful for applications where flexible feedback is needed as well for applications where very long waveforms which are not repeated (such as white noise) are used.

The Streaming mode works by use of two ring buffers which hold data. One is in PC memory and managed by the DLL, and one is in on-board STG memory. Data is transferred from PC memory to the STG via the USB bus in time slices of one millisecond.

The user can define both the size of the ring buffer in DLL memory and in the STG memory. Once the Streaming mode is started, the STG request data from the PC. The data rate from PC to STG is variable and controlled by the STG. The STG request data from the PC at a rate to keep its internal ringbuffer at about half full.

It is the responsibility of the user to keep the ring buffer in the memory of the PC filled, so the DLL can supply sufficient data to the STG. To do so, the Windows DLL allows to define a "callback" function which is called whenever new data is needed, or more precise, as soon as the ring buffer in the memory of the PC falls below the user defined threshold.

Small buffers have the advantage of a low latency between data generation in the callback function and its output as a analog signal from the STG. However for low latency to work, the user-written callback function has to be fast and to produce a steady flow of data.

In the Streaming mode, all triggers are available as well. Each of the eight analog and sync outputs can be assigned to one of the triggers.

The output rate is user defined with a maximum of 50 kHz

Mcs.Usb.CStg200xStreamingNet is the class for using the STG in streaming mode.

### 6.3.1 Memory Layout and Trigger Setup

With the constructor for Mcs.Usb.CStg200xStreamingNet.CStg200xStreamingNet, the name of the callback function for the data handler is provided. The data handler function is called automatically, whenever the STG needs new data. This data is first written to a ring buffer in the memory of the PC. The size for this ring buffer is defined as first argument in the constructor. The user provided delegate gets the trigger number which needs new data as argument

```
CStg200xStreamingNet device = new CStg200xStreamingNet(10000, dataHandler, errorHandler);
```

The callback function, which is defined in the constructor, is called whenever the STG needs new data for a trigger, or more precise, whenever the ring buffer in PC memory falls below the defined threshold.

The user can query the amount of space available for queuing by use of the Mcs.Usb.CStg200xStreamingNet.GetDataQueueSpace call. Its return value is the number of samples that can be send to the STG.

User code is required to fill an array analog and sync out data, sample by sample for up to the maximum number of samples as obtained by Mcs.Usb.CStg200xStreamingNet.GetDataQueueSpace or Mcs.Usb.CStg200xStreamingNet.GetSyncoutQueueSpace.

The values for the analog outputs are 16 bits signed integers. The lower bits are truncated according to the resolution of the STG. This behaviour is different to the behaviour in [download mode](#).

Note: Compression as described in the [download mode](#) can NOT be used for the streaming mode.

The new data is sent to the STG by using the Mcs.Usb.CStg200xStreamingNet.EnqueueData call.

```
void dataHandler(uint32_t trigger)
{
    double factor = 1;
    if (trigger == 0) // Callback for Trigger 1
    {
        // Handle Channel 1
        uint32_t channel = 0;
        for (; ; )
        {
            uint32_t space = device.GetDataQueueSpace(channel);
            if (space < 1000)
                break;
            short[] data = new short[1000];
```

```

        for (int i = 0; i < 1000; i++)
        {
            // Calc Sin-Wave (16 bits) lower bits will be removed according resolution
            double sin = factor * (Math.Pow(2, 16 - 1) - 1.0) *
                Math.Sin(2.0 * (double)i * Math.PI / (double)1000);
            data[i] = (short)sin;
        }
        uint32_t enqueued = device.EnqueueData(channel, data);
    }
}

// Handle Channel 3
uint32_t channel = 2;
for ( ; ; )
{
    uint32_t space = device.GetDataQueueSpace(channel);
    if (space < 700)
        break;
    short[] data = new short[700];
    for (int i = 0; i < 700; i++)
    {
        // Calc Sin-Wave (16 bits) lower bits will be removed according resolution
        double sin = factor * (Math.Pow(2, 16 - 1) - 1.0) *
            Math.Sin(2.0 * (double)i * Math.PI / (double)700);
        data[i] = (short)sin;
    }
    uint32_t enqueued = device.EnqueueData(channel, data);
}

// Handle Syncout 1
uint32_t channel = 0;
for ( ; ; )
{
    uint32_t space = device.GetSyncoutQueueSpace(channel);
    if (space < 1000)
        break;
    ushort[] data = new ushort[1000];
    for (int i = 0; i < 1000; i++)
        data[i] = (ushort)(i & 1);
    uint32_t enqueued = device.EnqueueSyncout(channel, data);
}
}

}

void errorHandler()
{
}

```

For connecting to an STG device see [Connecting to an MCS device](#).

With enabling or disabling the continuous mode it can be selected how the STG handles an "out of data" situation.

When `Mcs.Usb.CStg200xStreamingNet.EnableContinuousMode` is used, the STG does not stop when it runs out of data, but it keeps running and sends a zero voltage to its outputs.

When `Mcs.Usb.CStg200xStreamingNet.DisableContinuousMode` is used, the STG stops when it runs out of data. It has to be retriggered to resume the output.

```
device.EnableContinuousMode();
```

`Mcs.Usb.CStg200xStreamingNet.SetOutputRate` is used to set the sampling rate.

```
device.SetOutputRate(50000);
```

To use the Streaming mode, the memory layout of the STG has to be set up. To total amount of memory available in the STG is obtained by the `Mcs.Usb.CStg200xStreamingNet.GetTotalMemory` call.

This memory can be assigned to four ring buffers (one per trigger) which buffer the data received from the PC via USB cable. This is done with the `CStg200xStreaming::SetCapacity` call. The total amount of memory must not exceed the total memory size as obtained by `Mcs.Usb.CStg200xStreamingNet.GetTotalMemory`.

This internal ring buffer is crucial for proper operation of the Streaming mode. The size of the ring buffer determines the latency of the Streaming mode. The firmware of the STG requests data from the PC in order to keep the ring buffer about half full. Thus the average latency is:

$$\text{latency} = (\text{ringbuffersize in bytes}/4) / \text{output rate}$$

If the ring buffer size is too big, the latency of the STG might be too long. If the ring buffer size is too low, an overflow or underflow of data in the STG ringbuffer might occur, resulting in data jumps of the output signals or the "out of data" situation described earlier.

The following example divides the total memory equally among the four triggers:

```
uint32_t dwMemory = device.GetTotalMemory(); // obtain total memory available
uint32_t ntrigger = device.GetNumberOfTriggerInputs(); // obtain number of triggers in this STG
uint[] stg_triggercapacity = new uint[ntrigger];
for (int i = 0; i < ntrigger; i++)
    stg_triggercapacity[i] = dwMemory / ntrigger;
device.SetCapacity(stg_triggercapacity); // setup the STG
```

or fixed memory sizes:

```
uint32_t ntrigger = device.GetNumberOfTriggerInputs(); // obtain number of triggers in this STG
uint[] stg_triggercapacity = new uint[ntrigger];
for (int i = 0; i < ntrigger; i++)
    stg_triggercapacity[i] = 50000;
device.SetCapacity(stg_triggercapacity);
```

Before the STG can start, the trigger has to be configured. This is done by the `Mcs.Usb.CStg200xStreamingNet.SetupTrigger` call. Its arguments are a list of channelmaps, syncoutmaps, digoutmap, autostart and callback\_threshold, with one entry for each of the available triggers. channelmap is a bitmap, each bit representing one of the available channels. To assign channel 1 and 3 and syncout 1 to trigger 1 use:

```
uint32_t ntrigger = device.GetNumberOfTriggerInputs(); // obtain number of triggers in this STG
uint[] channelmap = new uint[ntrigger];
uint[] syncoutmap = new uint[ntrigger];
uint[] digoutmap = new uint[ntrigger];
uint[] autostart = new uint[ntrigger];
uint[] callback_threshold = new uint[ntrigger];
for (int i = 0; i < ntrigger; i++)
{
    channelmap[i] = 0;
    syncoutmap[i] = 0;
    digoutmap[i] = 0;
    autostart[i] = 0;
    callback_threshold[i] = 0;
}
channelmap[0] = 0x1 + 0x4; // Channel 1 und Channel 3 to Trigger 1
syncoutmap[0] = 0x1; // Syncout 1 to Trigger 1
autostart[0] = 1;
callback_threshold[0] = 50; // 50% of buffer size
device.SetupTrigger(channelmap, syncoutmap, digoutmap, autostart, callback_threshold);
device.StartLoop();
System.Threading.Thread.Sleep(1000); // Give StartLoop some time
```

Start Trigger by pushing the front button or by Software

```
device.SendStart(1);
```

see the `StgStreamingExampleNet` in the example directory.

## 7 Namespace Index

### 7.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">Mcs</a>	<a href="#">22</a>
<a href="#">Mcs::Usb</a>	<a href="#">22</a>

## 8 Hierarchical Index

### 8.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:



CW2100_FunctionNet::AudioChannelsNet	29
BatteryState	29
CCreateFilterNet	47
BesselFilterHighPassNet	30
BesselFilterLowPassNet	30
ButterworthFilterHighPassNet	31
ButterworthFilterLowPassNet	32
CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumTemplateNet >	57
CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumNet >	57
CDeviceGroupChannelInfoNet	55
CDeviceGroupChannelInfoTemplateNet< int >	57
CDeviceGroupChannelInfoGenericNet	55
CDeviceGroupChannelInfoTemplateNet< SCUDacqGroupChannelEnumNet >	57
CDeviceGroupChannelInfoSCUNet	56
CDeviceGroupChannelInfoTemplateNet< W2100DacqGroupChannelEnumNet >	57
CDeviceGroupChannelInfoW2100Net	57
CFilterCoefficientsNet	65
CFilterPropertyNet	71
CMcsUsbDacqNet::CHWInfo	103
CMcsUsbFunctionNet	217
CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumNet, DacqGroupChannelEnum, CDeviceGroupChannelInfoNet >	52
CDacqGroupChannelSelectionNet	52
CDacqGroupChannelSelectionTemplateNet< int, int, CDeviceGroupChannelInfoGenericNet >	52
CDacqGroupChannelGenericSelectionNet	51
CDacqGroupChannelSelectionTemplateNet< SCUDacqGroupChannelEnumNet, SCUDacqGroupChannelEnum, CDeviceGroupChannelInfoSCUNet >	52
CSCUDacqGroupChannelSelectionNet	398
CDacqGroupChannelSelectionTemplateNet< W2100DacqGroupChannelEnumNet, W2100DacqGroupChannelEnum, CDeviceGroupChannelInfoW2100Net >	52
CW2100DacqGroupChannelSelectionNet	515
CCMOSMea_FunctionNet	33
CDacCalibrationFunctionNet	49

CDacqGroupChannelSelectionTemplateNet < DacqGroupChannelEnumTemplateNet, Dacq↔ GroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet >	52
CDigOutStimulatorFunctionNet	58
CFilterConfigurationNet	68
CFilterConfigurationRegisterNet	70
CIntanMea_FunctionNet	105
CInterfaceboardFunctionNet	107
CMcsBus_AxisParametersNet	116
CMcsBus_ExtensionNet	118
CMcsBus_FYIExtensionNet	119
CMcsBus_MotorControlNet	121
CMcsBus_SensorNet	138
CMcsBus_TempSensorNet	148
CMcsBus_VoltageModeNet	150
CMcsBusNet	155
CMcsUsbDeviceStatePushFunctionNet	207
CMultiwellCallbackFunctionNet	290
CSCUFunctionNet	399
CMEA2100x256FunctionNet	247
CMeaAudioFunctionNet	249
CMeaDigitalDataFunctionNet	270
CMeaFeedbackFunctionNet	272
CMeFunctionNet	281
CMultiwellOptoStimFunctionNet	299
CPPCFunctionNet	326
CPPS_FunctionNet	333
CProgramPressureCurveNet	338
CPulseGeneratorFunctionNet	340
CRFFunctionNet	348
CRobo_FYIProgram_FunctionNet	352
CRobo_FYITemp_FunctionNet	354
CStimulusFunctionNet	462

CTEERFunctionNet	489
CW2100_StimulatorFunctionNet	509
CWarnerUssingFunctionNet	517
CWarnerValveControllerDeviceTesterFunctionNet	557
CWirelessBaseFunctionNet	565
CW2100_FunctionNet	501
CWClassicFunctionNet	560
CMcsUsbFunctionPointerContainer	219
CMcsUsbListEntryNet	219
CMcsUsbListNet	223
CMcsUsbNet	226
CExternDTesterDeviceNet	63
CFluidControlDeviceNet	73
CGenericDevelopDeviceNet	81
CGilsonDeviceNet	98
CMcsUsbDacqNet	159
CMeaDeviceNet	262
CMeaUSBDeviceNet	280
CCMOSMeaDeviceNet	44
CHLADacqNet	101
CLIH3DeviceNet	109
CMultiwellDeviceNet	293
CWarnerUssingDeviceNet	516
COctoPotDeviceNet	304
CRoboDacqNet	356
CMcsUsbDeviceStatePushNet	208
CWarnerValveControllerDeviceNet	534
CMcsUsbFactoryNet	209
CMeaCleanDeviceNet	252
CMeaCoatDeviceNet	256
CMeaImpedanceDeviceNet	276
CMeaSwitchDeviceNet	278

CChannelTestDeviceNet	32
CMultiBatteryChargerDeviceNet	283
CNF_GenDeviceNet	304
COkuvisionStimulatorDeviceNet	308
CPathIdentDeviceNet	313
CPedoterDeviceNet	314
CPeristalticPumpDeviceNet	315
CPgaDeviceNet	316
CPositionIIDeviceNet	318
CPositionImpDeviceNet	321
CPPCDeviceNet	324
CPPS_DeviceNet	332
CRadioControlledDevicesNet	343
CRetinaLedDeviceNet	346
CRoboDeviceNet	369
CEncapsulatorDeviceNet	62
CFYIDeviceNet	80
CHiClampDeviceNet	100
CHLADeviceNet	102
CMeasureTableDeviceNet	277
CPatchServerDeviceNet	312
CPPSDeviceNet	338
CRoboStatorDeviceNet	389
CRoboInjectDeviceNet	387
CRoboocyte2DeviceNet	387
CTEERMachineDeviceNet	499
CRoboFluidDeviceNet	383
CSafeISDeviceNet	395
CSerialPortNet	414
CStg200xBasicNet	416
CStg200xDownloadBasicNet	446
CStg200xDownloadNet	455

CSw2to64DeviceNet	473
CTcxDeviceNet	476
CMcsUsbPointerContainer	247
CCMOSMeaDeviceNet::CRegionOfInterestRect	345
CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet	501
DeviceIdNet	565
DigitalSource< digitalsourceenum >	567
DriverVersionNet Exception	568
CUsbExceptionNet	500
FirmwareDestinationNames	573
HeadstageIDTypeObject	580
HeadStageIDTypeState IComparable	581
HeadStageIDType	577
mkfilterNet	582
CRoboDeviceNet::RoboMainLowLevelCommands	585
CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands	592
CMeaAudioFunctionNet::s_setaudionet	592
CStimulusFunctionNet::SidebandData	592
StgStatusNet stgstreaming	593
CStg200xBasicNet	416
CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData	594
usbSetupPacket_t	595
W2100_StimulusParametersNet	596

## 9 Class Index

### 9.1 Class List

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

CW2100_FunctionNet::AudioChannelsNet	29
BatteryState	29

<a href="#">BesselFilterHighPassNet</a>	30
<a href="#">BesselFilterLowPassNet</a>	30
<a href="#">ButterworthFilterHighPassNet</a>	31
<a href="#">ButterworthFilterLowPassNet</a>	32
<a href="#">CChannelTestDeviceNet</a>	32
<a href="#">CCMOSMea_FunctionNet</a>	33
<a href="#">CCMOSMeaDeviceNet</a>	44
<a href="#">CCreateFilterNet</a>	47
<a href="#">CDacCalibrationFunctionNet</a> 49	
<a href="#">CDacqGroupChannelGenericSelectionNet</a>	51
<a href="#">CDacqGroupChannelSelectionNet</a>	52
<a href="#">CDacqGroupChannelSelectionTemplateNet&lt; DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplateNet &gt;</a> 52	
<a href="#">CDeviceGroupChannelInfoGenericNet</a>	55
<a href="#">CDeviceGroupChannelInfoNet</a>	55
<a href="#">CDeviceGroupChannelInfoSCUNet</a>	56
<a href="#">CDeviceGroupChannelInfoTemplateNet&lt; DacqGroupChannelEnumTemplateNet &gt;</a>	57
<a href="#">CDeviceGroupChannelInfoW2100Net</a>	57
<a href="#">CDigOutStimulatorFunctionNet</a> <a href="#">CDigOutStimulatorFunctionNet</a> is the class of the DigOut stimulator function class	58
<a href="#">CEncapsulatorDeviceNet</a> <a href="#">CEncapsulatorDeviceNet</a> is the to control the MCS HiClamp device	62
<a href="#">CExternDTesterDeviceNet</a> <a href="#">CExternDTesterDeviceNet</a> is the class to access the ExternD Tester (Handheld Device Tester D)	63
<a href="#">CFilterCoefficientsNet</a>	65
<a href="#">CFilterConfigurationNet</a>	68
<a href="#">CFilterConfigurationRegisterNet</a>	70
<a href="#">CFilterPropertyNet</a>	71
<a href="#">CFluidControlDeviceNet</a> <a href="#">CFluidControlDeviceNet</a> is the class to control MCS FluidControl (FCB and FCX) device	73
<a href="#">CFYIDeviceNet</a> <a href="#">CFYIDeviceNet</a> is the class to control the MCS FYI device	80
<a href="#">CGenericDevelopDeviceNet</a> <a href="#">CGenericDevelopDeviceNet</a> is the class to use during development of a new device	81

<a href="#">CGilsonDeviceNet</a>	
<a href="#">CGilsonDeviceNet</a> is the class to control a Gilson device	98
<a href="#">CHiClampDeviceNet</a>	
<a href="#">CHiClampDeviceNet</a> is the to control the MCS HiClamp device	100
<a href="#">CHLADacqNet</a>	101
<a href="#">CHLADeviceNet</a>	
<a href="#">CHLADeviceNet</a> is the to control the MCS HLA device	102
<a href="#">CMcsUsbDacqNet::CHWInfo</a>	
Class to provide hardware information about the device	103
<a href="#">CIntanMea_FunctionNet</a>	105
<a href="#">CInterfaceboardFunctionNet</a>	
<a href="#">CInterfaceboardFunctionNet</a> is the class to control the Interfaceboard	107
<a href="#">CLIH3DeviceNet</a>	
<a href="#">CLIH3DeviceNet</a> is the class to access the HEKA LIH3 device	109
<a href="#">CMcsBus_AxisParametersNet</a>	116
<a href="#">CMcsBus_ExtensionNet</a>	118
<a href="#">CMcsBus_FYIExtensionNet</a>	119
<a href="#">CMcsBus_MotorControlNet</a>	121
<a href="#">CMcsBus_SensorNet</a>	138
<a href="#">CMcsBus_TempSensorNet</a>	148
<a href="#">CMcsBus_VoltageModeNet</a>	150
<a href="#">CMcsBusNet</a>	155
<a href="#">CMcsUsbDacqNet</a>	
Base class for data acquisition devices	159
<a href="#">CMcsUsbDeviceStatePushFunctionNet</a>	207
<a href="#">CMcsUsbDeviceStatePushNet</a>	208
<a href="#">CMcsUsbFactoryNet</a>	209
<a href="#">CMcsUsbFunctionNet</a>	217
<a href="#">CMcsUsbFunctionPointerContainer</a>	219
<a href="#">CMcsUsbListEntryNet</a>	
<a href="#">McsUsbListEntryNet</a> identifies a connected device	219
<a href="#">CMcsUsbListNet</a>	
Class to handle a list of connected MCS USB devices	223
<a href="#">CMcsUsbNet</a>	
Base class to handle MCS USB devices. All device classes are derived from this class. Functionality that is provided by all MCS devices is handled by this class	226
<a href="#">CMcsUsbPointerContainer</a>	247

<a href="#">CMEA2100x256FunctionNet</a>	
<a href="#">CMEA2100x256FunctionNet</a> is the class to control the MEA2100-256 device needs #include " <a href="#">↩</a> Stg200xNet.h" to resolve documentation reference	247
<a href="#">CMeaAudioFunctionNet</a>	249
<a href="#">CMeaCleanDeviceNet</a>	
<a href="#">CMeaCleanDeviceNet</a> is the class to access the MEA Clean device	252
<a href="#">CMeaCoatDeviceNet</a>	
<a href="#">CMeaCoatDeviceNet</a> is the class to access the MEA Coat device	256
<a href="#">CMeaDeviceNet</a>	
Base class for MEA data acquisition devices	262
<a href="#">CMeaDigitalDataFunctionNet</a>	270
<a href="#">CMeaFeedbackFunctionNet</a>	272
<a href="#">CMeaImpedanceDeviceNet</a>	276
<a href="#">CMeasureTableDeviceNet</a>	
<a href="#">CMeasureTableDeviceNet</a> is the to control the MCS HLA device	277
<a href="#">CMeaSwitchDeviceNet</a>	
The class to control the USB-MEA-Switch	278
<a href="#">CMeaUSBDeviceNet</a>	
Class for data acquisition via ME and MEA USB amplifiers	280
<a href="#">CMeFunctionNet</a>	
281	
<a href="#">CMultiBatteryChargerDeviceNet</a>	
<a href="#">CMultiBatteryChargerDeviceNet</a> is the class to access the MBC-08 device	283
<a href="#">CMultiwellCallbackFunctionNet</a>	
<a href="#">CMultiwellCallbackFunctionNet</a> is the class to access the Multiwell-Mini-Stimulator	290
<a href="#">CMultiwellDeviceNet</a>	
<a href="#">CMultiwellDeviceNet</a> is the class to access the HEKA LIH3 device	293
<a href="#">CMultiwellOptoStimFunctionNet</a>	
<a href="#">CMultiwellOptoStimFunctionNet</a> is the class to access the optical properties of the Multiwell Optostim device	299
<a href="#">CNF_GenDeviceNet</a>	304
<a href="#">COctoPotDeviceNet</a>	304
<a href="#">CokuvisionStimulatorDeviceNet</a>	308
<a href="#">CPatchServerDeviceNet</a>	
<a href="#">CPatchServerDeviceNet</a> is the class to control the MCS PatchServer device	312
<a href="#">CPathIdentDeviceNet</a>	313
<a href="#">CPedoterDeviceNet</a>	
314	
<a href="#">CPeristalticPumpDeviceNet</a>	
<a href="#">CPeristalticPumpDeviceNet</a> is the class to control a Persistaltic Pump	315



<a href="#">CPgaDeviceNet</a>	316
<a href="#">CPositionIIDeviceNet</a>	
<a href="#">CPositionIIDeviceNet</a> is the class to control PositionII devices	318
<a href="#">CPositionImpDeviceNet</a>	
<a href="#">CPositionImpDeviceNet</a> is the class to access the Position/Imp devices	321
<a href="#">CPPCDeviceNet</a>	324
<a href="#">CPPCFunctionNet</a>	
<a href="#">CPPCFunctionNet</a> is the class to access the PPC (high precision Patch Peristaltic patch Pump	326
<a href="#">CPPS_DeviceNet</a>	332
<a href="#">CPPS_FunctionNet</a>	333
<a href="#">CPPSDeviceNet</a>	
<a href="#">CPPS4plus1DeviceNet</a> is the to control the MCS HLA device	338
<a href="#">CProgramPressureCurveNet</a>	
<a href="#">CProgramPressureCurveNet</a> is the class to program pressure curves	338
<a href="#">CPulseGeneratorFunctionNet</a>	
<a href="#">CPulseGeneratorFunctionNet</a> is the class to control the pulse generator for video tracking	340
<a href="#">CRadioControlledDevicesNet</a>	343
<a href="#">CCMOSMeaDeviceNet::CRegionOfInterestRect</a>	345
<a href="#">CRetinaLedDeviceNet</a>	346
<a href="#">CRFFFunctionNet</a>	
<a href="#">CRFFFunctionNet</a> is the class to control RF devices	348
<a href="#">CRobo_FYIProgram_FunctionNet</a>	352
<a href="#">CRobo_FYITemp_FunctionNet</a>	354
<a href="#">CRoboDacqNet</a>	356
<a href="#">CRoboDeviceNet</a>	
<a href="#">CRoboDeviceNet</a> is the base class for all Robo platform based devices	369
<a href="#">CRoboFluidDeviceNet</a>	383
<a href="#">CRoboInjectDeviceNet</a>	
<a href="#">CRoboInjectDeviceNet</a> is the to control the MCS RoboInject device	387
<a href="#">CRoboocyte2DeviceNet</a>	
<a href="#">CRoboocyte2DeviceNet</a> is the class to control the MCS Roboocyte2 device	387
<a href="#">CRoboStatorDeviceNet</a>	389
<a href="#">CSafeISDeviceNet</a>	
395	
<a href="#">CSCUDacqGroupChannelSelectionNet</a>	398
<a href="#">CSCUFunctionNet</a>	
<a href="#">CSCUFunctionNet</a> is the class to control the SCU device	399

<a href="#">CSerialPortNet</a>	414
<a href="#">CStg200xBasicNet</a> Base class for the Stg200x	416
<a href="#">CStg200xDownloadBasicNet</a> <a href="#">CStg200xDownloadBasicNet</a> is the base class to control the download mode of the MCS STG device	446
<a href="#">CStg200xDownloadNet</a> Main class for the STG download mode This class implements the STG download mode interface.	455
<a href="#">CStimulusFunctionNet</a>	462
<a href="#">CSw2to64DeviceNet</a> The class to control the MCS-USB-Sw2to64 device	473
<a href="#">CTcxDeviceNet</a> Class to control a Temperature Controller (TCX)	476
<a href="#">CTEERFunctionNet</a> <a href="#">CTEERFunctionNet</a> is the class to control the TEER device	489
<a href="#">CTEERMachineDeviceNet</a>	499
<a href="#">CUsbExceptionNet</a> Exception class that is thrown in case of an USB error	500
<a href="#">CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet</a>	501
<a href="#">CW2100_FunctionNet</a>	501
<a href="#">CW2100_StimulatorFunctionNet</a>	509
<a href="#">CW2100DacqGroupChannelSelectionNet</a>	515
<a href="#">CWarnerUssingDeviceNet</a> <a href="#">CWarnerUssingDeviceNet</a> is the class to control the Ussing device	516
<a href="#">CWarnerUssingFunctionNet</a> <a href="#">CWarnerUssingFunctionNet</a> is the class to control the Ussing device	517
<a href="#">CWarnerValveControllerDeviceNet</a> <a href="#">CWarnerValveControllerDeviceNet</a> is the class to access the Warner Valve Controller	534
<a href="#">CWarnerValveControllerDeviceTesterFunctionNet</a> <a href="#">CWarnerValveControllerDeviceTesterFunctionNet</a> is the class to access the functions for the Warner Valve Controller Device Tester	557
<a href="#">CWClassicFunctionNet</a>	560
<a href="#">CWirelessBaseFunctionNet</a>	565
<a href="#">DeviceIdNet</a> Device Id	565
<a href="#">DigitalSource&lt; digitalsourceenum &gt;</a>	567
<a href="#">DriverVersionNet</a> Class gives firmware versions of the device's firmware destinations	568

<a href="#">FirmwareDestinationNames</a>	573
<a href="#">HeadStageIDType</a>	577
<a href="#">HeadstageIDTypeObject</a>	580
<a href="#">HeadStageIDTypeState</a>	581
<a href="#">mkfilterNet</a>	582
<a href="#">CRoboDeviceNet::RoboMainLowLevelCommands</a>	585
<a href="#">CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands</a>	592
<a href="#">CMeaAudioFunctionNet::s_setaudionet</a>	592
<a href="#">CStimulusFunctionNet::SidebandData</a>	592
<a href="#">StgStatusNet</a>	593
<a href="#">CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData</a>	594
<a href="#">usbSetupPacket_t</a>	595
<a href="#">W2100_StimulusParametersNet</a>	596

## 10 Namespace Documentation

### 10.1 Mcs Namespace Reference

#### Namespaces

- [Usb](#)

### 10.2 Mcs::Usb Namespace Reference

#### Classes

- class [BatteryState](#)
- class [BesselFilterHighPassNet](#)
- class [BesselFilterLowPassNet](#)
- class [ButterworthFilterHighPassNet](#)
- class [ButterworthFilterLowPassNet](#)
- class [CChannelTestDeviceNet](#)
- class [CCMOSMea\\_FunctionNet](#)
- class [CCMOSMeaDeviceNet](#)
- class [CCreateFilterNet](#)
- class [CDacCalibrationFunctionNet](#)
- class [CDacqGroupChannelGenericSelectionNet](#)
- class [CDacqGroupChannelSelectionNet](#)
- class [CDacqGroupChannelSelectionTemplateNet](#)
- class [CDeviceGroupChannelInfoGenericNet](#)
- class [CDeviceGroupChannelInfoNet](#)
- class [CDeviceGroupChannelInfoSCUNet](#)

- class [CDeviceGroupChannelInfoTemplateNet](#)
- class [CDeviceGroupChannelInfoW2100Net](#)
- class [CDigOutStimulatorFunctionNet](#)  
*CDigOutStimulatorFunctionNet is the class of the DigOut stimulator function class.*
- class [CEncapsulatorDeviceNet](#)  
*CEncapsulatorDeviceNet is the to control the MCS HiClamp device*
- class [CExternDTesterDeviceNet](#)  
*CExternDTesterDeviceNet is the class to access the ExternD Tester (Handheld Device Tester D)*
- class [CFilterCoefficientsNet](#)
- class [CFilterConfigurationNet](#)
- class [CFilterConfigurationRegisterNet](#)
- class [CFilterPropertyNet](#)
- class [CFluidControlDeviceNet](#)  
*CFluidControlDeviceNet is the class to control MCS FluidControl (FCB and FCX) device.*
- class [CFYIDeviceNet](#)  
*CFYIDeviceNet is the class to control the MCS FYI device*
- class [CGenericDevelopDeviceNet](#)  
*CGenericDevelopDeviceNet is the class to use during development of a new device.*
- class [CGilsonDeviceNet](#)  
*CGilsonDeviceNet is the class to control a Gilson device.*
- class [CHiClampDeviceNet](#)  
*CHiClampDeviceNet is the to control the MCS HiClamp device*
- class [CHLADacqNet](#)
- class [CHLADeviceNet](#)  
*CHLADeviceNet is the to control the MCS HLA device*
- class [CIntanMea\\_FunctionNet](#)
- class [CInterfaceboardFunctionNet](#)  
*CInterfaceboardFunctionNet is the class to control the Interfaceboard*
- class [CLIH3DeviceNet](#)  
*CLIH3DeviceNet is the class to access the HEKA LIH3 device.*
- class [CMcsBus\\_AxisParametersNet](#)
- class [CMcsBus\\_ExtensionNet](#)
- class [CMcsBus\\_FYIExtensionNet](#)
- class [CMcsBus\\_MotorControlNet](#)
- class [CMcsBus\\_SensorNet](#)
- class [CMcsBus\\_TempSensorNet](#)
- class [CMcsBus\\_VoltageModeNet](#)
- class [CMcsBusNet](#)
- class [CMcsUsbDacqNet](#)  
*Base class for data acquisition devices.*
- class [CMcsUsbDeviceStatePushFunctionNet](#)
- class [CMcsUsbDeviceStatePushNet](#)
- class [CMcsUsbFactoryNet](#)
- class [CMcsUsbFunctionNet](#)
- class [CMcsUsbFunctionPointerContainer](#)
- class [CMcsUsbListEntryNet](#)  
*McsUsbListEntryNet identifies a connected device.*
- class [CMcsUsbListNet](#)  
*Class to handle a list of connected MCS USB devices.*
- class [CMcsUsbNet](#)  
*Base class to handle MCS USB devices. All device classes are derived from this class. Functionality that is provided by all MCS devices is handled by this class.*

- class [CMcsUsbPointerContainer](#)
- class [CMEA2100x256FunctionNet](#)  
*CMEA2100x256FunctionNet is the class to control the MEA2100-256 device needs #include "Stg200xNet.h" to resolve documentation reference*
- class [CMeaAudioFunctionNet](#)
- class [CMeaCleanDeviceNet](#)  
*CMeaCleanDeviceNet is the class to access the MEA Clean device.*
- class [CMeaCoatDeviceNet](#)  
*CMeaCoatDeviceNet is the class to access the MEA Coat device.*
- class [CMeaDeviceNet](#)  
*Base class for MEA data acquisition devices.*
- class [CMeaDigitalDataFunctionNet](#)
- class [CMeaFeedbackFunctionNet](#)
- class [CMeaImpedanceDeviceNet](#)
- class [CMeasureTableDeviceNet](#)  
*CMeasureTableDeviceNet is the to control the MCS HLA device*
- class [CMeaSwitchDeviceNet](#)  
*The class to control the USB-MEA-Switch.*
- class [CMeaUSBDeviceNet](#)  
*Class for data acquisition via ME and MEA USB amplifiers*
- class [CMeFunctionNet](#)
- class [CMultiBatteryChargerDeviceNet](#)  
*CMultiBatteryChargerDeviceNet is the class to access the MBC-08 device.*
- class [CMultiwellCallbackFunctionNet](#)  
*CMultiwellCallbackFunctionNet is the class to access the Multiwell-Mini-Stimulator*
- class [CMultiwellDeviceNet](#)  
*CMultiwellDeviceNet is the class to access the HEKA LIH3 device.*
- class [CMultiwellOptoStimFunctionNet](#)  
*CMultiwellOptoStimFunctionNet is the class to access the optical properties of the Multiwell Optostim device*
- class [CNF\\_GenDeviceNet](#)
- class [COctoPotDeviceNet](#)
- class [COKuvisionStimulatorDeviceNet](#)
- class [CPatchServerDeviceNet](#)  
*CPatchServerDeviceNet is the class to control the MCS PatchServer device*
- class [CPathIdentDeviceNet](#)
- class [CPedoterDeviceNet](#)
- class [CPeristalticPumpDeviceNet](#)  
*CPeristalticPumpDeviceNet is the class to control a Persistaltic Pump.*
- class [CPgaDeviceNet](#)
- class [CPositionIIDeviceNet](#)  
*CPositionIIDeviceNet is the class to control PositionII devices*
- class [CPositionImpDeviceNet](#)  
*CPositionImpDeviceNet is the class to access the Position/Imp devices*
- class [CPPCDeviceNet](#)
- class [CPPCFunctionNet](#)  
*CPPCFunctionNet is the class to access the PPC (high precision Patch Peristaltic patch Pump*
- class [CPPS\\_DeviceNet](#)
- class [CPPS\\_FunctionNet](#)
- class [CPPSDeviceNet](#)  
*CPPS4plus1DeviceNet is the to control the MCS HLA device*
- class [CProgramPressureCurveNet](#)  
*CProgramPressureCurveNet is the class to program pressure curves*

- class [CPulseGeneratorFunctionNet](#)  
*CPulseGeneratorFunctionNet is the class to control the pulse generator for video tracking*
- class [CRadioControlledDevicesNet](#)
- class [CRetinaLedDeviceNet](#)
- class [CRFFFunctionNet](#)  
*CRFFFunctionNet is the class to control RF devices*
- class [CRobo\\_FYIPProgram\\_FunctionNet](#)
- class [CRobo\\_FYITemp\\_FunctionNet](#)
- class [CRoboDacqNet](#)
- class [CRoboDeviceNet](#)  
*CRoboDeviceNet is the base class for all Robo platform based devices*
- class [CRoboFluidDeviceNet](#)
- class [CRoboInjectDeviceNet](#)  
*CRoboInjectDeviceNet is the to control the MCS RoboInject device*
- class [CRoboocyte2DeviceNet](#)  
*CRoboocyte2DeviceNet is the class to control the MCS Roboocyte2 device*
- class [CRoboStatorDeviceNet](#)
- class [CSafeISDeviceNet](#)
- class [CSCUDacqGroupChannelSelectionNet](#)
- class [CSCUFunctionNet](#)  
*CSCUFunctionNet is the class to control the SCU device*
- class [CSerialPortNet](#)
- class [CStg200xBasicNet](#)  
*Base class for the Stg200x.*
- class [CStg200xDownloadBasicNet](#)  
*CStg200xDownloadBasicNet is the base class to control the download mode of the MCS STG device.*
- class [CStg200xDownloadNet](#)  
*Main class for the STG download mode This class implements the STG download mode interface.*
- class [CStimulusFunctionNet](#)
- class [CSw2to64DeviceNet](#)  
*The class to control the MCS-USB-Sw2to64 device.*
- class [CTcxDeviceNet](#)  
*Class to control a Temperature Controller (TCX)*
- class [CTEERFunctionNet](#)  
*CTEERFunctionNet is the class to control the TEER device*
- class [CTEERMachineDeviceNet](#)
- class [CUsbExceptionNet](#)  
*Exception class that is thrown in case of an USB error.*
- class [CW2100\\_FunctionNet](#)
- class [CW2100\\_StimulatorFunctionNet](#)
- class [CW2100DacqGroupChannelSelectionNet](#)
- class [CWarnerUssingDeviceNet](#)  
*CWarnerUssingDeviceNet is the class to control the Ussing device*
- class [CWarnerUssingFunctionNet](#)  
*CWarnerUssingFunctionNet is the class to control the Ussing device*
- class [CWarnerValveControllerDeviceNet](#)  
*CWarnerValveControllerDeviceNet is the class to access the Warner Valve Controller*
- class [CWarnerValveControllerDeviceTesterFunctionNet](#)  
*CWarnerValveControllerDeviceTesterFunctionNet is the class to access the functions for the Warner Valve Controller Device Tester*
- class [CWClassicFunctionNet](#)
- class [CWirelessBaseFunctionNet](#)

- struct [DeviceIdNet](#)  
*Device Id.*
- class [DigitalSource](#)
- class [DriverVersionNet](#)  
*Class gives firmware versions of the device's firmware destinations.*
- class [FirmwareDestinationNames](#)
- class [HeadStageIDType](#)
- class [HeadstageIDTypeObject](#)
- class [HeadStageIDTypeState](#)
- class [mkfilterNet](#)
- class [StgStatusNet](#)
- class [usbSetupPacket\\_t](#)
- struct [W2100\\_StimulusParametersNet](#)

## Enumerations

- enum [enCMosMeaChipType](#) {  
    [unknown](#) = 0,  
    [nMos16LV](#) = 1,  
    [nMos32LV](#) = 3,  
    [nMos36LN](#) = 6,  
    [nMos64LN](#) = 7 }
- enum [EnSTG200x\\_STATUS](#) {  
    [OK](#),  
    [NOT\\_CONNECTED](#),  
    [DEVICE\\_NOT\\_FOUND](#) }

## Functions

- public delegate void [OnMcsUsbDeviceState](#) ([usbSetupPacket\\_t](#)^ request)
- private delegate void [OnMcsUsbDeviceStateCallback](#) (IntPtr pThis, uint32\_t size, IntPtr buffer)
- public delegate void [OnUpdateFirmwareStatusChange](#) (String^)
- public delegate void [OnUpdateFirmwareProgress](#) (int)
- public delegate void [OnDeviceArrivalRemoval](#) ([CMcsUsbListEntryNet](#)^ entry)  
*Delegate to show a device arrival or removal.*
- public delegate void [OnStgPollStatus](#) (unsigned int status, [StgStatusNet](#)^ stgStatusNet, array< int >^ index\_list)
- public delegate void [OnMwPollStatus](#) (unsigned int CurrentTemp, unsigned int PlateState, unsigned int SwitchState)
- public delegate void [RoboStatusEventDelegate](#) (array< unsigned char >^ buffer)
- public delegate void [OnStg200xDataHandler](#) (uint32\_t trigger)
- public delegate void [OnStg200xErrorHandler](#) ()
- public delegate void [OnChannelData](#) ([CMcsUsbDacqNet](#)^ dacq, int CbHandle, int numFrames)
- public delegate void [OnError](#) (String^ msg, int action)

### 10.2.1 Enumeration Type Documentation

#### 10.2.1.1 enCMosMeaChipType enum [enCMosMeaChipType](#) [strong]

## Enumerator

unknown	
nMos16LV	
nMos32LV	
nMos36LN	
nMos64LN	

**10.2.1.2 EnSTG200x\_STATUS** enum [EnSTG200x\\_STATUS](#) [strong]

## Enumerator

OK	
NOT_CONNECTED	
DEVICE_NOT_FOUND	

**10.2.2 Function Documentation**

**10.2.2.1 OnChannelData()** public delegate void Mcs::Usb::OnChannelData (   
 [CMcsUsbDacqNet](#)<sup>^</sup> *dacq*,   
 int *CbHandle*,   
 int *numFrames* )

**10.2.2.2 OnDeviceArrivalRemoval()** public delegate void Mcs::Usb::OnDeviceArrivalRemoval (   
 [CMcsUsbListEntryNet](#)<sup>^</sup> *entry* )

Delegate to show a device arrival or removal.

**10.2.2.3 OnError()** public delegate void Mcs::Usb::OnError (   
 String<sup>^</sup> *msg*,   
 int *action* )

**10.2.2.4 OnMcsUsbDeviceState()** public delegate void OnMcsUsbDeviceState (   
 [usbSetupPacket\\_t](#)<sup>^</sup> *request* )



**10.2.2.5 OnMcsUsbDeviceStateCallback()** private delegate void OnMcsUsbDeviceStateCallback (   
     IntPtr *pThis*,  
     uint32\_t *size*,  
     IntPtr *buffer* )

**10.2.2.6 OnMwPollStatus()** public delegate void Mcs::Usb::OnMwPollStatus (   
     unsigned int *CurrentTemp*,  
     unsigned int *PlateState*,  
     unsigned int *SwitchState* )

**10.2.2.7 OnStg200xDataHandler()** public delegate void Mcs::Usb::OnStg200xDataHandler (   
     uint32\_t *trigger* )

**10.2.2.8 OnStg200xErrorHandler()** public delegate void Mcs::Usb::OnStg200xErrorHandler ( )

**10.2.2.9 OnStgPollStatus()** public delegate void Mcs::Usb::OnStgPollStatus (   
     unsigned int *status*,  
     StgStatusNet^ *stgStatusNet*,  
     array< int >^ *index\_list* )

**10.2.2.10 OnUpdateFirmwareProgress()** public delegate void Mcs::Usb::OnUpdateFirmwareProgress   
 (   
     int )

**10.2.2.11 OnUpdateFirmwareStatusChange()** public delegate void Mcs::Usb::OnUpdateFirmware↵  
StatusChange (   
     String^ )

**10.2.2.12 RoboStatusEventDelegate()** public delegate void Mcs::Usb::RoboStatusEventDelegate (   
     array< unsigned char >^ *buffer* )

## 11 Class Documentation

### 11.1 CW2100\_FunctionNet::AudioChannelsNet Struct Reference

#### Public Attributes

- W2100DacqGroupChannelEnumNet [dacqgroup](#)
- int [channel](#)
- int [amplification](#)

#### 11.1.1 Member Data Documentation

**11.1.1.1 [amplification](#)**    int [amplification](#)

**11.1.1.2 [channel](#)**    int [channel](#)

**11.1.1.3 [dacqgroup](#)**    W2100DacqGroupChannelEnumNet [dacqgroup](#)

### 11.2 BatteryState Class Reference

#### Properties

- unsigned int [Charge](#)    [get]
- unsigned int [Voltage](#)    [get]
- System::String^ [ChargeString](#)    [get]
- System::String^ [ChargeRegionString](#)    [get]
- System::String^ [VoltageString](#)    [get]

#### 11.2.1 Property Documentation

**11.2.1.1 [Charge](#)**    unsigned int [Charge](#)    [get]

**11.2.1.2 [ChargeRegionString](#)**    System::String^ [ChargeRegionString](#)    [get]

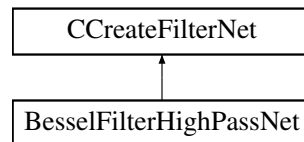
**11.2.1.3 ChargeString** `System:: String^ ChargeString [get]`

**11.2.1.4 Voltage** `unsigned int Voltage [get]`

**11.2.1.5 VoltageString** `System:: String^ VoltageString [get]`

## 11.3 BesselFilterHighPassNet Class Reference

Inheritance diagram for BesselFilterHighPassNet:



### Public Member Functions

- [BesselFilterHighPassNet](#) (int numCoefSets, int order, double sampleRate, double cutoffFrequency, double scale)

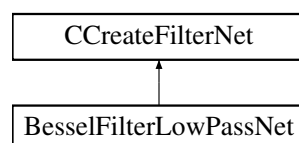
### Additional Inherited Members

#### 11.3.1 Constructor & Destructor Documentation

**11.3.1.1 BesselFilterHighPassNet()** `BesselFilterHighPassNet (`  
     int numCoefSets,  
     int order,  
     double sampleRate,  
     double cutoffFrequency,  
     double scale )

## 11.4 BesselFilterLowPassNet Class Reference

Inheritance diagram for BesselFilterLowPassNet:



## Public Member Functions

- [BesselFilterLowPassNet](#) (int numCoefSets, int order, double sampleRate, double cutoffFrequency, double scale)

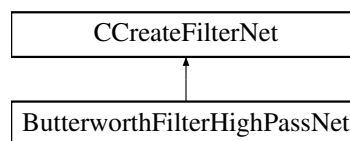
## Additional Inherited Members

### 11.4.1 Constructor & Destructor Documentation

**11.4.1.1 BesselFilterLowPassNet()** `BesselFilterLowPassNet (`  
     int numCoefSets,  
     int order,  
     double sampleRate,  
     double cutoffFrequency,  
     double scale )

## 11.5 ButterworthFilterHighPassNet Class Reference

Inheritance diagram for ButterworthFilterHighPassNet:



## Public Member Functions

- [ButterworthFilterHighPassNet](#) (int numCoefSets, int order, double sampleRate, double cutoffFrequency, double scale)

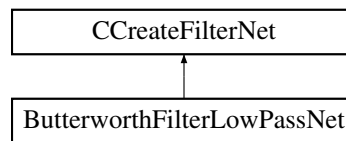
## Additional Inherited Members

### 11.5.1 Constructor & Destructor Documentation

**11.5.1.1 ButterworthFilterHighPassNet()** `ButterworthFilterHighPassNet (`  
     int numCoefSets,  
     int order,  
     double sampleRate,  
     double cutoffFrequency,  
     double scale )

## 11.6 ButterworthFilterLowPassNet Class Reference

Inheritance diagram for ButterworthFilterLowPassNet:



### Public Member Functions

- [ButterworthFilterLowPassNet](#) (int numCoefSets, int order, double sampleRate, double cutoffFrequency, double scale)

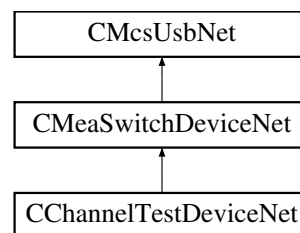
### Additional Inherited Members

#### 11.6.1 Constructor & Destructor Documentation

**11.6.1.1 ButterworthFilterLowPassNet()** [ButterworthFilterLowPassNet](#) (   
     int numCoefSets,   
     int order,   
     double sampleRate,   
     double cutoffFrequency,   
     double scale )

## 11.7 CChannelTestDeviceNet Class Reference

Inheritance diagram for CChannelTestDeviceNet:



### Public Member Functions

- [CChannelTestDeviceNet](#) ()
- [~CChannelTestDeviceNet](#) ()
- void [SetWaveform](#) (unsigned int Waveform)
- void [SetAmplitude](#) (unsigned int Amplitude)
- void [SetFrequency](#) (unsigned int Frequency)
- void [SetAttenuation](#) (unsigned int Attenuation)

**Additional Inherited Members****11.7.1 Constructor & Destructor Documentation**

**11.7.1.1 CChannelTestDeviceNet()** `CChannelTestDeviceNet ( )`

**11.7.1.2 ~CChannelTestDeviceNet()** `~CChannelTestDeviceNet ( )`

**11.7.2 Member Function Documentation**

**11.7.2.1 SetAmplitude()** `void SetAmplitude ( unsigned int Amplitude )`

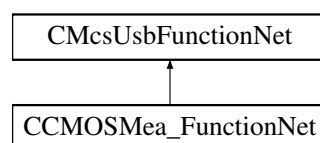
**11.7.2.2 SetAttenuation()** `void SetAttenuation ( unsigned int Attenuation )`

**11.7.2.3 SetFrequency()** `void SetFrequency ( unsigned int Frequency )`

**11.7.2.4 SetWaveform()** `void SetWaveform ( unsigned int Waveform )`

**11.8 CCMOSMea\_FunctionNet Class Reference**

Inheritance diagram for CCMOSMea\_FunctionNet:



## Public Member Functions

- [CCMOSMea\\_FunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> cMOSMea\_↔  
FunctionPointerContainer)
- [CCMOSMea\\_FunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- void [SetADCInputOffset](#) (int32\_t offset)
- int32\_t [GetADCInputOffset](#) ()
- void [SetSourceDrain](#) (int32\_t voltage)
- int32\_t [GetSourceDrain](#) ()
- void [SetSourceGate](#) (int32\_t voltage)
- int32\_t [GetSourceGate](#) ()
- void [SetSourceBulk](#) (int32\_t voltage)
- int32\_t [GetSourceBulk](#) ()
- void [SetGate](#) (int32\_t voltage)
- int32\_t [GetGate](#) ()
- void [SetBath](#) (int32\_t voltage)
- int32\_t [GetBath](#) ()
- int32\_t [GetGNDI](#) ()
- int32\_t [GetVDDI](#) ()
- int32\_t [GetVDD3I](#) ()
- void [UpdateTransistorVoltages](#) ()
- bool [AreTransistorVoltagesSet](#) ()
- void [PowerChip](#) (bool on)
- bool [IsChipPowered](#) ()
- [enCMosMeaChipType](#) [DetectChipType](#) ()
- void [SetGateToVOP](#) ()
- void [SetGateFloating](#) ()
- bool [IsGateFloating](#) ()
- void [VOPSTimerSetResetTimes](#) (uint32\_t ResetTime, uint32\_t IntervalTime)
- void [VOPSTimerSetResetTimes](#) (uint32\_t ResetTime, uint32\_t IntervalTime, uint32\_t HPFilterResetTime)
- void [SetBathMode](#) ([CMOSMeaBathModeEnumNet](#) Mode)
- [CMOSMeaBathModeEnumNet](#) [GetBathMode](#) ()
- void [SetNeurochipMemoryData](#) (uint16\_t MemAddress, uint32\_t MemData)
- void [SetNeurochipMemoryData](#) (uint16\_t MemAddress, array< uint32\_t ><sup>^</sup> MemData)
- uint32\_t [GetNeurochipMemoryData](#) (uint16\_t MemAddress)
- array< uint32\_t ><sup>^</sup> [GetNeurochipMemoryData](#) (uint16\_t MemAddress, uint32\_t RequestLength)
- uint32\_t [GetNeurochipMemorySize](#) ()
- uint32\_t [GetMaxNumOfColumns](#) (uint32\_t Samplerate)
- void [SetStimulusSites](#) (List< int16\_t ><sup>^</sup> SwitchPosition)
- List< int16\_t ><sup>^</sup> [GetStimulusSites](#) ()
- void [ClearSTGOutput](#) (uint32\_t Channel)
- uint32\_t [GetNumberOfSupportedGroups](#) ()
- uint32\_t [GetNumberOfSupportedGroups](#) (uint32\_t virtualDevice)
- [DacqGroupChannelEnumNet](#) [GetGroupID](#) (uint32\_t Index)
- [DacqGroupChannelEnumNet](#) [GetGroupID](#) (uint32\_t Index, uint32\_t virtualDevice)
- uint32\_t [GetGroupNumberOfChannels](#) ([DacqGroupChannelEnumNet](#) GroupID)
- uint32\_t [GetGroupNumberOfChannels](#) ([DacqGroupChannelEnumNet](#) GroupID, uint32\_t virtualDevice)
- [DacqMeaGroupTypeEnumNet](#) [GetGroupType](#) ([DacqGroupChannelEnumNet](#) GroupID)
- [DacqMeaGroupTypeEnumNet](#) [GetGroupType](#) ([DacqGroupChannelEnumNet](#) GroupID, uint32\_t virtual↔  
Device)
- void [EnableChannelsInGroup](#) ([DacqGroupChannelEnumNet](#) GroupID, List< bool ><sup>^</sup> EnabledChannelsBit↔  
Map)
- void [EnableChannelsInGroup](#) ([DacqGroupChannelEnumNet](#) GroupID, List< bool ><sup>^</sup> EnabledChannelsBit↔  
Map, uint32\_t virtualDevice)
- List< bool ><sup>^</sup> [GetEnabledChannelsInGroup](#) ([DacqGroupChannelEnumNet](#) GroupID)

- List< bool > ^ [GetEnabledChannelsInGroup](#) (DacqGroupChannelEnumNet GroupID, uint32\_t virtualDevice)
- SampleSizeNet [GetGroupSampleSize](#) (DacqGroupChannelEnumNet GroupID)
- SampleSizeNet [GetGroupSampleSize](#) (DacqGroupChannelEnumNet GroupID, uint32\_t virtualDevice)
- uint32\_t [GetGroupResolutionPerDigit](#) (DacqGroupChannelEnumNet GroupID)
- uint32\_t [GetGroupResolutionPerDigit](#) (DacqGroupChannelEnumNet GroupID, uint32\_t virtualDevice)
- CMOSMeaValueUnitEnumNet [GetGroupUnit](#) (DacqGroupChannelEnumNet GroupID)
- CMOSMeaValueUnitEnumNet [GetGroupUnit](#) (DacqGroupChannelEnumNet GroupID, uint32\_t virtualDevice)
- int32\_t [GetGroupDCOffset](#) (DacqGroupChannelEnumNet GroupID)
- int32\_t [GetGroupDCOffset](#) (DacqGroupChannelEnumNet GroupID, uint32\_t virtualDevice)
- int32\_t [GetGroupADCBits](#) (DacqGroupChannelEnumNet GroupID)
- int32\_t [GetGroupADCBits](#) (DacqGroupChannelEnumNet GroupID, uint32\_t virtualDevice)
- uint32\_t [GetGroupChannelBitmaskBySelect](#) (DacqGroupChannelEnumNet GroupID, uint32\_t Channel↔  
Number)
- uint32\_t [GetGroupChannelBitmaskBySelect](#) (DacqGroupChannelEnumNet GroupID, uint32\_t Channel↔  
Number, uint32\_t virtualDevice)
- CMOSMeaInterfaceADCEnumNet [GetGroupChannelBitmaskInterfaceADC](#) (uint32\_t ChannelNumber)
- CMOSMeaInterfaceADCEnumNet [GetGroupChannelBitmaskInterfaceADC](#) (uint32\_t ChannelNumber,  
uint32\_t virtualDevice)
- CMOSMeaIFDigChannelEnumNet [GetGroupChannelBitmaskIFDigChannels](#) (uint32\_t ChannelNumber)
- CMOSMeaIFDigChannelEnumNet [GetGroupChannelBitmaskIFDigChannels](#) (uint32\_t ChannelNumber,  
uint32\_t virtualDevice)
- CMOSMeaHeadstage1NCBathCurrentEnumNet [GetGroupChannelBitmaskHS1NCBathCurrent](#) (uint32\_t  
ChannelNumber)
- CMOSMeaHeadstage1NCBathCurrentEnumNet [GetGroupChannelBitmaskHS1NCBathCurrent](#) (uint32\_t  
ChannelNumber, uint32\_t virtualDevice)
- CMOSMeaHeadstage1NCCol2CurrentEnumNet [GetGroupChannelBitmaskHS1NCCol2Current](#) (uint32\_t  
ChannelNumber)
- CMOSMeaHeadstage1NCCol2CurrentEnumNet [GetGroupChannelBitmaskHS1NCCol2Current](#) (uint32\_t  
ChannelNumber, uint32\_t virtualDevice)
- CMOSMeaHeadstage1NChipTempEnumNet [GetGroupChannelBitmaskHS1NChipTemp](#) (uint32\_t Channel↔  
Number)
- CMOSMeaHeadstage1NChipTempEnumNet [GetGroupChannelBitmaskHS1NChipTemp](#) (uint32\_t Channel↔  
Number, uint32\_t virtualDevice)
- CMOSMeaSTG1DACSignalEnumNet [GetGroupChannelBitmaskSTG1DACSignal](#) (uint32\_t Channel↔  
Number)
- CMOSMeaSTG1DACSignalEnumNet [GetGroupChannelBitmaskSTG1DACSignal](#) (uint32\_t Channel↔  
Number, uint32\_t virtualDevice)
- CMOSMeaHS1SidebandEnumNet [GetGroupChannelBitmaskHS1Sidebands](#) (uint32\_t ChannelNumber)
- CMOSMeaHS1SidebandEnumNet [GetGroupChannelBitmaskHS1Sidebands](#) (uint32\_t ChannelNumber,  
uint32\_t virtualDevice)
- CMOSMeaHS1TriggerStatusEnumNet [GetGroupChannelBitmaskHS1TriggerStatus](#) (uint32\_t Channel↔  
Number)
- CMOSMeaHS1TriggerStatusEnumNet [GetGroupChannelBitmaskHS1TriggerStatus](#) (uint32\_t Channel↔  
Number, uint32\_t virtualDevice)
- CMOSMeaPacketFrameContextGroupEnumNet [GetGroupChannelBitmaskPacketFrameContext](#) (uint32\_↔  
t ChannelNumber)
- CMOSMeaPacketFrameContextGroupEnumNet [GetGroupChannelBitmaskPacketFrameContext](#) (uint32\_↔  
t ChannelNumber, uint32\_t virtualDevice)

## Additional Inherited Members

### 11.8.1 Constructor & Destructor Documentation



**11.8.1.1 CCMOSMea\_FunctionNet()** [1/2] `CCMOSMea_FunctionNet ( CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ cMOSMea_FunctionPointerContainer )`

**11.8.1.2 CCMOSMea\_FunctionNet()** [2/2] `CCMOSMea_FunctionNet ( CMcsUsbNet^ mcsusb )`

## 11.8.2 Member Function Documentation

**11.8.2.1 AreTransistorVoltagesSet()** `bool AreTransistorVoltagesSet ( )`

**11.8.2.2 ClearSTGOutput()** `void ClearSTGOutput ( uint32_t Channel )`

**11.8.2.3 DetectChipType()** `enCMosMeaChipType DetectChipType ( )`

**11.8.2.4 EnableChannelsInGroup()** [1/2] `void EnableChannelsInGroup ( DacqGroupChannelEnumNet GroupID, List< bool >^ EnabledChannelsBitMap )`

**11.8.2.5 EnableChannelsInGroup()** [2/2] `void EnableChannelsInGroup ( DacqGroupChannelEnumNet GroupID, List< bool >^ EnabledChannelsBitMap, uint32_t virtualDevice )`

**11.8.2.6 GetADCInputOffset()** `int32_t GetADCInputOffset ( )`

**11.8.2.7 GetBath()** `int32_t GetBath ( )`

**11.8.2.8 GetBathMode()** CMOSMeaBathModeEnumNet GetBathMode ( )

**11.8.2.9 GetEnabledChannelsInGroup()** [1/2] List<bool> ^ GetEnabledChannelsInGroup ( DacqGroupChannelEnumNet GroupID )

**11.8.2.10 GetEnabledChannelsInGroup()** [2/2] List<bool> ^ GetEnabledChannelsInGroup ( DacqGroupChannelEnumNet GroupID, uint32\_t virtualDevice )

**11.8.2.11 GetGate()** int32\_t GetGate ( )

**11.8.2.12 GetGNDI()** int32\_t GetGNDI ( )

**11.8.2.13 GetGroupADCBits()** [1/2] int32\_t GetGroupADCBits ( DacqGroupChannelEnumNet GroupID )

**11.8.2.14 GetGroupADCBits()** [2/2] int32\_t GetGroupADCBits ( DacqGroupChannelEnumNet GroupID, uint32\_t virtualDevice )

**11.8.2.15 GetGroupChannelBitmaskBySelect()** [1/2] uint32\_t GetGroupChannelBitmaskBySelect ( DacqGroupChannelEnumNet GroupID, uint32\_t ChannelNumber )

**11.8.2.16 GetGroupChannelBitmaskBySelect()** [2/2] uint32\_t GetGroupChannelBitmaskBySelect ( DacqGroupChannelEnumNet GroupID, uint32\_t ChannelNumber, uint32\_t virtualDevice )

**11.8.2.17 GetGroupChannelBitmaskHS1NCBathCurrent() [1/2]** CMOSMeaHeadstage1NCBathCurrentEnum↔  
 Net GetGroupChannelBitmaskHS1NCBathCurrent (   
     uint32\_t ChannelNumber )

**11.8.2.18 GetGroupChannelBitmaskHS1NCBathCurrent() [2/2]** CMOSMeaHeadstage1NCBathCurrentEnum↔  
 Net GetGroupChannelBitmaskHS1NCBathCurrent (   
     uint32\_t ChannelNumber,   
     uint32\_t virtualDevice )

**11.8.2.19 GetGroupChannelBitmaskHS1NCCol2Current() [1/2]** CMOSMeaHeadstage1NCCol2CurrentEnum↔  
 Net GetGroupChannelBitmaskHS1NCCol2Current (   
     uint32\_t ChannelNumber )

**11.8.2.20 GetGroupChannelBitmaskHS1NCCol2Current() [2/2]** CMOSMeaHeadstage1NCCol2CurrentEnum↔  
 Net GetGroupChannelBitmaskHS1NCCol2Current (   
     uint32\_t ChannelNumber,   
     uint32\_t virtualDevice )

**11.8.2.21 GetGroupChannelBitmaskHS1NChipTemp() [1/2]** CMOSMeaHeadstage1NChipTempEnumNet Get↔  
 GroupChannelBitmaskHS1NChipTemp (   
     uint32\_t ChannelNumber )

**11.8.2.22 GetGroupChannelBitmaskHS1NChipTemp() [2/2]** CMOSMeaHeadstage1NChipTempEnumNet Get↔  
 GroupChannelBitmaskHS1NChipTemp (   
     uint32\_t ChannelNumber,   
     uint32\_t virtualDevice )

**11.8.2.23 GetGroupChannelBitmaskHS1Sidebands() [1/2]** CMOSMeaHS1SidebandEnumNet GetGroup↔  
 ChannelBitmaskHS1Sidebands (   
     uint32\_t ChannelNumber )

**11.8.2.24 GetGroupChannelBitmaskHS1Sidebands() [2/2]** CMOSMeaHS1SidebandEnumNet GetGroup↔  
 ChannelBitmaskHS1Sidebands (   
     uint32\_t ChannelNumber,   
     uint32\_t virtualDevice )

**11.8.2.25 GetGroupChannelBitmaskHS1TriggerStatus()** [1/2] CMOSMeaHS1TriggerStatusEnumNet Get↔  
 GroupChannelBitmaskHS1TriggerStatus (   
     uint32\_t *ChannelNumber* )

**11.8.2.26 GetGroupChannelBitmaskHS1TriggerStatus()** [2/2] CMOSMeaHS1TriggerStatusEnumNet Get↔  
 GroupChannelBitmaskHS1TriggerStatus (   
     uint32\_t *ChannelNumber*,  
     uint32\_t *virtualDevice* )

**11.8.2.27 GetGroupChannelBitmaskIFDigChannels()** [1/2] CMOSMeaIFDigChannelEnumNet GetGroup↔  
 ChannelBitmaskIFDigChannels (   
     uint32\_t *ChannelNumber* )

**11.8.2.28 GetGroupChannelBitmaskIFDigChannels()** [2/2] CMOSMeaIFDigChannelEnumNet GetGroup↔  
 ChannelBitmaskIFDigChannels (   
     uint32\_t *ChannelNumber*,  
     uint32\_t *virtualDevice* )

**11.8.2.29 GetGroupChannelBitmaskInterfaceADC()** [1/2] CMOSMeaInterfaceADCEnumNet GetGroup↔  
 ChannelBitmaskInterfaceADC (   
     uint32\_t *ChannelNumber* )

**11.8.2.30 GetGroupChannelBitmaskInterfaceADC()** [2/2] CMOSMeaInterfaceADCEnumNet GetGroup↔  
 ChannelBitmaskInterfaceADC (   
     uint32\_t *ChannelNumber*,  
     uint32\_t *virtualDevice* )

**11.8.2.31 GetGroupChannelBitmaskPacketFrameContext()** [1/2] CMOSMeaPacketFrameContextGroup↔  
 EnumNet GetGroupChannelBitmaskPacketFrameContext (   
     uint32\_t *ChannelNumber* )

**11.8.2.32 GetGroupChannelBitmaskPacketFrameContext()** [2/2] CMOSMeaPacketFrameContextGroup↔  
 EnumNet GetGroupChannelBitmaskPacketFrameContext (   
     uint32\_t *ChannelNumber*,  
     uint32\_t *virtualDevice* )

**11.8.2.33 GetGroupChannelBitmaskSTG1DACSignal() [1/2]** CMOSMeaSTG1DACSignalEnumNet GetGroup↔  
ChannelBitmaskSTG1DACSignal (   
     uint32\_t *ChannelNumber* )

**11.8.2.34 GetGroupChannelBitmaskSTG1DACSignal() [2/2]** CMOSMeaSTG1DACSignalEnumNet GetGroup↔  
ChannelBitmaskSTG1DACSignal (   
     uint32\_t *ChannelNumber*,  
     uint32\_t *virtualDevice* )

**11.8.2.35 GetGroupDCOffset() [1/2]** int32\_t GetGroupDCOffset (   
     DacqGroupChannelEnumNet *GroupID* )

**11.8.2.36 GetGroupDCOffset() [2/2]** int32\_t GetGroupDCOffset (   
     DacqGroupChannelEnumNet *GroupID*,  
     uint32\_t *virtualDevice* )

**11.8.2.37 GetGroupID() [1/2]** DacqGroupChannelEnumNet GetGroupID (   
     uint32\_t *Index* )

**11.8.2.38 GetGroupID() [2/2]** DacqGroupChannelEnumNet GetGroupID (   
     uint32\_t *Index*,  
     uint32\_t *virtualDevice* )

**11.8.2.39 GetGroupNumberOfChannels() [1/2]** uint32\_t GetGroupNumberOfChannels (   
     DacqGroupChannelEnumNet *GroupID* )

**11.8.2.40 GetGroupNumberOfChannels() [2/2]** uint32\_t GetGroupNumberOfChannels (   
     DacqGroupChannelEnumNet *GroupID*,  
     uint32\_t *virtualDevice* )

**11.8.2.41 GetGroupResolutionPerDigit() [1/2]** uint32\_t GetGroupResolutionPerDigit (   
     DacqGroupChannelEnumNet *GroupID* )

**11.8.2.42 GetGroupResolutionPerDigit()** [2/2] uint32\_t GetGroupResolutionPerDigit (   
 DacqGroupChannelEnumNet GroupID,   
 uint32\_t virtualDevice )

**11.8.2.43 GetGroupSampleSize()** [1/2] SampleSizeNet GetGroupSampleSize (   
 DacqGroupChannelEnumNet GroupID )

**11.8.2.44 GetGroupSampleSize()** [2/2] SampleSizeNet GetGroupSampleSize (   
 DacqGroupChannelEnumNet GroupID,   
 uint32\_t virtualDevice )

**11.8.2.45 GetGroupType()** [1/2] DacqMeaGroupTypeEnumNet GetGroupType (   
 DacqGroupChannelEnumNet GroupID )

**11.8.2.46 GetGroupType()** [2/2] DacqMeaGroupTypeEnumNet GetGroupType (   
 DacqGroupChannelEnumNet GroupID,   
 uint32\_t virtualDevice )

**11.8.2.47 GetGroupUnit()** [1/2] CMOSMeaValueUnitEnumNet GetGroupUnit (   
 DacqGroupChannelEnumNet GroupID )

**11.8.2.48 GetGroupUnit()** [2/2] CMOSMeaValueUnitEnumNet GetGroupUnit (   
 DacqGroupChannelEnumNet GroupID,   
 uint32\_t virtualDevice )

**11.8.2.49 GetMaxNumOfColumns()** uint32\_t GetMaxNumOfColumns (   
 uint32\_t Samplerate )

**11.8.2.50 GetNeurochipMemoryData()** [1/2] uint32\_t GetNeurochipMemoryData (   
 uint16\_t MemAddress )

**11.8.2.51 GetNeurochipMemoryData()** [2/2] array<uint32\_t> ^ GetNeurochipMemoryData (   
     uint16\_t *MemAddress*,  
     uint32\_t *RegestLength* )

**11.8.2.52 GetNeurochipMemorySize()** uint32\_t GetNeurochipMemorySize ( )

**11.8.2.53 GetNumberOfSupportedGroups()** [1/2] uint32\_t GetNumberOfSupportedGroups ( )

**11.8.2.54 GetNumberOfSupportedGroups()** [2/2] uint32\_t GetNumberOfSupportedGroups (   
     uint32\_t *virtualDevice* )

**11.8.2.55 GetSourceBulk()** int32\_t GetSourceBulk ( )

**11.8.2.56 GetSourceDrain()** int32\_t GetSourceDrain ( )

**11.8.2.57 GetSourceGate()** int32\_t GetSourceGate ( )

**11.8.2.58 GetStimulusSites()** List<int16\_t> ^ GetStimulusSites ( )

**11.8.2.59 GetVDD3I()** int32\_t GetVDD3I ( )

**11.8.2.60 GetVDDI()** int32\_t GetVDDI ( )

**11.8.2.61 IsChipPowered()** bool IsChipPowered ( )

**11.8.2.62 IsGateFloating()** `bool IsGateFloating ( )`

**11.8.2.63 PowerChip()** `void PowerChip (`  
`bool on )`

**11.8.2.64 SetADCInputOffset()** `void SetADCInputOffset (`  
`int32_t offset )`

**11.8.2.65 SetBath()** `void SetBath (`  
`int32_t voltage )`

**11.8.2.66 SetBathMode()** `void SetBathMode (`  
`CMOSMeaBathModeEnumNet Mode )`

**11.8.2.67 SetGate()** `void SetGate (`  
`int32_t voltage )`

**11.8.2.68 SetGateFloating()** `void SetGateFloating ( )`

**11.8.2.69 SetGateToVOP()** `void SetGateToVOP ( )`

**11.8.2.70 SetNeurochipMemoryData() [1/2]** `void SetNeurochipMemoryData (`  
`uint16_t MemAddress,`  
`array< uint32_t >^ MemData )`

**11.8.2.71 SetNeurochipMemoryData() [2/2]** `void SetNeurochipMemoryData (`  
`uint16_t MemAddress,`  
`uint32_t MemData )`



**11.8.2.72 SetSourceBulk()** void SetSourceBulk (   
 int32\_t voltage )

**11.8.2.73 SetSourceDrain()** void SetSourceDrain (   
 int32\_t voltage )

**11.8.2.74 SetSourceGate()** void SetSourceGate (   
 int32\_t voltage )

**11.8.2.75 SetStimulusSites()** void SetStimulusSites (   
 List< int16\_t >^ SwitchPosition )

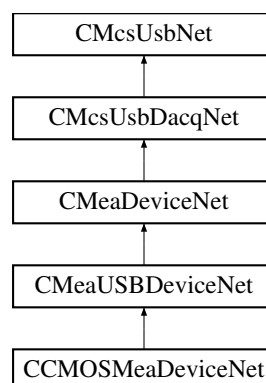
**11.8.2.76 UpdateTransistorVoltages()** void UpdateTransistorVoltages ( )

**11.8.2.77 VOPSTimerSetResetTimes()** [1/2] void VOPSTimerSetResetTimes (   
 uint32\_t ResetTime,   
 uint32\_t IntervalTime )

**11.8.2.78 VOPSTimerSetResetTimes()** [2/2] void VOPSTimerSetResetTimes (   
 uint32\_t ResetTime,   
 uint32\_t IntervalTime,   
 uint32\_t HPFilterResetTime )

## 11.9 CCMOSMeaDeviceNet Class Reference

Inheritance diagram for CCMOSMeaDeviceNet:



## Classes

- class [CRegionOfInterestRect](#)

## Public Member Functions

- [CCMOSMeaDeviceNet](#) (void)
- [~CCMOSMeaDeviceNet](#) ()
- void [SetBaseSamplerate](#) (int BaseSamplerate)
- int [GetBaseSamplerate](#) ()
- array< int > ^ [GetAvailableBaseSamplerates](#) ()
- void [SetRegionOfInterests](#) (System::Collections::Generic::Dictionary< int, [CRegionOfInterestRect](#)^>^ rois)
- void [UpdateChannelBlock](#) (int queuesize, int threshold, int channels\_in\_block)
- System::Collections::Generic::Dictionary< int, array< array< int16\_t >^>^ ^ [GetCMOSDataDictionary](#) (int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)
- System::Collections::Generic::Dictionary< int, array< uint16\_t >^> ^ [GetChannelDataUI16](#) (DacqGroup↔ ChannelEnumNet group, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)
- System::Collections::Generic::Dictionary< int, array< int16\_t >^> ^ [GetChannelDataI16](#) (DacqGroup↔ ChannelEnumNet group, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)
- System::Collections::Generic::Dictionary< int, array< uint32\_t >^> ^ [GetChannelDataUI32](#) (DacqGroup↔ ChannelEnumNet group, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)
- System::Collections::Generic::Dictionary< int, array< int32\_t >^> ^ [GetChannelDataI32](#) (DacqGroup↔ ChannelEnumNet group, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)

## Properties

- [CCMOSMea\\_FunctionNet](#)^ [CMosMea](#) [get]
- [CStimulusFunctionNet](#)^ [Stimulus](#) [get]

## Additional Inherited Members

### 11.9.1 Constructor & Destructor Documentation

**11.9.1.1 CCMOSMeaDeviceNet()** [CCMOSMeaDeviceNet](#) ( void )

**11.9.1.2 ~CCMOSMeaDeviceNet()** [~CCMOSMeaDeviceNet](#) ( )

### 11.9.2 Member Function Documentation

**11.9.2.1 GetAvailableBaseSamplerates()** `array<int> ^ GetAvailableBaseSamplerates ( )`

**11.9.2.2 GetBaseSamplerate()** `int GetBaseSamplerate ( )`

**11.9.2.3 GetChannelDataI16()** `System::Collections::Generic::Dictionary<int, array<int16_t>^> ^ GetChannelDataI16 (`  
`DacqGroupChannelEnumNet group,`  
`int frames,`  
`[System::Runtime::InteropServices::Out] int % frames_ret )`

**11.9.2.4 GetChannelDataI32()** `System::Collections::Generic::Dictionary<int, array<int32_t>^> ^ GetChannelDataI32 (`  
`DacqGroupChannelEnumNet group,`  
`int frames,`  
`[System::Runtime::InteropServices::Out] int % frames_ret )`

**11.9.2.5 GetChannelDataUI16()** `System::Collections::Generic::Dictionary<int, array<uint16_t>^> ^ GetChannelDataUI16 (`  
`DacqGroupChannelEnumNet group,`  
`int frames,`  
`[System::Runtime::InteropServices::Out] int % frames_ret )`

**11.9.2.6 GetChannelDataUI32()** `System::Collections::Generic::Dictionary<int, array<uint32_t>^> ^ GetChannelDataUI32 (`  
`DacqGroupChannelEnumNet group,`  
`int frames,`  
`[System::Runtime::InteropServices::Out] int % frames_ret )`

**11.9.2.7 GetCMOSDataDictionary()** `System::Collections::Generic::Dictionary<int, array<array<int16_t>^>^> ^ GetCMOSDataDictionary (`  
`int frames,`  
`[System::Runtime::InteropServices::Out] int % frames_ret )`

**11.9.2.8 SetBaseSamplerate()** `void SetBaseSamplerate (`  
`int BaseSamplerate )`

**11.9.2.9 SetRegionOfInterests()** void SetRegionOfInterests (   
 System::Collections::Generic::Dictionary< int, CRegionOfInterestRect^>^ rois )

**11.9.2.10 UpdateChannelBlock()** void UpdateChannelBlock (   
 int queuesize,   
 int threshold,   
 int channels\_in\_block )

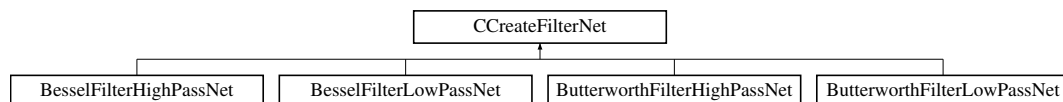
### 11.9.3 Property Documentation

**11.9.3.1 CMosMea** CCMOSMea\_FunctionNet^ CMosMea [get]

**11.9.3.2 Stimulus** CStimulusFunctionNet^ Stimulus [get]

## 11.10 CCreateFilterNet Class Reference

Inheritance diagram for CCreateFilterNet:



### Public Member Functions

- CCreateFilterNet (int numCoefSets, int order, double sampleRate, double cutoffFrequency, double scale)
- ~CCreateFilterNet ()
- CFilterCoefficientsNet ^ GetBiQuad (int index)
- array< CFilterCoefficientsNet^> ^ GetBiQuads ()

### Static Public Member Functions

- static int FindFilter (array< CFilterCoefficientsNet^>^ coef, array< CCreateFilterNet^>^ param)

### Protected Member Functions

- CCreateFilterNet (int numCoefSets, CCreateFilter \*pCreateFilter)

## Properties

- int [NumCoefSets](#) [get]
- int [Order](#) [get]
- double [SampleRate](#) [get]
- double [CutoffFrequency](#) [get]
- double [Scale](#) [get]

## 11.10.1 Constructor & Destructor Documentation

**11.10.1.1 CCreateFilterNet()** [1/2] [CCreateFilterNet](#) (  
     int *numCoefSets*,  
     int *order*,  
     double *sampleRate*,  
     double *cutoffFrequency*,  
     double *scale* )

**11.10.1.2 ~CCreateFilterNet()** [~CCreateFilterNet](#) ( )

**11.10.1.3 CCreateFilterNet()** [2/2] [CCreateFilterNet](#) (  
     int *numCoefSets*,  
     CCreateFilter \* *pCreateFilter* ) [protected]

## 11.10.2 Member Function Documentation

**11.10.2.1 FindFilter()** static int FindFilter (  
     array< [CFilterCoefficientsNet](#)>^ *coef*,  
     array< [CCreateFilterNet](#)>^ *param* ) [static]

**11.10.2.2 GetBiQuad()** [CFilterCoefficientsNet](#) ^ GetBiQuad (  
     int *index* )

**11.10.2.3 GetBiQuads()** array<[CFilterCoefficientsNet](#)> ^ GetBiQuads ( )

### 11.10.3 Property Documentation

**11.10.3.1 CutoffFrequency** `double CutoffFrequency [get]`

**11.10.3.2 NumCoefSets** `int NumCoefSets [get]`

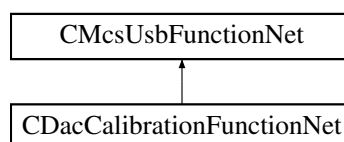
**11.10.3.3 Order** `int Order [get]`

**11.10.3.4 SampleRate** `double SampleRate [get]`

**11.10.3.5 Scale** `double Scale [get]`

## 11.11 CDacCalibrationFunctionNet Class Reference

Inheritance diagram for CDacCalibrationFunctionNet:



### Public Member Functions

- [CDacCalibrationFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> pDac↔  
CalibrationFunctionPointerContainer)  
*Initializes a new instance of the [CDacCalibrationFunctionNet](#) class.*
- [CDacCalibrationFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- virtual [~CDacCalibrationFunctionNet](#) ()
- [!CDacCalibrationFunctionNet](#) ()
- void [SetDacOffset](#) (uint16\_t dacChannel, uint32\_t offset)  
*Sets the offset of a DAC channel.*
- uint32\_t [GetDacOffset](#) (uint16\_t dacChannel)  
*Gets the offset of a DAC channel.*
- void [BurnDacOffset](#) (uint16\_t dacChannel)  
*Writes the offset of a DAC channel to permanent memory.*

## Additional Inherited Members

### 11.11.1 Detailed Description

### 11.11.2 Constructor & Destructor Documentation

**11.11.2.1 CDacCalibrationFunctionNet()** [1/2] `CDacCalibrationFunctionNet ( CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ pDacCalibrationFunctionPointerContainer )`

Initializes a new instance of the `CDacCalibrationFunctionNet` class.

**11.11.2.2 CDacCalibrationFunctionNet()** [2/2] `CDacCalibrationFunctionNet ( CMcsUsbNet^ mcsusb )`

**11.11.2.3 ~CDacCalibrationFunctionNet()** `virtual ~CDacCalibrationFunctionNet ( )` [virtual]

**11.11.2.4 !CDacCalibrationFunctionNet()** `!CDacCalibrationFunctionNet ( )`

### 11.11.3 Member Function Documentation

**11.11.3.1 BurnDacOffset()** `void BurnDacOffset ( uint16_t dacChannel )`

Writes the offset of a DAC channel to permanent memory.

#### Parameters

<code>dacChannel</code>	The DAC channel number.
-------------------------	-------------------------

**11.11.3.2 GetDacOffset()** `uint32_t GetDacOffset ( uint16_t dacChannel )`

Gets the offset of a DAC channel.

## Parameters

<i>dacChannel</i>	The DAC channel number.
-------------------	-------------------------

## Returns

The offset in digits.

**11.11.3.3 SetDacOffset()** `void SetDacOffset (`  
     `uint16_t dacChannel,`  
     `uint32_t offset )`

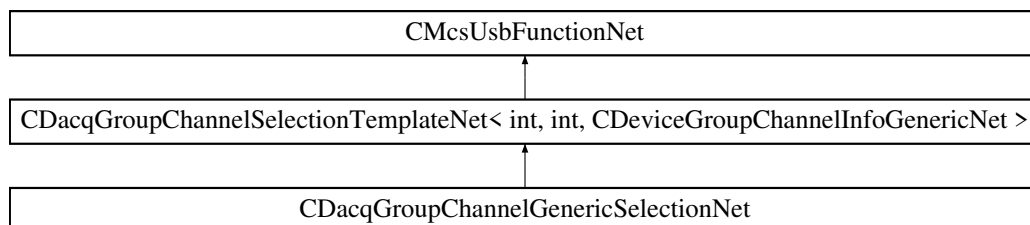
Sets the offset of a DAC channel.

## Parameters

<i>dacChannel</i>	The DAC channel number.
<i>offset</i>	The offset in digits.

## 11.12 CDacqGroupChannelGenericSelectionNet Class Reference

Inheritance diagram for CDacqGroupChannelGenericSelectionNet:



## Public Member Functions

- [CDacqGroupChannelGenericSelectionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)

## Additional Inherited Members

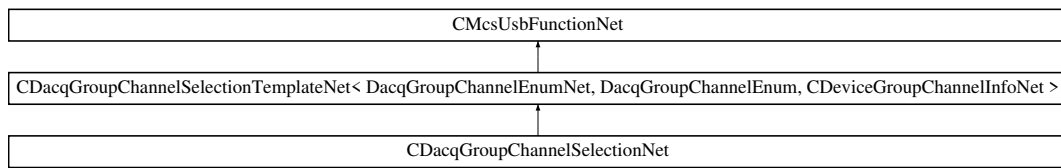
## 11.12.1 Constructor &amp; Destructor Documentation

**11.12.1.1 CDacqGroupChannelGenericSelectionNet()** [CDacqGroupChannelGenericSelectionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb )



## 11.13 CDacqGroupChannelSelectionNet Class Reference

Inheritance diagram for CDacqGroupChannelSelectionNet:



### Public Member Functions

- [CDacqGroupChannelSelectionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)

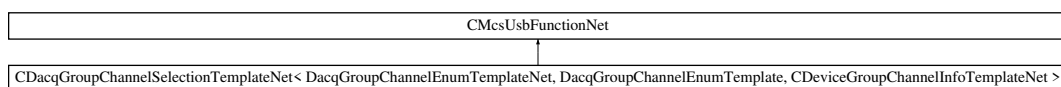
### Additional Inherited Members

#### 11.13.1 Constructor & Destructor Documentation

**11.13.1.1 CDacqGroupChannelSelectionNet()** [CDacqGroupChannelSelectionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb )

## 11.14 CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet > Class Template Reference

Inheritance diagram for CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet >:



### Public Member Functions

- [CDacqGroupChannelSelectionTemplateNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- [uint32\\_t GetNumberOfSupportedGroups](#) ()
- [uint32\\_t GetNumberOfSupportedGroups](#) ([uint32\\_t](#) virtualDevice)
- [DacqGroupChannelEnumTemplateNet GetGroupID](#) ([uint32\\_t](#) Index)
- [DacqGroupChannelEnumTemplateNet GetGroupID](#) ([uint32\\_t](#) Index, [uint32\\_t](#) virtualDevice)
- [uint32\\_t GetGroupNumberOfChannels](#) ([DacqGroupChannelEnumTemplateNet](#) GroupID)
- [uint32\\_t GetGroupNumberOfChannels](#) ([DacqGroupChannelEnumTemplateNet](#) GroupID, [uint32\\_t](#) virtualDevice)
- [DacqMeaGroupTypeEnumNet GetGroupType](#) ([DacqGroupChannelEnumTemplateNet](#) GroupID)
- [DacqMeaGroupTypeEnumNet GetGroupType](#) ([DacqGroupChannelEnumTemplateNet](#) GroupID, [uint32\\_t](#) virtualDevice)

## 11.14 CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet > Class Template Reference53

- void [EnableChannelsInGroup](#) (DacqGroupChannelEnumTemplateNet GroupID, List< bool >^ Enabled↔ ChannelsBitMap)
- void [EnableChannelsInGroup](#) (DacqGroupChannelEnumTemplateNet GroupID, List< bool >^ Enabled↔ ChannelsBitMap, uint32\_t virtualDevice)
- List< bool >^ [GetEnabledChannelsInGroup](#) (DacqGroupChannelEnumTemplateNet GroupID)
- List< bool >^ [GetEnabledChannelsInGroup](#) (DacqGroupChannelEnumTemplateNet GroupID, uint32\_t virtualDevice)
- SampleSizeNet [GetGroupSampleSize](#) (DacqGroupChannelEnumTemplateNet GroupID)
- SampleSizeNet [GetGroupSampleSize](#) (DacqGroupChannelEnumTemplateNet GroupID, uint32\_t virtualDevice)
- List< CDeviceGroupChannelInfoTemplateNet^>^ [GetDeviceGroupChannelInfos](#) ()
- List< CDeviceGroupChannelInfoTemplateNet^>^ [GetDeviceGroupChannelInfos](#) (uint32\_t virtualDevice)

### Additional Inherited Members

#### 11.14.1 Constructor & Destructor Documentation

**11.14.1.1 CDacqGroupChannelSelectionTemplateNet()** [CDacqGroupChannelSelectionTemplateNet](#) ( [CMcsUsbNet](#)^ *mcsusb* )

#### 11.14.2 Member Function Documentation

**11.14.2.1 EnableChannelsInGroup() [1/2]** void [EnableChannelsInGroup](#) ( DacqGroupChannelEnumTemplateNet *GroupID*, List< bool >^ *EnabledChannelsBitMap* )

**11.14.2.2 EnableChannelsInGroup() [2/2]** void [EnableChannelsInGroup](#) ( DacqGroupChannelEnumTemplateNet *GroupID*, List< bool >^ *EnabledChannelsBitMap*, uint32\_t *virtualDevice* )

**11.14.2.3 GetDeviceGroupChannelInfos() [1/2]** List<[CDeviceGroupChannelInfoTemplateNet](#)^>^ [GetDeviceGroupChannelInfos](#) ( )

**11.14.2.4 GetDeviceGroupChannelInfos() [2/2]** List<[CDeviceGroupChannelInfoTemplateNet](#)^>^ [GetDeviceGroupChannelInfos](#) ( uint32\_t *virtualDevice* )

**11.14.2.5 GetEnabledChannelsInGroup()** [1/2] List<bool> ^ GetEnabledChannelsInGroup (   
DacqGroupChannelEnumTemplateNet *GroupID* )

**11.14.2.6 GetEnabledChannelsInGroup()** [2/2] List<bool> ^ GetEnabledChannelsInGroup (   
DacqGroupChannelEnumTemplateNet *GroupID*,  
uint32\_t *virtualDevice* )

**11.14.2.7 GetGroupID()** [1/2] DacqGroupChannelEnumTemplateNet GetGroupID (   
uint32\_t *Index* )

**11.14.2.8 GetGroupID()** [2/2] DacqGroupChannelEnumTemplateNet GetGroupID (   
uint32\_t *Index*,  
uint32\_t *virtualDevice* )

**11.14.2.9 GetGroupNumberOfChannels()** [1/2] uint32\_t GetGroupNumberOfChannels (   
DacqGroupChannelEnumTemplateNet *GroupID* )

**11.14.2.10 GetGroupNumberOfChannels()** [2/2] uint32\_t GetGroupNumberOfChannels (   
DacqGroupChannelEnumTemplateNet *GroupID*,  
uint32\_t *virtualDevice* )

**11.14.2.11 GetGroupSampleSize()** [1/2] SampleSizeNet GetGroupSampleSize (   
DacqGroupChannelEnumTemplateNet *GroupID* )

**11.14.2.12 GetGroupSampleSize()** [2/2] SampleSizeNet GetGroupSampleSize (   
DacqGroupChannelEnumTemplateNet *GroupID*,  
uint32\_t *virtualDevice* )

**11.14.2.13 GetGroupType()** [1/2] DacqMeaGroupTypeEnumNet GetGroupType (   
DacqGroupChannelEnumTemplateNet *GroupID* )

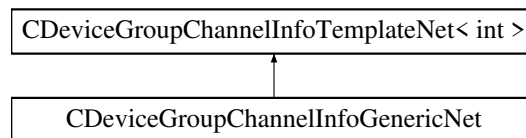
**11.14.2.14 GetGroupType()** [2/2] `DacqMeaGroupTypeEnumNet GetGroupType (`  
`DacqGroupChannelEnumTemplateNet GroupID,`  
`uint32_t virtualDevice )`

**11.14.2.15 GetNumberOfSupportedGroups()** [1/2] `uint32_t GetNumberOfSupportedGroups ( )`

**11.14.2.16 GetNumberOfSupportedGroups()** [2/2] `uint32_t GetNumberOfSupportedGroups (`  
`uint32_t virtualDevice )`

## 11.15 CDeviceGroupChannelInfoGenericNet Class Reference

Inheritance diagram for CDeviceGroupChannelInfoGenericNet:



### Public Member Functions

- [CDeviceGroupChannelInfoGenericNet](#) (int id, int channels, DacqMeaGroupTypeEnumNet type)

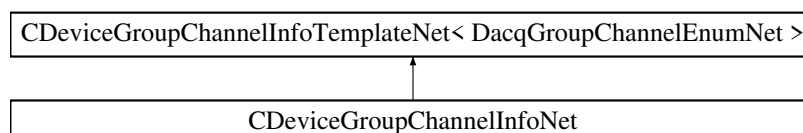
### Additional Inherited Members

#### 11.15.1 Constructor & Destructor Documentation

**11.15.1.1 CDeviceGroupChannelInfoGenericNet()** `CDeviceGroupChannelInfoGenericNet (`  
`int id,`  
`int channels,`  
`DacqMeaGroupTypeEnumNet type )`

## 11.16 CDeviceGroupChannelInfoNet Class Reference

Inheritance diagram for CDeviceGroupChannelInfoNet:



**Public Member Functions**

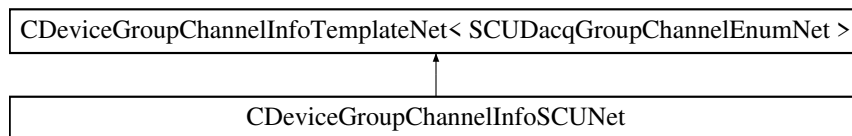
- [CDeviceGroupChannelInfoNet](#) (DacqGroupChannelEnumNet id, int channels, DacqMeaGroupTypeEnumNet type)

**Additional Inherited Members****11.16.1 Constructor & Destructor Documentation**

- 11.16.1.1 CDeviceGroupChannelInfoNet()** [CDeviceGroupChannelInfoNet](#) (
- ```
DacqGroupChannelEnumNet id,
int channels,
DacqMeaGroupTypeEnumNet type )
```

**11.17 CDeviceGroupChannelInfoSCUNet Class Reference**

Inheritance diagram for CDeviceGroupChannelInfoSCUNet:

**Public Member Functions**

- [CDeviceGroupChannelInfoSCUNet](#) (SCUDacqGroupChannelEnumNet id, int channels, DacqMeaGroupTypeEnumNet type)

**Additional Inherited Members****11.17.1 Constructor & Destructor Documentation**

- 11.17.1.1 CDeviceGroupChannelInfoSCUNet()** [CDeviceGroupChannelInfoSCUNet](#) (
- ```
SCUDacqGroupChannelEnumNet id,
int channels,
DacqMeaGroupTypeEnumNet type )
```

## 11.18 CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumTemplateNet > Class Template Reference

### Public Member Functions

- [CDeviceGroupChannelInfoTemplateNet](#) (DacqGroupChannelEnumTemplateNet id, int channels, DacqMeaGroupTypeEnumNet type)

### Public Attributes

- DacqGroupChannelEnumTemplateNet [GroupID](#)
- int [NumberOfChannels](#)
- DacqMeaGroupTypeEnumNet [GroupType](#)

### 11.18.1 Constructor & Destructor Documentation

**11.18.1.1 CDeviceGroupChannelInfoTemplateNet()** [CDeviceGroupChannelInfoTemplateNet](#) (   
DacqGroupChannelEnumTemplateNet *id*,   
int *channels*,   
DacqMeaGroupTypeEnumNet *type* )

### 11.18.2 Member Data Documentation

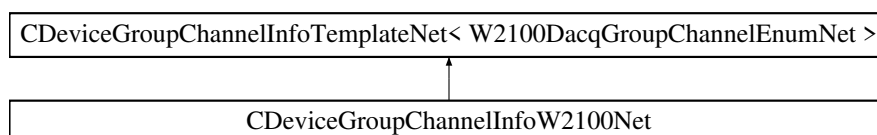
**11.18.2.1 GroupID** DacqGroupChannelEnumTemplateNet GroupID

**11.18.2.2 GroupType** DacqMeaGroupTypeEnumNet GroupType

**11.18.2.3 NumberOfChannels** int NumberOfChannels

## 11.19 CDeviceGroupChannelInfoW2100Net Class Reference

Inheritance diagram for CDeviceGroupChannelInfoW2100Net:



**Public Member Functions**

- [CDeviceGroupChannelInfoW2100Net](#) (W2100DacqGroupChannelEnumNet id, int channels, DacqMeaGroupTypeEnumNet type)

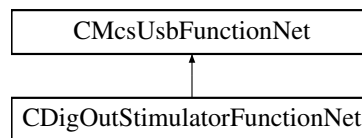
**Additional Inherited Members****11.19.1 Constructor & Destructor Documentation**

**11.19.1.1 CDeviceGroupChannelInfoW2100Net()** [CDeviceGroupChannelInfoW2100Net](#) ( W2100DacqGroupChannelEnumNet id, int channels, DacqMeaGroupTypeEnumNet type )

**11.20 CDigOutStimulatorFunctionNet Class Reference**

[CDigOutStimulatorFunctionNet](#) is the class of the DigOut stimulator function class.

Inheritance diagram for [CDigOutStimulatorFunctionNet](#):

**Public Member Functions**

- [CDigOutStimulatorFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> pDigOutStimulatorFunctionPointerContainer)
- *Initializes a new instance of the [CDigOutStimulatorFunctionNet](#) class.*
- [CDigOutStimulatorFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- virtual [~CDigOutStimulatorFunctionNet](#) ()
- [!CDigOutStimulatorFunctionNet](#) ()
- void [ClearChannel](#) (int32\_t NrChannel)
- *clear stimulation pattern*
- [CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData](#)<sup>^</sup> [PrepareChannelData](#) (array< int32\_t ><sup>^</sup> Amplitude, array< uint64\_t ><sup>^</sup> Duration)
- void [SendChannelData](#) (int32\_t NrChannel, [CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData](#)<sup>^</sup> device\_data\_and\_unrolled)
- int32\_t [GetNumberOfChannels](#) ()
- void [SetGlobalRepeat](#) (int32\_t NrChannel, bool value)
- *set repeat whole stimulation pattern*
- bool [GetGlobalRepeat](#) (int32\_t NrChannel)
- *get repeat whole stimulation pattern*
- void [SetStartTriggerSlope](#) (int32\_t NrChannel, DigitalStimulatorTriggerSlopeEnumNet Condition)
- *sets start condition of digital out stimulator*
- DigitalStimulatorTriggerSlopeEnumNet [GetStartTriggerSlope](#) (int32\_t NrChannel)
- *queries start condition of digital out stimulator*
- void [SetStopTriggerSlope](#) (int32\_t NrChannel, DigitalStimulatorTriggerSlopeEnumNet Condition)
- *sets stop condition of digital out stimulator*
- DigitalStimulatorTriggerSlopeEnumNet [GetStopTriggerSlope](#) (int32\_t NrChannel)
- *queries stop condition of digital out stimulator*

## Additional Inherited Members

### 11.20.1 Detailed Description

[CDigOutStimulatorFunctionNet](#) is the class of the DigOut stimulator function class.

### 11.20.2 Constructor & Destructor Documentation

**11.20.2.1 CDigOutStimulatorFunctionNet()** [1/2] [CDigOutStimulatorFunctionNet](#) (  
    [CMcsUsbNet](#)<sup>^</sup> *mcsusb*,  
    [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> *pDigOutStimulatorFunctionPointerContainer* )

Initializes a new instance of the [CDigOutStimulatorFunctionNet](#) class.

**11.20.2.2 CDigOutStimulatorFunctionNet()** [2/2] [CDigOutStimulatorFunctionNet](#) (  
    [CMcsUsbNet](#)<sup>^</sup> *mcsusb* )

**11.20.2.3 ~CDigOutStimulatorFunctionNet()** virtual [~CDigOutStimulatorFunctionNet](#) ( ) [virtual]

**11.20.2.4 "!CDigOutStimulatorFunctionNet()** [!CDigOutStimulatorFunctionNet](#) ( )

### 11.20.3 Member Function Documentation

**11.20.3.1 ClearChannel()** void [ClearChannel](#) (  
    int32\_t *NrChannel* )

clear stimulation pattern

#### Parameters

<i>NrChannel</i>	
------------------	--



**11.20.3.2 GetGlobalRepeat()** `bool GetGlobalRepeat (   
int32_t NrChannel )`

get repeat whole stimulation pattern

#### Parameters

<i>NrChannel</i>	channel number
------------------	----------------

#### Returns

current value

**11.20.3.3 GetNumberOfChannels()** `int32_t GetNumberOfChannels ( )`

#### Returns

**11.20.3.4 GetStartTriggerSlope()** `DigitalStimulatorTriggerSlopeEnumNet GetStartTriggerSlope (   
int32_t NrChannel )`

queries start condition of digital out stimulator

#### Parameters

<i>NrChannel</i>	channel number
------------------	----------------

#### Returns

start condition (rising or falling edge)

**11.20.3.5 GetStopTriggerSlope()** `DigitalStimulatorTriggerSlopeEnumNet GetStopTriggerSlope (   
int32_t NrChannel )`

queries stop condition of digital out stimulator

#### Parameters

<i>NrChannel</i>	channel number
------------------	----------------

**Returns**

stop condition (rising or falling edge)

**11.20.3.6 PrepareChannelData()** `CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData ^ PrepareChannelData (`

```
    array< int32_t >^ Amplitude,
    array< uint64_t >^ Duration )
```

**Parameters**

<i>Amplitude</i>	
<i>Duration</i>	

**Returns**
**11.20.3.7 SendChannelData()** `void SendChannelData (`  

```
    int32_t NrChannel,
    CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData^ device_data_and_unrolled
)

```

**Parameters**

<i>NrChannel</i>	
<i>device_data_and_unrolled</i>	

**11.20.3.8 SetGlobalRepeat()** `void SetGlobalRepeat (`  

```
    int32_t NrChannel,
    bool value )
```

set repeat whole stimulation pattern

**Parameters**

<i>NrChannel</i>	channel number
<i>value</i>	new value

**11.20.3.9 SetStartTriggerSlope()** `void SetStartTriggerSlope (`

```
int32_t NrChannel,
DigitalStimulatorTriggerSlopeEnumNet Condition )
```

sets start condition of digital out stimulator

#### Parameters

<i>NrChannel</i>	channel number
<i>Condition</i>	start condition (rising or falling edge)

**11.20.3.10 SetStopTriggerSlope()** void SetStopTriggerSlope (   
int32\_t NrChannel,   
DigitalStimulatorTriggerSlopeEnumNet Condition )

sets stop condition of digital out stimulator

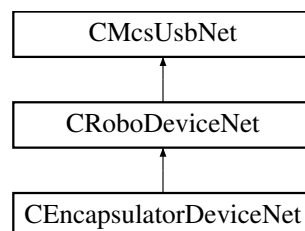
#### Parameters

<i>NrChannel</i>	channel number
<i>Condition</i>	stop condition (rising or falling edge)

## 11.21 CEncapsulatorDeviceNet Class Reference

[CEncapsulatorDeviceNet](#) is the to control the MCS HiClamp device

Inheritance diagram for CEncapsulatorDeviceNet:



#### Public Member Functions

- [CEncapsulatorDeviceNet](#) (void)
- [CRoboFluidDeviceNet](#) ^ [GetRoboFluidDevice](#) ()

#### Additional Inherited Members

##### 11.21.1 Detailed Description

[CEncapsulatorDeviceNet](#) is the to control the MCS HiClamp device

### 11.21.2 Constructor & Destructor Documentation

**11.21.2.1 CEncapsulatorDeviceNet()** `CEncapsulatorDeviceNet ( void )`

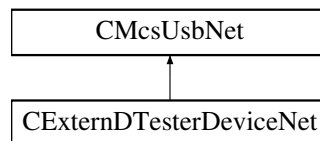
### 11.21.3 Member Function Documentation

**11.21.3.1 GetRoboFluidDevice()** `CRoboFluidDeviceNet ^ GetRoboFluidDevice ( )`

## 11.22 CExternDTesterDeviceNet Class Reference

`CExternDTesterDeviceNet` is the class to access the ExternD Tester (Handheld Device Tester D)

Inheritance diagram for `CExternDTesterDeviceNet`:



### Public Member Functions

- `CExternDTesterDeviceNet ( )`  
*Initializes a new instance of the `CExternDTesterDeviceNet` class.*
- virtual `~CExternDTesterDeviceNet ( )`
- `!CExternDTesterDeviceNet ( )`
- `array< uint8_t > ^ Read (int configString_Length)`  
*Reads the config string from the device.*
- `String ^ Read2 ( )`  
*Reads the config string from the device.*
- `void Write (array< uint8_t > ^ configString)`  
*Reads the config string from the device.*
- `void Write2 (String ^ configString)`  
*Reads the config string from the device.*

### Additional Inherited Members

#### 11.22.1 Detailed Description

`CExternDTesterDeviceNet` is the class to access the ExternD Tester (Handheld Device Tester D)

### 11.22.2 Constructor & Destructor Documentation

#### 11.22.2.1 CExternDTesterDeviceNet() [CExternDTesterDeviceNet](#) ( )

Initializes a new instance of the [CExternDTesterDeviceNet](#) class.

#### 11.22.2.2 ~CExternDTesterDeviceNet() [virtual](#) [~CExternDTesterDeviceNet](#) ( ) [\[virtual\]](#)

#### 11.22.2.3 "!CExternDTesterDeviceNet() [!CExternDTesterDeviceNet](#) ( )

### 11.22.3 Member Function Documentation

#### 11.22.3.1 Read() [array<uint8\\_t>](#) ^ [Read](#) ( [int](#) *configString\_Length* )

Reads the config string from the device.

##### Parameters

<i>configString_Length</i>	The maximal length of configString.
----------------------------	-------------------------------------

##### Returns

The config string.

#### 11.22.3.2 Read2() [String](#) ^ [Read2](#) ( )

Reads the config string from the device.

##### Returns

The config string.

#### 11.22.3.3 Write() [void](#) [Write](#) ( [array< uint8\\_t >](#) ^ *configString* )

Reads the config string from the device.

## Parameters

<i>configString</i>	The config string.
---------------------	--------------------

**11.22.3.4 Write2()** void Write2 (   
String^ *configString* )

Reads the config string from the device.

## Parameters

<i>configString</i>	The config string.
---------------------	--------------------

## 11.23 CFilterCoefficientsNet Class Reference

### Public Member Functions

- void [SetAFormat](#) (int vk, int nk, int pos, FilterCalculationDirectionEnumNet dir)
- void [SetBFormat](#) (int vk, int nk, int pos, FilterCalculationDirectionEnumNet dir)
- [CFilterCoefficientsNet](#) ()
- [CFilterCoefficientsNet](#) (double b0, double b1, double b2, double a1, double a2)
- [CFilterCoefficientsNet](#) (double b0, double b1, double a1)
- [CFilterCoefficientsNet](#) (array< double >^ b, array< double >^ a)
- [CFilterCoefficientsNet](#) (uint64\_t b0, uint64\_t b1, uint64\_t b2, uint64\_t a1, uint64\_t a2)
- [~CFilterCoefficientsNet](#) ()
- virtual bool [Equals](#) ([CFilterCoefficientsNet](#)^ coefficients)

### Properties

- array< double >^ [A](#) [get]
- array< double >^ [B](#) [get]
- double [B0](#) [get]
- double [B1](#) [get]
- double [B2](#) [get]
- double [A1](#) [get]
- double [A2](#) [get]
- uint64\_t [UIntB0](#) [get]
- uint64\_t [UIntB1](#) [get]
- uint64\_t [UIntB2](#) [get]
- uint64\_t [UIntA1](#) [get]
- uint64\_t [UIntA2](#) [get]

### 11.23.1 Constructor & Destructor Documentation

**11.23.1.1 CFilterCoefficientsNet()** [1/5] [CFilterCoefficientsNet](#) ( )

**11.23.1.2 CFilterCoefficientsNet()** [2/5] [CFilterCoefficientsNet](#) (   
double *b0*,  
double *b1*,  
double *b2*,  
double *a1*,  
double *a2* )

**11.23.1.3 CFilterCoefficientsNet()** [3/5] [CFilterCoefficientsNet](#) (   
double *b0*,  
double *b1*,  
double *a1* )

**11.23.1.4 CFilterCoefficientsNet()** [4/5] [CFilterCoefficientsNet](#) (   
array< double >^ *b*,  
array< double >^ *a* )

**11.23.1.5 CFilterCoefficientsNet()** [5/5] [CFilterCoefficientsNet](#) (   
uint64\_t *b0*,  
uint64\_t *b1*,  
uint64\_t *b2*,  
uint64\_t *a1*,  
uint64\_t *a2* )

**11.23.1.6 ~CFilterCoefficientsNet()** [~CFilterCoefficientsNet](#) ( )

## 11.23.2 Member Function Documentation

**11.23.2.1 Equals()** virtual bool Equals (   
[CFilterCoefficientsNet](#)^ *coefficients* ) [virtual]

**11.23.2.2 SetAFormat()** void SetAFormat (  
    int *vk*,  
    int *nk*,  
    int *pos*,  
    FilterCalculationDirectionEnumNet *dir* )

**11.23.2.3 SetBFormat()** void SetBFormat (  
    int *vk*,  
    int *nk*,  
    int *pos*,  
    FilterCalculationDirectionEnumNet *dir* )

### 11.23.3 Property Documentation

**11.23.3.1 A** array< double>^ A [get]

**11.23.3.2 A1** double A1 [get]

**11.23.3.3 A2** double A2 [get]

**11.23.3.4 B** array< double>^ B [get]

**11.23.3.5 B0** double B0 [get]

**11.23.3.6 B1** double B1 [get]

**11.23.3.7 B2** double B2 [get]



**11.23.3.8 UintA1** `uint64_t UintA1 [get]`

**11.23.3.9 UintA2** `uint64_t UintA2 [get]`

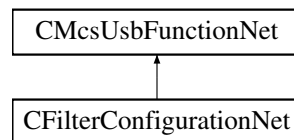
**11.23.3.10 UintB0** `uint64_t UintB0 [get]`

**11.23.3.11 UintB1** `uint64_t UintB1 [get]`

**11.23.3.12 UintB2** `uint64_t UintB2 [get]`

## 11.24 CFilterConfigurationNet Class Reference

Inheritance diagram for CFilterConfigurationNet:



### Public Member Functions

- [CFilterConfigurationNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- void [SetFilterParameter](#) (DacqGroupChannelEnumNet GroupID, uint32\_t FilterNumber, [CFilterCoefficientsNet](#)<sup>^</sup> Coefficients, [CFilterPropertyNet](#)<sup>^</sup> FilterProp)
- void [SetFilterParameter](#) (DacqGroupChannelEnumNet GroupID, uint32\_t FilterNumber, [CFilterCoefficientsNet](#)<sup>^</sup> CoefficientsSet1, [CFilterCoefficientsNet](#)<sup>^</sup> CoefficientsSet2, [CFilterPropertyNet](#)<sup>^</sup> FilterProp)
- void [SetFilterParameterPermanent](#) (DacqGroupChannelEnumNet GroupID, uint32\_t FilterNumber)
- void [EraseFilterParameterPermanent](#) (DacqGroupChannelEnumNet GroupID, uint32\_t FilterNumber)
- void [SetHighpassFilterEnable](#) (bool enable)
- bool [GetHighpassFilterEnable](#) ()
- void [ResetHighpassFilter](#) ()

### Additional Inherited Members

#### 11.24.1 Constructor & Destructor Documentation

**11.24.1.1 CFilterConfigurationNet()** `CFilterConfigurationNet ( CMcsUsbNet^ mcsusb )`

## 11.24.2 Member Function Documentation

**11.24.2.1 EraseFilterParameterPermanent()** `void EraseFilterParameterPermanent ( DacqGroupChannelEnumNet GroupID, uint32_t FilterNumber )`

**11.24.2.2 GetHighpassFilterEnable()** `bool GetHighpassFilterEnable ( )`

**11.24.2.3 ResetHighpassFilter()** `void ResetHighpassFilter ( )`

**11.24.2.4 SetFilterParameter() [1/2]** `void SetFilterParameter ( DacqGroupChannelEnumNet GroupID, uint32_t FilterNumber, CFilterCoefficientsNet^ Coefficients, CFilterPropertyNet^ FilterProp )`

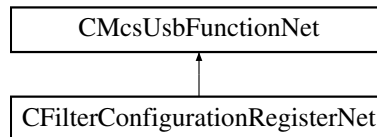
**11.24.2.5 SetFilterParameter() [2/2]** `void SetFilterParameter ( DacqGroupChannelEnumNet GroupID, uint32_t FilterNumber, CFilterCoefficientsNet^ CoefficientsSet1, CFilterCoefficientsNet^ CoefficientsSet2, CFilterPropertyNet^ FilterProp )`

**11.24.2.6 SetFilterParameterPermanent()** `void SetFilterParameterPermanent ( DacqGroupChannelEnumNet GroupID, uint32_t FilterNumber )`

**11.24.2.7 SetHighpassFilterEnable()** `void SetHighpassFilterEnable ( bool enable )`

## 11.25 CFilterConfigurationRegisterNet Class Reference

Inheritance diagram for CFilterConfigurationRegisterNet:



### Public Member Functions

- [CFilterConfigurationRegisterNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- void [SetFilterParameter](#) (uint32\_t FilterCoefRegBase, [CFilterCoefficientsNet](#)<sup>^</sup> Coefficients, uint32\_t FilterInfoRegBase, [CFilterPropertyNet](#)<sup>^</sup> FilterProp)
- void [SetFilterParameter](#) (uint32\_t FilterCoefSet1RegBase, [CFilterCoefficientsNet](#)<sup>^</sup> CoefficientsSet1, uint32\_t FilterCoefSet2RegBase, [CFilterCoefficientsNet](#)<sup>^</sup> CoefficientsSet2, uint32\_t FilterInfoRegBase, [CFilterPropertyNet](#)<sup>^</sup> FilterProp)
- void [SetFilterParameterPermanent](#) (uint32\_t FilterCoefRegBase, uint32\_t FilterCoefDmaReg, uint32\_t FilterInfoRegBase, uint32\_t FilterInfoDmaReg, uint32\_t EEPROMBase, uint32\_t EEPROMSize)
- void [SetFilterParameterPermanent](#) (uint32\_t FilterCoefSet1RegBase, uint32\_t FilterCoefSet1DmaReg, uint32\_t FilterCoefSet2RegBase, uint32\_t FilterCoefSet2DmaReg, uint32\_t FilterInfoRegBase, uint32\_t FilterInfoDmaReg, uint32\_t EEPROMBase, uint32\_t EEPROMSize)
- void [EraseFilterParameterPermanent](#) (uint32\_t FilterCoefDmaReg, uint32\_t FilterInfoDmaReg, uint32\_t EEPROMBase, uint32\_t EEPROMSize)
- void [EraseFilterParameterPermanent](#) (uint32\_t FilterCoefSet1DmaReg, uint32\_t FilterCoefSet2DmaReg, uint32\_t FilterInfoDmaReg, uint32\_t EEPROMBase, uint32\_t EEPROMSize)

### Additional Inherited Members

#### 11.25.1 Constructor & Destructor Documentation

**11.25.1.1 CFilterConfigurationRegisterNet()** [CFilterConfigurationRegisterNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb )

#### 11.25.2 Member Function Documentation

**11.25.2.1 EraseFilterParameterPermanent()** [1/2] void EraseFilterParameterPermanent (uint32\_t FilterCoefDmaReg, uint32\_t FilterInfoDmaReg, uint32\_t EEPROMBase, uint32\_t EEPROMSize )

**11.25.2.2 EraseFilterParameterPermanent() [2/2]** void EraseFilterParameterPermanent (   
 uint32\_t FilterCoefSet1DmaReg,   
 uint32\_t FilterCoefSet2DmaReg,   
 uint32\_t FilterInfoDmaReg,   
 uint32\_t EEPROMBase,   
 uint32\_t EEPROMSize )

**11.25.2.3 SetFilterParameter() [1/2]** void SetFilterParameter (   
 uint32\_t FilterCoefRegBase,   
 CFilterCoefficientsNet^ Coefficients,   
 uint32\_t FilterInfoRegBase,   
 CFilterPropertyNet^ FilterProp )

**11.25.2.4 SetFilterParameter() [2/2]** void SetFilterParameter (   
 uint32\_t FilterCoefSet1RegBase,   
 CFilterCoefficientsNet^ CoefficientsSet1,   
 uint32\_t FilterCoefSet2RegBase,   
 CFilterCoefficientsNet^ CoefficientsSet2,   
 uint32\_t FilterInfoRegBase,   
 CFilterPropertyNet^ FilterProp )

**11.25.2.5 SetFilterParameterPermanent() [1/2]** void SetFilterParameterPermanent (   
 uint32\_t FilterCoefRegBase,   
 uint32\_t FilterCoefDmaReg,   
 uint32\_t FilterInfoRegBase,   
 uint32\_t FilterInfoDmaReg,   
 uint32\_t EEPROMBase,   
 uint32\_t EEPROMSize )

**11.25.2.6 SetFilterParameterPermanent() [2/2]** void SetFilterParameterPermanent (   
 uint32\_t FilterCoefSet1RegBase,   
 uint32\_t FilterCoefSet1DmaReg,   
 uint32\_t FilterCoefSet2RegBase,   
 uint32\_t FilterCoefSet2DmaReg,   
 uint32\_t FilterInfoRegBase,   
 uint32\_t FilterInfoDmaReg,   
 uint32\_t EEPROMBase,   
 uint32\_t EEPROMSize )

## 11.26 CFilterPropertyNet Class Reference

### Public Member Functions

- CFilterPropertyNet (uint32\_t CornerFrequenzymHz, uint32\_t Order, FilterBandEnumNet FilterBand, FilterFamilyEnumNet FilterFamily, FilterTypeEnumNet FilterType, bool Active)
- ~CFilterPropertyNet ()
- virtual System::String ^ ToString () override

## Properties

- uint32\_t [CornerFrequencyMHz](#) [get]
- double [CornerFrequency](#) [get]
- uint32\_t [Order](#) [get]
- FilterBandEnumNet [FilterBand](#) [get]
- FilterFamilyEnumNet [FilterFamily](#) [get]
- FilterTypeEnumNet [FilterType](#) [get]
- bool [FilterActive](#) [get]

## 11.26.1 Constructor & Destructor Documentation

**11.26.1.1 CFilterPropertyNet()** [CFilterPropertyNet](#) (  
    uint32\_t *CornerFrequenzymHz*,  
    uint32\_t *Order*,  
    FilterBandEnumNet *FilterBand*,  
    FilterFamilyEnumNet *FilterFamily*,  
    FilterTypeEnumNet *FilterType*,  
    bool *Active* )

**11.26.1.2 ~CFilterPropertyNet()** [~CFilterPropertyNet](#) ( )

## 11.26.2 Member Function Documentation

**11.26.2.1 ToString()** virtual System::String ^ ToString ( ) [override], [virtual]

## 11.26.3 Property Documentation

**11.26.3.1 CornerFrequency** double [CornerFrequency](#) [get]

**11.26.3.2 CornerFrequencyMHz** uint32\_t [CornerFrequencyMHz](#) [get]

**11.26.3.3 FilterActive** `bool FilterActive [get]`

**11.26.3.4 FilterBand** `FilterBandEnumNet FilterBand [get]`

**11.26.3.5 FilterFamily** `FilterFamilyEnumNet FilterFamily [get]`

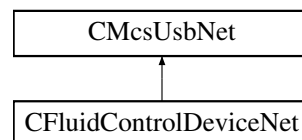
**11.26.3.6 FilterType** `FilterTypeEnumNet FilterType [get]`

**11.26.3.7 Order** `uint32_t Order [get]`

## 11.27 CFluidControlDeviceNet Class Reference

[CFluidControlDeviceNet](#) is the class to control MCS FluidControl (FCB and FCX) device.

Inheritance diagram for CFluidControlDeviceNet:



### Public Member Functions

- [CFluidControlDeviceNet](#) ()  
*Initialize a new instance of the [CFluidControlDeviceNet](#) class.*
- [~CFluidControlDeviceNet](#) ()  
*Default destructor.*
- void [SetValve](#) (unsigned int value)  
*Open or Close valves.*
- void [SetSingleValve](#) (unsigned short valve, unsigned short onoff)  
*Opens or Closes a valve.*
- void [SetDigout](#) (unsigned int value)  
*Define the pattern on the Digital Output.*
- void [SetPWM](#) (unsigned int channel, unsigned int value)  
*Sets the duty cycle of the PWM output.*
- void [CalibrateThermocouple](#) (unsigned int channel)  
*Calibrates the ADC which is used for the Thermocouple. For the calibration, Short circuit the Thermocouple and use this function to correct a possible offset of the ADC which measures the thermocouple.*
- void [SetThermocoupleNanovoltPerKelvin](#) (unsigned int channel, unsigned int value)

- Sets the proportional constant for the Thermocouple.*

  - unsigned int [GetValue](#) ()

*Gets the state of the valves.*

  - unsigned short [GetSingleValve](#) (unsigned short valve)

*Gets the state of a valve.*

  - unsigned int [GetDigout](#) ()

*Gets the state of the digital output.*

  - unsigned int [GetPWM](#) (unsigned int channel)

*Gets the state of the PWM output.*

  - unsigned int [GetAdc](#) (unsigned int channel)

*Reads an ADC Value.*

  - unsigned int [GetDigin](#) ()

*Reads the digital input.*

  - int [GetThermocoupleTemperature](#) (unsigned int channel)

*Reads the temperature from Thermocouple. The functions gives the temperature difference between both Thermocouple junctions. To get the absolute temperature, add the reference temperature.*

  - int [GetReferenceTemperature](#) (unsigned int channel)

*Reads the reference temperature for the Thermocouple.*

  - unsigned int [GetThermocoupleCalibration](#) (unsigned int channel)

*Gets the calibration constant for the Thermocouple ADC.*

  - unsigned int [GetThermocoupleNanovoltPerKelvin](#) (unsigned int channel)

*Reads the proportional constant for the Thermocouple.*

## Properties

- [CMcsBus\\_VoltageModeNet](#)<sup>^</sup> [McsBus\\_VoltageMode](#) [get]

## Additional Inherited Members

### 11.27.1 Detailed Description

[CFluidControlDeviceNet](#) is the class to control MCS FluidControl (FCB and FCX) device.

### 11.27.2 Constructor & Destructor Documentation

#### 11.27.2.1 CFluidControlDeviceNet() [CFluidControlDeviceNet](#) ( )

Initialize a new instance of the [CFluidControlDeviceNet](#) class.

#### 11.27.2.2 ~CFluidControlDeviceNet() [~CFluidControlDeviceNet](#) ( )

Default destructor.

### 11.27.3 Member Function Documentation

**11.27.3.1 CalibrateThermocouple()** `void CalibrateThermocouple (`  
`unsigned int channel )`

Calibrates the ADC which is used for the Thermocouple. For the calibration, Short circuit the Thermocouple and use this function to correct a possible offset of the ADC which measures the thermocouple.



## Parameters

<i>channel</i>	Thermocouple channel number.
----------------	------------------------------

**11.27.3.2 GetAdc()** `unsigned int GetAdc ( unsigned int channel )`

Reads an ADC Value.

## Parameters

<i>channel</i>	The ADC channel number to query.
----------------	----------------------------------

## Returns

The current ADC value.

**11.27.3.3 GetDigin()** `unsigned int GetDigin ( )`

Reads the digital input.

## Returns

The bit pattern of the state of the digital inputs.

**11.27.3.4 GetDigout()** `unsigned int GetDigout ( )`

Gets the state of the digital output.

## Returns

The current state of the digital outputs as a bit pattern.

**11.27.3.5 GetPWM()** `unsigned int GetPWM ( unsigned int channel )`

Gets the state of the PWM output.

## Returns

The current state of the PWM outputs duty cycle in permille.

**11.27.3.6 GetReferenceTemperature()** `int GetReferenceTemperature ( unsigned int channel )`

Reads the reference temperature for the Thermocouple.

## Parameters

<i>channel</i>	Thermocouple channel number.
----------------	------------------------------

## Returns

The temperature from the Thermocouple in 1/100 °C.

**11.27.3.7 GetSingleValve()** unsigned short GetSingleValve (  
unsigned short valve )

Gets the state of a valve.

## Parameters

<i>valve</i>	number of valve
--------------	-----------------

## Returns

state of the valve

**11.27.3.8 GetThermocoupleCalibration()** unsigned int GetThermocoupleCalibration (  
unsigned int channel )

Gets the calibration constant for the Thermocouple ADC.

## Parameters

<i>channel</i>	Thermocouple channel number.
----------------	------------------------------

## Returns

The calibration constant for the Thermocouple ADC.

**11.27.3.9 GetThermocoupleNanovoltPerKelvin()** unsigned int GetThermocoupleNanovoltPerKelvin (  
unsigned int channel )

Reads the proportional constant for the Thermocouple.

## Parameters

<i>channel</i>	Thermocouple channel number.
----------------	------------------------------

**Returns**

The proportional constant in Nanovolt per Kelvin.

**11.27.3.10 GetThermocoupleTemperature()** `int GetThermocoupleTemperature ( unsigned int channel )`

Reads the temperature from Thermocouple. The functions gives the temperature difference between both Thermocouple junctions. To get the absolute temperature, add the reference temperature.

**Parameters**

<i>channel</i>	Thermocouple channel number.
----------------	------------------------------

**Returns**

The temperature difference between both Thermocouple junctions in 1/100 °C.

**11.27.3.11 GetValve()** `unsigned int GetValve ( )`

Gets the state of the valves.

**Returns**

The current state of the valves as a bit pattern.

**11.27.3.12 SetDigout()** `void SetDigout ( unsigned int value )`

Define the pattern on the Digital Output.

**Parameters**

<i>value</i>	bit pattern on the digital output.
--------------	------------------------------------

**11.27.3.13 SetPWM()** `void SetPWM ( unsigned int channel, unsigned int value )`

Sets the duty cycle of the PWM output.

## Parameters

<i>channel</i>	PWM channel number.
<i>value</i>	duty cycle of the PWM output in permille.

**11.27.3.14 SetSingleValve()** `void SetSingleValve (`  
    `unsigned short valve,`  
    `unsigned short onoff )`

Opens or Closes a valve.

## Parameters

<i>valve</i>	number of valve to be changed.
--------------	--------------------------------

## Parameters

<i>onoff</i>	open or close the valve.
--------------	--------------------------

**11.27.3.15 SetThermocoupleNanovoltPerKelvin()** `void SetThermocoupleNanovoltPerKelvin (`  
    `unsigned int channel,`  
    `unsigned int value )`

Sets the proportinal constant for the Thermocouple.

## Parameters

<i>channel</i>	Thermocouple channel number.
<i>value</i>	proportinal constant for the Thermocouple in Nanovolt per Kelvin.

**11.27.3.16 SetValve()** `void SetValve (`  
    `unsigned int value )`

Open or Close valves.

## Parameters

<i>value</i>	bit pattern of valves which should be open.
--------------	---

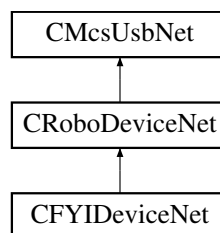
### 11.27.4 Property Documentation

**11.27.4.1 McsBus\_VoltageMode** [CMcsBus\\_VoltageModeNet](#)<sup>^</sup> McsBus\_VoltageMode [get]

## 11.28 CFYIDeviceNet Class Reference

[CFYIDeviceNet](#) is the class to control the MCS FYI device

Inheritance diagram for CFYIDeviceNet:



### Public Member Functions

- [CFYIDeviceNet](#) (void)

### Properties

- [CRobo\\_FYITemp\\_FunctionNet](#)<sup>^</sup> FYITemp [get]
- [CRobo\\_FYIProgram\\_FunctionNet](#)<sup>^</sup> FYIProgram [get]
- [CMcsBus\\_SensorNet](#)<sup>^</sup> Sensor [get]

### Additional Inherited Members

### 11.28.1 Detailed Description

[CFYIDeviceNet](#) is the class to control the MCS FYI device

### 11.28.2 Constructor & Destructor Documentation

**11.28.2.1 CFYIDeviceNet()** [CFYIDeviceNet](#) (  
void )

### 11.28.3 Property Documentation

**11.28.3.1 FYIProgram** `CRobo_FYIProgram_FunctionNet^ FYIProgram [get]`

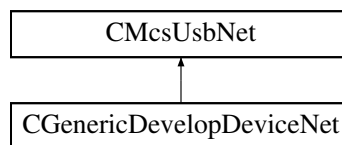
**11.28.3.2 FYITemp** `CRobo_FYITemp_FunctionNet^ FYITemp [get]`

**11.28.3.3 Sensor** `CMcsBus_SensorNet^ Sensor [get]`

## 11.29 CGenericDevelopDeviceNet Class Reference

`CGenericDevelopDeviceNet` is the class to use during development of a new device.

Inheritance diagram for `CGenericDevelopDeviceNet`:



### Public Member Functions

- `CGenericDevelopDeviceNet` (void)  
*Initialize a new instance of the `CGenericDevelopDeviceNet` class.*
- `~CGenericDevelopDeviceNet` (void)
- void `SetValue` (uint16\_t value, uint16\_t index)  
*Sets .*

#### Parameters

value	<i>The value of the request.</i>
-------	----------------------------------

#### Parameters

index	<i>The index of the request.</i>
-------	----------------------------------

- `template<typename C >`  
void `SetBuffer` (uint16\_t value, uint16\_t index, array< C >^ buffer)

- void [SetUByteBuffer](#) (uint16\_t value, uint16\_t index, array< unsigned char >^ buffer)

*Sends an array of type unsigned char to the device.*

*Parameters*

value	<i>The value of the request.</i>
-------	----------------------------------

*Parameters*

index	<i>The index of the request.</i>
-------	----------------------------------

*Parameters*

buffer	<i>The buffer to send.</i>
--------	----------------------------

- void [SetByteBuffer](#) (uint16\_t value, uint16\_t Index, array< char >^ buffer)

*Sends an array of type char to the device.*

*Parameters*

value	<i>The value of the request.</i>
-------	----------------------------------

*Parameters*

Index	<i>The index of the request.</i>
-------	----------------------------------

*Parameters*

buffer	<i>The buffer to send.</i>
--------	----------------------------

- void [SetUShortBuffer](#) (uint16\_t value, uint16\_t index, array< unsigned short >^ buffer)

*Sends an array of type unsigned short to the device.*

*Parameters*

value	<i>The value of the request.</i>
-------	----------------------------------

*Parameters*

index	<i>The index of the request.</i>
-------	----------------------------------

*Parameters*

buffer	<i>The buffer to send.</i>
--------	----------------------------

- void [SetShortBuffer](#) (uint16\_t value, uint16\_t index, array< short >^ buffer)  
*Sends an array of type short to the device.*

*Parameters*

value	<i>The value of the request.</i>
-------	----------------------------------

*Parameters*

index	<i>The index of the request.</i>
-------	----------------------------------

*Parameters*

buffer	<i>The buffer to send.</i>
--------	----------------------------

- void [SetUIntBuffer](#) (uint16\_t value, uint16\_t index, array< unsigned int >^ buffer)  
*Sends an array of unsigned int to the device.*

*Parameters*

value	<i>The value of the request.</i>
-------	----------------------------------

*Parameters*

index	<i>The index of the request.</i>
-------	----------------------------------

*Parameters*

buffer	<i>The buffer to send.</i>
--------	----------------------------

- void [SetIntBuffer](#) (uint16\_t value, uint16\_t index, array< int >^ buffer)  
*Sends an array of type int to the device.*



*Parameters*

value	<i>The value of the request.</i>
-------	----------------------------------

*Parameters*

index	<i>The index of the request.</i>
-------	----------------------------------

*Parameters*

buffer	<i>The buffer to send.</i>
--------	----------------------------

- `template<typename C >`  
`array< C > ^ GetBuffer (uint16_t value, uint16_t index, int size)`
- `array< unsigned char > ^ GetUByteBuffer (uint16_t value, uint16_t index, int size)`  
*Gets an array of type unsigned char from the device.*

*Parameters*

value	<i>The value of the request.</i>
-------	----------------------------------

*Parameters*

index	<i>The index of the request.</i>
-------	----------------------------------

*Parameters*

size	<i>The size of the array.</i>
------	-------------------------------

*Returns*

*The array of data from the device.*

- `array< char > ^ GetByteBuffer (uint16_t value, uint16_t index, int size)`  
*Gets an array of type char from the device.*

*Parameters*

value	<i>The value of the request.</i>
-------	----------------------------------

*Parameters*

index	<i>The index of the request.</i>
-------	----------------------------------

*Parameters*

size	<i>The size of the array.</i>
------	-------------------------------

*Returns*

*The array of data from the device.*

- array< unsigned short > ^ [GetUShortBuffer](#) (uint16\_t value, uint16\_t index, int size)  
*Gets an array of type unsigned short from the device.*

*Parameters*

value	<i>The value of the request.</i>
-------	----------------------------------

*Parameters*

index	<i>The index of the request.</i>
-------	----------------------------------

*Parameters*

size	<i>The size of the array.</i>
------	-------------------------------

*Returns*

*The array of data from the device.*

- array< short > ^ [GetShortBuffer](#) (uint16\_t value, uint16\_t index, int size)  
*Gets an array of type short from the device.*

*Parameters*

value	<i>The value of the request.</i>
-------	----------------------------------

*Parameters*

index	<i>The index of the request.</i>
-------	----------------------------------

*Parameters*

size	<i>The size of the array.</i>
------	-------------------------------

*Returns*

*The array of data from the device.*

- array< unsigned int > ^ [GetUIntBuffer](#) (uint16\_t value, uint16\_t index, int size)

*Gets an array of type unsigned int from the device.*

*Parameters*

value	<i>The value of the request.</i>
-------	----------------------------------

*Parameters*

index	<i>The index of the request.</i>
-------	----------------------------------

*Parameters*

size	<i>The size of the array.</i>
------	-------------------------------

*Returns*

*The array of data from the device.*

- array< int > ^ [GetIntBuffer](#) (uint16\_t value, uint16\_t index, int size)

*Gets an array of type int from the device.*

*Parameters*

value	<i>The value of the request.</i>
-------	----------------------------------

*Parameters*

index	<i>The index of the request.</i>
-------	----------------------------------

*Parameters*

size	<i>The size of the array.</i>
------	-------------------------------

*Returns**The array of data from the device.*

- IntPtr [OpenPipe](#) (uint8\_t endpointAddress)

*Open a Pipe to an USB Endpoint.**Parameters*

endpointAddress	<i>The Endpoint Number.</i>
-----------------	-----------------------------

*Returns**A handle to the endpoint.*

- void [ClosePipe](#) (IntPtr pipeHandle)

*Close a Pipe to an USB Endpoint.**Parameters*

pipeHandle	<i>The endpoint handle.</i>
------------	-----------------------------

- void [ResetPipe](#) (IntPtr pipeHandle)

*Reset a Pipe to an USB Endpoint.**Parameters*

pipeHandle	<i>The endpoint handle.</i>
------------	-----------------------------

- template<typename C >  
array< C > ^ [ReadPipe](#) (IntPtr pipeHandle, uint32\_t size)

*Read data from an USB Endpoint.**Parameters*

pipeHandle	<i>The endpoint handle.</i>
------------	-----------------------------

*Parameters*

size	<i>Number of elements to read.</i>
------	------------------------------------

*Returns**An array of data read.*

- template<typename C >  
void [WritePipe](#) (IntPtr pipeHandle, array< C >^ buffer)

*Write data to an USB Endpoint.**Parameters*

pipeHandle	<i>The endpoint handle.</i>
------------	-----------------------------

#### Parameters

<code>buffer</code>	<i>An array of data to write.</i>
---------------------	-----------------------------------

### Additional Inherited Members

#### 11.29.1 Detailed Description

[CGenericDevelopDeviceNet](#) is the class to use during development of a new device.

#### 11.29.2 Constructor & Destructor Documentation

**11.29.2.1 CGenericDevelopDeviceNet()** [CGenericDevelopDeviceNet](#) (  
    void )

Initialize a new instance of the [CGenericDevelopDeviceNet](#) class.

**11.29.2.2 ~CGenericDevelopDeviceNet()** [~CGenericDevelopDeviceNet](#) (  
    void )

#### 11.29.3 Member Function Documentation

**11.29.3.1 ClosePipe()** void ClosePipe (  
    IntPtr *pipeHandle* )

Close a Pipe to an USB Endpoint.

#### Parameters

<i>pipeHandle</i>	The endpoint handle.
-------------------	----------------------

**11.29.3.2 GetBuffer()** `array<C> ^ GetBuffer (`  
    `uint16_t value,`  
    `uint16_t index,`  
    `int size )`

**11.29.3.3 GetByteBuffer()** `array<char> ^ GetByteBuffer (`  
    `uint16_t value,`  
    `uint16_t index,`  
    `int size )`

Gets an array of type char from the device.

**Parameters**

<i>value</i>	The value of the request.
--------------	---------------------------

**Parameters**

<i>index</i>	The index of the request.
--------------	---------------------------

**Parameters**

<i>size</i>	The size of the array.
-------------	------------------------

**Returns**

The array of data from the device.

**11.29.3.4 GetIntBuffer()** `array<int> ^ GetIntBuffer (`  
    `uint16_t value,`  
    `uint16_t index,`  
    `int size )`

Gets an array of type int from the device.

**Parameters**

<i>value</i>	The value of the request.
--------------	---------------------------

**Parameters**

<i>index</i>	The index of the request.
--------------	---------------------------

**Parameters**

<i>size</i>	The size of the array.
-------------	------------------------

**Returns**

The array of data from the device.

**11.29.3.5 GetShortBuffer()** `array<short> ^ GetShortBuffer (`  
    `uint16_t value,`  
    `uint16_t index,`  
    `int size )`

Gets an array of type short from the device.

**Parameters**

<i>value</i>	The value of the request.
--------------	---------------------------

**Parameters**

<i>index</i>	The index of the request.
--------------	---------------------------

**Parameters**

<i>size</i>	The size of the array.
-------------	------------------------

**Returns**

The array of data from the device.

**11.29.3.6 GetUByteBuffer()** `array<unsigned char> ^ GetUByteBuffer (`  
    `uint16_t value,`  
    `uint16_t index,`  
    `int size )`

Gets an array of type unsigned char from the device.

**Parameters**

<i>value</i>	The value of the request.
--------------	---------------------------

**Parameters**

<i>index</i>	The index of the request.
--------------	---------------------------

**Parameters**

<i>size</i>	The size of the array.
-------------	------------------------

**Returns**

The array of data from the device.

**11.29.3.7 GetUIntBuffer()** `array<unsigned int> ^ GetUIntBuffer (`  
    `uint16_t value,`  
    `uint16_t index,`  
    `int size )`

Gets an array of type unsigned int from the device.

**Parameters**

<i>value</i>	The value of the request.
--------------	---------------------------



**Parameters**

<i>index</i>	The index of the request.
--------------	---------------------------

**Parameters**

<i>size</i>	The size of the array.
-------------	------------------------

**Returns**

The array of data from the device.

**11.29.3.8 GetUShortBuffer()** `array<unsigned short> ^ GetUShortBuffer (`  
    `uint16_t value,`  
    `uint16_t index,`  
    `int size )`

Gets an array of type unsigned short from the device.

**Parameters**

<i>value</i>	The value of the request.
--------------	---------------------------

**Parameters**

<i>index</i>	The index of the request.
--------------	---------------------------

**Parameters**

<i>size</i>	The size of the array.
-------------	------------------------

**Returns**

The array of data from the device.

**11.29.3.9 OpenPipe()** `IntPtr OpenPipe (`  
`uint8_t endpointAddress )`

Open a Pipe to an USB Endpoint.

**Parameters**

<i>endpointAddress</i>	The Endpoint Number.
------------------------	----------------------

**Returns**

A handle to the endpoint.

**11.29.3.10 ReadPipe()** `array<C> ^ ReadPipe (`  
`IntPtr pipeHandle,`  
`uint32_t size )`

Read data from an USB Endpoint.

**Parameters**

<i>pipeHandle</i>	The endpoint handle.
-------------------	----------------------

**Parameters**

<i>size</i>	Number of elements to read.
-------------	-----------------------------

**Returns**

An array of data read.

**11.29.3.11 ResetPipe()** `void ResetPipe (`  
    `IntPtr pipeHandle )`

Reset a Pipe to an USB Endpoint.

**Parameters**

<i>pipeHandle</i>	The endpoint handle.
-------------------	----------------------

**11.29.3.12 SetBuffer()** `void SetBuffer (`  
    `uint16_t value,`  
    `uint16_t index,`  
    `array< C >^ buffer )`

**11.29.3.13 SetByteBuffer()** `void SetByteBuffer (`  
    `uint16_t value,`  
    `uint16_t Index,`  
    `array< char >^ buffer )`

Sends an array of type char to the device.

**Parameters**

<i>value</i>	The value of the request.
--------------	---------------------------

**Parameters**

<i>Index</i>	The index of the request.
--------------	---------------------------

**Parameters**

<i>buffer</i>	The buffer to send.
---------------	---------------------

**11.29.3.14 SetIntBuffer()** `void SetIntBuffer (`  
    `uint16_t value,`  
    `uint16_t index,`  
    `array< int >^ buffer )`

Sends an array of type int to the device.

**Parameters**

<i>value</i>	The value of the request.
--------------	---------------------------

**Parameters**

<i>index</i>	The index of the request.
--------------	---------------------------

**Parameters**

<i>buffer</i>	The buffer to send.
---------------	---------------------

**11.29.3.15 SetShortBuffer()** `void SetShortBuffer (`  
    `uint16_t value,`  
    `uint16_t index,`  
    `array< short >^ buffer )`

Sends an array of type short to the device.

**Parameters**

<i>value</i>	The value of the request.
--------------	---------------------------

**Parameters**

<i>index</i>	The index of the request.
--------------	---------------------------

## Parameters

<i>buffer</i>	The buffer to send.
---------------	---------------------

**11.29.3.16 SetUByteBuffer()** void SetUByteBuffer (   
     uint16\_t *value*,   
     uint16\_t *index*,   
     array< unsigned char >^ *buffer* )

Sends an array of type unsigned char to the device.

## Parameters

<i>value</i>	The value of the request.
--------------	---------------------------

## Parameters

<i>index</i>	The index of the request.
--------------	---------------------------

## Parameters

<i>buffer</i>	The buffer to send.
---------------	---------------------

**11.29.3.17 SetUIntBuffer()** void SetUIntBuffer (   
     uint16\_t *value*,   
     uint16\_t *index*,   
     array< unsigned int >^ *buffer* )

Sends an array of unsigned int to the device.

## Parameters

<i>value</i>	The value of the request.
--------------	---------------------------

## Parameters

<i>index</i>	The index of the request.
--------------	---------------------------

## Parameters

<i>buffer</i>	The buffer to send.
---------------	---------------------

**11.29.3.18 SetUShortBuffer()** `void SetUShortBuffer (`  
    `uint16_t value,`  
    `uint16_t index,`  
    `array< unsigned short >^ buffer )`

Sends an array of type unsigned short to the device.

## Parameters

<i>value</i>	The value of the request.
--------------	---------------------------

## Parameters

<i>index</i>	The index of the request.
--------------	---------------------------

## Parameters

<i>buffer</i>	The buffer to send.
---------------	---------------------

**11.29.3.19 SetValue()** `void SetValue (`  
    `uint16_t value,`  
    `uint16_t index )`

Sets .

Parameters

<i>value</i>	The value of the request.
--------------	---------------------------

Parameters

<i>index</i>	The index of the request.
--------------	---------------------------

**11.29.3.20 WritePipe()** `void WritePipe (`  
    `IntPtr pipeHandle,`  
    `array< C >^ buffer )`

Write data to an USB Endpoint.

Parameters

<i>pipeHandle</i>	The endpoint handle.
-------------------	----------------------

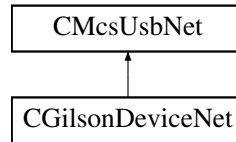
Parameters

<i>buffer</i>	An array of data to write.
---------------	----------------------------

## 11.30 CGilsonDeviceNet Class Reference

[CGilsonDeviceNet](#) is the class to control a Gilson device.

Inheritance diagram for CGilsonDeviceNet:



### Public Member Functions

- [CGilsonDeviceNet](#) (void)  
*Initialize a new instance of the [CGilsonDeviceNet](#) class.*
- [~CGilsonDeviceNet](#) (void)
- void [ConnectSlave](#) (byte ID)
- void [SendImmediate](#) (wchar\_t command)
- String ^ [SendImmediateGetResponse](#) (wchar\_t command)
- void [SendBuffered](#) (String^ command)
- String ^ [GetLastAnswer](#) ()

### Protected Attributes

- CGilsonDevice \* [m\\_pGilsonDevice](#)

### Additional Inherited Members

#### 11.30.1 Detailed Description

[CGilsonDeviceNet](#) is the class to control a Gilson device.

#### 11.30.2 Constructor & Destructor Documentation

**11.30.2.1 [CGilsonDeviceNet\(\)](#)** [CGilsonDeviceNet](#) (  
void )

Initialize a new instance of the [CGilsonDeviceNet](#) class.

**11.30.2.2 [~CGilsonDeviceNet\(\)](#)** [~CGilsonDeviceNet](#) (  
void )

#### 11.30.3 Member Function Documentation



**11.30.3.1 ConnectSlave()** `void ConnectSlave (`  
`byte ID )`

**11.30.3.2 GetLastAnswer()** `String ^ GetLastAnswer ( )`

**11.30.3.3 SendBuffered()** `void SendBuffered (`  
`String^ command )`

**11.30.3.4 SendImmediate()** `void SendImmediate (`  
`wchar_t command )`

**11.30.3.5 SendImmediateGetResponse()** `String ^ SendImmediateGetResponse (`  
`wchar_t command )`

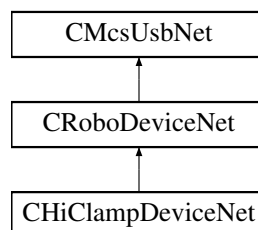
#### 11.30.4 Member Data Documentation

**11.30.4.1 m\_pGilsonDevice** `CGilsonDevice* m_pGilsonDevice [protected]`

## 11.31 CHiClampDeviceNet Class Reference

[CHiClampDeviceNet](#) is the to control the MCS HiClamp device

Inheritance diagram for CHiClampDeviceNet:



#### Public Member Functions

- [CHiClampDeviceNet](#) (void)

## Properties

- [CRoboDacqNet](#)<sup>^</sup> [RoboDacq](#) [get]

## Additional Inherited Members

### 11.31.1 Detailed Description

[CHiClampDeviceNet](#) is the to control the MCS HiClamp device

### 11.31.2 Constructor & Destructor Documentation

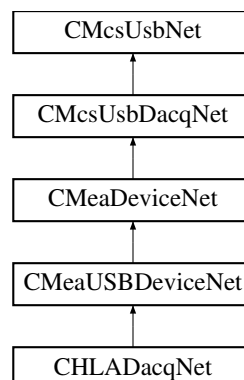
**11.31.2.1 CHiClampDeviceNet()** [CHiClampDeviceNet](#) (  
void )

### 11.31.3 Property Documentation

**11.31.3.1 RoboDacq** [CRoboDacqNet](#)<sup>^</sup> [RoboDacq](#) [get]

## 11.32 CHLADacqNet Class Reference

Inheritance diagram for CHLADacqNet:



## Public Member Functions

- [CHLADacqNet](#) (void)

## Additional Inherited Members

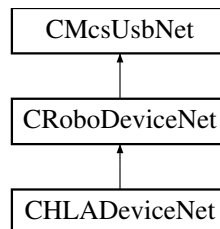
### 11.32.1 Constructor & Destructor Documentation

**11.32.1.1 CHLADacqNet()** `CHLADacqNet ( void )`

## 11.33 CHLADeviceNet Class Reference

[CHLADeviceNet](#) is the to control the MCS HLA device

Inheritance diagram for CHLADeviceNet:



## Public Member Functions

- [CHLADeviceNet](#) (void)

## Properties

- [CHLADacqNet](#)<sup>^</sup> [HLADacq](#) [get]
- [CSerialPortNet](#)<sup>^</sup> [SerialPort](#) [get]

## Additional Inherited Members

### 11.33.1 Detailed Description

[CHLADeviceNet](#) is the to control the MCS HLA device

### 11.33.2 Constructor & Destructor Documentation

**11.33.2.1 CHLADeviceNet()** `CHLADeviceNet ( void )`

### 11.33.3 Property Documentation

**11.33.3.1 HLADacq** [CHLADacqNet](#)<sup>^</sup> HLADacq [get]

**11.33.3.2 SerialPort** [CSerialPortNet](#)<sup>^</sup> SerialPort [get]

## 11.34 CMcsUsbDacqNet::CHWInfo Class Reference

Class to provide hardware information about the device.

### Classes

- class [CVoltageRangeInfoNet](#)

### Public Member Functions

- [CHWInfo](#) ([CMcsUsbDacqNet](#)<sup>^</sup> device)
- virtual uint32\_t [GetNumberOfHWADCCChannels](#) ([System::Runtime::InteropServices::Out]int% numberOf↔ Channels)  
*Get the number of analog channels the device supports.*
- virtual uint32\_t [GetNumberOfHWDigitalChannels](#) ([System::Runtime::InteropServices::Out]int% numberOf↔ Channels)  
*Get the number of digital channels the device supports.*
- virtual bool [IsDigitalChannelDedicated](#) ()  
*Query if the digital channel replaces an analog channel when enabled (e.g. on MC\_Card) or adds a channel link on USB devices.*
- virtual uint32\_t [GetAvailableSampleRates](#) ([System::Runtime::InteropServices::Out]System::Collections::↔ Generic::List< int32\_t >^% sampleRates)
- virtual uint32\_t [GetAvailableVoltageRangesInMicroVolt](#) ([System::Runtime::InteropServices::Out]System::↔ Collections::Generic::List< int32\_t >^% voltageRanges)
- virtual uint32\_t [GetAvailableVoltageRangesInMicroVoltAndStringsInMilliVolt](#) ([System::Runtime::Interop↔ Services::Out]System::Collections::Generic::List< [CVoltageRangeInfoNet](#)<sup>^</sup> >^% voltageRanges)

### 11.34.1 Detailed Description

Class to provide hardware information about the device.

### 11.34.2 Constructor & Destructor Documentation

**11.34.2.1 CHWInfo()** `CHWInfo ( CMcsUsbDacqNet^ device )`

### 11.34.3 Member Function Documentation

**11.34.3.1 GetAvailableSampleRates()** `virtual uint32_t GetAvailableSampleRates ( [System::Runtime::InteropServices::Out] System::Collections::Generic::List< int32↵  
_t >^% sampleRates ) [virtual]`

**11.34.3.2 GetAvailableVoltageRangesInMicroVolt()** `virtual uint32_t GetAvailableVoltageRangesIn↵  
MicroVolt ( [System::Runtime::InteropServices::Out] System::Collections::Generic::List< int32↵  
_t >^% voltageRanges ) [virtual]`

**11.34.3.3 GetAvailableVoltageRangesInMicroVoltAndStringsInMilliVolt()** `virtual uint32_t GetAvailable↵  
VoltageRangesInMicroVoltAndStringsInMilliVolt ( [System::Runtime::InteropServices::Out] System::Collections::Generic::List< CVoltageRangeInfoNet↵  
voltageRanges ) [virtual]`

**11.34.3.4 GetNumberOfHWADCCChannels()** `virtual uint32_t GetNumberOfHWADCCChannels ( [System::Runtime::InteropServices::Out] int% numberOfChannels ) [virtual]`

Get the number of analog channels the device supports.

#### Parameters

<i>numberOfChannels</i>	Number of analog channels the device supports.
-------------------------	--

#### Returns

Error Status. 0 on success.

**11.34.3.5 GetNumberOfHWDigitalChannels()** `virtual uint32_t GetNumberOfHWDigitalChannels ( [System::Runtime::InteropServices::Out] int% numberOfChannels ) [virtual]`

Get the number of digital channels the device supports.

## Parameters

<i>numberOfChannels</i>	Number of digital channels the device supports.
-------------------------	---

## Returns

Error Status. 0 on success.

### 11.34.3.6 IsDigitalChannelDedicated() `virtual bool IsDigitalChannelDedicated ( ) [virtual]`

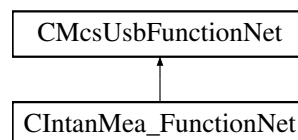
Query if the digital channel replaces an analog channel when enabled (e.g. on MC\_Card) or adds a channel link on USB devices.

## Returns

false when the digital channel replaces an analog channel when enabled, true when the digital channels is appended to the analog channels when enabled.

## 11.35 CIntanMea\_FunctionNet Class Reference

Inheritance diagram for CIntanMea\_FunctionNet:



### Public Member Functions

- [CIntanMea\\_FunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> intalMea\_FunctionPointerContainer)
- [CIntanMea\\_FunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- [int](#) [GetUpperFrequencyByIndex](#) (unsigned short index)
- [int](#) [GetLowerFrequencyByIndex](#) (unsigned short index)
- [int64\\_t](#) [GetDSPHighPassByIndex](#) (unsigned short index)
- [int](#) [GetIntanRegister](#) (unsigned short chip, unsigned short registernumber)
- [int](#) [GetImpedanceResult](#) (unsigned short channel)
- [void](#) [SetBandwidthByIndex](#) (int upper\_index, int lower\_index)
- [void](#) [SetDSPHighPassByIndex](#) (int index)
- [void](#) [AmplifierSettle](#) ()
- [void](#) [SetIntanRegister](#) (unsigned short register\_number, int value)
- [void](#) [SetDiagnosticMode](#) (unsigned char onoff)
- [void](#) [BeginImpedanceCheck](#) (array< int ><sup>^</sup> config\_values)

## Additional Inherited Members

### 11.35.1 Constructor & Destructor Documentation

**11.35.1.1 CIntanMea\_FunctionNet()** [1/2] `CIntanMea_FunctionNet ( CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ intalMea_FunctionPointerContainer )`

**11.35.1.2 CIntanMea\_FunctionNet()** [2/2] `CIntanMea_FunctionNet ( CMcsUsbNet^ mcsusb )`

### 11.35.2 Member Function Documentation

**11.35.2.1 AmplifierSettle()** `void AmplifierSettle ( )`

**11.35.2.2 BeginImpedanceCheck()** `void BeginImpedanceCheck ( array< int >^ config_values )`

**11.35.2.3 GetDSPHighPassByIndex()** `int64_t GetDSPHighPassByIndex ( unsigned short index )`

**11.35.2.4 GetImpedanceResult()** `int GetImpedanceResult ( unsigned short channel )`

**11.35.2.5 GetIntanRegister()** `int GetIntanRegister ( unsigned short chip, unsigned short registernumber )`

**11.35.2.6 GetLowerFrequencyByIndex()** `int GetLowerFrequencyByIndex (`  
`unsigned short index )`

**11.35.2.7 GetUpperFrequencyByIndex()** `int GetUpperFrequencyByIndex (`  
`unsigned short index )`

**11.35.2.8 SetBandwidthByIndex()** `void SetBandwidthByIndex (`  
`int upper_index,`  
`int lower_index )`

**11.35.2.9 SetDiagnosticMode()** `void SetDiagnosticMode (`  
`unsigned char onoff )`

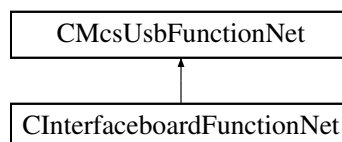
**11.35.2.10 SetDSPHighPassByIndex()** `void SetDSPHighPassByIndex (`  
`int index )`

**11.35.2.11 SetIntanRegister()** `void SetIntanRegister (`  
`unsigned short register_number,`  
`int value )`

## 11.36 CInterfaceboardFunctionNet Class Reference

[CInterfaceboardFunctionNet](#) is the class to control the Interfaceboard

Inheritance diagram for CInterfaceboardFunctionNet:





## Public Member Functions

- [CInterfaceboardFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> pInterfaceboardFunctionPointerContainer)  
*Initializes a new instance of the [CInterfaceboardFunctionNet](#) class.*
- [CInterfaceboardFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- virtual [~CInterfaceboardFunctionNet](#) ()
- [!CInterfaceboardFunctionNet](#) ()
- void [SetCardinalDacqSamplerate](#) (uint32\_t samplerate)  
*Sets the fundamental/cardinal data acquisition samplerate of the Interfaceboard, default is 50 kHz*
- uint32\_t [GetCardinalDacqSamplerate](#) ()  
*Gets the fundamental/cardinal data acquisition samplerate of the Interfaceboard, default is 50 kHz*
- void [SetCardinalStgOutputrate](#) (uint32\_t outputrate)  
*Sets the fundamental/cardinal STG output rate of the Interfaceboard, default is 50 kHz*
- uint32\_t [GetCardinalStgOutputrate](#) ()  
*Gets the fundamental/cardinal STG output rate of the Interfaceboard, default is 50 kHz*

## Additional Inherited Members

### 11.36.1 Detailed Description

[CInterfaceboardFunctionNet](#) is the class to control the Interfaceboard

### 11.36.2 Constructor & Destructor Documentation

**11.36.2.1 CInterfaceboardFunctionNet()** [1/2] [CInterfaceboardFunctionNet](#) (  
    [CMcsUsbNet](#)<sup>^</sup> mcsusb,  
    [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> pInterfaceboardFunctionPointerContainer )

Initializes a new instance of the [CInterfaceboardFunctionNet](#) class.

**11.36.2.2 CInterfaceboardFunctionNet()** [2/2] [CInterfaceboardFunctionNet](#) (  
    [CMcsUsbNet](#)<sup>^</sup> mcsusb )

**11.36.2.3 ~CInterfaceboardFunctionNet()** virtual [~CInterfaceboardFunctionNet](#) ( ) [virtual]

**11.36.2.4 !CInterfaceboardFunctionNet()** [!CInterfaceboardFunctionNet](#) ( )

### 11.36.3 Member Function Documentation

#### 11.36.3.1 GetCardinalDacqSamplerate() `uint32_t GetCardinalDacqSamplerate ( )`

Gets the fundamental/cardinal data acquisition samplerate of the Interfaceboard, default is 50 kHz

##### Returns

The samplerate in Hz.

#### 11.36.3.2 GetCardinalStgOutputrate() `uint32_t GetCardinalStgOutputrate ( )`

Gets the fundamental/cardinal STG output rate of the Interfaceboard, default is 50 kHz

##### Returns

The output rate in Hz.

#### 11.36.3.3 SetCardinalDacqSamplerate() `void SetCardinalDacqSamplerate ( uint32_t samplerate )`

Sets the fundamental/cardinal data acquisition samplerate of the Interfaceboard, default is 50 kHz

##### Parameters

<i>samplerate</i>	The samplerate in Hz.
-------------------	-----------------------

#### 11.36.3.4 SetCardinalStgOutputrate() `void SetCardinalStgOutputrate ( uint32_t outputrate )`

Sets the fundamental/cardinal STG output rate of the Interfaceboard, default is 50 kHz

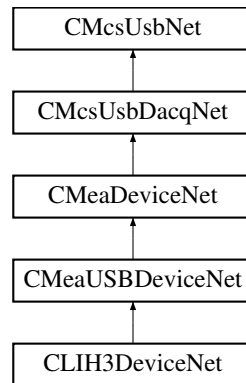
##### Parameters

<i>outputrate</i>	The output rate in Hz.
-------------------	------------------------

## 11.37 CLIH3DeviceNet Class Reference

[CLIH3DeviceNet](#) is the class to access the HEKA LIH3 device.

Inheritance diagram for CLIH3DeviceNet:



## Public Member Functions

- [CLIH3DeviceNet](#) ()  
*Initializes a new instance of the [CLIH3DeviceNet](#) class.*
- virtual [~CLIH3DeviceNet](#) ()
- [!CLIH3DeviceNet](#) ()
- void [DummyCommand](#) (uint32\_t dummyParameter)  
*Dummy command to show how to use the DLL.*
- void [SetEepromPage](#) (uint32\_t EepromStartAddress, array< int8\_t >^ EepromData, LIH30\_EPC10\_Bus\_EnumNet epc10bus)  
*Writes into Eeprom on the EPC10 EEPROM*
- array< int8\_t >^ [GetEepromPage](#) (uint32\_t EepromStartAddress, int EepromData\_Length, LIH30\_EPC10\_Bus\_EnumNet epc10bus)  
*Reads the requested amount of Eeprom byte from the EPC10 EEPROM*
- void [SetSampleInterval](#) (uint32\_t SampleInterval)  
*Sets the Sample Interval for the DACQ and Stimulation*
- uint32\_t [GetSampleInterval](#) ()  
*Gets the Sample Interval for the DACQ and Stimulation*
- void [SetADCOffset](#) (LIH30\_ADC\_Channel\_EnumNet AdcChannel, uint32\_t Offset)  
*Sets the ADC offset of the DACQ for a single channel*
- uint32\_t [GetADCOffset](#) (LIH30\_ADC\_Channel\_EnumNet AdcChannel)  
*Gets the ADC offset of the DACQ for a single channel*
- uint32\_t [ReadClipping](#) (LIH30\_EPC10\_Bus\_EnumNet epc10bus)  
*Gets the clipping information*
- void [SetDigOutState](#) (uint16\_t DigOutState)  
*Writes to the LIH30 digital output*
- uint16\_t [GetDigInState](#) ()  
*Reads from the LIH30 digital input*
- void [SendCommand](#) (LIH30\_EPC10\_Bus\_EnumNet epc10bus, uint16\_t Command)  
*Send command to the EPC10*
- uint16\_t [GetDacqRunStatus](#) ()  
*Gets the data acquisition running status*
- void [SetDacUseldleValue](#) (uint32\_t DacChannel, bool Useldle)  
*Sets if the DAC Idle value is used after stimulation*
- bool [GetDacUseldleValue](#) (uint32\_t DacChannel)  
*Gets if the DAC Idle value is used after stimulation*

- void [SetDacIdleValue](#) (uint32\_t DacChannel, int32\_t IdleValue)  
*Sets the DAC Idle value*
- int32\_t [GetDacIdleValue](#) (uint32\_t DacChannel)  
*Gets the DAC Idle value*
- void [EnableUserTrigger](#) (bool enable)  
*Enables the User Trigger*
- bool [IsUserTriggerEnabled](#) ()  
*Is the User Trigger enabled*

## Properties

- [CStimulusFunctionNet](#)<sup>^</sup> [StimulusFunction](#) [get]

## Additional Inherited Members

### 11.37.1 Detailed Description

[CLIH3DeviceNet](#) is the class to access the HEKA LIH3 device.

### 11.37.2 Constructor & Destructor Documentation

#### 11.37.2.1 CLIH3DeviceNet() [CLIH3DeviceNet](#) ( )

Initializes a new instance of the [CLIH3DeviceNet](#) class.

#### 11.37.2.2 ~CLIH3DeviceNet() [virtual](#) [~CLIH3DeviceNet](#) ( ) [virtual]

#### 11.37.2.3 "!CLIH3DeviceNet() [!CLIH3DeviceNet](#) ( )

### 11.37.3 Member Function Documentation

#### 11.37.3.1 DummyCommand() [void](#) [DummyCommand](#) ( uint32\_t *dummyParameter* )

Dummy command to show how to use the DLL.

## Parameters

<i>dummyParameter</i>	parameter to send to the device
-----------------------	---------------------------------

**11.37.3.2 EnableUserTrigger()** `void EnableUserTrigger (`  
    `bool enable )`

Enables the User Trigger

## Parameters

<i>enable</i>	Enable
---------------	--------

**11.37.3.3 GetADCOffset()** `uint32_t GetADCOffset (`  
    `LIH30_ADC_Channel_EnumNet AdcChannel )`

Gets the ADC offset of the DACQ for a single channel

## Parameters

<i>AdcChannel</i>	The ADC channel
-------------------	-----------------

## Returns

The offset for the given channel number

**11.37.3.4 GetDacIdleValue()** `int32_t GetDacIdleValue (`  
    `uint32_t DacChannel )`

Gets the DAC Idle value

## Parameters

<i>DacChannel</i>	The DAC channel
-------------------	-----------------

## Returns

The idle value

**11.37.3.5 GetDacqRunStatus()** `uint16_t GetDacqRunStatus ( )`

Gets the data acquisition running status

**Returns**

The status (1: running / 0: stopped)

**11.37.3.6 GetDacUseIdleValue()** `bool GetDacUseIdleValue (   
uint32_t DacChannel )`

Gets if the DAC Idle value is used after stimulation

**Parameters**

<i>DacChannel</i>	The DAC channel
-------------------	-----------------

**Returns**

Use idle value

**11.37.3.7 GetDigInState()** `uint16_t GetDigInState ( )`

Reads from the LIH30 digital input

**Returns**

The bit mask defining the digital input state

**11.37.3.8 GetEepromPage()** `array<int8_t> ^ GetEepromPage (   
uint32_t EepromStartAddress,   
int EepromData_Length,   
LIH30_EPC10_Bus_EnumNet epc10bus )`

Reads the requested amount of Eeprom byte from the EPC10 EEPROM

**Parameters**

<i>EepromStartAddress</i>	start address of memory area to read from
<i>EepromData_Length</i>	The maximal length of EepromData.
<i>epc10bus</i>	The EPC10 bus

**Returns**

pointer to internal memory for the requested amount of data

**11.37.3.9 GetSampleInterval()** `uint32_t GetSampleInterval ( )`

Gets the Sample Interval for the DACQ and Stimulation

**Returns**

Sample Interval configured on the device

**11.37.3.10 IsUserTriggerEnabled()** `bool IsUserTriggerEnabled ( )`

Is the User Trigger enabled

**Returns**

Enabled

**11.37.3.11 ReadClipping()** `uint32_t ReadClipping (`  
`LIH30_EPC10_Bus_EnumNet epc10bus )`

Gets the clipping information

**Parameters**

<i>epc10bus</i>	The EPC10 bus
-----------------	---------------

**Returns**

The clipping value

**11.37.3.12 SendCommand()** `void SendCommand (`  
`LIH30_EPC10_Bus_EnumNet epc10bus,`  
`uint16_t Command )`

Send command to the EPC10

## Parameters

<i>epc10bus</i>	The EPC10 bus
<i>Command</i>	The command

**11.37.3.13 SetADCOffset()** void SetADCOffset (   
     LIH30\_ADC\_Channel\_EnumNet *AdcChannel*,  
     uint32\_t *Offset* )

Sets the ADC offset of the DACQ for a single channel

## Parameters

<i>AdcChannel</i>	The ADC channel
<i>Offset</i>	The offset for the given channel number

**11.37.3.14 SetDacIdleValue()** void SetDacIdleValue (   
     uint32\_t *DacChannel*,  
     int32\_t *IdleValue* )

Sets the DAC Idle value

## Parameters

<i>DacChannel</i>	The DAC channel
<i>IdleValue</i>	The idle value

**11.37.3.15 SetDacUseIdleValue()** void SetDacUseIdleValue (   
     uint32\_t *DacChannel*,  
     bool *UseIdle* )

Sets if the DAC Idle value is used after stimulation

## Parameters

<i>DacChannel</i>	The DAC channel
<i>UseIdle</i>	Use idle value

**11.37.3.16 SetDigOutState()** void SetDigOutState (   
     uint16\_t *DigOutState* )



Writes to the LIH30 digital output

#### Parameters

<i>DigOutState</i>	The bit mask defining the digital output state
--------------------	--

**11.37.3.17 SetEepromPage()** void SetEepromPage (   
     uint32\_t *EepromStartAddress*,   
     array< int8\_t >^ *EepromData*,   
     LIH30\_EPC10\_Bus\_EnumNet *epc10bus* )

Writes into EEprom on the EPC10 EEPROM

#### Parameters

<i>EepromStartAddress</i>	start address of memory area to write to
<i>EepromData</i>	pointer to internal memory for the supported amount of data
<i>epc10bus</i>	The EPC10 bus

**11.37.3.18 SetSampleInterval()** void SetSampleInterval (   
     uint32\_t *SampleInterval* )

Sets the Sample Interval for the DACQ and Stimulation

#### Parameters

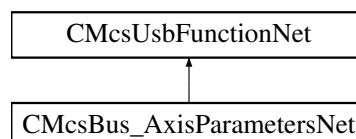
<i>SampleInterval</i>	between the samples, Sample interval is available from 1 to 4194303
-----------------------	---

## 11.37.4 Property Documentation

**11.37.4.1 StimulusFunction** CStimulusFunctionNet^ StimulusFunction [get]

## 11.38 CMcsBus\_AxisParametersNet Class Reference

Inheritance diagram for CMcsBus\_AxisParametersNet:



**Public Member Functions**

- [CMcsBus\\_AxisParametersNet](#) ([CMcsUsbNet](#)<sup>^</sup> device)
- [~CMcsBus\\_AxisParametersNet](#) (void)
- void [SetAxisParametersEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, unsigned short index, unsigned int parameter)
- void [SetAxisParametersEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, unsigned short index, int parameter)
- unsigned int [GetAxisParametersUnsignedEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, unsigned short index)
- int [GetAxisParametersSignedEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, unsigned short index)

**Additional Inherited Members****11.38.1 Constructor & Destructor Documentation**

**11.38.1.1 [CMcsBus\\_AxisParametersNet\(\)](#)** [CMcsBus\\_AxisParametersNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> device )

**11.38.1.2 [~CMcsBus\\_AxisParametersNet\(\)](#)** [~CMcsBus\\_AxisParametersNet](#) (  
void )

**11.38.2 Member Function Documentation**

**11.38.2.1 [GetAxisParametersSignedEeprom\(\)](#)** int [GetAxisParametersSignedEeprom](#) (  
unsigned char *busnumber*,  
unsigned char *busaddress*,  
unsigned char *axis*,  
unsigned short *index* )

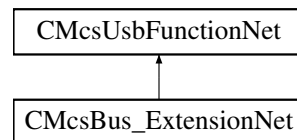
**11.38.2.2 [GetAxisParametersUnsignedEeprom\(\)](#)** unsigned int [GetAxisParametersUnsignedEeprom](#) (  
unsigned char *busnumber*,  
unsigned char *busaddress*,  
unsigned char *axis*,  
unsigned short *index* )

**11.38.2.3 SetAxisParametersEeprom() [1/2]** void SetAxisParametersEeprom (   
     unsigned char *busnumber*,   
     unsigned char *busaddress*,   
     unsigned char *axis*,   
     unsigned short *index*,   
     int *parameter* )

**11.38.2.4 SetAxisParametersEeprom() [2/2]** void SetAxisParametersEeprom (   
     unsigned char *busnumber*,   
     unsigned char *busaddress*,   
     unsigned char *axis*,   
     unsigned short *index*,   
     unsigned int *parameter* )

## 11.39 CMcsBus\_ExtensionNet Class Reference

Inheritance diagram for CMcsBus\_ExtensionNet:



### Public Member Functions

- [CMcsBus\\_ExtensionNet](#) ([CMcsUsbNet](#)<sup>^</sup> device)
- [~CMcsBus\\_ExtensionNet](#) (void)
- void [SetLEDSwitch](#) (unsigned char busnumber, unsigned char busaddress, unsigned short LEDSwitch)
- unsigned short [GetLEDSwitch](#) (unsigned char busnumber, unsigned char busaddress)

### Additional Inherited Members

#### 11.39.1 Constructor & Destructor Documentation

**11.39.1.1 CMcsBus\_ExtensionNet()** [CMcsBus\\_ExtensionNet](#) (   
     [CMcsUsbNet](#)<sup>^</sup> *device* )

**11.39.1.2 ~CMcsBus\_ExtensionNet()** [~CMcsBus\\_ExtensionNet](#) (   
     void )

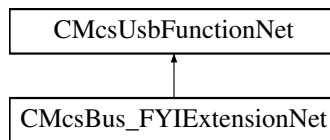
### 11.39.2 Member Function Documentation

**11.39.2.1 GetLEDSwitch()** unsigned short GetLEDSwitch (   
     unsigned char *busnumber*,   
     unsigned char *busaddress* )

**11.39.2.2 SetLEDSwitch()** void SetLEDSwitch (   
     unsigned char *busnumber*,   
     unsigned char *busaddress*,   
     unsigned short *LEDSwitch* )

## 11.40 CMcsBus\_FYIExtensionNet Class Reference

Inheritance diagram for CMcsBus\_FYIExtensionNet:



### Public Member Functions

- [CMcsBus\\_FYIExtensionNet](#) ([CMcsUsbNet](#)<sup>^</sup> device)
- [~CMcsBus\\_FYIExtensionNet](#) (void)
- void [SetValves](#) (unsigned char busnumber, unsigned char busaddress, unsigned int states)
- unsigned int [GetValves](#) (unsigned char busnumber, unsigned char busaddress)
- void [SetDIO](#) (unsigned char busnumber, unsigned char busaddress, unsigned short io)
- unsigned short [GetDIO](#) (unsigned char busnumber, unsigned char busaddress)
- void [SetSingleHeater](#) (unsigned char busnumber, unsigned char busaddress, short index, unsigned short power)
- unsigned short [GetSingleHeater](#) (unsigned char busnumber, unsigned char busaddress, short index)

### Additional Inherited Members

#### 11.40.1 Constructor & Destructor Documentation

**11.40.1.1 CMcsBus\_FYIExtensionNet()** [CMcsBus\\_FYIExtensionNet](#) (   
     [CMcsUsbNet](#)<sup>^</sup> *device* )

**11.40.1.2** `~CMcsBus_FYIExtensionNet()` `~CMcsBus_FYIExtensionNet` (  
    `void` )

## 11.40.2 Member Function Documentation

**11.40.2.1** `GetDIO()` `unsigned short GetDIO` (  
    `unsigned char busnumber`,  
    `unsigned char busaddress` )

**11.40.2.2** `GetSingleHeater()` `unsigned short GetSingleHeater` (  
    `unsigned char busnumber`,  
    `unsigned char busaddress`,  
    `short index` )

**11.40.2.3** `GetValves()` `unsigned int GetValves` (  
    `unsigned char busnumber`,  
    `unsigned char busaddress` )

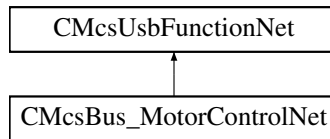
**11.40.2.4** `SetDIO()` `void SetDIO` (  
    `unsigned char busnumber`,  
    `unsigned char busaddress`,  
    `unsigned short io` )

**11.40.2.5** `SetSingleHeater()` `void SetSingleHeater` (  
    `unsigned char busnumber`,  
    `unsigned char busaddress`,  
    `short index`,  
    `unsigned short power` )

**11.40.2.6** `SetValves()` `void SetValves` (  
    `unsigned char busnumber`,  
    `unsigned char busaddress`,  
    `unsigned int states` )

## 11.41 CMcsBus\_MotorControlNet Class Reference

Inheritance diagram for CMcsBus\_MotorControlNet:



### Public Member Functions

- [CMcsBus\\_MotorControlNet](#) ([CMcsUsbNet](#)<sup>^</sup> device)
- [~CMcsBus\\_MotorControlNet](#) (void)
- void [SetMCScalingFactorEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, int factor)
- int [GetMCScalingFactorEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCScalingFactor](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, int factor)
- int [GetMCScalingFactor](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCMaxSpeedEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, unsigned short speed)
- unsigned short [GetMCMaxSpeedEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCMaxSpeed](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, unsigned short speed)
- unsigned short [GetMCMaxSpeed](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCMaxTravelEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, int travel)
- int [GetMCMaxTravelEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCMaxTravel](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, int travel)
- int [GetMCMaxTravel](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCCurrentEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, short current)
- short [GetMCCurrentEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCCurrent](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, short current)
- short [GetMCCurrent](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCMaxCurrentEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, short current)
- short [GetMCMaxCurrentEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCMaxCurrent](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, short current)
- short [GetMCMaxCurrent](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCRegulatorGainEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, short gain)
- short [GetMCRegulatorGainEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCRegulatorGain](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, short gain)
- short [GetMCRegulatorGain](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCMaxAccelerationEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, unsigned short acceleration)
- unsigned short [GetMCMaxAccelerationEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)

- void [SetMCMaxAcceleration](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, unsigned short acceleration)
- unsigned short [GetMCMaxAcceleration](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCStandbyCurrentEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, short percent)
- short [GetMCStandbyCurrentEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCStandbyCurrent](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, short percent)
- short [GetMCStandbyCurrent](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCStandbyTimeEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, short t)
- short [GetMCStandbyTimeEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCStandbyTime](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, short t)
- short [GetMCStandbyTime](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCBreakCurrentEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, short current)
- short [GetMCBreakCurrentEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCBreakCurrent](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, short current)
- short [GetMCBreakCurrent](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCConfigEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, unsigned short config)
- unsigned short [GetMCConfigEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCConfig](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, unsigned short config)
- unsigned short [GetMCConfig](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCSpeedEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, unsigned short speed)
- unsigned short [GetMCSpeedEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCSpeed](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, short speed)
- short [GetMCSpeed](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCAccelerationEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, unsigned short acceleration)
- unsigned short [GetMCAccelerationEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCAcceleration](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, unsigned short acceleration)
- unsigned short [GetMCAcceleration](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCReferenceCurrentEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, short current)
- short [GetMCReferenceCurrentEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCReferenceCurrent](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, short current)
- short [GetMCReferenceCurrent](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCCurrentModeEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, RoboCurrentModeEnumNet mode)
- RoboCurrentModeEnumNet [GetMCCurrentModeEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)

- void [SetMCCurrentMode](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, RoboCurrentModeEnumNet mode)
- RoboCurrentModeEnumNet [GetMCCurrentMode](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCAxisRevisionEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, unsigned short revision)
- unsigned short [GetMCAxisRevisionEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCSpeedUnitEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, int32\_t speedunit)
- int32\_t [GetMCSpeedUnitEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCOutputOnOff](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, bool OnOff\_status)
- bool [GetMCOutputOnOff](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCSpeedShortCommand](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, short speed)
- short [GetMCSpeedShortCommand](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCAccelerationShortCommand](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, unsigned short acceleration)
- unsigned short [GetMCAccelerationShortCommand](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCCurrentShortCommand](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, short current)
- short [GetMCCurrentShortCommand](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCMaxTravelShortCommand](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, int travel)
- int [GetMCMaxTravelShortCommand](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCCurrentPosition](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, int position)
- int [GetMCCurrentPosition](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCNewPosition](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, int position)
- int [GetMCNewPosition](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- short [GetMCCurrentSpeed](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [StartMCMovement](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCRotation](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, unsigned char onoff)
- unsigned short [GetMCMovement](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCReference](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, unsigned char switch\_enable, unsigned char switch\_polarity)
- unsigned char [GetMCReference](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, [System::Runtime::InteropServices::Out]unsigned char% switch\_port)
- void [StopMCMovement](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCCurrentModeShortCommand](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, RoboCurrentModeEnumNet mode)
- RoboCurrentModeEnumNet [GetMCCurrentModeShortCommand](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- unsigned short [GetMCPhase](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- unsigned short [GetMCPhaseOffset](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetSubChannel](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, unsigned short subchannel)
- unsigned short [GetSubChannel](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)



## Additional Inherited Members

### 11.41.1 Constructor & Destructor Documentation

**11.41.1.1 CMcsBus\_MotorControlNet()** `CMcsBus_MotorControlNet ( CMcsUsbNet^ device )`

**11.41.1.2 ~CMcsBus\_MotorControlNet()** `~CMcsBus_MotorControlNet ( void )`

### 11.41.2 Member Function Documentation

**11.41.2.1 GetMCAcceleration()** `unsigned short GetMCAcceleration ( unsigned char busnumber, unsigned char busaddress, unsigned char axis )`

**11.41.2.2 GetMCAccelerationEeprom()** `unsigned short GetMCAccelerationEeprom ( unsigned char busnumber, unsigned char busaddress, unsigned char axis )`

**11.41.2.3 GetMCAccelerationShortCommand()** `unsigned short GetMCAccelerationShortCommand ( unsigned char busnumber, unsigned char busaddress, unsigned char axis )`

**11.41.2.4 GetMCAXisRevisionEeprom()** `unsigned short GetMCAXisRevisionEeprom ( unsigned char busnumber, unsigned char busaddress, unsigned char axis )`

**11.41.2.5 GetMCBreakCurrent()** short GetMCBreakCurrent (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

**11.41.2.6 GetMCBreakCurrentEeprom()** short GetMCBreakCurrentEeprom (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

**11.41.2.7 GetMCConfig()** unsigned short GetMCConfig (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

**11.41.2.8 GetMCConfigEeprom()** unsigned short GetMCConfigEeprom (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

**11.41.2.9 GetMCCurrent()** short GetMCCurrent (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

**11.41.2.10 GetMCCurrentEeprom()** short GetMCCurrentEeprom (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

**11.41.2.11 GetMCCurrentMode()** RoboCurrentModeEnumNet GetMCCurrentMode (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

- 11.41.2.12 GetMCCurrentModeEeprom()** RoboCurrentModeEnumNet GetMCCurrentModeEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )
- 11.41.2.13 GetMCCurrentModeShortCommand()** RoboCurrentModeEnumNet GetMCCurrentModeShort←  
Command (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )
- 11.41.2.14 GetMCCurrentPosition()** int GetMCCurrentPosition (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )
- 11.41.2.15 GetMCCurrentShortCommand()** short GetMCCurrentShortCommand (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )
- 11.41.2.16 GetMCCurrentSpeed()** short GetMCCurrentSpeed (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )
- 11.41.2.17 GetMCMaxAcceleration()** unsigned short GetMCMaxAcceleration (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )
- 11.41.2.18 GetMCMaxAccelerationEeprom()** unsigned short GetMCMaxAccelerationEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )

**11.41.2.19 GetMCMaxCurrent()** short GetMCMaxCurrent (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

**11.41.2.20 GetMCMaxCurrentEeprom()** short GetMCMaxCurrentEeprom (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

**11.41.2.21 GetMCMaxSpeed()** unsigned short GetMCMaxSpeed (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

**11.41.2.22 GetMCMaxSpeedEeprom()** unsigned short GetMCMaxSpeedEeprom (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

**11.41.2.23 GetMCMaxTravel()** int GetMCMaxTravel (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

**11.41.2.24 GetMCMaxTravelEeprom()** int GetMCMaxTravelEeprom (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

**11.41.2.25 GetMCMaxTravelShortCommand()** int GetMCMaxTravelShortCommand (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

- 11.41.2.26 GetMCMovement()** unsigned short GetMCMovement (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )
- 11.41.2.27 GetMCNewPosition()** int GetMCNewPosition (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )
- 11.41.2.28 GetMCOutputOnOff()** bool GetMCOutputOnOff (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )
- 11.41.2.29 GetMCPhase()** unsigned short GetMCPhase (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )
- 11.41.2.30 GetMCPhaseOffset()** unsigned short GetMCPhaseOffset (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )
- 11.41.2.31 GetMCReference()** unsigned char GetMCReference (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 [System::Runtime::InteropServices::Out] unsigned char% *switch\_port* )
- 11.41.2.32 GetMCReferenceCurrent()** short GetMCReferenceCurrent (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )

**11.41.2.33 GetMCReferenceCurrentEeprom()** short GetMCReferenceCurrentEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )

**11.41.2.34 GetMCRegulatorGain()** short GetMCRegulatorGain (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )

**11.41.2.35 GetMCRegulatorGainEeprom()** short GetMCRegulatorGainEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )

**11.41.2.36 GetMCScalingFactor()** int GetMCScalingFactor (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )

**11.41.2.37 GetMCScalingFactorEeprom()** int GetMCScalingFactorEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )

**11.41.2.38 GetMCSpeed()** short GetMCSpeed (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )

**11.41.2.39 GetMCSpeedEeprom()** unsigned short GetMCSpeedEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )

**11.41.2.40 GetMCSpeedShortCommand()** short GetMCSpeedShortCommand (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )

**11.41.2.41 GetMCSpeedUnitEeprom()** int32\_t GetMCSpeedUnitEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )

**11.41.2.42 GetMCStandbyCurrent()** short GetMCStandbyCurrent (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )

**11.41.2.43 GetMCStandbyCurrentEeprom()** short GetMCStandbyCurrentEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )

**11.41.2.44 GetMCStandbyTime()** short GetMCStandbyTime (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )

**11.41.2.45 GetMCStandbyTimeEeprom()** short GetMCStandbyTimeEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )

**11.41.2.46 GetSubChannel()** unsigned short GetSubChannel (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )

**11.41.2.47 SetMCAcceleration()** void SetMCAcceleration (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis*,  
 unsigned short *acceleration* )

**11.41.2.48 SetMCAccelerationEeprom()** void SetMCAccelerationEeprom (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis*,  
 unsigned short *acceleration* )

**11.41.2.49 SetMCAccelerationShortCommand()** void SetMCAccelerationShortCommand (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis*,  
 unsigned short *acceleration* )

**11.41.2.50 SetMCAxisRevisionEeprom()** void SetMCAxisRevisionEeprom (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis*,  
 unsigned short *revision* )

**11.41.2.51 SetMCBreakCurrent()** void SetMCBreakCurrent (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis*,  
 short *current* )

**11.41.2.52 SetMCBreakCurrentEeprom()** void SetMCBreakCurrentEeprom (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis*,  
 short *current* )



**11.41.2.53 SetMCConfig()** void SetMCConfig (   
    unsigned char *busnumber*,   
    unsigned char *busaddress*,   
    unsigned char *axis*,   
    unsigned short *config* )

**11.41.2.54 SetMCConfigEeprom()** void SetMCConfigEeprom (   
    unsigned char *busnumber*,   
    unsigned char *busaddress*,   
    unsigned char *axis*,   
    unsigned short *config* )

**11.41.2.55 SetMCCurrent()** void SetMCCurrent (   
    unsigned char *busnumber*,   
    unsigned char *busaddress*,   
    unsigned char *axis*,   
    short *current* )

**11.41.2.56 SetMCCurrentEeprom()** void SetMCCurrentEeprom (   
    unsigned char *busnumber*,   
    unsigned char *busaddress*,   
    unsigned char *axis*,   
    short *current* )

**11.41.2.57 SetMCCurrentMode()** void SetMCCurrentMode (   
    unsigned char *busnumber*,   
    unsigned char *busaddress*,   
    unsigned char *axis*,   
    RoboCurrentModeEnumNet *mode* )

**11.41.2.58 SetMCCurrentModeEeprom()** void SetMCCurrentModeEeprom (   
    unsigned char *busnumber*,   
    unsigned char *busaddress*,   
    unsigned char *axis*,   
    RoboCurrentModeEnumNet *mode* )

**11.41.2.59 SetMCCurrentModeShortCommand()** void SetMCCurrentModeShortCommand (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 RoboCurrentModeEnumNet *mode* )

**11.41.2.60 SetMCCurrentPosition()** void SetMCCurrentPosition (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 int *position* )

**11.41.2.61 SetMCCurrentShortCommand()** void SetMCCurrentShortCommand (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 short *current* )

**11.41.2.62 SetMCMaxAcceleration()** void SetMCMaxAcceleration (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 unsigned short *acceleration* )

**11.41.2.63 SetMCMaxAccelerationEeprom()** void SetMCMaxAccelerationEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 unsigned short *acceleration* )

**11.41.2.64 SetMCMaxCurrent()** void SetMCMaxCurrent (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 short *current* )

**11.41.2.65 SetMCMaxCurrentEeprom()** void SetMCMaxCurrentEeprom (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis*,  
    short *current* )

**11.41.2.66 SetMCMaxSpeed()** void SetMCMaxSpeed (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis*,  
    unsigned short *speed* )

**11.41.2.67 SetMCMaxSpeedEeprom()** void SetMCMaxSpeedEeprom (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis*,  
    unsigned short *speed* )

**11.41.2.68 SetMCMaxTravel()** void SetMCMaxTravel (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis*,  
    int *travel* )

**11.41.2.69 SetMCMaxTravelEeprom()** void SetMCMaxTravelEeprom (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis*,  
    int *travel* )

**11.41.2.70 SetMCMaxTravelShortCommand()** void SetMCMaxTravelShortCommand (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis*,  
    int *travel* )

**11.41.2.71 SetMCNewPosition()** void SetMCNewPosition (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 int *position* )

**11.41.2.72 SetMCOutputOnOff()** void SetMCOutputOnOff (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 bool *OnOff\_status* )

**11.41.2.73 SetMCReference()** void SetMCReference (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 unsigned char *switch\_enable*,   
 unsigned char *switch\_polarity* )

**11.41.2.74 SetMCReferenceCurrent()** void SetMCReferenceCurrent (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 short *current* )

**11.41.2.75 SetMCReferenceCurrentEeprom()** void SetMCReferenceCurrentEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 short *current* )

**11.41.2.76 SetMCRegulatorGain()** void SetMCRegulatorGain (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 short *gain* )

**11.41.2.77 SetMCRegulatorGainEeprom()** void SetMCRegulatorGainEeprom (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis*,  
    short *gain* )

**11.41.2.78 SetMCRotation()** void SetMCRotation (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis*,  
    unsigned char *onoff* )

**11.41.2.79 SetMCScalingFactor()** void SetMCScalingFactor (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis*,  
    int *factor* )

**11.41.2.80 SetMCScalingFactorEeprom()** void SetMCScalingFactorEeprom (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis*,  
    int *factor* )

**11.41.2.81 SetMCSpeed()** void SetMCSpeed (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis*,  
    short *speed* )

**11.41.2.82 SetMCSpeedEeprom()** void SetMCSpeedEeprom (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis*,  
    unsigned short *speed* )

**11.41.2.83 SetMCSpeedShortCommand()** void SetMCSpeedShortCommand (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 short *speed* )

**11.41.2.84 SetMCSpeedUnitEeprom()** void SetMCSpeedUnitEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 int32\_t *speedunit* )

**11.41.2.85 SetMCStandbyCurrent()** void SetMCStandbyCurrent (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 short *percent* )

**11.41.2.86 SetMCStandbyCurrentEeprom()** void SetMCStandbyCurrentEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 short *percent* )

**11.41.2.87 SetMCStandbyTime()** void SetMCStandbyTime (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 short *t* )

**11.41.2.88 SetMCStandbyTimeEeprom()** void SetMCStandbyTimeEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 short *t* )

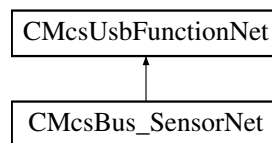
**11.41.2.89 SetSubChannel()** void SetSubChannel (   
     unsigned char *busnumber*,   
     unsigned char *busaddress*,   
     unsigned char *axis*,   
     unsigned short *subchannel* )

**11.41.2.90 StartMCMovement()** void StartMCMovement (   
     unsigned char *busnumber*,   
     unsigned char *busaddress*,   
     unsigned char *axis* )

**11.41.2.91 StopMCMovement()** void StopMCMovement (   
     unsigned char *busnumber*,   
     unsigned char *busaddress*,   
     unsigned char *axis* )

## 11.42 CMcsBus\_SensorNet Class Reference

Inheritance diagram for CMcsBus\_SensorNet:



### Public Member Functions

- `CMcsBus_SensorNet` (`CMcsUsbNet`<sup>^</sup> device)
- `~CMcsBus_SensorNet` (void)
- void `SetMinimalThreshold` (unsigned char busnumber, unsigned char busaddress, unsigned short threshold)
- unsigned short `GetMinimalThreshold` (unsigned char busnumber, unsigned char busaddress)
- void `SetDetectionThreshold` (unsigned char busnumber, unsigned char busaddress, unsigned short threshold)
- unsigned short `GetDetectionThreshold` (unsigned char busnumber, unsigned char busaddress)
- void `SetLatency` (unsigned char busnumber, unsigned char busaddress, unsigned short latency)
- unsigned short `GetLatency` (unsigned char busnumber, unsigned char busaddress)
- unsigned short `GetBubbleStatus` (unsigned char busnumber, unsigned char busaddress)
- unsigned short `GetLatencyCounter` (unsigned char busnumber, unsigned char busaddress)
- unsigned short `GetDetectorValue` (unsigned char busnumber, unsigned char busaddress)
- array< int > <sup>^</sup> `GetPressure` (unsigned char busnumber, unsigned char busaddress, int n)
- int `GetPressure` (unsigned char busnumber, unsigned char busaddress, unsigned short index)
- void `SetRegulatorOnOff` (unsigned char busnumber, unsigned char busaddress, unsigned short index, unsigned char onoff)
- unsigned char `GetRegulatorOnOff` (unsigned char busnumber, unsigned char busaddress, unsigned short index)

- void [SetSollPressure](#) (unsigned char busnumber, unsigned char busaddress, unsigned short index, int pressure)
- int [GetSollPressure](#) (unsigned char busnumber, unsigned char busaddress, unsigned short index)
- void [SetRegulatorFactor](#) (unsigned char busnumber, unsigned char busaddress, unsigned short index, int factor)
- int [GetRegulatorFactor](#) (unsigned char busnumber, unsigned char busaddress, unsigned short index)
- void [SetPressureOffset](#) (unsigned char busnumber, unsigned char busaddress, unsigned short index)
- array< unsigned short > ^ [GetPressureOffset](#) (unsigned char busnumber, unsigned char busaddress)
- int [GetPressureOffset](#) (unsigned char busnumber, unsigned char busaddress, unsigned short index)
- unsigned int [GetRegulatorStatus](#) (unsigned char busnumber, unsigned char busaddress, unsigned short index)
- void [SetRotatePump](#) (unsigned char busnumber, unsigned char busaddress, unsigned short index, short speed)
- short [GetRotatePump](#) (unsigned char busnumber, unsigned char busaddress, unsigned short index)
- void [SetMovePump](#) (unsigned char busnumber, unsigned char busaddress, unsigned short index, unsigned short speed, int position)
- void [GetMovePump](#) (unsigned char busnumber, unsigned char busaddress, unsigned short index, [System::Runtime::InteropServices::Out]unsigned short% speed, [System::Runtime::InteropServices::Out]int% position)
- void [SetRegulationTimeouts](#) (unsigned char busnumber, unsigned char busaddress, unsigned short MaxSpeedWait, unsigned short MaxSignChange)
- void [GetRegulationTimeouts](#) (unsigned char busnumber, unsigned char busaddress, [System::Runtime::InteropServices::Out]unsigned short% MaxSpeedWait, [System::Runtime::InteropServices::Out]unsigned short% MaxSignChange)
- array< int > ^ [Get4ADC](#) (unsigned char busnumber, unsigned char busaddress)
- array< int > ^ [Get4ADCAverage](#) (unsigned char busnumber, unsigned char busaddress)
- void [Set4DAC](#) (unsigned char busnumber, unsigned char busaddress, array< unsigned short > ^ dac)
- array< unsigned short > ^ [Get4DAC](#) (unsigned char busnumber, unsigned char busaddress)
- void [Set4ADCMode](#) (unsigned char busnumber, unsigned char busaddress, PatchServAdcModeEnumNet mode)
- PatchServAdcModeEnumNet [Get4ADCMode](#) (unsigned char busnumber, unsigned char busaddress)
- void [Set4ADCCatchampAverageShift](#) (unsigned char busnumber, unsigned char busaddress, unsigned int shift)
- unsigned int [Get4ADCCatchampAverageShift](#) (unsigned char busnumber, unsigned char busaddress)
- array< unsigned short > ^ [Get2AnalogInput](#) (unsigned char busnumber, unsigned char busaddress)
- unsigned short [Get2DigitalInput](#) (unsigned char busnumber, unsigned char busaddress)
- array< unsigned short > ^ [GetADCs](#) (unsigned char busnumber, unsigned char busaddress, int n)
- array< unsigned short > ^ [GetADCsLoop](#) (unsigned char busnumber, unsigned char busaddress, int n)
- void [SetPiezoState](#) (unsigned char busnumber, unsigned char busaddress, int state)
- void [GetPiezoState](#) (unsigned char busnumber, unsigned char busaddress, [System::Runtime::InteropServices::Out]int% state, [System::Runtime::InteropServices::Out]int% reason)
- void [SetDACs](#) (unsigned char busnumber, unsigned char busaddress, unsigned short index, array< unsigned short > ^ dac\_times\_voltages)
- array< unsigned short > ^ [GetDACs](#) (unsigned char busnumber, unsigned char busaddress, unsigned short index)
- void [SetSamplePeriode](#) (unsigned char busnumber, unsigned char busaddress, unsigned short periode)
- unsigned short [GetSamplePeriode](#) (unsigned char busnumber, unsigned char busaddress)
- void [StartSync](#) (unsigned char busnumber, unsigned char busaddress)
- unsigned short [GetSyncState](#) (unsigned char busnumber, unsigned char busaddress)
- void [CatchAmpSetDacAmplitude](#) (unsigned char busnumber, unsigned char busaddress, unsigned short dacAmplitude)
- unsigned short [CatchAmpGetDacAmplitude](#) (unsigned char busnumber, unsigned char busaddress)
- void [CatchAmpSetDacOffset](#) (unsigned char busnumber, unsigned char busaddress, short dacOffset)
- short [CatchAmpGetDacOffset](#) (unsigned char busnumber, unsigned char busaddress)
- int [CatchAmpGetAdcMean](#) (unsigned char busnumber, unsigned char busaddress)
- int [CatchAmpGetAdcValue](#) (unsigned char busnumber, unsigned char busaddress)



- int [CatchAmpGetAdcValueH](#) (unsigned char busnumber, unsigned char busaddress)
- int [CatchAmpGetAdcValueL](#) (unsigned char busnumber, unsigned char busaddress)
- void [CatchAmpSetPwmEnable](#) (unsigned char busnumber, unsigned char busaddress, bool pwmEnable)
- bool [CatchAmpGetPwmEnable](#) (unsigned char busnumber, unsigned char busaddress)
- void [CatchAmpSetDacEnable](#) (unsigned char busnumber, unsigned char busaddress, bool dacEnable)
- bool [CatchAmpGetDacEnable](#) (unsigned char busnumber, unsigned char busaddress)
- int [TactSwitchGetState](#) (unsigned char busnumber, unsigned char busaddress)
- void [TactSwitchSetDisplay](#) (unsigned char busnumber, unsigned char busaddress, int Melody)

## Additional Inherited Members

### 11.42.1 Constructor & Destructor Documentation

**11.42.1.1 CMcsBus\_SensorNet()** [CMcsBus\\_SensorNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> *device* )

**11.42.1.2 ~CMcsBus\_SensorNet()** [~CMcsBus\\_SensorNet](#) (  
void )

### 11.42.2 Member Function Documentation

**11.42.2.1 CatchAmpGetAdcMean()** int [CatchAmpGetAdcMean](#) (  
unsigned char *busnumber*,  
unsigned char *busaddress* )

**11.42.2.2 CatchAmpGetAdcValue()** int [CatchAmpGetAdcValue](#) (  
unsigned char *busnumber*,  
unsigned char *busaddress* )

**11.42.2.3 CatchAmpGetAdcValueH()** int [CatchAmpGetAdcValueH](#) (  
unsigned char *busnumber*,  
unsigned char *busaddress* )

**11.42.2.4 CatchAmpGetAdcValueL()** int CatchAmpGetAdcValueL (  
    unsigned char *busnumber*,  
    unsigned char *busaddress* )

**11.42.2.5 CatchAmpGetDacAmplitude()** unsigned short CatchAmpGetDacAmplitude (  
    unsigned char *busnumber*,  
    unsigned char *busaddress* )

**11.42.2.6 CatchAmpGetDacEnable()** bool CatchAmpGetDacEnable (  
    unsigned char *busnumber*,  
    unsigned char *busaddress* )

**11.42.2.7 CatchAmpGetDacOffset()** short CatchAmpGetDacOffset (  
    unsigned char *busnumber*,  
    unsigned char *busaddress* )

**11.42.2.8 CatchAmpGetPwmEnable()** bool CatchAmpGetPwmEnable (  
    unsigned char *busnumber*,  
    unsigned char *busaddress* )

**11.42.2.9 CatchAmpSetDacAmplitude()** void CatchAmpSetDacAmplitude (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned short *dacAmplitude* )

**11.42.2.10 CatchAmpSetDacEnable()** void CatchAmpSetDacEnable (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    bool *dacEnable* )

**11.42.2.11 CatchAmpSetDacOffset()** void CatchAmpSetDacOffset (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    short *dacOffset* )

**11.42.2.12 CatchAmpSetPwmEnable()** void CatchAmpSetPwmEnable (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    bool *pwmEnable* )

**11.42.2.13 Get2AnalogInput()** array<unsigned short> ^ Get2AnalogInput (   
    unsigned char *busnumber*,  
    unsigned char *busaddress* )

**11.42.2.14 Get2DigitalInput()** unsigned short Get2DigitalInput (   
    unsigned char *busnumber*,  
    unsigned char *busaddress* )

**11.42.2.15 Get4ADC()** array<int> ^ Get4ADC (   
    unsigned char *busnumber*,  
    unsigned char *busaddress* )

**11.42.2.16 Get4ADCAverage()** array<int> ^ Get4ADCAverage (   
    unsigned char *busnumber*,  
    unsigned char *busaddress* )

**11.42.2.17 Get4ADCCatchampAverageShift()** unsigned int Get4ADCCatchampAverageShift (   
    unsigned char *busnumber*,  
    unsigned char *busaddress* )

**11.42.2.18 Get4ADCMode()** PatchServAdcModeEnumNet Get4ADCMode (   
    unsigned char *busnumber*,  
    unsigned char *busaddress* )

**11.42.2.19 Get4DAC()** array<unsigned short> ^ Get4DAC (   
    unsigned char *busnumber*,  
    unsigned char *busaddress* )

**11.42.2.20 GetADCs()** array<unsigned short> ^ GetADCs (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 int *n* )

**11.42.2.21 GetADCsLoop()** array<unsigned short> ^ GetADCsLoop (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 int *n* )

**11.42.2.22 GetBubbleStatus()** unsigned short GetBubbleStatus (   
 unsigned char *busnumber*,   
 unsigned char *busaddress* )

**11.42.2.23 GetDACs()** array<unsigned short> ^ GetDACs (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned short *index* )

**11.42.2.24 GetDetectionThreshold()** unsigned short GetDetectionThreshold (   
 unsigned char *busnumber*,   
 unsigned char *busaddress* )

**11.42.2.25 GetDetectorValue()** unsigned short GetDetectorValue (   
 unsigned char *busnumber*,   
 unsigned char *busaddress* )

**11.42.2.26 GetLatency()** unsigned short GetLatency (   
 unsigned char *busnumber*,   
 unsigned char *busaddress* )

**11.42.2.27 GetLatencyCounter()** unsigned short GetLatencyCounter (   
 unsigned char *busnumber*,   
 unsigned char *busaddress* )

**11.42.2.28 GetMinimalThreshold()** unsigned short GetMinimalThreshold (   
 unsigned char *busnumber*,   
 unsigned char *busaddress* )

**11.42.2.29 GetMovePump()** void GetMovePump (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned short *index*,   
 [System::Runtime::InteropServices::Out] unsigned short% *speed*,   
 [System::Runtime::InteropServices::Out] int% *position* )

**11.42.2.30 GetPiezoState()** void GetPiezoState (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 [System::Runtime::InteropServices::Out] int% *state*,   
 [System::Runtime::InteropServices::Out] int% *reason* )

**11.42.2.31 GetPressure()** [1/2] array<int> ^ GetPressure (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 int *n* )

**11.42.2.32 GetPressure()** [2/2] int GetPressure (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned short *index* )

**11.42.2.33 GetPressureOffset()** [1/2] array<unsigned short> ^ GetPressureOffset (   
 unsigned char *busnumber*,   
 unsigned char *busaddress* )

**11.42.2.34 GetPressureOffset()** [2/2] int GetPressureOffset (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned short *index* )

**11.42.2.35 GetRegulationTimeouts()** void GetRegulationTimeouts (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 [System::Runtime::InteropServices::Out] unsigned short% *MaxSpeedWait*,   
 [System::Runtime::InteropServices::Out] unsigned short% *MaxSignChange* )

**11.42.2.36 GetRegulatorFactor()** int GetRegulatorFactor (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned short *index* )

**11.42.2.37 GetRegulatorOnOff()** unsigned char GetRegulatorOnOff (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned short *index* )

**11.42.2.38 GetRegulatorStatus()** unsigned int GetRegulatorStatus (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned short *index* )

**11.42.2.39 GetRotatePump()** short GetRotatePump (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned short *index* )

**11.42.2.40 GetSamplePeriode()** unsigned short GetSamplePeriode (   
 unsigned char *busnumber*,   
 unsigned char *busaddress* )

**11.42.2.41 GetSollPressure()** int GetSollPressure (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned short *index* )

**11.42.2.42 GetSyncState()** unsigned short GetSyncState (   
 unsigned char *busnumber*,   
 unsigned char *busaddress* )

**11.42.2.43 Set4ADCCatchampAverageShift()** void Set4ADCCatchampAverageShift (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned int *shift* )

**11.42.2.44 Set4ADCMode()** void Set4ADCMode (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 PatchServAdcModeEnumNet *mode* )

**11.42.2.45 Set4DAC()** void Set4DAC (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 array< unsigned short >^ *dac* )

**11.42.2.46 SetDACs()** void SetDACs (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned short *index*,   
 array< unsigned short >^ *dac\_times\_voltages* )

**11.42.2.47 SetDetectionThreshold()** void SetDetectionThreshold (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned short *threshold* )

**11.42.2.48 SetLatency()** void SetLatency (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned short *latency* )

**11.42.2.49 SetMinimalThreshold()** void SetMinimalThreshold (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned short *threshold* )

**11.42.2.50 SetMovePump()** void SetMovePump (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned short *index*,  
    unsigned short *speed*,  
    int *position* )

**11.42.2.51 SetPiezoState()** void SetPiezoState (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    int *state* )

**11.42.2.52 SetPressureOffset()** void SetPressureOffset (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned short *index* )

**11.42.2.53 SetRegulationTimeouts()** void SetRegulationTimeouts (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned short *MaxSpeedWait*,  
    unsigned short *MaxSignChange* )

**11.42.2.54 SetRegulatorFactor()** void SetRegulatorFactor (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned short *index*,  
    int *factor* )

**11.42.2.55 SetRegulatorOnOff()** void SetRegulatorOnOff (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned short *index*,  
    unsigned char *onoff* )



**11.42.2.56 SetRotatePump()** void SetRotatePump (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned short *index*,   
 short *speed* )

**11.42.2.57 SetSamplePeriode()** void SetSamplePeriode (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned short *periode* )

**11.42.2.58 SetSollPressure()** void SetSollPressure (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned short *index*,   
 int *pressure* )

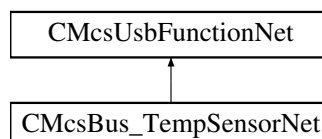
**11.42.2.59 StartSync()** void StartSync (   
 unsigned char *busnumber*,   
 unsigned char *busaddress* )

**11.42.2.60 TactSwitchGetState()** int TactSwitchGetState (   
 unsigned char *busnumber*,   
 unsigned char *busaddress* )

**11.42.2.61 TactSwitchSetDisplay()** void TactSwitchSetDisplay (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 int *Melody* )

## 11.43 CMcsBus\_TempSensorNet Class Reference

Inheritance diagram for CMcsBus\_TempSensorNet:



**Public Member Functions**

- [CMcsBus\\_TempSensorNet](#) ([CMcsUsbNet](#)<sup>^</sup> device)
- [~CMcsBus\\_TempSensorNet](#) (void)
- short [GetTemperatur](#) (unsigned char busnumber, unsigned char busaddress)
- short [GetTemperatur](#) (unsigned char busnumber, unsigned char busaddress, short index)
- void [SetNanoVoltsPerKelvin](#) (unsigned char busnumber, unsigned char busaddress, int nanovoltsperkelvin)
- int [GetNanoVoltsPerKelvin](#) (unsigned char busnumber, unsigned char busaddress)
- short [GetThermoVoltage](#) (unsigned char busnumber, unsigned char busaddress, short index)
- short [GetThermoTemp](#) (unsigned char busnumber, unsigned char busaddress, short index)
- void [SetThermoOffset](#) (unsigned char busnumber, unsigned char busaddress, short index, short offset)
- short [GetThermoOffset](#) (unsigned char busnumber, unsigned char busaddress, short index)

**Additional Inherited Members****11.43.1 Constructor & Destructor Documentation**

**11.43.1.1 CMcsBus\_TempSensorNet()** [CMcsBus\\_TempSensorNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> device )

**11.43.1.2 ~CMcsBus\_TempSensorNet()** [~CMcsBus\\_TempSensorNet](#) (  
void )

**11.43.2 Member Function Documentation**

**11.43.2.1 GetNanoVoltsPerKelvin()** int [GetNanoVoltsPerKelvin](#) (  
unsigned char *busnumber*,  
unsigned char *busaddress* )

**11.43.2.2 GetTemperatur()** [1/2] short [GetTemperatur](#) (  
unsigned char *busnumber*,  
unsigned char *busaddress* )

**11.43.2.3 GetTemperatur()** [2/2] short [GetTemperatur](#) (  
unsigned char *busnumber*,  
unsigned char *busaddress*,  
short *index* )

**11.43.2.4 GetThermoOffset()** short GetThermoOffset (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 short *index* )

**11.43.2.5 GetThermoTemp()** short GetThermoTemp (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 short *index* )

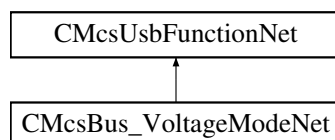
**11.43.2.6 GetThermoVoltage()** short GetThermoVoltage (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 short *index* )

**11.43.2.7 SetNanoVoltsPerKelvin()** void SetNanoVoltsPerKelvin (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 int *nanovoltsperkelvin* )

**11.43.2.8 SetThermoOffset()** void SetThermoOffset (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 short *index*,   
 short *offset* )

## 11.44 CMcsBus\_VoltageModeNet Class Reference

Inheritance diagram for CMcsBus\_VoltageModeNet:



**Public Member Functions**

- [CMcsBus\\_VoltageModeNet](#) ([CMcsUsbNet](#)<sup>^</sup> device)
- [~CMcsBus\\_VoltageModeNet](#) (void)
- void [SetVMMMaxPositiveCurrentEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel, short current)
- short [GetVMMMaxPositiveCurrentEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel)
- void [SetVMMMaxPositiveCurrent](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel, short current)
- short [GetVMMMaxPositiveCurrent](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel)
- void [SetVMMMaxNegativeCurrentEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel, short current)
- short [GetVMMMaxNegativeCurrentEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel)
- void [SetVMMMaxNegativeCurrent](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel, short current)
- short [GetVMMMaxNegativeCurrent](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel)
- void [SetVMMMaxPositiveVoltageEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel, short voltage)
- short [GetVMMMaxPositiveVoltageEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel)
- void [SetVMMMaxPositiveVoltage](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel, short voltage)
- short [GetVMMMaxPositiveVoltage](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel)
- void [SetVMMMaxNegativeVoltageEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel, short voltage)
- short [GetVMMMaxNegativeVoltageEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel)
- void [SetVMMMaxNegativeVoltage](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel, short voltage)
- short [GetVMMMaxNegativeVoltage](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel)
- void [SetVMOOutputOnOff](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel, unsigned short status)
- unsigned short [GetVMOOutputOnOff](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel)
- void [SetVMVoltage](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel, short voltage)
- short [GetVMVoltage](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel)

**Additional Inherited Members****11.44.1 Constructor & Destructor Documentation**
**11.44.1.1 CMcsBus\_VoltageModeNet()** [CMcsBus\\_VoltageModeNet](#) ( [CMcsUsbNet](#)<sup>^</sup> device )

**11.44.1.2** `~CMcsBus_VoltageModeNet()` `~CMcsBus_VoltageModeNet` (  
    void )

## 11.44.2 Member Function Documentation

**11.44.2.1** `GetVMMaxNegativeCurrent()` short `GetVMMaxNegativeCurrent` (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *channel* )

**11.44.2.2** `GetVMMaxNegativeCurrentEeprom()` short `GetVMMaxNegativeCurrentEeprom` (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *channel* )

**11.44.2.3** `GetVMMaxNegativeVoltage()` short `GetVMMaxNegativeVoltage` (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *channel* )

**11.44.2.4** `GetVMMaxNegativeVoltageEeprom()` short `GetVMMaxNegativeVoltageEeprom` (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *channel* )

**11.44.2.5** `GetVMMaxPositiveCurrent()` short `GetVMMaxPositiveCurrent` (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *channel* )

**11.44.2.6** `GetVMMaxPositiveCurrentEeprom()` short `GetVMMaxPositiveCurrentEeprom` (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *channel* )

**11.44.2.7 GetVMMaxPositiveVoltage()** short GetVMMaxPositiveVoltage (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *channel* )

**11.44.2.8 GetVMMaxPositiveVoltageEeprom()** short GetVMMaxPositiveVoltageEeprom (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *channel* )

**11.44.2.9 GetVMOutputOnOff()** unsigned short GetVMOutputOnOff (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *channel* )

**11.44.2.10 GetVMVoltage()** short GetVMVoltage (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *channel* )

**11.44.2.11 SetVMMaxNegativeCurrent()** void SetVMMaxNegativeCurrent (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *channel*,  
 short *current* )

**11.44.2.12 SetVMMaxNegativeCurrentEeprom()** void SetVMMaxNegativeCurrentEeprom (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *channel*,  
 short *current* )

**11.44.2.13 SetVMMaxNegativeVoltage()** void SetVMMaxNegativeVoltage (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *channel*,  
 short *voltage* )

**11.44.2.14 SetVMMaxNegativeVoltageEeprom()** void SetVMMaxNegativeVoltageEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *channel*,   
 short *voltage* )

**11.44.2.15 SetVMMaxPositiveCurrent()** void SetVMMaxPositiveCurrent (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *channel*,   
 short *current* )

**11.44.2.16 SetVMMaxPositiveCurrentEeprom()** void SetVMMaxPositiveCurrentEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *channel*,   
 short *current* )

**11.44.2.17 SetVMMaxPositiveVoltage()** void SetVMMaxPositiveVoltage (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *channel*,   
 short *voltage* )

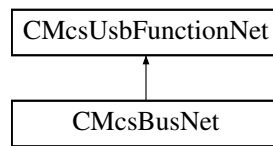
**11.44.2.18 SetVMMaxPositiveVoltageEeprom()** void SetVMMaxPositiveVoltageEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *channel*,   
 short *voltage* )

**11.44.2.19 SetVMOutputOnOff()** void SetVMOutputOnOff (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *channel*,   
 unsigned short *status* )

**11.44.2.20 SetVMVoltage()** void SetVMVoltage (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *channel*,   
 short *voltage* )

## 11.45 CMcsBusNet Class Reference

Inheritance diagram for CMcsBusNet:



### Public Member Functions

- [CMcsBusNet](#) ([CMcsUsbNet](#)<sup>^</sup> device)
- virtual [~CMcsBusNet](#) (void)
- void [SetCommand](#) (unsigned char command, unsigned char busnumber, unsigned char busaddress, unsigned char axis, unsigned short value)
- void [SetCommand](#) (unsigned char command, unsigned char busnumber, unsigned char busaddress, unsigned char axis, short value)
- void [SetCommand](#) (unsigned char command, unsigned char busnumber, unsigned char busaddress, unsigned char axis, unsigned int value)
- void [SetCommand](#) (unsigned char command, unsigned char busnumber, unsigned char busaddress, unsigned char axis, int value)
- void [GetCommand](#) (unsigned char command, unsigned char busnumber, unsigned char busaddress, unsigned char axis, [System::Runtime::InteropServices::Out]unsigned short% value)
- void [GetCommand](#) (unsigned char command, unsigned char busnumber, unsigned char busaddress, unsigned char axis, [System::Runtime::InteropServices::Out]short% value)
- void [GetCommand](#) (unsigned char command, unsigned char busnumber, unsigned char busaddress, unsigned char axis, [System::Runtime::InteropServices::Out]unsigned int% value)
- void [GetCommand](#) (unsigned char command, unsigned char busnumber, unsigned char busaddress, unsigned char axis, [System::Runtime::InteropServices::Out]int% value)
- void [SetBusAddressEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned short newaddress)
- unsigned short [GetBusAddressEeprom](#) (unsigned char busnumber, unsigned char busaddress)
- void [SetBusAddress](#) (unsigned char busnumber, unsigned char busaddress, unsigned short newaddress)
- unsigned short [GetBusAddress](#) (unsigned char busnumber, unsigned char busaddress)
- void [CMcsBusNet::SetModeEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned short mode)
- unsigned short [CMcsBusNet::GetModeEeprom](#) (unsigned char busnumber, unsigned char busaddress)
- void [CMcsBusNet::SetMode](#) (unsigned char busnumber, unsigned char busaddress, unsigned short mode)
- unsigned short [CMcsBusNet::GetMode](#) (unsigned char busnumber, unsigned char busaddress)
- void [SetHWRevisionEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned short revision)
- unsigned short [GetHWRevisionEeprom](#) (unsigned char busnumber, unsigned char busaddress)

### Additional Inherited Members

#### 11.45.1 Constructor & Destructor Documentation

##### 11.45.1.1 CMcsBusNet() [CMcsBusNet](#) ( [CMcsUsbNet](#)<sup>^</sup> device )



**11.45.1.2** `~CMcsBusNet()` virtual `~CMcsBusNet` (  
void ) [virtual]

## 11.45.2 Member Function Documentation

**11.45.2.1** `CMcsBusNet::GetMode()` unsigned short `CMcsBusNet::GetMode` (  
unsigned char *busnumber*,  
unsigned char *busaddress* )

**11.45.2.2** `CMcsBusNet::GetModeEeprom()` unsigned short `CMcsBusNet::GetModeEeprom` (  
unsigned char *busnumber*,  
unsigned char *busaddress* )

**11.45.2.3** `CMcsBusNet::SetMode()` void `CMcsBusNet::SetMode` (  
unsigned char *busnumber*,  
unsigned char *busaddress*,  
unsigned short *mode* )

**11.45.2.4** `CMcsBusNet::SetModeEeprom()` void `CMcsBusNet::SetModeEeprom` (  
unsigned char *busnumber*,  
unsigned char *busaddress*,  
unsigned short *mode* )

**11.45.2.5** `GetBusAddress()` unsigned short `GetBusAddress` (  
unsigned char *busnumber*,  
unsigned char *busaddress* )

**11.45.2.6** `GetBusAddressEeprom()` unsigned short `GetBusAddressEeprom` (  
unsigned char *busnumber*,  
unsigned char *busaddress* )

**11.45.2.7 GetCommand() [1/4]** void GetCommand (   
    unsigned char *command*,  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis*,  
    [System::Runtime::InteropServices::Out] int% *value* )

**11.45.2.8 GetCommand() [2/4]** void GetCommand (   
    unsigned char *command*,  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis*,  
    [System::Runtime::InteropServices::Out] short% *value* )

**11.45.2.9 GetCommand() [3/4]** void GetCommand (   
    unsigned char *command*,  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis*,  
    [System::Runtime::InteropServices::Out] unsigned int% *value* )

**11.45.2.10 GetCommand() [4/4]** void GetCommand (   
    unsigned char *command*,  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis*,  
    [System::Runtime::InteropServices::Out] unsigned short% *value* )

**11.45.2.11 GetHWRevisionEeprom()** unsigned short GetHWRevisionEeprom (   
    unsigned char *busnumber*,  
    unsigned char *busaddress* )

**11.45.2.12 SetBusAddress()** void SetBusAddress (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned short *newaddress* )

**11.45.2.13 SetBusAddressEeprom()** void SetBusAddressEeprom (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned short *newaddress* )

**11.45.2.14 SetCommand() [1/4]** void SetCommand (   
    unsigned char *command*,  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis*,  
    int *value* )

**11.45.2.15 SetCommand() [2/4]** void SetCommand (   
    unsigned char *command*,  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis*,  
    short *value* )

**11.45.2.16 SetCommand() [3/4]** void SetCommand (   
    unsigned char *command*,  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis*,  
    unsigned int *value* )

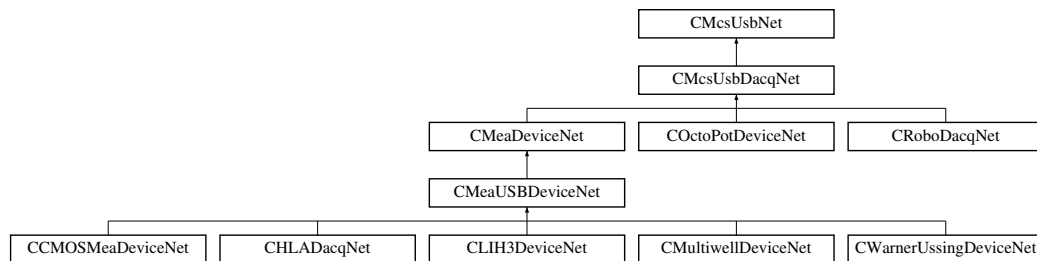
**11.45.2.17 SetCommand() [4/4]** void SetCommand (   
    unsigned char *command*,  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis*,  
    unsigned short *value* )

**11.45.2.18 SetHWRevisionEeprom()** void SetHWRevisionEeprom (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned short *revision* )

## 11.46 CMcsUsbDacqNet Class Reference

Base class for data acquisition devices.

Inheritance diagram for CMcsUsbDacqNet:



### Classes

- class [CHWInfo](#)  
*Class to provide hardware information about the device.*

### Public Member Functions

- [CMcsUsbDacqNet](#) ()
- [~CMcsUsbDacqNet](#) ()
- virtual uint32\_t [GetVoltageRangeIndex](#) (unsigned int virtualDevice)
- virtual void [SetVoltageRangeByIndex](#) (int32\_t voltageRangeIndex, unsigned int virtualDevice)  
*Sets the voltage range on devices which support multiple voltage ranges.*
- virtual void [SetVoltageRangeInMicroVolt](#) (int32\_t voltageRange, unsigned int virtualDevice)  
*Sets the voltage range on devices which support multiple voltage ranges.*
- virtual int32\_t [GetVoltageRangeInMicroVolt](#) (unsigned int virtualDevice)  
*Gets the currently selected voltage range on devices which support multiple voltage ranges.*
- virtual int32\_t [GetVoltageRangeInMilliVolt](#) ()  
*Gets the currently selected voltage range on devices which support multiple voltage ranges.*
- virtual void [SetDataMode](#) (DataModeEnumNet dataMode, unsigned int virtualDevice)  
*Sets the data mode, can be 16, 24 or 32bit, all signed or unsigned on the MEA2100 device.*
- virtual DataModeEnumNet [GetDataMode](#) (unsigned int virtualDevice)  
*Gets the data mode, can be 16, 24 or 32bit, all signed or unsigned on the MEA2100 device.*
- void [SetDigitalSource](#) (DigitalTargetEnumNet digitaltarget, int32\_t NrChannel, DigitalSourceEnumNet source, int bitnumber\_offset)  
*Sets the function/source of an digital output bit.*
- void [SetDigitalSource](#) (DigitalTargetEnumNet digitaltarget, int32\_t NrChannel, W2100DigitalSourceEnumNet source, int bitnumber\_offset)  
*Sets the function/source of an digital output bit.*
- void [SetDigitalSource](#) (DigitalTargetEnumNet digitaltarget, int32\_t NrChannel, SCUDigitalSourceEnumNet source, int bitnumber\_offset)  
*Sets the function/source of an digital output bit.*
- void [SetDigitalSource](#) (DigitalTargetEnumNet digitaltarget, int32\_t NrChannel, MEA2100\_256DigitalSourceEnumNet source, int bitnumber\_offset)  
*Sets the function/source of an digital output bit.*

- `template<typename digitalsourceenum >`  
`void SetDigitalSource (DigitalTargetEnumNet digitaltarget, int32_t NrChannel, DigitalSource< digital-`  
`sourceenum >^ source, int bitnumber_offset)`  
*Sets the function/source of an digital output bit.*
- `void GetDigitalSource (DigitalTargetEnumNet digitaltarget, int32_t NrChannel, [System::Runtime::Interop-`  
`Services::Out]DigitalSourceEnumNet% source, [System::Runtime::InteropServices::Out]int% bitnumber_`  
`offset)`  
*Gets the function/source of an digital output bit.*
- `void GetDigitalSource (DigitalTargetEnumNet digitaltarget, int32_t NrChannel, [System::Runtime::-`  
`InteropServices::Out]W2100DigitalSourceEnumNet% source, [System::Runtime::InteropServices::Out]int%`  
`bitnumber_offset)`  
*Gets the function/source of an digital output bit.*
- `void GetDigitalSource (DigitalTargetEnumNet digitaltarget, int32_t NrChannel, [System::Runtime::-`  
`InteropServices::Out]SCUDigitalSourceEnumNet% source, [System::Runtime::InteropServices::Out]int%`  
`bitnumber_offset)`  
*Gets the function/source of an digital output bit.*
- `void GetDigitalSource (DigitalTargetEnumNet digitaltarget, int32_t NrChannel, [System::Runtime::Interop-`  
`Services::Out]MEA2100_256DigitalSourceEnumNet% source, [System::Runtime::InteropServices::Out]int%`  
`bitnumber_offset)`  
*Gets the function/source of an digital output bit.*
- `template<typename digitalsourceenum >`  
`void GetDigitalSource (DigitalTargetEnumNet digitaltarget, int32_t NrChannel, [System::Runtime::Interop-`  
`Services::Out]DigitalSource< digitalsourceenum >^% source, [System::Runtime::InteropServices::Out]int%`  
`bitnumber_offset)`  
*Gets the function/source of an digital output bit.*
- `virtual AdapterTypeEnumNet GetAdapterType ()`  
*Gets the adapter which is connected to the MEA2100 device.*
- `virtual MeaLayoutEnumNet GetMeaLayout ()`  
*Gets the MEA layout which is connected to the MEA2100 device.*
- `virtual uint32_t GetAdcDataFormat (uint32_t virtualDevice)`  
*Gets the ADC data format, 16 means 16 bits, 24 means 24 bits, 32 means 32 bits.*
- `virtual uint32_t GetAnalogValueUnit (uint32_t virtualDevice, DacqGroupChannelEnumNet group, [System-`  
`::Runtime::InteropServices::Out] AnalogUnitEnumNet% unit)`
- `virtual uint32_t GetResolutionPerDigit (uint32_t virtualDevice, DacqGroupChannelEnumNet group,`  
`[System::Runtime::InteropServices::Out] int% res, [System::Runtime::InteropServices::Out] int% resUnit)`
- `virtual uint32_t GetAdcZero (uint32_t virtualDevice, DacqGroupChannelEnumNet group, [System::Runtime-`  
`::InteropServices::Out] int% adcz)`
- `virtual uint32_t GetHardwareMinRange (uint32_t virtualDevice, DacqGroupChannelEnumNet group,`  
`[System::Runtime::InteropServices::Out] int% r, [System::Runtime::InteropServices::Out] int% rUnit)`
- `virtual uint32_t GetHardwareMaxRange (uint32_t virtualDevice, DacqGroupChannelEnumNet group,`  
`[System::Runtime::InteropServices::Out] int% r, [System::Runtime::InteropServices::Out] int% rUnit)`
- `virtual uint32_t GetDataFormat (uint32_t virtualDevice, DacqGroupChannelEnumNet group, [System::-`  
`Runtime::InteropServices::Out] int% numberOfBits)`
- `virtual uint32_t GetNumberOfDataBits (uint32_t virtualDevice, DacqGroupChannelEnumNet group,`  
`[System::Runtime::InteropServices::Out] int% numberOfBits)`  
*Get the real number of data bits.*
- `virtual void SetSamplerate (int32_t rate, unsigned int oversample, unsigned int virtualDevice)`  
*Sets the sampling frequency of the device.*
- `virtual int32_t GetSamplerate (unsigned int virtualDevice)`  
*Gets the sampling frequency of the device.*
- `virtual uint32_t GetMaxSamplingFrequency (int virtualDevice)`  
*Gets the maximal sampling frequency of the device.*
- `virtual uint32_t GetMinSamplingFrequencyStepsize ()`  
*Gets the minimal sampling frequency step size increment value of the device.*

- virtual `int32_t` [GetChannelsInBlock](#) (unsigned int virtualDevice)  
*Get the number of 16 bit datawords which will be collected per sample frame, use after the device is configured.*
- virtual `uint32_t` [GetChannelLayout](#) ([System::Runtime::InteropServices::Out]int% AnalogChannels, [System::Runtime::InteropServices::Out]int% DigitalChannels, [System::Runtime::InteropServices::Out]int% ChecksumChannels, [System::Runtime::InteropServices::Out]int% TimestampChannels, [System::Runtime::InteropServices::Out]int% ChannelsInBlock, unsigned int virtualDevice)
- virtual void [SendStartDacq](#) ()  
*Start sampling.*
- virtual void [SendStartDacq](#) (int VirtualDacqMap)  
*Start sampling.*
- virtual void [SendStartStgAndDacq](#) (uint32\_t trigger\_map, int VirtualDacqMap)  
*Start sampling together with the STG.*
- virtual void [SendStopDacq](#) ()  
*Stop sampling.*
- virtual void [SendStopDacq](#) (int VirtualDacqMap)  
*Stop sampling.*

**Parameters**

VirtualDacqMap	
----------------	--

- virtual void [SendStopStgAndDacq](#) (uint32\_t trigger\_map, int VirtualDacqMap)  
*Stop sampling together with the STG.*
- virtual void [SendStopStgAndDacqWithOptions](#) (uint32\_t trigger\_map, int VirtualDacqMap, int options)  
*Stop sampling together with the STG and options.*
- virtual void [StartLoop](#) ()  
*Start the data acquisition thread.*
- virtual void [StartLoop](#) (int32\_t timeout)  
*Start the data acquisition thread.*
- virtual void [StartLoop](#) (int32\_t timeout, int32\_t numSubmittedUsbBuffers, int32\_t numUsbBuffers, int32\_t packetsInUrb)  
*Start the data acquisition thread.*
- virtual void [StartLoop](#) (int32\_t timeout, int32\_t numSubmittedUsbBuffers, int32\_t numUsbBuffers, int32\_t packetsInUrb, uint32\_t virtualDevice)  
*Start the data acquisition thread.*
- virtual void [StopLoop](#) ()
- virtual void [ClearBuffers](#) ()
- virtual void [StartDacq](#) ()  
*Start the data acquisition thread and sampling.*
- virtual void [StartDacq](#) (int32\_t timeout)  
*Start the data acquisition thread and sampling.*
- virtual void [StartDacq](#) (int32\_t timeout, int32\_t numSubmittedUsbBuffers, int32\_t numUsbBuffers, int32\_t packetsInUrb)  
*Start the data acquisition thread and sampling.*
- virtual void [StartDacq](#) (int32\_t timeout, int32\_t numSubmittedUsbBuffers, int32\_t numUsbBuffers, int32\_t packetsInUrb, uint32\_t virtualDevice)  
*Start the data acquisition thread and sampling.*
- virtual void [StopDacq](#) ()  
*Stop the data acquisition thread and sampling.*
- virtual void [StopDacq](#) (uint32\_t virtualDevice)  
*Stop the data acquisition thread and sampling.*

- virtual uint32\_t [SetPoti](#) (uint32\_t channel, uint32\_t value, bool write\_nvram)
- virtual uint32\_t [GetPoti](#) (uint32\_t channel, [System::Runtime::InteropServices::Out]uint32\_t% value)
- virtual [CFilterPropertyNet](#) ^ [GetFilterProperty](#) (DacqGroupChannelEnumNet GroupID, unsigned int index)
- virtual array< [CFilterPropertyNet](#) ^> ^ [CMcsUsbDacqNet::GetFilterProperties](#) (DacqGroupChannelEnumNet GroupID)
- int [GetChannelDataFillSize](#) ()
- virtual void [SetSelectedChannels](#) (int nChannels, int queuesize, int threshold, SampleSizeNet samplesize, int ChannelsInBlock)

*Create a FIFO queue per channel. Each channel will have its own FIFO and Callback function.*

- virtual void [SetSelectedChannels](#) (int nChannels, int queuesize, int threshold, SampleSizeNet samplesize, SampleDstSizeNet sampleDstSize, int ChannelsInBlock)
- virtual void [SetSelectedChannels](#) (array< bool >^ selectedChannels, int queuesize, int threshold, SampleSizeNet samplesize, int ChannelsInBlock)

*Create a FIFO queue per channel. Each channel will have its own FIFO and Callback function.*

- virtual void [SetSelectedChannels](#) (array< bool >^ selectedChannels, int queuesize, int threshold, SampleSizeNet samplesize, SampleDstSizeNet sampleDstSize, int ChannelsInBlock)
- virtual void [SetSelectedData](#) (int nChannels, int queuesize, int threshold, SampleSizeNet samplesize, int ChannelsInBlock)

*Create a common FIFO queue for all channels. Use handle = 0 in the ChannelBlock\_ReadFrames... functions.*

- virtual void [SetSelectedData](#) (int nChannels, int queuesize, int threshold, SampleSizeNet samplesize, SampleDstSizeNet sampleDstSize, int ChannelsInBlock)
- virtual void [SetSelectedData](#) (array< bool >^ selectedChannels, int queuesize, int threshold, SampleSizeNet samplesize, int ChannelsInBlock)

*Create a common FIFO queue for all channels. Use handle = 0 in the ChannelBlock\_ReadFrames... functions.*

- virtual void [SetSelectedData](#) (array< bool >^ selectedChannels, int queuesize, int threshold, SampleSizeNet samplesize, SampleDstSizeNet sampleDstSize, int ChannelsInBlock)
- virtual int [AddSelectedChannelsQueue](#) (int nByteOffset, int nChannelOffset, int nChannels, int queuesize, int threshold, SampleSizeNet samplesize)

*Adds a common FIFO queue for all channels. Data in callback will be a list per channel. Use ChannelBlock\_ReadFramesDict... with handle = 0 to read the data.*

- virtual int [AddSelectedChannelsQueue](#) (int nByteOffset, int nChannelOffset, int nChannels, int queuesize, int threshold, SampleSizeNet samplesize, SampleDstSizeNet sampleDstSize)
- virtual int [AddSelectedChannelsQueue](#) (int nByteOffset, int nChannelOffset, array< bool >^ selectedChannels, int queuesize, int threshold, SampleSizeNet samplesize)

*Adds a common FIFO queue for all channels. Data in callback will be a list per channel. Use ChannelBlock\_ReadFramesDict... with handle = 0 to read the data.*

- virtual int [AddSelectedChannelsQueue](#) (int nByteOffset, int nChannelOffset, array< bool >^ selectedChannels, int queuesize, int threshold, SampleSizeNet samplesize, SampleDstSizeNet sampleDstSize)
- virtual void [SetSelectedChannelsQueue](#) (int nChannels, int queuesize, int threshold, SampleSizeNet samplesize, int ChannelsInBlock)

*Create a common FIFO queue for all channels. Data in callback will be a list per channel. Use ChannelBlock\_ReadFramesDict... with handle = 0 to read the data.*

- virtual void [SetSelectedChannelsQueue](#) (int nChannels, int queuesize, int threshold, SampleSizeNet samplesize, SampleDstSizeNet sampleDstSize, int ChannelsInBlock)
- virtual void [SetSelectedChannelsQueue](#) (array< bool >^ selectedChannels, int queuesize, int threshold, SampleSizeNet samplesize, int ChannelsInBlock)

*Create a common FIFO queue for all channels. Data in callback will be a list per channel. Use ChannelBlock\_ReadFramesDict... with handle = 0 to read the data.*

- virtual void [SetSelectedChannelsQueue](#) (array< bool >^ selectedChannels, int queuesize, int threshold, SampleSizeNet samplesize, SampleDstSizeNet sampleDstSize, int ChannelsInBlock)
- virtual uint32\_t [ChannelBlock\\_AvailFrames](#) (int handle)

*Get the number of sample frames already available in the FIFO.*

- virtual uint32\_t [ChannelBlock\\_AvailFrames](#) (int handle, int queue)
- virtual array< uint16\_t > ^ [ChannelBlock\\_ReadFramesUI16](#) (int handle, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)

*Read data from a FIFO queue in uint16\_t data format*

- virtual uint32\_t [ChannelBlock\\_ReadFramesUI16](#) (int handle, array< uint16\_t >^ buffer, int frames\_pos, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)

*Read data from a FIFO queue in uint16\_t data format*

- virtual array< int16\_t > ^ [ChannelBlock\\_ReadFramesI16](#) (int handle, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)

*Read data from a FIFO queue in int16\_t data format*

- virtual uint32\_t [ChannelBlock\\_ReadFramesI16](#) (int handle, array< int16\_t >^ buffer, int frames\_pos, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)

*Read data from a FIFO queue in int16\_t data format*

- virtual array< uint32\_t > ^ [ChannelBlock\\_ReadFramesUI32](#) (int handle, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)

*Read data from a FIFO queue in uint32\_t data format*

- virtual uint32\_t [ChannelBlock\\_ReadFramesUI32](#) (int handle, array< uint32\_t >^ buffer, int frames\_pos, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)

*Read data from a FIFO queue in uint32\_t data format*

- virtual array< int32\_t > ^ [ChannelBlock\\_ReadFramesI32](#) (int handle, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)

*Read data from a FIFO queue in uint32\_t data format*

- virtual uint32\_t [ChannelBlock\\_ReadFramesI32](#) (int handle, array< int32\_t >^ buffer, int frames\_pos, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)

*Read data from a FIFO queue in uint32\_t data format*

- virtual array< array< uint16\_t >^> ^ [ChannelBlock\\_ReadAsFrameArrayUI16](#) (int handle, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)

*Read data from a FIFO queue as array of uint16\_t data frame arrays*

- virtual array< array< uint16\_t >^> ^ [ChannelBlock\\_ReadAsFrameArrayUI16](#) (int handle, int queue, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)

*Read data from a FIFO queue as array of uint16\_t data frame arrays*

- virtual array< array< int16\_t >^> ^ [ChannelBlock\\_ReadAsFrameArrayI16](#) (int handle, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)

*Read data from a FIFO queue as array of uint16\_t data frame arrays*

- virtual array< array< int16\_t >^> ^ [ChannelBlock\\_ReadAsFrameArrayI16](#) (int handle, int queue, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)

*Read data from a FIFO queue as array of uint16\_t data frame arrays*

- virtual array< array< uint32\_t >^> ^ [ChannelBlock\\_ReadAsFrameArrayUI32](#) (int handle, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)

*Read data from a FIFO queue as array of uint16\_t data frame arrays*

- virtual array< array< uint32\_t >^> ^ [ChannelBlock\\_ReadAsFrameArrayUI32](#) (int handle, int queue, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)

*Read data from a FIFO queue as array of uint16\_t data frame arrays*

- virtual array< array< int32\_t >^> ^ [ChannelBlock\\_ReadAsFrameArrayI32](#) (int handle, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)

*Read data from a FIFO queue as array of uint16\_t data frame arrays*

- virtual array< array< int32\_t >^> ^ [ChannelBlock\\_ReadAsFrameArrayI32](#) (int handle, int queue, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)

*Read data from a FIFO queue as array of uint16\_t data frame arrays*

- virtual System::Collections::Generic::Dictionary< int, array< uint16\_t >^> ^ [ChannelBlock\\_ReadFramesDictUI16](#) (int handle, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)

*Read data from a FIFO queue in uint16\_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number*

- virtual System::Collections::Generic::Dictionary< int, array< int16\_t >^> ^ [ChannelBlock\\_ReadFramesDictI16](#) (int handle, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)

*Read data from a FIFO queue in int16\_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number*



- virtual System::Collections::Generic::Dictionary< int, array< uint32\_t >^> ^ [ChannelBlock\\_ReadFramesDictUI32](#) (int handle, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)  
*Read data from a FIFO queue in uint32\_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number*
- virtual System::Collections::Generic::Dictionary< int, array< int32\_t >^> ^ [ChannelBlock\\_ReadFramesDictI32](#) (int handle, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)  
*Read data from a FIFO queue in int32\_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number*
- virtual System::Collections::Generic::Dictionary< int, array< uint16\_t >^> ^ [GetGroupChannelDataUI16](#) (DacqGroupChannelEnumNet group, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)  
*Read data from a FIFO queue in uint16\_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number*
- virtual System::Collections::Generic::Dictionary< int, array< int16\_t >^> ^ [GetGroupChannelDataI16](#) (DacqGroupChannelEnumNet group, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)  
*Read data from a FIFO queue in int16\_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number*
- virtual System::Collections::Generic::Dictionary< int, array< uint32\_t >^> ^ [GetGroupChannelDataUI32](#) (DacqGroupChannelEnumNet group, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)  
*Read data from a FIFO queue in uint32\_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number*
- virtual System::Collections::Generic::Dictionary< int, array< int32\_t >^> ^ [GetGroupChannelDataI32](#) (DacqGroupChannelEnumNet group, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)  
*Read data from a FIFO queue in int32\_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number*
- void [SetupGroupDacqQueue](#) (int queuesize, int threshold)
- [CHWInfo](#) ^ [HWInfo](#) ()

### Static Public Attributes

- static const int [Error\\_Callback\\_Queue\\_Full](#) = 0x100
- static const int [Error\\_Callback\\_Aquisition\\_Stopped](#) = 0x200
- static const int [Error\\_Callback\\_Packet\\_Error](#) = 1
- static const int [Error\\_Callback\\_RingQueue\\_Full](#) = 3
- static const int [Error\\_Callback\\_Frames\\_Lost](#) = 4
- static const int [Error\\_Callback\\_Data\\_lost](#) = 5

### Properties

- virtual int [Samplerate](#) [get, set]  
*The sampling frequency of the device in Hz.*

### Events

- [OnChannelData](#) ^ [ChannelDataEvent](#)
- [OnError](#) ^ [ErrorEvent](#)

### Additional Inherited Members

#### 11.46.1 Detailed Description

Base class for data acquisition devices.

## 11.46.2 Constructor & Destructor Documentation

**11.46.2.1 CMcsUsbDacqNet()** `CMcsUsbDacqNet ( )`

**11.46.2.2 ~CMcsUsbDacqNet()** `~CMcsUsbDacqNet ( )`

## 11.46.3 Member Function Documentation

**11.46.3.1 AddSelectedChannelsQueue()** [1/4] `virtual int AddSelectedChannelsQueue (`  
`int nByteOffset,`  
`int nChannelOffset,`  
`array< bool >^ selectedChannels,`  
`int queuesize,`  
`int threshold,`  
`SampleSizeNet samplesize ) [virtual]`

Adds a common FIFO queue for all channels. Data in callback will be a list per channel. Use `ChannelBlock_Read↔FramesDict...` with `handle = 0` to read the data.

When using 32 bit data format, `ChannelsInBlock` is still the number of 16 bit channels per frame, as obtained from `GetChannelsInBlock`, while `nChannels` is the number of 32 bit channels to be read from the device. So when all channels from a device are read in 32 bit data format `nChannels = ChannelsInBlock/2`

### Parameters

<code>nByteOffset</code>	Number of bytes to start with.
--------------------------	--------------------------------

### Parameters

<code>nChannelOffset</code>	Number of channel to start with (counted in <code>samplesize</code> bytes).
-----------------------------	---

### Parameters

<code>selectedChannels</code>	List of channels to be collected in the FIFO.
-------------------------------	---

## Parameters

<i>queuesize</i>	Size of sample frames the FIFO can hold.
------------------	--

## Parameters

<i>threshold</i>	Number of sample frames the FIFO must acquire before the callback function is called.
------------------	---

## Parameters

<i>samplesize</i>	size of the datawords, either 16 or 32bit.
-------------------	--

## Returns

The handle to the Queue.

**11.46.3.2 AddSelectedChannelsQueue() [2/4]** virtual int AddSelectedChannelsQueue (   
     int *nByteOffset*,   
     int *nChannelOffset*,   
     array< bool >^ *selectedChannels*,   
     int *queuesize*,   
     int *threshold*,   
     SampleSizeNet *samplesize*,   
     SampleDstSizeNet *sampleDstSize* ) [virtual]

**11.46.3.3 AddSelectedChannelsQueue() [3/4]** virtual int AddSelectedChannelsQueue (   
     int *nByteOffset*,   
     int *nChannelOffset*,   
     int *nChannels*,   
     int *queuesize*,   
     int *threshold*,   
     SampleSizeNet *samplesize* ) [virtual]

Adds a common FIFO queue for all channels. Data in callback will be a list per channel. Use ChannelBlock\_Read↔FramesDict... with handle = 0 to read the data.

When using 32 bit data format, ChannelsInBlock is still the number of 16 bit channels per frame, as obtained from GetChannelsInBlock, while nChannels is the number of 32 bit channels to be read from the device. So when all channels from a device are read in 32 bit data format nChannels = ChannelsInBlock/2

## Parameters

<i>nByteOffset</i>	Number of bytes to start with.
--------------------	--------------------------------

## Parameters

<i>nChannelOffset</i>	Number of channel to start with (counted in samplesize bytes).
-----------------------	--

## Parameters

<i>nChannels</i>	Number of channels to be collected in the FIFO.
------------------	---

## Parameters

<i>queuesize</i>	Size of sample frames the FIFO can hold.
------------------	--

## Parameters

<i>threshold</i>	Number of sample frames the FIFO must acquire before the callback function is called.
------------------	---

## Parameters

<i>samplesize</i>	size of the datawords, either 16 or 32bit.
-------------------	--

## Returns

The handle to the Queue.

**11.46.3.4 AddSelectedChannelsQueue()** [4/4] virtual int AddSelectedChannelsQueue (   
     int *nByteOffset*,   
     int *nChannelOffset*,   
     int *nChannels*,   
     int *queuesize*,   
     int *threshold*,   
     SampleSizeNet *samplesize*,   
     SampleDstSizeNet *sampleDstSize* ) [virtual]

**11.46.3.5 ChannelBlock\_AvailFrames()** [1/2] virtual uint32\_t ChannelBlock\_AvailFrames (   
     int *handle* ) [virtual]

Get the number of sample frames already available in the FIFO.

#### Parameters

<i>handle</i>	Handle of the FIFO queue. Either zero when the SetSelectedData call was used or the channel number.
---------------	---

#### Returns

Number of sample frames available in the FIFO.

**11.46.3.6 ChannelBlock\_AvailFrames()** [2/2] virtual uint32\_t ChannelBlock\_AvailFrames (   
     int *handle*,   
     int *queue* ) [virtual]

**11.46.3.7 ChannelBlock\_ReadAsFrameArrayI16()** [1/2] virtual array<array<int16\_t>> ^ Channel↔  
 Block\_ReadAsFrameArrayI16 (   
     int *handle*,   
     int *frames*,   
     [System::Runtime::InteropServices::Out] int % *frames\_ret* ) [virtual]

Read data from a FIFO queue as array of uint16\_t data frame arrays

#### Parameters

<i>handle</i>	Handle of the FIFO queue. Zero when the SetSelectedData call was used.
---------------	--

## Parameters

<i>frames</i>	Number of sample frames to read.
---------------	----------------------------------

## Parameters

<i>frames_ret</i>	Number of sample frames which were read, might be smaller than frames.
-------------------	--

## Returns

Array of int16\_t frame arrays.

### 11.46.3.8 ChannelBlock\_ReadAsFrameArrayI16() [2/2] virtual array<array<int16\_t>^> ^ Channel↔

```
Block_ReadAsFrameArrayI16 (
    int handle,
    int queue,
    int frames,
    [System::Runtime::InteropServices::Out] int % frames_ret ) [virtual]
```

Read data from a FIFO queue as array of uint16\_t data frame arrays

## Parameters

<i>handle</i>	Handle of the FIFO queue. Zero when the SetSelectedData call was used.
---------------	--

## Parameters

<i>queue</i>	Number of the sub queue.
<i>frames</i>	Number of sample frames to read.

## Parameters

<i>frames_ret</i>	Number of sample frames which were read, might be smaller than frames.
-------------------	--

**Returns**

Array of int16\_t frame arrays.

**11.46.3.9 ChannelBlock\_ReadAsFrameArrayI32()** [1/2] virtual array<array<int32\_t>^> ^ Channel↔  
 Block\_ReadAsFrameArrayI32 (  
     int *handle*,  
     int *frames*,  
     [System::Runtime::InteropServices::Out] int % *frames\_ret* ) [virtual]

Read data from a FIFO queue as array of uint16\_t data frame arrays

**Parameters**

<i>handle</i>	Handle of the FIFO queue. Zero when the SetSelectedData call was used.
---------------	--

**Parameters**

<i>frames</i>	Number of sample frames to read.
---------------	----------------------------------

**Parameters**

<i>frames_ret</i>	Number of sample frames which were read, might be smaller than frames.
-------------------	--

**Returns**

Array of int32\_t frame arrays.

**11.46.3.10 ChannelBlock\_ReadAsFrameArrayI32()** [2/2] virtual array<array<int32\_t>^> ^ Channel↔  
 Block\_ReadAsFrameArrayI32 (  
     int *handle*,  
     int *queue*,  
     int *frames*,  
     [System::Runtime::InteropServices::Out] int % *frames\_ret* ) [virtual]

Read data from a FIFO queue as array of uint16\_t data frame arrays

## Parameters

<i>handle</i>	Handle of the FIFO queue. Zero when the SetSelectedData call was used.
<i>queue</i>	Number of the sub queue.
<i>frames</i>	Number of sample frames to read.

## Parameters

<i>frames_ret</i>	Number of sample frames which were read, might be smaller than frames.
-------------------	--

## Returns

Array of int32\_t frame arrays.

### 11.46.3.11 ChannelBlock\_ReadAsFrameArrayUI16() [1/2] virtual array<array<uint16\_t>^> ^

```
ChannelBlock_ReadAsFrameArrayUI16 (
    int handle,
    int frames,
    [System::Runtime::InteropServices::Out] int % frames_ret ) [virtual]
```

Read data from a FIFO queue as array of uint16\_t data frame arrays

## Parameters

<i>handle</i>	Handle of the FIFO queue. Zero when the SetSelectedData call was used.
---------------	--

## Parameters

<i>frames</i>	Number of sample frames to read.
---------------	----------------------------------

## Parameters

<i>frames_ret</i>	Number of sample frames which were read, might be smaller than frames.
-------------------	--



**Returns**

Array of uint16\_t frame arrays.

**11.46.3.12 ChannelBlock\_ReadAsFrameArrayUI16()** [2/2] virtual array<array<uint16\_t>^> ^

```
ChannelBlock_ReadAsFrameArrayUI16 (
    int handle,
    int queue,
    int frames,
    [System::Runtime::InteropServices::Out] int % frames_ret ) [virtual]
```

Read data from a FIFO queue as array of uint16\_t data frame arrays

**Parameters**

<i>handle</i>	Handle of the FIFO queue. Zero when the SetSelectedData call was used.
---------------	--

**Parameters**

<i>queue</i>	Number of the sub queue.
<i>frames</i>	Number of sample frames to read.

**Parameters**

<i>frames_ret</i>	Number of sample frames which were read, might be smaller than frames.
-------------------	--

**Returns**

Array of uint16\_t frame arrays.

**11.46.3.13 ChannelBlock\_ReadAsFrameArrayUI32()** [1/2] virtual array<array<uint32\_t>^> ^

```
ChannelBlock_ReadAsFrameArrayUI32 (
    int handle,
    int frames,
    [System::Runtime::InteropServices::Out] int % frames_ret ) [virtual]
```

Read data from a FIFO queue as array of uint16\_t data frame arrays

## Parameters

<i>handle</i>	Handle of the FIFO queue. Zero when the SetSelectedData call was used.
---------------	--

## Parameters

<i>frames</i>	Number of sample frames to read.
---------------	----------------------------------

## Parameters

<i>frames_ret</i>	Number of sample frames which were read, might be smaller than frames.
-------------------	--

## Returns

Array of uint32\_t frame arrays.

#### 11.46.3.14 ChannelBlock\_ReadAsFrameArrayUI32() [2/2] virtual array<array<uint32\_t>^> ^

```
ChannelBlock_ReadAsFrameArrayUI32 (
    int handle,
    int queue,
    int frames,
    [System::Runtime::InteropServices::Out] int % frames_ret ) [virtual]
```

Read data from a FIFO queue as array of uint16\_t data frame arrays

## Parameters

<i>handle</i>	Handle of the FIFO queue. Zero when the SetSelectedData call was used.
---------------	--

## Parameters

<i>queue</i>	Number of the sub queue.
<i>frames</i>	Number of sample frames to read.

## Parameters

<i>frames_ret</i>	Number of sample frames which were read, might be smaller than frames.
-------------------	--

## Returns

Array of uint32\_t frame arrays.

**11.46.3.15 ChannelBlock\_ReadFramesDictI16()** virtual System::Collections::Generic::Dictionary<int, array<int16\_t>^> ^ ChannelBlock\_ReadFramesDictI16 (   
     int *handle*,  
     int *frames*,  
     [System::Runtime::InteropServices::Out] int % *frames\_ret* ) [virtual]

Read data from a FIFO queue in int16\_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number

## Parameters

<i>handle</i>	Handle of the FIFO queue. Zero when the SetSelectedChannelsQueue call was used.
---------------	---

## Parameters

<i>frames</i>	Number of sample frames to read.
<i>frames_ret</i>	Number of sample frames which were read, might be smaller than frames.

## Returns

Dictionary of int16\_t arrays and hardware channel as key.

**11.46.3.16 ChannelBlock\_ReadFramesDictI32()** virtual System::Collections::Generic::Dictionary<int, array<int32\_t>^> ^ ChannelBlock\_ReadFramesDictI32 (   
     int *handle*,  
     int *frames*,  
     [System::Runtime::InteropServices::Out] int % *frames\_ret* ) [virtual]

Read data from a FIFO queue in int32\_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number

## Parameters

<i>handle</i>	Handle of the FIFO queue. Zero when the SetSelectedChannelsQueue call was used.
---------------	---

## Parameters

<i>frames</i>	Number of sample frames to read.
<i>frames_ret</i>	Number of sample frames which were read, might be smaller than frames.

## Returns

Dictionary of int32\_t arrays and hardware channel as key.

```
11.46.3.17 ChannelBlock_ReadFramesDictUI16() virtual System::Collections::Generic::Dictionary<int,
array<uint16_t>> ^ ChannelBlock_ReadFramesDictUI16 (
    int handle,
    int frames,
    [System::Runtime::InteropServices::Out] int % frames_ret ) [virtual]
```

Read data from a FIFO queue in uint16\_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number

## Parameters

<i>handle</i>	Handle of the FIFO queue. Zero when the SetSelectedChannelsQueue call was used.
---------------	---

## Parameters

<i>frames</i>	Number of sample frames to read.
<i>frames_ret</i>	Number of sample frames which were read, might be smaller than frames.

## Returns

Dictionary of uint16\_t arrays and hardware channel as key.

**11.46.3.18 ChannelBlock\_ReadFramesDictUI32()** virtual System::Collections::Generic::Dictionary<int, array<uint32\_t>^> ^ ChannelBlock\_ReadFramesDictUI32 ( int *handle*, int *frames*, [System::Runtime::InteropServices::Out] int % *frames\_ret* ) [virtual]

Read data from a FIFO queue in uint32\_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number

#### Parameters

<i>handle</i>	Handle of the FIFO queue. Zero when the SetSelectedChannelsQueue call was used.
---------------	---

#### Parameters

<i>frames</i>	Number of sample frames to read.
<i>frames_ret</i>	Number of sample frames which were read, might be smaller than frames.

#### Returns

Dictionary of uint32\_t arrays and hardware channel as key.

**11.46.3.19 ChannelBlock\_ReadFramesI16()** [1/2] virtual uint32\_t ChannelBlock\_ReadFramesI16 ( int *handle*, array< int16\_t >^ *buffer*, int *frames\_pos*, int *frames*, [System::Runtime::InteropServices::Out] int % *frames\_ret* ) [virtual]

Read data from a FIFO queue in int16\_t data format

#### Parameters

<i>handle</i>	Handle of the FIFO queue. Either zero when the SetSelectedData call was used or the channel number.
---------------	---

#### Parameters

<i>buffer</i>	Buffer to put the data from the device in.
<i>frames_pos</i>	Position in buffer where to put the data.
<i>frames</i>	Number of sample frames to read.

## Parameters

<i>frames_ret</i>	Number of sample frames which were read, might be smaller than frames.
-------------------	--

## Returns

Error Status. 0 on success.

**11.46.3.20 ChannelBlock\_ReadFramesI16()** [2/2] `virtual array<int16_t> ^ ChannelBlock_ReadFramesI16 (`  
`int handle,`  
`int frames,`  
`[System::Runtime::InteropServices::Out] int % frames_ret ) [virtual]`

Read data from a FIFO queue in int16\_t data format

## Parameters

<i>handle</i>	Handle of the FIFO queue. Either zero when the SetSelectedData call was used or the channel number.
---------------	---

## Parameters

<i>frames</i>	Number of sample frames to read.
---------------	----------------------------------

## Parameters

<i>frames_ret</i>	Number of sample frames which were read, might be smaller than frames.
-------------------	--

**11.46.3.21 ChannelBlock\_ReadFramesI32()** [1/2] `virtual uint32_t ChannelBlock_ReadFramesI32 (`  
`int handle,`  
`array< int32_t >^ buffer,`  
`int frames_pos,`

```
int frames,
[System::Runtime::InteropServices::Out] int % frames_ret ) [virtual]
```

Read data from a FIFO queue in uint32\_t data format

#### Parameters

<i>handle</i>	Handle of the FIFO queue. Either zero when the SetSelectedData call was used or the channel number.
---------------	---

#### Parameters

<i>buffer</i>	Buffer to put the data from the device in.
<i>frames_pos</i>	Position in buffer where to put the data.
<i>frames</i>	Number of sample frames to read.

#### Parameters

<i>frames_ret</i>	Number of sample frames which were read, might be smaller than frames.
-------------------	--

#### Returns

Error Status. 0 on success.

**11.46.3.22 ChannelBlock\_ReadFramesI32()** [2/2] virtual array<int32\_t> ^ ChannelBlock\_ReadFramesI32 (

```
int handle,
int frames,
[System::Runtime::InteropServices::Out] int % frames_ret ) [virtual]
```

Read data from a FIFO queue in uint32\_t data format

#### Parameters

<i>handle</i>	Handle of the FIFO queue. Either zero when the SetSelectedData call was used or the channel number.
---------------	---

## Parameters

<i>frames</i>	Number of sample frames to read.
---------------	----------------------------------

## Parameters

<i>frames_ret</i>	Number of sample frames which were read, might be smaller than frames.
-------------------	--

**11.46.3.23 ChannelBlock\_ReadFramesUI16()** [1/2] `virtual uint32_t ChannelBlock_ReadFramesUI16 ( int handle, array< uint16_t >^ buffer, int frames_pos, int frames, [System::Runtime::InteropServices::Out] int % frames_ret ) [virtual]`

Read data from a FIFO queue in uint16\_t data format

## Parameters

<i>handle</i>	Handle of the FIFO queue. Either zero when the SetSelectedData call was used or the channel number.
<i>buffer</i>	Buffer to put the data from the device in.
<i>frames_pos</i>	Position in buffer where to put the data.
<i>frames</i>	Number of sample frames to read.

## Parameters

<i>frames_ret</i>	Number of sample frames which were read, might be smaller than frames.
-------------------	--

## Returns

Error Status. 0 on success.

**11.46.3.24 ChannelBlock\_ReadFramesUI16()** [2/2] `virtual array<uint16_t> ^ ChannelBlock_ReadFramesUI16 ( int handle, int frames, [System::Runtime::InteropServices::Out] int % frames_ret ) [virtual]`

Read data from a FIFO queue in uint16\_t data format



## Parameters

<i>handle</i>	Handle of the FIFO queue. Either zero when the SetSelectedData call was used or the channel number.
---------------	---

## Parameters

<i>frames</i>	Number of sample frames to read.
---------------	----------------------------------

## Parameters

<i>frames_ret</i>	Number of sample frames which were read, might be smaller than frames.
-------------------	--

## Returns

Array of data from the device.

**11.46.3.25 ChannelBlock\_ReadFramesUI32()** [1/2] `virtual uint32_t ChannelBlock_ReadFramesUI32 ( int handle, array< uint32_t >^ buffer, int frames_pos, int frames, [System::Runtime::InteropServices::Out] int % frames_ret ) [virtual]`

Read data from a FIFO queue in uint32\_t data format

## Parameters

<i>handle</i>	Handle of the FIFO queue. Either zero when the SetSelectedData call was used or the channel number.
---------------	---

## Parameters

<i>buffer</i>	Buffer to put the data from the device in.
<i>frames_pos</i>	Position in buffer where to put the data.
<i>frames</i>	Number of sample frames to read.

## Parameters

<i>frames_ret</i>	Number of sample frames which were read, might be smaller than frames.
-------------------	--

## Returns

Error Status. 0 on success.

**11.46.3.26 ChannelBlock\_ReadFramesUI32()** [2/2] `virtual array<uint32_t> ^ ChannelBlock_ReadFramesUI32 (`  
`int handle,`  
`int frames,`  
`[System::Runtime::InteropServices::Out] int % frames_ret ) [virtual]`

Read data from a FIFO queue in uint32\_t data format

## Parameters

<i>handle</i>	Handle of the FIFO queue. Either zero when the SetSelectedData call was used or the channel number.
---------------	---

## Parameters

<i>frames</i>	Number of sample frames to read.
---------------	----------------------------------

## Parameters

<i>frames_ret</i>	Number of sample frames which were read, might be smaller than frames.
-------------------	--

**11.46.3.27 ClearBuffers()** `virtual void ClearBuffers ( ) [virtual]`

**11.46.3.28 CMcsUsbDacqNet::GetFilterProperties()** virtual array<CFilterPropertyNet^> ^ CMcsUsbDacqNet::GetFilterProperties ( DacqGroupChannelEnumNet GroupID ) [virtual]

**11.46.3.29 GetAdapterType()** virtual AdapterTypeEnumNet GetAdapterType ( ) [virtual]

Gets the adapter which is connected to the MEA2100 device.

#### Returns

AdapterTypeEnumNet which enumerates the possible adapters.

**11.46.3.30 GetAdcDataFormat()** virtual uint32\_t GetAdcDataFormat ( uint32\_t virtualDevice ) [virtual]

Gets the ADC data format, 16 means 16 bits, 24 means 24 bits, 32 means 32 bits.

#### Returns

The data format in bits.

**11.46.3.31 GetAdcZero()** virtual uint32\_t GetAdcZero ( uint32\_t virtualDevice, DacqGroupChannelEnumNet group, [System::Runtime::InteropServices::Out] int% adcz ) [virtual]

**11.46.3.32 GetAnalogValueUnit()** virtual uint32\_t GetAnalogValueUnit ( uint32\_t virtualDevice, DacqGroupChannelEnumNet group, [System::Runtime::InteropServices::Out] AnalogUnitEnumNet% unit ) [virtual]

**11.46.3.33 GetChannelDataFillSize()** int GetChannelDataFillSize ( )

**11.46.3.34 GetChannelLayout()** virtual uint32\_t GetChannelLayout ( [System::Runtime::InteropServices::Out] int% *AnalogChannels*, [System::Runtime::InteropServices::Out] int% *DigitalChannels*, [System::Runtime::InteropServices::Out] int% *ChecksumChannels*, [System::Runtime::InteropServices::Out] int% *TimestampChannels*, [System::Runtime::InteropServices::Out] int% *ChannelsInBlock*, unsigned int *virtualDevice* ) [virtual]

**11.46.3.35 GetChannelsInBlock()** virtual int32\_t GetChannelsInBlock ( unsigned int *virtualDevice* ) [virtual]

Get the number of 16 bit datawords which will be collected per sample frame, use after the device is configured.

#### Returns

Number of 16 bit datawords per sample frame.

**11.46.3.36 GetDataFormat()** virtual uint32\_t GetDataFormat ( uint32\_t *virtualDevice*, DacqGroupChannelEnumNet *group*, [System::Runtime::InteropServices::Out] int% *numberOfBits* ) [virtual]

**11.46.3.37 GetDataMode()** virtual DataModeEnumNet GetDataMode ( unsigned int *virtualDevice* ) [virtual]

Gets the data mode, can be 16, 24 or 32bit, all signed or unsigned on the MEA2100 device.

#### Parameters

<i>virtualDevice</i>	Virtual device to use.
----------------------	------------------------

#### Returns

DataModeEnumNet which enumerates the possible data modes.

**11.46.3.38 GetDigitalSource()** [1/5] void GetDigitalSource ( DigitalTargetEnumNet *digitaltarget*, int32\_t *NrChannel*, [System::Runtime::InteropServices::Out] DigitalSource< digitalsourceenum >^% *source*, [System::Runtime::InteropServices::Out] int% *bitnumber\_offset* )

Gets the function/source of an digital output bit.

This is the templated generic implementation.

## Parameters

<i>digitaltarget</i>	The digital target to query.
<i>NrChannel</i>	The channel/bit of target to query.
<i>source</i>	The source/function assignd to the digital target.
<i>bitnumber_offset</i>	An offset / bit number with the source/function.

**11.46.3.39 GetDigitalSource() [2/5]** `void GetDigitalSource ( DigitalTargetEnumNet digitaltarget, int32_t NrChannel, [System::Runtime::InteropServices::Out] DigitalSourceEnumNet% source, [System::Runtime::InteropServices::Out] int% bitnumber_offset )`

Gets the function/source of an digital output bit.

This overload is for the MEA2100 device.

## Parameters

<i>digitaltarget</i>	The digital target to query.
<i>NrChannel</i>	The channel/bit of target to query.
<i>source</i>	The source/function assignd to the digital target.
<i>bitnumber_offset</i>	An offset / bit number with the source/function.

**11.46.3.40 GetDigitalSource() [3/5]** `void GetDigitalSource ( DigitalTargetEnumNet digitaltarget, int32_t NrChannel, [System::Runtime::InteropServices::Out] MEA2100_256DigitalSourceEnumNet% source, [System::Runtime::InteropServices::Out] int% bitnumber_offset )`

Gets the function/source of an digital output bit.

This overload is for the MEA2100-256 device.

## Parameters

<i>digitaltarget</i>	The digital target to query.
<i>NrChannel</i>	The channel/bit of target to query.
<i>source</i>	The source/function assignd to the digital target.
<i>bitnumber_offset</i>	An offset / bit number with the source/function.

**11.46.3.41 GetDigitalSource() [4/5]** `void GetDigitalSource ( DigitalTargetEnumNet digitaltarget,`

```
int32_t NrChannel,
[System::Runtime::InteropServices::Out] SCUDigitalSourceEnumNet% source,
[System::Runtime::InteropServices::Out] int% bitnumber_offset )
```

Gets the function/source of an digital output bit.

This overload is for the SCU device.

#### Parameters

<i>digitaltarget</i>	The digital target to query.
<i>NrChannel</i>	The channel/bit of target to query.
<i>source</i>	The source/function assignd to the digital target.
<i>bitnumber_offset</i>	An offset / bit number with the source/function.

**11.46.3.42 GetDigitalSource()** [5/5] void GetDigitalSource ( DigitalTargetEnumNet *digitaltarget*, int32\_t *NrChannel*, [System::Runtime::InteropServices::Out] W2100DigitalSourceEnumNet% *source*, [System::Runtime::InteropServices::Out] int% *bitnumber\_offset* )

Gets the function/source of an digital output bit.

This overload is for the W2100 device.

#### Parameters

<i>digitaltarget</i>	The digital target to query.
<i>NrChannel</i>	The channel/bit of target to query.
<i>source</i>	The source/function assignd to the digital target.
<i>bitnumber_offset</i>	An offset / bit number with the source/function.

**11.46.3.43 GetFilterProperty()** virtual CFilterPropertyNet ^ GetFilterProperty ( DacqGroupChannelEnumNet *GroupID*, unsigned int *index* ) [virtual]

**11.46.3.44 GetGroupChannelDataI16()** virtual System::Collections::Generic::Dictionary<int, array<int16\_t>^> ^ GetGroupChannelDataI16 ( DacqGroupChannelEnumNet *group*, int *frames*, [System::Runtime::InteropServices::Out] int % *frames\_ret* ) [virtual]

Read data from a FIFO queue in int16\_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number

## Parameters

<i>group</i>	Group selector supported by the device.
--------------	---

## Parameters

<i>frames</i>	Number of sample frames to read.
<i>frames_ret</i>	Number of sample frames which were read, might be smaller than frames.

## Returns

Dictionary of int16\_t arrays and hardware channel as key.

**11.46.3.45 GetGroupChannelDataI32()** `virtual System::Collections::Generic::Dictionary<int, array<int32_t>> ^ GetGroupChannelDataI32 ( DacqGroupChannelEnumNet group, int frames, [System::Runtime::InteropServices::Out] int % frames_ret ) [virtual]`

Read data from a FIFO queue in int32\_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number

## Parameters

<i>group</i>	Group selector supported by the device.
--------------	---

## Parameters

<i>frames</i>	Number of sample frames to read.
<i>frames_ret</i>	Number of sample frames which were read, might be smaller than frames.

## Returns

Dictionary of int32\_t arrays and hardware channel as key.

**11.46.3.46 GetGroupChannelDataUI16()** virtual System::Collections::Generic::Dictionary<int, array<uint16\_t>^> ^ GetGroupChannelDataUI16 ( DacqGroupChannelEnumNet group, int frames, [System::Runtime::InteropServices::Out] int % frames\_ret ) [virtual]

Read data from a FIFO queue in uint16\_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number

#### Parameters

<i>group</i>	Group selector supported by the device.
--------------	---

#### Parameters

<i>frames</i>	Number of sample frames to read.
<i>frames_ret</i>	Number of sample frames which were read, might be smaller than frames.

#### Returns

Dictionary of uint16\_t arrays and hardware channel as key.

**11.46.3.47 GetGroupChannelDataUI32()** virtual System::Collections::Generic::Dictionary<int, array<uint32\_t>^> ^ GetGroupChannelDataUI32 ( DacqGroupChannelEnumNet group, int frames, [System::Runtime::InteropServices::Out] int % frames\_ret ) [virtual]

Read data from a FIFO queue in uint32\_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number

#### Parameters

<i>group</i>	Group selector supported by the device.
--------------	---

#### Parameters

<i>frames</i>	Number of sample frames to read.
<i>frames_ret</i>	Number of sample frames which were read, might be smaller than frames.



**Returns**

Dictionary of uint32\_t arrays and hardware channel as key.

**11.46.3.48 GetHardwareMaxRange()** virtual uint32\_t GetHardwareMaxRange (   
uint32\_t *virtualDevice*,   
DacqGroupChannelEnumNet *group*,   
[System::Runtime::InteropServices::Out] int% *r*,   
[System::Runtime::InteropServices::Out] int% *rUnit* ) [virtual]

**11.46.3.49 GetHardwareMinRange()** virtual uint32\_t GetHardwareMinRange (   
uint32\_t *virtualDevice*,   
DacqGroupChannelEnumNet *group*,   
[System::Runtime::InteropServices::Out] int% *r*,   
[System::Runtime::InteropServices::Out] int% *rUnit* ) [virtual]

**11.46.3.50 GetMaxSamplingFrequency()** virtual uint32\_t GetMaxSamplingFrequency (   
int *virtualDevice* ) [virtual]

Gets the maximal sampling frequency of the device.

**Returns**

Sampling frequency in Hz.

**11.46.3.51 GetMeaLayout()** virtual MeaLayoutEnumNet GetMeaLayout ( ) [virtual]

Gets the MEA layout which is connected to the MEA2100 device.

**Returns**

MeaLayoutEnumNet which enumerates the MEA types.

**11.46.3.52 GetMinSamplingFrequencyStepsize()** virtual uint32\_t GetMinSamplingFrequencyStepsize ( ) [virtual]

Gets the minimal sampling frequency step size increment value of the device.

**Returns**

Sampling frequency step size in Hz.

**11.46.3.53 GetNumberOfDataBits()** virtual uint32\_t GetNumberOfDataBits ( uint32\_t *virtualDevice*, DacqGroupChannelEnumNet *group*, [System::Runtime::InteropServices::Out] int% *numberOfBits* ) [virtual]

Get the real number of data bits.

This value may be different from the value returned by GetDataFormat, e.g. in MC\_Card the data are shifted 2 bits so the real number is 14 while the data format is 16 bits

**11.46.3.54 GetPoti()** virtual uint32\_t GetPoti ( uint32\_t *channel*, [System::Runtime::InteropServices::Out] uint32\_t% *value* ) [virtual]

**11.46.3.55 GetResolutionPerDigit()** virtual uint32\_t GetResolutionPerDigit ( uint32\_t *virtualDevice*, DacqGroupChannelEnumNet *group*, [System::Runtime::InteropServices::Out] int% *res*, [System::Runtime::InteropServices::Out] int% *resUnit* ) [virtual]

**11.46.3.56 GetSamplerate()** virtual int32\_t GetSamplerate ( unsigned int *virtualDevice* ) [virtual]

Gets the sampling frequency of the device.

**Returns**

Sampling frequency in Hz.

**11.46.3.57 GetVoltageRangeIndex()** virtual uint32\_t GetVoltageRangeIndex ( unsigned int *virtualDevice* ) [virtual]

**11.46.3.58 GetVoltageRangeInMicroVolt()** `virtual int32_t GetVoltageRangeInMicroVolt ( unsigned int virtualDevice ) [virtual]`

Gets the currently selected voltage range on devices which support multiple voltage ranges.

#### Returns

The Voltage Range in uV.

**11.46.3.59 GetVoltageRangeInMilliVolt()** `virtual int32_t GetVoltageRangeInMilliVolt ( ) [virtual]`

Gets the currently selected voltage range on devices which support multiple voltage ranges.

#### Returns

The rounded Voltage Range in mV.

**11.46.3.60 HWInfo()** `CHWInfo ^ HWInfo ( )`

**11.46.3.61 SendStartDacq()** [1/2] `virtual void SendStartDacq ( ) [virtual]`

Start sampling.

**11.46.3.62 SendStartDacq()** [2/2] `virtual void SendStartDacq ( int VirtualDacqMap ) [virtual]`

Start sampling.

#### Parameters

<i>VirtualDacqMap</i>	
-----------------------	--

**11.46.3.63 SendStartStgAndDacq()** `virtual void SendStartStgAndDacq ( uint32_t trigger_map, int VirtualDacqMap ) [virtual]`

Start sampling together with the STG.

## Parameters

<i>trigger_map</i>	
<i>VirtualDacqMap</i>	

**11.46.3.64 SendStopDacq()** [1/2] virtual void SendStopDacq ( ) [virtual]

Stop sampling.

**11.46.3.65 SendStopDacq()** [2/2] virtual void SendStopDacq (   
int *VirtualDacqMap* ) [virtual]

Stop sampling.

## Parameters

<i>VirtualDacqMap</i>	
-----------------------	--

**11.46.3.66 SendStopStgAndDacq()** virtual void SendStopStgAndDacq (   
uint32\_t *trigger\_map*,   
int *VirtualDacqMap* ) [virtual]

Stop sampling together with the STG.

## Parameters

<i>trigger_map</i>	
--------------------	--

**11.46.3.67 SendStopStgAndDacqWithOptions()** virtual void SendStopStgAndDacqWithOptions (   
uint32\_t *trigger\_map*,   
int *VirtualDacqMap*,   
int *options* ) [virtual]

Stop sampling together with the STG and options.

## Parameters

<i>trigger_map</i>	
--------------------	--

## Parameters

<i>options</i>	
----------------	--

## Parameters

<i>VirtualDacqMap</i>	
-----------------------	--

**11.46.3.68 SetDataMode()** `virtual void SetDataMode ( DataModeEnumNet dataMode, unsigned int virtualDevice ) [virtual]`

Sets the data mode, can be 16, 24 or 32bit, all signed or unsigned on the MEA2100 device.

## Parameters

<i>dataMode</i>	DataModeEnumNet enumerates the possible data modes.
<i>virtualDevice</i>	Virtual device to use.

**11.46.3.69 SetDigitalSource()** [1/5] `void SetDigitalSource ( DigitalTargetEnumNet digitaltarget, int32_t NrChannel, DigitalSource< digitalsourceenum >^ source, int bitnumber_offset )`

Sets the function/source of an digital output bit.

This is the templated generic implementation.

## Parameters

<i>digitaltarget</i>	The digital target to change.
<i>NrChannel</i>	The channel/bit of target to change.
<i>source</i>	The source/function to assign to the digital target.
<i>bitnumber_offset</i>	An offset / bit number with the source/function.

**11.46.3.70 SetDigitalSource() [2/5]** `void SetDigitalSource (`  
`DigitalTargetEnumNet digitaltarget,`  
`int32_t NrChannel,`  
`DigitalSourceEnumNet source,`  
`int bitnumber_offset )`

Sets the function/source of an digital output bit.

This overload is for the MEA2100 device.

#### Parameters

<i>digitaltarget</i>	The digital target to change.
<i>NrChannel</i>	The channel/bit of target to change.
<i>source</i>	The source/function to assign to the digital target.
<i>bitnumber_offset</i>	An offset / bit number with the source/function.

**11.46.3.71 SetDigitalSource() [3/5]** `void SetDigitalSource (`  
`DigitalTargetEnumNet digitaltarget,`  
`int32_t NrChannel,`  
`MEA2100_256DigitalSourceEnumNet source,`  
`int bitnumber_offset )`

Sets the function/source of an digital output bit.

This overload is for the MEA2100-256 device.

#### Parameters

<i>digitaltarget</i>	The digital target to change.
<i>NrChannel</i>	The channel/bit of target to change.
<i>source</i>	The source/function to assign to the digital target.
<i>bitnumber_offset</i>	An offset / bit number with the source/function.

**11.46.3.72 SetDigitalSource() [4/5]** `void SetDigitalSource (`  
`DigitalTargetEnumNet digitaltarget,`  
`int32_t NrChannel,`  
`SCUDigitalSourceEnumNet source,`  
`int bitnumber_offset )`

Sets the function/source of an digital output bit.

This overload is for the SCU device.

## Parameters

<i>digitaltarget</i>	The digital target to change.
<i>NrChannel</i>	The channel/bit of target to change.
<i>source</i>	The source/function to assign to the digital target.
<i>bitnumber_offset</i>	An offset / bit number with the source/function.

**11.46.3.73 SetDigitalSource()** [5/5] `void SetDigitalSource ( DigitalTargetEnumNet digitaltarget, int32_t NrChannel, W2100DigitalSourceEnumNet source, int bitnumber_offset )`

Sets the function/source of an digital output bit.

This overload is for the W2100 device.

## Parameters

<i>digitaltarget</i>	The digital target to change.
<i>NrChannel</i>	The channel/bit of target to change.
<i>source</i>	The source/function to assign to the digital target.
<i>bitnumber_offset</i>	An offset / bit number with the source/function.

**11.46.3.74 SetPoti()** `virtual uint32_t SetPoti ( uint32_t channel, uint32_t value, bool write_nvram ) [virtual]`

**11.46.3.75 SetSamplerate()** `virtual void SetSamplerate ( int32_t rate, unsigned int oversample, unsigned int virtualDevice ) [virtual]`

Sets the sampling frequency of the device.

## Parameters

<i>rate</i>	Sampling frequency in Hz.
-------------	---------------------------

**11.46.3.76 SetSelectedChannels()** [1/4] `virtual void SetSelectedChannels (`

```

    array< bool >^ selectedChannels,
    int queuesize,
    int threshold,
    SampleSizeNet samplesize,
    int ChannelsInBlock ) [virtual]

```

Create a FIFO queue per channel. Each channel will have its own FIFO and Callback function.

When using a 32bit sample size, the number obtained from GetChannelsInBlock must be divided by 2 to be used here, since GetChannelsInBlock returns the number of 16 bit datapoints per sample frame, while this functions uses the number of sample frames in its own data format.

#### Parameters

<i>selectedChannels</i>	List of channels to be collected in the FIFO.
-------------------------	---

#### Parameters

<i>queuesize</i>	Size of sample frames the FIFO can hold.
------------------	--

#### Parameters

<i>threshold</i>	Number of sample frames the FIFO must acquire before the callback function is called.
------------------	---

#### Parameters

<i>samplesize</i>	size of the datawords, either 16 or 32bit.
-------------------	--

#### Parameters

<i>ChannelsInBlock</i>	value obtained from GetChannelsInBlock.
------------------------	---

**11.46.3.77 SetSelectedChannels() [2/4]** virtual void SetSelectedChannels (

```

    array< bool >^ selectedChannels,
    int queuesize,
    int threshold,
    SampleSizeNet samplesize,

```



```
SampleDstSizeNet sampleDstSize,
int ChannelsInBlock ) [virtual]
```

**11.46.3.78 SetSelectedChannels() [3/4]** virtual void SetSelectedChannels (

```
int nChannels,
int queuesize,
int threshold,
SampleSizeNet samplesize,
int ChannelsInBlock ) [virtual]
```

Create a FIFO queue per channel. Each channel will have its own FIFO and Callback function.

When using a 32bit sample size, the number obtained from GetChannelsInBlock must be divided by 2 to be used here, since GetChannelsInBlock returns the number of 16 bit datapoints per sample frame, while this functions uses the number of sample frames in its own data format.

#### Parameters

<i>nChannels</i>	Number of channels to be collected in the FIFO.
------------------	---

#### Parameters

<i>queuesize</i>	Size of sample frames the FIFO can hold.
------------------	--

#### Parameters

<i>threshold</i>	Number of samples frames the FIFO must acquire before the callback function is called.
------------------	--

#### Parameters

<i>samplesize</i>	size of the datawords, either 16 or 32bit.
<i>ChannelsInBlock</i>	value obtained from GetChannelsInBlock.

**11.46.3.79 SetSelectedChannels() [4/4]** virtual void SetSelectedChannels (

```
int nChannels,
int queuesize,
int threshold,
```

```

SampleSizeNet samplesize,
SampleDstSizeNet sampleDstSize,
int ChannelsInBlock ) [virtual]

```

**11.46.3.80 SetSelectedChannelsQueue()** [1/4] virtual void SetSelectedChannelsQueue (   
array< bool >^ selectedChannels,   
int queuesize,   
int threshold,   
SampleSizeNet samplesize,   
int ChannelsInBlock ) [virtual]

Create a common FIFO queue for all channels. Data in callback will be a list per channel. Use ChannelBlock\_↔ ReadFramesDict... with handle = 0 to read the data.

When using 32 bit data format, ChannelsInBlock is still the number of 16 bit channels per frame, as obtained from GetChannelsInBlock, while nChannels is the number of 32 bit channels to be read from the device. So when all channels from a device are read in 32 bit data format nChannels = ChannelsInBlock/2

#### Parameters

<i>selectedChannels</i>	List of channels to be collected in the FIFO.
-------------------------	---

#### Parameters

<i>queuesize</i>	Size of sample frames the FIFO can hold.
------------------	--

#### Parameters

<i>threshold</i>	Number of sample frames the FIFO must acquire before the callback function is called.
------------------	---

#### Parameters

<i>samplesize</i>	size of the datawords, either 16 or 32bit.
-------------------	--

#### Parameters

<i>ChannelsInBlock</i>	value obtained from GetChannelsInBlock.
------------------------	---

**11.46.3.81 SetSelectedChannelsQueue() [2/4]** virtual void SetSelectedChannelsQueue (   
     array< bool >^ *selectedChannels*,   
     int *queuesize*,   
     int *threshold*,   
     SampleSizeNet *samplesize*,   
     SampleDstSizeNet *sampleDstSize*,   
     int *ChannelsInBlock* ) [virtual]

**11.46.3.82 SetSelectedChannelsQueue() [3/4]** virtual void SetSelectedChannelsQueue (   
     int *nChannels*,   
     int *queuesize*,   
     int *threshold*,   
     SampleSizeNet *samplesize*,   
     int *ChannelsInBlock* ) [virtual]

Create a common FIFO queue for all channels. Data in callback will be a list per channel. Use `ChannelBlock_↔`  
`ReadFramesDict...` with `handle = 0` to read the data.

When using 32 bit data format, `ChannelsInBlock` is still the number of 16 bit channels per frame, as obtained from `GetChannelsInBlock`, while `nChannels` is the number of 32 bit channels to be read from the device. So when all channels from a device are read in 32 bit data format `nChannels = ChannelsInBlock/2`

#### Parameters

<i>nChannels</i>	Number of channels to be collected in the FIFO.
------------------	---

#### Parameters

<i>queuesize</i>	Size of sample frames the FIFO can hold.
------------------	--

#### Parameters

<i>threshold</i>	Number of sample frames the FIFO must acquire before the callback function is called.
------------------	---

#### Parameters

<i>samplesize</i>	size of the datawords, either 16 or 32bit.
-------------------	--

## Parameters

<i>ChannelsInBlock</i>	value obtained from GetChannelsInBlock.
------------------------	---

**11.46.3.83 SetSelectedChannelsQueue()** [4/4] virtual void SetSelectedChannelsQueue (   
     int *nChannels*,  
     int *queuesize*,  
     int *threshold*,  
     SampleSizeNet *samplesize*,  
     SampleDstSizeNet *sampleDstSize*,  
     int *ChannelsInBlock* ) [virtual]

**11.46.3.84 SetSelectedData()** [1/4] virtual void SetSelectedData (   
     array< bool >^ *selectedChannels*,  
     int *queuesize*,  
     int *threshold*,  
     SampleSizeNet *samplesize*,  
     int *ChannelsInBlock* ) [virtual]

Create a common FIFO queue for all channels. Use handle = 0 in the ChannelBlock\_ReadFrames... functions.

When using 32 bit data format, ChannelsInBlock is still the number of 16 bit channels per frame, as obtained from GetChannelsInBlock, while nChannels is the number of 32 bit channels to be read from the device. So when all channels from a device are read in 32 bit data format nChannels = ChannelsInBlock/2

## Parameters

<i>selectedChannels</i>	List of channels to be collected in the FIFO.
-------------------------	---

## Parameters

<i>queuesize</i>	Size of sample frames the FIFO can hold.
------------------	--

## Parameters

<i>threshold</i>	Number of sample frames the FIFO must acquire before the callback function is called.
------------------	---

## Parameters

<i>samplesize</i>	size of the datawords, either 16 or 32bit.
<i>ChannelsInBlock</i>	value obtained from GetChannelsInBlock.

**11.46.3.85 SetSelectedData() [2/4]** virtual void SetSelectedData (  
     array< bool >^ *selectedChannels*,  
     int *queuesize*,  
     int *threshold*,  
     SampleSizeNet *samplesize*,  
     SampleDstSizeNet *sampleDstSize*,  
     int *ChannelsInBlock* ) [virtual]

**11.46.3.86 SetSelectedData() [3/4]** virtual void SetSelectedData (  
     int *nChannels*,  
     int *queuesize*,  
     int *threshold*,  
     SampleSizeNet *samplesize*,  
     int *ChannelsInBlock* ) [virtual]

Create a common FIFO queue for all channels. Use handle = 0 in the ChannelBlock\_ReadFrames... functions.

When using 32 bit data format, ChannelsInBlock is still the number of 16 bit channels per frame, as obtained from GetChannelsInBlock, while nChannels is the number of 32 bit channels to be read from the device. So when all channels from a device are read in 32 bit data format nChannels = ChannelsInBlock/2

## Parameters

<i>nChannels</i>	Number of channels to be collected in the FIFO.
------------------	---

## Parameters

<i>queuesize</i>	Size of sample frames the FIFO can hold.
------------------	--

## Parameters

<i>threshold</i>	Number of sample frames the FIFO must acquire before the callback function is called.
------------------	---

## Parameters

<i>samplesize</i>	size of the datawords, either 16 or 32bit.
<i>ChannelsInBlock</i>	value obtained from GetChannelsInBlock.

**11.46.3.87 SetSelectedData()** [4/4] virtual void SetSelectedData (   
     int *nChannels*,  
     int *queuesize*,  
     int *threshold*,  
     SampleSizeNet *samplesize*,  
     SampleDstSizeNet *sampleDstSize*,  
     int *ChannelsInBlock* ) [virtual]

**11.46.3.88 SetupGroupDacqQueue()** void SetupGroupDacqQueue (   
     int *queuesize*,  
     int *threshold* )

**11.46.3.89 SetVoltageRangeByIndex()** virtual void SetVoltageRangeByIndex (   
     int32\_t *voltageRangeIndex*,  
     unsigned int *virtualDevice* ) [virtual]

Sets the voltage range on devices which support multiple voltage ranges.

## Parameters

<i>voltageRangeIndex</i>	Voltage Range to use as index, smaller values are larger voltage ranges.
--------------------------	--

**11.46.3.90 SetVoltageRangeInMicroVolt()** virtual void SetVoltageRangeInMicroVolt (   
     int32\_t *voltageRange*,  
     unsigned int *virtualDevice* ) [virtual]

Sets the voltage range on devices which support multiple voltage ranges.

## Parameters

<i>voltageRange</i>	Voltage Range to use in $\mu$ V.
---------------------	----------------------------------

This replaces SetVoltageRange, where the value of the range was in mV!

**11.46.3.91 StartDacq()** [1/4] `virtual void StartDacq ( ) [virtual]`

Start the data acquisition thread and sampling.

**11.46.3.92 StartDacq()** [2/4] `virtual void StartDacq (   
int32_t timeout ) [virtual]`

Start the data acquisition thread and sampling.

Parameters

<i>timeout</i>	Timeout in ms.
----------------	----------------

**11.46.3.93 StartDacq()** [3/4] `virtual void StartDacq (   
int32_t timeout,   
int32_t numSubmittedUsbBuffers,   
int32_t numUsbBuffers,   
int32_t packetsInUrb ) [virtual]`

Start the data acquisition thread and sampling.

Parameters

<i>timeout</i>	Timeout in ms.
----------------	----------------

Parameters

<i>numSubmittedUsbBuffers</i>	Number of USB Buffers that are simultaneously submitted.
-------------------------------	--

Parameters

<i>numUsbBuffers</i>	Number of USB Buffers to use.
----------------------	-------------------------------

Parameters

<i>packetsInUrb</i>	Packets in each URB.
---------------------	----------------------

**11.46.3.94 StartDacq()** [4/4] `virtual void StartDacq (`  
    `int32_t timeout,`  
    `int32_t numSubmittedUsbBuffers,`  
    `int32_t numUsbBuffers,`  
    `int32_t packetsInUrb,`  
    `uint32_t virtualDevice ) [virtual]`

Start the data acquisition thread and sampling.

#### Parameters

<i>numSubmittedUsbBuffers</i>	Number of USB Buffers that are simultaneously submitted.
-------------------------------	--

#### Parameters

<i>timeout</i>	Timeout in ms.
----------------	----------------

#### Parameters

<i>numUsbBuffers</i>	Number of USB Buffers to use.
----------------------	-------------------------------

#### Parameters

<i>packetsInUrb</i>	Packets in each URB.
---------------------	----------------------

#### Parameters

<i>virtualDevice</i>	Virtual Device to start.
----------------------	--------------------------

**11.46.3.95 StartLoop()** [1/4] `virtual void StartLoop ( ) [virtual]`

Start the data acquisition thread.



**11.46.3.96 StartLoop()** [2/4] virtual void StartLoop (  
int32\_t timeout ) [virtual]

Start the data acquisition thread.

Parameters

<i>timeout</i>	Timeout in ms.
----------------	----------------

**11.46.3.97 StartLoop()** [3/4] virtual void StartLoop (  
int32\_t timeout,  
int32\_t numSubmittedUsbBuffers,  
int32\_t numUsbBuffers,  
int32\_t packetsInUrb ) [virtual]

Start the data acquisition thread.

Parameters

<i>timeout</i>	Timeout in ms.
----------------	----------------

Parameters

<i>numSubmittedUsbBuffers</i>	Number of USB Buffers that are simultaneously submitted.
-------------------------------	--

Parameters

<i>numUsbBuffers</i>	Number of USB Buffers to use.
----------------------	-------------------------------

Parameters

<i>packetsInUrb</i>	Packets in each URB.
---------------------	----------------------

**11.46.3.98 StartLoop()** [4/4] virtual void StartLoop (  
int32\_t timeout,

```

    int32_t numSubmittedUsbBuffers,
    int32_t numUsbBuffers,
    int32_t packetsInUrb,
    uint32_t virtualDevice ) [virtual]

```

Start the data acquisition thread.

#### Parameters

<i>numSubmittedUsbBuffers</i>	Number of USB Buffers that are simultaneously submitted.
-------------------------------	--

#### Parameters

<i>timeout</i>	Timeout in ms.
----------------	----------------

#### Parameters

<i>numUsbBuffers</i>	Number of USB Buffers to use.
----------------------	-------------------------------

#### Parameters

<i>packetsInUrb</i>	Packets in each URB.
---------------------	----------------------

#### Parameters

<i>virtualDevice</i>	Virtual Device to start.
----------------------	--------------------------

**11.46.3.99 StopDacq()** [1/2] `virtual void StopDacq ( ) [virtual]`

Stop the data acquisition thread and sampling.

**11.46.3.100 StopDacq()** [2/2] `virtual void StopDacq (`  
`uint32_t virtualDevice ) [virtual]`

Stop the data acquisition thread and sampling.

## Parameters

<i>virtualDevice</i>	Virtual Device to start.
----------------------	--------------------------

**11.46.3.101 StopLoop()** `virtual void StopLoop ( ) [virtual]`

#### 11.46.4 Member Data Documentation

**11.46.4.1 Error\_Callback\_Aquisition\_Stopped** `const int Error_Callback_Aquisition_Stopped = 0x200 [static]`

**11.46.4.2 Error\_Callback\_Data\_lost** `const int Error_Callback_Data_lost = 5 [static]`

**11.46.4.3 Error\_Callback\_Frames\_Lost** `const int Error_Callback_Frames_Lost = 4 [static]`

**11.46.4.4 Error\_Callback\_Packet\_Error** `const int Error_Callback_Packet_Error = 1 [static]`

**11.46.4.5 Error\_Callback\_Queue\_Full** `const int Error_Callback_Queue_Full = 0x100 [static]`

**11.46.4.6 Error\_Callback\_RingQueue\_Full** `const int Error_Callback_RingQueue_Full = 3 [static]`

#### 11.46.5 Property Documentation

**11.46.5.1 Samplerate** `virtual int Samplerate [get], [set]`

The sampling frequency of the device in Hz.

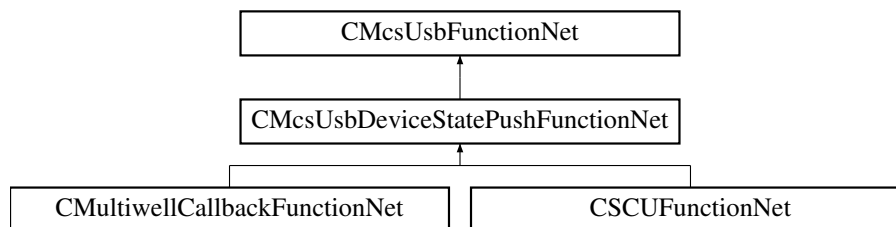
### 11.46.6 Event Documentation

**11.46.6.1 ChannelDataEvent** `OnChannelData^ ChannelDataEvent`

**11.46.6.2 ErrorEvent** `OnError^ ErrorEvent`

## 11.47 CMcsUsbDeviceStatePushFunctionNet Class Reference

Inheritance diagram for CMcsUsbDeviceStatePushFunctionNet:



### Public Member Functions

- void `TriggerStatus` ()

### Protected Member Functions

- `CMcsUsbDeviceStatePushFunctionNet` (`CMcsUsbNet^ mcsusb`, `CMcsUsbFunctionPointerContainer^ pDevice`)

### Events

- `OnMcsUsbDeviceState^ McsUsbDeviceStateEvent` [add, remove, raise]

### Additional Inherited Members

#### 11.47.1 Constructor & Destructor Documentation

**11.47.1.1 CMcsUsbDeviceStatePushFunctionNet()** `CMcsUsbDeviceStatePushFunctionNet` (`CMcsUsbNet^ mcsusb`, `CMcsUsbFunctionPointerContainer^ pDevice`) [protected]

### 11.47.2 Member Function Documentation

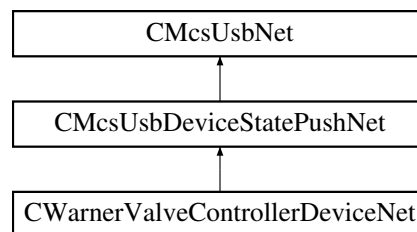
#### 11.47.2.1 TriggerStatus() `void TriggerStatus ( )`

### 11.47.3 Event Documentation

#### 11.47.3.1 McsUsbDeviceStateEvent `OnMcsUsbDeviceState^ McsUsbDeviceStateEvent [add], [remove], [raise]`

## 11.48 CMcsUsbDeviceStatePushNet Class Reference

Inheritance diagram for CMcsUsbDeviceStatePushNet:



### Public Member Functions

- `void TriggerStatus ( )`

### Protected Member Functions

- `CMcsUsbDeviceStatePushNet (CMcsUsbPointerContainer^ pDevice)`

### Events

- `OnMcsUsbDeviceState^ McsUsbDeviceStateEvent [add, remove, raise]`

### Additional Inherited Members

#### 11.48.1 Constructor & Destructor Documentation

**11.48.1.1 CMcsUsbDeviceStatePushNet()** `CMcsUsbDeviceStatePushNet ( CMcsUsbPointerContainer^ pDevice ) [protected]`

## 11.48.2 Member Function Documentation

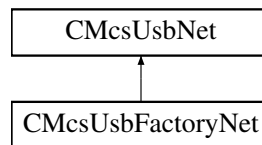
**11.48.2.1 TriggerStatus()** `void TriggerStatus ( )`

## 11.48.3 Event Documentation

**11.48.3.1 McsUsbDeviceStateEvent** `OnMcsUsbDeviceState^ McsUsbDeviceStateEvent [add], [remove], [raise]`

## 11.49 CMcsUsbFactoryNet Class Reference

Inheritance diagram for CMcsUsbFactoryNet:



### Public Member Functions

- `CMcsUsbFactoryNet ()`
- `~CMcsUsbFactoryNet ()`
- `unsigned int GetNumDestinations ()`
- `String ^ GetDestinationName (unsigned int index)`
- `String ^ GetDestinationName (CFirmwareDestinationNet dest)`
- `void SetDestinationSerialNumber (CFirmwareDestinationNet dest, String^ serialnumber)`
- `String ^ GetDestinationSerialNumber (CFirmwareDestinationNet dest)`
- `CFirmwareDestinationNet GetDestination (unsigned int index)`
- `CFirmwareDestinationNet GetDestination (String^ Key)`
- `unsigned int GetDestinationTargetAddress (CFirmwareDestinationNet destination)`  
*Gets the target base address for the destination.*
- `uint32_t ChangeSerialNumber (String^ serial)`
- `bool LoadUserFirmware (String^ FirmwareFile, CMcsUsbListEntryNet^ listEntry)`  
*Send the DSP Firmware to the MEA21 device.*
- `bool LoadUserFirmware (String^ FirmwareFile, CMcsUsbListEntryNet^ listEntry, uint32_t LockMask)`
- `bool UpdateFirmware (String^ FirmwareFile, CMcsUsbListEntryNet^ listEntry, CFirmwareDestinationNet Dest, OnUpdateFirmwareStatusChange^ deleg, OnUpdateFirmwareProgress^ progress, bool SkipWait)`  
*Flashes a firmware file to the device.*

- bool [UpdateFirmware](#) (String^ FirmwareFile, [CMcsUsbListEntryNet](#)^ listEntry, CFirmwareDestinationNet Dest, [OnUpdateFirmwareStatusChange](#)^ deleg, [OnUpdateFirmwareProgress](#)^ progress, bool SkipWait, unsigned int LockMask)
- bool [UpdateFirmware](#) (String^ FirmwareFile, [CMcsUsbListEntryNet](#)^ listEntry, CFirmwareDestinationNet dest)
- Flashes a firmware file to the device.*
- bool [UpdateFirmware](#) (String^ FirmwareFile, [CMcsUsbListEntryNet](#)^ listEntry, CFirmwareDestinationNet dest, bool SkipWait)
- Flashes a firmware file to the device.*
- bool [UpdateFirmware](#) (String^ FirmwareFile, [CMcsUsbListEntryNet](#)^ listEntry, CFirmwareDestinationNet dest, bool SkipWait, uint32\_t LockMask)
- bool [CompareFirmware](#) (String^ FirmwareFile, [CMcsUsbListEntryNet](#)^ listEntry, CFirmwareDestinationNet Dest, [OnUpdateFirmwareStatusChange](#)^ deleg, [OnUpdateFirmwareProgress](#)^ progress, String^ MessagePrefix, unsigned int LockMask, [System::Runtime::InteropServices::Out] String^% ErrorText, [System::Runtime::InteropServices::Out] String^% Protokoll)
- uint32\_t [Coldstart](#) (CFirmwareDestinationNet dest)
- int32\_t [GetXilinxFlashOffset](#) (CFirmwareDestinationNet dest)
- uint32\_t [GetXilinxFlashReadCommand](#) (CFirmwareDestinationNet dest)
- array< uint8\_t > ^ [DownloadFirmware](#) (CFirmwareDestinationNet Dest, uint32\_t Address, uint32\_t length)
- bool [GetUserCodeFromFlash](#) (unsigned int FPGA, unsigned int Address, [System::Runtime::InteropServices::Out] unsigned int% Usercode)
- array< unsigned char > ^ [ReadBlockFromFlash](#) (unsigned int FPGA, unsigned int Address)
- void [ReadBlockFromFlash](#) (unsigned int FPGA, unsigned int Address, array< unsigned char >^ buffer, int position)
- array< unsigned char > ^ [ReadBlockFromIFBGlobalEEPROM](#) (unsigned int Address)
- array< unsigned char > ^ [ReadBlockFromNVMEM](#) (unsigned int FPGA, unsigned int Offset, unsigned int Address)

### Static Public Member Functions

- static String ^ [GetDestinationDisplayLabel](#) (String^ RawLabel, CFirmwareDestinationNet dest)
- static String ^ [FindFirmwareVersionMagicInBuffer](#) (array< unsigned char >^ buffer, int length, [System::Runtime::InteropServices::Out] int% position)
- static bool [GetFirmwareVersionFromFile](#) (String^ FirmwareFile, [System::Runtime::InteropServices::Out] uint32\_t% Version)
- Retrives version info from a Firmware update file.*
- static bool [GetFirmwareVersionFromFile](#) (String^ FirmwareFile, [System::Runtime::InteropServices::Out] uint32\_t% Version, [System::Runtime::InteropServices::Out] uint32\_t% Position)
- static bool [GetFirmwareVersionFromHexFile](#) (String^ FirmwareFile, [System::Runtime::InteropServices::Out] uint32\_t% Version)
- static uint32\_t [GetChecksumFromFX3Image](#) (String^ FirmwareFile)
- static uint32\_t [GetUSBDeviceIDFromFX3Image](#) (String^ FirmwareFile)
- static bool [GetUserCodeFromBitFile](#) (String^ FirmwareFile, [System::Runtime::InteropServices::Out] unsigned int% Usercode)

### Static Public Attributes

- static const uint32\_t [FX3MCSDDataAddress](#) = 0x40037E00
- static const uint32\_t [FX3MCSDDataDeviceIDOffset](#) = 0x4
- static const uint32\_t [FX3MCSDDataVersionOffset](#) = 0x8

## Additional Inherited Members

### 11.49.1 Constructor & Destructor Documentation

**11.49.1.1 CMcsUsbFactoryNet()** `CMcsUsbFactoryNet ( )`

**11.49.1.2 ~CMcsUsbFactoryNet()** `~CMcsUsbFactoryNet ( )`

### 11.49.2 Member Function Documentation

**11.49.2.1 ChangeSerialNumber()** `uint32_t ChangeSerialNumber ( String^ serial )`

**11.49.2.2 Coldstart()** `uint32_t Coldstart ( CFirmwareDestinationNet dest )`

**11.49.2.3 CompareFirmware()** `bool CompareFirmware ( String^ FirmwareFile, CMcsUsbListEntryNet^ listEntry, CFirmwareDestinationNet Dest, OnUpdateFirmwareStatusChange^ deleg, OnUpdateFirmwareProgress^ progress, String^ MessagePrefix, unsigned int LockMask, [System::Runtime::InteropServices::Out] String^% ErrorText, [System::Runtime::InteropServices::Out] String^% Protokoll )`

**11.49.2.4 DownloadFirmware()** `array<uint8_t> ^ DownloadFirmware ( CFirmwareDestinationNet Dest, uint32_t Address, uint32_t length )`



**11.49.2.5 FindFirmwareVersionMagicInBuffer()** static String ^ FindFirmwareVersionMagicInBuffer ( array< unsigned char >^ *buffer*, int *length*, [System::Runtime::InteropServices::Out] int% *position* ) [static]

**11.49.2.6 GetChecksumFromFX3Image()** static uint32\_t GetChecksumFromFX3Image ( String^ *FirmwareFile* ) [static]

**11.49.2.7 GetDestination()** [1/2] CFirmwareDestinationNet GetDestination ( String^ *Key* )

**11.49.2.8 GetDestination()** [2/2] CFirmwareDestinationNet GetDestination ( unsigned int *index* )

**11.49.2.9 GetDestinationDisplayLabel()** static String ^ GetDestinationDisplayLabel ( String^ *RawLabel*, CFirmwareDestinationNet *dest* ) [static]

**11.49.2.10 GetDestinationName()** [1/2] String ^ GetDestinationName ( CFirmwareDestinationNet *dest* )

**11.49.2.11 GetDestinationName()** [2/2] String ^ GetDestinationName ( unsigned int *index* )

**11.49.2.12 GetDestinationSerialNumber()** String ^ GetDestinationSerialNumber ( CFirmwareDestinationNet *dest* )

**11.49.2.13 GetDestinationTargetAddress()** unsigned int GetDestinationTargetAddress ( CFirmwareDestinationNet *destination* )

Gets the target base address for the destination.

## Parameters

<i>destination</i>	The destination to be queried.
--------------------	--------------------------------

## Returns

The base address as a 32 bit number, only the lower 16 bit represent the address.

**11.49.2.14 GetFirmwareVersionFromFile()** [1/2] static bool GetFirmwareVersionFromFile (   
     String^ *FirmwareFile*,   
     [System::Runtime::InteropServices::Out] uint32\_t% *Version* ) [static]

Retrives version info from a Firmware update file.

**11.49.2.15 GetFirmwareVersionFromFile()** [2/2] static bool GetFirmwareVersionFromFile (   
     String^ *FirmwareFile*,   
     [System::Runtime::InteropServices::Out] uint32\_t% *Version*,   
     [System::Runtime::InteropServices::Out] uint32\_t% *Position* ) [static]

**11.49.2.16 GetFirmwareVersionFromHexFile()** static bool GetFirmwareVersionFromHexFile (   
     String^ *FirmwareFile*,   
     [System::Runtime::InteropServices::Out] uint32\_t% *Version* ) [static]

**11.49.2.17 GetNumDestinations()** unsigned int GetNumDestinations ( )

**11.49.2.18 GetUSBDeviceIDFromFX3Image()** static uint32\_t GetUSBDeviceIDFromFX3Image (   
     String^ *FirmwareFile* ) [static]

**11.49.2.19 GetUsercodeFromBitFile()** static bool GetUsercodeFromBitFile (   
     String^ *FirmwareFile*,   
     [System::Runtime::InteropServices::Out] unsigned int% *Usercode* ) [static]

**11.49.2.20 GetUsercodeFromFlash()** `bool GetUsercodeFromFlash ( unsigned int FPGA, unsigned int Address, [System::Runtime::InteropServices::Out] unsigned int% Usercode )`

**11.49.2.21 GetXilinxFlashOffset()** `int32_t GetXilinxFlashOffset ( CFirmwareDestinationNet dest )`

**11.49.2.22 GetXilinxFlashReadCommand()** `uint32_t GetXilinxFlashReadCommand ( CFirmwareDestinationNet dest )`

**11.49.2.23 LoadUserFirmware()** [1/2] `bool LoadUserFirmware ( String^ FirmwareFile, CMcsUsbListEntryNet^ listEntry )`

Send the DSP Firmware to the MEA21 device.

#### Parameters

<i>FirmwareFile</i>	Filename of the DSP Firmware (*.bin) file.
---------------------	--

#### Parameters

<i>listEntry</i>	Device to use for the connection. See <a href="#">CMcsUsbListNet</a> .
------------------	--

**11.49.2.24 LoadUserFirmware()** [2/2] `bool LoadUserFirmware ( String^ FirmwareFile, CMcsUsbListEntryNet^ listEntry, uint32_t LockMask )`

**11.49.2.25 ReadBlockFromFlash()** [1/2] `array<unsigned char> ^ ReadBlockFromFlash ( unsigned int FPGA, unsigned int Address )`

**11.49.2.26 ReadBlockFromFlash()** [2/2] void ReadBlockFromFlash (   
     unsigned int *FPGA*,   
     unsigned int *Address*,   
     array< unsigned char >^ *buffer*,   
     int *position* )

**11.49.2.27 ReadBlockFromIFBGlobalEEProm()** array<unsigned char> ^ ReadBlockFromIFBGlobalEEProm (   
     unsigned int *Address* )

**11.49.2.28 ReadBlockFromNVMEM()** array<unsigned char> ^ ReadBlockFromNVMEM (   
     unsigned int *FPGA*,   
     unsigned int *Offset*,   
     unsigned int *Address* )

**11.49.2.29 SetDestinationSerialNumber()** void SetDestinationSerialNumber (   
     CFirmwareDestinationNet *dest*,   
     String^ *serialnumber* )

**11.49.2.30 UpdateFirmware()** [1/5] bool UpdateFirmware (   
     String^ *FirmwareFile*,   
     CMcsUsbListEntryNet^ *listEntry*,   
     CFirmwareDestinationNet *dest* )

Flashes a firmware file to the device.

#### Parameters

<i>FirmwareFile</i>	Filename of the Firmware file.
---------------------	--------------------------------

#### Parameters

<i>listEntry</i>	Device to use for the connection.
------------------	-----------------------------------

**11.49.2.31 UpdateFirmware()** [2/5] bool UpdateFirmware (   
     String^ *FirmwareFile*,

```
CMcsUsbListEntryNet^ listEntry,
CFirmwareDestinationNet dest,
bool SkipWait )
```

Flashes a firmware file to the device.

#### Parameters

<i>FirmwareFile</i>	Filename of the Firmware file.
---------------------	--------------------------------

#### Parameters

<i>listEntry</i>	Device to use for the connection.
------------------	-----------------------------------

**11.49.2.32 UpdateFirmware() [3/5]** bool UpdateFirmware (   
String^ FirmwareFile,   
CMcsUsbListEntryNet^ listEntry,   
CFirmwareDestinationNet dest,   
bool SkipWait,   
uint32\_t LockMask )

**11.49.2.33 UpdateFirmware() [4/5]** bool UpdateFirmware (   
String^ FirmwareFile,   
CMcsUsbListEntryNet^ listEntry,   
CFirmwareDestinationNet Dest,   
OnUpdateFirmwareStatusChange^ deleg,   
OnUpdateFirmwareProgress^ progress,   
bool SkipWait )

Flashes a firmware file to the device.

#### Parameters

<i>FirmwareFile</i>	Filename of the Firmware file.
---------------------	--------------------------------

**11.49.2.34 UpdateFirmware() [5/5]** bool UpdateFirmware (   
String^ FirmwareFile,   
CMcsUsbListEntryNet^ listEntry,   
CFirmwareDestinationNet Dest,   
OnUpdateFirmwareStatusChange^ deleg,

```
OnUpdateFirmwareProgress^ progress,  
bool SkipWait,  
unsigned int LockMask )
```

### 11.49.3 Member Data Documentation

### 11.49.3.1 FX3MCSDDataAddress

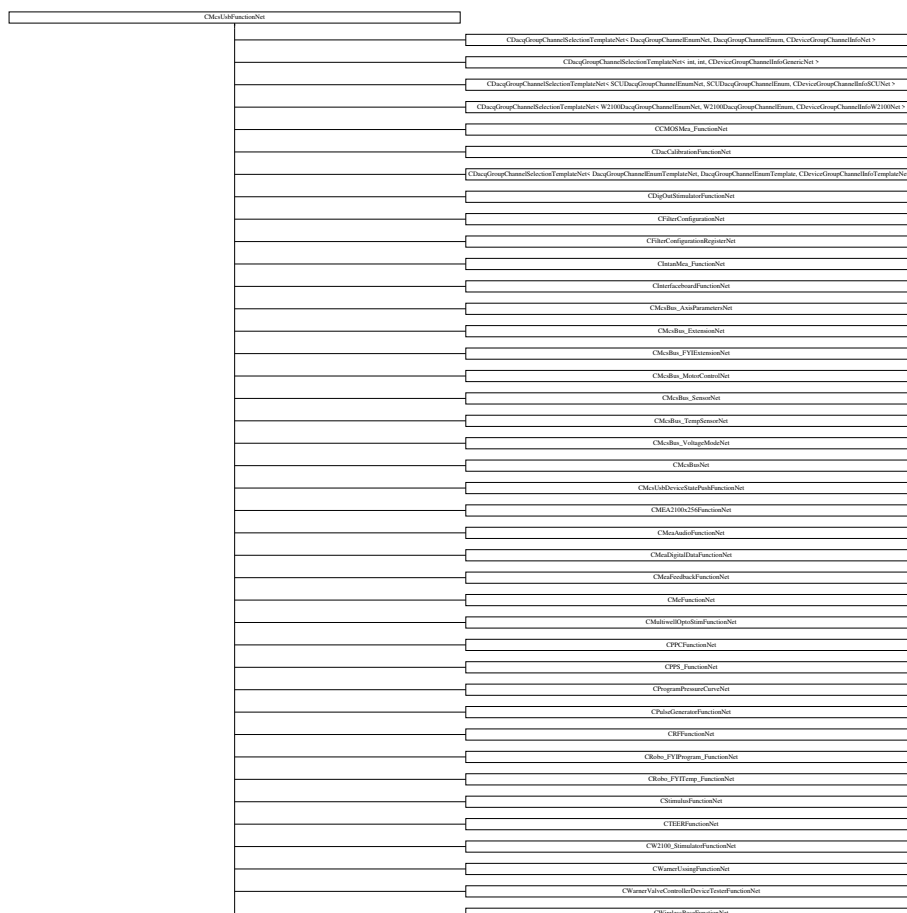
### 11.49.3.2 FX3MCSDataDeviceIdOffset

### 11.49.3.3 FX3MCSDataVersionOffset

```
const uint32_t FX3MCSDataVersionOffset = 0x8 [static]
```

## 11.50 CMcsUsbFunctionNet Class Reference

Inheritance diagram for CMcsUsbFunctionNet:



**Public Member Functions**

- [CMcsUsbFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- virtual [~CMcsUsbFunctionNet](#) (void)
- [!CMcsUsbFunctionNet](#) ()
- void [ThrowCUsbExceptionNetOnError](#) (uint32\_t status)

**Protected Member Functions**

- [CMcsUsbFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> mcsusbfunction)

**Protected Attributes**

- [CMcsUsbNet](#)<sup>^</sup> [m\\_pMcsUsb](#)
- [CMcsUsbFunction](#) \* [m\\_pMcsUsbFunction](#)

**11.50.1 Constructor & Destructor Documentation**

**11.50.1.1 [CMcsUsbFunctionNet\(\)](#)** [1/2] [CMcsUsbFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> *mcsusb* )

**11.50.1.2 [~CMcsUsbFunctionNet\(\)](#)** virtual [~CMcsUsbFunctionNet](#) (  
void ) [virtual]

**11.50.1.3 ["!CMcsUsbFunctionNet\(\)](#)** [!CMcsUsbFunctionNet](#) ( )

**11.50.1.4 [CMcsUsbFunctionNet\(\)](#)** [2/2] [CMcsUsbFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> *mcsusb*,  
[CMcsUsbFunctionPointerContainer](#)<sup>^</sup> *mcsusbfunction* ) [protected]

**11.50.2 Member Function Documentation**

**11.50.2.1 [ThrowCUsbExceptionNetOnError\(\)](#)** void [ThrowCUsbExceptionNetOnError](#) (  
uint32\_t *status* )

### 11.50.3 Member Data Documentation

**11.50.3.1** `m_pMcsUsb` [CMcsUsbNet](#) ^ `m_pMcsUsb` [protected]

**11.50.3.2** `m_pMcsUsbFunction` `CMcsUsbFunction*` `m_pMcsUsbFunction` [protected]

## 11.51 CMcsUsbFunctionPointerContainer Class Reference

## 11.52 CMcsUsbListEntryNet Class Reference

McsUsbListEntryNet identifies a connected device.

### Public Member Functions

- `~CMcsUsbListEntryNet` ()
- virtual bool `Equals` (Object^ obj) override  
*Checks weather two [CMcsUsbListEntryNet](#) represent the same USB device.*
- void `SetStringFormat` (String ^ format)  
*Specify the text the [CMcsUsbListEntryNet.ToString\(\)](#) function should return. The special code N expands to the device name and S expands to the serial number of the device.*
- virtual String ^ `ToString` () override

### Static Public Member Functions

- static `CMcsUsbListEntryNet` ^ `GetEntry` ()  
*Returns one [CMcsUsbListEntryNet](#) from the list of USB Devices connected to the computer.*
- static `CMcsUsbListEntryNet` ^ `GetEntry` (DeviceEnumNet McsUsbDevice)  
*Returns one [CMcsUsbListEntryNet](#) from the list of USB Devices connected to the computer.*
- static `CMcsUsbListEntryNet` ^ `GetEntry` (DeviceEnumNet McsUsbDevice, unsigned int index)  
*Returns one [CMcsUsbListEntryNet](#) from the list of USB Devices connected to the computer.*
- static unsigned int `GetEntryCount` ()  
*Returns the number of devices connected to the computer.*
- static unsigned int `GetEntryCount` (DeviceEnumNet McsUsbDevice)  
*Returns the number of devices connected to the computer.*

### Properties

- String^ `Manufacturer` [get]  
*The Manufacturer ID of the device represented by this [CMcsUsbListEntryNet](#).*
- String^ `Product` [get]  
*The Product ID of the device represented by this [CMcsUsbListEntryNet](#).*
- String^ `DeviceName` [get]  
*The device name of the device represented by this [CMcsUsbListEntryNet](#).*
- String^ `SerialNumber` [get]  
*The serial number of the device represented by this [CMcsUsbListEntryNet](#).*
- String^ `HwVersion` [get]  
*The hardware revision of the device represented by this [CMcsUsbListEntryNet](#).*
- `DeviceldNet`^ `Deviceld` [get]



### 11.52.1 Detailed Description

CMcsUsbListEntryNet identifies a connected device.

### 11.52.2 Constructor & Destructor Documentation

**11.52.2.1** `~CMcsUsbListEntryNet()` `~CMcsUsbListEntryNet ( )`

### 11.52.3 Member Function Documentation

**11.52.3.1** `Equals()` `virtual bool Equals (`  
`Object^ obj ) [override], [virtual]`

Checks weather two [CMcsUsbListEntryNet](#) represent the same USB device.

Parameters

<i>obj</i>	The <a href="#">CMcsUsbListEntryNet</a> to compare with.
------------	--

**11.52.3.2** `GetEntry()` `[1/3] static CMcsUsbListEntryNet ^ GetEntry ( ) [static]`

Returns one [CMcsUsbListEntryNet](#) from the list of USB Devices connected to the computer.

Returns

A [CMcsUsbListEntryNet](#) to be used to connect to the device.

**11.52.3.3** `GetEntry()` `[2/3] static CMcsUsbListEntryNet ^ GetEntry (`  
`DeviceEnumNet McsUsbDevice ) [static]`

Returns one [CMcsUsbListEntryNet](#) from the list of USB Devices connected to the computer.

Parameters

<i>McsUsbDevice</i>	Specifies the type of devices to look for.
---------------------	--

**Returns**

A [CMcsUsbListEntryNet](#) to be used to connect to the device.

**11.52.3.4 GetEntry()** [3/3] `static CMcsUsbListEntryNet ^ GetEntry ( DeviceEnumNet McsUsbDevice, unsigned int index ) [static]`

Returns one [CMcsUsbListEntryNet](#) from the list of USB Devices connected to the computer.

**Parameters**

<i>McsUsbDevice</i>	Specifies the type of devices to look for.
---------------------	--

**Parameters**

<i>index</i>	number of the entry to use.
--------------	-----------------------------

**Returns**

A [CMcsUsbListEntryNet](#) to be used to connect to the device.

**11.52.3.5 GetEntryCount()** [1/2] `static unsigned int GetEntryCount ( ) [static]`

Returns the number of devices connected to the computer.

**Returns**

The number of devices.

**11.52.3.6 GetEntryCount()** [2/2] `static unsigned int GetEntryCount ( DeviceEnumNet McsUsbDevice ) [static]`

Returns the number of devices connected to the computer.

## Parameters

<i>McsUsbDevice</i>	Specifies the type of devices to look for.
---------------------	--

## Returns

The number of devices.

**11.52.3.7 SetStringFormat()** `void SetStringFormat ( String ^ format )`

Specify the text the [CMcsUsbListEntryNet.ToString\(\)](#) function should return. The special code N expands to the device name and S expands to the serial number of the device.

## Parameters

<i>format</i>	A String containing the format template. Default is "%N (%S)".
---------------	--

**11.52.3.8 ToString()** `virtual String ^ ToString ( ) [override], [virtual]`

**11.52.4 Property Documentation**

**11.52.4.1 DeviceId** `DeviceIdNet^ DeviceId [get]`

**11.52.4.2 DeviceName** `String^ DeviceName [get]`

The device name of the device represented by this [CMcsUsbListEntryNet](#).

**11.52.4.3 HwVersion** `String^ HwVersion [get]`

The hardware revision of the device represented by this [CMcsUsbListEntryNet](#).

**11.52.4.4 Manufacturer** `String^ Manufacturer [get]`

The Manufacturer ID of the device represented by this [CMcsUsbListEntryNet](#).

**11.52.4.5 Product** `String^ Product [get]`

The Product ID of the device represented by this [CMcsUsbListEntryNet](#).

**11.52.4.6 SerialNumber** `String^ SerialNumber [get]`

The serial number of the device represented by this [CMcsUsbListEntryNet](#).

**11.53 CMcsUsbListNet Class Reference**

Class to handle a list of connected MCS USB devices.

**Public Member Functions**

- [CMcsUsbListNet](#) (DeviceEnumNet McsUsbDevice)  
*Initializes a new instance of [CMcsUsbListNet](#) class.*
- [CMcsUsbListNet](#) (array< [DeviceIdNet](#) ^> ^ DeviceIdList)  
*Initializes a new instance of [CMcsUsbListNet](#) class.*
- [~CMcsUsbListNet](#) ()  
*Destructor: called by [Dispose\(\)](#)*
- [!CMcsUsbListNet](#) ()  
*Finalizer: called by GC before collecting*
- void [SetStringFormat](#) (String ^ format)  
*Specify the text the [CMcsUsbListEntryNet.ToString\(\)](#) function should return. The special code N expands to the device name and S expands to the serial number of the device.*
- uint32\_t [GetNumberOfDevices](#) ()  
*Gets the number of devices currently in the list.*
- [CMcsUsbListEntryNet](#) ^ [GetUsbListEntry](#) (unsigned int index)  
*Returns one [CMcsUsbListEntryNet](#) from the list of USB Devices connected to the computer.*
- array< [CMcsUsbListEntryNet](#) ^> ^ [GetUsbListEntries](#) ()  
*Returns all entries from the list of USB Devices connected to the computer.*
- bool [IsDeviceTypeOf](#) ([CMcsUsbListEntryNet](#) ^ entry, DeviceEnumNet McsUsbDevice)

**Properties**

- uint32\_t [Count](#) [get]  
*Gets the number of devices currently in the list.*

**Events**

- [OnDeviceArrivalRemoval](#)^ [DeviceArrival](#)
- [OnDeviceArrivalRemoval](#)^ [DeviceRemoval](#)

**11.53.1 Detailed Description**

Class to handle a list of connected MCS USB devices.

**11.53.2 Constructor & Destructor Documentation**

**11.53.2.1 CMcsUsbListNet()** [1/2] [CMcsUsbListNet](#) (   
 [DeviceEnumNet](#) *McsUsbDevice* )

Initializes a new instance of [CMcsUsbListNet](#) class.

**11.53.2.2 CMcsUsbListNet()** [2/2] [CMcsUsbListNet](#) (   
 [array](#)< [DeviceIdNet](#)^>^ [DeviceIdList](#) )

Initializes a new instance of [CMcsUsbListNet](#) class.

**11.53.2.3 ~CMcsUsbListNet()** [~CMcsUsbListNet](#) ( )

Destructor: called by Dispose()

**11.53.2.4 ~!CMcsUsbListNet()** [!CMcsUsbListNet](#) ( )

Finalizer: called by GC before collecting

**11.53.3 Member Function Documentation**

**11.53.3.1 GetNumberOfDevices()** [uint32\\_t](#) [GetNumberOfDevices](#) ( )

Gets the number of devices currently in the list.

**Returns**

The number of devices currently in the list.

**11.53.3.2 GetUsbListEntries()** [array](#)<[CMcsUsbListEntryNet](#)^> ^ [GetUsbListEntries](#) ( )

Returns all entries from the list of USB Devices connected to the computer.

**11.53.3.3 GetUsbListEntry()** [CMcsUsbListEntryNet](#) ^ [GetUsbListEntry](#) (   
 [unsigned int](#) *index* )

Returns one [CMcsUsbListEntryNet](#) from the list of USB Devices connected to the computer.

## Parameters

<i>index</i>	number of the entry to use.
--------------	-----------------------------

**11.53.3.4 IsDeviceTypeOf()** `bool IsDeviceTypeOf (`  
     `CMcsUsbListEntryNet^ entry,`  
     `DeviceEnumNet McsUsbDevice )`

**11.53.3.5 SetStringFormat()** `void SetStringFormat (`  
     `String ^ format )`

Specify the text the [CMcsUsbListEntryNet.ToString\(\)](#) function should return. The special code N expands to the device name and S expands to the serial number of the device.

## Parameters

<i>format</i>	A String containing the format template. Default is "%N (%S)".
---------------	--

**11.53.4 Property Documentation**

**11.53.4.1 Count** `uint32_t Count [get]`

Gets the number of devices currently in the list.

**11.53.5 Event Documentation**

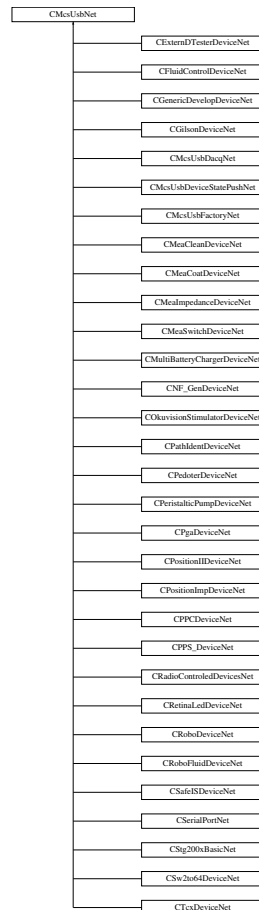
**11.53.5.1 DeviceArrival** `OnDeviceArrivalRemoval^ DeviceArrival`

**11.53.5.2 DeviceRemoval** `OnDeviceArrivalRemoval^ DeviceRemoval`

## 11.54 CMcsUsbNet Class Reference

Base class to handle MCS USB devices. All device classes are derived from this class. Functionality that is provided by all MCS devices is handled by this class.

Inheritance diagram for CMcsUsbNet:



### Public Member Functions

- [CMcsUsbNet](#) ()  
*Initializes a new instance of the base class to handle MCS USB devices.*
- [CMcsUsbNet](#) (McsBusTypeEnumNet bustype)  
*Initializes a new instance of the base class to handle MCS USB devices.*
- virtual [~CMcsUsbNet](#) ()
- [!CMcsUsbNet](#) ()
- DeviceEnumNet [GetDeviceEnum](#) ()
- virtual uint32\_t [Connect](#) ()  
*Opens a connection to the device.*
- virtual uint32\_t [Connect](#) (unsigned int LockMask)  
*Opens a connection to the device.*
- virtual uint32\_t [Connect](#) ([CMcsUsbListEntryNet](#)^ entry)  
*Opens a connection to the device.*
- virtual uint32\_t [Connect](#) ([CMcsUsbListEntryNet](#)^ entry, unsigned int LockMask)  
*Opens a connection to the device.*

- virtual uint32\_t [GetStatus](#) ([System::Runtime::InteropServices::Out]uint32\_t% iStatus)
- virtual bool [IsConnected](#) ()  
*Check if a device is Connected.*
- virtual void [Disconnect](#) ()  
*Disconnect from a device.*
- [CMcsUsbListEntryNet](#) ^ [GetUsbListEntry](#) ()
- virtual String ^ [GetSerialNumber](#) ()  
*Query the Serial Number of the device.*
- [DriverVersionNet](#) ^ [GetVersion](#) ()
- [DriverVersionNet](#) ^ [GetVersion](#) (CFirmwareDestinationNet dest)
- [DeviceldNet](#) ^ [GetDeviceld](#) ()
- uint32\_t [GetIdent](#) ([System::Runtime::InteropServices::Out]String^% Answer)
- void [MultibootSelectImage](#) (unsigned int sector)  
*Select the multiboot image specified by "sector" (range: 0..2) for IFB FPGA.*
- String ^ [MultibootGetImageld](#) (unsigned int sector)  
*Query the multiboot image id of the device located in specified sector (range: 0..2 / 0..9) of IFB1 / IFB2 FPGA.*
- uint32\_t [MultibootGetCypressImageld](#) (unsigned int sector)  
*Query the multiboot image id of the device located in specified sector (range: 0..9 0..9) of IFB2 Cypress.*
- uint32\_t [MultibootGetSelectedImage](#) ()  
*Gets sector index of selected FPGA boot image on IFB*
- uint32\_t [GetMea21UsbPort](#) ()  
*Gets the USB port if an IFB that is used by this connection*
- HeadstageIdEnumNet [GetHeadstageID](#) (uint32\_t headstage)  
*Gets the ID of a connected headstage.*
- bool [GetHeadstagePresent](#) (uint32\_t headstage)  
*queries whether a headstage is present*
- bool [GetHeadstageActive](#) (uint32\_t headstage)  
*queries whether a headstage is active*
- void [RescanHeadstage](#) (uint32\_t headstage)  
*rescans and activates a headstage*
- array< BYTE > ^ [GetSoftwareKey](#) (unsigned int index)
- void [SetSoftwareKey](#) (unsigned int index, array< BYTE >^ buffer)
- void [RemoveSoftwareKey](#) (unsigned int index)
- void [AddSoftwareKey](#) (String^ key)
- bool [EmptyKey](#) (String^ key)
- bool [ValidKey](#) (String^ key, [System::Runtime::InteropServices::Out]String^% serial\_number)
- bool [ValidKey](#) (String^ key, uint8\_t ProgrammID, uint8\_t majorversion, [System::Runtime::InteropServices::Out]String^% serial\_number)
- bool [HasSoftwareKey](#) (uint8\_t ProgrammID, uint8\_t majorversion)
- bool [HasSoftwareKey](#) (SoftwareKeyProgrammIdsNet::ProgrammIdsNet ProgrammID, uint8\_t majorversion)
- String ^ [GetSoftwareKeyString](#) (uint8\_t ProgrammID, uint8\_t majorversion)
- String ^ [GetSoftwareKeyString](#) (SoftwareKeyProgrammIdsNet::ProgrammIdsNet ProgrammID, uint8\_t majorversion)
- bool [IsDeviceHighSpeedCapable](#) ()
- bool [IsDeviceHighSpeed](#) ()
- McsUsbSpeedEnumNet [GetDeviceCapableSpeed](#) ()
- McsUsbSpeedEnumNet [GetDeviceSpeed](#) ()  
*Query the Connection Speed of the device.*
- unsigned int [TxnTestMemoryWrite](#) (unsigned short index)
- unsigned int [TxnTestMemoryReadAndCheck](#) (unsigned short index)
- void [TxnSetSerialNumber](#) (unsigned int number)
- unsigned int [TxnGetSerialNumber](#) ()
- unsigned int [ReadRegister](#) (unsigned int reg)



- array< uint32\_t > ^ [ReadRegister](#) (unsigned int reg, int length)
- unsigned int [ReadRegister32](#) (unsigned int adr)
- unsigned int [ReadRegisterTimeSlot](#) (unsigned int reg, int TimeSlot)
- void [WriteRegister](#) (unsigned int reg, unsigned int value)
- void [WriteRegisterValue](#) (unsigned int reg, unsigned int value)
- void [WriteRegister32](#) (unsigned int adr, unsigned int value)
- void [WriteRegister](#) (unsigned int reg, array< unsigned int >^ values)
- void [WriteRegisterArray](#) (unsigned int reg, array< unsigned int >^ values)
- void [WriteRegisterTimeSlot](#) (unsigned int reg, unsigned int value, int TimeSlot)
- void [WriteRegisterTimeSlot](#) (unsigned int reg, array< unsigned int >^ values, int TimeSlot)
- bool [ReadEepromRegisterPreconfig](#) (uint32\_t EEPROMBase, uint32\_t DMA\_reg, [System::Runtime::InteropServices::Out]uint32\_t% DMA\_value)
- bool [ReadEepromRegisterPreconfig](#) (uint32\_t EEPROMBase, uint32\_t DMA\_reg, [System::Runtime::InteropServices::Out]uint32\_t% DMA\_value, uint32\_t EEPROMSize)
- bool [ReadEepromRegisterPreconfig](#) (uint32\_t EEPROMBase, uint32\_t DMA\_reg, [System::Runtime::InteropServices::Out]uint32\_t% DMA\_value, uint32\_t EEPROMSize, uint32\_t EepromStartAddress)
- void [WriteEepromRegisterPreconfig](#) (uint32\_t EEPROMBase, uint32\_t DMA\_reg, uint32\_t DMA\_value)
- void [WriteEepromRegisterPreconfig](#) (uint32\_t EEPROMBase, uint32\_t DMA\_reg, uint32\_t DMA\_value, uint32\_t EEPROMSize)
- void [WriteEepromRegisterPreconfig](#) (uint32\_t EEPROMBase, uint32\_t DMA\_reg, uint32\_t DMA\_value, uint32\_t EEPROMSize, uint32\_t EepromStartAddress)
- void [EraseEepromRegisterPreconfig](#) (uint32\_t EEPROMBase, uint32\_t DMA\_reg)
- void [EraseEepromRegisterPreconfig](#) (uint32\_t EEPROMBase, uint32\_t DMA\_reg, uint32\_t EEPROMSize)
- void [EraseEepromRegisterPreconfig](#) (uint32\_t EEPROMBase, uint32\_t DMA\_reg, uint32\_t EEPROMSize, uint32\_t EepromStartAddress)
- unsigned int [GetLastUSBError](#) ()
- void [ThrowCUsbExceptionNetOnError](#) (uint32\_t status)
- uint32\_t [GetDeviceCannotStallOutRequests](#) ()
- String ^ [GetHardwareRevision](#) ()
- unsigned int [GetFirmwareVersion](#) (CFirmwareDestinationNet destination)  
*Gets the firmware version for the destination.*
- uint8\_t [GetNumConfigurations](#) ()
- uint8\_t [GetConfiguration](#) ()
- void [SetConfiguration](#) (uint8\_t config)
- uint32\_t [GetDeviceRootHubVendorID](#) ()  
*Gets the Vendor ID of the USB root hub the device is connected to.*
- UsbVendorIdEnumNet [GetDeviceRootHubVendorEnum](#) ()  
*Gets the Vendor ID of the USB root hub the device is connected to.*
- String ^ [GetDeviceRootHubVendorName](#) ()  
*Gets the Vendor Name of the USB root hub the device is connected to.*
- void [EnableExceptions](#) (bool enable)  
*Enables or Disables Exceptions for calls to McsUsb Devices. If Exceptions are disabled, the return value of a command can be queries with the GetStatusOfLastCommand call instead.*
- bool [IsExceptionsEnabled](#) ()
- uint32\_t [GetStatusOfLastCommand](#) ()  
*Gets the status of the last call to the McsUsb Library.*
- void [AssociateToThis](#) (CMcsUsbNet^ device)

### Static Public Member Functions

- static String ^ [GetErrorText](#) (unsigned int Status)  
*Gets the error text string that belongs to a status number.*

### Static Public Attributes

- static const uint32\_t [Status\\_Crc](#) = (0xE0100001L)
- static const uint32\_t [Status\\_Btstuff](#) = (0xE0100002L)
- static const uint32\_t [Status\\_DataToggleMismatch](#) = (0xE0100003L)
- static const uint32\_t [Status\\_Stall](#) = (0xE0100004L)
- static const uint32\_t [Status\\_DevNotResponding](#) = (0xE0100005L)
- static const uint32\_t [Status\\_PidCheckFailure](#) = (0xE0100006L)
- static const uint32\_t [Status\\_UnexpectedPid](#) = (0xE0100007L)
- static const uint32\_t [Status\\_DataOverrun](#) = (0xE0100008L)
- static const uint32\_t [Status\\_DataUnderrun](#) = (0xE0100009L)
- static const uint32\_t [Status\\_BufferOverrun](#) = (0xE010000CL)
- static const uint32\_t [Status\\_BufferUnderrun](#) = (0xE010000DL)
- static const uint32\_t [Status\\_NotAccessed](#) = (0xE010000FL)
- static const uint32\_t [Status\\_Fifo](#) = (0xE0100010L)
- static const uint32\_t [Status\\_EndpointHalted](#) = (0xE0100030L)
- static const uint32\_t [Status\\_NoMemory](#) = (0xE0100100L)
- static const uint32\_t [Status\\_InvalidUrbFunction](#) = (0xE0100200L)
- static const uint32\_t [Status\\_InvalidParameter](#) = (0xE0100300L)
- static const uint32\_t [Status\\_ErrorBusy](#) = (0xE0100400L)
- static const uint32\_t [Status\\_RequestFailed](#) = (0xE0100500L)
- static const uint32\_t [Status\\_InvalidPipeHandle](#) = (0xE0100600L)
- static const uint32\_t [Status\\_NoBandwidth](#) = (0xE0100700L)
- static const uint32\_t [Status\\_InternalHcError](#) = (0xE0100800L)
- static const uint32\_t [Status\\_ErrorShortTransfer](#) = (0xE0100900L)
- static const uint32\_t [Status\\_BadStartFrame](#) = (0xE0100A00L)
- static const uint32\_t [Status\\_IsochRequestFailed](#) = (0xE0100B00L)
- static const uint32\_t [Status\\_FrameControlOwned](#) = (0xE0100C00L)
- static const uint32\_t [Status\\_ControlNotOwned](#) = (0xE0100D00L)
- static const uint32\_t [Status\\_Canceled](#) = (0xE0110000L)
- static const uint32\_t [Status\\_Canceling](#) = (0xE0120000L)
- static const uint32\_t [Status\\_AlreadyConfigured](#) = (0xE0110001L)
- static const uint32\_t [Status\\_Unconfigured](#) = (0xE0110002L)
- static const uint32\_t [Status\\_NoSuchDevice](#) = (0xE01F0002L)
- static const uint32\_t [Status\\_DeviceNotFound](#) = (0xE01F0003L)
- static const uint32\_t [Status\\_NotSupported](#) = (0xE01F0005L)
- static const uint32\_t [Status\\_IoPending](#) = (0xE01F0006L)
- static const uint32\_t [Status\\_IoTimeout](#) = (0xE01F0007L)
- static const uint32\_t [Status\\_DeviceRemoved](#) = (0xE01F0008L)
- static const uint32\_t [Status\\_PipeNotLinked](#) = (0xE01F0009L)
- static const uint32\_t [Status\\_ConnectedPipes](#) = (0xE01F000AL)
- static const uint32\_t [Status\\_DeviceLocked](#) = (0xE01F0010L)
- static const uint32\_t [Status\\_RequestMutexTimeout](#) = (0xE01F0020L)
- static const uint32\_t [Status\\_RequestMutexFailed](#) = (0xE01F0021L)
- static const uint32\_t [Status\\_LastUsbErrorMismatch](#) = (0xE01F0022L)
- static const uint32\_t [WPAError\\_ScanningIsPending](#) = ( (0xA0220000L) | 0x0036 )

### Properties

- virtual String<sup>^</sup> [SerialNumber](#) [get]

### 11.54.1 Detailed Description

Base class to handle MCS USB devices. All device classes are derived from this class. Functionality that is provided by all MCS devices is handled by this class.

### 11.54.2 Constructor & Destructor Documentation

#### 11.54.2.1 CMcsUsbNet() [1/2] `CMcsUsbNet ( )`

Initializes a new instance of the base class to handle MCS USB devices.

#### 11.54.2.2 CMcsUsbNet() [2/2] `CMcsUsbNet ( McsBusTypeEnumNet bustype )`

Initializes a new instance of the base class to handle MCS USB devices.

##### Parameters

<i>bustype</i>	Type of device to use, either USB or PCI.
----------------	---

#### 11.54.2.3 ~CMcsUsbNet() `virtual ~CMcsUsbNet ( ) [virtual]`

#### 11.54.2.4 "!CMcsUsbNet() `!CMcsUsbNet ( )`

### 11.54.3 Member Function Documentation

#### 11.54.3.1 AddSoftwareKey() `void AddSoftwareKey ( String^ key )`

#### 11.54.3.2 AssociateToThis() `void AssociateToThis ( CMcsUsbNet^ device )`

**11.54.3.3 Connect()** [1/4] `virtual uint32_t Connect ( ) [virtual]`

Opens a connection to the device.

**Returns**

Error Status. 0 on success.

**11.54.3.4 Connect()** [2/4] `virtual uint32_t Connect (   
CMcsUsbListEntryNet^ entry ) [virtual]`

Opens a connection to the device.

**Parameters**

<i>entry</i>	The Device List Entry for the device to be connected.
--------------	---

**Returns**

Error Status. 0 on success.

**11.54.3.5 Connect()** [3/4] `virtual uint32_t Connect (   
CMcsUsbListEntryNet^ entry,   
unsigned int LockMask ) [virtual]`

Opens a connection to the device.

**Parameters**

<i>entry</i>	The Device List Entry for the device to be connected.
<i>LockMask</i>	The Lock Mask for this connection.

**Returns**

Error Status. 0 on success.

**11.54.3.6 Connect()** [4/4] `virtual uint32_t Connect (   
unsigned int LockMask ) [virtual]`

Opens a connection to the device.

## Parameters

<i>LockMask</i>	The Lock Mask for this connection.
-----------------	------------------------------------

## Returns

Error Status. 0 on success.

### 11.54.3.7 Disconnect() `virtual void Disconnect ( ) [virtual]`

Disconnect from a device.

### 11.54.3.8 EmptyKey() `bool EmptyKey ( String^ key )`

### 11.54.3.9 EnableExceptions() `void EnableExceptions ( bool enable )`

Enables or Disables Exceptions for calls to McsUsb Devices. If Exceptions are disabled, the return value of a command can be queries with the `GetStatusOfLastCommand` call instead.

## Parameters

<i>enable</i>	True to enable Exceptions, False to disable.
---------------	--

### 11.54.3.10 EraseEepromRegisterPreconfig() [1/3] `void EraseEepromRegisterPreconfig ( uint32_t EEPROMBase, uint32_t DMA_reg )`

### 11.54.3.11 EraseEepromRegisterPreconfig() [2/3] `void EraseEepromRegisterPreconfig ( uint32_t EEPROMBase, uint32_t DMA_reg, uint32_t EEPROMSize )`

**11.54.3.12 EraseEepromRegisterPreconfig()** [3/3] `void EraseEepromRegisterPreconfig (`  
    `uint32_t EEPROMBase,`  
    `uint32_t DMA_reg,`  
    `uint32_t EEPROMSize,`  
    `uint32_t EepromStartAddress )`

**11.54.3.13 GetConfiguration()** `uint8_t GetConfiguration ( )`

**11.54.3.14 GetDeviceCannotStallOutRequests()** `uint32_t GetDeviceCannotStallOutRequests ( )`

**11.54.3.15 GetDeviceCapableSpeed()** `McsUsbSpeedEnumNet GetDeviceCapableSpeed ( )`

**11.54.3.16 GetDeviceEnum()** `DeviceEnumNet GetDeviceEnum ( )`

**11.54.3.17 GetDeviceId()** `DeviceIdNet ^ GetDeviceId ( )`

**11.54.3.18 GetDeviceRootHubVendorEnum()** `UsbVendorIdEnumNet GetDeviceRootHubVendorEnum ( )`

Gets the Vendor ID of the USB root hub the device is connected to.

#### Returns

An enum which enumerates the PCI Vendor ID.

**11.54.3.19 GetDeviceRootHubVendorID()** `uint32_t GetDeviceRootHubVendorID ( )`

Gets the Vendor ID of the USB root hub the device is connected to.

#### Returns

The PCI Vendor ID, 0x8086 for Intel, 0x1912 for Renesas, 0x1b21 for ASMedia.

**11.54.3.20 GetDeviceRootHubVendorName()** `String ^ GetDeviceRootHubVendorName ( )`

Gets the Vendor Name of the USB root hub the device is connected to.

**Returns**

The PCI Vendor Name, either "Intel", "Renesas", "ASMedia" or "unknown".

**11.54.3.21 GetDeviceSpeed()** `McsUsbSpeedEnumNet GetDeviceSpeed ( )`

Query the Connection Speed of the device.

**Returns**

0 for Low-Speed, 1 for Full-Speed, 2 for High-Speed and 3 for SuperSpeed.

**11.54.3.22 GetErrorText()** `static String ^ GetErrorText ( unsigned int Status ) [static]`

Gets the error text string that belongs to a status number.

**Parameters**

<i>Status</i>	The status number you want the text for.
---------------	--

**Returns**

The error text string that belongs to the status number.

**11.54.3.23 GetFirmwareVersion()** `unsigned int GetFirmwareVersion ( CFirmwareDestinationNet destination )`

Gets the firmware version for the destination.

**Parameters**

<i>destination</i>	The destination to be queried.
--------------------	--------------------------------

**Returns**

The firmware version as a 32 bit number, the upper 16 bit contain the major version number, the lower 16 bit the minor version number.

**11.54.3.24 GetHardwareRevision()** `String ^ GetHardwareRevision ( )`

**11.54.3.25 GetHeadstageActive()** `bool GetHeadstageActive (   
    uint32_t headstage )`

queries whether a headstage is active

**Parameters**

in	<i>headstage</i>	the headstage number (0 or 1)
----	------------------	-------------------------------

**Returns**

true if the headstage is active

**11.54.3.26 GetHeadstageID()** `HeadstageIdEnumNet GetHeadstageID (   
    uint32_t headstage )`

Gets the ID of a connected headstage.

**Parameters**

in	<i>headstage</i>	the headstage number (0 or 1)
----	------------------	-------------------------------

**Returns**

enumerated Headstage ID

**11.54.3.27 GetHeadstagePresent()** `bool GetHeadstagePresent (   
    uint32_t headstage )`

queries whether a headstage is present

**Parameters**

in	<i>headstage</i>	the headstage number (0 or 1)
----	------------------	-------------------------------

**Returns**

true if the headstage is present



**11.54.3.28 GetIdent()** `uint32_t GetIdent ( [System::Runtime::InteropServices::Out] String^% Answer )`

**11.54.3.29 GetLastUSBError()** `unsigned int GetLastUSBError ( )`

**11.54.3.30 GetMea21UsbPort()** `uint32_t GetMea21UsbPort ( )`

Gets the USB port if an IFB that is used by this connection

Returns

number of used port; range: 0..1

**11.54.3.31 GetNumConfigurations()** `uint8_t GetNumConfigurations ( )`

**11.54.3.32 GetSerialNumber()** `virtual String ^ GetSerialNumber ( ) [virtual]`

Query the Serial Number of the device.

Returns

The Serial Number.

**11.54.3.33 GetSoftwareKey()** `array<BYTE> ^ GetSoftwareKey ( unsigned int index )`

**11.54.3.34 GetSoftwareKeyString()** `[1/2] String ^ GetSoftwareKeyString ( SoftwareKeyProgramIdsNet::ProgramIdsNet ProgrammID, uint8_t majorversion )`

**11.54.3.35 GetSoftwareKeyString()** `[2/2] String ^ GetSoftwareKeyString ( uint8_t ProgrammID, uint8_t majorversion )`

**11.54.3.36 GetStatus()** virtual uint32\_t GetStatus (   
 [System::Runtime::InteropServices::Out] uint32\_t% iStatus ) [virtual]

**11.54.3.37 GetStatusOfLastCommand()** uint32\_t GetStatusOfLastCommand ( )

Gets the status of the last call to the McsUsb Library.

#### Returns

The Error Status of the last McsUsb command. 0 on success.

**11.54.3.38 GetUsbListEntry()** CMcsUsbListEntryNet ^ GetUsbListEntry ( )

**11.54.3.39 GetVersion()** [1/2] DriverVersionNet ^ GetVersion ( )

**11.54.3.40 GetVersion()** [2/2] DriverVersionNet ^ GetVersion (   
 CFirmwareDestinationNet dest )

**11.54.3.41 HasSoftwareKey()** [1/2] bool HasSoftwareKey (   
 SoftwareKeyProgrammIdsNet::ProgrammIdsNet ProgrammID,   
 uint8\_t majorversion )

**11.54.3.42 HasSoftwareKey()** [2/2] bool HasSoftwareKey (   
 uint8\_t ProgrammID,   
 uint8\_t majorversion )

**11.54.3.43 IsConnected()** virtual bool IsConnected ( ) [virtual]

Check if a device is Connected.

#### Returns

true if the device is connected.

**11.54.3.44 IsDeviceHighSpeed()** `bool IsDeviceHighSpeed ( )`

**11.54.3.45 IsDeviceHighSpeedCapable()** `bool IsDeviceHighSpeedCapable ( )`

**11.54.3.46 IsExceptionsEnabled()** `bool IsExceptionsEnabled ( )`

**11.54.3.47 MultibootGetCypressImageId()** `uint32_t MultibootGetCypressImageId ( unsigned int sector )`

Query the multiboot image id of the device located in specified sector (range: 0..9 0..9) of IFB2 Cypress.

**Returns**

The magic ident code of the image.

**11.54.3.48 MultibootGetImageId()** `String ^ MultibootGetImageId ( unsigned int sector )`

Query the multiboot image id of the device located in specified sector (range: 0..2 / 0..9) of IFB1 / IFB2 FPGA.

**Returns**

The magic ident code of the image.

**11.54.3.49 MultibootGetSelectedImage()** `uint32_t MultibootGetSelectedImage ( )`

Gets sector index of selected FPGA boot image on IFB

**Returns**

Sector index of image; range: 0..2

**11.54.3.50 MultibootSelectImage()** `void MultibootSelectImage ( unsigned int sector )`

Select the multiboot image specified by "sector" (range: 0..2) for IFB FPGA.

#### Returns

Throws exception on error.

**11.54.3.51 ReadEepromRegisterPreconfig() [1/3]** `bool ReadEepromRegisterPreconfig ( uint32_t EEPROMBase,  
uint32_t DMA_reg,  
[System::Runtime::InteropServices::Out] uint32_t% DMA_value )`

**11.54.3.52 ReadEepromRegisterPreconfig() [2/3]** `bool ReadEepromRegisterPreconfig ( uint32_t EEPROMBase,  
uint32_t DMA_reg,  
[System::Runtime::InteropServices::Out] uint32_t% DMA_value,  
uint32_t EEPROMSize )`

**11.54.3.53 ReadEepromRegisterPreconfig() [3/3]** `bool ReadEepromRegisterPreconfig ( uint32_t EEPROMBase,  
uint32_t DMA_reg,  
[System::Runtime::InteropServices::Out] uint32_t% DMA_value,  
uint32_t EEPROMSize,  
uint32_t EepromStartAddress )`

**11.54.3.54 ReadRegister() [1/2]** `unsigned int ReadRegister ( unsigned int reg )`

**11.54.3.55 ReadRegister() [2/2]** `array<uint32_t> ^ ReadRegister ( unsigned int reg,  
int length )`

**11.54.3.56 ReadRegister32()** `unsigned int ReadRegister32 ( unsigned int adr )`

**11.54.3.57 ReadRegisterTimeSlot()** unsigned int ReadRegisterTimeSlot (   
 unsigned int *reg*,   
 int *TimeSlot* )

**11.54.3.58 RemoveSoftwareKey()** void RemoveSoftwareKey (   
 unsigned int *index* )

**11.54.3.59 RescanHeadstage()** void RescanHeadstage (   
 uint32\_t *headstage* )

rescans and activates a headstage

#### Parameters

in	<i>headstage</i>	the headstage number (0 or 1)
----	------------------	-------------------------------

**11.54.3.60 SetConfiguration()** void SetConfiguration (   
 uint8\_t *config* )

**11.54.3.61 SetSoftwareKey()** void SetSoftwareKey (   
 unsigned int *index*,   
 array< BYTE >^ *buffer* )

**11.54.3.62 ThrowCUsbExceptionNetOnError()** void ThrowCUsbExceptionNetOnError (   
 uint32\_t *status* )

**11.54.3.63 TxnGetSerialNumber()** unsigned int TxnGetSerialNumber ( )

**11.54.3.64 TxnSetSerialNumber()** void TxnSetSerialNumber (   
 unsigned int *number* )

**11.54.3.65 TxnTestMemoryReadAndCheck()** unsigned int TxnTestMemoryReadAndCheck (   
 unsigned short *index* )

**11.54.3.66 TxnTestMemoryWrite()** unsigned int TxnTestMemoryWrite (   
 unsigned short *index* )

**11.54.3.67 ValidKey() [1/2]** bool ValidKey (   
 String^ *key*,   
 [System::Runtime::InteropServices::Out] String^% *serial\_number* )

**11.54.3.68 ValidKey() [2/2]** bool ValidKey (   
 String^ *key*,   
 uint8\_t *ProgrammID*,   
 uint8\_t *majorversion*,   
 [System::Runtime::InteropServices::Out] String^% *serial\_number* )

**11.54.3.69 WriteEepromRegisterPreconfig() [1/3]** void WriteEepromRegisterPreconfig (   
 uint32\_t *EEPROMBase*,   
 uint32\_t *DMA\_reg*,   
 uint32\_t *DMA\_value* )

**11.54.3.70 WriteEepromRegisterPreconfig() [2/3]** void WriteEepromRegisterPreconfig (   
 uint32\_t *EEPROMBase*,   
 uint32\_t *DMA\_reg*,   
 uint32\_t *DMA\_value*,   
 uint32\_t *EEPROMSize* )

**11.54.3.71 WriteEepromRegisterPreconfig() [3/3]** void WriteEepromRegisterPreconfig (   
 uint32\_t *EEPROMBase*,   
 uint32\_t *DMA\_reg*,   
 uint32\_t *DMA\_value*,   
 uint32\_t *EEPROMSize*,   
 uint32\_t *EepromStartAddress* )

**11.54.3.72 WriteRegister()** [1/2] void WriteRegister (   
 unsigned int *reg*,   
 array< unsigned int >^ *values* )

**11.54.3.73 WriteRegister()** [2/2] void WriteRegister (   
 unsigned int *reg*,   
 unsigned int *value* )

**11.54.3.74 WriteRegister32()** void WriteRegister32 (   
 unsigned int *adr*,   
 unsigned int *value* )

**11.54.3.75 WriteRegisterArray()** void WriteRegisterArray (   
 unsigned int *reg*,   
 array< unsigned int >^ *values* )

**11.54.3.76 WriteRegisterTimeSlot()** [1/2] void WriteRegisterTimeSlot (   
 unsigned int *reg*,   
 array< unsigned int >^ *values*,   
 int *TimeSlot* )

**11.54.3.77 WriteRegisterTimeSlot()** [2/2] void WriteRegisterTimeSlot (   
 unsigned int *reg*,   
 unsigned int *value*,   
 int *TimeSlot* )

**11.54.3.78 WriteRegisterValue()** void WriteRegisterValue (   
 unsigned int *reg*,   
 unsigned int *value* )

## 11.54.4 Member Data Documentation

**11.54.4.1 Status\_AlreadyConfigured** `const uint32_t Status_AlreadyConfigured = (0xE0110001L)`  
[static]

**11.54.4.2 Status\_BadStartFrame** `const uint32_t Status_BadStartFrame = (0xE0100A00L)` [static]

**11.54.4.3 Status\_Btstuff** `const uint32_t Status_Btstuff = (0xE0100002L)` [static]

**11.54.4.4 Status\_BufferOverrun** `const uint32_t Status_BufferOverrun = (0xE010000CL)` [static]

**11.54.4.5 Status\_BufferUnderrun** `const uint32_t Status_BufferUnderrun = (0xE010000DL)` [static]

**11.54.4.6 Status\_Canceled** `const uint32_t Status_Canceled = (0xE0110000L)` [static]

**11.54.4.7 Status\_Canceling** `const uint32_t Status_Canceling = (0xE0120000L)` [static]

**11.54.4.8 Status\_ConnectedPipes** `const uint32_t Status_ConnectedPipes = (0xE01F000AL)` [static]

**11.54.4.9 Status\_ControlNotOwned** `const uint32_t Status_ControlNotOwned = (0xE0100D00L)` [static]

**11.54.4.10 Status\_Crc** `const uint32_t Status_Crc = (0xE0100001L)` [static]

**11.54.4.11 Status\_DataOverrun** `const uint32_t Status_DataOverrun = (0xE0100008L)` [static]



**11.54.4.12 Status\_DataToggleMismatch** `const uint32_t Status_DataToggleMismatch = (0xE0100003L)`  
[static]

**11.54.4.13 Status\_DataUnderrun** `const uint32_t Status_DataUnderrun = (0xE0100009L)` [static]

**11.54.4.14 Status\_DeviceLocked** `const uint32_t Status_DeviceLocked = (0xE01F0010L)` [static]

**11.54.4.15 Status\_DeviceNotFound** `const uint32_t Status_DeviceNotFound = (0xE01F0003L)` [static]

**11.54.4.16 Status\_DeviceRemoved** `const uint32_t Status_DeviceRemoved = (0xE01F0008L)` [static]

**11.54.4.17 Status\_DevNotResponding** `const uint32_t Status_DevNotResponding = (0xE0100005L)`  
[static]

**11.54.4.18 Status\_EndpointHalted** `const uint32_t Status_EndpointHalted = (0xE0100030L)` [static]

**11.54.4.19 Status\_ErrorBusy** `const uint32_t Status_ErrorBusy = (0xE0100400L)` [static]

**11.54.4.20 Status\_ErrorShortTransfer** `const uint32_t Status_ErrorShortTransfer = (0xE0100900L)`  
[static]

**11.54.4.21 Status\_Fifo** `const uint32_t Status_Fifo = (0xE0100010L)` [static]

**11.54.4.22 Status\_FrameControlOwned** `const uint32_t Status_FrameControlOwned = (0xE0100C00L)`  
[static]

**11.54.4.23 Status\_InternalHcError** `const uint32_t Status_InternalHcError = (0xE0100800L) [static]`

**11.54.4.24 Status\_InvalidParameter** `const uint32_t Status_InvalidParameter = (0xE0100300L) [static]`

**11.54.4.25 Status\_InvalidPipeHandle** `const uint32_t Status_InvalidPipeHandle = (0xE0100600L) [static]`

**11.54.4.26 Status\_InvalidUrbFunction** `const uint32_t Status_InvalidUrbFunction = (0xE0100200L) [static]`

**11.54.4.27 Status\_IoPending** `const uint32_t Status_IoPending = (0xE01F0006L) [static]`

**11.54.4.28 Status\_IoTimeout** `const uint32_t Status_IoTimeout = (0xE01F0007L) [static]`

**11.54.4.29 Status\_IsochRequestFailed** `const uint32_t Status_IsochRequestFailed = (0xE0100B00L) [static]`

**11.54.4.30 Status\_LastUsbErrorMismatch** `const uint32_t Status_LastUsbErrorMismatch = (0xE01F0022L) [static]`

**11.54.4.31 Status\_NoBandwidth** `const uint32_t Status_NoBandwidth = (0xE0100700L) [static]`

**11.54.4.32 Status\_NoMemory** `const uint32_t Status_NoMemory = (0xE0100100L) [static]`

**11.54.4.33 Status\_NoSuchDevice** `const uint32_t Status_NoSuchDevice = (0xE01F0002L) [static]`

**11.54.4.34 Status\_NotAccessed** `const uint32_t Status_NotAccessed = (0xE010000FL) [static]`

**11.54.4.35 Status\_NotSupported** `const uint32_t Status_NotSupported = (0xE01F0005L) [static]`

**11.54.4.36 Status\_PidCheckFailure** `const uint32_t Status_PidCheckFailure = (0xE0100006L) [static]`

**11.54.4.37 Status\_PipeNotLinked** `const uint32_t Status_PipeNotLinked = (0xE01F0009L) [static]`

**11.54.4.38 Status\_RequestFailed** `const uint32_t Status_RequestFailed = (0xE0100500L) [static]`

**11.54.4.39 Status\_RequestMutexFailed** `const uint32_t Status_RequestMutexFailed = (0xE01F0021L) [static]`

**11.54.4.40 Status\_RequestMutexTimeout** `const uint32_t Status_RequestMutexTimeout = (0xE01F0020L) [static]`

**11.54.4.41 Status\_Stall** `const uint32_t Status_Stall = (0xE0100004L) [static]`

**11.54.4.42 Status\_Unconfigured** `const uint32_t Status_Unconfigured = (0xE0110002L) [static]`

**11.54.4.43 Status\_UnexpectedPid** `const uint32_t Status_UnexpectedPid = (0xE0100007L) [static]`

**11.54.4.44 WPAError\_ScanningIsPending** `const uint32_t WPAError_ScanningIsPending = ( (0x←A0220000L) | 0x0036 ) [static]`

### 11.54.5 Property Documentation

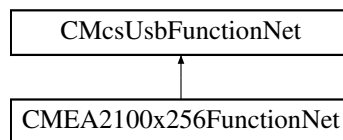
**11.54.5.1 SerialNumber** `virtual String^ SerialNumber [get]`

## 11.55 CMcsUsbPointerContainer Class Reference

## 11.56 CMEA2100x256FunctionNet Class Reference

[CMEA2100x256FunctionNet](#) is the class to control the MEA2100-256 device needs `#include "Stg200xNet.h"` to resolve documentation reference

Inheritance diagram for CMEA2100x256FunctionNet:



### Public Member Functions

- [CMEA2100x256FunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> pMEA2100x256FunctionPointerContainer)  
*Initializes a new instance of the [CMEA2100x256FunctionNet](#) class.*
- [CMEA2100x256FunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- virtual [~CMEA2100x256FunctionNet](#) ()
- [ICMEA2100x256FunctionNet](#) ()
- [StimulationLayoutConfigurationEnumNet](#) [GetLayoutConfiguration](#) ()  
*Gets the stimulation layout configuration. Can be single well, 6-well or 9-well. The number of D↔ AC channels available per well is [Mcs::Usb::CStg200xBasicNet::GetNumberOfAnalogChannels](#) divided by [Mcs::Usb::CStg200xBasicNet::GetNumberOfStimulationSourcesPerElectrode](#).*
- void [SetLayoutConfiguration](#) ([StimulationLayoutConfigurationEnumNet](#) LayoutConfiguration)  
*Sets the stimulation layout configuration. Can be single well, 6-well or 9-well. The number of D↔ AC channels available per well is [Mcs::Usb::CStg200xBasicNet::GetNumberOfAnalogChannels](#) divided by [Mcs::Usb::CStg200xBasicNet::GetNumberOfStimulationSourcesPerElectrode](#).*

### Additional Inherited Members

#### 11.56.1 Detailed Description

[CMEA2100x256FunctionNet](#) is the class to control the MEA2100-256 device needs `#include "Stg200xNet.h"` to resolve documentation reference

#### 11.56.2 Constructor & Destructor Documentation

**11.56.2.1 CMEA2100x256FunctionNet()** [1/2] `CMEA2100x256FunctionNet ( CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ pMEA2100x256FunctionPointerContainer )`

Initializes a new instance of the [CMEA2100x256FunctionNet](#) class.

**11.56.2.2 CMEA2100x256FunctionNet()** [2/2] `CMEA2100x256FunctionNet ( CMcsUsbNet^ mcsusb )`

**11.56.2.3 ~CMEA2100x256FunctionNet()** `virtual ~CMEA2100x256FunctionNet ( ) [virtual]`

**11.56.2.4 !"CMEA2100x256FunctionNet()** `!CMEA2100x256FunctionNet ( )`

### 11.56.3 Member Function Documentation

**11.56.3.1 GetLayoutConfiguration()** `StimulationLayoutConfigurationEnumNet GetLayoutConfiguration ( )`

Gets the stimulation layout configuration. Can be single well, 6-well or 9-well. The number of D↔ AC channels available per well is [Mcs::Usb::CStg200xBasicNet::GetNumberOfAnalogChannels](#) divided by [Mcs::Usb::CStg200xBasicNet::GetNumberOfStimulationSourcesPerElectrode](#).

#### Returns

The currently active stimulation layout configuration.

**11.56.3.2 SetLayoutConfiguration()** `void SetLayoutConfiguration ( StimulationLayoutConfigurationEnumNet LayoutConfiguration )`

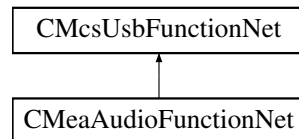
Sets the stimulation layout configuration. Can be single well, 6-well or 9-well. The number of D↔ AC channels available per well is [Mcs::Usb::CStg200xBasicNet::GetNumberOfAnalogChannels](#) divided by [Mcs::Usb::CStg200xBasicNet::GetNumberOfStimulationSourcesPerElectrode](#).

#### Parameters

<i>LayoutConfiguration</i>	The new stimulation layout configuration.
----------------------------	---

## 11.57 CMeaAudioFunctionNet Class Reference

Inheritance diagram for CMeaAudioFunctionNet:



### Classes

- struct [s\\_setaudionet](#)

### Public Member Functions

- [CMeaAudioFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> meaAudioFunctionPointerContainer)
- [CMeaAudioFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- virtual uint32\_t [GetNumberOfAudioChannels](#) ()  
*Gets the number of available audio channels.*
- virtual uint32\_t [SetAudioChannels](#) (array< [s\\_setaudionet](#)<sup>^</sup>> channels)  
*Sets the electrode to monitor and amplification for the audio channels.*
- virtual uint32\_t [SetAudioChannels](#) (array< [s\\_setaudionet](#)<sup>^</sup>> channels, unsigned int virtualDevice)  
*Sets the electrode to monitor and amplification for the audio channels.*
- virtual uint32\_t [GetAudioChannels](#) ([System::Runtime::InteropServices::Out]array< [s\\_setaudionet](#)<sup>^</sup>> channels)  
*Gets the electrode to monitor and amplification for the audio channels.*
- virtual uint32\_t [GetAudioChannels](#) ([System::Runtime::InteropServices::Out]array< [s\\_setaudionet](#)<sup>^</sup>> channels, unsigned int virtualDevice)  
*Gets the electrode to monitor and amplification for the audio channels.*

### Additional Inherited Members

#### 11.57.1 Constructor & Destructor Documentation

**11.57.1.1 CMeaAudioFunctionNet() [1/2]** [CMeaAudioFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb,  
[CMcsUsbFunctionPointerContainer](#)<sup>^</sup> meaAudioFunctionPointerContainer )

**11.57.1.2 CMeaAudioFunctionNet() [2/2]** [CMeaAudioFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb )

### 11.57.2 Member Function Documentation

**11.57.2.1 GetAudioChannels()** [1/2] virtual uint32\_t GetAudioChannels (   
 [System::Runtime::InteropServices::Out] array< [s\\_setaudionet](#)^>^% *channels* )   
 [virtual]

Gets the electrode to monitor and amplification for the audio channels.

## Parameters

<i>channels</i>	Struct which contains the electrode (channel) and amplification on return.
-----------------	--

## Returns

Error Status. 0 on success.

**11.57.2.2 GetAudioChannels()** [2/2] `virtual uint32_t GetAudioChannels ( [System::Runtime::InteropServices::Out] array< s\_setaudionet>^% channels, unsigned int virtualDevice ) [virtual]`

Gets the electrode to monitor and amplification for the audio channels.

## Parameters

<i>channels</i>	Struct which contains the electrode (channel) and amplification on return.
-----------------	--

## Parameters

<i>virtualDevice</i>	Virtual device to use.
----------------------	------------------------

## Returns

Error Status. 0 on success.

**11.57.2.3 GetNumberOfAudioChannels()** `virtual uint32_t GetNumberOfAudioChannels ( ) [virtual]`

Gets the number of available audio channels.

## Returns

The number of audio channels available, 0 when there are none.

**11.57.2.4 SetAudioChannels()** [1/2] `virtual uint32_t SetAudioChannels ( array< s\_setaudionet>^ channels ) [virtual]`

Sets the electrode to monitor and amplification for the audio channels.



## Parameters

<i>channels</i>	Struct which defines the electrode (channel) and amplification.
-----------------	---

## Returns

Error Status. 0 on success.

**11.57.2.5 SetAudioChannels()** [2/2] virtual uint32\_t SetAudioChannels (   
     array< [s\\_setaudionet](#)>^ channels,   
     unsigned int *virtualDevice* ) [virtual]

Sets the electrode to monitor and amplification for the audio channels.

## Parameters

<i>channels</i>	Struct which defines the electrode (channel) and amplification.
-----------------	---

## Parameters

<i>virtualDevice</i>	Virtual device to use.
----------------------	------------------------

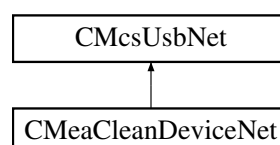
## Returns

Error Status. 0 on success.

## 11.58 CMeaCleanDeviceNet Class Reference

[CMeaCleanDeviceNet](#) is the class to access the MEA Clean device.

Inheritance diagram for CMeaCleanDeviceNet:



**Public Member Functions**

- [CMeaCleanDeviceNet](#) ()  
*Initializes a new instance of the [CMeaCleanDeviceNet](#) class.*
- virtual [~CMeaCleanDeviceNet](#) ()
- [!CMeaCleanDeviceNet](#) ()
- void [Start](#) ()  
*Starts a MEA Clean run.*
- void [Stop](#) ()  
*Stops a MEA Clean run.*
- void [SetSlope](#) (uint32\_t voltageSlope)  
*Sets the voltage slope.*
- void [SetCycles](#) (uint32\_t cycles)  
*Sets the number of cycles.*
- void [SetMinVoltage](#) (int32\_t voltageMin)  
*Sets the lower voltage level.*
- void [SetMaxVoltage](#) (int32\_t voltageMax)  
*Sets the upper voltage level.*
- bool [IsRunning](#) ()  
*Gets if the MEA Clean device is running.*
- uint32\_t [GetSlope](#) ()  
*Gets the voltage slope.*
- uint32\_t [GetCycles](#) ()  
*Gets the number of cycles.*
- int32\_t [GetMinVoltage](#) ()  
*Gets the lower voltage level.*
- int32\_t [GetMaxVoltage](#) ()  
*Gets the upper voltage level.*
- int32\_t [GetOutputVoltage](#) ()  
*Gets the output voltage.*
- int32\_t [GetCycle](#) ()  
*Gets the current cycle.*

**Additional Inherited Members****11.58.1 Detailed Description**

[CMeaCleanDeviceNet](#) is the class to access the MEA Clean device.

**11.58.2 Constructor & Destructor Documentation****11.58.2.1 CMeaCleanDeviceNet() [CMeaCleanDeviceNet](#) ( )**

Initializes a new instance of the [CMeaCleanDeviceNet](#) class.

**11.58.2.2** `~CMeaCleanDeviceNet()` `virtual ~CMeaCleanDeviceNet ( ) [virtual]`

**11.58.2.3** `!CMeaCleanDeviceNet()` `!CMeaCleanDeviceNet ( )`

### 11.58.3 Member Function Documentation

**11.58.3.1** `GetCycle()` `int32_t GetCycle ( )`

Gets the current cycle.

#### Returns

The cycle number.

**11.58.3.2** `GetCycles()` `uint32_t GetCycles ( )`

Gets the number of cycles.

#### Returns

The number of cycles to run for.

**11.58.3.3** `GetMaxVoltage()` `int32_t GetMaxVoltage ( )`

Gets the upper voltage level

#### Returns

The upper voltage level in mV.

**11.58.3.4** `GetMinVoltage()` `int32_t GetMinVoltage ( )`

Gets the lower voltage level.

#### Returns

The lower voltage level in mV.

**11.58.3.5 GetOutputVoltage()** `int32_t GetOutputVoltage ( )`

Gets the output voltage.

**Returns**

The output voltage in mV.

**11.58.3.6 GetSlope()** `uint32_t GetSlope ( )`

Gets the voltage slope.

**Returns**

The voltage slope in mV/s.

**11.58.3.7 IsRunning()** `bool IsRunning ( )`

Gets if the MEA Clean device is running.

**Returns**

"true" when a run is in progress, otherwise "false".

**11.58.3.8 SetCycles()** `void SetCycles (   
uint32_t cycles )`

Sets the number of cycles.

**Parameters**

<i>cycles</i>	The number of cycles to run for (0 .. 99).
---------------	--

**11.58.3.9 SetMaxVoltage()** `void SetMaxVoltage (   
int32_t voltageMax )`

Sets the upper voltage level.

**Parameters**

<i>voltageMax</i>	The upper voltage level in mV (-1.6 .. 1.6 V).
-------------------	--

**11.58.3.10 SetMinVoltage()** `void SetMinVoltage (`  
`int32_t voltageMin )`

Sets the lower voltage level.

Parameters

<i>voltageMin</i>	The lower voltage level in mV (-1.6 .. 1.6 V).
-------------------	--

**11.58.3.11 SetSlope()** `void SetSlope (`  
`uint32_t voltageSlope )`

Sets the voltage slope.

Parameters

<i>voltageSlope</i>	The voltage slope in mV/s (range 0 .. 60 V/s).
---------------------	--

**11.58.3.12 Start()** `void Start ( )`

Starts a MEA Clean run.

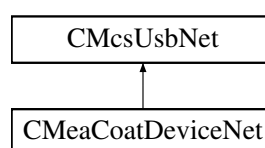
**11.58.3.13 Stop()** `void Stop ( )`

Stops a MEA Clean run.

## 11.59 CMeaCoatDeviceNet Class Reference

[CMeaCoatDeviceNet](#) is the class to access the MEA Coat device.

Inheritance diagram for CMeaCoatDeviceNet:



## Public Member Functions

- [CMeaCoatDeviceNet](#) ()  
*Initializes a new instance of the [CMeaCoatDeviceNet](#) class.*
- virtual [~CMeaCoatDeviceNet](#) ()
- [!CMeaCoatDeviceNet](#) ()
- void [Start](#) ()  
*Starts a MEA Coat run.*
- void [Stop](#) ()  
*Stops a MEA Coat run.*
- void [SetSlope](#) (int32\_t currentSlope)  
*Sets the current slope.*
- void [SetDuration](#) (uint32\_t duration)  
*Sets the duration of a MEA Coat run.*
- void [SetMaxCurrent](#) (uint32\_t currentMax)  
*Sets the limit of the current ramp (absolute value).*
- void [SetOffsetCurrent](#) (int32\_t currentOffset)  
*Sets the offset of the current.*
- bool [IsRunning](#) ()  
*Gets if the MEA Clean device is running.*
- int32\_t [GetSlope](#) ()  
*Gets the current slope.*
- uint32\_t [GetDuration](#) ()  
*Gets the duration of a MEA Coat run.*
- uint32\_t [GetMaxCurrent](#) ()  
*Gets the limit of the current ramp (absolute value).*
- int32\_t [GetOffsetCurrent](#) ()  
*Gets the offset of the current.*
- int32\_t [GetOutputCurrent](#) ()  
*Gets the output current.*
- int32\_t [GetTimeInPlateau](#) ()  
*Gets the time in the plateau.*
- void [SetPauseDuration](#) (uint32\_t pauseDuration)  
*Sets the duration of the pause between MEA Coat pulses.*
- uint32\_t [GetPauseDuration](#) ()  
*Gets the duration of the pause between MEA Coat pulses.*
- int32\_t [GetTimeInPause](#) ()  
*Gets the time in the pause.*
- void [SetCycles](#) (uint32\_t cycles)  
*Sets the number of cycles.*
- uint32\_t [GetCycles](#) ()  
*Gets the number of cycles.*
- int32\_t [GetCurrentCycle](#) ()  
*Gets the current cycle.*

## Additional Inherited Members

### 11.59.1 Detailed Description

[CMeaCoatDeviceNet](#) is the class to access the MEA Coat device.

## 11.59.2 Constructor & Destructor Documentation

### 11.59.2.1 CMeaCoatDeviceNet() [CMeaCoatDeviceNet](#) ( )

Initializes a new instance of the [CMeaCoatDeviceNet](#) class.

### 11.59.2.2 ~CMeaCoatDeviceNet() [virtual](#) [~CMeaCoatDeviceNet](#) ( ) [\[virtual\]](#)

### 11.59.2.3 "!CMeaCoatDeviceNet() [!CMeaCoatDeviceNet](#) ( )

## 11.59.3 Member Function Documentation

### 11.59.3.1 GetCurrentCycle() [int32\\_t](#) [GetCurrentCycle](#) ( )

Gets the current cycle.

#### Returns

The cycle number.

### 11.59.3.2 GetCycles() [uint32\\_t](#) [GetCycles](#) ( )

Gets the number of cycles.

#### Returns

The number of cycles to run for.

### 11.59.3.3 GetDuration() [uint32\\_t](#) [GetDuration](#) ( )

Gets the duration of a MEA Coat run.

#### Returns

The duration in ms.

**11.59.3.4 GetMaxCurrent()** `uint32_t GetMaxCurrent ( )`

Gets the limit of the current ramp (absolute value).

**Returns**

The limit of the current ramp in pA (absolute value).

**11.59.3.5 GetOffsetCurrent()** `int32_t GetOffsetCurrent ( )`

Gets the offset of the current.

**Returns**

The offset of the current in pA.

**11.59.3.6 GetOutputCurrent()** `int32_t GetOutputCurrent ( )`

Gets the output current.

**Returns**

The output current in pA.

**11.59.3.7 GetPauseDuration()** `uint32_t GetPauseDuration ( )`

Gets the duration of the pause between MEA Coat pulses.

**Returns**

The duration in ms.

**11.59.3.8 GetSlope()** `int32_t GetSlope ( )`

Gets the current slope.

**Returns**

The current slope in pA/s.



**11.59.3.9 GetTimeInPause()** `int32_t GetTimeInPause ( )`

Gets the time in the pause.

**Returns**

The time in the pause in ms.

**11.59.3.10 GetTimeInPlateau()** `int32_t GetTimeInPlateau ( )`

Gets the time in the plateau.

**Returns**

The time in the plateau in ms.

**11.59.3.11 IsRunning()** `bool IsRunning ( )`

Gets if the MEA Clean device is running.

**Returns**

"true" when a run is in progress, otherwise "false".

**11.59.3.12 SetCycles()** `void SetCycles (   
uint32_t cycles )`

Sets the number of cycles.

**Parameters**

<i>cycles</i>	The number of cycles to run for (0 .. 99).
---------------	--

**11.59.3.13 SetDuration()** `void SetDuration (   
uint32_t duration )`

Sets the duration of a MEA Coat run.

**Parameters**

<i>duration</i>	The duration in ms (range 0 .. 65 s).
-----------------	---------------------------------------

**11.59.3.14 SetMaxCurrent()** `void SetMaxCurrent (`  
    `uint32_t currentMax )`

Sets the limit of the current ramp (absolute value).

Parameters

<i>currentMax</i>	The limit of the current ramp in pA (absolute value, 0 .. 18 nA).
-------------------	---

**11.59.3.15 SetOffsetCurrent()** `void SetOffsetCurrent (`  
    `int32_t currentOffset )`

Sets the offset of the current.

Parameters

<i>currentOffset</i>	The offset of the current in pA (-10 .. 10 nA).
----------------------	---

**11.59.3.16 SetPauseDuration()** `void SetPauseDuration (`  
    `uint32_t pauseDuration )`

Sets the duration of the pause between MEA Coat pulses.

Parameters

<i>pauseDuration</i>	The duration in ms (range 0 .. 65 s).
----------------------	---------------------------------------

**11.59.3.17 SetSlope()** `void SetSlope (`  
    `int32_t currentSlope )`

Sets the current slope.

Parameters

<i>currentSlope</i>	The current slope in pA/s (range -65 .. 65 nA/s).
---------------------	---

**11.59.3.18 Start()** `void Start ( )`

Starts a MEA Coat run.

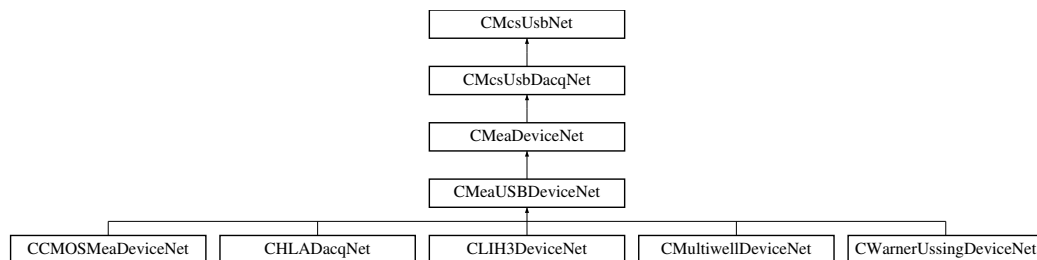
### 11.59.3.19 Stop() void Stop ( )

Stops a MEA Coat run.

## 11.60 CMeaDeviceNet Class Reference

Base class for MEA data acquisition devices.

Inheritance diagram for CMeaDeviceNet:



### Public Member Functions

- [CMeaDeviceNet](#) (McsBusTypeEnumNet bustype)  
*Initializes a new instance of [CMeaDeviceNet](#) class.*
- [CMeaDeviceNet](#) (McsBusTypeEnumNet bustype, [OnChannelData](#)<sup>^</sup> channelData, [OnError](#)<sup>^</sup> error)  
*Initializes a new instance of [CMeaDeviceNet](#) class.*
- [~CMeaDeviceNet](#) ( )
- virtual int32\_t [GetGain](#) ( )  
*Gets the amplifier gain of the device.*
- int32\_t [GetEnumerationSpeed](#) ( )
- virtual int32\_t [GetAnalogGain](#) ( )  
*Gets the gain of the analog inputs of the device.*
- virtual uint32\_t [EnableDigitalIn](#) (bool enable, unsigned int virtualDevice)  
*Enable the digital data word in the datastream.*
- virtual uint32\_t [EnableDigitalIn](#) (DigitalDatastreamEnableEnumNet enable, unsigned int virtualDevice)  
*Enable digital data words in the datastream.*
- virtual uint32\_t [EnableTimestamp](#) (bool enable, unsigned int virtualDevice)  
*Enable the timestamp data word in the datastream. The timestamp is a 64 bit counter.*
- virtual uint32\_t [EnableChecksum](#) (bool enable, unsigned int virtualDevice)  
*Enable the checksum data word in the datastream. The checksum is a 32 bit counter and 2x16 bit magic numbers.*
- virtual void [SetDigitalOut](#) (unsigned int digout\_value, int pulselength)  
*Generate a pulse on the digital output.*
- virtual uint32\_t [SetNumberOfChannels](#) (int NumberOfChannels)  
*Sets the number of analog channels in the datastream.*
- virtual uint32\_t [SetNumberOfChannels](#) (int NumberOfChannels, unsigned int virtualDevice)  
*Sets the number of analog channels in the datastream.*

- virtual uint32\_t [SetNumberOfAnalogChannels](#) (unsigned int NumberOfChannels\_HS1, unsigned int NumberOfChannels\_HS2, unsigned int NumberOfChannels\_DSP, unsigned int NumberOfChannels\_IF, unsigned int virtualDevice)  
*Sets the number of analog channels in the datastream for the MEA2100 device.*
- virtual uint32\_t [SetTriggerPeriod](#) (int samples, unsigned int virtualDevice)  
*Sets the maximum number of samples per trigger.*
- virtual uint32\_t [SetTriggerMaskValue](#) (unsigned int mask, unsigned int value, unsigned int virtualDevice)  
*Defines a pattern on the digital dataword which will start a trigger when found.*

## Properties

- [CMeFunctionNet](#)^ [MeFunctionNet](#) [get]
- [CWClassicFunctionNet](#)^ [WClassicFunctionNet](#) [get]
- [CW2100\\_FunctionNet](#)^ [W2100\\_FunctionNet](#) [get]
- [CMeaAudioFunctionNet](#)^ [MeaAudioFunctionNet](#) [get]
- [CMeaDigitalDataFunctionNet](#)^ [MeaDigitalDataFunctionNet](#) [get]
- [CMeaFeedbackFunctionNet](#)^ [MeaFeedbackFunctionNet](#) [get]
- virtual int [Gain](#) [get]  
*The amplifier gain of the device. Value is gain times 1000, a value of 1000 corresponds to a gain of 1.0.*
- virtual int [AnalogGain](#) [get]  
*The gain of the analog inputs of the device. Value is gain times 1000, a value of 1000 corresponds to a gain of 1.0.*

## Additional Inherited Members

### 11.60.1 Detailed Description

Base class for MEA data acquisition devices.

There are two different device types for MEA data acquisition devices. There are the USB-MEA devices and the MC↔\_Card. In .NET both classes can be accessed by the constructor of the base class [CMeaDeviceNet](#), which constructs the correct underlying C++ class for the USB-MEA device on the one hand or the MC\_Card device on the other hand. Through this interface both device types USB-MEA devices and MC\_Card devices can be accessed

### 11.60.2 Constructor & Destructor Documentation

#### 11.60.2.1 [CMeaDeviceNet\(\)](#) [1/2] [CMeaDeviceNet](#) ( McsBusTypeEnumNet *bustype* )

Initializes a new instance of [CMeaDeviceNet](#) class.

#### Parameters

<i>bustype</i>	Type of device to use, either USB or PCI.
----------------	---

**11.60.2.2 CMeaDeviceNet()** [2/2] [CMeaDeviceNet](#) (   
 McsBusTypeEnumNet *bustype*,  
 [OnChannelData](#)<sup>^</sup> *channelData*,  
 [OnError](#)<sup>^</sup> *error* )

Initializes a new instance of [CMeaDeviceNet](#) class.

#### Parameters

<i>bustype</i>	Type of device to use, either USB or PCI.
----------------	---

#### Parameters

<i>channelData</i>	Callback to call when new data is available.
--------------------	--

#### Parameters

<i>error</i>	Callback to call when an error occurred.
--------------	--

**11.60.2.3 ~CMeaDeviceNet()** [~CMeaDeviceNet](#) ( )

### 11.60.3 Member Function Documentation

**11.60.3.1 EnableChecksum()** virtual uint32\_t EnableChecksum (   
 bool *enable*,  
 unsigned int *virtualDevice* ) [virtual]

Enable the checksum data word in the datastream. The checksum is a 32 bit counter and 2x16 bit magic numbers.

#### Parameters

<i>enable</i>	True to enable, False to disable.
<i>virtualDevice</i>	virtual device to use.

#### Returns

Error Status. 0 on success.

**11.60.3.2 EnableDigitalIn()** [1/2] virtual uint32\_t EnableDigitalIn (  
    bool *enable*,  
    unsigned int *virtualDevice* ) [virtual]

Enable the digital data word in the datastream.

#### Parameters

<i>enable</i>	True to enable, False to disable.
<i>virtualDevice</i>	virtual device to use.

#### Returns

Error Status. 0 on success.

**11.60.3.3 EnableDigitalIn()** [2/2] virtual uint32\_t EnableDigitalIn (  
    DigitalDatastreamEnableEnumNet *enable*,  
    unsigned int *virtualDevice* ) [virtual]

Enable digital data words in the datastream.

#### Parameters

<i>enable</i>	True to enable, False to disable.
<i>virtualDevice</i>	virtual device to use.

#### Returns

Error Status. 0 on success.

**11.60.3.4 EnableTimestamp()** virtual uint32\_t EnableTimestamp (  
    bool *enable*,  
    unsigned int *virtualDevice* ) [virtual]

Enable the timestamp data word in the datastream. The timestamp is a 64 bit counter.

#### Parameters

<i>enable</i>	True to enable, False to disable.
<i>virtualDevice</i>	virtual device to use.

**Returns**

Error Status. 0 on success.

**11.60.3.5 GetAnalogGain()** `virtual int32_t GetAnalogGain ( ) [virtual]`

Gets the gain of the analog inputs of the device.

**Returns**

Gain times 1000, a value of 1000 corresponds to a gain of 1.0.

**11.60.3.6 GetEnumerationSpeed()** `int32_t GetEnumerationSpeed ( )`**11.60.3.7 GetGain()** `virtual int32_t GetGain ( ) [virtual]`

Gets the amplifier gain of the device.

**Returns**

Gain times 1000, a value of 1000 corresponds to a gain of 1.0.

**11.60.3.8 SetDigitalOut()** `virtual void SetDigitalOut (   
 unsigned int digout_value,   
 int pulselength ) [virtual]`

Generate a pulse on the digital output.

**Parameters**

<i>digout_value</i>	Bitmask to set on the digital out.
---------------------	------------------------------------

**Parameters**

<i>pulselength</i>	Pulselength in ms.
--------------------	--------------------

**11.60.3.9 SetNumberOfAnalogChannels()** virtual uint32\_t SetNumberOfAnalogChannels (   
 unsigned int *NumberOfChannels\_HS1*,   
 unsigned int *NumberOfChannels\_HS2*,   
 unsigned int *NumberOfChannels\_DSP*,   
 unsigned int *NumberOfChannels\_IF*,   
 unsigned int *virtualDevice* ) [virtual]

Sets the number of analog channels in the datastream for the MEA2100 device.

**Parameters**

<i>NumberOfChannels_HS1</i>	Number of analog channels from the Headstage 1.
-----------------------------	---

**Parameters**

<i>NumberOfChannels_HS2</i>	Number of analog channels from the Headstage 2.
-----------------------------	---

**Parameters**

<i>NumberOfChannels_DSP</i>	Number of data words from the DSP.
-----------------------------	------------------------------------

**Parameters**

<i>NumberOfChannels_IF</i>	Number of analog channels from the Interfaceboard.
----------------------------	--

**Parameters**

<i>virtualDevice</i>	virtualDevice to use.
----------------------	-----------------------

**Returns**

Error Status. 0 on success.



**11.60.3.10 SetNumberOfChannels()** [1/2] virtual uint32\_t SetNumberOfChannels (  
int *NumberOfChannels* ) [virtual]

Sets the number of analog channels in the datastream.

**Parameters**

<i>NumberOfChannels</i>	Number of analog channels.
-------------------------	----------------------------

**Returns**

Error Status. 0 on success.

**11.60.3.11 SetNumberOfChannels()** [2/2] virtual uint32\_t SetNumberOfChannels (  
int *NumberOfChannels*,  
unsigned int *virtualDevice* ) [virtual]

Sets the number of analog channels in the datastream.

**Parameters**

<i>NumberOfChannels</i>	Number of analog channels.
<i>virtualDevice</i>	virtual device to use.

**Returns**

Error Status. 0 on success.

**11.60.3.12 SetTriggerMaskValue()** virtual uint32\_t SetTriggerMaskValue (  
unsigned int *mask*,  
unsigned int *value*,  
unsigned int *virtualDevice* ) [virtual]

Defines a pattern on the digital dataword which will start a trigger when found.

**Parameters**

<i>mask</i>	Bits in the digital dataword which are monitored for a match with value.
-------------	--

**Parameters**

<i>value</i>	Pattern which must match for the trigger to start.
--------------	--

**Returns**

Error Status. 0 on success.

**11.60.3.13 SetTriggerPeriod()** `virtual uint32_t SetTriggerPeriod (`  
`int samples,`  
`unsigned int virtualDevice ) [virtual]`

Sets the maximum number of samples per trigger.

**Parameters**

<i>samples</i>	Number of samples to acquire after the trigger condition is met.
----------------	--

**Returns**

Error Status. 0 on success.

**11.60.4 Property Documentation**

**11.60.4.1 AnalogGain** `virtual int AnalogGain [get]`

The gain of the analog inputs of the device. Value is gain times 1000, a value of 1000 corresponds to a gain of 1.0.

**11.60.4.2 Gain** `virtual int Gain [get]`

The amplifier gain of the device. Value is gain times 1000, a value of 1000 corresponds to a gain of 1.0.

**11.60.4.3 MeaAudioFunctionNet** `CMeaAudioFunctionNet^ MeaAudioFunctionNet [get]`

**11.60.4.4 MeaDigitalDataFunctionNet** `CMeaDigitalDataFunctionNet^ MeaDigitalDataFunctionNet [get]`

**11.60.4.5 MeaFeedbackFunctionNet** [CMeaFeedbackFunctionNet](#)<sup>^</sup> MeaFeedbackFunctionNet [get]

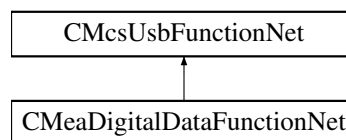
**11.60.4.6 MeFunctionNet** [CMeFunctionNet](#)<sup>^</sup> MeFunctionNet [get]

**11.60.4.7 W2100\_FunctionNet** [CW2100\\_FunctionNet](#)<sup>^</sup> W2100\_FunctionNet [get]

**11.60.4.8 WClassicFunctionNet** [CWClassicFunctionNet](#)<sup>^</sup> WClassicFunctionNet [get]

## 11.61 CMeaDigitalDataFunctionNet Class Reference

Inheritance diagram for CMeaDigitalDataFunctionNet:



### Public Member Functions

- [CMeaDigitalDataFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> meaDigitalFunctionPointerContainer)
- [CMeaDigitalDataFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- void [SetDigitalData](#) (unsigned int digital\_value, unsigned int digital\_value\_mask)  
*Generate a value on the digital output.*
- void [SetDigitalData](#) (unsigned int bit\_number, bool value)  
*Generate a value on the digital output.*
- unsigned int [GetDigitalData](#) ()  
*Get the value of the digital output.*

### Additional Inherited Members

#### 11.61.1 Constructor & Destructor Documentation

**11.61.1.1 CMeaDigitalDataFunctionNet()** [1/2] [CMeaDigitalDataFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb,  
[CMcsUsbFunctionPointerContainer](#)<sup>^</sup> meaDigitalFunctionPointerContainer )

**11.61.1.2 CMeaDigitalDataFunctionNet()** [2/2] `CMeaDigitalDataFunctionNet ( CMcsUsbNet^ mcsusb )`

### 11.61.2 Member Function Documentation

**11.61.2.1 GetDigitalData()** `unsigned int GetDigitalData ( )`

Get the value of the digital output.

#### Returns

Value on the digital data register.

**11.61.2.2 SetDigitalData()** [1/2] `void SetDigitalData ( unsigned int bit_number, bool value )`

Generate a value on the digital output.

#### Parameters

<i>bit_number</i>	Bit number to change.
-------------------	-----------------------

#### Parameters

<i>value</i>	Bit value.
--------------	------------

**11.61.2.3 SetDigitalData()** [2/2] `void SetDigitalData ( unsigned int digital_value, unsigned int digital_value_mask )`

Generate a value on the digital output.

#### Parameters

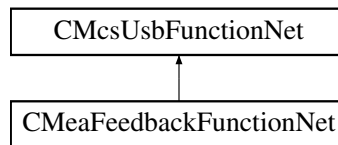
<i>digital_value</i>	Value to set.
----------------------	---------------

## Parameters

<i>digital_value_mask</i>	Mask for change.
---------------------------	------------------

## 11.62 CMeaFeedbackFunctionNet Class Reference

Inheritance diagram for CMeaFeedbackFunctionNet:



## Public Member Functions

- [CMeaFeedbackFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> meaFeedback↔  
FunctionNet)
- [CMeaFeedbackFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- void [FeedbackSetFeedback](#) (unsigned char on, unsigned short digoutmask, unsigned short diginmask)
- unsigned int [FeedbackGetSampleTimerCount](#) ([System::Runtime::InteropServices::Out]unsigned int%  
CurrentCount, [System::Runtime::InteropServices::Out]unsigned int% LastKnownCount, [System::Runtime↔  
::InteropServices::Out]bool% On)
- void [FeedbackSetDigitalMapping](#) (unsigned short channel, unsigned short outmapping, unsigned short in-  
mapping)
- void [FeedbackSetFilterParameter](#) (unsigned char filter, array< short ><sup>^</sup> parameters)
- void [FeedbackSetFilterParameter32](#) (unsigned char filter, array< int ><sup>^</sup> parameters)
- void [FeedbackSetIIRFilterParameter](#) (unsigned char filter, int length, array< double ><sup>^</sup> parameters)
- void [FeedbackSetMkFilter](#) (unsigned char filter, String<sup>^</sup> filtertype, double cheb\_ripple, String<sup>^</sup> passtype, int  
order, double alpha1, double alpha2)
- void [FeedbackSetChannelFilter](#) (short channel, char filter)
- void [FeedbackSetGlobalChannelFilter](#) (char filter, unsigned short firstchannel, unsigned short lastchannel)
- void [FeedbackSetFilterOff](#) ()
- void [FeedbackSetNumberOfSpikeDetectors](#) (unsigned short number)
- void [FeedbackSetSpikeDetectorThreshold](#) (unsigned short position, unsigned short sourcechannel, unsigned  
short resultchannel, unsigned short trigger, unsigned short totzeit, int threshold1, int threshold2, short slope)
- void [FeedbackSetNumberOfRateCounter](#) (unsigned short number)
- void [FeedbackSetRateCounter](#) (unsigned short position, unsigned short sourcechannel, unsigned short re-  
sultchannel)
- void [FeedbackSetNumberOfRateDetectors](#) (unsigned short number)
- void [FeedbackSetRateDetector](#) (unsigned short position, unsigned short resultchannel, unsigned short trig-  
ger, unsigned short totzeit, unsigned short pulses, unsigned int duration1, unsigned int duration2)
- void [FeedbackSetNumberOfLogics](#) (unsigned short number)
- void [FeedbackSetLogic](#) (unsigned short position, array< unsigned short ><sup>^</sup> sourcechannel, unsigned short  
resultchannel, unsigned int lookup)
- void [FeedbackSetNumberOfTriggers](#) (unsigned short number)
- void [FeedbackSetTrigger](#) (unsigned short position, unsigned short sourcechannel, unsigned short resultchan-  
nel, unsigned short trigger, unsigned short totzeit)
- void [FeedbackSetAnalogSource](#) (AnalogSourceEnumNet AnalogSource, unsigned int Channels, unsigned int  
Offset)

## Additional Inherited Members

### 11.62.1 Constructor & Destructor Documentation

**11.62.1.1 CMeaFeedbackFunctionNet()** [1/2] CMeaFeedbackFunctionNet (   
 CMcsUsbNet^ *mcsusb*,   
 CMcsUsbFunctionPointerContainer^ *meaFeedbackFunctionNet* )

**11.62.1.2 CMeaFeedbackFunctionNet()** [2/2] CMeaFeedbackFunctionNet (   
 CMcsUsbNet^ *mcsusb* )

### 11.62.2 Member Function Documentation

**11.62.2.1 FeedbackGetSampleTimerCount()** unsigned int FeedbackGetSampleTimerCount (   
 [System::Runtime::InteropServices::Out] unsigned int% *CurrentCount*,   
 [System::Runtime::InteropServices::Out] unsigned int% *LastKnownCount*,   
 [System::Runtime::InteropServices::Out] bool% *On* )

**11.62.2.2 FeedbackSetAnalogSource()** void FeedbackSetAnalogSource (   
 AnalogSourceEnumNet *AnalogSource*,   
 unsigned int *Channels*,   
 unsigned int *Offset* )

**11.62.2.3 FeedbackSetChannelFilter()** void FeedbackSetChannelFilter (   
 short *channel*,   
 char *filter* )

**11.62.2.4 FeedbackSetDigitalMapping()** void FeedbackSetDigitalMapping (   
 unsigned short *channel*,   
 unsigned short *outmapping*,   
 unsigned short *inmapping* )

**11.62.2.5 FeedbackSetFeedback()** void FeedbackSetFeedback (   
 unsigned char *on*,   
 unsigned short *digoutmask*,   
 unsigned short *diginmask* )

**11.62.2.6 FeedbackSetFilterOff()** void FeedbackSetFilterOff ( )

**11.62.2.7 FeedbackSetFilterParameter()** void FeedbackSetFilterParameter (   
 unsigned char *filter*,   
 array< short >^ *parameters* )

**11.62.2.8 FeedbackSetFilterParameter32()** void FeedbackSetFilterParameter32 (   
 unsigned char *filter*,   
 array< int >^ *parameters* )

**11.62.2.9 FeedbackSetGlobalChannelFilter()** void FeedbackSetGlobalChannelFilter (   
 char *filter*,   
 unsigned short *firstchannel*,   
 unsigned short *lastchannel* )

**11.62.2.10 FeedbackSetIIRFilterParameter()** void FeedbackSetIIRFilterParameter (   
 unsigned char *filter*,   
 int *length*,   
 array< double >^ *parameters* )

**11.62.2.11 FeedbackSetLogic()** void FeedbackSetLogic (   
 unsigned short *position*,   
 array< unsigned short >^ *sourcechannel*,   
 unsigned short *resultchannel*,   
 unsigned int *lookup* )

**11.62.2.12 FeedbackSetMkFilter()** void FeedbackSetMkFilter (   
 unsigned char *filter*,   
 String^ *filtertype*,   
 double *cheb\_ribble*,   
 String^ *passtype*,   
 int *order*,   
 double *alpha1*,   
 double *alpha2* )

**11.62.2.13 FeedbackSetNumberOfLogics()** void FeedbackSetNumberOfLogics (   
 unsigned short *number* )

**11.62.2.14 FeedbackSetNumberOfRateCounter()** void FeedbackSetNumberOfRateCounter (   
 unsigned short *number* )

**11.62.2.15 FeedbackSetNumberOfRateDetectors()** void FeedbackSetNumberOfRateDetectors (   
 unsigned short *number* )

**11.62.2.16 FeedbackSetNumberOfSpikeDetectors()** void FeedbackSetNumberOfSpikeDetectors (   
 unsigned short *number* )

**11.62.2.17 FeedbackSetNumberOfTriggers()** void FeedbackSetNumberOfTriggers (   
 unsigned short *number* )

**11.62.2.18 FeedbackSetRateCounter()** void FeedbackSetRateCounter (   
 unsigned short *position*,   
 unsigned short *sourcechannel*,   
 unsigned short *resultchannel* )

**11.62.2.19 FeedbackSetRateDetector()** void FeedbackSetRateDetector (   
 unsigned short *position*,   
 unsigned short *resultchannel*,   
 unsigned short *trigger*,   
 unsigned short *totzeit*,   
 unsigned short *pulses*,   
 unsigned int *duration1*,   
 unsigned int *duration2* )

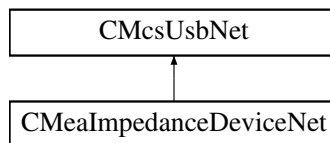


**11.62.2.20 FeedbackSetSpikeDetectorThreshold()** `void FeedbackSetSpikeDetectorThreshold (`  
     `unsigned short position,`  
     `unsigned short sourcechannel,`  
     `unsigned short resultchannel,`  
     `unsigned short trigger,`  
     `unsigned short totzeit,`  
     `int threshold1,`  
     `int threshold2,`  
     `short slope )`

**11.62.2.21 FeedbackSetTrigger()** `void FeedbackSetTrigger (`  
     `unsigned short position,`  
     `unsigned short sourcechannel,`  
     `unsigned short resultchannel,`  
     `unsigned short trigger,`  
     `unsigned short totzeit )`

## 11.63 CMealImpedanceDeviceNet Class Reference

Inheritance diagram for CMealImpedanceDeviceNet:



### Public Member Functions

- [CMealImpedanceDeviceNet](#) ()
- [~CMealImpedanceDeviceNet](#) ()
- void [StartMeasurement](#) (unsigned short channel)
- unsigned short [GetReady](#) ()
- unsigned short [GetArraySize](#) ()
- array< unsigned short > ^ [GetResult](#) ()
- unsigned short [GetAdapterCode](#) ()
- unsigned int [GetImpedanceTestFrequency](#) ()
- void [SetImpedanceTestFrequency](#) (unsigned int TestFrequency\_Hertz)

### Additional Inherited Members

#### 11.63.1 Constructor & Destructor Documentation

**11.63.1.1 CMealImpedanceDeviceNet()** `CMealImpedanceDeviceNet ( )`

11.63.1.2 `~CMeaImpedanceDeviceNet()` `~CMeaImpedanceDeviceNet ( )`

### 11.63.2 Member Function Documentation

11.63.2.1 `GetAdapterCode()` `unsigned short GetAdapterCode ( )`

11.63.2.2 `GetArraySize()` `unsigned short GetArraySize ( )`

11.63.2.3 `GetImpedanceTestFrequency()` `unsigned int GetImpedanceTestFrequency ( )`

11.63.2.4 `GetReady()` `unsigned short GetReady ( )`

11.63.2.5 `GetResult()` `array<unsigned short> ^ GetResult ( )`

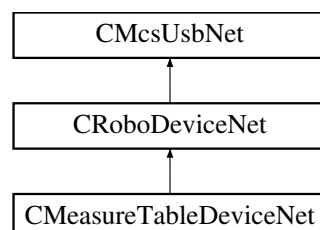
11.63.2.6 `SetImpedanceTestFrequency()` `void SetImpedanceTestFrequency (`  
`unsigned int TestFrequency_Hertz )`

11.63.2.7 `StartMeasurement()` `void StartMeasurement (`  
`unsigned short channel )`

## 11.64 CMeasureTableDeviceNet Class Reference

[CMeasureTableDeviceNet](#) is the to control the MCS HLA device

Inheritance diagram for CMeasureTableDeviceNet:



### Public Member Functions

- [CMeasureTableDeviceNet](#) (void)

### Properties

- [CMcsBus\\_SensorNet^ Sensor](#) [get]

### Additional Inherited Members

#### 11.64.1 Detailed Description

[CMeasureTableDeviceNet](#) is the to control the MCS HLA device

#### 11.64.2 Constructor & Destructor Documentation

**11.64.2.1 CMeasureTableDeviceNet()** [CMeasureTableDeviceNet](#) (  
void )

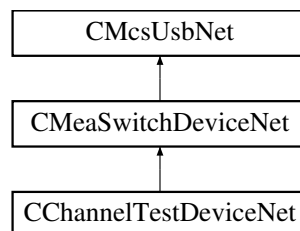
#### 11.64.3 Property Documentation

**11.64.3.1 Sensor** [CMcsBus\\_SensorNet^ Sensor](#) [get]

## 11.65 CMeaSwitchDeviceNet Class Reference

The class to control the USB-MEA-Switch.

Inheritance diagram for CMeaSwitchDeviceNet:



**Public Member Functions**

- [CMeaSwitchDeviceNet](#) ()  
*Constructor.*
- [~CMeaSwitchDeviceNet](#) ()  
*Destructor.*
- unsigned short [GetNumber](#) ()  
*Gets the number of boards in the device.*
- array< unsigned char > ^ [GetPattern](#) ()  
*Gets the pattern of the switches that are currently set in the device as char array.*
- array< bool > ^ [GetPatternBool](#) ()  
*Gets the pattern of the switches that are currently set in the device as bools.*
- void [SetPattern](#) (array< unsigned char >^ pattern)  
*Sets the pattern of switches from a char array.*
- void [SetPatternBool](#) (array< bool >^ pattern)  
*Sets the pattern of switches from a.*

**Additional Inherited Members****11.65.1 Detailed Description**

The class to control the USB-MEA-Switch.

This class controls the settings of the USB-MEA-Switch. The box has two inputs for signals from a MEA amplifier. Each of the 64 outputs can be connected to one of the MEAs at the same channel.

**11.65.2 Constructor & Destructor Documentation****11.65.2.1 CMeaSwitchDeviceNet()** [CMeaSwitchDeviceNet](#) ( )

Constructor.

**11.65.2.2 ~CMeaSwitchDeviceNet()** [~CMeaSwitchDeviceNet](#) ( )

Destructor.

**11.65.3 Member Function Documentation****11.65.3.1 GetNumber()** unsigned short [GetNumber](#) ( )

Gets the number of boards in the device.

The MEA-Switch are delivered with 64 or 128 channels

### 11.65.3.2 GetPattern() `array<unsigned char> ^ GetPattern ( )`

Gets the pattern of the switches that are currently set in the device as char array.

### 11.65.3.3 GetPatternBool() `array<bool> ^ GetPatternBool ( )`

Gets the pattern of the switches that are currently set in the device as bools.

### 11.65.3.4 SetPattern() `void SetPattern ( array< unsigned char >^ pattern )`

Sets the pattern of switches from a char array.

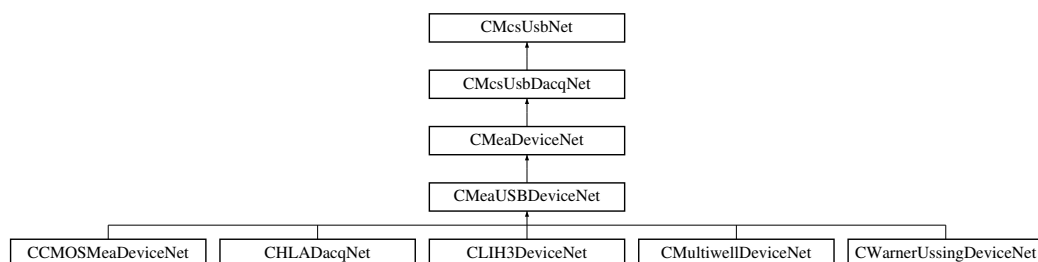
### 11.65.3.5 SetPatternBool() `void SetPatternBool ( array< bool >^ pattern )`

Sets the pattern of switches from a.

## 11.66 CMeaUSBDeviceNet Class Reference

Class for data acquisition via ME and MEA USB amplifiers

Inheritance diagram for CMeaUSBDeviceNet:



### Public Member Functions

- [CMeaUSBDeviceNet](#) ([OnChannelData](#)^ channelData, [OnError](#)^ error)  
*Initializes a new instance of [CMeaDeviceNet](#) class.*
- [CMeaUSBDeviceNet](#) ()  
*Initializes a new instance of [CMeaDeviceNet](#) class.*
- [~CMeaUSBDeviceNet](#) ()

**Additional Inherited Members****11.66.1 Detailed Description**

Class for data acquisition via ME and MEA USB amplifiers

**11.66.2 Constructor & Destructor Documentation**
**11.66.2.1 CMeaUSBDeviceNet() [1/2]** `CMeaUSBDeviceNet (`  
`OnChannelData^ channelData,`  
`OnError^ error )`

Initializes a new instance of `CMeaDeviceNet` class.

**Parameters**

<code>channelData</code>	Handler to call when new data is available.
--------------------------	---

**Parameters**

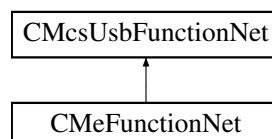
<code>error</code>	Handler to call when an error occurs.
--------------------	---------------------------------------

**11.66.2.2 CMeaUSBDeviceNet() [2/2]** `CMeaUSBDeviceNet ( )`

Initializes a new instance of `CMeaDeviceNet` class.

**11.66.2.3 ~CMeaUSBDeviceNet()** `~CMeaUSBDeviceNet ( )`
**11.67 CMeFunctionNet Class Reference**

Inheritance diagram for CMeFunctionNet:



## Public Member Functions

- [CMeFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> meFunctionPointer↔ Container)  
*Initializes a new instance of the [CDacCalibrationFunctionNet](#) class.*
- [CMeFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- virtual [~CMeFunctionNet](#) (void)
- [!CMeFunctionNet](#) (void)
- void [SetTransformer](#) (unsigned int index, bool onoff)

## Additional Inherited Members

### 11.67.1 Detailed Description

### 11.67.2 Constructor & Destructor Documentation

**11.67.2.1 CMeFunctionNet()** [1/2] [CMeFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb,  
[CMcsUsbFunctionPointerContainer](#)<sup>^</sup> meFunctionPointerContainer )

Initializes a new instance of the [CDacCalibrationFunctionNet](#) class.

**11.67.2.2 CMeFunctionNet()** [2/2] [CMeFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb )

**11.67.2.3 ~CMeFunctionNet()** virtual [~CMeFunctionNet](#) (  
void ) [virtual]

**11.67.2.4 !CMeFunctionNet()** [!CMeFunctionNet](#) (  
void )

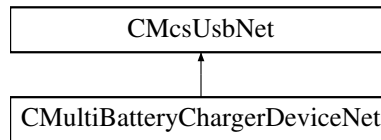
### 11.67.3 Member Function Documentation

**11.67.3.1 SetTransformer()** void [SetTransformer](#) (  
unsigned int index,  
bool onoff )

## 11.68 CMultiBatteryChargerDeviceNet Class Reference

[CMultiBatteryChargerDeviceNet](#) is the class to access the MBC-08 device.

Inheritance diagram for CMultiBatteryChargerDeviceNet:



### Public Member Functions

- [CMultiBatteryChargerDeviceNet](#) ()  
*Initializes a new instance of the [CMultiBatteryChargerDeviceNet](#) class.*
- virtual [~CMultiBatteryChargerDeviceNet](#) ()
- [!CMultiBatteryChargerDeviceNet](#) ()
- uint32\_t [GetChargeCurrent](#) (uint32\_t NrChannel)  
*gets the charge current; unit: mA*
- uint32\_t [GetDischargeCurrent](#) (uint32\_t NrChannel)  
*gets the discharge current; unit: mA*
- void [SetDischargeCurrentSetPoint](#) (uint32\_t NrChannel, uint32\_t DischargeCurrent\_mA)  
*sets the setpoint for the discharge current; unit: mA*
- uint32\_t [GetDischargeCurrentSetPoint](#) (uint32\_t NrChannel)  
*gets the setpoint for the discharge current; unit: mA*
- void [SetFinalDischargeVoltage](#) (uint32\_t NrChannel, uint32\_t FinalDischargeVoltage\_mV)  
*sets the final discharge voltage; unit: mV*
- uint32\_t [GetFinalDischargeVoltage](#) (uint32\_t NrChannel)  
*gets the final discharge voltage; unit: mV*
- uint32\_t [GetDischargeCapacity](#) (uint32\_t NrChannel)  
*gets the discharge capacity; unit: ?Ah*
- uint32\_t [GetChargeCapacity](#) (uint32\_t NrChannel)  
*gets the charge capacity; unit: ?Ah*
- uint32\_t [GetBatteryVoltage](#) (uint32\_t NrChannel)  
*gets the battery voltage; unit: mV*
- uint32\_t [GetChannels](#) ()  
*gets number of channels*
- void [SetRatedCapacityVolatile](#) (uint32\_t NrChannel, MbcRatedCapacityEnumNet NewRatedCapacity)  
*sets the rated capacity (i.e. charge current) without storing it persistently*
- void [SetChargingMode](#) (uint32\_t NrChannel, MbcChargingModeEnumNet NewOperatingMode)  
*sets the charging mode: StorageCharge, LowCurrentCharge and HighCurrentCharge*
- MbcChargingModeEnumNet [GetChargingMode](#) (uint32\_t NrChannel)  
*gets the charging mode: StorageCharge, LowCurrentCharge and HighCurrentCharge*
- MbcChannelStateEnumNet [GetChannelState](#) (uint32\_t NrChannel)  
*gets the channel state: IdleNoBattery, IdleChargeFinished, CapacityTestPreCharge, CapacityTestDischarge, StorageCharge, LowCurrentCharge, HighCurrentCharge*
- void [CapacityTest](#) (uint32\_t NrChannel)  
*start capacity test on channel*
- void [ChannelReset](#) (uint32\_t NrChannel)  
*cancel charging and capacity test functions; check if battery is connected*



- void [SetChargingPCoefficient](#) (uint32\_t pCoefficient)  
*sets the p-coefficient for charging in mA/V / nominal charging current*
- uint32\_t [GetChargingPCoefficient](#) ()  
*gets the p-coefficient for charging in mA/V / nominal charging current*
- void [SetRatedCapacity](#) (uint32\_t NrChannel, MbcRatedCapacityEnumNet NewRatedCapacity)  
*sets the rated capacity*
- MbcRatedCapacityEnumNet [GetRatedCapacity](#) (uint32\_t NrChannel)  
*gets the rated capacity*

## Additional Inherited Members

### 11.68.1 Detailed Description

[CMultiBatteryChargerDeviceNet](#) is the class to access the MBC-08 device.

### 11.68.2 Constructor & Destructor Documentation

#### 11.68.2.1 CMultiBatteryChargerDeviceNet() [CMultiBatteryChargerDeviceNet](#) ( )

Initializes a new instance of the [CMultiBatteryChargerDeviceNet](#) class.

#### 11.68.2.2 ~CMultiBatteryChargerDeviceNet() [virtual ~CMultiBatteryChargerDeviceNet](#) ( ) [virtual]

#### 11.68.2.3 "!CMultiBatteryChargerDeviceNet() [!CMultiBatteryChargerDeviceNet](#) ( )

### 11.68.3 Member Function Documentation

#### 11.68.3.1 CapacityTest() [void CapacityTest](#) ( uint32\_t NrChannel )

start capacity test on channel

##### Parameters

<i>NrChannel</i>	the channel number
------------------	--------------------

**11.68.3.2 ChannelReset()** `void ChannelReset (`  
    `uint32_t NrChannel )`

cancel charging and capacity test functions; check if battery is connected

**Parameters**

<i>NrChannel</i>	the channel number
------------------	--------------------

**11.68.3.3 GetBatteryVoltage()** `uint32_t GetBatteryVoltage (`  
    `uint32_t NrChannel )`

gets the battery voltage; unit: mV

**Parameters**

<i>NrChannel</i>	the channel number
------------------	--------------------

**Returns**

the battery voltage in mV

**11.68.3.4 GetChannels()** `uint32_t GetChannels ( )`

gets number of channels

**Returns**

number of channels

**11.68.3.5 GetChannelState()** `MbcChannelStateEnumNet GetChannelState (`  
    `uint32_t NrChannel )`

gets the channel state: IdleNoBattery, IdleChargeFinished, CapacityTestPreCharge, CapacityTestDischarge, StorageCharge, LowCurrentCharge, HighCurrentCharge

**Parameters**

<i>NrChannel</i>	the channel number
------------------	--------------------

**Returns**

the current state

**11.68.3.6 GetChargeCapacity()** `uint32_t GetChargeCapacity (`  
`uint32_t NrChannel )`

gets the charge capacity; unit: ?Ah

**Parameters**

<i>NrChannel</i>	the channel number
------------------	--------------------

**Returns**

the capacity in uAh

**11.68.3.7 GetChargeCurrent()** `uint32_t GetChargeCurrent (`  
`uint32_t NrChannel )`

gets the charge current; unit: mA

**Parameters**

<i>NrChannel</i>	the channel number
------------------	--------------------

**Returns**

the measured charge current in mA

**11.68.3.8 GetChargingMode()** `MbcChargingModeEnumNet GetChargingMode (`  
`uint32_t NrChannel )`

gets the charging mode: StorageCharge, LowCurrentCharge and HighCurrentCharge

**Parameters**

<i>NrChannel</i>	the channel number
------------------	--------------------

**Returns**

the charging mode

**11.68.3.9 GetChargingPCoefficient()** `uint32_t GetChargingPCoefficient ( )`

gets the p-coefficient for charging in mA/V / nominal charging current

**Returns**

the p-coefficient

**11.68.3.10 GetDischargeCapacity()** `uint32_t GetDischargeCapacity (   
uint32_t NrChannel )`

gets the discharge capacity; unit: ?Ah

**Parameters**

<i>NrChannel</i>	the channel number
------------------	--------------------

**Returns**

the capacity in uAh

**11.68.3.11 GetDischargeCurrent()** `uint32_t GetDischargeCurrent (   
uint32_t NrChannel )`

gets the discharge current; unit: mA

**Parameters**

<i>NrChannel</i>	the channel number
------------------	--------------------

**Returns**

the measured discharge current in mA

**11.68.3.12 GetDischargeCurrentSetPoint()** `uint32_t GetDischargeCurrentSetPoint (   
uint32_t NrChannel )`

gets the setpoint for the discharge current; unit: mA

**Parameters**

<i>NrChannel</i>	the channel number
------------------	--------------------

**Returns**

the discharge current in mA

**11.68.3.13 GetFinalDischargeVoltage()** `uint32_t GetFinalDischargeVoltage (   
 uint32_t NrChannel )`

gets the final discharge voltage; unit: mV

**Parameters**

<i>NrChannel</i>	the channel number
------------------	--------------------

**Returns**

the battery voltage in mV at the end of discharge

**11.68.3.14 GetRatedCapacity()** `MbcRatedCapacityEnumNet GetRatedCapacity (   
 uint32_t NrChannel )`

gets the rated capacity

**Parameters**

<i>NrChannel</i>	the channel number
------------------	--------------------

**Returns**

the capacity

**11.68.3.15 SetChargingMode()** `void SetChargingMode (   
 uint32_t NrChannel,   
 MbcChargingModeEnumNet NewOperatingMode )`

sets the charging mode: StorageCharge, LowCurrentCharge and HighCurrentCharge

**Parameters**

<i>NrChannel</i>	the channel number
<i>NewOperatingMode</i>	the charging mode

**11.68.3.16 SetChargingPCoefficient()** void SetChargingPCoefficient (   
 uint32\_t *pCoefficient* )

sets the p-coefficient for charging in mA/V / nominal charging current

Parameters

<i>pCoefficient</i>	the p-coefficient
---------------------	-------------------

**11.68.3.17 SetDischargeCurrentSetPoint()** void SetDischargeCurrentSetPoint (   
 uint32\_t *NrChannel*,   
 uint32\_t *DischargeCurrent\_mA* )

sets the setpoint for the discharge current; unit: mA

Parameters

<i>NrChannel</i>	the channel number
<i>DischargeCurrent_mA</i>	the discharge current in mA

**11.68.3.18 SetFinalDischargeVoltage()** void SetFinalDischargeVoltage (   
 uint32\_t *NrChannel*,   
 uint32\_t *FinalDischargeVoltage\_mV* )

sets the final discharge voltage; unit: mV

Parameters

<i>NrChannel</i>	the channel number
<i>FinalDischargeVoltage_mV</i>	the battery voltage in mV at the end of discharge

**11.68.3.19 SetRatedCapacity()** void SetRatedCapacity (   
 uint32\_t *NrChannel*,   
 MbcRatedCapacityEnumNet *NewRatedCapacity* )

sets the rated capacity

Parameters

<i>NrChannel</i>	the channel number
<i>NewRatedCapacity</i>	the capacity

**11.68.3.20 SetRatedCapacityVolatile()** `void SetRatedCapacityVolatile (`  
`uint32_t NrChannel,`  
`MbcRatedCapacityEnumNet NewRatedCapacity )`

sets the rated capacity (i.e. charge current) without storing it persistently

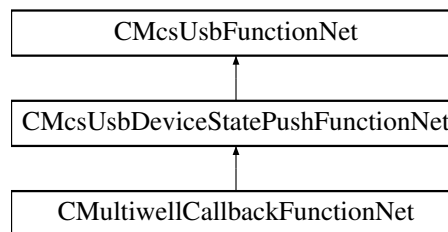
#### Parameters

<i>NrChannel</i>	the channel number
<i>NewRatedCapacity</i>	the capacity

## 11.69 CMultiwellCallbackFunctionNet Class Reference

[CMultiwellCallbackFunctionNet](#) is the class to access the Multiwell-Mini-Stimulator

Inheritance diagram for CMultiwellCallbackFunctionNet:



### Public Member Functions

- delegate void [OnGetPlateClampStateByHeadstage](#) (uint32\_t Headstage, PlateClampEnumNet plateState)
- delegate void [OnGetPlateTypeByHeadstage](#) (uint32\_t Headstage, MultiwellPlateTypeEnumNet plateType)
- delegate void [OnIsPlateTypeValidByHeadstage](#) (uint32\_t Headstage, bool isValid)
- [CMultiwellCallbackFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> pMultiwell↔  
 CallbackFunctionPointerContainer)  
*Initializes a new instance of the [CMultiwellCallbackFunctionNet](#) class.*
- [CMultiwellCallbackFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- virtual [~CMultiwellCallbackFunctionNet](#) ()
- [!CMultiwellCallbackFunctionNet](#) ()
- PlateClampEnumNet [GetPlateClampStateByHeadstage](#) (uint32\_t Headstage)  
*Gets the state of the plate*
- MultiwellPlateTypeEnumNet [GetPlateTypeByHeadstage](#) (uint32\_t Headstage)  
*Gets the plate type.*
- bool [IsPlateTypeValidByHeadstage](#) (uint32\_t Headstage)  
*Checks whether the plate type is valid, meaning all pins have contact.*

### Events

- [OnGetPlateClampStateByHeadstage](#)<sup>^</sup> [GetPlateClampStateByHeadstageEvent](#) [add, remove, raise]  
*Event fires when the plate state for the headstage number has changed*
- [OnGetPlateTypeByHeadstage](#)<sup>^</sup> [GetPlateTypeByHeadstageEvent](#) [add, remove, raise]  
*Event fires when the plate type for the headstage to query has changed*
- [OnIsPlateTypeValidByHeadstage](#)<sup>^</sup> [IsPlateTypeValidByHeadstageEvent](#) [add, remove, raise]  
*Event fires when "true" when all pins have contact, otherwise "false" for the headstage to query has changed*

## Additional Inherited Members

### 11.69.1 Detailed Description

[CMultiwellCallbackFunctionNet](#) is the class to access the Multiwell-Mini-Stimulator

### 11.69.2 Constructor & Destructor Documentation

**11.69.2.1 CMultiwellCallbackFunctionNet()** [1/2] [CMultiwellCallbackFunctionNet](#) (   
 [CMcsUsbNet](#)<sup>^</sup> *mcsusb*,   
 [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> *pMultiwellCallbackFunctionPointerContainer* )

Initializes a new instance of the [CMultiwellCallbackFunctionNet](#) class.

**11.69.2.2 CMultiwellCallbackFunctionNet()** [2/2] [CMultiwellCallbackFunctionNet](#) (   
 [CMcsUsbNet](#)<sup>^</sup> *mcsusb* )

**11.69.2.3 ~CMultiwellCallbackFunctionNet()** `virtual ~CMultiwellCallbackFunctionNet ( )` [virtual]

**11.69.2.4 ~!CMultiwellCallbackFunctionNet()** `!CMultiwellCallbackFunctionNet ( )`

### 11.69.3 Member Function Documentation

**11.69.3.1 GetPlateClampStateByHeadstage()** `PlateClampEnumNet GetPlateClampStateByHeadstage (   
 uint32_t Headstage )`

Gets the state of the plate

#### Parameters

<i>Headstage</i>	The headstage number
------------------	----------------------

#### Returns

The plate state



**11.69.3.2 GetPlateTypeByHeadstage()** `MultiwellPlateTypeEnumNet GetPlateTypeByHeadstage ( uint32_t Headstage )`

Gets the plate type.

**Parameters**

<i>Headstage</i>	The headstage to query.
------------------	-------------------------

**Returns**

the plate type

**11.69.3.3 IsPlateTypeValidByHeadstage()** `bool IsPlateTypeValidByHeadstage ( uint32_t Headstage )`

Checks whether the plate type is valid, meaning all pins have contact.

**Parameters**

<i>Headstage</i>	The headstage to query.
------------------	-------------------------

**Returns**

"true" when all pins have contact, otherwise "false".

**11.69.3.4 OnGetPlateClampStateByHeadstage()** `delegate void OnGetPlateClampStateByHeadstage ( uint32_t Headstage, PlateClampEnumNet plateState )`

**11.69.3.5 OnGetPlateTypeByHeadstage()** `delegate void OnGetPlateTypeByHeadstage ( uint32_t Headstage, MultiwellPlateTypeEnumNet plateType )`

**11.69.3.6 OnIsPlateTypeValidByHeadstage()** `delegate void OnIsPlateTypeValidByHeadstage ( uint32_t Headstage, bool isValid )`

#### 11.69.4 Event Documentation

**11.69.4.1 GetPlateClampStateByHeadstageEvent** [OnGetPlateClampStateByHeadstage](#)<sup>^</sup> [GetPlateClampStateByHeadstageEvent](#) [add], [remove], [raise]

Event fires when the plate state for the headstage number has changed

**11.69.4.2 GetPlateTypeByHeadstageEvent** [OnGetPlateTypeByHeadstage](#)<sup>^</sup> [GetPlateTypeByHeadstageEvent](#) [add], [remove], [raise]

Event fires when the plate type for the headstage to query has changed

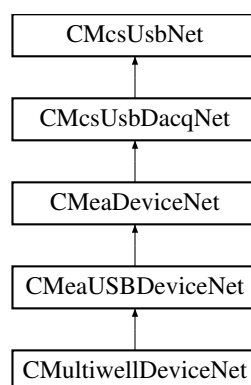
**11.69.4.3 IsPlateTypeValidByHeadstageEvent** [OnIsPlateTypeValidByHeadstage](#)<sup>^</sup> [IsPlateTypeValidByHeadstageEvent](#) [add], [remove], [raise]

Event fires when "true" when all pins have contact, otherwise "false" for the headstage to query has changed

## 11.70 CMultiwellDeviceNet Class Reference

[CMultiwellDeviceNet](#) is the class to access the HEKA LIH3 device.

Inheritance diagram for CMultiwellDeviceNet:



## Public Member Functions

- [CMultiwellDeviceNet](#) ()  
*Initializes a new instance of the [CMultiwellDeviceNet](#) class.*
- virtual [~CMultiwellDeviceNet](#) ()
- [ICMultiwellDeviceNet](#) ()
- [PlateClampEnumNet](#) [GetPlateClampState](#) ()  
*Gets the state of the Multiwell plate clamp.*
- [PlateClampEnumNet](#) [GetPlateClampStateByHeadstage](#) (uint32\_t Headstage)  
*Gets the state of the plate*
- void [OpenPlateClamp](#) ()  
*Opens the plate clamp.*
- void [ClosePlateClamp](#) ()  
*Closes the plate clamp.*
- void [StopPlateClamp](#) ()  
*Stops the plate clamp movement.*
- uint32\_t [GetPlateClampLockState](#) ()  
*Gets the state of the plate clamp lock.*
- void [LockPlateClamp](#) ()  
*Locks the plate clamp.*
- void [UnlockPlateClamp](#) ()  
*Unlocks the plate clamp.*
- [MultiwellPlateTypeEnumNet](#) [GetPlateType](#) ()  
*Gets the plate type.*
- [MultiwellPlateTypeEnumNet](#) [GetPlateTypeByHeadstage](#) (uint32\_t Headstage)  
*Gets the plate type.*
- void [SetPlateType](#) ([MultiwellPlateTypeEnumNet](#) plateType)  
*Sets the plate type.*
- void [SetPlateTypeByHeadstage](#) (uint32\_t Headstage, [MultiwellPlateTypeEnumNet](#) plateType)  
*Sets the plate type.*
- void [SetPlateMux](#) (uint32\_t muxSelection)  
*Selects a one quarter of the electrodes on a high density Multiwell plate.*
- void [SetPlateMuxByHeadstage](#) (uint32\_t Headstage, uint32\_t muxSelection)  
*Selects a one quarter of the electrodes on a high density Multiwell plate.*
- uint32\_t [GetPlateMux](#) ()  
*Gets the selected quarter of the electrodes on a high density Multiwell plate.*
- uint32\_t [GetPlateMuxByHeadstage](#) (uint32\_t Headstage)  
*Gets the selected quarter of the electrodes on a high density Multiwell plate.*
- bool [IsPlateTypeValid](#) ()  
*Checks whether the plate type is valid, meaning all pins have contact.*
- bool [IsPlateTypeValidByHeadstage](#) (uint32\_t Headstage)  
*Checks whether the plate type is valid, meaning all pins have contact.*

## Additional Inherited Members

### 11.70.1 Detailed Description

[CMultiwellDeviceNet](#) is the class to access the HEKA LIH3 device.

## 11.70.2 Constructor & Destructor Documentation

### 11.70.2.1 CMultiwellDeviceNet() `CMultiwellDeviceNet ( )`

Initializes a new instance of the `CMultiwellDeviceNet` class.

### 11.70.2.2 ~CMultiwellDeviceNet() `virtual ~CMultiwellDeviceNet ( ) [virtual]`

### 11.70.2.3 "!CMultiwellDeviceNet() `!CMultiwellDeviceNet ( )`

## 11.70.3 Member Function Documentation

### 11.70.3.1 ClosePlateClamp() `void ClosePlateClamp ( )`

Closes the plate clamp.

### 11.70.3.2 GetPlateClampLockState() `uint32_t GetPlateClampLockState ( )`

Gets the state of the plate clamp lock.

#### Returns

the state of the plate lock (unlocked/locked)

### 11.70.3.3 GetPlateClampState() `PlateClampEnumNet GetPlateClampState ( )`

Gets the state of the Multiwell plate clamp.

#### Returns

the state of the plate clamp (open/closed)

### 11.70.3.4 GetPlateClampStateByHeadstage() `PlateClampEnumNet GetPlateClampStateByHeadstage ( uint32_t Headstage )`

Gets the state of the plate

## Parameters

<i>Headstage</i>	The headstage number
------------------	----------------------

## Returns

The plate state

**11.70.3.5 GetPlateMux()** `uint32_t GetPlateMux ( )`

Gets the selected quarter of the electrodes on a high density Multiwell plate.

## Returns

the selected quarter

**11.70.3.6 GetPlateMuxByHeadstage()** `uint32_t GetPlateMuxByHeadstage (   
uint32_t Headstage )`

Gets the selected quarter of the electrodes on a high density Multiwell plate.

## Parameters

<i>Headstage</i>	The headstage to query.
------------------	-------------------------

## Returns

the selected quarter

**11.70.3.7 GetPlateType()** `MultiwellPlateTypeEnumNet GetPlateType ( )`

Gets the plate type.

## Returns

the plate type

**11.70.3.8 GetPlateTypeByHeadstage()** `MultiwellPlateTypeEnumNet GetPlateTypeByHeadstage (   
uint32_t Headstage )`

Gets the plate type.

## Parameters

<i>Headstage</i>	The headstage to query.
------------------	-------------------------

## Returns

the plate type

**11.70.3.9 IsPlateTypeValid()** `bool IsPlateTypeValid ( )`

Checks whether the plate type is valid, meaning all pins have contact.

## Returns

"true" when all pins have contact, otherwise "false".

**11.70.3.10 IsPlateTypeValidByHeadstage()** `bool IsPlateTypeValidByHeadstage (   
uint32_t Headstage )`

Checks whether the plate type is valid, meaning all pins have contact.

## Parameters

<i>Headstage</i>	The headstage to query.
------------------	-------------------------

## Returns

"true" when all pins have contact, otherwise "false".

**11.70.3.11 LockPlateClamp()** `void LockPlateClamp ( )`

Locks the plate clamp.

**11.70.3.12 OpenPlateClamp()** `void OpenPlateClamp ( )`

Opens the plate clamp.

**11.70.3.13 SetPlateMux()** `void SetPlateMux (   
uint32_t muxSelection )`

Selects a one quarter of the electrodes on a high density Multiwell plate.

## Parameters

<i>muxSelection</i>	the selected quarter
---------------------	----------------------

**11.70.3.14 SetPlateMuxByHeadstage()** `void SetPlateMuxByHeadstage (`  
    `uint32_t Headstage,`  
    `uint32_t muxSelection )`

Selects a one quarter of the electrodes on a high density Multiwell plate.

## Parameters

<i>Headstage</i>	The headstage to query.
<i>muxSelection</i>	the selected quarter

**11.70.3.15 SetPlateType()** `void SetPlateType (`  
    `MultiwellPlateTypeEnumNet plateType )`

Sets the plate type.

## Parameters

<i>plateType</i>	the plate type
------------------	----------------

**11.70.3.16 SetPlateTypeByHeadstage()** `void SetPlateTypeByHeadstage (`  
    `uint32_t Headstage,`  
    `MultiwellPlateTypeEnumNet plateType )`

Sets the plate type.

## Parameters

<i>Headstage</i>	The headstage to query.
<i>plateType</i>	the plate type

**11.70.3.17 StopPlateClamp()** `void StopPlateClamp ( )`

Stops the plate clamp movement.

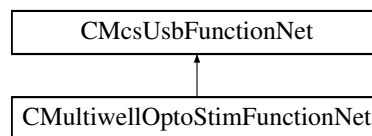
### 11.70.3.18 UnlockPlateClamp() void UnlockPlateClamp ( )

Unlocks the plate clamp.

## 11.71 CMultiwellOptoStimFunctionNet Class Reference

[CMultiwellOptoStimFunctionNet](#) is the class to access the optical properties of the Multiwell Optostim device

Inheritance diagram for CMultiwellOptoStimFunctionNet:



### Public Member Functions

- [CMultiwellOptoStimFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> pMultiwellOptoStimFunctionPointerContainer)  
*Initializes a new instance of the [CMultiwellOptoStimFunctionNet](#) class.*
- [CMultiwellOptoStimFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- virtual [~CMultiwellOptoStimFunctionNet](#) ( )
- [!CMultiwellOptoStimFunctionNet](#) ( )
- [uint32\\_t](#) [GetWaveLengthInNanometer](#) ([uint16\\_t](#) channel)
- [uint32\\_t](#) [GetAbsMaxCurrentInMicroAmp](#) ([uint16\\_t](#) channel)
- [uint32\\_t](#) [GetMaxDurationHighCurrentInMicroSec](#) ([uint16\\_t](#) channel)
- [uint32\\_t](#) [GetMaxDutyCycleHighCurrent](#) ([uint16\\_t](#) channel)
- [uint32\\_t](#) [GetPermanentCurrentInMicroAmp](#) ([uint16\\_t](#) channel)
- [uint32\\_t](#) [GetColorRgb](#) ([uint16\\_t](#) channel)
- [String](#)<sup>^</sup> [GetColorStr](#) ([uint16\\_t](#) channel)
- void [SetWaveLengthInNanometer](#) ([uint16\\_t](#) channel, [uint32\\_t](#) WaveLength\_nm)
- void [SetAbsMaxCurrentInMicroAmp](#) ([uint16\\_t](#) channel, [uint32\\_t](#) AbsoluteMaxCurrent\_uA)
- void [SetMaxDurationHighCurrentInMicroSec](#) ([uint16\\_t](#) channel, [uint32\\_t](#) AbsoluteMaxDuration\_us)
- void [SetMaxDutyCycleHighCurrent](#) ([uint16\\_t](#) channel, [uint32\\_t](#) MaxDutyCycleHighCurrent)
- void [SetPermanentCurrentInMicroAmp](#) ([uint16\\_t](#) channel, [uint32\\_t](#) PermanentCurrent\_uA)
- void [SetColorRgb](#) ([uint16\\_t](#) channel, [uint32\\_t](#) ColorRGB)
- void [SetColorStr](#) ([uint16\\_t](#) channel, [String](#)<sup>^</sup> ColorString)

### Additional Inherited Members

#### 11.71.1 Detailed Description

[CMultiwellOptoStimFunctionNet](#) is the class to access the optical properties of the Multiwell Optostim device

#### 11.71.2 Constructor & Destructor Documentation



**11.71.2.1 CMultiwellOptoStimFunctionNet()** [1/2] `CMultiwellOptoStimFunctionNet ( CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ pMultiwellOptoStimFunctionPointerContainer )`

Initializes a new instance of the `CMultiwellOptoStimFunctionNet` class.

**11.71.2.2 CMultiwellOptoStimFunctionNet()** [2/2] `CMultiwellOptoStimFunctionNet ( CMcsUsbNet^ mcsusb )`

**11.71.2.3 ~CMultiwellOptoStimFunctionNet()** `virtual ~CMultiwellOptoStimFunctionNet ( )` [virtual]

**11.71.2.4 !CMultiwellOptoStimFunctionNet()** `!CMultiwellOptoStimFunctionNet ( )`

### 11.71.3 Member Function Documentation

**11.71.3.1 GetAbsMaxCurrentInMicroAmp()** `uint32_t GetAbsMaxCurrentInMicroAmp ( uint16_t channel )`

#### Parameters

<i>channel</i>	the (analog) channel number
----------------	-----------------------------

#### Returns

absolute max. current; unit: uA

**11.71.3.2 GetColorRgb()** `uint32_t GetColorRgb ( uint16_t channel )`

#### Parameters

<i>channel</i>	the (analog) channel number
----------------	-----------------------------

#### Returns

RGB-value of LED color

**11.71.3.3 GetColorStr()** `String ^ GetColorStr (`  
`uint16_t channel )`

**Parameters**

<i>channel</i>	the (analog) channel number
----------------	-----------------------------

**Returns**

LED color as string

**11.71.3.4 GetMaxDurationHighCurrentInMicroSec()** `uint32_t GetMaxDurationHighCurrentInMicroSec (`  
`uint16_t channel )`

**Parameters**

<i>channel</i>	the (analog) channel number
----------------	-----------------------------

**Returns**

max. duration the LED can stand the abs. max current; unit: us

**11.71.3.5 GetMaxDutyCycleHighCurrent()** `uint32_t GetMaxDutyCycleHighCurrent (`  
`uint16_t channel )`

**Parameters**

<i>channel</i>	the (analog) channel number
----------------	-----------------------------

**Returns**

max. duty cycle at max. current; unit: 100\*%

**11.71.3.6 GetPermanentCurrentInMicroAmp()** `uint32_t GetPermanentCurrentInMicroAmp (`  
`uint16_t channel )`

**Parameters**

<i>channel</i>	the (analog) channel number
----------------	-----------------------------

**Returns**

max. current the LED can stand when always switched on; unit: uA

**11.71.3.7 GetWaveLengthInNanometer()** `uint32_t GetWaveLengthInNanometer ( uint16_t channel )`

**Parameters**

<i>channel</i>	the (analog) channel number
----------------	-----------------------------

**Returns**

wavelength of this channel's LEDs; unit: nm

**11.71.3.8 SetAbsMaxCurrentInMicroAmp()** `void SetAbsMaxCurrentInMicroAmp ( uint16_t channel, uint32_t AbsoluteMaxCurrent_uA )`

**Parameters**

<i>channel</i>	the (analog) channel number
<i>AbsoluteMaxCurrent_uA</i>	absolute max. current; unit: uA

**11.71.3.9 SetColorRgb()** `void SetColorRgb ( uint16_t channel, uint32_t ColorRGB )`

**Parameters**

<i>channel</i>	the (analog) channel number
<i>ColorRGB</i>	RGB-value of LED color

**11.71.3.10 SetColorStr()** `void SetColorStr ( uint16_t channel, String^ ColorString )`

**Parameters**

<i>channel</i>	the (analog) channel number
<i>ColorString</i>	LED color as string

**11.71.3.11 SetMaxDurationHighCurrentInMicroSec()** void SetMaxDurationHighCurrentInMicroSec (   
     uint16\_t *channel*,  
     uint32\_t *AbsoluteMaxDuration\_us* )

**Parameters**

<i>channel</i>	the (analog) channel number
<i>AbsoluteMaxDuration_us</i>	max. duration the LED can stand the abs. max current; unit: us

**11.71.3.12 SetMaxDutyCycleHighCurrent()** void SetMaxDutyCycleHighCurrent (   
     uint16\_t *channel*,  
     uint32\_t *MaxDutyCycleHighCurrent* )

**Parameters**

<i>channel</i>	the (analog) channel number
<i>MaxDutyCycleHighCurrent</i>	max. duty cycle at max. current; unit: 100*%

**11.71.3.13 SetPermanentCurrentInMicroAmp()** void SetPermanentCurrentInMicroAmp (   
     uint16\_t *channel*,  
     uint32\_t *PermanentCurrent\_uA* )

**Parameters**

<i>channel</i>	the (analog) channel number
<i>PermanentCurrent_uA</i>	max. current the LED can stand when always switched on; unit: uA

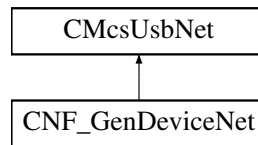
**11.71.3.14 SetWaveLengthInNanometer()** void SetWaveLengthInNanometer (   
     uint16\_t *channel*,  
     uint32\_t *WaveLength\_nm* )

**Parameters**

<i>channel</i>	the (analog) channel number
<i>WaveLength_nm</i>	wavelength of this channel's LEDs; unit: nm

## 11.72 CNF\_GenDeviceNet Class Reference

Inheritance diagram for CNF\_GenDeviceNet:



### Public Member Functions

- [CNF\\_GenDeviceNet](#) (void)
- [~CNF\\_GenDeviceNet](#) (void)
- void [Set\\_Values](#) (unsigned int frequency, unsigned int amplitude)

### Additional Inherited Members

#### 11.72.1 Constructor & Destructor Documentation

**11.72.1.1** [CNF\\_GenDeviceNet\(\)](#) [CNF\\_GenDeviceNet](#) (   
 void )

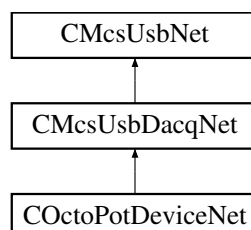
**11.72.1.2** [~CNF\\_GenDeviceNet\(\)](#) [~CNF\\_GenDeviceNet](#) (   
 void )

#### 11.72.2 Member Function Documentation

**11.72.2.1** [Set\\_Values\(\)](#) void Set\_Values (   
 unsigned int *frequency*,   
 unsigned int *amplitude* )

## 11.73 COctoPotDeviceNet Class Reference

Inheritance diagram for COctoPotDeviceNet:



**Public Member Functions**

- [COctoPotDeviceNet](#) (void)
- [COctoPotDeviceNet](#) ([OnChannelData](#)^ channelData, [OnError](#)^ error)
- uint32\_t [SetOutputRate](#) (uint32\_t rate)
- uint32\_t [SetBathclamp](#) (unsigned int block, bool enable)
- uint32\_t [SetDacValue](#) (int channel, int value)
- uint32\_t [SetDacAutoControl](#) (unsigned int channel)
- uint32\_t [SetPidParameter](#) (unsigned int channel, int const\_p, int const\_i, int shift\_p, int shift\_i)
- uint32\_t [SetRampParameter](#) (unsigned int channel, int start, int min, int max, int slope, int slope2, int pause, unsigned int samples)
- uint32\_t [RampStart](#) (int channelmap)
- uint32\_t [SetSineParameter](#) (unsigned int channel, int amplitude)
- uint32\_t [SineStart](#) (int channelmap)
- uint32\_t [SetPatternListEntry](#) (unsigned int channel, unsigned int position, unsigned int duration, int value)
- uint32\_t [PatternListStart](#) (int channelmap)
- uint32\_t [SetAdcOffset](#) (unsigned int channel, int offset)
- uint32\_t [SetDacOffset](#) (unsigned int channel, int offset)
- uint32\_t [ResetAdcOffset](#) (unsigned int channel)
- uint32\_t [ResetDacOffset](#) (unsigned int channel)
- uint32\_t [BurnAdcOffset](#) ()
- uint32\_t [BurnDacOffset](#) ()
- uint32\_t [GetAdcOffset](#) (unsigned int channel, [System::Runtime::InteropServices::Out] int ^ offset)
- uint32\_t [GetDacOffset](#) (unsigned int channel, [System::Runtime::InteropServices::Out] int ^ offset)
- uint32\_t [SetAmplificationSwitch](#) (unsigned int channel, unsigned int state)
- uint32\_t [SetChannelSwitch](#) (unsigned int channel, unsigned int state)
- uint32\_t [SetNumberOfChannels](#) (unsigned int NumberOfChannels)
- uint32\_t [EnableDigitalIn](#) (bool enable)
- uint32\_t [EnableTimestamp](#) (bool enable)
- uint32\_t [EnableChecksum](#) (bool enable)

**Additional Inherited Members****11.73.1 Constructor & Destructor Documentation**

**11.73.1.1 COctoPotDeviceNet()** [1/2] [COctoPotDeviceNet](#) ( void )

**11.73.1.2 COctoPotDeviceNet()** [2/2] [COctoPotDeviceNet](#) ( [OnChannelData](#)^ channelData, [OnError](#)^ error )

**11.73.2 Member Function Documentation**

**11.73.2.1 BurnAdcOffset()** uint32\_t BurnAdcOffset ( )

**11.73.2.2 BurnDacOffset()** uint32\_t BurnDacOffset ( )

**11.73.2.3 EnableChecksum()** uint32\_t EnableChecksum (   
bool *enable* )

**11.73.2.4 EnableDigitalIn()** uint32\_t EnableDigitalIn (   
bool *enable* )

**11.73.2.5 EnableTimestamp()** uint32\_t EnableTimestamp (   
bool *enable* )

**11.73.2.6 GetAdcOffset()** uint32\_t GetAdcOffset (   
unsigned int *channel*,   
[System::Runtime::InteropServices::Out] int ^ *offset* )

**11.73.2.7 GetDacOffset()** uint32\_t GetDacOffset (   
unsigned int *channel*,   
[System::Runtime::InteropServices::Out] int ^ *offset* )

**11.73.2.8 PatternListStart()** uint32\_t PatternListStart (   
int *channelmap* )

**11.73.2.9 RampStart()** uint32\_t RampStart (   
int *channelmap* )

**11.73.2.10 ResetAdcOffset()** uint32\_t ResetAdcOffset (   
unsigned int *channel* )

**11.73.2.11 ResetDacOffset()** uint32\_t ResetDacOffset (   
 unsigned int *channel* )

**11.73.2.12 SetAdcOffset()** uint32\_t SetAdcOffset (   
 unsigned int *channel*,   
 int *offset* )

**11.73.2.13 SetAmplificationSwitch()** uint32\_t SetAmplificationSwitch (   
 unsigned int *channel*,   
 unsigned int *state* )

**11.73.2.14 SetBathclamp()** uint32\_t SetBathclamp (   
 unsigned int *block*,   
 bool *enable* )

**11.73.2.15 SetChannelSwitch()** uint32\_t SetChannelSwitch (   
 unsigned int *channel*,   
 unsigned int *state* )

**11.73.2.16 SetDacAutoControl()** uint32\_t SetDacAutoControl (   
 unsigned int *channel* )

**11.73.2.17 SetDacOffset()** uint32\_t SetDacOffset (   
 unsigned int *channel*,   
 int *offset* )

**11.73.2.18 SetDacValue()** uint32\_t SetDacValue (   
 int *channel*,   
 int *value* )

**11.73.2.19 SetNumberOfChannels()** uint32\_t SetNumberOfChannels (   
 unsigned int *NumberOfChannels* )



**11.73.2.20 SetOutputRate()** uint32\_t SetOutputRate (   
 uint32\_t *rate* )

**11.73.2.21 SetPatternListEntry()** uint32\_t SetPatternListEntry (   
 unsigned int *channel*,   
 unsigned int *position*,   
 unsigned int *duration*,   
 int *value* )

**11.73.2.22 SetPidParameter()** uint32\_t SetPidParameter (   
 unsigned int *channel*,   
 int *const\_p*,   
 int *const\_i*,   
 int *shift\_p*,   
 int *shift\_i* )

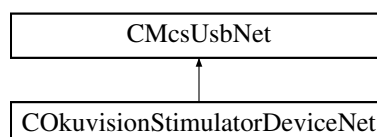
**11.73.2.23 SetRampParameter()** uint32\_t SetRampParameter (   
 unsigned int *channel*,   
 int *start*,   
 int *min*,   
 int *max*,   
 int *slope*,   
 int *slope2*,   
 int *pause*,   
 unsigned int *samples* )

**11.73.2.24 SetSineParameter()** uint32\_t SetSineParameter (   
 unsigned int *channel*,   
 int *amplitude* )

**11.73.2.25 SineStart()** uint32\_t SineStart (   
 int *channelmap* )

## 11.74 COkuvisionStimulatorDeviceNet Class Reference

Inheritance diagram for COkuvisionStimulatorDeviceNet:



**Public Member Functions**

- [COkuvisionStimulatorDeviceNet](#) (void)
- [~COkuvisionStimulatorDeviceNet](#) (void)
- void [SetPulseform](#) (int channel, int current, int pulsewidth, int periode, int duration)
- void [GetPulseform](#) (int channel, [System::Runtime::InteropServices::Out] int% current, [System::Runtime::InteropServices::Out] int% pulsewidth, [System::Runtime::InteropServices::Out] int% periode, [System::Runtime::InteropServices::Out] int% duration)
- void [SetMaxPower](#) (int channel, int power)
- int [GetMaxPower](#) (int channel)
- void [SetMaxVoltage](#) (int channel, int voltage)
- int [GetMaxVoltage](#) (int channel)
- void [SetCheckVoltage](#) (int channel, int voltage)
- int [GetCheckVoltage](#) (int channel)
- int [GetVoltage](#) (int channel)
- void [SetDACOffset](#) (int channel, int part, int offset)
- int [GetDACOffset](#) (int channel, int part)
- void [SetRTC](#) (uint8\_t year, uint8\_t month, uint8\_t day, uint8\_t hour, uint8\_t minute, uint8\_t second)
- void [GetRTC](#) ([System::Runtime::InteropServices::Out] uint8\_t% year, [System::Runtime::InteropServices::Out] uint8\_t% month, [System::Runtime::InteropServices::Out] uint8\_t% day, [System::Runtime::InteropServices::Out] uint8\_t% hour, [System::Runtime::InteropServices::Out] uint8\_t% minute, [System::Runtime::InteropServices::Out] uint8\_t% second)
- void [SetRTC](#) (DateTime timestamp)
- DateTime [GetRTC](#) ()
- void [GetStimulatorStatus](#) ([System::Runtime::InteropServices::Out] int% startstop, [System::Runtime::InteropServices::Out] int% last\_error, [System::Runtime::InteropServices::Out] int% battery\_status)
- void [SetCurrentFactor](#) (int channel, int factor)
- int [GetCurrentFactor](#) (int channel)

**Additional Inherited Members****11.74.1 Constructor & Destructor Documentation**

**11.74.1.1 COkuvisionStimulatorDeviceNet()** [COkuvisionStimulatorDeviceNet](#) (   
 void )

**11.74.1.2 ~COkuvisionStimulatorDeviceNet()** [~COkuvisionStimulatorDeviceNet](#) (   
 void )

**11.74.2 Member Function Documentation**

**11.74.2.1 GetCheckVoltage()** int GetCheckVoltage (   
 int channel )

**11.74.2.2 GetCurrentFactor()** `int GetCurrentFactor (`  
`int channel )`

**11.74.2.3 GetDACOffset()** `int GetDACOffset (`  
`int channel,`  
`int part )`

**11.74.2.4 GetMaxPower()** `int GetMaxPower (`  
`int channel )`

**11.74.2.5 GetMaxVoltage()** `int GetMaxVoltage (`  
`int channel )`

**11.74.2.6 GetPulseform()** `void GetPulseform (`  
`int channel,`  
`[System::Runtime::InteropServices::Out] int% current,`  
`[System::Runtime::InteropServices::Out] int% pulsewidth,`  
`[System::Runtime::InteropServices::Out] int% periode,`  
`[System::Runtime::InteropServices::Out] int% duration )`

**11.74.2.7 GetRTC()** [1/2] `DateTime GetRTC ( )`

**11.74.2.8 GetRTC()** [2/2] `void GetRTC (`  
`[System::Runtime::InteropServices::Out] uint8_t% year,`  
`[System::Runtime::InteropServices::Out] uint8_t% month,`  
`[System::Runtime::InteropServices::Out] uint8_t% day,`  
`[System::Runtime::InteropServices::Out] uint8_t% hour,`  
`[System::Runtime::InteropServices::Out] uint8_t% minute,`  
`[System::Runtime::InteropServices::Out] uint8_t% second )`

**11.74.2.9 GetStimulatorStatus()** `void GetStimulatorStatus (`  
`[System::Runtime::InteropServices::Out] int% startstop,`  
`[System::Runtime::InteropServices::Out] int% last_error,`  
`[System::Runtime::InteropServices::Out] int% battery_status )`

**11.74.2.10 GetVoltage()** `int GetVoltage (`  
`int channel )`

**11.74.2.11 SetCheckVoltage()** `void SetCheckVoltage (`  
`int channel,`  
`int voltage )`

**11.74.2.12 SetCurrentFactor()** `void SetCurrentFactor (`  
`int channel,`  
`int factor )`

**11.74.2.13 SetDACOffset()** `void SetDACOffset (`  
`int channel,`  
`int part,`  
`int offset )`

**11.74.2.14 SetMaxPower()** `void SetMaxPower (`  
`int channel,`  
`int power )`

**11.74.2.15 SetMaxVoltage()** `void SetMaxVoltage (`  
`int channel,`  
`int voltage )`

**11.74.2.16 SetPulseform()** `void SetPulseform (`  
`int channel,`  
`int current,`  
`int pulsewidth,`  
`int periode,`  
`int duration )`

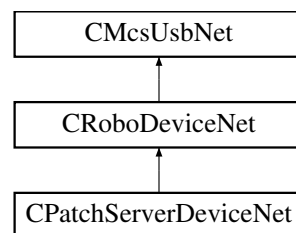
**11.74.2.17 SetRTC()** `[1/2] void SetRTC (`  
`DateTime timestamp )`

**11.74.2.18 SetRTC()** [2/2] `void SetRTC (`  
    `uint8_t year,`  
    `uint8_t month,`  
    `uint8_t day,`  
    `uint8_t hour,`  
    `uint8_t minute,`  
    `uint8_t second )`

## 11.75 CPatchServerDeviceNet Class Reference

[CPatchServerDeviceNet](#) is the class to control the MCS PatchServer device

Inheritance diagram for CPatchServerDeviceNet:



### Public Member Functions

- [CPatchServerDeviceNet](#) (void)

### Properties

- [CMcsBus\\_SensorNet](#)<sup>^</sup> [Sensor](#) [get]

### Additional Inherited Members

#### 11.75.1 Detailed Description

[CPatchServerDeviceNet](#) is the class to control the MCS PatchServer device

#### 11.75.2 Constructor & Destructor Documentation

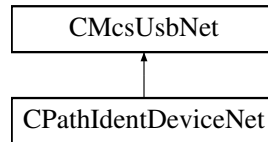
**11.75.2.1 CPatchServerDeviceNet()** [CPatchServerDeviceNet](#) (  
    void )

#### 11.75.3 Property Documentation

### 11.75.3.1 Sensor `CMcsBus_SensorNet`^ Sensor [get]

## 11.76 CPathIdentDeviceNet Class Reference

Inheritance diagram for CPathIdentDeviceNet:



### Public Member Functions

- `CPathIdentDeviceNet` (void)
- `~CPathIdentDeviceNet` (void)
- void `Set_Values` (unsigned int frequency, unsigned int amplitude)
- void `Measure` ([System::Runtime::InteropServices::Out] unsigned int% phase, [System::Runtime::InteropServices::Out] unsigned int% amplitude)

### Additional Inherited Members

#### 11.76.1 Constructor & Destructor Documentation

**11.76.1.1 `CPathIdentDeviceNet()`** `CPathIdentDeviceNet` ( void )

**11.76.1.2 `~CPathIdentDeviceNet()`** `~CPathIdentDeviceNet` ( void )

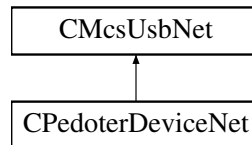
#### 11.76.2 Member Function Documentation

**11.76.2.1 `Measure()`** void `Measure` ( [System::Runtime::InteropServices::Out] unsigned int% *phase*, [System::Runtime::InteropServices::Out] unsigned int% *amplitude* )

**11.76.2.2 Set\_Values()** `void Set_Values (`  
     `unsigned int frequency,`  
     `unsigned int amplitude )`

## 11.77 CPedoterDeviceNet Class Reference

Inheritance diagram for CPedoterDeviceNet:



### Public Member Functions

- [CPedoterDeviceNet](#) ()  
*Initializes a new instance of the [CPedoterDeviceNet](#) class.*
- virtual [~CPedoterDeviceNet](#) ()
- [!CPedoterDeviceNet](#) ()
- uint32\_t [GetCommand](#) (uint16\_t Argument)  
*Get value from the pedoter device*
- void [SetCommand](#) (uint16\_t Argument, uint32\_t pData)  
*Set value on the pedoter device*

### Additional Inherited Members

#### 11.77.1 Detailed Description

#### 11.77.2 Constructor & Destructor Documentation

**11.77.2.1 CPedoterDeviceNet()** [CPedoterDeviceNet](#) ( )

Initializes a new instance of the [CPedoterDeviceNet](#) class.

**11.77.2.2 ~CPedoterDeviceNet()** `virtual ~CPedoterDeviceNet ( )` [virtual]

**11.77.2.3 "!CPedoterDeviceNet()** [!CPedoterDeviceNet](#) ( )

#### 11.77.3 Member Function Documentation

**11.77.3.1 GetCommand()** `uint32_t GetCommand (`  
     `uint16_t Argument )`

Get value from the pedoter device

## Parameters

<i>Argument</i>	argument
-----------------	----------

## Returns

value

**11.77.3.2 SetCommand()** void SetCommand (   
     uint16\_t *Argument*,   
     uint32\_t *pData* )

Set value on the pedoter device

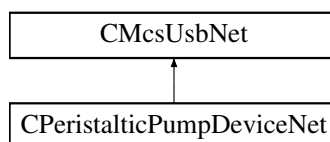
## Parameters

<i>Argument</i>	argument
<i>pData</i>	value

## 11.78 CPeristalticPumpDeviceNet Class Reference

[CPeristalticPumpDeviceNet](#) is the class to control a Persistaltic Pump.

Inheritance diagram for CPeristalticPumpDeviceNet:



## Public Member Functions

- [CPeristalticPumpDeviceNet](#) (void)  
     Initialize a new instance of the [CPeristalticPumpDeviceNet](#) class.
- [~CPeristalticPumpDeviceNet](#) (void)

## Properties

- [CMcsBus\\_MotorControlNet](#)^ [McsBus\\_MotorControl](#) [get]

## Additional Inherited Members

## 11.78.1 Detailed Description

[CPeristalticPumpDeviceNet](#) is the class to control a Persistaltic Pump.



## 11.78.2 Constructor & Destructor Documentation

**11.78.2.1 CPeristalticPumpDeviceNet()** `CPeristalticPumpDeviceNet (void )`

Initialize a new instance of the `CPeristalticPumpDeviceNet` class.

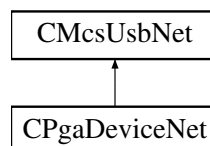
**11.78.2.2 ~CPeristalticPumpDeviceNet()** `~CPeristalticPumpDeviceNet (void )`

## 11.78.3 Property Documentation

**11.78.3.1 McsBus\_MotorControl** `CMcsBus_MotorControlNet^ McsBus_MotorControl [get]`

## 11.79 CPgaDeviceNet Class Reference

Inheritance diagram for CPgaDeviceNet:



### Public Member Functions

- `CPgaDeviceNet ()`
- `~CPgaDeviceNet ()`
- `uint32_t GetNumFrequencyRanges ([System::Runtime::InteropServices::Out]int% numRanges)`
- `uint32_t GetFrequencyRange (int rangeIndex, [System::Runtime::InteropServices::Out]int% low, [System::Runtime::InteropServices::Out]int% high, [System::Runtime::InteropServices::Out]int% channels, [System::Runtime::InteropServices::Out]int% gain)`
- `uint32_t GetNumAmplifications ([System::Runtime::InteropServices::Out]int% number)`
- `uint32_t GetAmplification (int index, [System::Runtime::InteropServices::Out]int% amplification, [System::Runtime::InteropServices::Out]int% poti1, [System::Runtime::InteropServices::Out]int% poti2)`
- `uint32_t DefineNumFrequencyRanges (int rnum)`
- `uint32_t DefineFrequencyRange (int index, int low, int high, int channels, int gain)`
- `uint32_t DefineNumAmplifications (int number)`
- `uint32_t DefineAmplification (int index, int amplification, int poti1, int poti2)`
- `uint32_t SetGain (int channel, int Gain, int poti1, int poti2)`
- `uint32_t GetGain (int channel, [System::Runtime::InteropServices::Out]int% Gain, [System::Runtime::InteropServices::Out]int% poti1, [System::Runtime::InteropServices::Out]int% poti2)`
- `uint32_t ApplyGains ()`

## Additional Inherited Members

### 11.79.1 Constructor & Destructor Documentation

**11.79.1.1 CPgaDeviceNet()** `CPgaDeviceNet ( )`

**11.79.1.2 ~CPgaDeviceNet()** `~CPgaDeviceNet ( )`

### 11.79.2 Member Function Documentation

**11.79.2.1 ApplyGains()** `uint32_t ApplyGains ( )`

**11.79.2.2 DefineAmplification()** `uint32_t DefineAmplification (`  
    `int index,`  
    `int amplification,`  
    `int poti1,`  
    `int poti2 )`

**11.79.2.3 DefineFrequencyRange()** `uint32_t DefineFrequencyRange (`  
    `int index,`  
    `int low,`  
    `int high,`  
    `int channels,`  
    `int gain )`

**11.79.2.4 DefineNumAmplifications()** `uint32_t DefineNumAmplifications (`  
    `int number )`

**11.79.2.5 DefineNumFrequencyRanges()** `uint32_t DefineNumFrequencyRanges (`  
    `int rnum )`

**11.79.2.6 GetAmplification()** uint32\_t GetAmplification (   
     int *index*,   
     [System::Runtime::InteropServices::Out] int% *amplification*,   
     [System::Runtime::InteropServices::Out] int% *potil*,   
     [System::Runtime::InteropServices::Out] int% *poti2* )

**11.79.2.7 GetFrequencyRange()** uint32\_t GetFrequencyRange (   
     int *rangeIndex*,   
     [System::Runtime::InteropServices::Out] int% *low*,   
     [System::Runtime::InteropServices::Out] int% *high*,   
     [System::Runtime::InteropServices::Out] int% *channels*,   
     [System::Runtime::InteropServices::Out] int% *gain* )

**11.79.2.8 GetGain()** uint32\_t GetGain (   
     int *channel*,   
     [System::Runtime::InteropServices::Out] int% *Gain*,   
     [System::Runtime::InteropServices::Out] int% *potil*,   
     [System::Runtime::InteropServices::Out] int% *poti2* )

**11.79.2.9 GetNumAmplifications()** uint32\_t GetNumAmplifications (   
     [System::Runtime::InteropServices::Out] int% *number* )

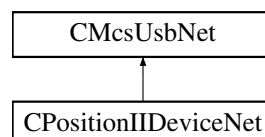
**11.79.2.10 GetNumFrequencyRanges()** uint32\_t GetNumFrequencyRanges (   
     [System::Runtime::InteropServices::Out] int% *numRanges* )

**11.79.2.11 SetGain()** uint32\_t SetGain (   
     int *channel*,   
     int *Gain*,   
     int *potil*,   
     int *poti2* )

## 11.80 CPositionIIDeviceNet Class Reference

[CPositionIIDeviceNet](#) is the class to control PositionII devices

Inheritance diagram for CPositionIIDeviceNet:



**Public Member Functions**

- [CPositionIIDeviceNet](#) ()  
*Initializes a new instance of the [CPositionIIDeviceNet](#) class.*
- virtual [~CPositionIIDeviceNet](#) ()
- [!CPositionIIDeviceNet](#) ()
- uint32\_t [GetCoilCommunication](#) (uint16\_t coil)  
*get if the communication to the coil is working*
- uint32\_t [GetOnOff](#) (uint16\_t coil)  
*get if the coil is switched on/off*
- void [SwitchOnOff](#) (uint16\_t coil, uint32\_t on)  
*switched the coil on of*
- uint32\_t [GetImplantState](#) (uint16\_t coil)  
*gets the implantat state*
- uint32\_t [GetImplantCurrentSetpoint](#) (uint16\_t coil)  
*sets the implant current setpoint*
- void [SetImplantCurrentSetpoint](#) (uint16\_t coil, uint32\_t current)  
*gets the implant current setpoint*

**Properties**

- [CRFFunctionNet](#)<sup>^</sup> [RFFunction](#) [get]

**Additional Inherited Members****11.80.1 Detailed Description**

[CPositionIIDeviceNet](#) is the class to control PositionII devices

**11.80.2 Constructor & Destructor Documentation****11.80.2.1 CPositionIIDeviceNet()** [CPositionIIDeviceNet](#) ( )

Initializes a new instance of the [CPositionIIDeviceNet](#) class.

**11.80.2.2 ~CPositionIIDeviceNet()** virtual [~CPositionIIDeviceNet](#) ( ) [virtual]**11.80.2.3 !CPositionIIDeviceNet()** [!CPositionIIDeviceNet](#) ( )**11.80.3 Member Function Documentation****11.80.3.1 GetCoilCommunication()** uint32\_t [GetCoilCommunication](#) (uint16\_t coil )

get if the communication to the coil is working

## Parameters

<i>coil</i>	the coil
-------------	----------

## Returns

is communicating

**11.80.3.2 GetImplantCurrentSetpoint()** `uint32_t GetImplantCurrentSetpoint (uint16_t coil )`

sets the implant current setpoint

## Parameters

<i>coil</i>	the coil
-------------	----------

## Returns

the current

**11.80.3.3 GetImplantState()** `uint32_t GetImplantState (uint16_t coil )`

gets the implantat state

## Parameters

<i>coil</i>	the coil
-------------	----------

## Returns

**11.80.3.4 GetOnOff()** `uint32_t GetOnOff (uint16_t coil )`

get if the coil is switched on/off

## Parameters

<i>coil</i>	the coil
-------------	----------

## Returns

0 = off, 1 = on

**11.80.3.5 SetImplantCurrentSetpoint()** `void SetImplantCurrentSetpoint (`  
     `uint16_t coil,`  
     `uint32_t current )`

gets the implant current setpoint

## Parameters

<i>coil</i>	the coil
<i>current</i>	the current

**11.80.3.6 SwitchOnOff()** `void SwitchOnOff (`  
     `uint16_t coil,`  
     `uint32_t on )`

switched the coild on of

## Parameters

<i>coil</i>	the coil
<i>on</i>	0 = off, 1 = on

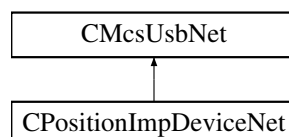
## 11.80.4 Property Documentation

**11.80.4.1 RFFunction** `CRFFunctionNet^ RFFunction [get]`

## 11.81 CPositionImpDeviceNet Class Reference

[CPositionImpDeviceNet](#) is the class to access the Position/Imp devices

Inheritance diagram for CPositionImpDeviceNet:



**Public Member Functions**

- [CPositionImpDeviceNet](#) ()  
*Initializes a new instance of the [CPositionImpDeviceNet](#) class.*
- virtual [~CPositionImpDeviceNet](#) ()
- [!CPositionImpDeviceNet](#) ()
- void [ConnectImp](#) (uint32\_t id)  
*Connect to a Imp device with a certain ID*
- uint32\_t [ConnectedImp](#) ()  
*The ID of the connected Imp device*
- int32\_t [GetRFFrequency](#) ()  
*Gets currently used RF frequency*
- void [SetRFFrequency](#) (int32\_t frequency)  
*Sets the current RF frequency*
- uint32\_t [GetDeviceList](#) (int32\_t index)  
*Gets the device list*
- void [SetDeviceList](#) (int32\_t index, uint32\_t id)  
*Sets the device list*
- uint32\_t [GetImpId](#) ()  
*Gets the ID of the impedance measure device*
- void [SetImpId](#) (uint32\_t id)  
*Sets the ID of the impedance measure device*

**Additional Inherited Members****11.81.1 Detailed Description**

[CPositionImpDeviceNet](#) is the class to access the Position/Imp devices

**11.81.2 Constructor & Destructor Documentation****11.81.2.1 CPositionImpDeviceNet() [CPositionImpDeviceNet](#) ( )**

Initializes a new instance of the [CPositionImpDeviceNet](#) class.

**11.81.2.2 ~CPositionImpDeviceNet() virtual [~CPositionImpDeviceNet](#) ( ) [virtual]****11.81.2.3 "!CPositionImpDeviceNet() [!CPositionImpDeviceNet](#) ( )**

### 11.81.3 Member Function Documentation

#### 11.81.3.1 ConnectedImp() `uint32_t ConnectedImp ( )`

The ID of the connected Imp device

##### Returns

The ID

#### 11.81.3.2 ConnectImp() `void ConnectImp ( uint32_t id )`

Connect to a Imp device with a certain ID

##### Parameters

<i>id</i>	The ID
-----------	--------

#### 11.81.3.3 GetDeviceList() `uint32_t GetDeviceList ( int32_t index )`

Gets the device list

##### Parameters

<i>index</i>	the index
--------------	-----------

##### Returns

the ID

#### 11.81.3.4 GetImpId() `uint32_t GetImpId ( )`

Gets the ID of the impedance measure device

##### Returns

the ID



**11.81.3.5 GetRFFrequency()** `int32_t GetRFFrequency ( )`

Gets currently used RF frequency

**Returns**

The frequency

**11.81.3.6 SetDeviceList()** `void SetDeviceList (   
int32_t index,   
uint32_t id )`

Sets the device list

**Parameters**

<i>index</i>	the index
<i>id</i>	the ID

**11.81.3.7 SetImpId()** `void SetImpId (   
uint32_t id )`

Sets the ID of the impedance measure device

**Parameters**

<i>id</i>	the ID
-----------	--------

**11.81.3.8 SetRFFrequency()** `void SetRFFrequency (   
int32_t frequency )`

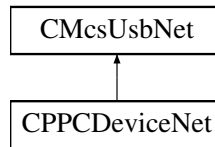
Sets the current RF frequency

**Parameters**

<i>frequency</i>	The frequency
------------------	---------------

**11.82 CPPCDeviceNet Class Reference**

Inheritance diagram for CPPCDeviceNet:



### Public Member Functions

- [CPPCDeviceNet](#) (void)

### Properties

- [CPPCFunctionNet](#)<sup>^</sup> [PPCFunction](#) [get]
- [CMcsBusNet](#)<sup>^</sup> [McsBus](#) [get]
- [CMcsBus\\_MotorControlNet](#)<sup>^</sup> [McsBus\\_MotorControl](#) [get]
- [CMcsBus\\_SensorNet](#)<sup>^</sup> [McsBus\\_Sensor](#) [get]

### Additional Inherited Members

#### 11.82.1 Constructor & Destructor Documentation

**11.82.1.1** [CPPCDeviceNet](#)() [CPPCDeviceNet](#) (  
void )

#### 11.82.2 Property Documentation

**11.82.2.1** [McsBus](#) [CMcsBusNet](#)<sup>^</sup> [McsBus](#) [get]

**11.82.2.2** [McsBus\\_MotorControl](#) [CMcsBus\\_MotorControlNet](#)<sup>^</sup> [McsBus\\_MotorControl](#) [get]

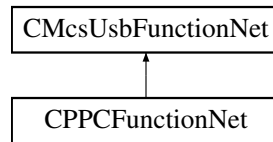
**11.82.2.3** [McsBus\\_Sensor](#) [CMcsBus\\_SensorNet](#)<sup>^</sup> [McsBus\\_Sensor](#) [get]

**11.82.2.4** [PPCFunction](#) [CPPCFunctionNet](#)<sup>^</sup> [PPCFunction](#) [get]

## 11.83 CPPCFunctionNet Class Reference

[CPPCFunctionNet](#) is the class to access the PPC (high precision Patch Peristaltic patch Pump

Inheritance diagram for CPPCFunctionNet:



### Public Member Functions

- [CPPCFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> pPPCFunctionPointer↔ Container)  
*Initializes a new instance of the [CPPCFunctionNet](#) class.*
- [CPPCFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- virtual [~CPPCFunctionNet](#) ()
- [!CPPCFunctionNet](#) ()
- int [GetPumpSpeedUnit](#) (uint16\_t channel)  
*Reads the Pump Speed Unit*
- void [SetPumpSpeedUnit](#) (uint16\_t channel, int SpeedUnit)  
*Writes the Pump Speed Unit*
- PP\_Pump\_Mode\_Type\_EnumNet [GetPumpModeType](#) (uint16\_t channel)  
*Reads the Pump Mode Type.*
- void [SetPumpModeType](#) (uint16\_t channel, PP\_Pump\_Mode\_Type\_EnumNet PumpMode)  
*Writes the config string from the device.*
- void [GetAnalogVoltageRange](#) (uint16\_t channel, [System::Runtime::InteropServices::Out]uint16\_t% min\_↔ voltage, [System::Runtime::InteropServices::Out]uint16\_t% max\_voltage)  
*Reads the Analog Input Voltage Range*
- void [SetAnalogVoltageRange](#) (uint16\_t channel, uint16\_t min\_voltage, uint16\_t max\_voltage)  
*Writes the Analog Input Voltage Range*
- void [GetPressureRange](#) (uint16\_t channel, [System::Runtime::InteropServices::Out]int32\_t% lower\_↔ pressure, [System::Runtime::InteropServices::Out]int32\_t% upper\_pressure)  
*Get the pressure range that is used between the analog voltage or the digital states*
- void [SetPressureRange](#) (uint16\_t channel, int32\_t lower\_pressure, int32\_t upper\_pressure)  
*Get the pressure range that is used between the analog voltage or the digital states*
- uint16\_t [GetSupplyVoltage](#) ()  
*Reads the current supply voltage in mV*
- uint16\_t [GetAnalogVoltage](#) (uint16\_t channel)  
*Reads the current analog voltage*
- uint16\_t [GetDigitalIn](#) (uint16\_t channel)  
*Reads the digital input state*
- int [GetValveActive](#) (uint16\_t valve)  
*Gets the valve active/inactive state*
- void [SetValveActive](#) (uint16\_t valve, int valveActive)  
*Sets the valve active/inactive state*
- void [SetPressureOffset](#) ()  
*Sets the pressure offset*
- void [LoadPressure](#) (int32\_t pressure, uint32\_t options)

- Loads the reservoir with a pressure*
- void [IsBusy](#) ([System::Runtime::InteropServices::Out]int16\_t% task, [System::Runtime::InteropServices::Out]int16\_t% wait)
- Is the PPC busy with a task*
- void [FirePressurePulse](#) (int32\_t duration, int32\_t nextpressure)
- Fire a pressure pulse from the reservoir*
- int32\_t [MeasureReservoir](#) ()
- Measures the reservoir pressure*

## Additional Inherited Members

### 11.83.1 Detailed Description

[CPPCFunctionNet](#) is the class to access the PPC (high precision Patch Peristaltic patch Pump

### 11.83.2 Constructor & Destructor Documentation

**11.83.2.1 CPPCFunctionNet()** [1/2] [CPPCFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb,  
[CMcsUsbFunctionPointerContainer](#)<sup>^</sup> pPPCFunctionPointerContainer )

Initializes a new instance of the [CPPCFunctionNet](#) class.

**11.83.2.2 CPPCFunctionNet()** [2/2] [CPPCFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb )

**11.83.2.3 ~CPPCFunctionNet()** virtual ~[CPPCFunctionNet](#) ( ) [virtual]

**11.83.2.4 "!CPPCFunctionNet()** ![CPPCFunctionNet](#) ( )

### 11.83.3 Member Function Documentation

**11.83.3.1 FirePressurePulse()** void [FirePressurePulse](#) (  
int32\_t duration,  
int32\_t nextpressure )

Fire a pressure pulse from the reservoir

## Parameters

<i>duration</i>	The pulse duration (valves open)
<i>nextpressure</i>	The next pressure

**11.83.3.2 GetAnalogVoltage()** `uint16_t GetAnalogVoltage (`  
`uint16_t channel )`

Reads the current analog voltage

## Parameters

<i>channel</i>	The Channel Number
----------------	--------------------

## Returns

The Analog Voltage

**11.83.3.3 GetAnalogVoltageRange()** `void GetAnalogVoltageRange (`  
`uint16_t channel,`  
`[System::Runtime::InteropServices::Out] uint16_t% min_voltage,`  
`[System::Runtime::InteropServices::Out] uint16_t% max_voltage )`

Reads the Analog Input Voltage Range

## Parameters

<i>channel</i>	The Channel Number
<i>min_voltage</i>	The voltage that should be seen as the minimum voltage
<i>max_voltage</i>	The voltage that should be seen as the maximum voltage

**11.83.3.4 GetDigitalIn()** `uint16_t GetDigitalIn (`  
`uint16_t channel )`

Reads the digital input state

## Parameters

<i>channel</i>	The Channel Number
----------------	--------------------

**Returns**

The Digital State

**11.83.3.5 GetPressureRange()** void GetPressureRange (   
     uint16\_t *channel*,   
     [System::Runtime::InteropServices::Out] int32\_t% *lower\_pressure*,   
     [System::Runtime::InteropServices::Out] int32\_t% *upper\_pressure* )

Get the pressure range that is used between the analog voltage or the digital states

**Parameters**

<i>channel</i>	The Channel Number
<i>lower_pressure</i>	The lower border of the pressure range
<i>upper_pressure</i>	The upper border of the pressure range

**11.83.3.6 GetPumpModeType()** PP\_Pump\_Mode\_Type\_EnumNet GetPumpModeType (   
     uint16\_t *channel* )

Reads the Pump Mode Type.

**Parameters**

<i>channel</i>	The Channel Number
----------------	--------------------

**Returns**

The Pump Mode Type.

**11.83.3.7 GetPumpSpeedUnit()** int GetPumpSpeedUnit (   
     uint16\_t *channel* )

Reads the Pump Speed Unit

**Parameters**

<i>channel</i>	The Channel Number
----------------	--------------------

**Returns**

The Speed Unit

**11.83.3.8 GetSupplyVoltage()** `uint16_t GetSupplyVoltage ( )`

Reads the current supply voltage in mV

**Returns**

The supply voltage

**11.83.3.9 GetValveActive()** `int GetValveActive (   
 uint16_t valve )`

Gets the valve active/inactive state

**Parameters**

<i>valve</i>	The valve number
--------------	------------------

**Returns**

The valve state

**11.83.3.10 IsBusy()** `void IsBusy (   
 [System::Runtime::InteropServices::Out] int16_t% task,   
 [System::Runtime::InteropServices::Out] int16_t% wait )`

Is the PPC busy with a task

**Parameters**

<i>task</i>	The task state
<i>wait</i>	The wait state

**11.83.3.11 LoadPressure()** `void LoadPressure (   
 int32_t pressure,   
 uint32_t options )`

Loads the reservoir with a pressure

**Parameters**

<i>pressure</i>	The pressure
<i>options</i>	The options: end with 0=regulate on patch 1=regulate on reservoir

**11.83.3.12 MeasureReservoir()** `int32_t MeasureReservoir ( )`

Measures the reservoir pressure

**Returns**

The pressure

**11.83.3.13 SetAnalogVoltageRange()** `void SetAnalogVoltageRange (`  
`uint16_t channel,`  
`uint16_t min_voltage,`  
`uint16_t max_voltage )`

Writes the Analog Input Voltage Range

**Parameters**

<i>channel</i>	The Channel Number
<i>min_voltage</i>	The voltage that should be seen as the minimum voltage
<i>max_voltage</i>	The voltage that should be seen as the maximum voltage

**11.83.3.14 SetPressureOffset()** `void SetPressureOffset ( )`

Sets the pressure offset

**11.83.3.15 SetPressureRange()** `void SetPressureRange (`  
`uint16_t channel,`  
`int32_t lower_pressure,`  
`int32_t upper_pressure )`

Get the pressure range that is used between the analog voltage or the digital states

**Parameters**

<i>channel</i>	The Channel Number
<i>lower_pressure</i>	The lower border of the pressure range
<i>upper_pressure</i>	The upper border of the pressure range



**11.83.3.16 SetPumpModeType()** `void SetPumpModeType (`  
     `uint16_t channel,`  
     `PP_Pump_Mode_Type_EnumNet PumpMode )`

Writes the config string from the device.

#### Parameters

<i>channel</i>	The Channel Number
<i>PumpMode</i>	The Pump Mode Type.

**11.83.3.17 SetPumpSpeedUnit()** `void SetPumpSpeedUnit (`  
     `uint16_t channel,`  
     `int SpeedUnit )`

Writes the Pump Speed Unit

#### Parameters

<i>channel</i>	The Channel Number
<i>SpeedUnit</i>	The Speed Unit

**11.83.3.18 SetValveActive()** `void SetValveActive (`  
     `uint16_t valve,`  
     `int valveActive )`

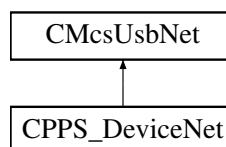
Sets the valve active/inactive state

#### Parameters

<i>valve</i>	The valve number
<i>valveActive</i>	The valve state

## 11.84 CPPS\_DeviceNet Class Reference

Inheritance diagram for CPPS\_DeviceNet:



#### Public Member Functions

- [CPPS\\_DeviceNet](#) (void)

## Properties

- [CPPS\\_FunctionNet](#)^ [PPS\\_Function](#) [get]
- [CMcsBusNet](#)^ [McsBus](#) [get]
- [CMcsBus\\_MotorControlNet](#)^ [McsBus\\_MotorControl](#) [get]
- [CMcsBus\\_SensorNet](#)^ [McsBus\\_Sensor](#) [get]

## Additional Inherited Members

### 11.84.1 Constructor & Destructor Documentation

**11.84.1.1** [CPPS\\_DeviceNet\(\)](#) [CPPS\\_DeviceNet](#) (  
void )

### 11.84.2 Property Documentation

**11.84.2.1** **McsBus** [CMcsBusNet](#)^ [McsBus](#) [get]

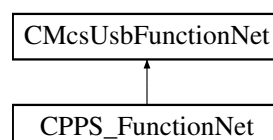
**11.84.2.2** **McsBus\_MotorControl** [CMcsBus\\_MotorControlNet](#)^ [McsBus\\_MotorControl](#) [get]

**11.84.2.3** **McsBus\_Sensor** [CMcsBus\\_SensorNet](#)^ [McsBus\\_Sensor](#) [get]

**11.84.2.4** **PPS\_Function** [CPPS\\_FunctionNet](#)^ [PPS\\_Function](#) [get]

## 11.85 CPPS\_FunctionNet Class Reference

Inheritance diagram for CPPS\_FunctionNet:



## Public Member Functions

- [CPPS\\_FunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> cPPS\_FunctionPointerContainer)
- [CPPS\\_FunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- void [SetPumpMaxSpeed](#) (unsigned short index, unsigned short maxspeed)
- unsigned short [GetPumpMaxSpeed](#) (unsigned short index)
- void [SetPumpSpeedUnit](#) (unsigned short index, int speedunit)
- int [GetPumpSpeedUnit](#) (unsigned short index)
- void [SetPumpModeType](#) (unsigned short index, PP\_Pump\_Mode\_Type\_EnumNet type)
- PP\_Pump\_Mode\_Type\_EnumNet [GetPumpModeType](#) (unsigned short index)
- void [SetPumpCouple](#) (unsigned int i)
- unsigned int [GetPumpCouple](#) ()
- void [SetPumpEnableSpeedRatio](#) (unsigned int enable)
- unsigned int [GetPumpEnableSpeedRatio](#) ()
- void [SetPumpManualOnOff](#) (unsigned short index, unsigned int onoff)
- unsigned int [GetPumpManualOnOff](#) (unsigned short index)
- void [SetPumpFunctionSpeeds](#) (unsigned short index, short offspeed, short onspeed)
- void [GetPumpFunctionSpeeds](#) (unsigned short index, [System::Runtime::InteropServices::Out]short% offspeed, [System::Runtime::InteropServices::Out]short% onspeed)
- void [SetPumpSpeedRatio](#) (int ratio)
- int [GetPumpSpeedRatio](#) ()
- void [SetPumpFastOnOff](#) (unsigned short index, unsigned int onoff)
- unsigned int [GetPumpFastOnOff](#) (unsigned short index)
- void [SetPumpFastSpeed](#) (unsigned short index, short fastspeed)
- short [GetPumpFastSpeed](#) (unsigned short index)
- void [SetAnalogVoltages](#) (unsigned short index, unsigned short minvoltage, unsigned short maxvoltage)
- void [GetAnalogVoltages](#) (unsigned short index, [System::Runtime::InteropServices::Out]unsigned short% minvoltage, [System::Runtime::InteropServices::Out]unsigned short% maxvoltage)
- void [SetUseBubble](#) (unsigned short index, unsigned int usebubble)
- unsigned int [GetUseBubble](#) (unsigned short index)
- unsigned short [GetSupplyVoltage](#) ()
- unsigned short [GetAnalogVoltage](#) (unsigned short index)
- unsigned short [GetDigitalIn](#) (unsigned short index)
- unsigned short [GetBubbleState](#) ()

## Additional Inherited Members

### 11.85.1 Constructor & Destructor Documentation

**11.85.1.1 CPPS\_FunctionNet()** [1/2] [CPPS\\_FunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb,  
[CMcsUsbFunctionPointerContainer](#)<sup>^</sup> cPPS\_FunctionPointerContainer )

**11.85.1.2 CPPS\_FunctionNet()** [2/2] [CPPS\\_FunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb )

## 11.85.2 Member Function Documentation

**11.85.2.1 GetAnalogVoltage()** unsigned short GetAnalogVoltage (   
 unsigned short *index* )

**11.85.2.2 GetAnalogVoltages()** void GetAnalogVoltages (   
 unsigned short *index*,   
 [System::Runtime::InteropServices::Out] unsigned short% *minvoltage*,   
 [System::Runtime::InteropServices::Out] unsigned short% *maxvoltage* )

**11.85.2.3 GetBubbleState()** unsigned short GetBubbleState ( )

**11.85.2.4 GetDigitalIn()** unsigned short GetDigitalIn (   
 unsigned short *index* )

**11.85.2.5 GetPumpCouple()** unsigned int GetPumpCouple ( )

**11.85.2.6 GetPumpEnableSpeedRatio()** unsigned int GetPumpEnableSpeedRatio ( )

**11.85.2.7 GetPumpFastOnOff()** unsigned int GetPumpFastOnOff (   
 unsigned short *index* )

**11.85.2.8 GetPumpFastSpeed()** short GetPumpFastSpeed (   
 unsigned short *index* )

**11.85.2.9 GetPumpFunctionSpeeds()** void GetPumpFunctionSpeeds (   
 unsigned short *index*,   
 [System::Runtime::InteropServices::Out] short% *offspeed*,   
 [System::Runtime::InteropServices::Out] short% *onspeed* )

**11.85.2.10 GetPumpManualOnOff()** unsigned int GetPumpManualOnOff (   
 unsigned short *index* )

**11.85.2.11 GetPumpMaxSpeed()** unsigned short GetPumpMaxSpeed (   
 unsigned short *index* )

**11.85.2.12 GetPumpModeType()** PP\_Pump\_Mode\_Type\_EnumNet GetPumpModeType (   
 unsigned short *index* )

**11.85.2.13 GetPumpSpeedRatio()** int GetPumpSpeedRatio ( )

**11.85.2.14 GetPumpSpeedUnit()** int GetPumpSpeedUnit (   
 unsigned short *index* )

**11.85.2.15 GetSupplyVoltage()** unsigned short GetSupplyVoltage ( )

**11.85.2.16 GetUseBubble()** unsigned int GetUseBubble (   
 unsigned short *index* )

**11.85.2.17 SetAnalogVoltages()** void SetAnalogVoltages (   
 unsigned short *index*,   
 unsigned short *minvoltage*,   
 unsigned short *maxvoltage* )

**11.85.2.18 SetPumpCouple()** void SetPumpCouple (   
 unsigned int *i* )

**11.85.2.19 SetPumpEnableSpeedRatio()** void SetPumpEnableSpeedRatio (   
 unsigned int *enable* )

**11.85.2.20 SetPumpFastOnOff()** void SetPumpFastOnOff (   
 unsigned short *index*,   
 unsigned int *onoff* )

**11.85.2.21 SetPumpFastSpeed()** void SetPumpFastSpeed (   
 unsigned short *index*,   
 short *fastspeed* )

**11.85.2.22 SetPumpFunctionSpeeds()** void SetPumpFunctionSpeeds (   
 unsigned short *index*,   
 short *offspeed*,   
 short *onspeed* )

**11.85.2.23 SetPumpManualOnOff()** void SetPumpManualOnOff (   
 unsigned short *index*,   
 unsigned int *onoff* )

**11.85.2.24 SetPumpMaxSpeed()** void SetPumpMaxSpeed (   
 unsigned short *index*,   
 unsigned short *maxspeed* )

**11.85.2.25 SetPumpModeType()** void SetPumpModeType (   
 unsigned short *index*,   
 PP\_Pump\_Mode\_Type\_EnumNet *type* )

**11.85.2.26 SetPumpSpeedRatio()** void SetPumpSpeedRatio (   
 int *ratio* )

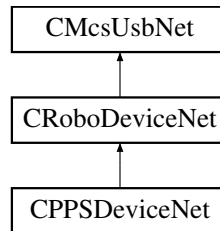
**11.85.2.27 SetPumpSpeedUnit()** void SetPumpSpeedUnit (   
 unsigned short *index*,   
 int *speedunit* )

**11.85.2.28 SetUseBubble()** `void SetUseBubble (`  
     `unsigned short index,`  
     `unsigned int usebubble )`

## 11.86 CPPSDeviceNet Class Reference

CPPS4plus1DeviceNet is the to control the MCS HLA device

Inheritance diagram for CPPSDeviceNet:



### Public Member Functions

- [CPPSDeviceNet](#) (void)

### Additional Inherited Members

#### 11.86.1 Detailed Description

CPPS4plus1DeviceNet is the to control the MCS HLA device

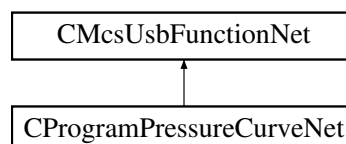
#### 11.86.2 Constructor & Destructor Documentation

**11.86.2.1 CPPSDeviceNet()** `CPPSDeviceNet (`  
     `void )`

## 11.87 CProgramPressureCurveNet Class Reference

[CProgramPressureCurveNet](#) is the class to program pressure curves

Inheritance diagram for CProgramPressureCurveNet:



## Public Member Functions

- [CProgramPressureCurveNet](#) ([CMcsUsbNet](#)^ mcsusb)  
*Initializes a new instance of the [CPulseGeneratorFunctionNet](#) class.*
- virtual [~CProgramPressureCurveNet](#) (void)
- [ICProgramPressureCurveNet](#) (void)
- void [Program](#) (unsigned char busnumber, unsigned char busaddress, int32\_t channel, array< int32\_t >^ pressures, array< int32\_t >^ steps, array< int16\_t >^ durations)
- void [SetRepeats](#) (unsigned char busnumber, unsigned char busaddress, int32\_t channel, uint32\_t repeats)
- unsigned int [GetRepeats](#) (unsigned char busnumber, unsigned char busaddress, int32\_t channel)

## Additional Inherited Members

### 11.87.1 Detailed Description

[CProgramPressureCurveNet](#) is the class to program pressure curves

### 11.87.2 Constructor & Destructor Documentation

**11.87.2.1 CProgramPressureCurveNet()** [CProgramPressureCurveNet](#) (  
    [CMcsUsbNet](#)^ mcsusb )

Initializes a new instance of the [CPulseGeneratorFunctionNet](#) class.

**11.87.2.2 ~CProgramPressureCurveNet()** virtual [~CProgramPressureCurveNet](#) (  
    void ) [virtual]

**11.87.2.3 !ICProgramPressureCurveNet()** [!CProgramPressureCurveNet](#) (  
    void )

### 11.87.3 Member Function Documentation

**11.87.3.1 GetRepeats()** unsigned int GetRepeats (  
    unsigned char busnumber,  
    unsigned char busaddress,  
    int32\_t channel )



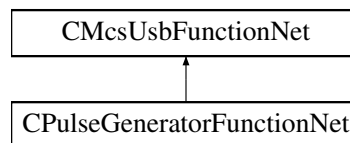
**11.87.3.2 Program()** void Program (   
     unsigned char *busnumber*,   
     unsigned char *busaddress*,   
     int32\_t *channel*,   
     array< int32\_t >^ *pressures*,   
     array< int32\_t >^ *steps*,   
     array< int16\_t >^ *durations* )

**11.87.3.3 SetRepeats()** void SetRepeats (   
     unsigned char *busnumber*,   
     unsigned char *busaddress*,   
     int32\_t *channel*,   
     uint32\_t *repeats* )

## 11.88 CPulseGeneratorFunctionNet Class Reference

[CPulseGeneratorFunctionNet](#) is the class to control the pulse generator for video tracking

Inheritance diagram for CPulseGeneratorFunctionNet:



### Public Member Functions

- [CPulseGeneratorFunctionNet](#) ([CMcsUsbNet](#)^ mcsusb, [CMcsUsbFunctionPointerContainer](#)^ pPulseGeneratorFunctionPointerContainer)   
*Initializes a new instance of the [CPulseGeneratorFunctionNet](#) class.*
- [CPulseGeneratorFunctionNet](#) ([CMcsUsbNet](#)^ mcsusb)
- virtual [~CPulseGeneratorFunctionNet](#) ()
- [!CPulseGeneratorFunctionNet](#) ()
- int32\_t [GetPeriod](#) (int32\_t generator\_number)   
*Reads the generator period*
- void [SetPeriod](#) (int32\_t generator\_number, int32\_t period\_in\_samples)   
*Writes the generator period*
- int32\_t [GetPulseLength](#) (int32\_t generator\_number)   
*Reads the generator pulse length*
- void [SetPulseLength](#) (int32\_t generator\_number, int32\_t pulselength\_in\_10us)   
*Writes the generator pulse length*
- void [GetModeSelect](#) (int32\_t generator\_number, [System::Runtime::InteropServices::Out]PulseGenerator\_Mode\_EnumNet% mode, [System::Runtime::InteropServices::Out]int32\_t% digitalchannel)   
*Reads the generator mode*
- void [SetModeSelect](#) (int32\_t generator\_number, PulseGenerator\_Mode\_EnumNet mode, int32\_t digitalchannel)   
*Writes the generator mode*

## Additional Inherited Members

### 11.88.1 Detailed Description

[CPulseGeneratorFunctionNet](#) is the class to control the pulse generator for video tracking

### 11.88.2 Constructor & Destructor Documentation

**11.88.2.1 CPulseGeneratorFunctionNet()** [1/2] [CPulseGeneratorFunctionNet](#) (   
[CMcsUsbNet](#)<sup>^</sup> *mcsusb*,   
[CMcsUsbFunctionPointerContainer](#)<sup>^</sup> *pPulseGeneratorFunctionPointerContainer* )

Initializes a new instance of the [CPulseGeneratorFunctionNet](#) class.

**11.88.2.2 CPulseGeneratorFunctionNet()** [2/2] [CPulseGeneratorFunctionNet](#) (   
[CMcsUsbNet](#)<sup>^</sup> *mcsusb* )

**11.88.2.3 ~CPulseGeneratorFunctionNet()** [virtual ~CPulseGeneratorFunctionNet](#) ( ) [virtual]

**11.88.2.4 "!CPulseGeneratorFunctionNet()** [!CPulseGeneratorFunctionNet](#) ( )

### 11.88.3 Member Function Documentation

**11.88.3.1 GetModeSelect()** [void GetModeSelect](#) (   
[int32\\_t](#) *generator\_number*,   
[\[System::Runtime::InteropServices::Out\]](#) [PulseGenerator\\_Mode\\_EnumNet](#)% *mode*,   
[\[System::Runtime::InteropServices::Out\]](#) [int32\\_t](#)% *digitalchannel* )

Reads the generator mode

#### Parameters

<i>generator_number</i>	The generator number
<i>mode</i>	The generator mode
<i>digitalchannel</i>	The digital in channel used as gate

**11.88.3.2 GetPeriod()** `int32_t GetPeriod (`  
`int32_t generator_number )`

Reads the generator period

Parameters

<i>generator_number</i>	The generator number
-------------------------	----------------------

Returns

The period

**11.88.3.3 GetPulseLength()** `int32_t GetPulseLength (`  
`int32_t generator_number )`

Reads the generator pulse length

Parameters

<i>generator_number</i>	The generator number
-------------------------	----------------------

Returns

The pulse length

**11.88.3.4 SetModeSelect()** `void SetModeSelect (`  
`int32_t generator_number,`  
`PulseGenerator_Mode_EnumNet mode,`  
`int32_t digitalchannel )`

Writes the generator mode

Parameters

<i>generator_number</i>	The generator number
<i>mode</i>	The generator mode
<i>digitalchannel</i>	The digital in channel used as gate

**11.88.3.5 SetPeriod()** `void SetPeriod (`

```
int32_t generator_number,
int32_t period_in_samples )
```

Writes the generator period

#### Parameters

<i>generator_number</i>	The generator number
<i>period_in_samples</i>	The period

**11.88.3.6 SetPulseLength()** void SetPulseLength (   
int32\_t generator\_number,   
int32\_t pulselength\_in\_10us )

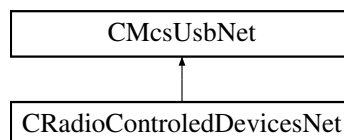
Writes the generator pulse length

#### Parameters

<i>generator_number</i>	The generator number
<i>pulselength_in_10us</i>	The pulse length

## 11.89 CRadioControlledDevicesNet Class Reference

Inheritance diagram for CRadioControlledDevicesNet:



### Public Member Functions

- [CRadioControlledDevicesNet](#) (void)
- bool [HasRadioControl](#) ()
- array< unsigned short > ^ [GetDeviceNames](#) ()
- void [ConnectDevice](#) (unsigned short sn)
- void [DisConnectDevice](#) ()
- bool [StillConnected](#) ()
- void [SetFrequency](#) (unsigned short frequency)
- unsigned short [GetFrequency](#) ()

### Protected Member Functions

- [CRadioControlledDevicesNet](#) (CRadioControlledDevices \*pRadioControlled)

## Additional Inherited Members

### 11.89.1 Constructor & Destructor Documentation

**11.89.1.1 CRadioControlledDevicesNet()** [1/2] `CRadioControlledDevicesNet ( void )`

**11.89.1.2 CRadioControlledDevicesNet()** [2/2] `CRadioControlledDevicesNet ( CRadioControlledDevices * pRadioControlled ) [protected]`

### 11.89.2 Member Function Documentation

**11.89.2.1 ConnectDevice()** `void ConnectDevice ( unsigned short sn )`

**11.89.2.2 DisconnectDevice()** `void DisconnectDevice ( )`

**11.89.2.3 GetDeviceNames()** `array<unsigned short> ^ GetDeviceNames ( )`

**11.89.2.4 GetFrequency()** `unsigned short GetFrequency ( )`

**11.89.2.5 HasRadioControl()** `bool HasRadioControl ( )`

**11.89.2.6 SetFrequency()** `void SetFrequency ( unsigned short frequency )`

**11.89.2.7 StillConnected()** `bool StillConnected ( )`

## 11.90 CCMOSMeaDeviceNet::CRegionOfInterestRect Class Reference

### Public Member Functions

- [CRegionOfInterestRect](#) (int left, int top, int right, int bottom)
- [CRegionOfInterestRect](#) ^ [DeepCopy](#) ()

### Public Attributes

- int [m\\_Left](#)
- int [m\\_Top](#)
- int [m\\_Right](#)
- int [m\\_Bottom](#)

### 11.90.1 Constructor & Destructor Documentation

**11.90.1.1** [CRegionOfInterestRect](#)() [CRegionOfInterestRect](#) (  
    int *left*,  
    int *top*,  
    int *right*,  
    int *bottom* )

### 11.90.2 Member Function Documentation

**11.90.2.1** [DeepCopy](#)() [CRegionOfInterestRect](#) ^ [DeepCopy](#) ( )

### 11.90.3 Member Data Documentation

**11.90.3.1** [m\\_Bottom](#) int m\_Bottom

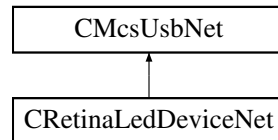
**11.90.3.2** [m\\_Left](#) int m\_Left

**11.90.3.3** [m\\_Right](#) int m\_Right

11.90.3.4 m\_Top `int m_Top`

## 11.91 CRetinaLedDeviceNet Class Reference

Inheritance diagram for CRetinaLedDeviceNet:



## Public Member Functions

- [CRetinaLedDeviceNet](#) ()
- [~CRetinaLedDeviceNet](#) ()
- unsigned int [SetTrigger](#) (int enable)
- unsigned int [SetLED](#) (unsigned long long pattern)
- unsigned int [SetTablepointer](#) (int position)
- unsigned int [GetTablepointer](#) (int % position)
- unsigned int [ClearTable](#) ()
- unsigned int [AddTableEntry](#) (unsigned long long pattern)
- unsigned int [AddLoopEntry](#) (unsigned short repeats, unsigned short steps\_back)
- unsigned int [SetRepeat](#) (int repeat)
- unsigned int [SetLumi](#) (int lumi)
- unsigned int [SetPersistency](#) (unsigned int persistency)

## Additional Inherited Members

## 11.91.1 Constructor &amp; Destructor Documentation

11.91.1.1 [CRetinaLedDeviceNet\(\)](#) [CRetinaLedDeviceNet](#) ( )

11.91.1.2 [~CRetinaLedDeviceNet\(\)](#) [~CRetinaLedDeviceNet](#) ( )

## 11.91.2 Member Function Documentation

11.91.2.1 [AddLoopEntry\(\)](#) unsigned int [AddLoopEntry](#) (  
     unsigned short *repeats*,  
     unsigned short *steps\_back* )

**11.91.2.2 AddTableEntry()** unsigned int AddTableEntry (  
    unsigned long long *pattern* )

**11.91.2.3 ClearTable()** unsigned int ClearTable ( )

**11.91.2.4 GetTablepointer()** unsigned int GetTablepointer (  
    int % *position* )

**11.91.2.5 SetLED()** unsigned int SetLED (  
    unsigned long long *pattern* )

**11.91.2.6 SetLumi()** unsigned int SetLumi (  
    int *lumi* )

**11.91.2.7 SetPersistency()** unsigned int SetPersistency (  
    unsigned int *persistency* )

**11.91.2.8 SetRepeat()** unsigned int SetRepeat (  
    int *repeat* )

**11.91.2.9 SetTablepointer()** unsigned int SetTablepointer (  
    int *position* )

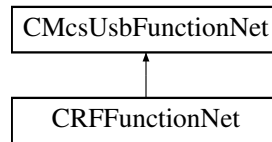
**11.91.2.10 SetTrigger()** unsigned int SetTrigger (  
    int *enable* )



## 11.92 CRFFunctionNet Class Reference

[CRFFunctionNet](#) is the class to control RF devices

Inheritance diagram for CRFFunctionNet:



### Public Member Functions

- [CRFFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> pRFFunctionPointerContainer)
  - Initializes a new instance of the [CRFFunctionNet](#) class.*
- [CRFFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- virtual [~CRFFunctionNet](#) ()
- [!CRFFunctionNet](#) ()
- uint32\_t [GetBaseFrequency](#) (CFirmwareDestinationNet destination)
  - gets the base advertise frequency*
- void [SetBaseFrequency](#) (CFirmwareDestinationNet destination, uint32\_t frequency)
  - sets the base advertise frequency*
- uint32\_t [GetWorkingFrequency](#) ()
  - gets the working frequency*
- void [SetWorkingFrequency](#) (uint32\_t frequency)
  - sets the working frequency*
- array< uint32\_t > <sup>^</sup> [GetAvailableDeviceListEx](#) (int list\_Length)
  - get a list of available devices*
- array< uint32\_t > <sup>^</sup> [GetAvailableDeviceList](#) ()
  - get a list of available devices*
- array< uint32\_t > <sup>^</sup> [GetAvailableStateListEx](#) (int list\_Length)
  - get a list of the states of the available devices*
- array< uint32\_t > <sup>^</sup> [GetAvailableStateList](#) ()
  - get a list of the states of the available devices*
- void [Connect](#) (uint32\_t sn)
  - connect to a RF device, use 0 to disconnect*
- uint32\_t [GetConnectedDevice](#) ()
  - get connect RF device, 0 = no device connected*
- uint32\_t [GetState](#) ()
  - get connection state*
- void [SetTestMode](#) (uint32\_t mode)
  - set test mode*
- uint32\_t [GetTestMode](#) ()
  - gets test mode*

### Additional Inherited Members

#### 11.92.1 Detailed Description

[CRFFunctionNet](#) is the class to control RF devices

### 11.92.2 Constructor & Destructor Documentation

**11.92.2.1 CRFFunctionNet()** [1/2] `CRFFunctionNet ( CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ pRFFunctionPointerContainer )`

Initializes a new instance of the `CRFFunctionNet` class.

**11.92.2.2 CRFFunctionNet()** [2/2] `CRFFunctionNet ( CMcsUsbNet^ mcsusb )`

**11.92.2.3 ~CRFFunctionNet()** `virtual ~CRFFunctionNet ( ) [virtual]`

**11.92.2.4 !CRFFunctionNet()** `!CRFFunctionNet ( )`

### 11.92.3 Member Function Documentation

**11.92.3.1 Connect()** `void Connect ( uint32_t sn )`

connect to a RF device, use 0 to disconnect

Parameters

<i>sn</i>	
-----------	--

**11.92.3.2 GetAvailableDeviceList()** `array<uint32_t> ^ GetAvailableDeviceList ( )`

get a list of available devices

Returns

**11.92.3.3 GetAvailableDeviceListEx()** `array<uint32_t> ^ GetAvailableDeviceListEx (`  
`int list_Length )`

get a list of available devices

Parameters

<i>list_Length</i>	The maximal length of list.
--------------------	-----------------------------

Returns

**11.92.3.4 GetAvailableStateList()** `array<uint32_t> ^ GetAvailableStateList ( )`

get a list of the states of the available devices

Returns

**11.92.3.5 GetAvailableStateListEx()** `array<uint32_t> ^ GetAvailableStateListEx (`  
`int list_Length )`

get a list of the states of the available devices

Parameters

<i>list_Length</i>	The maximal length of list.
--------------------	-----------------------------

Returns

**11.92.3.6 GetBaseFrequency()** `uint32_t GetBaseFrequency (`  
`CFirmwareDestinationNet destination )`

gets the base advertise frequency

Parameters

<i>destination</i>	
--------------------	--

Returns

**11.92.3.7 GetConnectedDevice()** `uint32_t GetConnectedDevice ( )`

get connect RF device, 0 = no device connected

Returns

**11.92.3.8 GetState()** `uint32_t GetState ( )`

get connection state

Returns

**11.92.3.9 GetTestMode()** `uint32_t GetTestMode ( )`

gets test mode

Returns

**11.92.3.10 GetWorkingFrequency()** `uint32_t GetWorkingFrequency ( )`

gets the working frequency

Returns

**11.92.3.11 SetBaseFrequency()** `void SetBaseFrequency (`  
    `CFirmwareDestinationNet destination,`  
    `uint32_t frequency )`

gets the base advertise frequency

## Parameters

<i>destination</i>	
<i>frequency</i>	

**11.92.3.12 SetTestMode()** void SetTestMode (   
 uint32\_t mode )

set test mode

## Parameters

<i>mode</i>	
-------------	--

**11.92.3.13 SetWorkingFrequency()** void SetWorkingFrequency (   
 uint32\_t frequency )

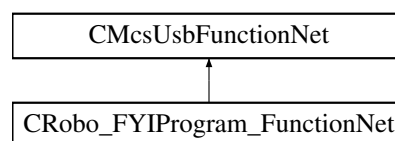
sets the working frequency

## Parameters

<i>frequency</i>	
------------------	--

## 11.93 CRobo\_FYIProgram\_FunctionNet Class Reference

Inheritance diagram for CRobo\_FYIProgram\_FunctionNet:



### Public Member Functions

- [CRobo\\_FYIProgram\\_FunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> robo\_FY↔ IProgram\_FunctionPointerContainer)
- [CRobo\\_FYIProgram\\_FunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- void [SetValve1](#) (unsigned char index, unsigned int valve1)
- unsigned int [GetValve1](#) (unsigned char index)
- void [SetValve2](#) (unsigned char index, unsigned int valve2)
- unsigned int [GetValve2](#) (unsigned char index)
- void [SetLength](#) (unsigned char index, int length)
- int [GetLength](#) (unsigned char index)
- void [Start](#) ()
- int [GetState](#) ()

## Additional Inherited Members

### 11.93.1 Constructor & Destructor Documentation

**11.93.1.1 CRobo\_FYIPProgram\_FunctionNet()** [1/2] CRobo\_FYIPProgram\_FunctionNet (   
 CMcsUsbNet^ *mcsusb*,   
 CMcsUsbFunctionPointerContainer^ *robo\_FYIPProgram\_FunctionPointerContainer* )

**11.93.1.2 CRobo\_FYIPProgram\_FunctionNet()** [2/2] CRobo\_FYIPProgram\_FunctionNet (   
 CMcsUsbNet^ *mcsusb* )

### 11.93.2 Member Function Documentation

**11.93.2.1 GetLength()** int GetLength (   
 unsigned char *index* )

**11.93.2.2 GetState()** int GetState ( )

**11.93.2.3 GetValve1()** unsigned int GetValve1 (   
 unsigned char *index* )

**11.93.2.4 GetValve2()** unsigned int GetValve2 (   
 unsigned char *index* )

**11.93.2.5 SetLength()** void SetLength (   
 unsigned char *index*,   
 int *length* )

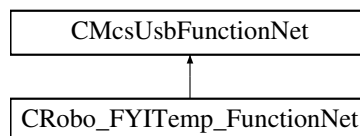
**11.93.2.6 SetValve1()** void SetValve1 (   
     unsigned char *index*,   
     unsigned int *valve1* )

**11.93.2.7 SetValve2()** void SetValve2 (   
     unsigned char *index*,   
     unsigned int *valve2* )

**11.93.2.8 Start()** void Start ( )

## 11.94 CRobo\_FYITemp\_FunctionNet Class Reference

Inheritance diagram for CRobo\_FYITemp\_FunctionNet:



### Public Member Functions

- [CRobo\\_FYITemp\\_FunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- void [SetRegulatorOnOff](#) (unsigned char *index*, int *onoff*)
- int [GetRegulatorOnOff](#) (unsigned char *index*)
- void [SetSollTemp](#) (unsigned char *index*, int *temp*)
- int [GetSollTemp](#) (unsigned char *index*)
- void [SetPCoeff](#) (unsigned char *index*, int *pcoeff*)
- int [GetPCoeff](#) (unsigned char *index*)
- void [SetICoeff](#) (unsigned char *index*, int *icoeff*)
- int [GetICoeff](#) (unsigned char *index*)
- void [SetMaxPower](#) (unsigned char *index*, int *power*)
- int [GetMaxPower](#) (unsigned char *index*)

### Additional Inherited Members

#### 11.94.1 Constructor & Destructor Documentation

**11.94.1.1 CRobo\_FYITemp\_FunctionNet()** [CRobo\\_FYITemp\\_FunctionNet](#) (   
     [CMcsUsbNet](#)<sup>^</sup> *mcsusb* )

## 11.94.2 Member Function Documentation

**11.94.2.1 GetICoeff()** `int GetICoeff ( unsigned char index )`

**11.94.2.2 GetMaxPower()** `int GetMaxPower ( unsigned char index )`

**11.94.2.3 GetPCoeff()** `int GetPCoeff ( unsigned char index )`

**11.94.2.4 GetRegulatorOnOff()** `int GetRegulatorOnOff ( unsigned char index )`

**11.94.2.5 GetSollTemp()** `int GetSollTemp ( unsigned char index )`

**11.94.2.6 SetICoeff()** `void SetICoeff ( unsigned char index,  
int icoeff )`

**11.94.2.7 SetMaxPower()** `void SetMaxPower ( unsigned char index,  
int power )`

**11.94.2.8 SetPCoeff()** `void SetPCoeff ( unsigned char index,  
int pcoeff )`

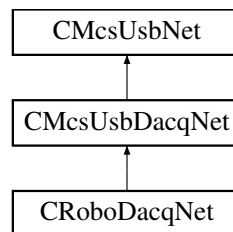


**11.94.2.9 SetRegulatorOnOff()** void SetRegulatorOnOff (   
     unsigned char *index*,   
     int *onoff* )

**11.94.2.10 SetSollTemp()** void SetSollTemp (   
     unsigned char *index*,   
     int *temp* )

## 11.95 CRoboDacqNet Class Reference

Inheritance diagram for CRoboDacqNet:



### Public Member Functions

- [CRoboDacqNet](#) (void)
- [CRoboDacqNet](#) ([CRoboDeviceNet](#)<sup>^</sup> robodevice)
- void [RunTable](#) ()
- void [RunTable](#) (int timeout)
- void [StopTable](#) ()
- void [StopTable](#) (int timeout)
- void [CancelTableLoop](#) ()
- void [CancelTableLoopAndStopTable](#) ()
- void [SetConfigurationBit](#) (unsigned short bit, bool value)
- void [SetConfigurationBitSupply](#) (bool value)
- void [SetConfigurationBitRelais](#) (bool value)
- void [SetConfigurationBitStream](#) (bool value)
- void [SetConfigurationBitAxc](#) (bool value)
- void [SetConfigurationBitCC\\_Gen](#) (bool value)
- void [SetConfigurationBitCV\\_Gen](#) (bool value)
- void [SetConfigurationBitRC\\_Gen](#) (bool value)
- void [SetConfigurationBitRV\\_Gen](#) (bool value)
- void [SetConfigurationBitBlu\\_Led](#) (bool value)
- void [SetConfigurationBitRed\\_Led](#) (bool value)
- void [SetConfigurationBitBlu\\_LedToggleSlow](#) (bool value)
- void [SetConfigurationBitRed\\_LedToggleSlow](#) (bool value)
- void [SetConfigurationBitBlu\\_LedToggleFast](#) (bool value)
- void [SetConfigurationBitRed\\_LedToggleFast](#) (bool value)
- void [SetConfigurationBitRed\\_LedSaturation](#) (bool value)
- void [SetSimulation](#) (unsigned int enable)
- void [SetUClamp](#) (int uClamp)
- void [SetIClamp](#) (int iClamp)

- void [SetPGain](#) (int pGain)
- void [SetIGain](#) (int iGain)
- void [SetFilter](#) (int filter)
- void [SetUJOffset](#) (int UJOffset)
- void [SetUCOffset](#) (int UCOffset)
- void [SetICOffset](#) (int ICOffset)
- void [SetCrossTalkOffset](#) (int CrossTalk)
- void [SetXGain](#) (int xGain)
- void [SetCrossTalkOptimum](#) (int cxOptimum)
- void [SetRecordingNumber](#) (unsigned int recordingNumber)
- void [ClampAmpRestart](#) ()
- void [DoRamp](#) (int startValue, int endValue, int duration, int mode)
- unsigned int [GetClampAmpSerialNumber](#) ()
- unsigned int [GetConfigurationBits](#) ()
- bool [GetConfigurationBit](#) (unsigned short bit)
- bool [GetConfigurationBitSupply](#) ()
- bool [GetConfigurationBitRelais](#) ()
- bool [GetConfigurationBitStream](#) ()
- bool [GetConfigurationBitAxc](#) ()
- bool [GetConfigurationBitCC\\_Gen](#) ()
- bool [GetConfigurationBitCV\\_Gen](#) ()
- bool [GetConfigurationBitRC\\_Gen](#) ()
- bool [GetConfigurationBitRV\\_Gen](#) ()
- bool [GetConfigurationBitBlu\\_Led](#) ()
- bool [GetConfigurationBitRed\\_Led](#) ()
- bool [GetConfigurationBitBlu\\_LedToggleSlow](#) ()
- bool [GetConfigurationBitRed\\_LedToggleSlow](#) ()
- bool [GetConfigurationBitBlu\\_LedToggleFast](#) ()
- bool [GetConfigurationBitRed\\_LedToggleFast](#) ()
- bool [GetConfigurationBitRed\\_LedSaturation](#) ()
- unsigned int [GetSimulation](#) ()
- int [GetUClamp](#) ()
- int [GetIClamp](#) ()
- int [GetPGain](#) ()
- int [GetIGain](#) ()
- int [GetFilter](#) ()
- int [GetUJOffset](#) ()
- int [GetUCOffset](#) ()
- int [GetICOffset](#) ()
- int [GetCrossTalkOffset](#) ()
- int [GetXGain](#) ()
- int [GetCrossTalkOptimum](#) ()
- unsigned int [GetRecordingNumber](#) ()
- int [GetResistanceC](#) ()
- int [GetResistanceV](#) ()
- int [GetCapacityC](#) ()
- int [GetCapacityV](#) ()
- int [GetCapacityX](#) ()
- int [GetUV](#) ()
- int [GetUC](#) ()
- int [GetIC](#) ()
- int [GetNUV\\_MS](#) ()
- int [GetNUC\\_MS](#) ()
- int [GetNIC\\_MS](#) ()
- void [SetAllDigout](#) (uint32\_t value)

- uint32\_t [GetAllDigout](#) ()
- void [SetCommand](#) (unsigned char command, int value)
- int [GetCommand](#) (unsigned char command)
- void [SetDigout](#) (uint16\_t index, bool enable)
- bool [GetDigout](#) (uint16\_t index)
- void [TableDefBegin](#) ()
- void [TableDefEnd](#) ()
- void [Table\\_Wait](#) (unsigned int tableWait)
- void [SetDownsampleFactor](#) (int index, int downsample\_factor)
- void [SetFilterCoeffs](#) (int index, array< int > ^ coeffs)
- int [GetDownsampleFactor](#) (int index)
- array< int > ^ [GetFilterCoeffs](#) (int index)
- void [Emu\\_SetElectrodeResists](#) (int emuElectrodeResist)
- void [Emu\\_SetCellResists](#) (int emuCellResist)
- void [Emu\\_SetCellCapacity](#) (int emuCellCapacity)
- void [Emu\\_SetCellPotential](#) (int emuCellPotential)
- void [Emu\\_SetNoise](#) (int emuNoise)
- int [Emu\\_GetElectrodeResists](#) ()
- int [Emu\\_GetCellResists](#) ()
- int [Emu\\_GetCellCapacity](#) ()
- int [Emu\\_GetCellPotential](#) ()
- int [Emu\\_GetNoise](#) ()
- void [SetDisplayText](#) (int index, String ^ displayText)
- void [SetScreen](#) (int screen)
- void [UpdateDisplay](#) ()
- String ^ [GetDisplayText](#) (int index)
- int [GetScreen](#) ()
- int [GetUpdateDisplay](#) ()

## Additional Inherited Members

### 11.95.1 Constructor & Destructor Documentation

**11.95.1.1 CRoboDacqNet()** [1/2] [CRoboDacqNet](#) (   
 void )

**11.95.1.2 CRoboDacqNet()** [2/2] [CRoboDacqNet](#) (   
 [CRoboDeviceNet](#) ^ robodevice )

### 11.95.2 Member Function Documentation

**11.95.2.1 CancelTableLoop()** void CancelTableLoop ( )

**11.95.2.2 CancelTableLoopAndStopTable()** `void CancelTableLoopAndStopTable ( )`

**11.95.2.3 ClampAmpRestart()** `void ClampAmpRestart ( )`

**11.95.2.4 DoRamp()** `void DoRamp (`  
    `int startValue,`  
    `int endValue,`  
    `int duration,`  
    `int mode )`

**11.95.2.5 Emu\_GetCellCapacity()** `int Emu_GetCellCapacity ( )`

**11.95.2.6 Emu\_GetCellPotential()** `int Emu_GetCellPotential ( )`

**11.95.2.7 Emu\_GetCellResists()** `int Emu_GetCellResists ( )`

**11.95.2.8 Emu\_GetElectrodeResists()** `int Emu_GetElectrodeResists ( )`

**11.95.2.9 Emu\_GetNoise()** `int Emu_GetNoise ( )`

**11.95.2.10 Emu\_SetCellCapacity()** `void Emu_SetCellCapacity (`  
    `int emuCellCapacity )`

**11.95.2.11 Emu\_SetCellPotential()** `void Emu_SetCellPotential (`  
    `int emuCellPotential )`

**11.95.2.12 Emu\_SetCellResists()** void Emu\_SetCellResists (   
int *emuCellResist* )

**11.95.2.13 Emu\_SetElectrodeResists()** void Emu\_SetElectrodeResists (   
int *emuElectrodeResist* )

**11.95.2.14 Emu\_SetNoise()** void Emu\_SetNoise (   
int *emuNoise* )

**11.95.2.15 GetAllDigout()** uint32\_t GetAllDigout ( )

**11.95.2.16 GetCapacityC()** int GetCapacityC ( )

**11.95.2.17 GetCapacityV()** int GetCapacityV ( )

**11.95.2.18 GetCapacityX()** int GetCapacityX ( )

**11.95.2.19 GetClampAmpSerialNumber()** unsigned int GetClampAmpSerialNumber ( )

**11.95.2.20 GetCommand()** int GetCommand (   
unsigned char *command* )

**11.95.2.21 GetConfigurationBit()** bool GetConfigurationBit (   
unsigned short *bit* )

**11.95.2.22 GetConfigurationBitAxc()** `bool GetConfigurationBitAxc ( )`

**11.95.2.23 GetConfigurationBitBlu\_Led()** `bool GetConfigurationBitBlu_Led ( )`

**11.95.2.24 GetConfigurationBitBlu\_LedToggleFast()** `bool GetConfigurationBitBlu_LedToggleFast ( )`

**11.95.2.25 GetConfigurationBitBlu\_LedToggleSlow()** `bool GetConfigurationBitBlu_LedToggleSlow ( )`

**11.95.2.26 GetConfigurationBitCC\_Gen()** `bool GetConfigurationBitCC_Gen ( )`

**11.95.2.27 GetConfigurationBitCV\_Gen()** `bool GetConfigurationBitCV_Gen ( )`

**11.95.2.28 GetConfigurationBitRC\_Gen()** `bool GetConfigurationBitRC_Gen ( )`

**11.95.2.29 GetConfigurationBitRed\_Led()** `bool GetConfigurationBitRed_Led ( )`

**11.95.2.30 GetConfigurationBitRed\_LedSaturation()** `bool GetConfigurationBitRed_LedSaturation ( )`

**11.95.2.31 GetConfigurationBitRed\_LedToggleFast()** `bool GetConfigurationBitRed_LedToggleFast ( )`

**11.95.2.32 GetConfigurationBitRed\_LedToggleSlow()** `bool GetConfigurationBitRed_LedToggleSlow ( )`

**11.95.2.33 GetConfigurationBitRelais()** `bool GetConfigurationBitRelais ( )`

**11.95.2.34 GetConfigurationBitRV\_Gen()** `bool GetConfigurationBitRV_Gen ( )`

**11.95.2.35 GetConfigurationBits()** `unsigned int GetConfigurationBits ( )`

**11.95.2.36 GetConfigurationBitStream()** `bool GetConfigurationBitStream ( )`

**11.95.2.37 GetConfigurationBitSupply()** `bool GetConfigurationBitSupply ( )`

**11.95.2.38 GetCrossTalkOffset()** `int GetCrossTalkOffset ( )`

**11.95.2.39 GetCrossTalkOptimum()** `int GetCrossTalkOptimum ( )`

**11.95.2.40 GetDigout()** `bool GetDigout (`  
`uint16_t index )`

**11.95.2.41 GetDisplayText()** `String ^ GetDisplayText (`  
`int index )`

**11.95.2.42 GetDownsampleFactor()** `int GetDownsampleFactor (`  
`int index )`

**11.95.2.43 GetFilter()** `int GetFilter ( )`

**11.95.2.44 GetFilterCoeffs()** `array<int> ^ GetFilterCoeffs (`  
`int index )`

**11.95.2.45 GetIC()** `int GetIC ( )`

**11.95.2.46 GetIClamp()** `int GetIClamp ( )`

**11.95.2.47 GetICOffset()** `int GetICOffset ( )`

**11.95.2.48 GetIGain()** `int GetIGain ( )`

**11.95.2.49 GetNIC\_MS()** `int GetNIC_MS ( )`

**11.95.2.50 GetNUC\_MS()** `int GetNUC_MS ( )`

**11.95.2.51 GetNUV\_MS()** `int GetNUV_MS ( )`

**11.95.2.52 GetPGain()** `int GetPGain ( )`

**11.95.2.53 GetRecordingNumber()** `unsigned int GetRecordingNumber ( )`

**11.95.2.54 GetResistanceC()** `int GetResistanceC ( )`



**11.95.2.55 GetResistanceV()** `int GetResistanceV ( )`

**11.95.2.56 GetScreen()** `int GetScreen ( )`

**11.95.2.57 GetSimulation()** `unsigned int GetSimulation ( )`

**11.95.2.58 GetUC()** `int GetUC ( )`

**11.95.2.59 GetUClamp()** `int GetUClamp ( )`

**11.95.2.60 GetUCOffset()** `int GetUCOffset ( )`

**11.95.2.61 GetUpdateDisplay()** `int GetUpdateDisplay ( )`

**11.95.2.62 GetUV()** `int GetUV ( )`

**11.95.2.63 GetUVOffset()** `int GetUVOffset ( )`

**11.95.2.64 GetXGain()** `int GetXGain ( )`

**11.95.2.65 RunTable()** `[1/2] void RunTable ( )`

**11.95.2.66 RunTable()** [2/2] void RunTable (  
int *timeout* )

**11.95.2.67 SetAllDigout()** void SetAllDigout (  
uint32\_t *value* )

**11.95.2.68 SetCommand()** void SetCommand (  
unsigned char *command*,  
int *value* )

**11.95.2.69 SetConfigurationBit()** void SetConfigurationBit (  
unsigned short *bit*,  
bool *value* )

**11.95.2.70 SetConfigurationBitAxc()** void SetConfigurationBitAxc (  
bool *value* )

**11.95.2.71 SetConfigurationBitBlu\_Led()** void SetConfigurationBitBlu\_Led (  
bool *value* )

**11.95.2.72 SetConfigurationBitBlu\_LedToggleFast()** void SetConfigurationBitBlu\_LedToggleFast (  
bool *value* )

**11.95.2.73 SetConfigurationBitBlu\_LedToggleSlow()** void SetConfigurationBitBlu\_LedToggleSlow (  
bool *value* )

**11.95.2.74 SetConfigurationBitCC\_Gen()** void SetConfigurationBitCC\_Gen (  
bool *value* )

**11.95.2.75 SetConfigurationBitCV\_Gen()** void SetConfigurationBitCV\_Gen (  
    bool value )

**11.95.2.76 SetConfigurationBitRC\_Gen()** void SetConfigurationBitRC\_Gen (  
    bool value )

**11.95.2.77 SetConfigurationBitRed\_Led()** void SetConfigurationBitRed\_Led (  
    bool value )

**11.95.2.78 SetConfigurationBitRed\_LedSaturation()** void SetConfigurationBitRed\_LedSaturation (  
    bool value )

**11.95.2.79 SetConfigurationBitRed\_LedToggleFast()** void SetConfigurationBitRed\_LedToggleFast (  
    bool value )

**11.95.2.80 SetConfigurationBitRed\_LedToggleSlow()** void SetConfigurationBitRed\_LedToggleSlow (  
    bool value )

**11.95.2.81 SetConfigurationBitRelais()** void SetConfigurationBitRelais (  
    bool value )

**11.95.2.82 SetConfigurationBitRV\_Gen()** void SetConfigurationBitRV\_Gen (  
    bool value )

**11.95.2.83 SetConfigurationBitStream()** void SetConfigurationBitStream (  
    bool value )

**11.95.2.84 SetConfigurationBitSupply()** void SetConfigurationBitSupply (  
    bool value )

**11.95.2.85 SetCrossTalkOffset()** void SetCrossTalkOffset (   
int *CrossTalk* )

**11.95.2.86 SetCrossTalkOptimum()** void SetCrossTalkOptimum (   
int *cxOptimum* )

**11.95.2.87 SetDigout()** void SetDigout (   
uint16\_t *index*,   
bool *enable* )

**11.95.2.88 SetDisplayText()** void SetDisplayText (   
int *index*,   
String^ *displayText* )

**11.95.2.89 SetDownsampleFactor()** void SetDownsampleFactor (   
int *index*,   
int *downsample\_factor* )

**11.95.2.90 SetFilter()** void SetFilter (   
int *filter* )

**11.95.2.91 SetFilterCoeffs()** void SetFilterCoeffs (   
int *index*,   
array< int >^ *coeffs* )

**11.95.2.92 SetIClamp()** void SetIClamp (   
int *iClamp* )

**11.95.2.93 SetICOffset()** void SetICOffset (   
int *ICOffset* )

**11.95.2.94 SetIGain()** void SetIGain (  
int *iGain* )

**11.95.2.95 SetPGain()** void SetPGain (  
int *pGain* )

**11.95.2.96 SetRecordingNumber()** void SetRecordingNumber (  
unsigned int *recordingNumber* )

**11.95.2.97 SetScreen()** void SetScreen (  
int *screen* )

**11.95.2.98 SetSimulation()** void SetSimulation (  
unsigned int *enable* )

**11.95.2.99 SetUClamp()** void SetUClamp (  
int *uClamp* )

**11.95.2.100 SetUCOffset()** void SetUCOffset (  
int *UCOffset* )

**11.95.2.101 SetUVOffset()** void SetUVOffset (  
int *UVOffset* )

**11.95.2.102 SetXGain()** void SetXGain (  
int *xGain* )

**11.95.2.103 StopTable()** [1/2] void StopTable ( )

**11.95.2.104 StopTable()** [2/2] `void StopTable (`  
`int timeout )`

**11.95.2.105 Table\_Wait()** `void Table_Wait (`  
`unsigned int tableWait )`

**11.95.2.106 TableDefBegin()** `void TableDefBegin ( )`

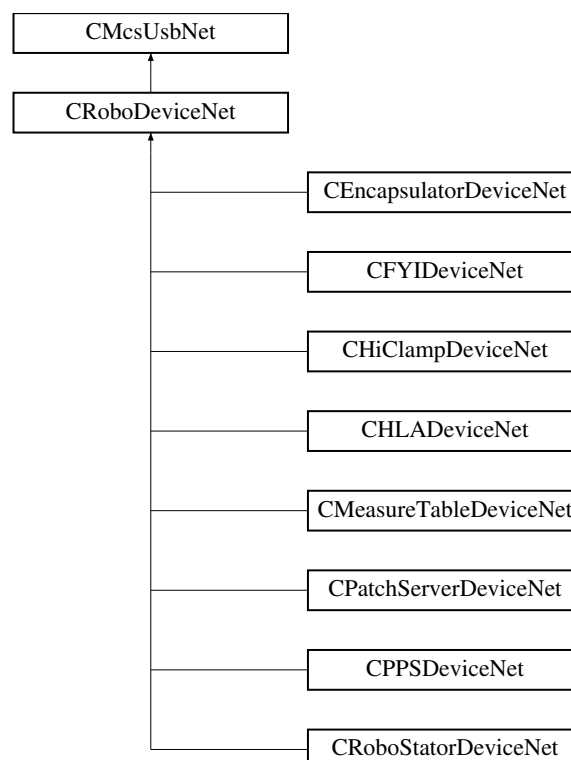
**11.95.2.107 TableDefEnd()** `void TableDefEnd ( )`

**11.95.2.108 UpdateDisplay()** `void UpdateDisplay ( )`

## 11.96 CRoboDeviceNet Class Reference

[CRoboDeviceNet](#) is the base class for all Robo platform based devices

Inheritance diagram for CRoboDeviceNet:



## Classes

- class [RoboMainLowLevelCommands](#)

## Public Member Functions

- [CRoboDeviceNet](#) (void)
- [~CRoboDeviceNet](#) (void)
- void [SetInMovement](#) ()  
*Low level command, sets the internal state to "In Movement"*
- bool [GetInMovement](#) ()  
*Low level command, gets the internal state "In Movement"*
- uint32\_t [GetMovementError](#) ()  
*Low level command, gets the error of the last movement end*
- void [FindReference](#) (unsigned char busaddress, char axes)
- void [FindReference](#) (unsigned char busaddress, char axes, int timeout)  
*Searches the reference position of the motor*
- void [MoveAbs](#) (unsigned char busaddress, char axes, int x, int y)
- void [MoveAbs](#) (unsigned char busaddress, char axes, int x, int y, int timeout)  
*Moves the motor to the new absolute position*
- void [StopMovement](#) (unsigned char busaddress, char axes)
- void [StopMovement](#) (unsigned char busaddress, char axes, int timeout)  
*Stops the current movement*
- void [SetCurrentAndAir](#) (unsigned char busaddress, char axes, unsigned short onoff)
- void [SetCurrentAndAir](#) (unsigned char busaddress, char axes, unsigned short onoff, int timeout)
- void [CancelPoolLoop](#) ()
- void [CancelPoolLoopAndStopMovement](#) ()
- void [GetCurrentPosition](#) (unsigned char busaddress, char axes, [System::Runtime::InteropServices::Out]int% x, [System::Runtime::InteropServices::Out]int% y)  
*Gets the current position of motors*
- void [SetAirValve](#) (unsigned int onoff)
- unsigned int [GetAirValve](#) ()
- unsigned int [GetVoltageValves](#) ()
- unsigned int [GetVoltageRs485A](#) ()
- unsigned int [GetVoltageRs485B](#) ()
- unsigned int [GetVoltageAirvalve](#) ()
- unsigned int [GetCurrentAirvalve](#) ()
- unsigned int [GetVoltage12V](#) ()
- unsigned int [GetAirpressure](#) ()
- unsigned int [GetVoltage5V](#) ()
- unsigned int [GetErrorVoltageValves](#) ()
- unsigned int [GetErrorVoltageRs485A](#) ()
- unsigned int [GetErrorVoltageRs485B](#) ()
- unsigned int [GetErrorVoltageAirvalve](#) ()
- unsigned int [GetErrorCurrentAirvalve](#) ()
- unsigned int [GetErrorVoltage12V](#) ()
- unsigned int [GetErrorAirpressure](#) ()
- unsigned int [GetErrorVoltage5V](#) ()
- void [SetVoltageValvesLimit](#) (unsigned int lowervoltage, unsigned int uppervoltage)
- void [SetVoltageRs485ALimit](#) (unsigned int lowervoltage, unsigned int uppervoltage)
- void [SetVoltageRs485BLimit](#) (unsigned int lowervoltage, unsigned int uppervoltage)
- void [SetVoltageAirvalveLimit](#) (unsigned int lowervoltage, unsigned int uppervoltage)
- void [SetCurrentAirvalveLimit](#) (unsigned int lowercurrent, unsigned int uppercurrent)

- void [SetVoltage12VLimit](#) (unsigned int lowervoltage, unsigned int uppervoltage)
- void [SetAirpressureLimit](#) (unsigned int lowerpressure, unsigned int upperpressure)
- void [SetVoltage5VLimit](#) (unsigned int lowervoltage, unsigned int uppervoltage)
- void [GetVoltageValvesLimit](#) ([System::Runtime::InteropServices::Out] unsigned int% lowervoltage, [System::Runtime::InteropServices::Out] unsigned int% uppervoltage)
- void [GetVoltageRs485ALimit](#) ([System::Runtime::InteropServices::Out] unsigned int% lowervoltage, [System::Runtime::InteropServices::Out] unsigned int% uppervoltage)
- void [GetVoltageRs485BLimit](#) ([System::Runtime::InteropServices::Out] unsigned int% lowervoltage, [System::Runtime::InteropServices::Out] unsigned int% uppervoltage)
- void [GetVoltageAirvalveLimit](#) ([System::Runtime::InteropServices::Out] unsigned int% lowervoltage, [System::Runtime::InteropServices::Out] unsigned int% uppervoltage)
- void [GetCurrentAirvalveLimit](#) ([System::Runtime::InteropServices::Out] unsigned int% lowercurrent, [System::Runtime::InteropServices::Out] unsigned int% uppercurrent)
- void [GetVoltage12VLimit](#) ([System::Runtime::InteropServices::Out] unsigned int% lowervoltage, [System::Runtime::InteropServices::Out] unsigned int% uppervoltage)
- void [GetAirpressureLimit](#) ([System::Runtime::InteropServices::Out] unsigned int% lowerpressure, [System::Runtime::InteropServices::Out] unsigned int% upperpressure)
- void [GetVoltage5VLimit](#) ([System::Runtime::InteropServices::Out] unsigned int% lowervoltage, [System::Runtime::InteropServices::Out] unsigned int% uppervoltage)
- void [SetMinPressure](#) (int pressure)
- int [GetMinPressure](#) ()

### Static Public Attributes

- static const uint32\_t [RoboError\\_Base](#) = (0xA0110000L)
- static const uint32\_t [RoboError\\_UnknownCommand](#) = ( (0xA0110000L) )
- static const uint32\_t [RoboError\\_Timeout](#) = ( (0xA0110000L) | 0x0001 )
- static const uint32\_t [RoboError\\_Pressure](#) = ( (0xA0110000L) | 0x0002 )
- static const uint32\_t [RoboError\\_RangeExceeded](#) = ( (0xA0110000L) | 0x0003 )
- static const uint32\_t [RoboError\\_CommunicationTimeout](#) = ( (0xA0110000L) | 0x0004 )
- static const uint32\_t [RoboError\\_AnotherMaster](#) = ( (0xA0110000L) | 0x0005 )
- static const uint32\_t [RoboError\\_FindReferenceMethod](#) = ( (0xA0110000L) | 0x0006 )
- static const uint32\_t [RoboError\\_NoSpeedOrAcceleration](#) = ( (0xA0110000L) | 0x0007 )
- static const uint32\_t [RoboError\\_NoEndSwitch](#) = ( (0xA0110000L) | 0x0008 )
- static const uint32\_t [RoboError\\_CannotEscapeEndSwitch](#) = ( (0xA0110000L) | 0x0009 )
- static const uint32\_t [RoboError\\_CommandAlreadyInProgress](#) = ( (0xA0110000L) | 0x000A )
- static const uint32\_t [RoboError\\_NoReference](#) = ( (0xA0110000L) | 0x000B )
- static const uint32\_t [RoboError\\_OverPressure](#) = ( (0xA0110000L) | 0x000C )
- static const uint32\_t [RoboError\\_Phase0OutOfRange](#) = ( (0xA0110000L) | 0x000D )
- static const uint32\_t [RoboError\\_PeristalticTimeout](#) = ( (0xA0110000L) | 0x000E )
- static const uint32\_t [RoboError\\_GilsonTimeout](#) = ( (0xA0110000L) | 0x000F )
- static const uint32\_t [RoboError\\_GilsonWrondID](#) = ( (0xA0110000L) | 0x0010 )
- static const uint32\_t [RoboError\\_GilsonCommandPending](#) = ( (0xA0110000L) | 0x0011 )
- static const uint32\_t [RoboError\\_ParameterNotAllowed](#) = ( (0xA0110000L) | 0x0012 )
- static const uint32\_t [RoboError\\_StateChangeNotPossible](#) = ( (0xA0110000L) | 0x0013 )
- static const uint32\_t [RoboError\\_CommandNotPossible](#) = ( (0xA0110000L) | 0x0014 )
- static const uint32\_t [RoboError\\_DacqNotReady](#) = ( (0xA0110000L) | 0x0015 )
- static const uint32\_t [RoboError\\_NoMoreData](#) = ( (0xA0110000L) | 0x0016 )
- static const uint32\_t [RoboError\\_McsBus\\_UnknownCommand](#) = ( (0xA0110000L) | 0x003F )
- static const uint32\_t [RoboError\\_DLLMovementTimeout](#) = ( (0xA0110000L) | 0x1001 )
- static const uint32\_t [RoboError\\_PollLoopCanceled](#) = ( (0xA0110000L) | 0x1002 )
- static const uint32\_t [RoboError\\_PollLoopCanceledAndStopMovement](#) = ( (0xA0110000L) | 0x1003 )
- static const byte [McsBus\\_XY](#) = 1

*McsBus address for the xy-plane*



- static const byte [McsBus\\_ZI](#) = 2  
*McsBus address for the z and i axes*
- static const byte [Axis\\_X](#) = 0  
*Axis number of x for axis argument*
- static const byte [Axis\\_Y](#) = 1  
*Axis number of y for axis argument*
- static const byte [Axis\\_Z](#) = 0  
*Axis number of z for axis argument*
- static const byte [Axis\\_I](#) = 1  
*Axis number of i for axis argument*
- static const char [Axes\\_X](#) = 1  
*Bit pattern for x axis for axes argument*
- static const char [Axes\\_Y](#) = 2  
*Bit pattern for y axis for axes argument*
- static const char [Axes\\_Z](#) = 1  
*Bit pattern for z axis for axes argument*
- static const char [Axes\\_I](#) = 2  
*Bit pattern for i axis for axes argument*

## Properties

- [CMcsBusNet](#)<sup>^</sup> [McsBus](#) [get]
- [CMcsBus\\_MotorControlNet](#)<sup>^</sup> [McsBus\\_MotorControl](#) [get]
- [RoboMainLowLevelCommands](#)<sup>^</sup> [RoboMainLowLevelCommand](#) [get]

## Events

- [RoboStatusEventDelegate](#)<sup>^</sup> [RoboStatusEvent](#)

## Additional Inherited Members

### 11.96.1 Detailed Description

[CRoboDeviceNet](#) is the base class for all Robo platform based devices

### 11.96.2 Constructor & Destructor Documentation

**11.96.2.1 CRoboDeviceNet()** [CRoboDeviceNet](#) (  
void )

**11.96.2.2 ~CRoboDeviceNet()** [~CRoboDeviceNet](#) (  
void )

### 11.96.3 Member Function Documentation

**11.96.3.1 CancelPoolLoop()** `void CancelPoolLoop ( )`

**11.96.3.2 CancelPoolLoopAndStopMovement()** `void CancelPoolLoopAndStopMovement ( )`

**11.96.3.3 FindReference() [1/2]** `void FindReference (`  
    `unsigned char busaddress,`  
    `char axes )`

**11.96.3.4 FindReference() [2/2]** `void FindReference (`  
    `unsigned char busaddress,`  
    `char axes,`  
    `int timeout )`

Searches the reference position of the motor

#### Parameters

<i>busaddress</i>	Address of the McsBus
<i>axes</i>	Bit pattern of axes to drive
<i>timeout</i>	Timeout of maximal waiting for the end of the command (-1 is forever)

**11.96.3.5 GetAirpressure()** `unsigned int GetAirpressure ( )`

**11.96.3.6 GetAirpressureLimit()** `void GetAirpressureLimit (`  
    `[System::Runtime::InteropServices::Out] unsigned int% lowerpressure,`  
    `[System::Runtime::InteropServices::Out] unsigned int% upperpressure )`

**11.96.3.7 GetAirValve()** `unsigned int GetAirValve ( )`

**11.96.3.8 GetCurrentAirvalve()** unsigned int GetCurrentAirvalve ( )

**11.96.3.9 GetCurrentAirvalveLimit()** void GetCurrentAirvalveLimit (   
 [System::Runtime::InteropServices::Out] unsigned int% *lowercurrent*,   
 [System::Runtime::InteropServices::Out] unsigned int% *uppercurrent* )

**11.96.3.10 GetCurrentPosition()** void GetCurrentPosition (   
 unsigned char *busaddress*,   
 char *axes*,   
 [System::Runtime::InteropServices::Out] int% *x*,   
 [System::Runtime::InteropServices::Out] int% *y* )

Gets the current position of motors

**Parameters**

<i>busaddress</i>	Address of the McsBus
<i>axes</i>	Bit pattern of axes to drive
<i>x</i>	Current position of first axis if pattern in axes is set
<i>y</i>	Current position of second axis if pattern in axes is set

**11.96.3.11 GetErrorAirpressure()** unsigned int GetErrorAirpressure ( )

**11.96.3.12 GetErrorCurrentAirvalve()** unsigned int GetErrorCurrentAirvalve ( )

**11.96.3.13 GetErrorVoltage12V()** unsigned int GetErrorVoltage12V ( )

**11.96.3.14 GetErrorVoltage5V()** unsigned int GetErrorVoltage5V ( )

**11.96.3.15 GetErrorVoltageAirvalve()** unsigned int GetErrorVoltageAirvalve ( )

**11.96.3.16 GetErrorVoltageRs485A()** unsigned int GetErrorVoltageRs485A ( )

**11.96.3.17 GetErrorVoltageRs485B()** unsigned int GetErrorVoltageRs485B ( )

**11.96.3.18 GetErrorVoltageValves()** unsigned int GetErrorVoltageValves ( )

**11.96.3.19 GetInMovement()** bool GetInMovement ( )

Low level command, gets the internal state "In Movement"

**11.96.3.20 GetMinPressure()** int GetMinPressure ( )

**11.96.3.21 GetMovementError()** uint32\_t GetMovementError ( )

Low level command, gets the error of the last movement end

**11.96.3.22 GetVoltage12V()** unsigned int GetVoltage12V ( )

**11.96.3.23 GetVoltage12VLimit()** void GetVoltage12VLimit (   
 [System::Runtime::InteropServices::Out] unsigned int% *lowervoltage*,   
 [System::Runtime::InteropServices::Out] unsigned int% *uppervoltage* )

**11.96.3.24 GetVoltage5V()** unsigned int GetVoltage5V ( )

**11.96.3.25 GetVoltage5VLimit()** void GetVoltage5VLimit (   
 [System::Runtime::InteropServices::Out] unsigned int% *lowervoltage*,   
 [System::Runtime::InteropServices::Out] unsigned int% *uppervoltage* )

**11.96.3.26 GetVoltageAirvalve()** unsigned int GetVoltageAirvalve ( )

**11.96.3.27 GetVoltageAirvalveLimit()** void GetVoltageAirvalveLimit (   
 [System::Runtime::InteropServices::Out] unsigned int% *lowervoltage*,   
 [System::Runtime::InteropServices::Out] unsigned int% *uppervoltage* )

**11.96.3.28 GetVoltageRs485A()** unsigned int GetVoltageRs485A ( )

**11.96.3.29 GetVoltageRs485ALimit()** void GetVoltageRs485ALimit (   
 [System::Runtime::InteropServices::Out] unsigned int% *lowervoltage*,   
 [System::Runtime::InteropServices::Out] unsigned int% *uppervoltage* )

**11.96.3.30 GetVoltageRs485B()** unsigned int GetVoltageRs485B ( )

**11.96.3.31 GetVoltageRs485BLimit()** void GetVoltageRs485BLimit (   
 [System::Runtime::InteropServices::Out] unsigned int% *lowervoltage*,   
 [System::Runtime::InteropServices::Out] unsigned int% *uppervoltage* )

**11.96.3.32 GetVoltageValves()** unsigned int GetVoltageValves ( )

**11.96.3.33 GetVoltageValvesLimit()** void GetVoltageValvesLimit (   
 [System::Runtime::InteropServices::Out] unsigned int% *lowervoltage*,   
 [System::Runtime::InteropServices::Out] unsigned int% *uppervoltage* )

**11.96.3.34 MoveAbs() [1/2]** void MoveAbs (   
 unsigned char *busaddress*,   
 char *axes*,   
 int *x*,   
 int *y* )

**11.96.3.35 MoveAbs() [2/2]** void MoveAbs (   
 unsigned char *busaddress*,   
 char *axes*,   
 int *x*,   
 int *y*,   
 int *timeout* )

Moves the motor to the new absolute position

## Parameters

<i>busaddress</i>	Address of the McsBus
<i>axes</i>	Bit pattern of axes to drive
<i>x</i>	Position of first axis, if pattern in axes is set
<i>y</i>	Position of second axis if pattern in axes is set
<i>timeout</i>	Timeout of maximal waiting for the end of the command (-1 is forever)

**11.96.3.36 SetAirpressureLimit()** `void SetAirpressureLimit (`  
    `unsigned int lowerpressure,`  
    `unsigned int upperpressure )`

**11.96.3.37 SetAirValve()** `void SetAirValve (`  
    `unsigned int onoff )`

**11.96.3.38 SetCurrentAirvalveLimit()** `void SetCurrentAirvalveLimit (`  
    `unsigned int lowercurrent,`  
    `unsigned int uppercurrent )`

**11.96.3.39 SetCurrentAndAir() [1/2]** `void SetCurrentAndAir (`  
    `unsigned char busaddress,`  
    `char axes,`  
    `unsigned short onoff )`

**11.96.3.40 SetCurrentAndAir() [2/2]** `void SetCurrentAndAir (`  
    `unsigned char busaddress,`  
    `char axes,`  
    `unsigned short onoff,`  
    `int timeout )`

**11.96.3.41 SetInMovement()** `void SetInMovement ( )`

Low level command, sets the internal state to "In Movement"

**11.96.3.42 SetMinPressure()** void SetMinPressure (   
int *pressure* )

**11.96.3.43 SetVoltage12VLimit()** void SetVoltage12VLimit (   
unsigned int *lowervoltage*,   
unsigned int *uppervoltage* )

**11.96.3.44 SetVoltage5VLimit()** void SetVoltage5VLimit (   
unsigned int *lowervoltage*,   
unsigned int *uppervoltage* )

**11.96.3.45 SetVoltageAirvalveLimit()** void SetVoltageAirvalveLimit (   
unsigned int *lowervoltage*,   
unsigned int *uppervoltage* )

**11.96.3.46 SetVoltageRs485ALimit()** void SetVoltageRs485ALimit (   
unsigned int *lowervoltage*,   
unsigned int *uppervoltage* )

**11.96.3.47 SetVoltageRs485BLimit()** void SetVoltageRs485BLimit (   
unsigned int *lowervoltage*,   
unsigned int *uppervoltage* )

**11.96.3.48 SetVoltageValvesLimit()** void SetVoltageValvesLimit (   
unsigned int *lowervoltage*,   
unsigned int *uppervoltage* )

**11.96.3.49 StopMovement() [1/2]** void StopMovement (   
unsigned char *busaddress*,   
char *axes* )

**11.96.3.50 StopMovement() [2/2]** void StopMovement (   
unsigned char *busaddress*,   
char *axes*,   
int *timeout* )

Stops the current movement

## Parameters

<i>busaddress</i>	Address of the McsBus
<i>axes</i>	Bit pattern of axes to drive
<i>timeout</i>	Timeout of maximal waiting for the end of the command (-1 is forever)

#### 11.96.4 Member Data Documentation

**11.96.4.1 Axes\_I** `const char Axes_I = 2 [static]`

Bit pattern for i axis for axes argument

**11.96.4.2 Axes\_X** `const char Axes_X = 1 [static]`

Bit pattern for x axis for axes argument

**11.96.4.3 Axes\_Y** `const char Axes_Y = 2 [static]`

Bit pattern for y axis for axes argument

**11.96.4.4 Axes\_Z** `const char Axes_Z = 1 [static]`

Bit pattern for z axis for axes argument

**11.96.4.5 Axis\_I** `const byte Axis_I = 1 [static]`

Axis number of i for axis argument

**11.96.4.6 Axis\_X** `const byte Axis_X = 0 [static]`

Axis number of x for axis argument



**11.96.4.7 Axis\_Y** `const byte Axis_Y = 1 [static]`

Axis number of y for axis argument

**11.96.4.8 Axis\_Z** `const byte Axis_Z = 0 [static]`

Axis number of z for axis argument

**11.96.4.9 McsBus\_XY** `const byte McsBus_XY = 1 [static]`

McsBus address for the xy-plane

**11.96.4.10 McsBus\_ZI** `const byte McsBus_ZI = 2 [static]`

McsBus address for the z and i axes

**11.96.4.11 RoboError\_AnotherMaster** `const uint32_t RoboError_AnotherMaster = ( (0xA0110000L) | 0x0005 ) [static]`

**11.96.4.12 RoboError\_Base** `const uint32_t RoboError_Base = (0xA0110000L) [static]`

**11.96.4.13 RoboError\_CannotEscapeEndSwitch** `const uint32_t RoboError_CannotEscapeEndSwitch = ( (0xA0110000L) | 0x0009 ) [static]`

**11.96.4.14 RoboError\_CommandAlreadyInProgress** `const uint32_t RoboError_CommandAlreadyInProgress = ( (0xA0110000L) | 0x000A ) [static]`

**11.96.4.15 RoboError\_CommandNotPossible** `const uint32_t RoboError_CommandNotPossible = ( (0xA0110000L) | 0x0014 ) [static]`

**11.96.4.16 RoboError\_CommunicationTimeout** `const uint32_t RoboError_CommunicationTimeout = ( (0xA0110000L) | 0x0004 ) [static]`

**11.96.4.17 RoboError\_DacqNotReady** `const uint32_t RoboError_DacqNotReady = ( (0xA0110000L) | 0x0015 ) [static]`

**11.96.4.18 RoboError\_DLLMovementTimeout** `const uint32_t RoboError_DLLMovementTimeout = ( (0xA0110000L) | 0x1001 ) [static]`

**11.96.4.19 RoboError\_FindReferenceMethod** `const uint32_t RoboError_FindReferenceMethod = ( (0xA0110000L) | 0x0006 ) [static]`

**11.96.4.20 RoboError\_GilsonCommandPending** `const uint32_t RoboError_GilsonCommandPending = ( (0xA0110000L) | 0x0011 ) [static]`

**11.96.4.21 RoboError\_GilsonTimeout** `const uint32_t RoboError_GilsonTimeout = ( (0xA0110000L) | 0x000F ) [static]`

**11.96.4.22 RoboError\_GilsonWrondID** `const uint32_t RoboError_GilsonWrondID = ( (0xA0110000L) | 0x0010 ) [static]`

**11.96.4.23 RoboError\_McsBus\_UnknownCommand** `const uint32_t RoboError_McsBus_UnknownCommand = ( (0xA0110000L) | 0x003F ) [static]`

**11.96.4.24 RoboError\_NoEndSwitch** `const uint32_t RoboError_NoEndSwitch = ( (0xA0110000L) | 0x0008 ) [static]`

**11.96.4.25 RoboError\_NoMoreData** `const uint32_t RoboError_NoMoreData = ( (0xA0110000L) | 0x0016 ) [static]`

**11.96.4.26 RoboError\_NoReference** `const uint32_t RoboError_NoReference = ( (0xA0110000L) | 0x000B ) [static]`

**11.96.4.27 RoboError\_NoSpeedOrAcceleration** `const uint32_t RoboError_NoSpeedOrAcceleration = ( (0xA0110000L) | 0x0007 ) [static]`

**11.96.4.28 RoboError\_OverPressure** `const uint32_t RoboError_OverPressure = ( (0xA0110000L) | 0x000C ) [static]`

**11.96.4.29 RoboError\_ParameterNotAllowed** `const uint32_t RoboError_ParameterNotAllowed = ( (0xA0110000L) | 0x0012 ) [static]`

**11.96.4.30 RoboError\_PeristalticTimeout** `const uint32_t RoboError_PeristalticTimeout = ( (0xA0110000L) | 0x000E ) [static]`

**11.96.4.31 RoboError\_Phase0OutOfRange** `const uint32_t RoboError_Phase0OutOfRange = ( (0xA0110000L) | 0x000D ) [static]`

**11.96.4.32 RoboError\_PollLoopCanceled** `const uint32_t RoboError_PollLoopCanceled = ( (0xA0110000L) | 0x1002 ) [static]`

**11.96.4.33 RoboError\_PollLoopCanceledAndStopMovement** `const uint32_t RoboError_PollLoopCanceledAndStopMovement = ( (0xA0110000L) | 0x1003 ) [static]`

**11.96.4.34 RoboError\_Pressure** `const uint32_t RoboError_Pressure = ( (0xA0110000L) | 0x0002 ) [static]`

**11.96.4.35 RoboError\_RangeExceeded** `const uint32_t RoboError_RangeExceeded = ( (0xA0110000L) | 0x0003 ) [static]`

**11.96.4.36 RoboError\_StateChangeNotPossible** `const uint32_t RoboError_StateChangeNotPossible = ( (0xA0110000L) | 0x0013 ) [static]`

**11.96.4.37 RoboError\_Timeout** `const uint32_t RoboError_Timeout = ( (0xA0110000L) | 0x0001 ) [static]`

**11.96.4.38 RoboError\_UnknownCommand** `const uint32_t RoboError_UnknownCommand = ( (0xA0110000L) ) [static]`

## 11.96.5 Property Documentation

**11.96.5.1 McsBus** `CMcsBusNet^ McsBus [get]`

**11.96.5.2 McsBus\_MotorControl** `CMcsBus_MotorControlNet^ McsBus_MotorControl [get]`

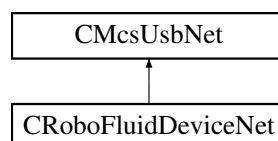
**11.96.5.3 RoboMainLowLevelCommand** `RoboMainLowLevelCommands^ RoboMainLowLevelCommand [get]`

## 11.96.6 Event Documentation

**11.96.6.1 RoboStatusEvent** `RoboStatusEventDelegate^ RoboStatusEvent`

## 11.97 CRoboFluidDeviceNet Class Reference

Inheritance diagram for CRoboFluidDeviceNet:



## Public Member Functions

- [CRoboFluidDeviceNet](#) (void)
- [~CRoboFluidDeviceNet](#) (void)
- void [SetValve](#) (int value)  
*Open or Close valves.*
- void [SetSingleValve](#) (int valve, bool onoff)  
*Opens or Closes a valve.*
- int [GetValve](#) ()  
*Query the state of the values.*
- bool [GetSingleValve](#) (int valve)  
*Query the state of a valve.*
- void [CloseAllValves](#) ()
- void [PumpOn](#) (int index, short speed)
- void [SetPumpSpeed](#) (int index, short speed)
- void [PumpOff](#) (int index)
- short [GetPumpSpeed](#) (int index)
- bool [IsPumpMotorOn](#) (int index)

## Protected Attributes

- CRoboFluidDevice \* [m\\_pRoboFluidDevice](#)
- [CMcsBus\\_MotorControlNet](#) ^ [m\\_pMcsBus\\_MotorControlNet](#)

## Properties

- [CMcsBus\\_MotorControlNet](#) ^ [McsBus\\_MotorControl](#) [get]

## Additional Inherited Members

### 11.97.1 Constructor & Destructor Documentation

**11.97.1.1 CRoboFluidDeviceNet()** [CRoboFluidDeviceNet](#) (  
void )

**11.97.1.2 ~CRoboFluidDeviceNet()** [~CRoboFluidDeviceNet](#) (  
void )

### 11.97.2 Member Function Documentation

**11.97.2.1 CloseAllValves()** `void CloseAllValves ( )`

**11.97.2.2 GetPumpSpeed()** `short GetPumpSpeed (`  
`int index )`

**11.97.2.3 GetSingleValve()** `bool GetSingleValve (`  
`int valve )`

Query the state of a valve.

#### Parameters

<i>valve</i>	number of valve /*!
--------------	---------------------

#### Returns

state of the valve

**11.97.2.4 GetValve()** `int GetValve ( )`

Query the state of the values.

#### Returns

the current state of the valves as a bit pattern.

**11.97.2.5 IsPumpMotorOn()** `bool IsPumpMotorOn (`  
`int index )`

**11.97.2.6 PumpOff()** `void PumpOff (`  
`int index )`

**11.97.2.7 PumpOn()** `void PumpOn (`  
`int index,`  
`short speed )`

**11.97.2.8 SetPumpSpeed()** `void SetPumpSpeed (`  
     `int index,`  
     `short speed )`

**11.97.2.9 SetSingleValve()** `void SetSingleValve (`  
     `int valve,`  
     `bool onoff )`

Opens or Closes a valve.

Parameters

<i>valve</i>	number of valve to be changed /*!
<i>onoff</i>	open or close the valve

**11.97.2.10 SetValve()** `void SetValve (`  
     `int value )`

Open or Close valves.

Parameters

<i>value</i>	bit pattern of valves which should be open.
--------------	---

### 11.97.3 Member Data Documentation

**11.97.3.1 m\_pMcsBus\_MotorControlNet** `CMcsBus_MotorControlNet ^ m_pMcsBus_MotorControlNet`  
 [protected]

**11.97.3.2 m\_pRoboFluidDevice** `CRoboFluidDevice* m_pRoboFluidDevice` [protected]

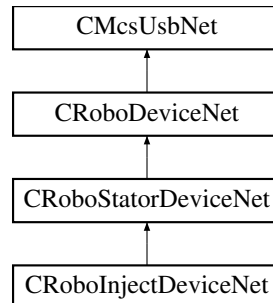
### 11.97.4 Property Documentation

**11.97.4.1 McsBus\_MotorControl** `CMcsBus_MotorControlNet^ McsBus_MotorControl` [get]

## 11.98 CRobolInjectDeviceNet Class Reference

[CRobolInjectDeviceNet](#) is the to control the MCS RobolInject device

Inheritance diagram for CRobolInjectDeviceNet:



### Public Member Functions

- [CRobolInjectDeviceNet](#) (void)

### Additional Inherited Members

#### 11.98.1 Detailed Description

[CRobolInjectDeviceNet](#) is the to control the MCS RobolInject device

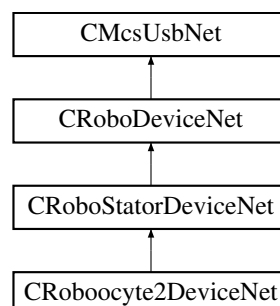
#### 11.98.2 Constructor & Destructor Documentation

**11.98.2.1 CRobolInjectDeviceNet()** [CRobolInjectDeviceNet](#) (  
void )

## 11.99 CRoboocyte2DeviceNet Class Reference

[CRoboocyte2DeviceNet](#) is the class to control the MCS Roboocyte2 device

Inheritance diagram for CRoboocyte2DeviceNet:





## Public Member Functions

- [CRobocyte2DeviceNet](#) (void)
- void [SetAxisLED](#) (bool onoff)
- bool [GetAxisLED](#) ()
- [CRoboDacqNet](#) ^ [GetRoboDacq](#) ()
- [CRoboFluidDeviceNet](#) ^ [GetRoboFluidDevice](#) ()
- [CGilsonDeviceNet](#) ^ [GetGilsonDevice](#) ()
- [CMcsBus\\_ExtensionNet](#) ^ [GetMcsBus\\_Extension](#) ()

## Additional Inherited Members

### 11.99.1 Detailed Description

[CRobocyte2DeviceNet](#) is the class to control the MCS Robocyte2 device

### 11.99.2 Constructor & Destructor Documentation

**11.99.2.1 CRobocyte2DeviceNet()** [CRobocyte2DeviceNet](#) (  
void )

### 11.99.3 Member Function Documentation

**11.99.3.1 GetAxisLED()** bool [GetAxisLED](#) ( )

**11.99.3.2 GetGilsonDevice()** [CGilsonDeviceNet](#) ^ [GetGilsonDevice](#) ( )

**11.99.3.3 GetMcsBus\_Extension()** [CMcsBus\\_ExtensionNet](#) ^ [GetMcsBus\\_Extension](#) ( )

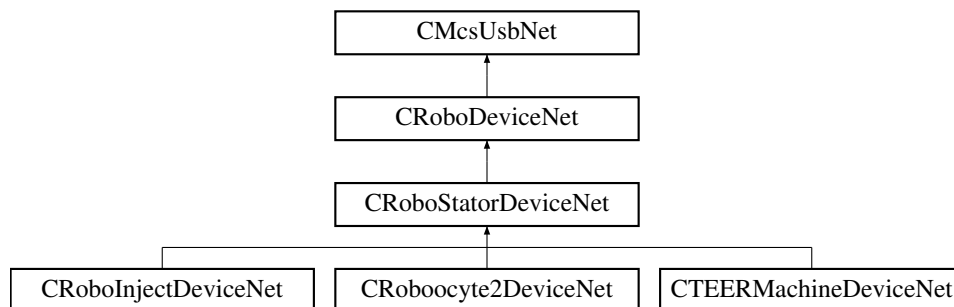
**11.99.3.4 GetRoboDacq()** [CRoboDacqNet](#) ^ [GetRoboDacq](#) ( )

**11.99.3.5 GetRoboFluidDevice()** [CRoboFluidDeviceNet](#) ^ GetRoboFluidDevice ( )

**11.99.3.6 SetAxisLED()** void SetAxisLED (   
 bool *onoff* )

## 11.100 CRoboStatorDeviceNet Class Reference

Inheritance diagram for CRoboStatorDeviceNet:



### Classes

- class [RoboMainStatorLowLevelCommands](#)

### Public Member Functions

- [CRoboStatorDeviceNet](#) (void)
- void [FindReferenceXY](#) ()
- void [FindReferenceXY](#) (int timeout)
- void [FindReferenceZ](#) ()
- void [FindReferenceZ](#) (int timeout)
- void [FindReferenceI](#) ()
- void [FindReferenceI](#) (int timeout)
- unsigned char [HasRefXY](#) ()
- unsigned char [HasRefZ](#) ()
- unsigned char [HasRefI](#) ()
- void [MoveAbsXY](#) (int x, int y)
- void [MoveAbsXY](#) (int x, int y, int timeout)
- void [MoveAbsZ](#) (int z)
- void [MoveAbsZ](#) (int z, int timeout)
- void [MoveAbsI](#) (int i)
- void [MoveAbsI](#) (int i, int timeout)
- void [StopMovementXY](#) ()
- void [StopMovementXY](#) (int timeout)
- void [StopMovementZ](#) ()
- void [StopMovementZ](#) (int timeout)
- void [StopMovementI](#) ()
- void [StopMovementI](#) (int timeout)
- void [SetCurrentAndAirXY](#) (unsigned short onoff)

- void [SetCurrentAndAirXY](#) (unsigned short onoff, int timeout)
- void [GetCurrentPositionXY](#) ([System::Runtime::InteropServices::Out]int% x, [System::Runtime::InteropServices::Out]int% y)
- void [GetCurrentPositionZ](#) ([System::Runtime::InteropServices::Out]int% z)
- void [GetCurrentPositionI](#) ([System::Runtime::InteropServices::Out]int% i)
- void [SetVelocityXY](#) (int v)
- void [SetVelocityZ](#) (int v)
- void [SetVelocityI](#) (int v)
- void [SetSpeedXY](#) (int v)
- void [SetSpeedZ](#) (int v)
- void [SetSpeedI](#) (int v)
- void [SetSpeedNativeXY](#) (int v)
- void [SetSpeedNativeZ](#) (int v)
- void [SetSpeedNativeI](#) (int v)
- void [SetAccelerationXY](#) (int a)
- void [SetAccelerationZ](#) (int a)
- void [SetAccelerationI](#) (int a)
- void [SetAccelerationNativeXY](#) (int a)
- void [SetAccelerationNativeZ](#) (int a)
- void [SetAccelerationNativeI](#) (int a)

## Properties

- [RoboMainStatorLowLevelCommands](#)<sup>^</sup> [RoboMainStatorLowLevelCommand](#) [get]

## Additional Inherited Members

### 11.100.1 Constructor & Destructor Documentation

**11.100.1.1 CRoboStatorDeviceNet()** [CRoboStatorDeviceNet](#) (   
 void )

### 11.100.2 Member Function Documentation

**11.100.2.1 FindReferenceI()** [1/2] void FindReferenceI ( )

**11.100.2.2 FindReferenceI()** [2/2] void FindReferenceI (   
 int timeout )

**11.100.2.3 FindReferenceXY()** [1/2] void FindReferenceXY ( )

**11.100.2.4 FindReferenceXY()** [2/2] void FindReferenceXY (   
int *timeout* )

**11.100.2.5 FindReferenceZ()** [1/2] void FindReferenceZ ( )

**11.100.2.6 FindReferenceZ()** [2/2] void FindReferenceZ (   
int *timeout* )

**11.100.2.7 GetCurrentPositionI()** void GetCurrentPositionI (   
[System::Runtime::InteropServices::Out] int% *i* )

**11.100.2.8 GetCurrentPositionXY()** void GetCurrentPositionXY (   
[System::Runtime::InteropServices::Out] int% *x*,   
[System::Runtime::InteropServices::Out] int% *y* )

**11.100.2.9 GetCurrentPositionZ()** void GetCurrentPositionZ (   
[System::Runtime::InteropServices::Out] int% *z* )

**11.100.2.10 HasRefI()** unsigned char HasRefI ( )

**11.100.2.11 HasRefXY()** unsigned char HasRefXY ( )

**11.100.2.12 HasRefZ()** unsigned char HasRefZ ( )

**11.100.2.13 MoveAbsI()** [1/2] void MoveAbsI (  
int *i* )

**11.100.2.14 MoveAbsI()** [2/2] void MoveAbsI (  
int *i*,  
int *timeout* )

**11.100.2.15 MoveAbsXY()** [1/2] void MoveAbsXY (  
int *x*,  
int *y* )

**11.100.2.16 MoveAbsXY()** [2/2] void MoveAbsXY (  
int *x*,  
int *y*,  
int *timeout* )

**11.100.2.17 MoveAbsZ()** [1/2] void MoveAbsZ (  
int *z* )

**11.100.2.18 MoveAbsZ()** [2/2] void MoveAbsZ (  
int *z*,  
int *timeout* )

**11.100.2.19 SetAccelerationI()** void SetAccelerationI (  
int *a* )

**11.100.2.20 SetAccelerationNativeI()** void SetAccelerationNativeI (  
int *a* )

**11.100.2.21 SetAccelerationNativeXY()** void SetAccelerationNativeXY (  
int *a* )

**11.100.2.22 SetAccelerationNativeZ()** `void SetAccelerationNativeZ (`  
    `int a )`

**11.100.2.23 SetAccelerationXY()** `void SetAccelerationXY (`  
    `int a )`

**11.100.2.24 SetAccelerationZ()** `void SetAccelerationZ (`  
    `int a )`

**11.100.2.25 SetCurrentAndAirXY() [1/2]** `void SetCurrentAndAirXY (`  
    `unsigned short onoff )`

**11.100.2.26 SetCurrentAndAirXY() [2/2]** `void SetCurrentAndAirXY (`  
    `unsigned short onoff,`  
    `int timeout )`

**11.100.2.27 SetSpeedI()** `void SetSpeedI (`  
    `int v )`

**11.100.2.28 SetSpeedNativeI()** `void SetSpeedNativeI (`  
    `int v )`

**11.100.2.29 SetSpeedNativeXY()** `void SetSpeedNativeXY (`  
    `int v )`

**11.100.2.30 SetSpeedNativeZ()** `void SetSpeedNativeZ (`  
    `int v )`

**11.100.2.31 SetSpeedXY()** void SetSpeedXY (   
int v )

**11.100.2.32 SetSpeedZ()** void SetSpeedZ (   
int v )

**11.100.2.33 SetVelocityI()** void SetVelocityI (   
int v )

**11.100.2.34 SetVelocityXY()** void SetVelocityXY (   
int v )

**11.100.2.35 SetVelocityZ()** void SetVelocityZ (   
int v )

**11.100.2.36 StopMovementI()** [1/2] void StopMovementI ( )

**11.100.2.37 StopMovementI()** [2/2] void StopMovementI (   
int timeout )

**11.100.2.38 StopMovementXY()** [1/2] void StopMovementXY ( )

**11.100.2.39 StopMovementXY()** [2/2] void StopMovementXY (   
int timeout )

**11.100.2.40 StopMovementZ()** [1/2] void StopMovementZ ( )

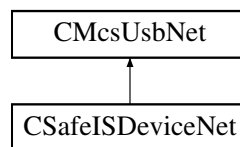
**11.100.2.41 StopMovementZ()** [2/2] void StopMovementZ (  
int timeout )

### 11.100.3 Property Documentation

**11.100.3.1 RoboMainStatorLowLevelCommand** [RoboMainStatorLowLevelCommands](#)<sup>^</sup> RoboMainStator↔  
LowLevelCommand [get]

## 11.101 CSafeISDeviceNet Class Reference

Inheritance diagram for CSafeISDeviceNet:



### Public Member Functions

- [CSafeISDeviceNet](#) (void)  
*Initializes a new instance of the [CSafeISDeviceNet](#) class.*
- [~CSafeISDeviceNet](#) (void)  
*Releases unmanaged resources and performs other cleanup operations before the [CSafeISDeviceNet](#) is reclaimed by garbage collection.*
- void [SetSwitches](#) (unsigned short switches)  
*Sets the switches for all electrodes on the device. Do not use during measurement*
- void [SetAdcChannels](#) (unsigned char channels)  
*Sets the ADC channels you want to be sampled*
- void [SetAdcSamplePos](#) (array< unsigned short >^ positions)  
*Sets the sample position of the ADC.*
- void [SetDacMode](#) (unsigned char mode)  
*Sets the DAC mode.*
- void [SetDacPulseform](#) (array< short >^ pulseform)  
*Sets the DAC pulseform.*
- void [SetDacPeriode](#) (unsigned int periode)  
*Sets the DAC periode.*

### Properties

- [CRoboDeviceNet](#)<sup>^</sup> [RoboDevice](#) [get]  
*Gets the [CRoboDeviceNet](#). Use this to control the syringe.*
- [CFluidControlDeviceNet](#)<sup>^</sup> [FluidControlDevice](#) [get]  
*Gets the [CFluidControlDeviceNet](#). Use this to control the valves. Only SetSingleValve is implemented for [CSafeISDeviceNet](#).*
- [CMcsUsbDacqNet](#)<sup>^</sup> [DacqDevice](#) [get]  
*Gets the [CMcsUsbDacqNet](#). Use this to control the data aquisition.*



## Additional Inherited Members

### 11.101.1 Detailed Description

### 11.101.2 Constructor & Destructor Documentation

#### 11.101.2.1 CSafeISDeviceNet() CSafeISDeviceNet ( void )

Initializes a new instance of the [CSafeISDeviceNet](#) class.

#### 11.101.2.2 ~CSafeISDeviceNet() ~CSafeISDeviceNet ( void )

Releases unmanaged resources and performs other cleanup operations before the [CSafeISDeviceNet](#) is reclaimed by garbage collection.

### 11.101.3 Member Function Documentation

#### 11.101.3.1 SetAdcChannels() void SetAdcChannels ( unsigned char channels )

Sets the ADC channels you want to be sampled

##### Parameters

<i>channels</i>	The bitmap of the 8 channels. Set bit to 1 for the channels you want measure
-----------------	--

#### 11.101.3.2 SetAdcSamplePos() void SetAdcSamplePos ( array< unsigned short >^ positions )

Sets the sample position of the ADC.

##### Parameters

<i>positions</i>	The positions in units of 0.1μs.
------------------	----------------------------------

**11.101.3.3 SetDacMode()** `void SetDacMode (`  
`unsigned char mode )`

Sets the DAC mode.

Parameters

<i>mode</i>	The mode: 0 = Impedance ; 1 = Amperometry
-------------	---

**11.101.3.4 SetDacPeriode()** `void SetDacPeriode (`  
`unsigned int periode )`

Sets the DAC periode.

Parameters

<i>periode</i>	The periode in units of 10µs.
----------------	-------------------------------

**11.101.3.5 SetDacPulseform()** `void SetDacPulseform (`  
`array< short >^ pulseform )`

Sets the DAC pulseform.

Parameters

<i>pulseform</i>	The pulseform.
------------------	----------------

**11.101.3.6 SetSwitches()** `void SetSwitches (`  
`unsigned short switches )`

Sets the switches for all electrodes on the device. Do not use during measurement

Parameters

<i>switches</i>	The switches: See Schematics for the meaning
-----------------	--

## 11.101.4 Property Documentation

**11.101.4.1 DacqDevice** `CMcsUsbDacqNet^ DacqDevice [get]`

Gets the `CMcsUsbDacqNet`. Use this to control the data acquisition.

**11.101.4.2 FluidControlDevice** `CFluidControlDeviceNet^ FluidControlDevice [get]`

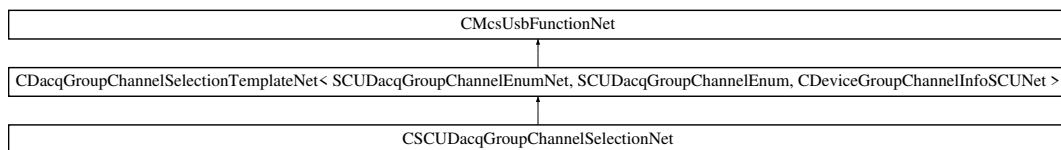
Gets the `CFluidControlDeviceNet`. Use this to control the valves. Only `SetSingleValve` is implemented for `CSafeISDeviceNet`.

**11.101.4.3 RoboDevice** `CRoboDeviceNet^ RoboDevice [get]`

Gets the `CRoboDeviceNet`. Use this to control the syringe.

**11.102 CSCUDacqGroupChannelSelectionNet Class Reference**

Inheritance diagram for `CSCUDacqGroupChannelSelectionNet`:

**Public Member Functions**

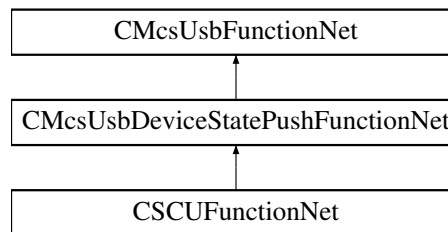
- `CSCUDacqGroupChannelSelectionNet (CMcsUsbNet^ mcsusb)`

**Additional Inherited Members****11.102.1 Constructor & Destructor Documentation****11.102.1.1 CSCUDacqGroupChannelSelectionNet()** `CSCUDacqGroupChannelSelectionNet ( CMcsUsbNet^ mcsusb )`

## 11.103 CSCUFunctionNet Class Reference

[CSCUFunctionNet](#) is the class to control the SCU device

Inheritance diagram for CSCUFunctionNet:



### Public Member Functions

- delegate void [OnGetAvailableHeadstages](#) (uint32\_t AvailableHeadstages)
- delegate void [OnIsHeadstageAvailable](#) (uint32\_t Headstage, bool available)
- [CSCUFunctionNet](#) (CMcsUsbNet^ mcsusb, [CMcsUsbFunctionPointerContainer](#)^ pSCUFunctionPointerContainer)
  - Initializes a new instance of the [CSCUFunctionNet](#) class.*
- [CSCUFunctionNet](#) (CMcsUsbNet^ mcsusb)
- virtual [~CSCUFunctionNet](#) ()
- [ICSCUFunctionNet](#) ()
- uint32\_t [GetAvailableHeadstages](#) ()
  - Gets a bitmap of available headstages.*
- bool [IsInDacqLegacyMode](#) ()
  - Is the SCU in legacy mode*
- void [SetDacqLegacyMode](#) (bool enable)
  - Enable the SCU legacy mode*
- uint32\_t [GetMaxStimulusChannelsPerHeadstage](#) ()
  - Gets the maximal number of stimulation channels a headstage can have.*
- uint32\_t [GetMaxNumberOfHeadstages](#) ()
  - Gets the maximal number of headstages.*
- SCU\_HeadstageIdEnumNet [GetHeadstageID](#) (uint32\_t Headstage)
  - Gets the headstage fpga ID.*
- bool [IsHeadstageAvailable](#) (uint32\_t Headstage)
  - Checks whether the given headstage is available.*
- void [PowerHS](#) (uint32\_t Headstage, bool power)
  - Power the HS*
- bool [IsHSPowered](#) (uint32\_t Headstage)
  - Is the HS powered*
- bool [HasHSPowerSwitch](#) ()
  - Has SCU HS power switch*
- String ^ [GetHeadstageSerialNumber](#) (uint32\_t Headstage)
  - Gets the serial number of a given headstage.*
- uint32\_t [GetHeadstageNumberOfAnalogChannels](#) (uint32\_t Headstage)
  - Gets the number of analog channels for a given headstage.*
- uint32\_t [GetHeadstageNumberOfStimulationChannels](#) (uint32\_t Headstage)
  - Gets the number of stimulation channels for a given headstage.*
- uint32\_t [GetHeadstageGainInPer mille](#) (uint32\_t Headstage)

- Gets the gain factor in permille for a given headstage.*
- uint32\_t [GetHeadstageAdcRangeInMicroVolt](#) (uint32\_t Headstage)
  - Gets the ADC Range in uV for a given headstage.*
- uint32\_t [GetHeadstageAdcBits](#) (uint32\_t Headstage)
  - Gets the Number of ADC bits for a given headstage.*
- uint32\_t [GetHeadstageDacVoltageRangeInMilliVolt](#) (uint32\_t Headstage)
  - Gets the DAC Voltage Range in mV for a given headstage.*
- uint32\_t [GetHeadstageDacVoltageResolutionInMicroVolt](#) (uint32\_t Headstage)
  - Gets the DAC Voltage Resolution in uV for a given headstage.*
- uint32\_t [GetHeadstageDacCurrentRangeInMicroAmpere](#) (uint32\_t Headstage)
  - Gets the DAC Current Range in uA for a given headstage.*
- uint32\_t [GetHeadstageDacCurrentResolutionInNanoAmpere](#) (uint32\_t Headstage)
  - Gets the DAC Current Resolution in nA for a given headstage.*
- uint32\_t [GetHeadstageDacBits](#) (uint32\_t Headstage)
  - Gets the Number of DAC bits for a given headstage.*
- uint32\_t [GetHeadstageSamplerate](#) (uint32\_t Headstage)
  - Gets the Samplerate of a given headstage.*
- bool [HasGalvanicIsolation](#) ()
  - Has galvanic isolated hardware*
- bool [HasAnalogOut](#) ()
  - Has AnalogOut hardware*
- void [EnableAnalogOut](#) (bool enable)
  - Enables AnalogOut globally*
- bool [IsAnalogOutEnabled](#) ()
  - Is AnalogOut enabled*
- void [SetAnalogOutDACRange](#) (AnalogOut\_DAC\_Range\_EnumNet range)
  - Sets the analog out DAC range*
- AnalogOut\_DAC\_Range\_EnumNet [GetAnalogOutDACRange](#) ()
  - Gets the analog out DAC range*
- void [SetAnalogOutADCRange](#) (uint32\_t range)
  - Sets the analog out ADC range*
- uint32\_t [GetAnalogOutADCRange](#) ()
  - Gets the analog out ADC range*
- void [AutomaticAnalogOut](#) (bool automatic)
  - Sets automatic source channel selection*
- bool [IsAutomaticAnalogOut](#) ()
  - Is Automatic source channel selection selected*
- void [SetAnalogOutChannels](#) (uint32\_t out\_channel, uint32\_t source\_channel)
  - Set the source channel number for a certain output channel*
- uint32\_t [GetAnalogOutChannels](#) (uint32\_t out\_channel)
  - Get the connected source channel number for a certain output channel*
- void [SetReferenceElectrodeSwitchState](#) (uint32\_t Headstage, ReferenceElectrodeSwitchPositionEnumNet NewSwitchPos)
  - Sets the position of the switch for the reference electrode*
- ReferenceElectrodeSwitchPositionEnumNet [GetReferenceElectrodeSwitchState](#) (uint32\_t Headstage)
  - Gets the position of the switch for the reference electrode*
- void [SetReferenceElectrodeMode](#) (uint32\_t Headstage, ReferenceElectrodeModeEnumNet NewValue)
  - Sets the mode for the reference electrode*
- ReferenceElectrodeModeEnumNet [GetReferenceElectrodeMode](#) (uint32\_t Headstage)
  - Gets the mode for the reference electrode*
- [CFilterPropertyNet](#) ^ [GetFilterProperty](#) (SCUDacqGroupChannelEnumNet GroupID, uint32\_t FilterNumber)

*Gets the filter property*

- array< [CFilterPropertyNet](#)> ^ [GetFilterProperties](#) (SCUDacqGroupChannelEnumNet GroupID, int filter← Configurations\_Length)

*Gets multiple filter properties*

## Events

- [OnGetAvailableHeadstages](#) ^ [GetAvailableHeadstagesEvent](#) [add, remove, raise]  
*Event fires when the bitmap of available headstages has changed*
- [OnIsHeadstageAvailable](#) ^ [IsHeadstageAvailableEvent](#) [add, remove, raise]  
*Event fires when 'true' if the headstage is connected for the headstage to query has changed*

## Additional Inherited Members

### 11.103.1 Detailed Description

[CSCUFunctionNet](#) is the class to control the SCU device

### 11.103.2 Constructor & Destructor Documentation

**11.103.2.1 [CSCUFunctionNet\(\)](#) [1/2]** [CSCUFunctionNet](#) (   
 [CMcsUsbNet](#) ^ *mcsusb*,  
 [CMcsUsbFunctionPointerContainer](#) ^ *pSCUFunctionPointerContainer* )

Initializes a new instance of the [CSCUFunctionNet](#) class.

**11.103.2.2 [CSCUFunctionNet\(\)](#) [2/2]** [CSCUFunctionNet](#) (   
 [CMcsUsbNet](#) ^ *mcsusb* )

**11.103.2.3 [~CSCUFunctionNet\(\)](#)** virtual [~CSCUFunctionNet](#) ( ) [virtual]

**11.103.2.4 ["!CSCUFunctionNet\(\)](#)** [!CSCUFunctionNet](#) ( )

### 11.103.3 Member Function Documentation

**11.103.3.1 [AutomaticAnalogOut\(\)](#)** void [AutomaticAnalogOut](#) (   
 bool *automatic* )

Sets automatic source channel selection

## Parameters

<i>automatic</i>	Automatic
------------------	-----------

**11.103.3.2 EnableAnalogOut()** `void EnableAnalogOut (`  
    `bool enable )`

Enables AnalogOut globally

## Parameters

<i>enable</i>	Enable
---------------	--------

**11.103.3.3 GetAnalogOutADCRange()** `uint32_t GetAnalogOutADCRange ( )`

Gets the analog out ADC range

## Returns

Range

**11.103.3.4 GetAnalogOutChannels()** `uint32_t GetAnalogOutChannels (`  
    `uint32_t out_channel )`

Get the connected source channel number for a certain output channel

## Parameters

<i>out_channel</i>	Output channel number
--------------------	-----------------------

## Returns

Source channel number

**11.103.3.5 GetAnalogOutDACRange()** `AnalogOut_DAC_Range_EnumNet GetAnalogOutDACRange ( )`

Gets the analog out DAC range

## Returns

Range

**11.103.3.6 GetAvailableHeadstages()** `uint32_t GetAvailableHeadstages ( )`

Gets a bitmap of available headstages.

**Returns**

The bitmap of available headstages.

**11.103.3.7 GetFilterProperties()** `array<CFilterPropertyNet^> ^ GetFilterProperties ( SCUDacqGroupChannelEnumNet GroupID, int filterConfigurations_Length )`

Gets multiple filter properties

**Parameters**

<i>GroupID</i>	The group ID
<i>filterConfigurations_Length</i>	The maximal length of filterConfigurations.

**Returns**

array of filter properties

**11.103.3.8 GetFilterProperty()** `CFilterPropertyNet ^ GetFilterProperty ( SCUDacqGroupChannelEnumNet GroupID, uint32_t FilterNumber )`

Gets the filter property

**Parameters**

<i>GroupID</i>	The group ID
<i>FilterNumber</i>	The filter number

**Returns**

The filter property

**11.103.3.9 GetHeadstageAdcBits()** `uint32_t GetHeadstageAdcBits ( uint32_t Headstage )`

Gets the Number of ADC bits for a given headstage.



**Parameters**

<i>Headstage</i>	The headstage to query.
------------------	-------------------------

**Returns**

The number of bits the ADC has for the given headstage.

**11.103.3.10 GetHeadstageAdcRangeInMicroVolt()** `uint32_t GetHeadstageAdcRangeInMicroVolt (uint32_t Headstage )`

Gets the ADC Range in uV for a given headstage.

**Parameters**

<i>Headstage</i>	The headstage to query.
------------------	-------------------------

**Returns**

The ADC Range in uV for the given headstage.

**11.103.3.11 GetHeadstageDacBits()** `uint32_t GetHeadstageDacBits (uint32_t Headstage )`

Gets the Number of DAC bits for a given headstage.

**Parameters**

<i>Headstage</i>	The headstage to query.
------------------	-------------------------

**Returns**

The number of bits the DAC has for the given headstage.

**11.103.3.12 GetHeadstageDacCurrentRangeInMicroAmpere()** `uint32_t GetHeadstageDacCurrentRangeInMicroAmpere (uint32_t Headstage )`

Gets the DAC Current Range in uA for a given headstage.

**Parameters**

<i>Headstage</i>	The headstage to query.
------------------	-------------------------

**Returns**

The DAC Current Range in uA for the given headstage.

**11.103.3.13 GetHeadstageDacCurrentResolutionInNanoAmpere()** `uint32_t GetHeadstageDacCurrentResolutionInNanoAmpere (uint32_t Headstage )`

Gets the DAC Current Resolution in nA for a given headstage.

**Parameters**

<i>Headstage</i>	The headstage to query.
------------------	-------------------------

**Returns**

The DAC Current Resolution in nA for the given headstage.

**11.103.3.14 GetHeadstageDacVoltageRangeInMilliVolt()** `uint32_t GetHeadstageDacVoltageRangeInMilliVolt (uint32_t Headstage )`

Gets the DAC Voltage Range in mV for a given headstage.

**Parameters**

<i>Headstage</i>	The headstage to query.
------------------	-------------------------

**Returns**

The DAC Voltage Range in mV for the given headstage.

**11.103.3.15 GetHeadstageDacVoltageResolutionInMicroVolt()** `uint32_t GetHeadstageDacVoltageResolutionInMicroVolt (uint32_t Headstage )`

Gets the DAC Voltage Resolution in uV for a given headstage.

**Parameters**

<i>Headstage</i>	The headstage to query.
------------------	-------------------------

**Returns**

The DAC Voltage Resolution in uV for the given headstage.

**11.103.3.16 GetHeadstageGainInPermille()** `uint32_t GetHeadstageGainInPermille (uint32_t Headstage )`

Gets the gain factor in permille for a given headstage.

**Parameters**

<i>Headstage</i>	The headstage to query.
------------------	-------------------------

**Returns**

The gain factor in permille for the given headstage.

**11.103.3.17 GetHeadstageID()** `SCU_HeadstageIdEnumNet GetHeadstageID (uint32_t Headstage )`

Gets the headstage fpga ID.

**Parameters**

<i>Headstage</i>	The headstage to query.
------------------	-------------------------

**Returns**

The headstage fpga ID.

**11.103.3.18 GetHeadstageNumberOfAnalogChannels()** `uint32_t GetHeadstageNumberOfAnalogChannels (uint32_t Headstage )`

Gets the number of analog channels for a given headstage.

**Parameters**

<i>Headstage</i>	The headstage to query.
------------------	-------------------------

**Returns**

The number of analog channels the headstage has.

**11.103.3.19 GetHeadstageNumberOfStimulationChannels()** `uint32_t GetHeadstageNumberOfStimulationChannels ( uint32_t Headstage )`

Gets the number of stimulation channels for a given headstage.

**Parameters**

<i>Headstage</i>	The headstage to query.
------------------	-------------------------

**Returns**

The number of stimulation channels the headstage has.

**11.103.3.20 GetHeadstageSamplerate()** `uint32_t GetHeadstageSamplerate ( uint32_t Headstage )`

Gets the Samplerate of a given headstage.

**Parameters**

<i>Headstage</i>	The headstage to query.
------------------	-------------------------

**Returns**

The samplerate in Hz for the given headstage.

**11.103.3.21 GetHeadstageSerialNumber()** `String ^ GetHeadstageSerialNumber ( uint32_t Headstage )`

Gets the serial number of a given headstage.

**Parameters**

<i>Headstage</i>	The headstage to query.
------------------	-------------------------

**Returns**

The serial number of the headstage.

**11.103.3.22 GetMaxNumberOfHeadstages()** `uint32_t GetMaxNumberOfHeadstages ( )`

Gets the maximal number of headstages.

**Returns**

The maximal number of headstages.

**11.103.3.23 GetMaxStimulusChannelsPerHeadstage()** `uint32_t GetMaxStimulusChannelsPerHeadstage ( )`

Gets the maximal number of stimulation channels a headstage can have.

**Returns**

The maximal number of stimulation channels a headstage can have.

**11.103.3.24 GetReferenceElectrodeMode()** `ReferenceElectrodeModeEnumNet GetReferenceElectrodeMode ( uint32_t Headstage )`

Gets the mode for the reference electrode

**Parameters**

<i>Headstage</i>	The headstage number
------------------	----------------------

**Returns**

The mode

**11.103.3.25 GetReferenceElectrodeSwitchState()** `ReferenceElectrodeSwitchPositionEnumNet Get↔  
ReferenceElectrodeSwitchState (   
    uint32_t Headstage )`

Gets the position of the switch for the reference electrode

#### Parameters

<i>Headstage</i>	The headstage number
------------------	----------------------

#### Returns

The switch position

**11.103.3.26 HasAnalogOut()** `bool HasAnalogOut ( )`

Has AnalogOut hardware

#### Returns

Enabled

**11.103.3.27 HasGalvanicIsolation()** `bool HasGalvanicIsolation ( )`

Has galvanic isolated hardware

#### Returns

Enabled

**11.103.3.28 HasHSPowerSwitch()** `bool HasHSPowerSwitch ( )`

Has SCU HS power switch

#### Returns

Has Switch

**11.103.3.29 IsAnalogOutEnabled()** `bool IsAnalogOutEnabled ( )`

Is AnalogOut enabled

**Returns**

Enabled

**11.103.3.30 IsAutomaticAnalogOut()** `bool IsAutomaticAnalogOut ( )`

Is Automatic source channel selection selected

**Returns**

Automatic

**11.103.3.31 IsHeadstageAvailable()** `bool IsHeadstageAvailable (   
uint32_t Headstage )`

Checks whether the given headstage is available.

**Parameters**

<i>Headstage</i>	The headstage to query.
------------------	-------------------------

**Returns**

'true' if the headstage is connected.

**11.103.3.32 IsHSPowered()** `bool IsHSPowered (   
uint32_t Headstage )`

Is the HS powered

**Parameters**

<i>Headstage</i>	The headstage to query.
------------------	-------------------------

**Returns**

'true' if the headstage is powered.

**11.103.3.33 IsInDacqLegacyMode()** `bool IsInDacqLegacyMode ( )`

Is the SCU in legacy mode

**Returns**

Is Enabled

**11.103.3.34 OnGetAvailableHeadstages()** `delegate void OnGetAvailableHeadstages (   
uint32_t AvailableHeadstages )`**11.103.3.35 OnIsHeadstageAvailable()** `delegate void OnIsHeadstageAvailable (   
uint32_t Headstage,   
bool available )`**11.103.3.36 PowerHS()** `void PowerHS (   
uint32_t Headstage,   
bool power )`

Power the HS

**Parameters**

<i>Headstage</i>	The headstage to query.
<i>power</i>	'true' if the headstage is powered.

**11.103.3.37 SetAnalogOutADCRange()** `void SetAnalogOutADCRange (   
uint32_t range )`

Sets the analog out ADC range

**Parameters**

<i>range</i>	Range
--------------	-------

**11.103.3.38 SetAnalogOutChannels()** `void SetAnalogOutChannels (   
uint32_t out_channel,   
uint32_t source_channel )`



Set the source channel number for a certain output channel

## Parameters

<i>out_channel</i>	Output channel number
<i>source_channel</i>	Source channel number

**11.103.3.39 SetAnalogOutDACRange()** `void SetAnalogOutDACRange ( AnalogOut_DAC_Range_EnumNet range )`

Sets the analog out DAC range

## Parameters

<i>range</i>	Range
--------------	-------

**11.103.3.40 SetDacqLegacyMode()** `void SetDacqLegacyMode ( bool enable )`

Enable the SCU legacy mode

## Parameters

<i>enable</i>	Enable
---------------	--------

**11.103.3.41 SetReferenceElectrodeMode()** `void SetReferenceElectrodeMode ( uint32_t Headstage, ReferenceElectrodeModeEnumNet NewValue )`

Sets the mode for the reference electrode

## Parameters

<i>Headstage</i>	The headstage number
<i>NewValue</i>	The mode

**11.103.3.42 SetReferenceElectrodeSwitchState()** `void SetReferenceElectrodeSwitchState ( uint32_t Headstage, ReferenceElectrodeSwitchPositionEnumNet NewSwitchPos )`

Sets the position of the switch for the reference electrode

## Parameters

<i>Headstage</i>	The headstage number
<i>NewSwitchPos</i>	The switch position

## 11.103.4 Event Documentation

**11.103.4.1 GetAvailableHeadstagesEvent** [OnGetAvailableHeadstages](#)<sup>^</sup> [GetAvailableHeadstagesEvent](#) [add], [remove], [raise]

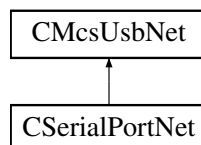
Event fires when the bitmap of available headstages has changed

**11.103.4.2 IsHeadstageAvailableEvent** [OnIsHeadstageAvailable](#)<sup>^</sup> [IsHeadstageAvailableEvent](#) [add], [remove], [raise]

Event fires when 'true' if the headstage is connected for the headstage to query has changed

## 11.104 CSerialPortNet Class Reference

Inheritance diagram for CSerialPortNet:



## Public Member Functions

- [CSerialPortNet](#) (void)
- void [Send](#) (array< byte > ^ buffer)
- void [Send](#) (String ^ command)
- array< byte > ^ [Receive](#) (void)
- array< byte > ^ [Receive](#) (int length)
- String ^ [ReceiveString](#) (void)
- String ^ [ReceiveString](#) (int length)
- int [GetBytesAvailable](#) (void)

## Additional Inherited Members

## 11.104.1 Constructor &amp; Destructor Documentation

**11.104.1.1 CSerialPortNet()** `CSerialPortNet (`  
`void )`

## 11.104.2 Member Function Documentation

**11.104.2.1 GetBytesAvailable()** `int GetBytesAvailable (`  
`void )`

**11.104.2.2 Receive()** [1/2] `array<byte> ^ Receive (`  
`int length )`

**11.104.2.3 Receive()** [2/2] `array<byte> ^ Receive (`  
`void )`

**11.104.2.4 ReceiveString()** [1/2] `String ^ ReceiveString (`  
`int length )`

**11.104.2.5 ReceiveString()** [2/2] `String ^ ReceiveString (`  
`void )`

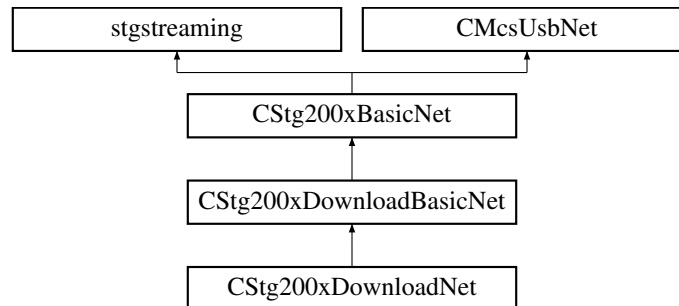
**11.104.2.6 Send()** [1/2] `void Send (`  
`array< byte > ^ buffer )`

**11.104.2.7 Send()** [2/2] `void Send (`  
`String ^ command )`

## 11.105 CStg200xBasicNet Class Reference

Base class for the Stg200x.

Inheritance diagram for CStg200xBasicNet:



### Public Member Functions

- virtual [~CStg200xBasicNet](#) ()  
*The destructor.*
- void [SetOutputRate](#) (uint32\_t rate)  
*Change the output rate of the STG. Valid rates are from 1000 Hz to 50000 Hz.*
- uint32\_t [GetOutputRate](#) ()  
*Queries the output rate of the STG. Valid rates are from 1000 Hz to 50000 Hz.*
- void [SendStart](#) (uint32\_t triggermap)  
*Start (Trigger) the STG. The startup delay is in the range of a few ms.*
- void [SendStop](#) (uint32\_t triggermap)  
*Stop some or all triggers of the STG.*
- void [SendStop](#) (uint32\_t triggermap, int options)  
*Stop some or all triggers of the STG.*
- void [GetStgVersionInfo](#) ([Out]String^% SwVersion, [Out]String^% HwVersion)  
*Queries software and hardware version.*
- void [GetAnalogRanges](#) (int channel, [Out]int% URange, [Out]int% IRange)  
*Gets the range of the analog outputs.*
- void [GetAnalogResolution](#) (int channel, [Out]int% URes, [Out]int% IRes)  
*Gets the resolution of the analog outputs.*
- virtual int32\_t [GetDACResolution](#) ()  
*Gets number of bits of the DAC resolution.*
- virtual int32\_t [GetVoltageRangeInMicroVolt](#) (uint32\_t channel)  
*Gets the Voltage Range of the specified channel in Microvolts.*
- virtual int32\_t [GetVoltageResolutionInMicroVolt](#) (uint32\_t channel)  
*Gets the Voltage Resolution of the specified channel in Microvolts.*
- virtual int32\_t [GetCurrentRangeInNanoAmp](#) (uint32\_t channel)  
*Gets the Current Range of the specified channel in Nanoamps.*
- virtual int32\_t [GetCurrentResolutionInNanoAmp](#) (uint32\_t channel)  
*Gets the Current Resolution of the specified channel in Nanoamps.*
- void [GetStgProgramInfo](#) ([Out]bool% IsProgrammed, [Out]System::Runtime::InteropServices::ComTypes::FILETIME% timestamp, [Out]String^% filename, [Out]Guid% guid)  
*Queries Download information from the STG. If download information was stored by the use of [SetStgProgramInfo](#), this function can be used to retrieve it.*

- void [GetStgProgramInfo](#) ([Out]bool% IsProgrammed, [Out]DateTime% timestamp, [Out]String^% filename, [Out]Guid% guid)  
*Queries Download information from the STG. If download information was stored by the use of [SetStgProgramInfo](#), this function can be used to retrieve it.*
- void [SetStgProgramInfo](#) (DateTime timestamp, String^ filename, Guid guid)  
*Store Download information in the STG. This function can be used to store the filename and timestamp of the last download for later query.*
- uint32\_t [GetAvailableMemory](#) ()  
*Gets the amount of memory available in the currently selected segment of the STG.*
- uint32\_t [GetTotalMemory](#) ()  
*Gets the total amount of memory available on the STG (all segments).*
- virtual uint32\_t [GetNumberOfAnalogChannels](#) ()  
*Gets the Number of available analog channels of the device.*
- virtual uint32\_t [GetNumberOfSyncoutChannels](#) ()  
*Gets the Number of available syncout channels of the device.*
- virtual uint32\_t [GetNumberOfTriggerInputs](#) ()  
*Gets the Number of trigger inputs of the device.*
- virtual uint32\_t [GetNumberOfHWDACPaths](#) ()  
*Gets the Number of HW Stimulation DACs of the device.*
- virtual uint32\_t [GetNumberOfStimulationSourcesPerElectrode](#) ()  
*Gets the number of stimulation sources (DACs) per electrode.*
- virtual void [SetVoltageMode](#) (unsigned int channel)  
*Sets a channel to voltage mode (STG3008-FA and STG400x only).*
- virtual void [SetCurrentMode](#) (unsigned int channel)  
*Sets a channel to current mode (STG3008-FA and STG400x only).*
- virtual void [SetVoltageMode](#) ()  
*Sets all channels to voltage mode (STG3008-FA and STG400x only).*
- virtual void [SetCurrentMode](#) ()  
*Sets all channels to current mode (STG3008-FA and STG400x only).*
- virtual void [SetMeasurementMode](#) (unsigned int channel)  
*Sets a channel to measurement mode (STG3008-FA).*
- virtual void [SetFAAmplification](#) (unsigned int amplification)
- virtual uint32\_t [GetFAAmplification](#) ()
- virtual void [SetAutocalibrationDisabled](#) (unsigned int channel, bool disable)  
*Sets the autocalibration configuration.*
- virtual bool [GetAutocalibrationDisabled](#) (unsigned int channel)  
*Gets the autocalibration configuration.*
- virtual void [SetElectrodeMode](#) (uint32\_t electrode, array< ElectrodeModeEnumNet >^ mode)  
*Puts an electrode in either automatic or manual mode.*
- virtual void [SetElectrodeMode](#) (uint32\_t electrode, ElectrodeModeEnumNet mode)  
*Puts an electrode in either automatic or manual mode.*
- virtual void [SetElectrodeMode](#) (uint32\_t Scu\_HS, uint32\_t electrode, array< ElectrodeModeEnumNet >^ mode)  
*Puts an electrode in either automatic or manual mode.*
- virtual void [SetElectrodeMode](#) (uint32\_t Scu\_HS, uint32\_t electrode, ElectrodeModeEnumNet mode)  
*Puts an electrode in either automatic or manual mode.*
- virtual uint32\_t [GetElectrodeMode](#) (uint32\_t electrode)  
*Gets the mode an electrode is in.*
- virtual uint32\_t [GetElectrodeMode](#) (uint32\_t Scu\_HS, uint32\_t electrode)  
*Gets the mode an electrode is in.*
- virtual void [SetElectrodeDacMux](#) (uint32\_t electrode, uint32\_t listModelIndex, array< ElectrodeDacMuxEnumNet >^ dacMux)

*Defines the DAC to use for an electrode.*

- virtual void [SetElectrodeDacMux](#) (uint32\_t electrode, uint32\_t listmodelIndex, ElectrodeDacMuxEnumNet dacMux)

*Defines the DAC to use for an electrode.*

- virtual void [SetElectrodeDacMux](#) (uint32\_t Scu\_HS, uint32\_t electrode, uint32\_t listmodelIndex, ElectrodeDacMuxEnumNet dacMux)

*Defines the DAC to use for an electrode.*

- virtual void [SetElectrodeDacMux](#) (uint32\_t Scu\_HS, uint32\_t electrode, uint32\_t listmodelIndex, array< ElectrodeDacMuxEnumNet >^ dacMux)

*Defines the DAC to use for an electrode.*

- virtual ElectrodeDacMuxEnumNet [GetElectrodeDacMux](#) (uint32\_t electrode, uint32\_t listmodelIndex)

*Gets the DAC which is used for an electrode.*

- virtual ElectrodeDacMuxEnumNet [GetElectrodeDacMux](#) (uint32\_t Scu\_HS, uint32\_t electrode, uint32\_t listmodelIndex)

*Gets the DAC which is used for an electrode.*

- virtual void [SetElectrodeEnable](#) (uint32\_t electrode, uint32\_t listmodelIndex, array< bool >^ enable)

*Enables or disables the stimulation switch for an electrode.*

- virtual void [SetElectrodeEnable](#) (uint32\_t electrode, uint32\_t listmodelIndex, bool enable)

*Enables or disables the stimulation switch for an electrode.*

- virtual void [SetElectrodeEnable](#) (uint32\_t Scu\_HS, uint32\_t electrode, uint32\_t listmodelIndex, bool enable)

*Enables or disables the stimulation switch for an electrode.*

- virtual void [SetElectrodeEnable](#) (uint32\_t Scu\_HS, uint32\_t electrode, uint32\_t listmodelIndex, array< bool >^ enable)

*Enables or disables the stimulation switch for an electrode.*

- virtual bool [GetElectrodeEnable](#) (uint32\_t electrode, uint32\_t listmodelIndex)

*Gets whether an electrode is enabled or disabled for stimulation.*

- virtual bool [GetElectrodeEnable](#) (uint32\_t Scu\_HS, uint32\_t electrode, uint32\_t listmodelIndex)

*Gets whether an electrode is enabled or disabled for stimulation.*

- virtual void [SetExternalElectrodeEnable](#) (uint32\_t electrode, uint32\_t listmodelIndex, array< bool >^ enable)

*Enables or disables the stimulation switch for an external electrode.*

- virtual void [SetExternalElectrodeEnable](#) (uint32\_t electrode, uint32\_t listmodelIndex, bool enable)

*Enables or disables the stimulation switch for an external electrode.*

- virtual bool [GetExternalElectrodeEnable](#) (uint32\_t electrode, uint32\_t listmodelIndex)

*Gets whether an electrode is enabled or disabled for stimulation.*

- virtual void [SetBlankingEnable](#) (uint32\_t electrode, bool enable)

*Defines whether an electrode should be blanked while stimulation is in progress.*

- virtual void [SetBlankingEnable](#) (uint32\_t electrode, array< bool >^ enable)

*Defines whether an electrode should be blanked while stimulation is in progress.*

- virtual void [SetBlankingEnable](#) (uint32\_t Scu\_HS, uint32\_t electrode, bool enable)

*Defines whether an electrode should be blanked while stimulation is in progress.*

- virtual void [SetBlankingEnable](#) (uint32\_t Scu\_HS, uint32\_t electrode, array< bool >^ enable)

*Defines whether an electrode should be blanked while stimulation is in progress.*

- virtual bool [GetBlankingEnable](#) (uint32\_t electrode)

*Gets whether an electrode should be blanked while stimulation is in progress.*

- virtual bool [GetBlankingEnable](#) (uint32\_t Scu\_HS, uint32\_t electrode)

*Gets whether an electrode should be blanked while stimulation is in progress.*

- virtual void [SetEnableAmplifierProtectionSwitch](#) (uint32\_t electrode, bool enable)

*Defines whether the Amplifier Protection Switch is openend while stimulation is in progress.*

- virtual void [SetEnableAmplifierProtectionSwitch](#) (uint32\_t electrode, array< bool >^ enable)

*Defines whether the Amplifier Protection Switch is openend while stimulation is in progress.*

- virtual void [SetEnableAmplifierProtectionSwitch](#) (uint32\_t Scu\_HS, uint32\_t electrode, bool enable)

- Defines whether the Amplifier Protection Switch is openend while stimulation is in progress.*

  - virtual void [SetEnableAmplifierProtectionSwitch](#) (uint32\_t Scu\_HS, uint32\_t electrode, array< bool >^ enable)
- Defines whether the Amplifier Protection Switch is openend while stimulation is in progress.*

  - virtual bool [GetEnableAmplifierProtectionSwitch](#) (uint32\_t electrode)
- Gets whether the Amplifier Protection Switch is openend while stimulation is in progress.*

  - virtual bool [GetEnableAmplifierProtectionSwitch](#) (uint32\_t Scu\_HS, uint32\_t electrode)
- Gets whether the Amplifier Protection Switch is openend while stimulation is in progress.*

  - virtual uint32\_t [GetNumberOfStimulationElectrodes](#) ()
- template<typename digitalsourceenum >

  - virtual void [SetTriggerSource](#) (unsigned int triggernum, [DigitalSource](#)< digitalsourceenum >^ triggersource, int bitnum\_offset)
  - virtual void [SetTriggerSource](#) (unsigned int triggernum, TriggerSourceEnumNet triggersource, int bitnum\_offset)
  - virtual void [SetTriggerSource](#) (unsigned int triggernum, TriggerSourceEnumNet triggersource)
  - virtual TriggerSourceEnumNet [GetTriggerSource](#) (unsigned int triggernum)
  - virtual void [SetListmodeIndexRange](#) (unsigned int sideband, unsigned int startIndex, unsigned int endIndex, unsigned int mode)
  - virtual void [GetListmodeIndexRange](#) (unsigned int sideband, unsigned int &startIndex, unsigned int &endIndex, unsigned int &mode)
  - virtual void [SetListmodeTriggerSource](#) (unsigned int sideband, TriggerSourceEnumNet triggersource, int bitnumOffset)
  - virtual void [SetListmodeTriggerSource](#) (unsigned int sideband, TriggerSourceEnumNet triggersource)
  - virtual TriggerSourceEnumNet [GetListmodeTriggerSource](#) (unsigned int sideband)
  - virtual void [ListModeSendStart](#) (unsigned int sidebandMask)
  - virtual void [ListModeSendStop](#) (unsigned int sidebandMask)
  - virtual void [SetHeadstage](#) (unsigned int headstage)
  - virtual uint32\_t [GetHeadstage](#) ()
  - virtual void [SetDacAmplificationFactor](#) (uint32\_t DacNumber, double Factor)
- Set the amplification factor for a DAC.*

  - virtual double [GetDacAmplificationFactor](#) (uint32\_t DacNumber)
- Get the amplification factor for a DAC.*

  - virtual void [SetDigoutMode](#) (Stg200xDigoutModeEnumNet digoutMode)
- Sets the operation mode of the digital output port, can be Monitor, Manual or SyncOut*

  - virtual Stg200xDigoutModeEnumNet [GetDigoutMode](#) ()
- Gets the operation mode of the digital output port, can be Monitor, Manual or SyncOut*

  - virtual void [SetDigoutValue](#) (uint32\_t digoutValue)
- Sets the Value on the digital output port when in manual mode.*

  - virtual uint32\_t [GetDigoutValue](#) ()
- Gets the Value on the digital output port.*

  - virtual uint32\_t [GetDiginValue](#) ()
- Gets the Value on the digital input port.*

  - virtual void [SetSyncoutMap](#) (uint32\_t channel, uint32\_t syncoutMap)
- Sets the mapping between external syncout outputs and internal syncout channels.*

  - virtual uint32\_t [GetSyncoutMap](#) (uint32\_t channel)
- Gets the mapping between external syncout outputs and internal syncout channels.*



## Additional Inherited Members

### 11.105.1 Detailed Description

Base class for the Stg200x.

From this class all STG related classes are derived: [Mcs.Usb.CStg200xDownloadBasicNet](#) [Mcs.Usb.CStg200xDownloadNet](#) for [Download Mode](#) and [Mcs.Usb.CStg200xStreamingNet](#) for [Streaming Mode](#).

[CStg200xBasicNet](#) is the base class to control MCS STG device.

### 11.105.2 Constructor & Destructor Documentation

#### 11.105.2.1 `~CStg200xBasicNet()` `virtual ~CStg200xBasicNet ( ) [virtual]`

The destructor.

### 11.105.3 Member Function Documentation

#### 11.105.3.1 `GetAnalogRanges()` `void GetAnalogRanges (` `int channel,` `[Out] int% URange,` `[Out] int% IRange )`

Gets the range of the analog outputs.

##### Parameters

<i>channel</i>	The channel which is queried.
<i>URange</i>	The Voltage range in mV.
<i>IRange</i>	The Current range in uA.

#### 11.105.3.2 `GetAnalogResolution()` `void GetAnalogResolution (` `int channel,`

```
[Out] int% URes,  
[Out] int% IRes )
```

Gets the resolution of the analog outputs.

#### Parameters

<i>channel</i>	The channel which is queried.
<i>URes</i>	The Voltage resolution in mV.
<i>IRes</i>	The Current resolution in uA.

**11.105.3.3 GetAutocalibrationDisabled()** `virtual bool GetAutocalibrationDisabled ( unsigned int channel ) [virtual]`

Gets the autocalibration configuration.

#### Parameters

<i>channel</i>	The channel number.
----------------	---------------------

#### Returns

`true` if autocalibration is disabled.

**11.105.3.4 GetAvailableMemory()** `uint32_t GetAvailableMemory ( )`

Gets the amount of memory available in the currently selected segment of the STG.

#### Returns

The memory available in the currently selected segment in bytes.

**11.105.3.5 GetBlankingEnable()** `[1/2] virtual bool GetBlankingEnable ( uint32_t electrode ) [virtual]`

Gets whether an electrode should be blanked while stimulation is in progress.

#### Parameters

<i>electrode</i>	The electrode number.
------------------	-----------------------

**Returns**

true if blanking is enabled while stimulation is in progress.

**11.105.3.6 GetBlankingEnable()** [2/2] `virtual bool GetBlankingEnable (`  
    `uint32_t Scu_HS,`  
    `uint32_t electrode ) [virtual]`

Gets whether an electrode should be blanked while stimulation is in progress.

**Parameters**

<i>Scu_HS</i>	The SCU headstage number.
---------------	---------------------------

**Parameters**

<i>electrode</i>	The electrode number.
------------------	-----------------------

**Returns**

true if blanking is enabled while stimulation is in progress.

**11.105.3.7 GetCurrentRangeInNanoAmp()** `virtual int32_t GetCurrentRangeInNanoAmp (`  
    `uint32_t channel ) [virtual]`

Gets the Current Range of the specified channel in Nanoamps.

**Parameters**

<i>channel</i>	Channel which is queried.
----------------	---------------------------

**Returns**

The Current Range of the specified channel in Nanoamps.

**11.105.3.8 GetCurrentResolutionInNanoAmp()** `virtual int32_t GetCurrentResolutionInNanoAmp (`  
    `uint32_t channel ) [virtual]`

Gets the Current Resolution of the specified channel in Nanoamps.

**Parameters**

<i>channel</i>	Channel which is queried.
----------------	---------------------------

**Returns**

The Current Resolution of the specified channel in Nanoamps.

**11.105.3.9 GetDacAmplificationFactor()** `virtual double GetDacAmplificationFactor ( uint32_t DacNumber ) [virtual]`

Get the amplification factor for a DAC.

**Parameters**

<i>DacNumber</i>	The number of the DAC.
------------------	------------------------

**Returns**

the amplification factor for the DAC queried, range is from -1.99999 to +1.99999.

**11.105.3.10 GetDACResolution()** `virtual int32_t GetDACResolution ( ) [virtual]`

Gets number of bits of the DAC resolution.

**Returns**

The DAC resolution in bits.

**11.105.3.11 GetDiginValue()** `virtual uint32_t GetDiginValue ( ) [virtual]`

Gets the Value on the digital input port.

**Returns**

The current value on the digital inputs.

**11.105.3.12 GetDigoutMode()** `virtual Stg200xDigoutModeEnumNet GetDigoutMode ( ) [virtual]`

Gets the operation mode of the digital output port, can be Monitor, Manual or SyncOut

**Returns**

The current operation mode.

**11.105.3.13 GetDigoutValue()** `virtual uint32_t GetDigoutValue ( ) [virtual]`

Gets the Value on the digital output port.

**Returns**

The current value on the digital outputs.

**11.105.3.14 GetElectrodeDacMux()** [1/2] `virtual ElectrodeDacMuxEnumNet GetElectrodeDacMux ( uint32_t electrode, uint32_t listmodeIndex ) [virtual]`

Gets the DAC which is used for an electrode.

**Parameters**

<i>electrode</i>	The electrode number.
<i>listmodeIndex</i>	The index for listmode.

**Returns**

The DAC in use, can be 1, 2 or 3. If the electrode is grounded 0 is returned.

**11.105.3.15 GetElectrodeDacMux()** [2/2] `virtual ElectrodeDacMuxEnumNet GetElectrodeDacMux ( uint32_t Scu_HS, uint32_t electrode, uint32_t listmodeIndex ) [virtual]`

Gets the DAC which is used for an electrode.

**Parameters**

<i>Scu_HS</i>	The SCU headstage number.
---------------	---------------------------

**Parameters**

<i>electrode</i>	The electrode number.
<i>listmodeIndex</i>	The index for listmode.

**Returns**

The DAC in use, can be 1, 2 or 3. If the electrode is grounded 0 is returned.

**11.105.3.16 GetElectrodeEnable() [1/2]** virtual bool GetElectrodeEnable (  
    uint32\_t *electrode*,  
    uint32\_t *listmodeIndex* ) [virtual]

Gets whether an electrode is enabled or disabled for stimulation.

**Parameters**

<i>electrode</i>	The electrode number.
<i>listmodeIndex</i>	The index for listmode.

**Returns**

true if the electrode is enabled, false if it is disabled.

**11.105.3.17 GetElectrodeEnable() [2/2]** virtual bool GetElectrodeEnable (  
    uint32\_t *Scu\_HS*,  
    uint32\_t *electrode*,  
    uint32\_t *listmodeIndex* ) [virtual]

Gets whether an electrode is enabled or disabled for stimulation.

**Parameters**

<i>Scu_HS</i>	The SCU headstage number.
---------------	---------------------------

**Parameters**

<i>electrode</i>	The electrode number.
<i>listmodeIndex</i>	The index for listmode.

**Returns**

true if the electrode is enabled, false if it is disabled.

**11.105.3.18 GetElectrodeMode()** [1/2] virtual uint32\_t GetElectrodeMode (  
uint32\_t *electrode* ) [virtual]

Gets the mode an electrode is in.

**Parameters**

<i>electrode</i>	The electrode number.
------------------	-----------------------

**Returns**

0 for automatic and 3 for manual mode.

**11.105.3.19 GetElectrodeMode()** [2/2] virtual uint32\_t GetElectrodeMode (  
uint32\_t *Scu\_HS*,  
uint32\_t *electrode* ) [virtual]

Gets the mode an electrode is in.

**Parameters**

<i>Scu_HS</i>	The SCU headstage number.
---------------	---------------------------

**Parameters**

<i>electrode</i>	The electrode number.
------------------	-----------------------

**Returns**

0 for automatic and 3 for manual mode.

**11.105.3.20 GetEnableAmplifierProtectionSwitch()** [1/2] `virtual bool GetEnableAmplifierProtectionSwitch (`  
`uint32_t electrode ) [virtual]`

Gets whether the Amplifier Protection Switch is openend while stimulation is in progress.

#### Parameters

<i>electrode</i>	The electrode number.
------------------	-----------------------

#### Returns

true if the switch is to be opened, false if it is closed while stimulation is in progress.

**11.105.3.21 GetEnableAmplifierProtectionSwitch()** [2/2] `virtual bool GetEnableAmplifierProtectionSwitch (`  
`uint32_t Scu_HS,`  
`uint32_t electrode ) [virtual]`

Gets whether the Amplifier Protection Switch is openend while stimulation is in progress.

#### Parameters

<i>Scu_HS</i>	The SCU headstage number.
---------------	---------------------------

#### Parameters

<i>electrode</i>	The electrode number.
------------------	-----------------------

#### Returns

true if the switch is to be opened, false if it is closed while stimulation is in progress.

**11.105.3.22 GetExternalElectrodeEnable()** `virtual bool GetExternalElectrodeEnable (`  
`uint32_t electrode,`  
`uint32_t listmodeIndex ) [virtual]`

Gets weather an electrode is enabled or disabled for stimulation.

#### Parameters

<i>electrode</i>	The electrode number.
<i>listmodeIndex</i>	The index for listmode.



**Returns**

true if the electrode is enabled, false if it is disabled.

**11.105.3.23 GetFAAmplification()** `virtual uint32_t GetFAAmplification ( ) [virtual]`

**11.105.3.24 GetHeadstage()** `virtual uint32_t GetHeadstage ( ) [virtual]`

**11.105.3.25 GetListmodeIndexRange()** `virtual void GetListmodeIndexRange (`  
    `unsigned int sideband,`  
    `unsigned int & startIndex,`  
    `unsigned int & endIndex,`  
    `unsigned int & mode ) [virtual]`

**11.105.3.26 GetListmodeTriggerSource()** `virtual TriggerSourceEnumNet GetListmodeTriggerSource (`  
    `unsigned int sideband ) [virtual]`

**11.105.3.27 GetNumberOfAnalogChannels()** `virtual uint32_t GetNumberOfAnalogChannels ( ) [virtual]`

Gets the Number of available analog channels of the device.

**Returns**

The number of analog channels.

**11.105.3.28 GetNumberOfHWDACPaths()** `virtual uint32_t GetNumberOfHWDACPaths ( ) [virtual]`

Gets the Number of HW Stimulation DACs of the device.

**Returns**

The number of independent HW Stimulation outputs.

**11.105.3.29 GetNumberOfStimulationElectrodes()** virtual uint32\_t GetNumberOfStimulationElectrodes ( ) [virtual]

**11.105.3.30 GetNumberOfStimulationSourcesPerElectrode()** virtual uint32\_t GetNumberOfStimulationSourcesPerElectrode ( ) [virtual]

Gets the number of stimulation sources (DACs) per electrode.

#### Returns

The number of stimulation sources (DACs) per electrode.

**11.105.3.31 GetNumberOfSyncoutChannels()** virtual uint32\_t GetNumberOfSyncoutChannels ( ) [virtual]

Gets the Number of available syncout channels of the device.

#### Returns

The number of analog channels.

**11.105.3.32 GetNumberOfTriggerInputs()** virtual uint32\_t GetNumberOfTriggerInputs ( ) [virtual]

Gets the Number of trigger inputs of the device.

#### Returns

The number of trigger inputs.

**11.105.3.33 GetOutputRate()** uint32\_t GetOutputRate ( )

Queries the output rate of the STG. Valid rates are from 1000 Hz to 50000 Hz.

#### Returns

Returns the current output rate in Hz.

**11.105.3.34 GetStgProgramInfo()** [1/2] void GetStgProgramInfo ( [Out] bool% *IsProgrammed*, [Out] DateTime% *timestamp*, [Out] String^% *filename*, [Out] Guid% *guid* )

Queries Download information from the STG. If download information was stored by the use of [SetStgProgramInfo](#), this function can be used to retrieve it.

## Parameters

<i>IsProgrammed</i>	Flag wether download information is valid.
<i>timestamp</i>	The timestamp of last download.
<i>filename</i>	The filename of the downloaoded waveform.
<i>guid</i>	A GUID.

**11.105.3.35 GetStgProgramInfo()** [2/2] `void GetStgProgramInfo (`  
`[Out] bool% IsProgrammed,`  
`[Out] System::Runtime::InteropServices::ComTypes::FILETIME% timestamp,`  
`[Out] String^% filename,`  
`[Out] Guid% guid )`

Queries Download information from the STG. If download information was stored by the use of [SetStgProgramInfo](#), this function can be used to retrieve it.

## Parameters

<i>IsProgrammed</i>	Flag wether download information is valid.
<i>timestamp</i>	The timestamp of last download.
<i>filename</i>	The filename of the downloaoded waveform.

**11.105.3.36 GetStgVersionInfo()** `void GetStgVersionInfo (`  
`[Out] String^% SwVersion,`  
`[Out] String^% HwVersion )`

Queries software and hardware version.

## Parameters

<i>SwVersion</i>	The current Software Version of the STG.
<i>HwVersion</i>	The Hardware Revision of the STG.

**11.105.3.37 GetSyncoutMap()** `virtual uint32_t GetSyncoutMap (`  
`uint32_t channel ) [virtual]`

Gets the mapping between external syncout outputs and internal syncout channels.

## Parameters

<i>channel</i>	The external syncout output channel number (zero based).
----------------	--

**Returns**

The bitmap of internal syncout channels mapped to channel.

**11.105.3.38 GetTotalMemory()** `uint32_t GetTotalMemory ( )`

Gets the total amount of memory available on the STG (all segments).

**Returns**

The total memory available on the STG in bytes.

**11.105.3.39 GetTriggerSource()** `virtual TriggerSourceEnumNet GetTriggerSource ( unsigned int triggernum ) [virtual]`**11.105.3.40 GetVoltageRangeInMicroVolt()** `virtual int32_t GetVoltageRangeInMicroVolt ( uint32_t channel ) [virtual]`

Gets the Voltage Range of the specified channel in Microvolts.

**Parameters**

<i>channel</i>	Channel which is queried.
----------------	---------------------------

**Returns**

The Voltage Range of the specified channel in Microvolts.

**11.105.3.41 GetVoltageResolutionInMicroVolt()** `virtual int32_t GetVoltageResolutionInMicroVolt ( uint32_t channel ) [virtual]`

Gets the Voltage Resolution of the specified channel in Microvolts.

**Parameters**

<i>channel</i>	Channel which is queried.
----------------	---------------------------

**Returns**

The Voltage Resolution of the specified channel in Microvolts.

**11.105.3.42 ListModeSendStart()** virtual void ListModeSendStart (   
     unsigned int *sidebandMask* ) [virtual]

**11.105.3.43 ListModeSendStop()** virtual void ListModeSendStop (   
     unsigned int *sidebandMask* ) [virtual]

**11.105.3.44 SendStart()** void SendStart (   
     uint32\_t *triggermap* )

Start (Trigger) the STG. The startup delay is in the range of a few ms.

#### Parameters

<i>triggermap</i>	A bitmap of triggers which will be started.
-------------------	---

**11.105.3.45 SendStop() [1/2]** void SendStop (   
     uint32\_t *triggermap* )

Stop some or all triggers of the STG.

#### Parameters

<i>triggermap</i>	A bitmap of triggers which will be stopped.
-------------------	---

**11.105.3.46 SendStop() [2/2]** void SendStop (   
     uint32\_t *triggermap*,   
     int *options* )

Stop some or all triggers of the STG.

#### Parameters

<i>triggermap</i>	A bitmap of triggers which will be stopped.
<i>options</i>	bitmap of options, currently only STOP_OPTION_SAVESTOP (0x80) is defined, which bypasses the stop commands when a syncout associated with a given sync-out has bit 1 (0x02) set. Can be used e.g. to prevent a stop while a biphasic stimulation pulse is active..

**11.105.3.47 SetAutocalibrationDisabled()** `virtual void SetAutocalibrationDisabled ( unsigned int channel, bool disable ) [virtual]`

Sets the autocalibration configuration.

Parameters

<i>channel</i>	The channel number.
<i>disable</i>	true if autocalibration is to be disabled.

**11.105.3.48 SetBlankingEnable() [1/4]** `virtual void SetBlankingEnable ( uint32_t electrode, array< bool >^ enable ) [virtual]`

Defines whether an electrode should be blanked while stimulation is in progress.

Parameters

<i>electrode</i>	The electrode number.
<i>enable</i>	True if the switch is to be opened, false if it is to remain closed while stimulation is in progress.

**11.105.3.49 SetBlankingEnable() [2/4]** `virtual void SetBlankingEnable ( uint32_t electrode, bool enable ) [virtual]`

Defines whether an electrode should be blanked while stimulation is in progress.

Parameters

<i>electrode</i>	The electrode number.
<i>enable</i>	True if the switch is to be opened, false if it is to remain closed while stimulation is in progress.

**11.105.3.50 SetBlankingEnable() [3/4]** `virtual void SetBlankingEnable ( uint32_t Scu_HS, uint32_t electrode, array< bool >^ enable ) [virtual]`

Defines whether an electrode should be blanked while stimulation is in progress.

Parameters

<i>Scu_HS</i>	The SCU headstage number.
---------------	---------------------------

## Parameters

<i>electrode</i>	The electrode number.
<i>enable</i>	True if the switch is to be opened, false if it is to remain closed while stimulation is in progress.

**11.105.3.51 SetBlankingEnable()** [4/4] `virtual void SetBlankingEnable (`  
    `uint32_t Scu_HS,`  
    `uint32_t electrode,`  
    `bool enable ) [virtual]`

Defines whether an electrode should be blanked while stimulation is in progress.

## Parameters

<i>Scu_HS</i>	The SCU headstage number.
---------------	---------------------------

## Parameters

<i>electrode</i>	The electrode number.
<i>enable</i>	True if the switch is to be opened, false if it is to remain closed while stimulation is in progress.

**11.105.3.52 SetCurrentMode()** [1/2] `virtual void SetCurrentMode ( ) [virtual]`

Sets all channels to current mode (STG3008-FA and STG400x only).

**11.105.3.53 SetCurrentMode()** [2/2] `virtual void SetCurrentMode (`  
    `unsigned int channel ) [virtual]`

Sets a channel to current mode (STG3008-FA and STG400x only).

## Parameters

<i>channel</i>	The channel to change.
----------------	------------------------

**11.105.3.54 SetDacAmplificationFactor()** virtual void SetDacAmplificationFactor (   
     uint32\_t *DacNumber*,  
     double *Factor* ) [virtual]

Set the amplification factor for a DAC.

Parameters

<i>DacNumber</i>	The number of the DAC.
<i>Factor</i>	the amplification factor for that DAC, range is from -1.99999 to +1.99999.

**11.105.3.55 SetDigoutMode()** virtual void SetDigoutMode (   
     Stg200xDigoutModeEnumNet *digoutMode* ) [virtual]

Sets the operation mode of the digital output port, can be Monitor, Manual or SyncOut

Parameters

<i>digoutMode</i>	The new operation mode.
-------------------	-------------------------

**11.105.3.56 SetDigoutValue()** virtual void SetDigoutValue (   
     uint32\_t *digoutValue* ) [virtual]

Sets the Value on the digital output port when in manual mode.

Parameters

<i>digoutValue</i>	The new value on the digital outputs.
--------------------	---------------------------------------

**11.105.3.57 SetElectrodeDacMux()** [1/4] virtual void SetElectrodeDacMux (   
     uint32\_t *electrode*,  
     uint32\_t *listmodeIndex*,  
     array< ElectrodeDacMuxEnumNet >^ *dacMux* ) [virtual]

Defines the DAC to use for an electrode.

Parameters

<i>electrode</i>	The electrode number.
------------------	-----------------------



## Parameters

<i>listmodeIndex</i>	The index for listmode.
<i>dacMux</i>	The DAC to use, can be ElectrodeDacMuxEnumNet.Stg1 (1), ElectrodeDacMuxEnumNet.Stg2 (2) or ElectrodeDacMuxEnumNet.Stg3 (3). To ground an electrode, use ElectrodeDacMuxEnumNet.Ground (0).

**11.105.3.58 SetElectrodeDacMux() [2/4]** virtual void SetElectrodeDacMux (   
     uint32\_t *electrode*,   
     uint32\_t *listmodeIndex*,   
     ElectrodeDacMuxEnumNet *dacMux* ) [virtual]

Defines the DAC to use for an electrode.

## Parameters

<i>electrode</i>	The electrode number.
------------------	-----------------------

## Parameters

<i>listmodeIndex</i>	The index for listmode.
<i>dacMux</i>	The DAC to use, can be ElectrodeDacMuxEnumNet.Stg1 (1), ElectrodeDacMuxEnumNet.Stg2 (2) or ElectrodeDacMuxEnumNet.Stg3 (3). To ground an electrode, use ElectrodeDacMuxEnumNet.Ground (0).

**11.105.3.59 SetElectrodeDacMux() [3/4]** virtual void SetElectrodeDacMux (   
     uint32\_t *Scu\_HS*,   
     uint32\_t *electrode*,   
     uint32\_t *listmodeIndex*,   
     array< ElectrodeDacMuxEnumNet >^ *dacMux* ) [virtual]

Defines the DAC to use for an electrode.

## Parameters

<i>Scu_HS</i>	The SCU headstage number.
---------------	---------------------------

## Parameters

<i>electrode</i>	The electrode number.
------------------	-----------------------

## Parameters

<i>listmodeIndex</i>	The index for listmode.
<i>dacMux</i>	The DAC to use, can be ElectrodeDacMuxEnumNet.Stg1 (1), ElectrodeDacMuxEnumNet.Stg2 (2) or ElectrodeDacMuxEnumNet.Stg3 (3). To ground an electrode, use ElectrodeDacMuxEnumNet.Ground (0).

**11.105.3.60 SetElectrodeDacMux()** [4/4] `virtual void SetElectrodeDacMux (`  
`uint32_t Scu_HS,`  
`uint32_t electrode,`  
`uint32_t listmodeIndex,`  
`ElectrodeDacMuxEnumNet dacMux ) [virtual]`

Defines the DAC to use for an electrode.

## Parameters

<i>Scu_HS</i>	The SCU headstage number.
---------------	---------------------------

## Parameters

<i>electrode</i>	The electrode number.
------------------	-----------------------

## Parameters

<i>listmodeIndex</i>	The index for listmode.
<i>dacMux</i>	The DAC to use, can be ElectrodeDacMuxEnumNet.Stg1 (1), ElectrodeDacMuxEnumNet.Stg2 (2) or ElectrodeDacMuxEnumNet.Stg3 (3). To ground an electrode, use ElectrodeDacMuxEnumNet.Ground (0).

**11.105.3.61 SetElectrodeEnable()** [1/4] `virtual void SetElectrodeEnable (`  
`uint32_t electrode,`

```
uint32_t listmodeIndex,
array< bool >^ enable ) [virtual]
```

Enables or disables the stimulation switch for an electrode.

#### Parameters

<i>electrode</i>	The electrode number.
------------------	-----------------------

#### Parameters

<i>listmodeIndex</i>	The index for listmode.
<i>enable</i>	1 to enable the electrode, 0 to disable.

**11.105.3.62 SetElectrodeEnable() [2/4]** virtual void SetElectrodeEnable (

```
uint32_t electrode,
uint32_t listmodeIndex,
bool enable ) [virtual]
```

Enables or disables the stimulation switch for an electrode.

#### Parameters

<i>electrode</i>	The electrode number.
------------------	-----------------------

#### Parameters

<i>listmodeIndex</i>	The index for listmode.
<i>enable</i>	1 to enable the electrode, 0 to disable.

**11.105.3.63 SetElectrodeEnable() [3/4]** virtual void SetElectrodeEnable (

```
uint32_t Scu_HS,
uint32_t electrode,
uint32_t listmodeIndex,
array< bool >^ enable ) [virtual]
```

Enables or disables the stimulation switch for an electrode.

## Parameters

<i>Scu_HS</i>	The SCU headstage number.
---------------	---------------------------

## Parameters

<i>electrode</i>	The electrode number.
------------------	-----------------------

## Parameters

<i>listmodeIndex</i>	The index for listmode.
<i>enable</i>	1 to enable the electrode, 0 to disable.

**11.105.3.64 SetElectrodeEnable()** [4/4] `virtual void SetElectrodeEnable (`  
    `uint32_t Scu_HS,`  
    `uint32_t electrode,`  
    `uint32_t listmodeIndex,`  
    `bool enable ) [virtual]`

Enables or disables the stimulation switch for an electrode.

## Parameters

<i>Scu_HS</i>	The SCU headstage number.
---------------	---------------------------

## Parameters

<i>electrode</i>	The electrode number.
------------------	-----------------------

## Parameters

<i>listmodeIndex</i>	The index for listmode.
<i>enable</i>	1 to enable the electrode, 0 to disable.

**11.105.3.65 SetElectrodeMode()** [1/4] virtual void SetElectrodeMode (   
 uint32\_t *electrode*,  
 array< ElectrodeModeEnumNet >^ *mode* ) [virtual]

Puts an electrode in either automatic or manual mode.

**Parameters**

<i>electrode</i>	The electrode number.
------------------	-----------------------

**Returns**

0 for automatic and 3 for manual mode.

**11.105.3.66 SetElectrodeMode()** [2/4] virtual void SetElectrodeMode (   
 uint32\_t *electrode*,  
 ElectrodeModeEnumNet *mode* ) [virtual]

Puts an electrode in either automatic or manual mode.

**Parameters**

<i>electrode</i>	The electrode number.
------------------	-----------------------

**Parameters**

<i>mode</i>	0 for automatic and 3 for manual mode.
-------------	--

**11.105.3.67 SetElectrodeMode()** [3/4] virtual void SetElectrodeMode (   
 uint32\_t *Scu\_HS*,  
 uint32\_t *electrode*,  
 array< ElectrodeModeEnumNet >^ *mode* ) [virtual]

Puts an electrode in either automatic or manual mode.

## Parameters

<i>Scu_HS</i>	The SCU headstage number.
---------------	---------------------------

## Parameters

<i>electrode</i>	The electrode number.
------------------	-----------------------

## Returns

0 for automatic and 3 for manual mode.

**11.105.3.68 SetElectrodeMode()** [4/4] `virtual void SetElectrodeMode (`  
`uint32_t Scu_HS,`  
`uint32_t electrode,`  
`ElectrodeModeEnumNet mode ) [virtual]`

Puts an electrode in either automatic or manual mode.

## Parameters

<i>Scu_HS</i>	The SCU headstage number.
---------------	---------------------------

## Parameters

<i>electrode</i>	The electrode number.
------------------	-----------------------

## Parameters

<i>mode</i>	0 for automatic and 3 for manual mode.
-------------	--

**11.105.3.69 SetEnableAmplifierProtectionSwitch()** [1/4] `virtual void SetEnableAmplifierProtection↔`  
`Switch (`

```
uint32_t electrode,
array< bool >^ enable ) [virtual]
```

Defines whether the Amplifier Protection Switch is openend while stimulation is in progress.

#### Parameters

<i>electrode</i>	The electrode number.
<i>enable</i>	True if the switch is to be opened, false if it is to remain closed while stimulation is in progress.

**11.105.3.70 SetEnableAmplifierProtectionSwitch() [2/4]** virtual void SetEnableAmplifierProtection↔  
Switch (

```
uint32_t electrode,
bool enable ) [virtual]
```

Defines whether the Amplifier Protection Switch is openend while stimulation is in progress.

#### Parameters

<i>electrode</i>	The electrode number.
<i>enable</i>	True if the switch is to be opened, false if it is to remain closed while stimulation is in progress.

**11.105.3.71 SetEnableAmplifierProtectionSwitch() [3/4]** virtual void SetEnableAmplifierProtection↔  
Switch (

```
uint32_t Scu_HS,
uint32_t electrode,
array< bool >^ enable ) [virtual]
```

Defines whether the Amplifier Protection Switch is openend while stimulation is in progress.

#### Parameters

<i>Scu_HS</i>	The SCU headstage number.
---------------	---------------------------

#### Parameters

<i>electrode</i>	The electrode number.
<i>enable</i>	True if the switch is to be opened, false if it is to remain closed while stimulation is in progress.

**11.105.3.72 SetEnableAmplifierProtectionSwitch() [4/4]** virtual void SetEnableAmplifierProtection↔  
Switch (

```
uint32_t Scu_HS,
uint32_t electrode,
bool enable ) [virtual]
```

Defines whether the Amplifier Protection Switch is openend while stimulation is in progress.

#### Parameters

<i>Scu_HS</i>	The SCU headstage number.
---------------	---------------------------

#### Parameters

<i>electrode</i>	The electrode number.
<i>enable</i>	True if the switch is to be opened, false if it is to remain closed while stimulation is in progress.

**11.105.3.73 SetExternalElectrodeEnable() [1/2]** virtual void SetExternalElectrodeEnable (   
uint32\_t electrode,   
uint32\_t listmodeIndex,   
array< bool >^ enable ) [virtual]

Enables or disables the stimulation switch for an external electrode.

#### Parameters

<i>electrode</i>	The electrode number.
------------------	-----------------------

#### Parameters

<i>listmodeIndex</i>	The index for listmode.
<i>enable</i>	1 to enable the electrode, 0 to disable.

**11.105.3.74 SetExternalElectrodeEnable() [2/2]** virtual void SetExternalElectrodeEnable (   
uint32\_t electrode,   
uint32\_t listmodeIndex,   
bool enable ) [virtual]

Enables or disables the stimulation switch for an external electrode.



## Parameters

<i>electrode</i>	The electrode number.
------------------	-----------------------

## Parameters

<i>listmodeIndex</i>	The index for listmode.
<i>enable</i>	1 to enable the electrode, 0 to disable.

**11.105.3.75 SetFAAmplification()** virtual void SetFAAmplification (   
 unsigned int *amplification* ) [virtual]

**11.105.3.76 SetHeadstage()** virtual void SetHeadstage (   
 unsigned int *headstage* ) [virtual]

**11.105.3.77 SetListmodeIndexRange()** virtual void SetListmodeIndexRange (   
 unsigned int *sideband*,   
 unsigned int *startIndex*,   
 unsigned int *endIndex*,   
 unsigned int *mode* ) [virtual]

**11.105.3.78 SetListmodeTriggerSource()** [1/2] virtual void SetListmodeTriggerSource (   
 unsigned int *sideband*,   
 TriggerSourceEnumNet *triggersource* ) [virtual]

**11.105.3.79 SetListmodeTriggerSource()** [2/2] virtual void SetListmodeTriggerSource (   
 unsigned int *sideband*,   
 TriggerSourceEnumNet *triggersource*,   
 int *bitnumOffset* ) [virtual]

**11.105.3.80 SetMeasurementMode()** virtual void SetMeasurementMode (   
 unsigned int *channel* ) [virtual]

Sets a channel to measurement mode (STG3008-FA).

## Parameters

<i>channel</i>	The channel to change.
----------------	------------------------

**11.105.3.81 SetOutputRate()** `void SetOutputRate (`  
     `uint32_t rate )`

Change the output rate of the STG. Valid rates are from 1000 Hz to 50000 Hz.

## Parameters

<i>rate</i>	The new output rate in Hz.
-------------	----------------------------

**11.105.3.82 SetStgProgramInfo()** `void SetStgProgramInfo (`  
     `DateTime timestamp,`  
     `String^ filename,`  
     `Guid guid )`

Store Download information in the STG. This function can be used to store the filename and timestamp of the last download for later query.

## Parameters

<i>timestamp</i>	The timestamp of last download.
<i>filename</i>	The filename of the downloaded waveform.

**11.105.3.83 SetSyncoutMap()** `virtual void SetSyncoutMap (`  
     `uint32_t channel,`  
     `uint32_t syncoutMap ) [virtual]`

Sets the mapping between external syncout outputs and internal syncout channels.

## Parameters

<i>channel</i>	The external syncout output channel number (zero based).
<i>syncoutMap</i>	A bitmap of internal syncout channels to map to channel.

**11.105.3.84 SetTriggerSource()** [1/3] `virtual void SetTriggerSource (`  
     `unsigned int triggernum,`

```
DigitalSource< digitalsourceenum >^ triggersource,
int bitnum_offset ) [virtual]
```

**11.105.3.85 SetTriggerSource() [2/3]** virtual void SetTriggerSource ( unsigned int *triggernum*, TriggerSourceEnumNet *triggersource* ) [virtual]

**11.105.3.86 SetTriggerSource() [3/3]** virtual void SetTriggerSource ( unsigned int *triggernum*, TriggerSourceEnumNet *triggersource*, int *bitnum\_offset* ) [virtual]

**11.105.3.87 SetVoltageMode() [1/2]** virtual void SetVoltageMode ( ) [virtual]

Sets all channels to voltage mode (STG3008-FA and STG400x only).

**11.105.3.88 SetVoltageMode() [2/2]** virtual void SetVoltageMode ( unsigned int *channel* ) [virtual]

Sets a channel to voltage mode (STG3008-FA and STG400x only).

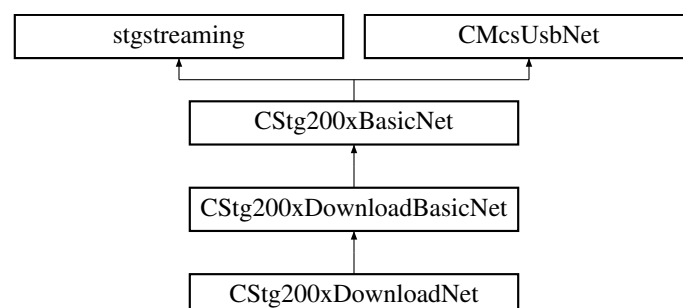
#### Parameters

<i>channel</i>	The channel to change.
----------------	------------------------

## 11.106 CStg200xDownloadBasicNet Class Reference

[CStg200xDownloadBasicNet](#) is the base class to control the download mode of the MCS STG device.

Inheritance diagram for CStg200xDownloadBasicNet:



## Public Member Functions

- virtual void [SetupTrigger](#) (uint32\_t first\_trigger, array< uint32\_t >^ channelmap, array< uint32\_t >^ syncoutmap, array< uint32\_t >^ repeat)  
*Configures the trigger settings for the STG. Note that all memory segments have their own trigger setting.*
- virtual void [SetupTriggerSingle](#) (uint32\_t trigger, uint32\_t channelmap, uint32\_t syncoutmap, uint32\_t repeat)  
*Configures the trigger settings for the STG. Note that all memory segments have their own trigger setting.*
- void [GetTrigger](#) ([Out] array< uint32\_t >^% channelmap, [Out] array< uint32\_t >^% syncoutmap, [Out] array< uint32\_t >^% repeat)  
*Queries the trigger settings for the STG. Note that all memory segments have their own trigger setting.*
- void [GetSweepCount](#) ([Out] array< uint32\_t >^% sweeps, [Out] array< uint32\_t >^% triggers)  
*Get the sweep and trigger count of the STG.*
  - The triggercount tells how many times each trigger was active and is reset to zero on download of new channel data.
  - The sweepcount tells how many times each trigger was already repeated. This count is set to zero on trigger and counts up to repeat in [CStg200xDownloadBasicNet::SetupTrigger](#).
- void [ForceStatusEvent](#) ()  
*Force a status event.*
- void [ResetStatus](#) (uint32\_t triggermap)  
*Reset the status flag.*
- uint32\_t [GetMemoryUsageDAC](#) (uint32\_t Channel)  
*Queries the memory usage of the current segment and selected analog DAC channel.*
- uint32\_t [GetMemoryUsageSyncout](#) (uint32\_t Channel)  
*Queries the memory usage of the current segment and selected syncout channel.*
- virtual void [ClearSyncData](#) (uint32\_t channel)  
*Delete a SyncOut pattern for a channel from STG memory.*
- virtual void [SendSyncData](#) (uint32\_t channel, array< uint16\_t >^ pData, array< uint64\_t >^ tData)  
*Uploads sync output data to the STG.*  
*Sends sync output data to a given channel on the STG. The list of datapoints will be sent to the selected sync output channel. Sync output data previously sent to the channel is overwritten.*  
*Each datapoint is represented by an integer value and can be either 0 or 1.*  
*The duration is given as a list of 64 bit integers. Durations are given in units of  $\mu$ s. The STG has a resolution of 20  $\mu$ s. If your application can not handle 64 bit integers, use the [STG200x\\_SendSyncData32\(\)](#) call instead.*
- virtual void [ClearChannelData](#) (uint32\_t channel)  
*Delete a stimulus pattern for a channel from STG memory*
- virtual void [SendChannelData](#) (uint32\_t channel, array< uint16\_t >^ pData, array< uint64\_t >^ tData)  
*Uploads analog data (stimulus patterns) to the STG.*  
*Sends datapoints to a given channel on the STG. The list of datapoints will be sent to the selected channel. Data previously sent to the channel is overwritten.*  
*Each datapoint is represented by an integer value in the range from 0 to 4095 (bit 0 to 11), its sign is taken from bit 12, 0 is for positive amplitude, and 1 for negative amplitude Bits 13 to 15 have to be zero.*  
*The duration is given as a list of 64 bit integers. Durations are given in units of  $\mu$ s. The STG has a resolution of 20  $\mu$ s.*
- virtual void [EnableAutoReset](#) ()  
*Enable AutoReset of the STG Status.*
- virtual void [DisableAutoReset](#) ()  
*Disable AutoReset of the STG Status.*
- virtual void [SetupRetriggerMode](#) (int8\_t trigger, RetriggerActionEnumNet same\_trigger, RetriggerActionEnumNet other\_trigger)  
*Define the action on triggers while the STG is running.*  
*The STG has three options how to handle a successive trigger while a trigger is active.*
  - stop this trigger (default action)
  - restart this trigger
  - ignore the signal
- virtual void [SetupRetriggerMode](#) (RetriggerActionEnumNet same\_trigger, RetriggerActionEnumNet other\_trigger)

*Define the action on triggers while the STG is running.*

*The STG has three options how to handle a successive trigger while a trigger is active.*

- *stop this trigger (default action)*
- *restart this trigger*
- *ignore the signal*

## Properties

- [CStimulusFunctionNet](#)<sup>^</sup> [Stimulus](#) [get]

## Additional Inherited Members

### 11.106.1 Detailed Description

[CStg200xDownloadBasicNet](#) is the base class to control the download mode of the MCS STG device.

### 11.106.2 Member Function Documentation

**11.106.2.1 ClearChannelData()** `virtual void ClearChannelData (   
 uint32_t channel ) [virtual]`

Delete a stimulus pattern for a channel from STG memory

#### Parameters

<i>channel</i>	Specifies the channel to clear.
----------------	---------------------------------

**11.106.2.2 ClearSyncData()** `virtual void ClearSyncData (   
 uint32_t channel ) [virtual]`

Delete a SyncOut pattern for a channel from STG memory.

#### Parameters

<i>channel</i>	Specifies the syncout channel to clear.
----------------	---

**11.106.2.3 DisableAutoReset()** `virtual void DisableAutoReset ( ) [virtual]`

Disable AutoReset of the STG Status.

If autoreset is disabled, the STG status switches to FINISHED after the defined number of sweeps is finished. To switch back to the IDLE status, use CStg200xDownload::ResetStatus()

#### 11.106.2.4 EnableAutoReset() `virtual void EnableAutoReset ( ) [virtual]`

Enable AutoReset of the STG Status.

This is the default on power up. If autoreset is enabled, the STG status switches to FINISHED only for one poll cycle after this, it switches to IDLE automatically.

#### 11.106.2.5 ForceStatusEvent() `void ForceStatusEvent ( )`

Force a status event.

Force the DLL to create a PollMessage event and to call the pPollCallback function, even if no new status information is available.

#### 11.106.2.6 GetMemoryUsageDAC() `uint32_t GetMemoryUsageDAC ( uint32_t Channel )`

Queries the memory usage of the current segment and selected analog DAC channel.

The currently used memory is reported for the requested channel.

##### Parameters

<i>Channel</i>	channel for the amount of interested usage.
----------------	---

##### Returns

Returns the usage in STG memory.

#### 11.106.2.7 GetMemoryUsageSyncout() `uint32_t GetMemoryUsageSyncout ( uint32_t Channel )`

Queries the memory usage of the current segment and selected syncout channel.

The currently used memory is reported for the requested channel.

##### Parameters

<i>Channel</i>	channel for the amount of interested usage.
----------------	---

**Returns**

Returns the usage in STG memory.

**11.106.2.8 GetSweepCount()** `void GetSweepCount (`  
     `[Out] array< uint32_t >^% sweeps,`  
     `[Out] array< uint32_t >^% triggers )`

Get the sweep and trigger count of the STG.

- The triggercount tells how many times each trigger was active and is reset to zero on download of new channel data.
- The sweepcount tells how many times each trigger was already repeated. This count is set to zero on trigger and counts up to repeat in [CStg200xDownloadBasicNet::SetupTrigger](#).

**Parameters**

<i>sweeps</i>	on return contains the number of sweeps for each trigger.
<i>triggers</i>	on return contains the number of trigger events seen for each trigger.

**11.106.2.9 GetTrigger()** `void GetTrigger (`  
     `[Out] array< uint32_t >^% channelmap,`  
     `[Out] array< uint32_t >^% syncoutmap,`  
     `[Out] array< uint32_t >^% repeat )`

Queries the trigger settings for the STG. Note that all memory segments have their own trigger setting.

**Parameters**

<i>channelmap</i>	For each trigger, a bitmap of channels that belong to this trigger.
-------------------	---

**Parameters**

<i>syncoutmap</i>	For each trigger, a bitmap of syncouts that belong to this trigger.
<i>repeat</i>	For each trigger, define the number of times this trigger should be repeated.

**11.106.2.10 ResetStatus()** `void ResetStatus (`  
     `uint32_t triggermap )`

Reset the status flag.



## Parameters

<i>triggermap</i>	bitmap of trigger for which to reset the status.
-------------------	--

**11.106.2.11 SendChannelData()** virtual void SendChannelData (   
     uint32\_t *channel*,   
     array< uint16\_t >^ *pData*,   
     array< uint64\_t >^ *tData* ) [virtual]

Uploads analog data (stimulus patterns) to the STG.

Sends datapoints to a given channel on the STG. The list of datapoints will be sent to the selected channel. Data previously sent to the channel is overwritten.

Each datapoint is represented by an integer value in the range from 0 to 4095 (bit 0 to 11), its sign is taken from bit 12, 0 is for positive amplitude, and 1 for negative amplitude Bits 13 to 15 have to be zero.

The duration is given as a list of 64 bit integers. Durations are given in units of  $\mu$ s. The STG has a resolution of 20  $\mu$ s.

## Parameters

<i>channel</i>	Specifies the channel to append the data to.
<i>pData</i>	A list of datapoints.
<i>tData</i>	A list of durations as int64_t. The time is given in units of $\mu$ s.

**11.106.2.12 SendSyncData()** virtual void SendSyncData (   
     uint32\_t *channel*,   
     array< uint16\_t >^ *pData*,   
     array< uint64\_t >^ *tData* ) [virtual]

Uploads sync output data to the STG.

Sends sync output data to a given channel on the STG. The list of datapoints will be sent to the selected sync output channel. Sync output data previously sent to the channel is overwritten.

Each datapoint is represented by an integer value and can be either 0 or 1.

The duration is given as a list of 64 bit integers. Durations are given in units of  $\mu$ s. The STG has a resolution of 20  $\mu$ s. If your application can not handle 64 bit integers, use the STG200x\_SendSyncData32() call instead.

## Parameters

<i>channel</i>	Specifies the sync output channel to append the data to.
<i>pData</i>	A list of datapoints.
<i>tData</i>	A list of durations as int64_t. The time is given in units of $\mu$ s.

**11.106.2.13 SetupRetriggerMode()** [1/2] `virtual void SetupRetriggerMode (`  
`int8_t trigger,`  
`RetriggerActionEnumNet same_trigger,`  
`RetriggerActionEnumNet other_trigger ) [virtual]`

Define the action on triggers while the STG is running.

The STG has three options how to handle a successive trigger while a trigger is active.

- stop this trigger (default action)
- restart this trigger
- ignore the signal

#### Parameters

<i>trigger</i>	The trigger to change.
<i>same_trigger</i>	Action for successive triggers in Normal Mode, and for triggers to the currently selected segment in Multi - File Mode.
<i>other_trigger</i>	Action for successive triggers in Multi-File Mode for a trigger on a segment not currently selected. Not used in Normal Mode.

**11.106.2.14 SetupRetriggerMode()** [2/2] `virtual void SetupRetriggerMode (`  
`RetriggerActionEnumNet same_trigger,`  
`RetriggerActionEnumNet other_trigger ) [virtual]`

Define the action on triggers while the STG is running.

The STG has three options how to handle a successive trigger while a trigger is active.

- stop this trigger (default action)
- restart this trigger
- ignore the signal

#### Parameters

<i>same_trigger</i>	Action for successive triggers in Normal Mode, and for triggers to the currently selected segment in Multi - File Mode.
<i>other_trigger</i>	Action for successive triggers in Multi-File Mode for a trigger on a segment not currently selected. Not used in Normal Mode.

**11.106.2.15 SetupTrigger()** `virtual void SetupTrigger (`

```
uint32_t first_trigger,
array< uint32_t >^ channelmap,
array< uint32_t >^ syncoutmap,
array< uint32_t >^ repeat ) [virtual]
```

Configures the trigger settings for the STG. Note that all memory segments have their own trigger setting.

#### Parameters

<i>first_trigger</i>	The number of the first trigger to change.
----------------------	--

#### Parameters

<i>channelmap</i>	For each trigger, a bitmap of channels that belong to this trigger.
-------------------	---

#### Parameters

<i>syncoutmap</i>	For each trigger, a bitmap of syncouts that belong to this trigger.
<i>repeat</i>	For each trigger, define the number of times this trigger should be repeated.

**11.106.2.16 SetupTriggerSingle()** virtual void SetupTriggerSingle (

```
uint32_t trigger,
uint32_t channelmap,
uint32_t syncoutmap,
uint32_t repeat ) [virtual]
```

Configures the trigger settings for the STG. Note that all memory segments have their own trigger setting.

#### Parameters

<i>trigger</i>	The trigger to change.
----------------	------------------------

#### Parameters

<i>channelmap</i>	A bitmap of channels that belong to this trigger.
-------------------	---

## Parameters

<i>syncoutmap</i>	A bitmap of syncouts that belong to this trigger.
<i>repeat</i>	The number of times this trigger should be repeated.

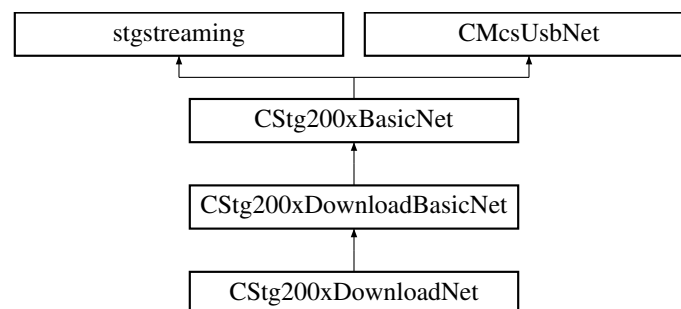
## 11.106.3 Property Documentation

11.106.3.1 Stimulus [CStimulusFunctionNet](#)<sup>^</sup> Stimulus [get]

## 11.107 CStg200xDownloadNet Class Reference

Main class for the STG download mode This class implements the STG download mode interface.

Inheritance diagram for CStg200xDownloadNet:



## Public Member Functions

- [CStg200xDownloadNet](#) ()  
*Use this constructor if you do not want to use the status callback.*
- [CStg200xDownloadNet](#) ([OnStgPollStatus](#)<sup>^</sup> pollStatus)  
*Use this constructor if you want to use the status callback.*
- [~CStg200xDownloadNet](#) ()
- void [PrepareAndSendData](#) (uint32\_t channel, array< int32\_t ><sup>^</sup> amplitude, array< uint64\_t ><sup>^</sup> duration, STG\_DestinationEnumNet destType)  
*Prepare and send data to a given channel on the STG. Previous data sent to that channel is erased first.*
- void [PrepareAndAppendData](#) (uint32\_t channel, array< int32\_t ><sup>^</sup> amplitude, array< uint64\_t ><sup>^</sup> duration, STG\_DestinationEnumNet destType)  
*Prepare and append data to a given channel on the STG.*
- void [ClearChannel\\_PrepareAndSendData](#) (uint32\_t channel, array< int32\_t ><sup>^</sup> amplitude, array< uint64\_t ><sup>^</sup> duration, STG\_DestinationEnumNet destType, bool doClear)  
*Prepare and append data to a given channel on the STG.*
- void [SendSegmentDefine](#) (array< uint32\_t ><sup>^</sup> segment\_list)

*Defines the segment memory layout of the STG.*

- void [SendSegmentStart](#) (uint32\_t triggermap, uint32\_t segment, Stg200xSegmentFlagsEnumNet segmentflags)

*Switchs segment and starts trigger.*

- void [SendSegmentSelect](#) (uint32\_t segment, Stg200xSegmentFlagsEnumNet segmentflags)

*Switchs segment.*

- void [EnableMultiFileMode](#) (uint32\_t submode)

*Enable the Multi-File mode of the STG.*

- void [DisableMultiFileMode](#) ()

*Disable the Multi-File mode of the STG*

- [StgStatusNet](#) ^ [QueryTriggerstatus](#) ()
- void [SetOutputMap](#) (uint32\_t ChannelLayout[])
- int32\_t [GetModuleTemp](#) (unsigned int channel)
- uint32\_t [GetModuleCurrent](#) (unsigned int channel)

## Events

- [OnStgPollStatus](#) ^ [Stg200xPollStatusEvent](#) [add, remove, raise]
- [OnMwPollStatus](#) ^ [MwPollStatusEvent](#) [add, remove, raise]

## Additional Inherited Members

### 11.107.1 Detailed Description

Main class for the STG download mode This class implements the STG download mode interface.

### 11.107.2 Constructor & Destructor Documentation

#### 11.107.2.1 CStg200xDownloadNet() [1/2] [CStg200xDownloadNet](#) ( )

Use this constructor if you do not want to use the status callback.

#### 11.107.2.2 CStg200xDownloadNet() [2/2] [CStg200xDownloadNet](#) ( [OnStgPollStatus](#) ^ *pollStatus* )

Use this constructor if you want to use the status callback.

#### 11.107.2.3 ~CStg200xDownloadNet() [~CStg200xDownloadNet](#) ( )

### 11.107.3 Member Function Documentation

**11.107.3.1 ClearChannel\_PrepareAndSendData()** `void ClearChannel_PrepareAndSendData (`  
     `uint32_t channel,`  
     `array< int32_t >^ amplitude,`  
     `array< uint64_t >^ duration,`  
     `STG_DestinationEnumNet destType,`  
     `bool doClear )`

Prepare and append data to a given channel on the STG.

Each datapoint is represented by an signed 32bit integer value. When using voltage stimulation, the values are in multiple of 1 uV, thus the possible range is += 2000 V. When using current stimulation, the values are in multiple of 1 nA, this the possible range is += 2000 mA.

The duration is given as a list of 64 bit integers. Durations are given in units of  $\mu$ s. The STG has a resolution of 20  $\mu$ s.

Blocks of data which should repeat can be defined by prepending such a block with an entry in the arrays where both amplitude and duration is zero. The end of such an block is marked by an entry where the duration is set to zero and the amplitude beeing set to the number of times the block should run. Blocks can be nested.

#### Parameters

<i>channel</i>	The channel number to send data to.
----------------	-------------------------------------

#### Parameters

<i>amplitude</i>	A list of amplitudes in units of $\mu$ V and nA in voltage and current mode, respectively.
------------------	--

#### Parameters

<i>duration</i>	A list of durations in units of $\mu$ s.
<i>destType</i>	specifies wheather the data is for syncout, current or voltage stimulation.

**11.107.3.2 DisableMultiFileMode()** `void DisableMultiFileMode ( )`

Disable the Multi-File mode of the STG

Switch the STG back to normal mode. In this mode, trigger inputs are assigned to channels, not to segments.

**11.107.3.3 EnableMultiFileMode()** `void EnableMultiFileMode (`  
`uint32_t submode )`

Enable the Multi-File mode of the STG.

In Multi-File mode, the trigger inputs switch between segments. To use this mode, define up to as many segments as trigger inputs are available and fill each segment with a stimulus pattern.

Now a trigger on trigger input 1 switches the STG to the first segment and starts all triggers in this segment. Likewise, a trigger on trigger input 2, 3 and 4 selects the respective segment and start all triggers in this segment. So the Multi-File Mode can be used to predefine up to four different stimuli which can be selected without the need for a computer connection.

#### Parameters

<i>submode</i>	The submode. Submode 0 is regular Multi-File mode as described above, submode 1 is extended Multi-File mode, where the segment is selected based on the digital pattern on the digital inputs. In this mode, 256 different segments can be defined and used.
----------------	--

**11.107.3.4 GetModuleCurrent()** `uint32_t GetModuleCurrent (`  
`unsigned int channel )`

**11.107.3.5 GetModuleTemp()** `int32_t GetModuleTemp (`  
`unsigned int channel )`

**11.107.3.6 PrepareAndAppendData()** `void PrepareAndAppendData (`  
`uint32_t channel,`  
`array< int32_t >^ amplitude,`  
`array< uint64_t >^ duration,`  
`STG_DestinationEnumNet destType )`

Prepare and append data to a given channel on the STG.

Each datapoint is represented by an signed 32bit integer value. When using voltage stimulation, the values are in multiple of 1 uV, thus the possible range is  $\pm 2000$  V. When using current stimulation, the values are in multiple of 1 nA, this the possible range is  $\pm 2000$  mA.

The duration is given as a list of 64 bit integers. Durations are given in units of  $\mu$ s. The STG has a resolution of 20  $\mu$ s.

Blocks of data which should repeat can be defined by prepending such a block with an entry in the arrays where both amplitude and duration is zero. The end of such an block is marked by an entry where the duration is set to zero and the amplitude beeing set to the number of times the block should run. Blocks can be nested.

## Parameters

<i>channel</i>	The channel number to send data to.
----------------	-------------------------------------

## Parameters

<i>amplitude</i>	A list of amplitudes in units of $\mu\text{V}$ and nA in voltage and current mode, respectively.
------------------	--

## Parameters

<i>duration</i>	A list of durations in units of $\mu\text{s}$ .
<i>destType</i>	specifies wheather the data is for syncout, current or voltage stimulation.

**11.107.3.7 PrepareAndSendData()** `void PrepareAndSendData (`  
`uint32_t channel,`  
`array< int32_t >^ amplitude,`  
`array< uint64_t >^ duration,`  
`STG_DestinationEnumNet destType )`

Prepare and send data to a given channel on the STG. Previous data sent to that channel is erased first.

Each datapoint is represented by an signed 32bit integer value. When using voltage stimulation, the values are in multiple of 1  $\mu\text{V}$ , thus the possible range is  $\pm 2000 \text{ V}$ . When using current stimulation, the values are in multiple of 1 nA, this the possible range is  $\pm 2000 \text{ mA}$ .

The duration is given as a list of 64 bit integers. Durations are given in units of  $\mu\text{s}$ . The STG has a resolution of 20  $\mu\text{s}$ .

Blocks of data which should repeat can be defined by prepending such a block with an entry in the arrays where both amplitude and duration is zero. The end of such a block is marked by an entry where the duration is set to zero and the amplitude beeing set to the number of times the block should run. Blocks can be nested.

## Parameters

<i>channel</i>	The channel number to send data to.
----------------	-------------------------------------

## Parameters

<i>amplitude</i>	A list of amplitudes in units of $\mu\text{V}$ and nA in voltage and current mode, respectively.
------------------	--



## Parameters

<i>duration</i>	A list of durations in units of $\mu$ s.
<i>destType</i>	specifies wheather the data is for syncout, current or voltage stimulation.

**11.107.3.8 QueryTriggerstatus()** `StgStatusNet ^ QueryTriggerstatus ( )`**11.107.3.9 SendSegmentDefine()** `void SendSegmentDefine ( array< uint32_t >^ segment_list )`

Defines the segment memory layout of the STG.

On reset, the STG has one segment containing all available memory.

With this command, the STG memory can be devided into several segments. Each segment can be filled with stimulus data.

## Parameters

<i>segment_list</i>	The List of memory sizes (one per segment).
---------------------	---

**11.107.3.10 SendSegmentSelect()** `void SendSegmentSelect ( uint32_t segment, Stg200xSegmentFlagsEnumNet segmentflags )`

Switchs segment.

## Parameters

<i>segment</i>	The number of the segment to select.
----------------	--------------------------------------

## Parameters

<i>segmentflags</i>	A bitmap of flags, bit 1: assign all channels to the trigger number equal to the segment.
---------------------	---

**11.107.3.11 SendSegmentStart()** `void SendSegmentStart (`  
`uint32_t triggermap,`  
`uint32_t segment,`  
`Stg200xSegmentFlagsEnumNet segmentflags )`

Switchs segment and starts trigger.

#### Parameters

<i>triggermap</i>	A bitmap of triggers that will be started.
-------------------	--

#### Parameters

<i>segment</i>	The number of the segment to select.
----------------	--------------------------------------

#### Parameters

<i>segmentflags</i>	A bitmap of flags, bit 1: assign all channels to the trigger number equal to the segment.
---------------------	---

**11.107.3.12 SetOutputMap()** `void SetOutputMap (`  
`uint32_t ChannelLayout [ ] )`

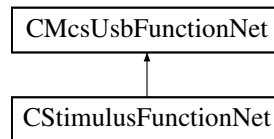
## 11.107.4 Event Documentation

**11.107.4.1 MwPollStatusEvent** [OnMwPollStatus](#)<sup>^</sup> MwPollStatusEvent [add], [remove], [raise]

**11.107.4.2 Stg200xPollStatusEvent** [OnStgPollStatus](#)<sup>^</sup> Stg200xPollStatusEvent [add], [remove], [raise]

## 11.108 CStimulusFunctionNet Class Reference

Inheritance diagram for CStimulusFunctionNet:



### Classes

- class [SidebandData](#)
- class [StimulusDeviceDataAndUnrolledData](#)

### Public Member Functions

- [CStimulusFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> stimulusFunctionPointerContainer)
- [CStimulusFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- void [StartPoll](#) ()  
*Starts the interrupt fetching thread and delivers events*
- void [StopPoll](#) ()  
*Stops the interrupt fetching thread and delivers events*
- void [ForceStatusEvent](#) ()  
*Force a status event. Force the DLL to create a PollMessage event and to call the pPollCallback function, even if no new status information is available.*
- void [SendStart](#) (uint32\_t triggermap)  
*Start (Trigger) the STG. The startup delay is in the range of a few ms.*
- void [SendStop](#) (uint32\_t triggermap)  
*Stop some or all triggers of the STG.*
- void [SendStop](#) (uint32\_t triggermap, int options)  
*Stop some or all triggers of the STG.*
- void [ClearChannelData](#) (int channel)  
*Delete a Stimulus Pattern from STG memory*
- void [ClearSyncData](#) (int channel)  
*Delete a Syncout Pattern from STG memory*
- void [PrepareAndSendData](#) (uint32\_t channel, array< int32\_t ><sup>^</sup> amplitude, array< uint64\_t ><sup>^</sup> duration, STG\_DestinationEnumNet destType)  
*Prepare and send data to a given channel on the STG. Previous data sent to that channel is erased first.*
- void [PrepareAndAppendData](#) (uint32\_t channel, array< int32\_t ><sup>^</sup> amplitude, array< uint64\_t ><sup>^</sup> duration, STG\_DestinationEnumNet destType)  
*Prepare and append data to a given channel on the STG.*
- void [ClearChannel\\_PrepareAndSendData](#) (uint32\_t channel, array< int32\_t ><sup>^</sup> amplitude, array< uint64\_t ><sup>^</sup> duration, STG\_DestinationEnumNet destType, bool doClear)
- [StimulusDeviceDataAndUnrolledData](#)<sup>^</sup> [PrepareData](#) (int channel, array< int32\_t ><sup>^</sup> amplitude, array< uint64\_t ><sup>^</sup> duration, STG\_DestinationEnumNet destType)
- void [SendPreparedData](#) (int channel, [StimulusDeviceDataAndUnrolledData](#)<sup>^</sup> device\_data\_and\_unrolled, STG\_DestinationEnumNet destType)
- [SidebandData](#)<sup>^</sup> [CreateSideband](#) (array< int32\_t ><sup>^</sup> StimulusActive, array< int32\_t ><sup>^</sup> Syncout, array< uint64\_t ><sup>^</sup> Duration, uint32\_t Bit0Time, uint32\_t Bit3Time, uint32\_t Bit4Time)

- Creates the Sideband Channel for the MEA2100 device.*

  - void [ClearMultiplexedData](#) ()
- Clears the Stimulation Memory in the STG device.*

  - void [SendMultiplexedData](#) (array< uint16\_t >^ data)

*Sends stimulus data in multiplexed form. All 16 bits words for the enabled DAC and digital channels are muxed together per time slice.*
- int [GetMultiplexedDataChannelsInBlock](#) ()

*Gets the number of stimulus data channels to send per time slice. Might be greater than the number of configured channels. Fill unused channels with dummy data in SendMultiplexedData*
- int [GetDACResolution](#) ()

*Gets number of bits of the DAC resolution.*
- int [GetVoltageRangeInMicroVolt](#) (uint32\_t channel)

*Gets the Voltage Range of the specified channel in Microvolts.*
- int [GetVoltageResolutionInMicroVolt](#) (uint32\_t channel)

*Gets the Voltage Resolution of the specified channel in Microvolts.*
- int [GetCurrentRangeInNanoAmp](#) (uint32\_t channel)

*Gets the Current Range of the specified channel in Nanoamps.*
- int [GetCurrentResolutionInNanoAmp](#) (uint32\_t channel)

*Gets the Current Resolution of the specified channel in Nanoamps.*
- void [SetupTrigger](#) (uint32\_t first\_trigger, array< uint32\_t >^ channelmap, array< uint32\_t >^ syncoutmap, array< uint32\_t >^ repeat)

*Configures the trigger settings for the STG. Note that all memory segments have their own trigger setting.*
- void [SetupTriggerSingle](#) (uint32\_t trigger, uint32\_t channelmap, uint32\_t syncoutmap, uint32\_t repeat)

*Configures the trigger settings for the STG. Note that all memory segments have their own trigger setting.*
- uint32\_t [GetTotalMemory](#) ()

*Get the total amount of memory available on the STG (all segments).*
- uint32\_t [GetAvailableMemory](#) ()

*Get the amount of memory available in the currently selected segment of the STG.*
- int [GetNumberOfAnalogChannels](#) ()

*Get the number of STG channels.*

## Events

- [OnStgPollStatus](#)^ [PollStatusEvent](#)

## Additional Inherited Members

### 11.108.1 Constructor & Destructor Documentation

**11.108.1.1 CStimulusFunctionNet()** [1/2] [CStimulusFunctionNet](#) ( [CMcsUsbNet](#)^ mcsusb, [CMcsUsbFunctionPointerContainer](#)^ stimulusFunctionPointerContainer )

**11.108.1.2 CStimulusFunctionNet()** [2/2] [CStimulusFunctionNet](#) ( [CMcsUsbNet](#)^ mcsusb )

## 11.108.2 Member Function Documentation

**11.108.2.1 ClearChannel\_PrepareAndSendData()** `void ClearChannel_PrepareAndSendData (`  
     `uint32_t channel,`  
     `array< int32_t >^ amplitude,`  
     `array< uint64_t >^ duration,`  
     `STG_DestinationEnumNet destType,`  
     `bool doClear )`

**11.108.2.2 ClearChannelData()** `void ClearChannelData (`  
     `int channel )`

Delete a Stimulus Pattern from STG memory

### Parameters

<i>channel</i>	specifies the channel to clear.
----------------	---------------------------------

**11.108.2.3 ClearMultiplexedData()** `void ClearMultiplexedData ( )`

Clears the Stimulation Memory in the STG device.

**11.108.2.4 ClearSyncData()** `void ClearSyncData (`  
     `int channel )`

Delete a Syncout Pattern from STG memory

### Parameters

<i>channel</i>	specifies the channel to clear.
----------------	---------------------------------

**11.108.2.5 CreateSideband()** `SidebandData ^ CreateSideband (`  
     `array< int32_t >^ StimulusActive,`  
     `array< int32_t >^ Syncout,`  
     `array< uint64_t >^ Duration,`  
     `uint32_t Bit0Time,`  
     `uint32_t Bit3Time,`  
     `uint32_t Bit4Time )`

Creates the Sideband Channel for the MEA2100 device.

Each datapoint is represented by an signed 32bit integer value. A value 0 means that the stimulation is active during that time. A value 1 means that the stimulation is not active during that time.

The duration is given as a list of 64 bit integers. Durations are given in units of  $\mu\text{s}$ . The STG has a resolution of 20  $\mu\text{s}$ .

#### Parameters

<i>StimulusActive</i>	A list of datapoints which define weather the Stimulus is active or idle at that time as int32.
-----------------------	---

#### Parameters

<i>Duration</i>	A list of durations as uint64. The time is given in units of $\mu\text{s}$ .
<i>Bit0Time</i>	Time in $\mu\text{s}$ for which Bit 0 (Blanking) is to be extended.

#### Parameters

<i>Bit3Time</i>	Time in $\mu\text{s}$ for which Bit 3 (Stimulus Enable) is to be extended.
-----------------	--

#### Parameters

<i>Bit4Time</i>	Time in $\mu\text{s}$ for which Bit 4 (Stimulus Selector) is to be extended.
-----------------	--

#### Returns

Error Status. 0 on success.

#### 11.108.2.6 ForceStatusEvent() `void ForceStatusEvent ( )`

Force a status event. Force the DLL to create a PollMessage event and to call the pPollCallback function, even if no new status information is available.

**11.108.2.7 GetAvailableMemory()** `uint32_t GetAvailableMemory ( )`

Get the amount of memory available in the currently selected segment of the STG.

**Returns**

The total memory available on the STG in bytes.

**11.108.2.8 GetCurrentRangeInNanoAmp()** `int GetCurrentRangeInNanoAmp (   
uint32_t channel )`

Gets the Current Range of the specified channel in Nanoamps.

**Parameters**

<i>channel</i>	Channel which is queried.
----------------	---------------------------

**Returns**

The Current Range of the specified channel in Nanoamps.

**11.108.2.9 GetCurrentResolutionInNanoAmp()** `int GetCurrentResolutionInNanoAmp (   
uint32_t channel )`

Gets the Current Resolution of the specified channel in Nanoamps.

**Parameters**

<i>channel</i>	Channel which is queried.
----------------	---------------------------

**Returns**

The Current Resolution of the specified channel in Nanoamps.

**11.108.2.10 GetDACResolution()** `int GetDACResolution ( )`

Gets number of bits of the DAC resolution.

**Returns**

The DAC resolution in bits.

**11.108.2.11 GetMultiplexedDataChannelsInBlock()** `int GetMultiplexedDataChannelsInBlock ( )`

Gets the number of stimulus data channels to send per time slice. Might be greater than the number of configured channels. Fill unused channels with dummy data in `SendMultiplexedData`

**11.108.2.12 GetNumberOfAnalogChannels()** `int GetNumberOfAnalogChannels ( )`

Get the number of STG channels.

**Returns**

The number of STG channels.

**11.108.2.13 GetTotalMemory()** `uint32_t GetTotalMemory ( )`

Get the total amount of memory available on the STG (all segments).

**Returns**

The total memory available on the STG in bytes.

**11.108.2.14 GetVoltageRangeInMicroVolt()** `int GetVoltageRangeInMicroVolt (   
uint32_t channel )`

Gets the Voltage Range of the specified channel in Microvolts.

**Parameters**

<i>channel</i>	Channel which is queried.
----------------	---------------------------

**Returns**

The Voltage Range of the specified channel in Microvolts.

**11.108.2.15 GetVoltageResolutionInMicroVolt()** `int GetVoltageResolutionInMicroVolt (   
uint32_t channel )`

Gets the Voltage Resolution of the specified channel in Microvolts.



## Parameters

<i>channel</i>	Channel which is queried.
----------------	---------------------------

## Returns

The Voltage Resolution of the specified channel in Microvolts.

**11.108.2.16 PrepareAndAppendData()** `void PrepareAndAppendData (`  
`uint32_t channel,`  
`array< int32_t >^ amplitude,`  
`array< uint64_t >^ duration,`  
`STG_DestinationEnumNet destType )`

Prepare and append data to a given channel on the STG.

Each datapoint is represented by an signed 32bit integer value. When using voltage stimulation, the values are in multiple of 1 uV, thus the possible range is  $\pm 2000$  V. When using current stimulation, the values are in multiple of 1 nA, this the possible range is  $\pm 2000$  mA.

The duration is given as a list of 64 bit integers. Durations are given in units of  $\mu$ s. The STG has a resolution of 20  $\mu$ s.

Blocks of data which should repeat can be defined by prepending such a block with an entry in the arrays where both amplitude and duration is zero. The end of such an block is marked by an entry where the duration is set to zero and the amplitude beeing set to the number of times the block should run. Blocks can be nested.

## Parameters

<i>channel</i>	The channel number to send data to.
----------------	-------------------------------------

## Parameters

<i>amplitude</i>	A list of amplitudes in units of $\mu$ V and nA in voltage and current mode, respectively.
------------------	--

## Parameters

<i>duration</i>	A list of durations in units of $\mu$ s.
<i>destType</i>	specifies wheather the data is for syncout, current or voltage stimulation.

**Returns**

Error Status. 0 on success.

**11.108.2.17 PrepareAndSendData()** `void PrepareAndSendData (`  
`uint32_t channel,`  
`array< int32_t >^ amplitude,`  
`array< uint64_t >^ duration,`  
`STG_DestinationEnumNet destType )`

Prepare and send data to a given channel on the STG. Previous data sent to that channel is erased first.

Each datapoint is represented by an signed 32bit integer value. When using voltage stimulation, the values are in multiple of 1 uV, thus the possible range is += 2000 V. When using current stimulation, the values are in multiple of 1 nA, this the possible range is += 2000 mA.

The duration is given as a list of 64 bit integers. Durations are given in units of  $\mu$ s. The STG has a resolution of 20  $\mu$ s.

Blocks of data which should repeat can be defined by prepending such a block with an entry in the arrays where both amplitude and duration is zero. The end of such an block is marked by an entry where the duration is set to zero and the amplitude beeing set to the number of times the block should run. Blocks can be nested.

**Parameters**

<i>channel</i>	The channel number to send data to.
----------------	-------------------------------------

**Parameters**

<i>amplitude</i>	A list of amplitudes in units of $\mu$ V and nA in voltage and current mode, respectively.
------------------	--

**Parameters**

<i>duration</i>	A list of durations in units of $\mu$ s.
<i>destType</i>	specifies wheather the data is for syncout, current or voltage stimulation.

**Returns**

Error Status. 0 on success.

**11.108.2.18 PrepareData()** `StimulusDeviceDataAndUnrolledData` ^ PrepareData (   
     int *channel*,  
     array< int32\_t >^ *amplitude*,  
     array< uint64\_t >^ *duration*,  
     STG\_DestinationEnumNet *destType* )

**11.108.2.19 SendMultiplexedData()** void SendMultiplexedData (   
     array< uint16\_t >^ *data* )

Sends stimulus data in multiplexed form. All 16 bits words for the enabled DAC and digital channels are muxed together per time slice.

#### Parameters

<i>data</i>	Array of data to be sent.
-------------	---------------------------

**11.108.2.20 SendPreparedData()** void SendPreparedData (   
     int *channel*,  
     `StimulusDeviceDataAndUnrolledData`^ *device\_data\_and\_unrolled*,  
     STG\_DestinationEnumNet *destType* )

**11.108.2.21 SendStart()** void SendStart (   
     uint32\_t *triggermap* )

Start (Trigger) the STG. The startup delay is in the range of a few ms.

#### Parameters

<i>triggermap</i>	A bitmap of triggers which will be started.
-------------------	---

**11.108.2.22 SendStop()** [1/2] void SendStop (   
     uint32\_t *triggermap* )

Stop some or all triggers of the STG.

#### Parameters

<i>triggermap</i>	A bitmap of triggers which will be stopped.
-------------------	---

**11.108.2.23 SendStop()** [2/2] `void SendStop (`  
`uint32_t triggermap,`  
`int options )`

Stop some or all triggers of the STG.

#### Parameters

<i>triggermap</i>	A bitmap of triggers which will be stopped.
<i>options</i>	bitmap of options, currently only STOP_OPTION_SAVESTOP (0x80) is defined, which bypasses the stop commands when a syncout associated with a given sync-out has bit 1 (0x02) set. Can be used e.g. to prevent a stop while a biphasic stimulation pulse is active..

**11.108.2.24 SetupTrigger()** `void SetupTrigger (`  
`uint32_t first_trigger,`  
`array< uint32_t >^ channelmap,`  
`array< uint32_t >^ syncoutmap,`  
`array< uint32_t >^ repeat )`

Configures the trigger settings for the STG. Note that all memory segments have their own trigger setting.

#### Parameters

<i>first_trigger</i>	The number of the first trigger to change.
----------------------	--

#### Parameters

<i>channelmap</i>	For each trigger, a bitmap of channels that belong to this trigger.
-------------------	---

#### Parameters

<i>syncoutmap</i>	For each trigger, a bitmap of syncouts that belong to this trigger.
<i>repeat</i>	For each trigger, define the number of times this trigger should be repeated.

**11.108.2.25 SetupTriggerSingle()** `void SetupTriggerSingle (`  
`uint32_t trigger,`  
`uint32_t channelmap,`  
`uint32_t syncoutmap,`  
`uint32_t repeat )`

Configures the trigger settings for the STG. Note that all memory segments have their own trigger setting.

## Parameters

<i>trigger</i>	The trigger to change.
----------------	------------------------

## Parameters

<i>channelmap</i>	A bitmap of channels that belong to this trigger.
-------------------	---

## Parameters

<i>syncoutmap</i>	A bitmap of syncouts that belong to this trigger.
<i>repeat</i>	The number of times this trigger should be repeated.

**11.108.2.26 StartPoll()** `void StartPoll ( )`

Starts the interrupt fetching thread and delivers events

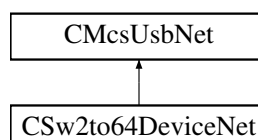
**11.108.2.27 StopPoll()** `void StopPoll ( )`

Stops the interrupt fetching thread and delivers events

**11.108.3 Event Documentation****11.108.3.1 PollStatusEvent** `OnStgPollStatus^ PollStatusEvent`**11.109 CSw2to64DeviceNet Class Reference**

The class to control the MCS-USB-Sw2to64 device.

Inheritance diagram for CSw2to64DeviceNet:



**Public Member Functions**

- [CSw2to64DeviceNet](#) ()
- [~CSw2to64DeviceNet](#) ()
- unsigned short [GetNumber](#) ()  
*Gets the number of channels that can be switched in this box.*
- array< unsigned char > ^ [GetChannels](#) ()  
*Gets the current switch positions as char array.*
- void [SetChannels](#) (array< unsigned char > ^ pattern)  
*Sets the switch positions from a char array.*
- unsigned char [GetChannel](#) (unsigned short index)  
*Gets one current switch position.*
- void [SetChannel](#) (unsigned short index, unsigned char pattern)  
*Sets one switch position.*

**Additional Inherited Members****11.109.1 Detailed Description**

The class to control the MCS-USB-Sw2to64 device.

This class controls the settings of the MCS-USB-Sw2to64. The box has two inputs for signals. Each of the 64 outputs can be connected to one of the input signals, could be held open or connected ground. Valid switch states are 0, 1, 2 or 3 for each of the settings.

**11.109.2 Constructor & Destructor Documentation**

**11.109.2.1** [CSw2to64DeviceNet\(\)](#) [CSw2to64DeviceNet](#) ( )

**11.109.2.2** [~CSw2to64DeviceNet\(\)](#) [~CSw2to64DeviceNet](#) ( )

**11.109.3 Member Function Documentation**

**11.109.3.1** [GetChannel\(\)](#) unsigned char [GetChannel](#) (  
unsigned short *index* )

Gets one current switch position.

**Parameters**

in	<i>index</i>	number of channel to read the switch position from
----	--------------	--

**Returns**

switch position of desired channel

**11.109.3.2 GetChannels()** `array<unsigned char> ^ GetChannels ( )`

Gets the current switch positions as char array.

**Returns**

array of char with the size of the number of channels, each char has the setting of a channel

**11.109.3.3 GetNumber()** `unsigned short GetNumber ( )`

Gets the number of channels that can be switched in this box.

The box can have a different number of channels it can switch. Up to now usually 64 channels are returned

**11.109.3.4 SetChannel()** `void SetChannel (   
    unsigned short index,   
    unsigned char pattern )`

Sets one switch position.

**Parameters**

in	<i>index</i>	number of channel to write the switch position to
in	<i>pattern</i>	switch position of the channel

**11.109.3.5 SetChannels()** `void SetChannels (   
    array< unsigned char >^ pattern )`

Sets the switch positions from a char array.

**Parameters**

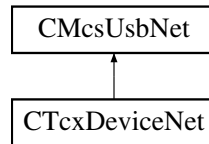
in	<i>pattern</i>	array of char with the size of the number of channels, each char has the setting of a channel
----	----------------	---



## 11.110 CTcxDeviceNet Class Reference

Class to control a Temperature Controller (TCX)

Inheritance diagram for CTcxDeviceNet:



### Public Member Functions

- [CTcxDeviceNet](#) ()  
*Initializes a new instance of [CTcxDeviceNet](#) class.*
- [~CTcxDeviceNet](#) ()
- unsigned int [GetNumControlChannels](#) ()  
*Gets the number of channels the device can control/regulate.*
- unsigned int [GetNumMeasureChannels](#) ()  
*Gets the number of channels the device can measure.*
- int [GetValue](#) (unsigned int channel)  
*Gets the temperate of the specified channel in units of 0.1 °C.*
- int [GetValueHires](#) (unsigned int channel)  
*Gets the temperate of the specified channel in units of 0.01 °C.*
- int [GetHeaterTemp](#) (unsigned int channel)  
*Gets the temperate of the specified heater in units of 0.1 °C.*
- int [GetHeaterLimit](#) (unsigned int device)  
*Gets the temperate limit of the specified heater in units of 0.1 °C.*
- double [GetMaxHeaterPowerMultiwell](#) ()  
*queries the max. heater power that the Multiwell temperature controller will apply; unit: W; useful range: 5.2W..7.6W*
- void [SetMaxHeaterPowerMultiwell](#) (double MaxPowerWatt)  
*sets the max. heater power that the Multiwell temperature controller will apply; unit: W; useful range: 5.2W..7.6W*
- bool [GetHasThermocouple](#) ()  
*Gets weather the device supports a thermocouple.*
- bool [GetEnableHeaterLimit](#) (unsigned int device)
- bool [GetEnableThermocouple](#) (unsigned int device)
- TcxSensorTypeEnumNet [GetSensorType](#) (unsigned int device)
- String ^ [GetUnit](#) (unsigned int channel)
- unsigned int [GetBoardTemp](#) ()  
*Gets the temperate of the mainboard in units of 0.1 °C.*
- unsigned int [GetVolts](#) (unsigned int channel)
- unsigned int [GetNumDevices](#) ()
- void [SetSetpoint](#) (unsigned int channel, int sp)  
*Sets the target temperate of specified channel in units of 0.1 °C.*
- void [SetDevice](#) (unsigned int channel, int device)
- void [SetOnOff](#) (unsigned int channel, bool on)  
*Switches the specified channel on or off.*
- void [SetCalibration](#) (unsigned int channel, int calib)
- void [SetP](#) (unsigned int device, int p\_coeff)  
*Sets the P-coefficient of the specified device.*

- void [SetI](#) (unsigned int device, int i\_coeff)  
*Sets the I-coefficient of the specified device.*
- void [SetD](#) (unsigned int device, int d\_coeff)  
*Sets the D-coefficient of the specified device.*
- void [SetMaxP](#) (unsigned int device, int maxp)  
*Sets the maximum heater power of the specified device.*
- void [SetHeaterLimit](#) (unsigned int device, int heater\_limit)
- void [SetEnableHeaterLimit](#) (unsigned int device, bool enable)
- void [SetEnableThermocouple](#) (unsigned int device, bool enable)
- void [SetSensorType](#) (unsigned int device, TcxSensorTypeEnumNet type)
- void [SetDevname](#) (unsigned int device, String^ Devicename)
- int [GetSetpoint](#) (unsigned int channel)  
*Gets the target temperate of specified channel in units of 0.1 °C.*
- int [GetDevice](#) (unsigned int channel)
- int [GetOnOff](#) (unsigned int channel)  
*Gets if the specified channel is on or off.*
- int [GetCalibration](#) (unsigned int channel)
- int [GetP](#) (unsigned int device)  
*Gets the P-coefficient of the specified device.*
- int [GetI](#) (unsigned int device)  
*Gets the I-coefficient of the specified device.*
- int [GetD](#) (unsigned int device)  
*Gets the D-coefficient of the specified device.*
- int [GetMaxP](#) (unsigned int device)  
*Gets the maximum heater power of the specified device.*
- String ^ [GetDevname](#) (unsigned int device)
- TcxDeviceTypeEnumNet [GetDeviceType](#) ()
- int [GetSetpointMin](#) (unsigned int channel)
- int [GetCalibrationMin](#) (unsigned int channel)
- int [GetPMin](#) (unsigned int device)
- int [GetIMin](#) (unsigned int device)
- int [GetDMin](#) (unsigned int device)
- int [GetMaxpMin](#) (unsigned int device)
- int [GetSetpointMax](#) (unsigned int channel)
- int [GetCalibrationMax](#) (unsigned int channel)
- int [GetPMax](#) (unsigned int device)
- int [GetIMax](#) (unsigned int device)
- int [GetDMax](#) (unsigned int device)
- int [GetMaxpMax](#) (unsigned int device)
- int [GetSetpointDecp](#) (unsigned int channel)
- int [GetCalibrationDecp](#) (unsigned int channel)
- int [GetPDecp](#) (unsigned int device)
- int [GetIDecp](#) (unsigned int device)
- int [GetDDecp](#) (unsigned int device)
- int [GetMaxpDecp](#) (unsigned int device)
- int [GetResX](#) (unsigned int channel)
- int [GetResS](#) (unsigned int channel)
- int [GetRes1](#) (unsigned int channel)
- int [GetRes2](#) (unsigned int channel)
- int [GetPwrSet](#) (unsigned int channel)
- int [GetPwrOut](#) (unsigned int channel)
- int [GetDuty](#) (unsigned int channel)  
*Gets the duty cycle of the heating element.*

- int [GetUOut](#) (unsigned int channel)  
*Gets the voltage on the heating element.*
- int [GetIOut](#) (unsigned int channel)  
*Gets the current through the heating element.*
- int [GetROut](#) (unsigned int channel)  
*Gets the resistance of the heating element.*
- int [GetPOut](#) (unsigned int channel)  
*Gets the output power of the heating element.*
- int [GetCurrent](#) (unsigned int channel)
- int [GetThermocoupleTemp](#) (unsigned int channel)
- int [GetThermocoupleTempAbs](#) (unsigned int channel)
- int [GetThermocoupleReferenceTemp](#) (unsigned int channel)
- unsigned int [GetThermocoupleNanovoltPerKelvin](#) (unsigned int channel)  
*Gets the proportional constant for the thermocouple.*
- void [SetThermocoupleNanovoltPerKelvin](#) (unsigned int channel, unsigned int value)  
*Sets the proportional constant for the thermocouple.*
- int [GetThermocoupleCalibration](#) (unsigned int channel)
- void [CalibrateThermocouple](#) (unsigned int channel)
- void [SetDeviceType](#) (TcxDeviceTypeEnumNet devicetype)
- void [FactoryReset](#) ()

## Additional Inherited Members

### 11.110.1 Detailed Description

Class to control a Temperature Controller (TCX)

### 11.110.2 Constructor & Destructor Documentation

#### 11.110.2.1 CTcxDeviceNet() [CTcxDeviceNet](#) ( )

Initializes a new instance of [CTcxDeviceNet](#) class.

#### 11.110.2.2 ~CTcxDeviceNet() [~CTcxDeviceNet](#) ( )

### 11.110.3 Member Function Documentation

#### 11.110.3.1 CalibrateThermocouple() [void CalibrateThermocouple](#) ( unsigned int *channel* )

**11.110.3.2 FactoryReset()** `void FactoryReset ( )`

**11.110.3.3 GetBoardTemp()** `unsigned int GetBoardTemp ( )`

Gets the temperate of the mainboard in units of 0.1 °C.

**11.110.3.4 GetCalibration()** `int GetCalibration (`  
`unsigned int channel )`

**11.110.3.5 GetCalibrationDecp()** `int GetCalibrationDecp (`  
`unsigned int channel )`

**11.110.3.6 GetCalibrationMax()** `int GetCalibrationMax (`  
`unsigned int channel )`

**11.110.3.7 GetCalibrationMin()** `int GetCalibrationMin (`  
`unsigned int channel )`

**11.110.3.8 GetCurrent()** `int GetCurrent (`  
`unsigned int channel )`

**11.110.3.9 GetD()** `int GetD (`  
`unsigned int device )`

Gets the D-coefficient of the specified device.

**11.110.3.10 GetDDecp()** `int GetDDecp (`  
`unsigned int device )`

**11.110.3.11 GetDevice()** `int GetDevice (`  
`unsigned int channel )`

**11.110.3.12 GetDeviceType()** `TcxDeviceTypeEnumNet GetDeviceType ( )`

**11.110.3.13 GetDevname()** `String ^ GetDevname (`  
`unsigned int device )`

**11.110.3.14 GetDMax()** `int GetDMax (`  
`unsigned int device )`

**11.110.3.15 GetDMin()** `int GetDMin (`  
`unsigned int device )`

**11.110.3.16 GetDuty()** `int GetDuty (`  
`unsigned int channel )`

Gets the duty cycle of the heating element.

#### Parameters

<i>channel</i>	The channel number.
----------------	---------------------

#### Returns

The duty cycle in percent, the value of  $320 * 64$  corresponds to 100 %.

**11.110.3.17 GetEnableHeaterLimit()** `bool GetEnableHeaterLimit (`  
`unsigned int device )`

**11.110.3.18 GetEnableThermocouple()** `bool GetEnableThermocouple (`  
`unsigned int device )`

**11.110.3.19 GetHasThermocouple()** `bool GetHasThermocouple ( )`

Gets whether the device supports a thermocouple.

**11.110.3.20 GetHeaterLimit()** `int GetHeaterLimit (   
 unsigned int device )`

Gets the temperature limit of the specified heater in units of 0.1 °C.

**11.110.3.21 GetHeaterTemp()** `int GetHeaterTemp (   
 unsigned int channel )`

Gets the temperature of the specified heater in units of 0.1 °C.

**11.110.3.22 GetI()** `int GetI (   
 unsigned int device )`

Gets the I-coefficient of the specified device.

**11.110.3.23 GetIDecp()** `int GetIDecp (   
 unsigned int device )`**11.110.3.24 GetIMax()** `int GetIMax (   
 unsigned int device )`**11.110.3.25 GetIMin()** `int GetIMin (   
 unsigned int device )`**11.110.3.26 GetIOut()** `int GetIOut (   
 unsigned int channel )`

Gets the current through the heating element.

## Parameters

<i>channel</i>	The channel number.
----------------	---------------------

## Returns

The current in units of mA.

**11.110.3.27 GetMaxHeaterPowerMultiwell()** `double GetMaxHeaterPowerMultiwell ( )`

queries the max. heater power that the Multiwell temperature controller will apply; unit: W; useful range: 5.2W..7.6W

**11.110.3.28 GetMaxP()** `int GetMaxP (   
 unsigned int device )`

Gets the maximum heater power of the specified device.

**11.110.3.29 GetMaxpDecp()** `int GetMaxpDecp (   
 unsigned int device )`

**11.110.3.30 GetMaxpMax()** `int GetMaxpMax (   
 unsigned int device )`

**11.110.3.31 GetMaxpMin()** `int GetMaxpMin (   
 unsigned int device )`

**11.110.3.32 GetNumControlChannels()** `unsigned int GetNumControlChannels ( )`

Gets the number of channels the device can control/regulate.

**11.110.3.33 GetNumDevices()** `unsigned int GetNumDevices ( )`

**11.110.3.34 GetNumMeasureChannels()** `unsigned int GetNumMeasureChannels ( )`

Gets the number of channels the device can measure.

**11.110.3.35 GetOnOff()** `int GetOnOff (   
 unsigned int channel )`

Gets if the specified channel is on or off.

**11.110.3.36 GetP()** `int GetP (   
 unsigned int device )`

Gets the P-coefficient of the specified device.

**11.110.3.37 GetPDecp()** `int GetPDecp (   
 unsigned int device )`**11.110.3.38 GetPMax()** `int GetPMax (   
 unsigned int device )`**11.110.3.39 GetPMin()** `int GetPMin (   
 unsigned int device )`**11.110.3.40 GetPOut()** `int GetPOut (   
 unsigned int channel )`

Gets the output power of the heating element.

**Parameters**

<i>channel</i>	The channel number.
----------------	---------------------

**Returns**

The resistance in units of mW.



**11.110.3.41 GetPwrOut()** `int GetPwrOut ( unsigned int channel )`

**11.110.3.42 GetPwrSet()** `int GetPwrSet ( unsigned int channel )`

**11.110.3.43 GetRes1()** `int GetRes1 ( unsigned int channel )`

**11.110.3.44 GetRes2()** `int GetRes2 ( unsigned int channel )`

**11.110.3.45 GetResS()** `int GetResS ( unsigned int channel )`

**11.110.3.46 GetResX()** `int GetResX ( unsigned int channel )`

**11.110.3.47 GetROut()** `int GetROut ( unsigned int channel )`

Gets the resistance of the heating element.

#### Parameters

<i>channel</i>	The channel number.
----------------	---------------------

#### Returns

The resistance in units of 0.1 Ohm.

**11.110.3.48 GetSensorType()** `TcxSensorTypeEnumNet GetSensorType ( unsigned int device )`

**11.110.3.49 GetSetpoint()** `int GetSetpoint ( unsigned int channel )`

Gets the target temperate of specified channel in units of 0.1 °C.

**11.110.3.50 GetSetpointDecp()** `int GetSetpointDecp ( unsigned int channel )`

**11.110.3.51 GetSetpointMax()** `int GetSetpointMax ( unsigned int channel )`

**11.110.3.52 GetSetpointMin()** `int GetSetpointMin ( unsigned int channel )`

**11.110.3.53 GetThermocoupleCalibration()** `int GetThermocoupleCalibration ( unsigned int channel )`

**11.110.3.54 GetThermocoupleNanovoltPerKelvin()** `unsigned int GetThermocoupleNanovoltPerKelvin ( unsigned int channel )`

Gets the proportional constant for the thermocouple.

#### Parameters

<i>channel</i>	Thermocouple channel number.
----------------	------------------------------

#### Returns

The proportional constant in Nanovolt per Kelvin.

**11.110.3.55 GetThermocoupleReferenceTemp()** `int GetThermocoupleReferenceTemp ( unsigned int channel )`

**11.110.3.56 GetThermocoupleTemp()** `int GetThermocoupleTemp ( unsigned int channel )`

**11.110.3.57 GetThermocoupleTempAbs()** `int GetThermocoupleTempAbs ( unsigned int channel )`

**11.110.3.58 GetUnit()** `String ^ GetUnit ( unsigned int channel )`

**11.110.3.59 GetUOut()** `int GetUOut ( unsigned int channel )`

Gets the voltage on the heating element.

#### Parameters

<i>channel</i>	The channel number.
----------------	---------------------

#### Returns

The voltage in units of mV.

**11.110.3.60 GetValue()** `int GetValue ( unsigned int channel )`

Gets the temperate of the specified channel in units of 0.1 °C.

**11.110.3.61 GetValueHires()** `int GetValueHires ( unsigned int channel )`

Gets the temperate of the specified channel in units of 0.01 °C.

**11.110.3.62 GetVolti()** `unsigned int GetVolti ( unsigned int channel )`

**11.110.3.63 SetCalibration()** void SetCalibration (   
 unsigned int *channel*,   
 int *calib* )

**11.110.3.64 SetD()** void SetD (   
 unsigned int *device*,   
 int *d\_coeff* )

Sets the D-coefficient of the specified device.

**11.110.3.65 SetDevice()** void SetDevice (   
 unsigned int *channel*,   
 int *device* )

**11.110.3.66 SetDeviceType()** void SetDeviceType (   
 TcxDeviceTypeEnumNet *devicetype* )

**11.110.3.67 SetDevname()** void SetDevname (   
 unsigned int *device*,   
 String^ *Devicename* )

**11.110.3.68 SetEnableHeaterLimit()** void SetEnableHeaterLimit (   
 unsigned int *device*,   
 bool *enable* )

**11.110.3.69 SetEnableThermocouple()** void SetEnableThermocouple (   
 unsigned int *device*,   
 bool *enable* )

**11.110.3.70 SetHeaterLimit()** void SetHeaterLimit (   
 unsigned int *device*,   
 int *heater\_limit* )

**11.110.3.71 SetI()** void SetI (   
 unsigned int *device*,   
 int *i\_coeff* )

Sets the I-coefficient of the specified device.

**11.110.3.72 SetMaxHeaterPowerMultiwell()** void SetMaxHeaterPowerMultiwell (   
 double *MaxPowerWatt* )

sets the max. heater power that the Multiwell temperature controller will apply; unit: W; useful range: 5.2W..7.6W

**11.110.3.73 SetMaxP()** void SetMaxP (   
 unsigned int *device*,   
 int *maxp* )

Sets the maximum heater power of the specified device.

**11.110.3.74 SetOnOff()** void SetOnOff (   
 unsigned int *channel*,   
 bool *on* )

Switches the specified channel on or off.

#### Parameters

<i>channel</i>	The channel number.
----------------	---------------------

**11.110.3.75 SetP()** void SetP (   
 unsigned int *device*,   
 int *p\_coeff* )

Sets the P-coefficient of the specified device.

**11.110.3.76 SetSensorType()** void SetSensorType (   
 unsigned int *device*,   
 TcxSensorTypeEnumNet *type* )

**11.110.3.77 SetSetpoint()** `void SetSetpoint (`  
     `unsigned int channel,`  
     `int sp )`

Sets the target temperate of specified channel in units of 0.1 °C.

**11.110.3.78 SetThermocoupleNanovoltPerKelvin()** `void SetThermocoupleNanovoltPerKelvin (`  
     `unsigned int channel,`  
     `unsigned int value )`

Sets the proportional constant for the thermocouple.

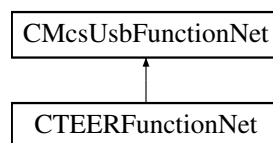
#### Parameters

<i>channel</i>	Thermocouple channel number.
<i>value</i>	Proportional constant in Nanovolt per Kelvin.

## 11.111 CTEERFunctionNet Class Reference

[CTEERFunctionNet](#) is the class to control the TEER device

Inheritance diagram for CTEERFunctionNet:



### Public Member Functions

- [CTEERFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> pTEERFunctionPointerContainer)  
*Initializes a new instance of the [CTEERFunctionNet](#) class.*
- [CTEERFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- virtual [~CTEERFunctionNet](#) ()
- [!CTEERFunctionNet](#) ()
- `uint32_t` [GetPeriod\\_us](#) ()  
*gets the period of TEER stimulation in us*
- void [SetPeriod\\_us](#) (`uint32_t` period\_us)  
*sets the period of TEER stimulation in us*
- `uint32_t` [GetAmplitude\\_nA](#) ()  
*gets TEER stimulation amplitude in nA*
- void [SetAmplitude\\_nA](#) (`uint32_t` Amplitude\_nA)  
*sets TEER stimulation amplitude in nA*
- TeerWaveformEnumNet [GetWaveform](#) ()  
*gets TEER stimulation waveform (sine/rect)*

- void [SetWaveform](#) (TeerWaveformEnumNet Waveform)  
*sets TEER stimulation waveform (sine/rect)*
- TeerClampModeEnumNet [GetClampMode](#) ()  
*gets TEER clamp mode (voltage/current)*
- void [SetClampMode](#) (TeerClampModeEnumNet ClampMode)  
*sets TEER clamp mode (voltage/current)*
- void [StartSampling](#) (uint32\_t NumberOfCycles)  
*starts TEER stimulation (duration: n cycles) and samples during last cycle*
- void [StopSampling](#) ()  
*stops TEER stimulation and sampling*
- uint32\_t [IsSamplingFinished](#) ()  
*returns false iff stimulation/sampling is going on, otherwise true*
- void [SetControllerParams](#) (uint32\_t P, uint32\_t I, uint32\_t D)  
*sets PID controller parameters for voltage clamp mode*
- void [GetControllerParams](#) ([System::Runtime::InteropServices::Out]uint32\_t% P, [System::Runtime::InteropServices::Out]uint32\_t% I, [System::Runtime::InteropServices::Out]uint32\_t% D)  
*gets PID controller parameters for voltage clamp mode*
- array< int32\_t > ^ [GetSampleBufferChunk](#) (int Buffer\_Length)  
*private function to query max. 100 bytes of sample buffer; called internally*
- array< int32\_t > ^ [GetSampleVoltageBuffer\\_uV](#) (int Buffer\_Length)  
*returns voltage sample buffer (max. 500 values); unit: uV*
- uint32\_t [GetMaxChunkSize\\_Byte](#) ()  
*private function to be called internally only*
- uint32\_t [GetBytesPerSample](#) ()  
*private function to be called internally only*
- uint32\_t [GetNumberOfAvailableSamples](#) ()  
*private function to be called internally only*
- void [SetBufferIndex](#) (uint32\_t NewBufferIndex)  
*pre-selects sample buffer to be tranferred by [GetSampleVoltageBuffer\\_uV\(\)](#)*
- uint32\_t [GetAdapterCode](#) ()
- uint32\_t [GetRotaryPositionCode](#) ()
- void [SetExternalLED](#) (uint32\_t NewState)
- void [SetCurrentEnable](#) (bool NewCurrentEnable)  
*when disabled, no current will flow through chamber*
- bool [GetCurrentEnable](#) ()  
*when disabled, no current will flow through chamber*
- int32\_t [GetUptimeSeconds](#) ()  
*returns time in seconds since device was powered up*
- void [StartInternalCalibration](#) ()  
*starts determination of internal DAC-offset; result is used internally; NON-BLOCKING call*
- bool [IsInternalCalibrationFinished](#) ()  
*queries whether internal calibration has finished*
- int [GetDacZero](#) ()  
*returns DAC-offset (result of internal calibration); use to check for plausibility only*
- void [CancelInternalCalibration](#) ()  
*in case the internal calibration "hangs", this will cancel it*
- void [SetLiquidResistance](#) (int32\_t NewLiquidResistance\_Ohm)
- int32\_t [GetLiquidResistance](#) ()
- int [GetScaleFactorU1](#) ()  
*returns U1 scale factor times 10<sup>6</sup> (result of internal calibration)*
- int [GetScaleFactorU2](#) ()

- returns U2 scale factor times  $10^6$  (result of internal calibration)*  
 • int [GetAdcOffsetU1](#) ()  
*returns ADC offset of U1 channel (result of internal calibration)*
- int [GetAdcOffsetU2](#) ()  
*returns ADC offset of U2 channel (result of internal calibration)*

## Additional Inherited Members

### 11.111.1 Detailed Description

[CTEERFunctionNet](#) is the class to control the TEER device

### 11.111.2 Constructor & Destructor Documentation

**11.111.2.1 CTEERFunctionNet()** [1/2] [CTEERFunctionNet](#) (  
     [CMcsUsbNet](#)<sup>^</sup> *mcsusb*,  
     [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> *pTEERFunctionPointerContainer* )

Initializes a new instance of the [CTEERFunctionNet](#) class.

**11.111.2.2 CTEERFunctionNet()** [2/2] [CTEERFunctionNet](#) (  
     [CMcsUsbNet](#)<sup>^</sup> *mcsusb* )

**11.111.2.3 ~CTEERFunctionNet()** virtual [~CTEERFunctionNet](#) ( ) [virtual]

**11.111.2.4 "!CTEERFunctionNet()** [!CTEERFunctionNet](#) ( )

### 11.111.3 Member Function Documentation

**11.111.3.1 CancelInternalCalibration()** void [CancelInternalCalibration](#) ( )

in case the internal calibration "hangs", this will cancel it



**11.111.3.2 GetAdapterCode()** uint32\_t GetAdapterCode ( )

Returns

**11.111.3.3 GetAdcOffsetU1()** int GetAdcOffsetU1 ( )

returns ADC offset of U1 channel (result of internal calibration)

Returns

**11.111.3.4 GetAdcOffsetU2()** int GetAdcOffsetU2 ( )

returns ADC offset of U2 channel (result of internal calibration)

Returns

**11.111.3.5 GetAmplitude\_nA()** uint32\_t GetAmplitude\_nA ( )

gets TEER stimulation amplitude in nA

Returns

current stimulation amplitude in nA

**11.111.3.6 GetBytesPerSample()** uint32\_t GetBytesPerSample ( )

private function to be called internally only

Returns

**11.111.3.7 GetClampMode()** TeerClampModeEnumNet GetClampMode ( )

gets TEER clamp mode (voltage/current)

Returns

current TEER clamp mode

**11.111.3.8 GetControllerParams()** void GetControllerParams (   
 [System::Runtime::InteropServices::Out] uint32\_t% P,   
 [System::Runtime::InteropServices::Out] uint32\_t% I,   
 [System::Runtime::InteropServices::Out] uint32\_t% D )

gets PID controller parameters for voltage clamp mode

## Parameters

<i>P</i>	
<i>I</i>	
<i>D</i>	

**11.111.3.9 GetCurrentEnable()** `bool GetCurrentEnable ( )`

when disabled, no current will flow through chamber

## Returns

**11.111.3.10 GetDacZero()** `int GetDacZero ( )`

returns DAC-offset (result of internal calibration); use to check for plausibility only

## Returns

**11.111.3.11 GetLiquidResistance()** `int32_t GetLiquidResistance ( )`

## Returns

**11.111.3.12 GetMaxChunkSize\_Byte()** `uint32_t GetMaxChunkSize_Byte ( )`

private function to be called internally only

## Returns

**11.111.3.13 GetNumberOfAvailableSamples()** `uint32_t GetNumberOfAvailableSamples ( )`

private function to be called internally only

Returns

**11.111.3.14 GetPeriod\_us()** `uint32_t GetPeriod_us ( )`

gets the period of TEER stimulation in us

Returns

**11.111.3.15 GetRotaryPositionCode()** `uint32_t GetRotaryPositionCode ( )`

Returns

**11.111.3.16 GetSampleBufferChunk()** `array<int32_t> ^ GetSampleBufferChunk (   
int Buffer_Length )`

private function to query max. 100 bytes of sample buffer; called internally

Parameters

<i>Buffer_Length</i>	The maximal length of Buffer.
----------------------	-------------------------------

Returns

**11.111.3.17 GetSampleVoltageBuffer\_uV()** `array<int32_t> ^ GetSampleVoltageBuffer_uV (   
int Buffer_Length )`

returns voltage sample buffer (max. 500 values); unit: uV

## Parameters

<i>Buffer_Length</i>	The maximal length of Buffer.
----------------------	-------------------------------

## Returns

**11.111.3.18 GetScaleFactorU1()** `int GetScaleFactorU1 ( )`

returns U1 scale factor times  $10^6$  (result of internal calibration)

## Returns

**11.111.3.19 GetScaleFactorU2()** `int GetScaleFactorU2 ( )`

returns U2 scale factor times  $10^6$  (result of internal calibration)

## Returns

**11.111.3.20 GetUptimeSeconds()** `int32_t GetUptimeSeconds ( )`

returns time in seconds since device was powered up

## Returns

**11.111.3.21 GetWaveform()** `TeerWaveformEnumNet GetWaveform ( )`

gets TEER stimulation waveform (sine/rect)

## Returns

waveform enum

**11.111.3.22 IsInternalCalibrationFinished()** `bool IsInternalCalibrationFinished ( )`

queries whether internal calibration has finished

Returns

**11.111.3.23 IsSamplingFinished()** `uint32_t IsSamplingFinished ( )`

returns false iff stimulation/sampling is going on, otherwise true

Returns

**11.111.3.24 SetAmplitude\_nA()** `void SetAmplitude_nA (   
uint32_t Amplitude_nA )`

sets TEER stimulation amplitude in nA

Parameters

<i>Amplitude_nA</i>	new stimulation amplitude in nA
---------------------	---------------------------------

**11.111.3.25 SetBufferIndex()** `void SetBufferIndex (   
uint32_t NewBufferIndex )`

pre-selects sample buffer to be tranferred by [GetSampleVoltageBuffer\\_uV\(\)](#)

Parameters

<i>NewBufferIndex</i>	0 - chamber voltage; 1 - compliance voltage
-----------------------	---

**11.111.3.26 SetClampMode()** `void SetClampMode (   
TeerClampModeEnumNet ClampMode )`

sets TEER clamp mode (voltage/current)

## Parameters

<i>ClampMode</i>	new TEER clamp mode
------------------	---------------------

**11.111.3.27 SetControllerParams()** void SetControllerParams (   
     uint32\_t *P*,   
     uint32\_t *I*,   
     uint32\_t *D* )

sets PID controller parameters for voltage clamp mode

## Parameters

<i>P</i>	
<i>I</i>	
<i>D</i>	

**11.111.3.28 SetCurrentEnable()** void SetCurrentEnable (   
     bool *NewCurrentEnable* )

when disabled, no current will flow through chamber

## Parameters

<i>NewCurrentEnable</i>	
-------------------------	--

**11.111.3.29 SetExternalLED()** void SetExternalLED (   
     uint32\_t *NewState* )

## Parameters

<i>NewState</i>	
-----------------	--

**11.111.3.30 SetLiquidResistance()** void SetLiquidResistance (   
     int32\_t *NewLiquidResistance\_Ohm* )

## Parameters

<i>NewLiquidResistance_Ohm</i>	
--------------------------------	--

**11.111.3.31 SetPeriod\_us()** `void SetPeriod_us (`  
`uint32_t period_us )`

sets the period of TEER stimulation in us

Parameters

<i>period_us</i>	
------------------	--

**11.111.3.32 SetWaveform()** `void SetWaveform (`  
`TeerWaveformEnumNet Waveform )`

sets TEER stimulation waveform (sine/rect)

Parameters

<i>Waveform</i>	waveform enum
-----------------	---------------

**11.111.3.33 StartInternalCalibration()** `void StartInternalCalibration ( )`

starts determination of internal DAC-offset; result is used internally; NON-BLOCKING call

**11.111.3.34 StartSampling()** `void StartSampling (`  
`uint32_t NumberOfCycles )`

starts TEER stimulation (duration: n cycles) and samples during last cycle

Parameters

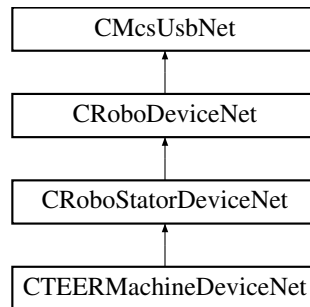
<i>NumberOfCycles</i>	number of cycles (sine or rect) to output (0 - loop forever)
-----------------------	--

**11.111.3.35 StopSampling()** `void StopSampling ( )`

stops TEER stimulation and sampling

## 11.112 CTEERMachineDeviceNet Class Reference

Inheritance diagram for CTEERMachineDeviceNet:



### Public Member Functions

- [CTEERMachineDeviceNet](#) ()
- [~CTEERMachineDeviceNet](#) ()

### Properties

- [CTEERFunctionNet](#)<sup>^</sup> [TEERFunctionNet](#) [get]

### Additional Inherited Members

#### 11.112.1 Constructor & Destructor Documentation

**11.112.1.1** [CTEERMachineDeviceNet](#)() [CTEERMachineDeviceNet](#) ( )

**11.112.1.2** [~CTEERMachineDeviceNet](#)() [~CTEERMachineDeviceNet](#) ( )

#### 11.112.2 Property Documentation

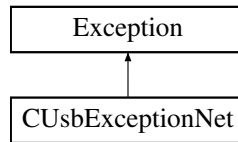
**11.112.2.1** [TEERFunctionNet](#) [CTEERFunctionNet](#)<sup>^</sup> [TEERFunctionNet](#) [get]



## 11.113 CUsbExceptionNet Class Reference

Exception class that is thrown in case of an USB error.

Inheritance diagram for CUsbExceptionNet:



### Public Member Functions

- [CUsbExceptionNet](#) (uint32\_t status)  
*Constructor of a CUsbException.*
- [CUsbExceptionNet](#) (uint32\_t status, String^ message)

### Properties

- uint32\_t [Status](#) [get]

#### 11.113.1 Detailed Description

Exception class that is thrown in case of an USB error.

#### 11.113.2 Constructor & Destructor Documentation

##### 11.113.2.1 CUsbExceptionNet() [1/2] [CUsbExceptionNet](#) ( uint32\_t status )

Constructor of a CUsbException.

##### Parameters

<i>status</i>	the status number
---------------	-------------------

##### 11.113.2.2 CUsbExceptionNet() [2/2] [CUsbExceptionNet](#) ( uint32\_t status, String^ message )

### 11.113.3 Property Documentation

**11.113.3.1 Status** `uint32_t Status [get]`

## 11.114 CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet Class Reference

### Public Member Functions

- [CVoltageRangeInfoNet](#) (int vr, String^ vrString)

### Public Attributes

- int [VoltageRangeInMicroVolt](#)
- String ^ [VoltageRangeDisplayStringMilliVolt](#)

### 11.114.1 Constructor & Destructor Documentation

**11.114.1.1 CVoltageRangeInfoNet()** [CVoltageRangeInfoNet](#) (   
int vr,   
String^ vrString )

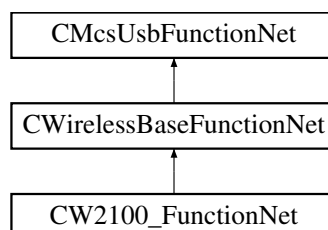
### 11.114.2 Member Data Documentation

**11.114.2.1 VoltageRangeDisplayStringMilliVolt** String ^ VoltageRangeDisplayStringMilliVolt

**11.114.2.2 VoltageRangeInMicroVolt** int VoltageRangeInMicroVolt

## 11.115 CW2100\_FunctionNet Class Reference

Inheritance diagram for CW2100\_FunctionNet:



**Classes**

- struct [AudioChannelsNet](#)

**Public Member Functions**

- [CW2100\\_FunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> w2100\_FunctionPointerContainer)
- [CW2100\\_FunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- [array](#)< [HeadStageIDType](#)<sup>^</sup>> <sup>^</sup> [GetAvailableHeadstages](#) (unsigned int max\_length)
- void [SelectHeadstage](#) (unsigned int IDorEntry, int TimeSlotNr)
- void [DeselectHeadstage](#) (int TimeSlotNr)
- void [DeselectAllHeadstages](#) ()
- [HeadStageIDTypeState](#) <sup>^</sup> [GetSelectedHeadstageState](#) (int TimeSlotNr)
- [BatteryState](#) <sup>^</sup> [GetBatteryState](#) (int TimeSlotNr)
- [System::String](#) <sup>^</sup> [GetUserDefinedName](#) (unsigned short ID)
- [System::String](#) <sup>^</sup> [GetUserDefinedNameFromSelectedHS](#) (int TimeSlotNr)
- [System::String](#) <sup>^</sup> [GetUserDefinedNameCache](#) (unsigned short ID)
- [uint32\\_t](#) [GetUserDefinedNameCache](#) (unsigned short ID, [[System::Runtime::InteropServices::Out](#)][System::String](#)<sup>^</sup>% Name)
- [W2100\\_StimulusParametersNet](#) <sup>^</sup> [GetStimulusParameters](#) (unsigned short ID)
- [W2100\\_StimulusParametersNet](#) <sup>^</sup> [GetStimulusParametersFromSelectedHS](#) (int TimeSlotNr)
- [W2100\\_StimulusParametersNet](#) <sup>^</sup> [GetStimulusParametersCache](#) (unsigned int typeValue)
- [uint32\\_t](#) [GetStimulusParametersCache](#) (unsigned int typeValue, [[System::Runtime::InteropServices::Out](#)][W2100\\_StimulusParametersNet](#)<sup>^</sup>% StimulusParameters)
- void [SetSelectedChannels](#) ([array](#)< [BYTE](#) ><sup>^</sup> channels, int TimeSlotNr)
- [array](#)< [BYTE](#) > <sup>^</sup> [GetSelectedChannels](#) (int TimeSlotNr)
- void [SetMultiHeadstageMode](#) (bool Mode)
- bool [GetMultiHeadstageMode](#) ()
- void [SetHeadstageSamplingActive](#) (bool Active, int TimeSlotNr)
- bool [GetHeadstageSamplingActive](#) (int TimeSlotNr)
- void [SetHeadstageToSleep](#) (unsigned int Sleep16ms, int TimeSlotNr)
- void [SetHeadstageOnOff](#) (unsigned short On, int TimeSlotNr)
- unsigned short [GetHeadstageOnOff](#) (int TimeSlotNr)
- unsigned int [GetAnalogOutChannel](#) ([[System::Runtime::InteropServices::Out](#)]int % automatic, unsigned short index)
- void [SetAnalogOutChannel](#) (int automatic, unsigned short index, unsigned int Channel)
- [array](#)< unsigned int > <sup>^</sup> [GetAnalogOutFilter](#) ([[System::Runtime::InteropServices::Out](#)]int % automatic)
- void [SetAnalogOutFilter](#) (int automatic, [array](#)< unsigned int ><sup>^</sup> Coeffs)
- [AnalogOut\\_DAC\\_Range\\_EnumNet](#) [GetDacRange](#) ()
- void [SetDacRange](#) ([AnalogOut\\_DAC\\_Range\\_EnumNet](#) range)
- [CFilterPropertyNet](#) <sup>^</sup> [GetFilterProperty](#) ([W2100DacqGroupChannelEnumNet](#) GroupID, unsigned int index)
- [array](#)< [CFilterPropertyNet](#)<sup>^</sup>> <sup>^</sup> [GetFilterProperties](#) ([W2100DacqGroupChannelEnumNet](#) GroupID)
- void [SetAccelGyroEnabled](#) ([W2100\\_Accel\\_Gyro\\_Select\\_EnumNet](#) enable, int TimeSlotNr)
- [W2100\\_Accel\\_Gyro\\_Select\\_EnumNet](#) [GetAccelGyroEnabled](#) (int TimeSlotNr)
- void [SetAccelGyroDesiredRate](#) (int rate, int TimeSlotNr)
- int [GetAccelGyroDesiredRate](#) (int TimeSlotNr)
- int [GetAccelGyroCurrentRate](#) (int TimeSlotNr)
- void [SetAccelRange](#) (int range, int TimeSlotNr)
- int [GetAccelRange](#) (int TimeSlotNr)
- void [SetGyroRange](#) (int range, int TimeSlotNr)
- int [GetGyroRange](#) (int TimeSlotNr)
- void [SetAudioChannels](#) ([array](#)< [AudioChannelsNet](#)<sup>^</sup>><sup>^</sup> channels)
- [array](#)< [AudioChannelsNet](#)<sup>^</sup>> <sup>^</sup> [GetAudioChannels](#) ()
- unsigned int [GetPicFirmwareType](#) (int TimeSlotNr)
- unsigned int [GetFPGA FirmwareType](#) (int TimeSlotNr)

**Static Public Member Functions**

- static void [ClearUserDefinedNameCache](#) ()
- static void [ClearUserDefinedNameCache](#) (unsigned short ID)
- static void [ClearStimulusParametersCache](#) ()
- static void [ClearStimulusParametersCache](#) (unsigned short ID)

**Properties**

- [CW2100\\_StimulatorFunctionNet](#)<sup>^</sup> [Stimulator](#) [get]
- [CPulseGeneratorFunctionNet](#)<sup>^</sup> [PulseGenerator](#) [get]

**Additional Inherited Members****11.115.1 Constructor & Destructor Documentation**

**11.115.1.1 CW2100\_FunctionNet()** [1/2] [CW2100\\_FunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> *mcsusb*,  
[CMcsUsbFunctionPointerContainer](#)<sup>^</sup> *w2100\_FunctionPointerContainer* )

**11.115.1.2 CW2100\_FunctionNet()** [2/2] [CW2100\\_FunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> *mcsusb* )

**11.115.2 Member Function Documentation**

**11.115.2.1 ClearStimulusParametersCache()** [1/2] static void [ClearStimulusParametersCache](#) ( )  
[static]

**11.115.2.2 ClearStimulusParametersCache()** [2/2] static void [ClearStimulusParametersCache](#) (  
unsigned short *ID* ) [static]

**11.115.2.3 ClearUserDefinedNameCache()** [1/2] static void [ClearUserDefinedNameCache](#) ( ) [static]

**11.115.2.4 ClearUserDefinedNameCache()** [2/2] static void ClearUserDefinedNameCache ( unsigned short *ID* ) [static]

**11.115.2.5 DeselectAllHeadstages()** void DeselectAllHeadstages ( )

**11.115.2.6 DeselectHeadstage()** void DeselectHeadstage ( int *TimeSlotNr* )

**11.115.2.7 GetAccelGyroCurrentRate()** int GetAccelGyroCurrentRate ( int *TimeSlotNr* )

**11.115.2.8 GetAccelGyroDesiredRate()** int GetAccelGyroDesiredRate ( int *TimeSlotNr* )

**11.115.2.9 GetAccelGyroEnabled()** W2100\_Accel\_Gyro\_Select\_EnumNet GetAccelGyroEnabled ( int *TimeSlotNr* )

**11.115.2.10 GetAccelRange()** int GetAccelRange ( int *TimeSlotNr* )

**11.115.2.11 GetAnalogOutChannel()** unsigned int GetAnalogOutChannel ( [System::Runtime::InteropServices::Out] int % *automatic*, unsigned short *index* )

**11.115.2.12 GetAnalogOutFilter()** array<unsigned int> ^ GetAnalogOutFilter ( [System::Runtime::InteropServices::Out] int % *automatic* )

**11.115.2.13 GetAudioChannels()** array<AudioChannelsNet^> ^ GetAudioChannels ( )

**11.115.2.14 GetAvailableHeadstages()** array<HeadStageIDType^> ^ GetAvailableHeadstages (   
 unsigned int *max\_length* )

**11.115.2.15 GetBatteryState()** BatteryState ^ GetBatteryState (   
 int *TimeSlotNr* )

**11.115.2.16 GetDacRange()** AnalogOut\_DAC\_Range\_EnumNet GetDacRange ( )

**11.115.2.17 GetFilterProperties()** array<CFilterPropertyNet^> ^ GetFilterProperties (   
 W2100DacqGroupChannelEnumNet *GroupID* )

**11.115.2.18 GetFilterProperty()** CFilterPropertyNet ^ GetFilterProperty (   
 W2100DacqGroupChannelEnumNet *GroupID*,   
 unsigned int *index* )

**11.115.2.19 GetFPGA FirmwareType()** unsigned int GetFPGA FirmwareType (   
 int *TimeSlotNr* )

**11.115.2.20 GetGyroRange()** int GetGyroRange (   
 int *TimeSlotNr* )

**11.115.2.21 GetHeadstageOnOff()** unsigned short GetHeadstageOnOff (   
 int *TimeSlotNr* )

**11.115.2.22 GetHeadstageSamplingActive()** bool GetHeadstageSamplingActive (   
 int *TimeSlotNr* )

**11.115.2.23 GetMultiHeadstageMode()** bool GetMultiHeadstageMode ( )

**11.115.2.24 GetPicFirmwareType()** unsigned int GetPicFirmwareType (   
int *TimeSlotNr* )

**11.115.2.25 GetSelectedChannels()** array<BYTE> ^ GetSelectedChannels (   
int *TimeSlotNr* )

**11.115.2.26 GetSelectedHeadstageState()** HeadStageIDTypeState ^ GetSelectedHeadstageState (   
int *TimeSlotNr* )

**11.115.2.27 GetStimulusParametersCache()** [1/2] W2100\_StimulusParametersNet ^ GetStimulus↔   
ParametersCache (   
unsigned int *typeValue* )

**11.115.2.28 GetStimulusParametersCache()** [2/2] uint32\_t GetStimulusParametersCache (   
unsigned int *typeValue*,   
[System::Runtime::InteropServices::Out] W2100\_StimulusParametersNet ^% *Stimulus↔   
Parameters* )

**11.115.2.29 GetStimulusParametersFromSelectedHS()** W2100\_StimulusParametersNet ^ GetStimulus↔   
ParametersFromSelectedHS (   
int *TimeSlotNr* )

**11.115.2.30 GetStiumlusParameters()** W2100\_StimulusParametersNet ^ GetStiumlusParameters (   
unsigned short *ID* )

**11.115.2.31 GetUserDefinedName()** System::String ^ GetUserDefinedName (   
unsigned short *ID* )

**11.115.2.32 GetUserDefinedNameCache()** [1/2] System::String ^ GetUserDefinedNameCache (   
unsigned short *ID* )

**11.115.2.33 GetUserDefinedNameCache()** [2/2] uint32\_t GetUserDefinedNameCache ( unsigned short *ID*, [System::Runtime::InteropServices::Out] System::String^% *Name* )

**11.115.2.34 GetUserDefinedNameFromSelectedHS()** System::String ^ GetUserDefinedNameFromSelectedHS ( int *TimeSlotNr* )

**11.115.2.35 SelectHeadstage()** void SelectHeadstage ( unsigned int *IDorEntry*, int *TimeSlotNr* )

**11.115.2.36 SetAccelGyroDesiredRate()** void SetAccelGyroDesiredRate ( int *rate*, int *TimeSlotNr* )

**11.115.2.37 SetAccelGyroEnabled()** void SetAccelGyroEnabled ( W2100\_Accel\_Gyro\_Select\_EnumNet *enable*, int *TimeSlotNr* )

**11.115.2.38 SetAccelRange()** void SetAccelRange ( int *range*, int *TimeSlotNr* )

**11.115.2.39 SetAnalogOutChannel()** void SetAnalogOutChannel ( int *automatic*, unsigned short *index*, unsigned int *Channel* )

**11.115.2.40 SetAnalogOutFilter()** void SetAnalogOutFilter ( int *automatic*, array< unsigned int >^ *Coeffs* )



**11.115.2.41 SetAudioChannels()** void SetAudioChannels (   
array< AudioChannelsNet^>^ channels )

**11.115.2.42 SetDacRange()** void SetDacRange (   
AnalogOut\_DAC\_Range\_EnumNet range )

**11.115.2.43 SetGyroRange()** void SetGyroRange (   
int range,   
int TimeSlotNr )

**11.115.2.44 SetHeadstageOnOff()** void SetHeadstageOnOff (   
unsigned short On,   
int TimeSlotNr )

**11.115.2.45 SetHeadstageSamplingActive()** void SetHeadstageSamplingActive (   
bool Active,   
int TimeSlotNr )

**11.115.2.46 SetHeadstageToSleep()** void SetHeadstageToSleep (   
unsigned int Sleep16ms,   
int TimeSlotNr )

**11.115.2.47 SetMultiHeadstageMode()** void SetMultiHeadstageMode (   
bool Mode )

**11.115.2.48 SetSelectedChannels()** void SetSelectedChannels (   
array< BYTE >^ channels,   
int TimeSlotNr )

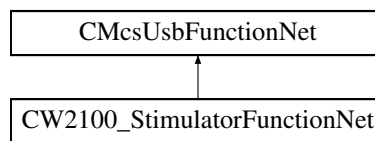
### 11.115.3 Property Documentation

**11.115.3.1 PulseGenerator** [CPulseGeneratorFunctionNet](#)<sup>^</sup> PulseGenerator [get]

**11.115.3.2 Stimulator** [CW2100\\_StimulatorFunctionNet](#)<sup>^</sup> Stimulator [get]

## 11.116 CW2100\_StimulatorFunctionNet Class Reference

Inheritance diagram for CW2100\_StimulatorFunctionNet:



### Public Member Functions

- [CW2100\\_StimulatorFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- void [SendStart](#) (uint32\_t triggermap)  
*Start (Trigger) the STG. The startup delay is in the range of a few ms.*
- void [SendStop](#) (uint32\_t triggermap)  
*Stop some or all triggers of the STG.*
- [CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData](#)<sup>^</sup> [PrepareData](#) (int channel, array< int32\_t ><sup>^</sup> amplitude, array< uint64\_t ><sup>^</sup> duration, STG\_DestinationEnumNet destType, uint32\_t repeat)
- [CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData](#)<sup>^</sup> [PrepareDataSync](#) (int channel, array< int32\_t ><sup>^</sup> amplitude, array< uint32\_t ><sup>^</sup> Sync, array< uint64\_t ><sup>^</sup> duration, STG\_DestinationEnumNet destType, uint32\_t repeat)
- void [SendPreparedData](#) (int channel, [CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData](#)<sup>^</sup> device\_data\_and\_unrolled, STG\_DestinationEnumNet destType)
- void [ClearChannelData](#) (int channel)  
*Delete a Stimulus Pattern from STG memory*
- int [GetDACResolution](#) ()  
*Gets number of bits of the DAC resolution.*
- int [GetTimeResolutionInNanoSeconds](#) ()  
*Gets number of bits of the DAC resolution.*
- int [GetVoltageRangeInMicroVolt](#) (uint32\_t channel)  
*Gets the Voltage Range of the specified channel in Microvolts.*
- int [GetVoltageResolutionInMicroVolt](#) (uint32\_t channel)  
*Gets the Voltage Resolution of the specified channel in Microvolts.*
- int [GetCurrentRangeInNanoAmp](#) (uint32\_t channel)  
*Gets the Current Range of the specified channel in Nanoamps.*
- int [GetCurrentResolutionInNanoAmp](#) (uint32\_t channel)  
*Gets the Current Resolution of the specified channel in Nanoamps.*
- uint32\_t [GetNumberOfAnalogChannels](#) ()
- uint32\_t [GetNumberOfSyncoutChannels](#) ()
- uint32\_t [GetNumberOfTriggerInputs](#) ()
- void [SelectTimeSlot](#) (int TimeSlotNr)
- int [GetTimeSlot](#) ()
- uint32\_t [GetStimulationPatternMemory](#) ()

- uint32\_t [GetBoostPreTime](#) ()
- void [SetBoostAlwaysOnMode](#) (uint32\_t alwayson\_mode)
- uint32\_t [GetBoostAlwaysOnMode](#) ()
- void [SetDigitalStimulatorTrigger](#) (int TimeSlotNr, DigitalStimulatorTriggerEventEnumNet trigger\_event, int trigger\_number, W2100DigitalSourceEnumNet digstream\_source, int bitnumber\_offset)
- void [GetDigitalStimulatorTrigger](#) (int TimeSlotNr, DigitalStimulatorTriggerEventEnumNet trigger\_event, int trigger\_number, [System::Runtime::InteropServices::Out]W2100DigitalSourceEnumNet% digstream\_source, [System::Runtime::InteropServices::Out]int% bitnumber\_offset)
- void [SetDigitalStimulatorTriggerSlope](#) (int TimeSlotNr, DigitalStimulatorTriggerEventEnumNet trigger\_event, int trigger\_number, DigitalStimulatorTriggerSlopeEnumNet slope)
- DigitalStimulatorTriggerSlopeEnumNet [GetDigitalStimulatorTriggerSlope](#) (int TimeSlotNr, DigitalStimulatorTriggerEventEnumNet trigger\_event, int trigger\_number)
- void [StartPoll](#) ()
- void [StopPoll](#) ()

### Static Public Attributes

- static const uint32\_t [BOOST\\_BIT](#) = (1 << 0)
- static const uint32\_t [GND\\_SWITCH\\_BIT](#) = (1 << 1)
- static const uint32\_t [SYNC\\_BIT0](#) = (1 << 2)
- static const uint32\_t [SYNC\\_BIT1](#) = (1 << 3)

### Events

- [OnStgPollStatus](#)<sup>^</sup> [PollStatusEvent](#)

### Additional Inherited Members

#### 11.116.1 Constructor & Destructor Documentation

**11.116.1.1 CW2100\_StimulatorFunctionNet()** [CW2100\\_StimulatorFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> *mcsusb* )

#### 11.116.2 Member Function Documentation

**11.116.2.1 ClearChannelData()** void [ClearChannelData](#) (  
int *channel* )

Delete a Stimulus Pattern from STG memory

#### Parameters

<i>channel</i>	specifies the channel to clear.
----------------	---------------------------------

**11.116.2.2 GetBoostAlwaysOnMode()** `uint32_t GetBoostAlwaysOnMode ( )`

**11.116.2.3 GetBoostPreTime()** `uint32_t GetBoostPreTime ( )`

**11.116.2.4 GetCurrentRangeInNanoAmp()** `int GetCurrentRangeInNanoAmp (   
uint32_t channel )`

Gets the Current Range of the specified channel in Nanoamps.

**Parameters**

<i>channel</i>	Channel which is queried.
----------------	---------------------------

**Returns**

The Current Range of the specified channel in Nanoamps.

**11.116.2.5 GetCurrentResolutionInNanoAmp()** `int GetCurrentResolutionInNanoAmp (   
uint32_t channel )`

Gets the Current Resolution of the specified channel in Nanoamps.

**Parameters**

<i>channel</i>	Channel which is queried.
----------------	---------------------------

**Returns**

The Current Resolution of the specified channel in Nanoamps.

**11.116.2.6 GetDACResolution()** `int GetDACResolution ( )`

Gets number of bits of the DAC resolution.

**Returns**

The DAC resolution in bits.

**11.116.2.7 GetDigitalStimulatorTrigger()** void GetDigitalStimulatorTrigger (   
     int *TimeSlotNr*,   
     DigitalStimulatorTriggerEventEnumNet *trigger\_event*,   
     int *trigger\_number*,   
     [System::Runtime::InteropServices::Out] W2100DigitalSourceEnumNet% *digstream\_↵*  
*source*,   
     [System::Runtime::InteropServices::Out] int% *bitnumber\_offset* )

**11.116.2.8 GetDigitalStimulatorTriggerSlope()** DigitalStimulatorTriggerSlopeEnumNet GetDigital↵  
 StimulatorTriggerSlope (   
     int *TimeSlotNr*,   
     DigitalStimulatorTriggerEventEnumNet *trigger\_event*,   
     int *trigger\_number* )

**11.116.2.9 GetNumberOfAnalogChannels()** uint32\_t GetNumberOfAnalogChannels ( )

**11.116.2.10 GetNumberOfSyncoutChannels()** uint32\_t GetNumberOfSyncoutChannels ( )

**11.116.2.11 GetNumberOfTriggerInputs()** uint32\_t GetNumberOfTriggerInputs ( )

**11.116.2.12 GetStimulationPatternMemory()** uint32\_t GetStimulationPatternMemory ( )

**11.116.2.13 GetTimeResolutionInNanoSeconds()** int GetTimeResolutionInNanoSeconds ( )

Gets number of bits of the DAC resolution.

#### Returns

The time resolution in ns.

**11.116.2.14 GetTimeSlot()** int GetTimeSlot ( )

**11.116.2.15 GetVoltageRangeInMicroVolt()** int GetVoltageRangeInMicroVolt (   
     uint32\_t *channel* )

Gets the Voltage Range of the specified channel in Microvolts.

## Parameters

<i>channel</i>	Channel which is queried.
----------------	---------------------------

## Returns

The Voltage Range of the specified channel in Microvolts.

**11.116.2.16 GetVoltageResolutionInMicroVolt()** `int GetVoltageResolutionInMicroVolt ( uint32_t channel )`

Gets the Voltage Resolution of the specified channel in Microvolts.

## Parameters

<i>channel</i>	Channel which is queried.
----------------	---------------------------

## Returns

The Voltage Resolution of the specified channel in Microvolts.

**11.116.2.17 PrepareData()** `CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData ^ PrepareData (`

```
int channel,
array< int32_t >^ amplitude,
array< uint64_t >^ duration,
STG_DestinationEnumNet destType,
uint32_t repeat )
```

**11.116.2.18 PrepareDataSync()** `CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData ^ PrepareDataSync (`

```
int channel,
array< int32_t >^ amplitude,
array< uint32_t >^ Sync,
array< uint64_t >^ duration,
STG_DestinationEnumNet destType,
uint32_t repeat )
```

**11.116.2.19 SelectTimeSlot()** `void SelectTimeSlot ( int TimeSlotNr )`

**11.116.2.20 SendPreparedData()** void SendPreparedData (   
     int *channel*,   
     CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData^ *device\_data\_and\_unrolled*,   
     STG\_DestinationEnumNet *destType* )

**11.116.2.21 SendStart()** void SendStart (   
     uint32\_t *triggermap* )

Start (Trigger) the STG. The startup delay is in the range of a few ms.

#### Parameters

<i>triggermap</i>	A bitmap of triggers which will be started.
-------------------	---

**11.116.2.22 SendStop()** void SendStop (   
     uint32\_t *triggermap* )

Stop some or all triggers of the STG.

#### Parameters

<i>triggermap</i>	A bitmap of triggers which will be stopped.
-------------------	---

**11.116.2.23 SetBoostAlwaysOnMode()** void SetBoostAlwaysOnMode (   
     uint32\_t *alwayson\_mode* )

**11.116.2.24 SetDigitalStimulatorTrigger()** void SetDigitalStimulatorTrigger (   
     int *TimeSlotNr*,   
     DigitalStimulatorTriggerEventEnumNet *trigger\_event*,   
     int *trigger\_number*,   
     W2100DigitalSourceEnumNet *digstream\_source*,   
     int *bitnumber\_offset* )

**11.116.2.25 SetDigitalStimulatorTriggerSlope()** void SetDigitalStimulatorTriggerSlope (   
     int *TimeSlotNr*,   
     DigitalStimulatorTriggerEventEnumNet *trigger\_event*,   
     int *trigger\_number*,   
     DigitalStimulatorTriggerSlopeEnumNet *slope* )

**11.116.2.26 StartPoll()** `void StartPoll ( )`

**11.116.2.27 StopPoll()** `void StopPoll ( )`

### 11.116.3 Member Data Documentation

**11.116.3.1 BOOST\_BIT** `const uint32_t BOOST_BIT = (1 << 0) [static]`

**11.116.3.2 GND\_SWITCH\_BIT** `const uint32_t GND_SWITCH_BIT = (1 << 1) [static]`

**11.116.3.3 SYNC\_BIT0** `const uint32_t SYNC_BIT0 = (1 << 2) [static]`

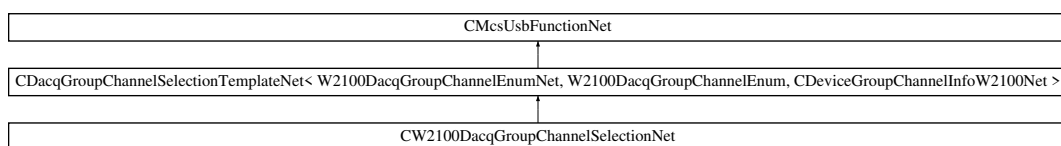
**11.116.3.4 SYNC\_BIT1** `const uint32_t SYNC_BIT1 = (1 << 3) [static]`

### 11.116.4 Event Documentation

**11.116.4.1 PollStatusEvent** `OnStgPollStatus^ PollStatusEvent`

## 11.117 CW2100DacqGroupChannelSelectionNet Class Reference

Inheritance diagram for CW2100DacqGroupChannelSelectionNet:



### Public Member Functions

- `CW2100DacqGroupChannelSelectionNet (CMcsUsbNet^ mcsusb)`



## Additional Inherited Members

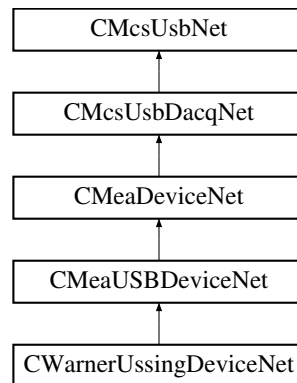
### 11.117.1 Constructor & Destructor Documentation

**11.117.1.1 CW2100DacqGroupChannelSelectionNet()** [CW2100DacqGroupChannelSelectionNet](#) (  
[CMcsUsbNet](#)^ *mcsusb* )

## 11.118 CWarnerUssingDeviceNet Class Reference

[CWarnerUssingDeviceNet](#) is the class to control the Ussing device

Inheritance diagram for CWarnerUssingDeviceNet:



## Public Member Functions

- [CWarnerUssingDeviceNet](#) ()  
*Initializes a new instance of the [CWarnerUssingDeviceNet](#) class.*
- virtual [~CWarnerUssingDeviceNet](#) ()
- [ICWarnerUssingDeviceNet](#) ()

## Properties

- [CWarnerUssingFunctionNet](#)^ [WarnerUssingFunction](#) [get]

## Additional Inherited Members

### 11.118.1 Detailed Description

[CWarnerUssingDeviceNet](#) is the class to control the Ussing device

## 11.118.2 Constructor & Destructor Documentation

### 11.118.2.1 CWarnerUssingDeviceNet() CWarnerUssingDeviceNet ( )

Initializes a new instance of the CWarnerUssingDeviceNet class.

### 11.118.2.2 ~CWarnerUssingDeviceNet() virtual ~CWarnerUssingDeviceNet ( ) [virtual]

### 11.118.2.3 "!CWarnerUssingDeviceNet() !CWarnerUssingDeviceNet ( )

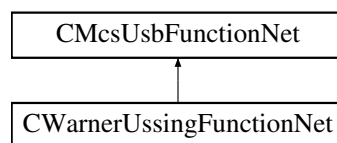
## 11.118.3 Property Documentation

### 11.118.3.1 WarnerUssingFunction CWarnerUssingFunctionNet^ WarnerUssingFunction [get]

## 11.119 CWarnerUssingFunctionNet Class Reference

CWarnerUssingFunctionNet is the class to control the Ussing device

Inheritance diagram for CWarnerUssingFunctionNet:



## Public Member Functions

- [CWarnerUssingFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> pWarner←  
UssingFunctionPointerContainer)  
*Initializes a new instance of the [CWarnerUssingFunctionNet](#) class.*
- [CWarnerUssingFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- virtual [~CWarnerUssingFunctionNet](#) ()
- [ICWarnerUssingFunctionNet](#) ()
- [int32\\_t](#) [GetChannelsCountOfChamber](#) ([int32\\_t](#) ChamberId)  
*gets number of channels in datastream from chamber amp with given index*
- [int32\\_t](#) [GetNumberOfHardwareSlotsForChambers](#) ()  
*gets number of physical hardware slots for chambers amps*
- [int32\\_t](#) [GetNumberOfAvailableChambers](#) ()  
*gets number of actually connected chamber amps*
- [bool](#) [IsChamberAvailable](#) ([int32\\_t](#) ChamberId)  
*checks whether chamber amp is connected to slot*
- void [SetPulse](#) ([int32\\_t](#) ChamberId, [UssingClampModeEnumNet](#) StgMode, [int32\\_t](#) NumberOfRepetitions,  
array< [int](#) ><sup>^</sup> Amplitudes, array< [int](#) ><sup>^</sup> Durations)  
*defines stimulation pulse pattern for voltage or current stimulation; CAUTION: zero-length amplitude will be briefly applied -> choose matching to neighbour to avoid spikes*
- void [SetVoltageClampControllerParam\\_P](#) ([int32\\_t](#) ChamberId, [uint32\\_t](#) P)  
*sets P value of PID controller;*
- void [SetVoltageClampControllerParam\\_I](#) ([int32\\_t](#) ChamberId, [uint32\\_t](#) I)  
*sets I value of PID controller;*
- void [SetVoltageClampControllerParam\\_D](#) ([int32\\_t](#) ChamberId, [uint32\\_t](#) D)  
*sets D value of PID controller;*
- [uint32\\_t](#) [GetVoltageClampControllerParam\\_P](#) ([int32\\_t](#) ChamberId)  
*gets P value of PID controller;*
- [uint32\\_t](#) [GetVoltageClampControllerParam\\_I](#) ([int32\\_t](#) ChamberId)  
*gets I value of PID controller;*
- [uint32\\_t](#) [GetVoltageClampControllerParam\\_D](#) ([int32\\_t](#) ChamberId)  
*gets D value of PID controller;*
- void [SetClampMode](#) ([int32\\_t](#) ChamberId, [UssingClampModeEnumNet](#) NewClampMode)  
*sets clamp mode (voltage, current or open clamp)*
- [UssingClampModeEnumNet](#) [GetClampMode](#) ([int32\\_t](#) ChamberId)  
*gets clamp mode (voltage, current or open clamp; do not use when device is in internal calibration mode)*
- [bool](#) [IsInternalCalibrationFinished](#) ([int32\\_t](#) ChamberId)  
*when internal calibration is finished, values for U1,2\_offset and U1,2\_reference and DAC\_offset are available*
- [int32\\_t](#) [GetU1Offset](#) ([int32\\_t](#) ChamberId)
- [int32\\_t](#) [GetU2Offset](#) ([int32\\_t](#) ChamberId)
- [int32\\_t](#) [GetU1Reference](#) ([int32\\_t](#) ChamberId)
- [int32\\_t](#) [GetU2Reference](#) ([int32\\_t](#) ChamberId)
- [int32\\_t](#) [GetDacZero](#) ([int32\\_t](#) ChamberId)
- void [SetHighCurrentMode](#) ([int32\\_t](#) ChamberId)
- void [SetLowCurrentMode](#) ([int32\\_t](#) ChamberId)
- [bool](#) [IsHighCurrentMode](#) ([int32\\_t](#) ChamberId)
- [uint32\\_t](#) [GetLowCurrentRange](#) ([int32\\_t](#) ChamberId)  
*unit: nA*
- [uint32\\_t](#) [GetHighCurrentRange](#) ([int32\\_t](#) ChamberId)  
*unit: nA*
- [uint32\\_t](#) [GetDacPampsPerDigitLowCurrentRange](#) ([int32\\_t](#) ChamberId)  
*unit: pA/digit*

- uint32\_t [GetDacNampsPerDigitHighCurrentRange](#) (int32\_t ChamberId)  
*unit: nA/digit*
- uint32\_t [GetUnitsPerDigit](#) (int32\_t ChamberId, int32\_t ChannelId)  
*gets amps/volts per digit for specified chamber and channel*
- int32\_t [GetUnitExponent](#) (int32\_t ChamberId, int32\_t ChannelId)
- UssingUnitEnumNet [GetUnitName](#) (int32\_t ChamberId, int32\_t ChannelId)  
*gets the channel's unit name*
- String ^ [GetUnitDescription](#) (int32\_t ChamberId, int32\_t ChannelId)
- array< int > ^ [GetAvailableChambers](#) ()  
*returns array with (zero-based) ChamberIds of all available chambers*
- int32\_t [GetUptimeSeconds](#) (int32\_t ChamberId)
- void [SetIdleModeOffset](#) (int32\_t ChamberId, UssingClampModeEnumNet ClampMode, int32\_t NewIdleOffset)  
*sets the offset (voltage or current) that will be applied when clamping is DISABLED*
- int32\_t [GetIdleModeOffset](#) (int32\_t ChamberId, UssingClampModeEnumNet ClampMode)  
*gets the offset (voltage or current) that will be applied when clamping is DISABLED*
- void [SetEnablePulse](#) (int32\_t ChamberId, UssingClampModeEnumNet ClampMode, bool Enable)  
*enable pulse of given chamber and mode (voltage/current clamp) of this chamber*
- bool [IsPulseEnabled](#) (int32\_t ChamberId, UssingClampModeEnumNet ClampMode)  
*returns true when pulse of given chamber and current mode (voltage/current clamp) of this chamber is ENABLED*
- void [SetLiquidResistance](#) (int32\_t ChamberId, int32\_t NewLiquidResistance\_Ohm)
- int32\_t [GetLiquidResistance](#) (int32\_t ChamberId)
- int32\_t [GetComplianceVoltage](#) (int32\_t ChamberId)  
*returns compliance voltage threshold in digits; when Uc is above, current source is overloaded*
- bool [CompensateElectrodeOffset](#) (int32\_t ChamberId)  
*blocking call to compensate electrode offset of one chamber; returns true when successful*
- bool [WaitForChamber](#) (int32\_t ChamberId)  
*blocking call that waits for chamber boot-up calibration to complete*
- bool [WaitForAllChambers](#) ()  
*blocking call that waits for ALL chambers' boot-up calibration to complete*

## Additional Inherited Members

### 11.119.1 Detailed Description

[CWarnerUssingFunctionNet](#) is the class to control the Ussing device

### 11.119.2 Constructor & Destructor Documentation

**11.119.2.1 CWarnerUssingFunctionNet()** [1/2] [CWarnerUssingFunctionNet](#) (   
[CMcsUsbNet](#)^ mcsusb,   
[CMcsUsbFunctionPointerContainer](#)^ pWarnerUssingFunctionPointerContainer )

Initializes a new instance of the [CWarnerUssingFunctionNet](#) class.

**11.119.2.2 CWarnerUssingFunctionNet()** [2/2] `CWarnerUssingFunctionNet ( CMcsUsbNet^ mcsusb )`

**11.119.2.3 ~CWarnerUssingFunctionNet()** `virtual ~CWarnerUssingFunctionNet ( ) [virtual]`

**11.119.2.4 "!CWarnerUssingFunctionNet()** `!CWarnerUssingFunctionNet ( )`

### 11.119.3 Member Function Documentation

**11.119.3.1 CompensateElectrodeOffset()** `bool CompensateElectrodeOffset ( int32_t ChamberId )`

blocking call to compensate electrode offset of one chamber; returns true when successful

#### Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based)
------------------	---

#### Returns

**11.119.3.2 GetAvailableChambers()** `array<int> ^ GetAvailableChambers ( )`

returns array with (zero-based) ChamberIds of all available chambers

**11.119.3.3 GetChannelsCountOfChamber()** `int32_t GetChannelsCountOfChamber ( int32_t ChamberId )`

gets number of channels in datastream from chamber amp with given index

#### Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based)
------------------	---

**Returns**

return value of zero means that amp is not placed

**11.119.3.4 GetClampMode()** `UssingClampModeEnumNet GetClampMode (`  
`int32_t ChamberId )`

gets clamp mode (voltage, current or open clamp; do not use when device is in internal calibration mode)

**Parameters**

<i>Chamber↔ Id</i>	index of hardware chamber slot (zero-based)
------------------------	---

**Returns**

**11.119.3.5 GetComplianceVoltage()** `int32_t GetComplianceVoltage (`  
`int32_t ChamberId )`

returns compliance voltage threshold in digits; when  $U_c$  is above, current source is overloaded

**Parameters**

<i>Chamber↔ Id</i>	index of hardware chamber slot (zero-based)
------------------------	---

**Returns**

**11.119.3.6 GetDacNampsPerDigitHighCurrentRange()** `uint32_t GetDacNampsPerDigitHighCurrentRange`  
`(`  
`int32_t ChamberId )`

unit: nA/digit

**Parameters**

<i>Chamber↔ Id</i>	index of hardware chamber slot (zero-based)
------------------------	---

## Returns

**11.119.3.7 GetDacPampsPerDigitLowCurrentRange()** `uint32_t GetDacPampsPerDigitLowCurrentRange ( int32_t ChamberId )`

unit: pA/digit

## Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based)
------------------	---

## Returns

**11.119.3.8 GetDacZero()** `int32_t GetDacZero ( int32_t ChamberId )`

- diagnostic function only - ; gets real zero value of DAC in digits (0 -> neg. current; 32767 -> near zero; 65535 -> pos. current)

## Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based)
------------------	---

## Returns

**11.119.3.9 GetHighCurrentRange()** `uint32_t GetHighCurrentRange ( int32_t ChamberId )`

unit: nA

## Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based)
------------------	---

## Returns

**11.119.3.10 GetIdleModeOffset()** `int32_t GetIdleModeOffset (`  
    `int32_t ChamberId,`  
    `UssingClampModeEnumNet ClampMode )`

gets the offset (voltage or current) that will be applied when clamping is DISABLED

## Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based)
<i>ClampMode</i>	voltage or current clamp stimulation

## Returns

unit: nA or mV

**11.119.3.11 GetLiquidResistance()** `int32_t GetLiquidResistance (`  
    `int32_t ChamberId )`

## Parameters

<i>Chamber↔ Id</i>	index of hardware chamber slot (zero-based)
------------------------	---

## Returns

**11.119.3.12 GetLowCurrentRange()** `uint32_t GetLowCurrentRange (`  
    `int32_t ChamberId )`

unit: nA

## Parameters

<i>Chamber↔ Id</i>	index of hardware chamber slot (zero-based)
------------------------	---



Returns

**11.119.3.13 GetNumberOfAvailableChambers()** `int32_t GetNumberOfAvailableChambers ( )`

gets number of actually connected chamber amps

Returns

**11.119.3.14 GetNumberOfHardwareSlotsForChambers()** `int32_t GetNumberOfHardwareSlotsForChambers ( )`

gets number of physical hardware slots for chambers amps

Returns

**11.119.3.15 GetU1Offset()** `int32_t GetU1Offset (   
int32_t ChamberId )`

- diagnostic function only -

Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based)
------------------	---

Returns

**11.119.3.16 GetU1Reference()** `int32_t GetU1Reference (   
int32_t ChamberId )`

- diagnostic function only -

## Parameters

<i>Chamber↔ Id</i>	index of hardware chamber slot (zero-based)
------------------------	---

## Returns

**11.119.3.17 GetU2Offset()** `int32_t GetU2Offset (`  
`int32_t ChamberId )`

- diagnostic function only -

## Parameters

<i>Chamber↔ Id</i>	index of hardware chamber slot (zero-based)
------------------------	---

## Returns

**11.119.3.18 GetU2Reference()** `int32_t GetU2Reference (`  
`int32_t ChamberId )`

- diagnostic function only -

## Parameters

<i>Chamber↔ Id</i>	index of hardware chamber slot (zero-based)
------------------------	---

## Returns

**11.119.3.19 GetUnitDescription()** `String ^ GetUnitDescription (`  
`int32_t ChamberId,`  
`int32_t ChannelId )`

## Parameters

<i>Chamber↔ Id</i>	index of hardware chamber slot (zero-based)
<i>ChannelId</i>	index of channel (zero-based)

## Returns

**11.119.3.20 GetUnitExponent()** `int32_t GetUnitExponent (`  
    `int32_t ChamberId,`  
    `int32_t ChannelId )`

## Parameters

<i>Chamber↔ Id</i>	index of hardware chamber slot (zero-based)
<i>ChannelId</i>	index of channel (zero-based)

## Returns

example: return value -9 means that amps/volts per digit is in nano

**11.119.3.21 GetUnitName()** `UssingUnitEnumNet GetUnitName (`  
    `int32_t ChamberId,`  
    `int32_t ChannelId )`

gets the channel's unit name

## Parameters

<i>Chamber↔ Id</i>	index of hardware chamber slot (zero-based)
<i>ChannelId</i>	index of channel (zero-based)

## Returns

**11.119.3.22 GetUnitsPerDigit()** `uint32_t GetUnitsPerDigit (`  
    `int32_t ChamberId,`  
    `int32_t ChannelId )`

gets amps/volts per digit for specified chamber and channel

## Parameters

<i>Chamber↔ Id</i>	index of hardware chamber slot (zero-based)
<i>ChannelId</i>	index of channel (zero-based)

## Returns

**11.119.3.23 GetUptimeSeconds()** `int32_t GetUptimeSeconds (`  
    `int32_t ChamberId )`

## Parameters

<i>Chamber↔ Id</i>	index of hardware chamber slot (zero-based)
------------------------	---

## Returns

**11.119.3.24 GetVoltageClampControllerParam\_D()** `uint32_t GetVoltageClampControllerParam_D (`  
    `int32_t ChamberId )`

gets D value of PID controller;

## Parameters

<i>Chamber↔ Id</i>	index of hardware chamber slot (zero-based)
------------------------	---

## Returns

**11.119.3.25 GetVoltageClampControllerParam\_I()** `uint32_t GetVoltageClampControllerParam_I (`  
    `int32_t ChamberId )`

gets I value of PID controller;

## Parameters

<i>Chamber↔ Id</i>	index of hardware chamber slot (zero-based)
------------------------	---

## Returns

**11.119.3.26 GetVoltageClampControllerParam\_P()** `uint32_t GetVoltageClampControllerParam_P (`  
`int32_t ChamberId )`

gets P value of PID controller;

## Parameters

<i>Chamber↔ Id</i>	index of hardware chamber slot (zero-based)
------------------------	---

## Returns

**11.119.3.27 IsChamberAvailable()** `bool IsChamberAvailable (`  
`int32_t ChamberId )`

checks whether chamber amp is connected to slot

## Parameters

<i>Chamber↔ Id</i>	index of hardware chamber slot (zero-based)
------------------------	---

## Returns

**11.119.3.28 IsHighCurrentMode()** `bool IsHighCurrentMode (`  
`int32_t ChamberId )`

## Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based)
------------------	---

## Returns

**11.119.3.29 IsInternalCalibrationFinished()** `bool IsInternalCalibrationFinished ( int32_t ChamberId )`

when internal calibration is finished, values for U1,2\_offset and U1,2\_reference and DAC\_offset are available

## Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based)
------------------	---

## Returns

**11.119.3.30 IsPulseEnabled()** `bool IsPulseEnabled ( int32_t ChamberId, UssingClampModeEnumNet ClampMode )`

returns true when pulse of given chamber and current mode (voltage/current clamp) of this chamber is ENABLED

## Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based)
<i>ClampMode</i>	voltage or current clamp stimulation

## Returns

when ENABLED, previously defined pulse pattern will be applied, otherwise the chamber current/voltage will be kept at specified offset level

**11.119.3.31 SetClampMode()** `void SetClampMode ( int32_t ChamberId, UssingClampModeEnumNet NewClampMode )`

sets clamp mode (voltage, current or open clamp)

## Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based)
<i>NewClampMode</i>	

**11.119.3.32 SetEnablePulse()** void SetEnablePulse (   
     int32\_t *ChamberId*,  
     UssingClampModeEnumNet *ClampMode*,  
     bool *Enable* )

enable pulse of given chamber and mode (voltage/current clamp) of this chamber

## Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based)
<i>ClampMode</i>	voltage or current clamp stimulation
<i>Enable</i>	when ENABLED, previously defined pulse pattern will be applied, otherwise the chamber current/voltage will be kept at specified offset level

**11.119.3.33 SetHighCurrentMode()** void SetHighCurrentMode (   
     int32\_t *ChamberId* )

## Parameters

<i>ChamberId</i>	
------------------	--

**11.119.3.34 SetIdleModeOffset()** void SetIdleModeOffset (   
     int32\_t *ChamberId*,  
     UssingClampModeEnumNet *ClampMode*,  
     int32\_t *NewIdleOffset* )

sets the offset (voltage or current) that will be applied when clamping is DISABLED

## Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based)
<i>ClampMode</i>	voltage or current clamp stimulation
<i>NewIdleOffset</i>	unit: nA or mV



**11.119.3.35 SetLiquidResistance()** void SetLiquidResistance (   
     int32\_t ChamberId,   
     int32\_t NewLiquidResistance\_Ohm )

#### Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based)
<i>NewLiquidResistance_Ohm</i>	

**11.119.3.36 SetLowCurrentMode()** void SetLowCurrentMode (   
     int32\_t ChamberId )

#### Parameters

<i>ChamberId</i>	
------------------	--

**11.119.3.37 SetPulse()** void SetPulse (   
     int32\_t ChamberId,   
     UssingClampModeEnumNet StgMode,   
     int32\_t NumberOfRepetitions,   
     array< int >^ Amplitudes,   
     array< int >^ Durations )

defines stimulation pulse pattern for voltage or current stimulation; CAUTION: zero-length amplitude will be briefly applied -> choose matching to neighbour to avoid spikes

#### Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based); send pattern to connected amp
<i>StgMode</i>	voltage or current clamp stimulation
<i>NumberOfRepetitions</i>	number of repetitions for pulse pattern (-1 for infinite; n means pattern is applied n+1 times)
<i>Amplitudes</i>	amplitude; unit in voltage clamp: mV; unit in current clamp: nA
<i>Durations</i>	duration in 100us; CAUTION: first element is applied only one; auto-loop back to second element after last one

**11.119.3.38 SetVoltageClampControllerParam\_D()** void SetVoltageClampControllerParam\_D (   
     int32\_t ChamberId,   
     uint32\_t D )

sets D value of PID controller;

## Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based)
<i>D</i>	useful range: 0..700

**11.119.3.39 SetVoltageClampControllerParam\_I()** `void SetVoltageClampControllerParam_I ( int32_t ChamberId, uint32_t I )`

sets I value of PID controller;

## Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based)
<i>I</i>	useful range: 80000..120000

**11.119.3.40 SetVoltageClampControllerParam\_P()** `void SetVoltageClampControllerParam_P ( int32_t ChamberId, uint32_t P )`

sets P value of PID controller;

## Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based)
<i>P</i>	useful value: 130000

**11.119.3.41 WaitForAllChambers()** `bool WaitForAllChambers ( )`

blocking call that waits for ALL chambers' boot-up calibration to complete

## Returns

returns false when at least one chamber's calibration fails (e.g. timeout...)

**11.119.3.42 WaitForChamber()** `bool WaitForChamber ( int32_t ChamberId )`

blocking call that waits for chamber boot-up calibration to complete

## Parameters

<i>Chamber↔ Id</i>	index of hardware chamber slot (zero-based)
------------------------	---

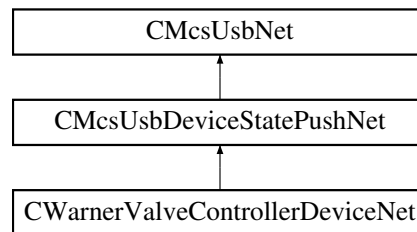
## Returns

returns false when calibration fails (e.g. timeout...)

## 11.120 CWarnerValveControllerDeviceNet Class Reference

[CWarnerValveControllerDeviceNet](#) is the class to access the Warner Valve Controller

Inheritance diagram for CWarnerValveControllerDeviceNet:



## Public Member Functions

- delegate void [OnGetValveActive](#) (uint16\_t valve, int valveActive)
- delegate void [OnGetValveManualState](#) (uint16\_t valve, int32\_t valveManualState)
- delegate void [OnGetValveManualGroup](#) (uint16\_t valve, int32\_t valveManualGroup)
- delegate void [OnGetValveMode](#) (uint16\_t valve, WvcValveModeEnumNet ValveMode)
- delegate void [OnGetAnalogThresholdLow](#) (uint16\_t valve, int32\_t threshold)
- delegate void [OnGetAnalogThresholdHigh](#) (uint16\_t valve, int32\_t threshold)
- delegate void [OnGetDigitalPortDirection](#) (uint16\_t port, PortDirectionEnumNet direction)
- delegate void [OnIsValveDigitalInInverted](#) (uint16\_t valve, bool isInverted)
- delegate void [OnGetValveDigitalInPort](#) (uint16\_t valve, uint32\_t digitalInPort)
- delegate void [OnIsDigitalOutPortInverted](#) (uint16\_t digitalOutPort, bool isInverted)
- delegate void [OnGetDigitalOutPortValve](#) (uint16\_t digitalOutPort, uint32\_t valve)
- delegate void [OnIsValveOpen](#) (uint16\_t valve, bool valveOpen)
- delegate void [OnIsValveOpenInDigitalMode](#) (uint16\_t valve, bool valveOpen)
- delegate void [OnIsValveOpenInAnalogMode](#) (uint16\_t valve, bool valveOpen)
- delegate void [OnGetAnalogVoltage](#) (int32\_t voltage)
- delegate void [OnTableEntryChanged](#) (uint16\_t tableNumber)
- delegate void [OnGetTableNamebyIndex](#) (uint16\_t tableNumber, String^ tableName)
- delegate void [OnGetActiveRunningTableNumber](#) (uint32\_t tableNumber)
- delegate void [OnGetCurrentNumberOfValves](#) (int32\_t numberOfValves)
- delegate void [OnGetValveBoardRevision](#) (uint32\_t revision)
- delegate void [OnGetDisplayMode](#) (WvcDisplayModeEnumNet DisplayMode)
- [CWarnerValveControllerDeviceNet](#) ()  
*Initializes a new instance of the [CWarnerValveControllerDeviceNet](#) class.*
- virtual [~CWarnerValveControllerDeviceNet](#) ()
- [!CWarnerValveControllerDeviceNet](#) ()
- int [GetValveActive](#) (uint16\_t valve)

- Gets the valve active/inactive state*
- void [SetValveActive](#) (uint16\_t valve, int valveActive)
- Sets the valve active/inactive state*
- uint32\_t [GetValvesActiveMap](#) ()
- Gets the valves active/inactive states*
- void [SetValvesActiveMap](#) (uint32\_t valvesActive)
- Sets the valve active/inactive state*
- int32\_t [GetValveManualState](#) (uint16\_t valve)
- Gets the valve manual on/off state*
- void [SetValveManualState](#) (uint16\_t valve, int32\_t valveManualState)
- Sets the valve manual on/off state*
- uint32\_t [GetValvesManualStateMap](#) ()
- Gets the valves manual on/off states*
- void [SetValvesManualStateMap](#) (uint32\_t valveaManualState)
- Sets the valve manual on/off state*
- int32\_t [GetValveManualGroup](#) (uint16\_t valve)
- Gets the valve manual group*
- void [SetValveManualGroup](#) (uint16\_t valve, int32\_t valveManualGroup)
- Sets the valve manual group*
- WvcValveModeEnumNet [GetValveMode](#) (uint16\_t valve)
- Reads the valve mode*
- void [SetValveMode](#) (uint16\_t valve, WvcValveModeEnumNet ValveMode)
- Writes the valve mode*
- int32\_t [GetAnalogThresholdLow](#) (uint16\_t valve)
- Gets the lower threshold for the analog in port per valve*
- void [SetAnalogThresholdLow](#) (uint16\_t valve, int32\_t threshold)
- Sets the lower threshold for the analog in port per valve*
- int32\_t [GetAnalogThresholdHigh](#) (uint16\_t valve)
- Gets the upper threshold for the analog in port per valve*
- void [SetAnalogThresholdHigh](#) (uint16\_t valve, int32\_t threshold)
- Sets the upper threshold for the analog in port per valve*
- PortDirectionEnumNet [GetDigitalPortDirection](#) (uint16\_t port)
- Gets the direction of a digital port*
- void [SetDigitalPortDirection](#) (uint16\_t port, PortDirectionEnumNet direction)
- Sets the direction of a digital port*
- bool [IsValveDigitalInInverted](#) (uint16\_t valve)
- Is digital in inverted*
- void [SetValveDigitalInInvert](#) (uint16\_t valve, bool isInverted)
- Invert digital in*
- uint32\_t [GetValveDigitalInPort](#) (uint16\_t valve)
- Gets the number of the digital in port which is mapped to a valve*
- void [SetValveDigitalInPort](#) (uint16\_t valve, uint32\_t digitalInPort)
- Map a digital in port to a valve*
- bool [IsDigitalOutPortInverted](#) (uint16\_t digitalOutPort)
- Gets the number of the valve which is mapped to a digital out port*
- void [SetDigitalOutPortInvert](#) (uint16\_t digitalOutPort, bool isInverted)
- Map a valve to a digital out port*
- uint32\_t [GetDigitalOutPortValve](#) (uint16\_t digitalOutPort)
- Gets the number of the valve which is mapped to a digital out port*
- void [SetDigitalOutPortValve](#) (uint16\_t digitalOutPort, uint32\_t valve)
- Map a valve to a digital out port*

- void [SetDefault](#) ()  
*Sets the settings of the valve controller to default*
- bool [IsValveOpen](#) (uint16\_t valve)  
*Is valve open*
- bool [IsValveOpenInDigitalMode](#) (uint16\_t valve)  
*True, if the valve would be open when the device is in digital mode*
- bool [IsValveOpenInAnalogMode](#) (uint16\_t valve)  
*True, if the valve would be open when the device is in analog mode*
- int32\_t [GetAnalogVoltage](#) ()  
*Reads the voltage on the analog in port*
- void [GetValveTableEntry](#) (uint16\_t valve, uint16\_t index, [System::Runtime::InteropServices::Out]uint32\_t% duration, [System::Runtime::InteropServices::Out]bool% state)  
*Read an entry from the valve protocol table*
- void [SetValveTableEntry](#) (uint16\_t valve, uint16\_t index, uint32\_t duration, bool state)  
*Write an entry to the valve protocol table*
- void [ClearValveTable](#) (uint16\_t valve)  
*Clear the valve protocol table*
- void [LoadValveTable](#) ()  
*Load the current table from permanent memory*
- void [StoreValveTable](#) ()  
*Store the current table in permanent memory*
- String ^ [GetTableNamebyIndex](#) (uint16\_t tableNumber)  
*Get the name of a protocol table*
- String ^ [GetTableName](#) ()  
*Get the name of the current protocol table*
- void [SetTableName](#) (String^ tableName)  
*Set the name of the current protocol table*
- uint32\_t [GetActiveRunningTableNumber](#) ()  
*Gets the number of the table that is active for running*
- void [SetActiveRunningTableNumber](#) (uint32\_t tableNumber)  
*Sets the number of the tanle that is active for running*
- uint32\_t [GetCurrentEditTableNumber](#) ()  
*Gets the number of the table that is current for editing*
- void [SetCurrentEditTableNumber](#) (uint32\_t tableNumber)  
*Sets the number of the table that is current for editing*
- void [ClearTableName](#) ()  
*Clear the name of current protocol table*
- void [SetTableStep](#) (uint16\_t valve, int32\_t steps)  
*Skips the table protocol for a valve by steps*
- void [SetTableStepAll](#) (int32\_t steps)  
*Skips the table protocol for all valves by steps*
- int32\_t [GetTotalNumberOfValves](#) ()  
*Get the total number of valves in the system*
- int32\_t [GetTotalNumberOfDigitalPorts](#) ()  
*Get the total number of digital ports in the system*
- int32\_t [GetTotalTableSize](#) ()  
*Get the total table size in the system*
- int32\_t [GetTotalNumberOfTables](#) ()  
*Get the total number of tables in the system*
- int32\_t [GetCurrentNumberOfValves](#) ()  
*Get the current number of valves connected to the system*

- uint32\_t [GetValveBoardRevision](#) ()  
*Gets the revision code of the valve board*
- WvcDisplayModeEnumNet [GetDisplayMode](#) ()  
*Reads the display mode*
- void [SetDisplayMode](#) (WvcDisplayModeEnumNet DisplayMode, int32\_t lockTimeMs)  
*Writes the display mode*
- String ^ [GetValveBoardRevisionString](#) ()  
*Gets the revision name of the valve board*

## Events

- [OnGetValveActive](#) ^ [GetValveActiveEvent](#) [add, remove, raise]  
*Event fires when the valve state for the valve number has changed*
- [OnGetValveManualState](#) ^ [GetValveManualStateEvent](#) [add, remove, raise]  
*Event fires when the manual valve state for the valve number has changed*
- [OnGetValveManualGroup](#) ^ [GetValveManualGroupEvent](#) [add, remove, raise]  
*Event fires when the manual valve group for the valve number has changed*
- [OnGetValveMode](#) ^ [GetValveModeEvent](#) [add, remove, raise]  
*Event fires when the valve mode for the valve number has changed*
- [OnGetAnalogThresholdLow](#) ^ [GetAnalogThresholdLowEvent](#) [add, remove, raise]  
*Event fires when the threshold in mV for the valve number has changed*
- [OnGetAnalogThresholdHigh](#) ^ [GetAnalogThresholdHighEvent](#) [add, remove, raise]  
*Event fires when the threshold in mV for the valve number has changed*
- [OnGetDigitalPortDirection](#) ^ [GetDigitalPortDirectionEvent](#) [add, remove, raise]  
*Event fires when the direction for the port number has changed*
- [OnIsValveDigitalInInverted](#) ^ [IsValveDigitalInInvertedEvent](#) [add, remove, raise]  
*Event fires when is inverted for the valve number has changed*
- [OnGetValveDigitalInPort](#) ^ [GetValveDigitalInPortEvent](#) [add, remove, raise]  
*Event fires when the digital in port for the valve number has changed*
- [OnIsDigitalOutPortInverted](#) ^ [IsDigitalOutPortInvertedEvent](#) [add, remove, raise]  
*Event fires when is inverted for the digital out port has changed*
- [OnGetDigitalOutPortValve](#) ^ [GetDigitalOutPortValveEvent](#) [add, remove, raise]  
*Event fires when the valve number for the digital out port has changed*
- [OnIsValveOpen](#) ^ [IsValveOpenEvent](#) [add, remove, raise]  
*Event fires when is open for the valve number has changed*
- [OnIsValveOpenInDigitalMode](#) ^ [IsValveOpenInDigitalModeEvent](#) [add, remove, raise]  
*Event fires when is open for the valve number has changed*
- [OnIsValveOpenInAnalogMode](#) ^ [IsValveOpenInAnalogModeEvent](#) [add, remove, raise]  
*Event fires when is open for the valve number has changed*
- [OnGetAnalogVoltage](#) ^ [GetAnalogVoltageEvent](#) [add, remove, raise]  
*Event fires when the voltage in mV has changed*
- [OnTableEntryChanged](#) ^ [TableEntryChangedEvent](#) [add, remove, raise]  
*Event fires when an entry of a table changed*
- [OnGetTableNamebyIndex](#) ^ [GetTableNamebyIndexEvent](#) [add, remove, raise]  
*Event fires when the name of the table for the table number has changed*
- [OnGetActiveRunningTableNumber](#) ^ [GetActiveRunningTableNumberEvent](#) [add, remove, raise]  
*Event fires when the table number has changed*
- [OnGetCurrentNumberOfValves](#) ^ [GetCurrentNumberOfValvesEvent](#) [add, remove, raise]  
*Event fires when the number of valves has changed*
- [OnGetValveBoardRevision](#) ^ [GetValveBoardRevisionEvent](#) [add, remove, raise]  
*Event fires when the revision code has changed*
- [OnGetDisplayMode](#) ^ [GetDisplayModeEvent](#) [add, remove, raise]  
*Event fires when the display mode has changed*

## Additional Inherited Members

### 11.120.1 Detailed Description

[CWarnerValveControllerDeviceNet](#) is the class to access the Warner Valve Controller

### 11.120.2 Constructor & Destructor Documentation

#### 11.120.2.1 CWarnerValveControllerDeviceNet() [CWarnerValveControllerDeviceNet](#) ( )

Initializes a new instance of the [CWarnerValveControllerDeviceNet](#) class.

#### 11.120.2.2 ~CWarnerValveControllerDeviceNet() [virtual](#) [~CWarnerValveControllerDeviceNet](#) ( ) [virtual]

#### 11.120.2.3 ~!CWarnerValveControllerDeviceNet() [!CWarnerValveControllerDeviceNet](#) ( )

### 11.120.3 Member Function Documentation

#### 11.120.3.1 ClearTableName() [void](#) [ClearTableName](#) ( )

Clear the name of current protocol table

#### 11.120.3.2 ClearValveTable() [void](#) [ClearValveTable](#) ( [uint16\\_t](#) *valve* )

Clear the valve protocol table

##### Parameters

<i>valve</i>	The valve number
--------------	------------------

#### 11.120.3.3 GetActiveRunningTableNumber() [uint32\\_t](#) [GetActiveRunningTableNumber](#) ( )

Gets the number of the table that is active for running

**Returns**

The table number

**11.120.3.4 GetAnalogThresholdHigh()** `int32_t GetAnalogThresholdHigh ( uint16_t valve )`

Gets the upper threshold for the analog in port per valve

**Parameters**

<i>valve</i>	The valve number
--------------	------------------

**Returns**

The threshold in mV

**11.120.3.5 GetAnalogThresholdLow()** `int32_t GetAnalogThresholdLow ( uint16_t valve )`

Gets the lower threshold for the analog in port per valve

**Parameters**

<i>valve</i>	The valve number
--------------	------------------

**Returns**

The threshold in mV

**11.120.3.6 GetAnalogVoltage()** `int32_t GetAnalogVoltage ( )`

Reads the voltage on the analog in port

**Returns**

The voltage in mV



**11.120.3.7 GetCurrentEditTableNumber()** `uint32_t GetCurrentEditTableNumber ( )`

Gets the number of the table that is current for editing

**Returns**

The table number

**11.120.3.8 GetCurrentNumberOfValves()** `int32_t GetCurrentNumberOfValves ( )`

Get the current number of valves connected to the system

**Returns**

The number of valves

**11.120.3.9 GetDigitalOutPortValve()** `uint32_t GetDigitalOutPortValve (   
uint16_t digitalOutPort )`

Gets the number of the valve which is mapped to a digital out port

**Parameters**

<i>digitalOutPort</i>	The digital out port
-----------------------	----------------------

**Returns**

The valve number

**11.120.3.10 GetDigitalPortDirection()** `PortDirectionEnumNet GetDigitalPortDirection (   
uint16_t port )`

Gets the direction of a digital port

**Parameters**

<i>port</i>	The port number
-------------	-----------------

**Returns**

the direction

**11.120.3.11 GetDisplayMode()** `WvcDisplayModeEnumNet GetDisplayMode ( )`

Reads the display mode

**Returns**

The display mode

**11.120.3.12 GetTableName()** `String ^ GetTableName ( )`

Get the name of the current protocol table

**Returns**

The name of the table

**11.120.3.13 GetTableNamebyIndex()** `String ^ GetTableNamebyIndex (   
uint16_t tableNumber )`

Get the name of a protocol table

**Parameters**

<i>tableNumber</i>	The table number
--------------------	------------------

**Returns**

The name of the table

**11.120.3.14 GetTotalNumberOfDigitalPorts()** `int32_t GetTotalNumberOfDigitalPorts ( )`

Get the total number of digital ports in the system

**Returns**

The number of digital ports

**11.120.3.15 GetTotalNumberOfTables()** `int32_t GetTotalNumberOfTables ( )`

Get the total number of tables in the system

**Returns**

The number of tables

**11.120.3.16 GetTotalNumberOfValves()** `int32_t GetTotalNumberOfValves ( )`

Get the total number of valves in the system

**Returns**

The number of valves

**11.120.3.17 GetTotalTableSize()** `int32_t GetTotalTableSize ( )`

Get the total table size in the system

**Returns**

The table size

**11.120.3.18 GetValveActive()** `int GetValveActive (   
uint16_t valve )`

Gets the valve active/inactive state

**Parameters**

<i>valve</i>	The valve number
--------------	------------------

**Returns**

The valve state

**11.120.3.19 GetValveBoardRevision()** `uint32_t GetValveBoardRevision ( )`

Gets the revision code of the valve board

**Returns**

The revision code

**11.120.3.20 GetValveBoardRevisionString()** `String ^ GetValveBoardRevisionString ( )`

Gets the revision name of the valve board

**Returns**

The revision name

**11.120.3.21 GetValveDigitalInPort()** `uint32_t GetValveDigitalInPort ( uint16_t valve )`

Gets the number of the digital in port which is mapped to a valve

**Parameters**

<i>valve</i>	The valve number
--------------	------------------

**Returns**

The digital in port

**11.120.3.22 GetValveManualGroup()** `int32_t GetValveManualGroup ( uint16_t valve )`

Gets the valve manual group

**Parameters**

<i>valve</i>	The valve number
--------------	------------------

**Returns**

The manual valve group

**11.120.3.23 GetValveManualState()** `int32_t GetValveManualState ( uint16_t valve )`

Gets the valve manual on/off state

**Parameters**

<i>valve</i>	The valve number
--------------	------------------

**Returns**

The manual valve state

**11.120.3.24 GetValveMode()** `WvcValveModeEnumNet GetValveMode ( uint16_t valve )`

Reads the valve mode

## Parameters

<i>valve</i>	The valve number
--------------	------------------

## Returns

The valve mode

**11.120.3.25 GetValvesActiveMap()** uint32\_t GetValvesActiveMap ( )

Gets the valves active/inactive states

## Returns

The valves states

**11.120.3.26 GetValvesManualStateMap()** uint32\_t GetValvesManualStateMap ( )

Gets the valves manual on/off states

## Returns

The manual valves states

**11.120.3.27 GetValveTableEntry()** void GetValveTableEntry (   
uint16\_t *valve*,   
uint16\_t *index*,   
[System::Runtime::InteropServices::Out] uint32\_t% *duration*,   
[System::Runtime::InteropServices::Out] bool% *state* )

Read an entry from the valve protocol table

## Parameters

<i>valve</i>	The valve number
<i>index</i>	The index in the table
<i>duration</i>	the duration in ms
<i>state</i>	the state

**11.120.3.28 IsDigitalOutPortInverted()** bool IsDigitalOutPortInverted (   
uint16\_t *digitalOutPort* )

Gets the number of the valve which is mapped to a digital out port

#### Parameters

<i>digitalOutPort</i>	The digital out port
-----------------------	----------------------

#### Returns

is inverted

**11.120.3.29 IsValveDigitalInInverted()** `bool IsValveDigitalInInverted ( uint16_t valve )`

Is digital in inverted

#### Parameters

<i>valve</i>	The valve number
--------------	------------------

#### Returns

is inverted

**11.120.3.30 IsValveOpen()** `bool IsValveOpen ( uint16_t valve )`

Is valve open

#### Parameters

<i>valve</i>	The valve number
--------------	------------------

#### Returns

is open

**11.120.3.31 IsValveOpenInAnalogMode()** `bool IsValveOpenInAnalogMode ( uint16_t valve )`

True, if the valve would be open when the device is in analog mode

## Parameters

<i>valve</i>	The valve number
--------------	------------------

## Returns

is open

**11.120.3.32 IsValveOpenInDigitalMode()** `bool IsValveOpenInDigitalMode ( uint16_t valve )`

True, if the valve would be open when the device is in digital mode

## Parameters

<i>valve</i>	The valve number
--------------	------------------

## Returns

is open

**11.120.3.33 LoadValveTable()** `void LoadValveTable ( )`

Load the current table from permanent memory

**11.120.3.34 OnGetActiveRunningTableNumber()** `delegate void OnGetActiveRunningTableNumber ( uint32_t tableNumber )`

**11.120.3.35 OnGetAnalogThresholdHigh()** `delegate void OnGetAnalogThresholdHigh ( uint16_t valve, int32_t threshold )`

**11.120.3.36 OnGetAnalogThresholdLow()** `delegate void OnGetAnalogThresholdLow ( uint16_t valve, int32_t threshold )`

**11.120.3.37 OnGetAnalogVoltage()** delegate void OnGetAnalogVoltage (   
int32\_t voltage )

**11.120.3.38 OnGetCurrentNumberOfValves()** delegate void OnGetCurrentNumberOfValves (   
int32\_t numberOfValves )

**11.120.3.39 OnGetDigitalOutPortValve()** delegate void OnGetDigitalOutPortValve (   
uint16\_t digitalOutPort,   
uint32\_t valve )

**11.120.3.40 OnGetDigitalPortDirection()** delegate void OnGetDigitalPortDirection (   
uint16\_t port,   
PortDirectionEnumNet direction )

**11.120.3.41 OnGetDisplayMode()** delegate void OnGetDisplayMode (   
WvcDisplayModeEnumNet DisplayMode )

**11.120.3.42 OnGetTableNamebyIndex()** delegate void OnGetTableNamebyIndex (   
uint16\_t tableNumber,   
String^ tableName )

**11.120.3.43 OnGetValveActive()** delegate void OnGetValveActive (   
uint16\_t valve,   
int valveActive )

**11.120.3.44 OnGetValveBoardRevision()** delegate void OnGetValveBoardRevision (   
uint32\_t revision )

**11.120.3.45 OnGetValveDigitalInPort()** delegate void OnGetValveDigitalInPort (   
uint16\_t valve,   
uint32\_t digitalInPort )



**11.120.3.46 OnGetValveManualGroup()** delegate void OnGetValveManualGroup (   
 uint16\_t valve,   
 int32\_t valveManualGroup )

**11.120.3.47 OnGetValveManualState()** delegate void OnGetValveManualState (   
 uint16\_t valve,   
 int32\_t valveManualState )

**11.120.3.48 OnGetValveMode()** delegate void OnGetValveMode (   
 uint16\_t valve,   
 WvcValveModeEnumNet ValveMode )

**11.120.3.49 OnIsDigitalOutPortInverted()** delegate void OnIsDigitalOutPortInverted (   
 uint16\_t digitalOutPort,   
 bool isInverted )

**11.120.3.50 OnIsValveDigitalInInverted()** delegate void OnIsValveDigitalInInverted (   
 uint16\_t valve,   
 bool isInverted )

**11.120.3.51 OnIsValveOpen()** delegate void OnIsValveOpen (   
 uint16\_t valve,   
 bool valveOpen )

**11.120.3.52 OnIsValveOpenInAnalogMode()** delegate void OnIsValveOpenInAnalogMode (   
 uint16\_t valve,   
 bool valveOpen )

**11.120.3.53 OnIsValveOpenInDigitalMode()** delegate void OnIsValveOpenInDigitalMode (   
 uint16\_t valve,   
 bool valveOpen )

**11.120.3.54 OnTableEntryChanged()** delegate void OnTableEntryChanged (   
 uint16\_t tableNumber )

**11.120.3.55 SetActiveRunningTableNumber()** void SetActiveRunningTableNumber (   
 uint32\_t tableNumber )

Sets the number of the tanle that is active for running

## Parameters

<i>tableNumber</i>	The table number
--------------------	------------------

**11.120.3.56 SetAnalogThresholdHigh()** `void SetAnalogThresholdHigh (`  
    `uint16_t valve,`  
    `int32_t threshold )`

Sets the upper threshold for the analog in port per valve

## Parameters

<i>valve</i>	The valve number
<i>threshold</i>	The threshold in mV

**11.120.3.57 SetAnalogThresholdLow()** `void SetAnalogThresholdLow (`  
    `uint16_t valve,`  
    `int32_t threshold )`

Sets the lower threshold for the analog in port per valve

## Parameters

<i>valve</i>	The valve number
<i>threshold</i>	The threshold in mV

**11.120.3.58 SetCurrentEditTableNumber()** `void SetCurrentEditTableNumber (`  
    `uint32_t tableNumber )`

Sets the number of the table that is current for editing

## Parameters

<i>tableNumber</i>	The table number
--------------------	------------------

**11.120.3.59 SetDefault()** `void SetDefault ( )`

Sets the settings of the valve controller to default

**11.120.3.60 SetDigitalOutPortInvert()** void SetDigitalOutPortInvert (   
    uint16\_t *digitalOutPort*,   
    bool *isInverted* )

Map a valve to a digital out port

Parameters

<i>digitalOutPort</i>	The digital out port
<i>isInverted</i>	True if digital out is to be inverted

**11.120.3.61 SetDigitalOutPortValve()** void SetDigitalOutPortValve (   
    uint16\_t *digitalOutPort*,   
    uint32\_t *valve* )

Map a valve to a digital out port

Parameters

<i>digitalOutPort</i>	The digital out port
<i>valve</i>	The valve number

**11.120.3.62 SetDigitalPortDirection()** void SetDigitalPortDirection (   
    uint16\_t *port*,   
    PortDirectionEnumNet *direction* )

Sets the direction of a digital port

Parameters

<i>port</i>	The port number
<i>direction</i>	the direction

**11.120.3.63 SetDisplayMode()** void SetDisplayMode (   
    WvcDisplayModeEnumNet *DisplayMode*,   
    int32\_t *lockTimeMs* )

Writes the display mode

Parameters

<i>DisplayMode</i>	The display mode
<i>lockTimeMs</i>	Locks the display for ms

**11.120.3.64 SetTableName()** void SetTableName (   
String^ *tableName* )

Set the name of the current protocol table

Parameters

<i>tableName</i>	The name of the table
------------------	-----------------------

**11.120.3.65 SetTableStep()** void SetTableStep (   
uint16\_t *valve*,   
int32\_t *steps* )

Skips the table protocol for a valve by steps

Parameters

<i>valve</i>	The valve number
<i>steps</i>	Number of steps

**11.120.3.66 SetTableStepAll()** void SetTableStepAll (   
int32\_t *steps* )

Skips the table protocol for all valves by steps

Parameters

<i>steps</i>	Number of steps
--------------	-----------------

**11.120.3.67 SetValveActive()** void SetValveActive (   
uint16\_t *valve*,   
int *valveActive* )

Sets the valve active/inactive state

Parameters

<i>valve</i>	The valve number
<i>valveActive</i>	The valve state

**11.120.3.68 SetValveDigitalInInvert()** `void SetValveDigitalInInvert (`  
    `uint16_t valve,`  
    `bool isInverted )`

Invert digital in

Parameters

<i>valve</i>	The valve number
<i>isInverted</i>	True if digital in is to be inverted

**11.120.3.69 SetValveDigitalInPort()** `void SetValveDigitalInPort (`  
    `uint16_t valve,`  
    `uint32_t digitalInPort )`

Map a digital in port to a valve

Parameters

<i>valve</i>	The valve number
<i>digitalInPort</i>	The digital in port

**11.120.3.70 SetValveManualGroup()** `void SetValveManualGroup (`  
    `uint16_t valve,`  
    `int32_t valveManualGroup )`

Sets the valve manual group

Parameters

<i>valve</i>	The valve number
<i>valveManualGroup</i>	The manual valve group

**11.120.3.71 SetValveManualState()** `void SetValveManualState (`  
    `uint16_t valve,`  
    `int32_t valveManualState )`

Sets the valve manual on/off state

## Parameters

<i>valve</i>	The valve number
<i>valveManualState</i>	The manual valve state

**11.120.3.72 SetValveMode()** void SetValveMode (   
     uint16\_t valve,   
     WvcValveModeEnumNet ValveMode )

Writes the valve mode

## Parameters

<i>valve</i>	The valve number
<i>ValveMode</i>	The valve mode

**11.120.3.73 SetValvesActiveMap()** void SetValvesActiveMap (   
     uint32\_t valvesActive )

Sets the valve active/inactive state

## Parameters

<i>valvesActive</i>	The valves states
---------------------	-------------------

**11.120.3.74 SetValvesManualStateMap()** void SetValvesManualStateMap (   
     uint32\_t valveaManualState )

Sets the valve manual on/off state

## Parameters

<i>valveaManualState</i>	The manual valves states
--------------------------	--------------------------

**11.120.3.75 SetValveTableEntry()** void SetValveTableEntry (   
     uint16\_t valve,   
     uint16\_t index,   
     uint32\_t duration,   
     bool state )

Write an entry to the valve protocol table

## Parameters

<i>valve</i>	The valve number
<i>index</i>	The index in the table
<i>duration</i>	the duration in ms
<i>state</i>	the state

**11.120.3.76 StoreValveTable()** `void StoreValveTable ( )`

Store the current table in permanent memory

**11.120.4 Event Documentation****11.120.4.1 GetActiveRunningTableNumberEvent** `OnGetActiveRunningTableNumber^ GetActiveRunning↔  
TableNumberEvent [add], [remove], [raise]`

Event fires when the table number has changed

**11.120.4.2 GetAnalogThresholdHighEvent** `OnGetAnalogThresholdHigh^ GetAnalogThresholdHighEvent  
[add], [remove], [raise]`

Event fires when the threshold in mV for the valve number has changed

**11.120.4.3 GetAnalogThresholdLowEvent** `OnGetAnalogThresholdLow^ GetAnalogThresholdLowEvent  
[add], [remove], [raise]`

Event fires when the threshold in mV for the valve number has changed

**11.120.4.4 GetAnalogVoltageEvent** `OnGetAnalogVoltage^ GetAnalogVoltageEvent [add], [remove],  
[raise]`

Event fires when the voltage in mV has changed

**11.120.4.5 GetCurrentNumberOfValvesEvent** [OnGetCurrentNumberOfValves](#)<sup>^</sup> [GetCurrentNumberOfValvesEvent](#) [add], [remove], [raise]

Event fires when the number of valves has changed

**11.120.4.6 GetDigitalOutPortValveEvent** [OnGetDigitalOutPortValve](#)<sup>^</sup> [GetDigitalOutPortValveEvent](#) [add], [remove], [raise]

Event fires when the valve number for the digital out port has changed

**11.120.4.7 GetDigitalPortDirectionEvent** [OnGetDigitalPortDirection](#)<sup>^</sup> [GetDigitalPortDirectionEvent](#) [add], [remove], [raise]

Event fires when the direction for the port number has changed

**11.120.4.8 GetDisplayModeEvent** [OnGetDisplayMode](#)<sup>^</sup> [GetDisplayModeEvent](#) [add], [remove], [raise]

Event fires when the display mode has changed

**11.120.4.9 GetTableNamebyIndexEvent** [OnGetTableNamebyIndex](#)<sup>^</sup> [GetTableNamebyIndexEvent](#) [add], [remove], [raise]

Event fires when the name of the table for the table number has changed

**11.120.4.10 GetValveActiveEvent** [OnGetValveActive](#)<sup>^</sup> [GetValveActiveEvent](#) [add], [remove], [raise]

Event fires when the valve state for the valve number has changed

**11.120.4.11 GetValveBoardRevisionEvent** [OnGetValveBoardRevision](#)<sup>^</sup> [GetValveBoardRevisionEvent](#) [add], [remove], [raise]

Event fires when the revision code has changed



**11.120.4.12 GetValveDigitalInPortEvent** [OnGetValveDigitalInPort](#)<sup>^</sup> GetValveDigitalInPortEvent [add], [remove], [raise]

Event fires when the digital in port for the valve number has changed

**11.120.4.13 GetValveManualGroupEvent** [OnGetValveManualGroup](#)<sup>^</sup> GetValveManualGroupEvent [add], [remove], [raise]

Event fires when the manual valve group for the valve number has changed

**11.120.4.14 GetValveManualStateEvent** [OnGetValveManualState](#)<sup>^</sup> GetValveManualStateEvent [add], [remove], [raise]

Event fires when the manual valve state for the valve number has changed

**11.120.4.15 GetValveModeEvent** [OnGetValveMode](#)<sup>^</sup> GetValveModeEvent [add], [remove], [raise]

Event fires when the valve mode for the valve number has changed

**11.120.4.16 IsDigitalOutPortInvertedEvent** [OnIsDigitalOutPortInverted](#)<sup>^</sup> IsDigitalOutPortInverted↔ Event [add], [remove], [raise]

Event fires when is inverted for the digital out port has changed

**11.120.4.17 IsValveDigitalInInvertedEvent** [OnIsValveDigitalInInverted](#)<sup>^</sup> IsValveDigitalInInverted↔ Event [add], [remove], [raise]

Event fires when is inverted for the valve number has changed

**11.120.4.18 IsValveOpenEvent** [OnIsValveOpen](#)<sup>^</sup> IsValveOpenEvent [add], [remove], [raise]

Event fires when is open for the valve number has changed

**11.120.4.19 IsValveOpenInAnalogModeEvent** [OnIsValveOpenInAnalogMode](#)<sup>^</sup> [IsValveOpenInAnalogMode](#)↔  
Event [add], [remove], [raise]

Event fires when is open for the valve number has changed

**11.120.4.20 IsValveOpenInDigitalModeEvent** [OnIsValveOpenInDigitalMode](#)<sup>^</sup> [IsValveOpenInDigital](#)↔  
ModeEvent [add], [remove], [raise]

Event fires when is open for the valve number has changed

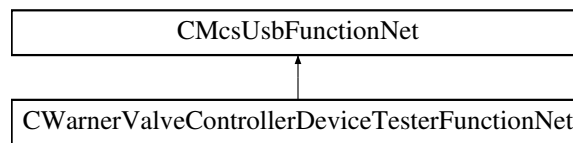
**11.120.4.21 TableEntryChangedEvent** [OnTableEntryChanged](#)<sup>^</sup> [TableEntryChangedEvent](#) [add], [remove], [raise]

Event fires when an entry of a table changed

## 11.121 CWarnerValveControllerDeviceTesterFunctionNet Class Reference

[CWarnerValveControllerDeviceTesterFunctionNet](#) is the class to access the functions for the Warner Valve Controller Device Tester

Inheritance diagram for CWarnerValveControllerDeviceTesterFunctionNet:



### Public Member Functions

- [CWarnerValveControllerDeviceTesterFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> pWarnerValveControllerDeviceTesterFunctionPointerContainer)  
*Initializes a new instance of the [CWarnerValveControllerDeviceTesterFunctionNet](#) class.*
- [CWarnerValveControllerDeviceTesterFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- virtual [~CWarnerValveControllerDeviceTesterFunctionNet](#) ()
- [!CWarnerValveControllerDeviceTesterFunctionNet](#) ()
- void [SetADC](#) (uint32\_t onoff)  
*Sets the ADC port of the tester*
- uint32\_t [GetSync](#) ()  
*Gets the output from the sync port*
- void [SetTrigger](#) (uint32\_t trigger)  
*Sets the input to the trigger port*
- void [SetTriggerSyncDirection](#) (uint32\_t direction)  
*Sets the direction of the trigger/sync test port*
- uint32\_t [GetIO](#) ()  
*Gets the output from the io ports*
- void [SetIO](#) (uint32\_t io)  
*Sets the input to the io ports*
- void [SetIODirection](#) (uint32\_t direction)  
*Sets the direction of the IO test ports*

## Additional Inherited Members

### 11.121.1 Detailed Description

[CWarnerValveControllerDeviceTesterFunctionNet](#) is the class to access the functions for the Warner Valve Controller Device Tester

### 11.121.2 Constructor & Destructor Documentation

**11.121.2.1 CWarnerValveControllerDeviceTesterFunctionNet() [1/2]** [CWarnerValveControllerDeviceTesterFunctionNet](#)  
(  
    [CMcsUsbNet](#)<sup>^</sup> *mcsusb*,  
    [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> *pWarnerValveControllerDeviceTesterFunction*  
    *PointerContainer* )

Initializes a new instance of the [CWarnerValveControllerDeviceTesterFunctionNet](#) class.

**11.121.2.2 CWarnerValveControllerDeviceTesterFunctionNet() [2/2]** [CWarnerValveControllerDeviceTesterFunctionNet](#)  
(  
    [CMcsUsbNet](#)<sup>^</sup> *mcsusb* )

**11.121.2.3 ~CWarnerValveControllerDeviceTesterFunctionNet()** virtual [~CWarnerValveControllerDeviceTesterFunctionNet](#)  
( ) [virtual]

**11.121.2.4 !CWarnerValveControllerDeviceTesterFunctionNet()** [!CWarnerValveControllerDeviceTesterFunctionNet](#)  
( )

### 11.121.3 Member Function Documentation

**11.121.3.1 GetIO()** uint32\_t GetIO ( )

Gets the output from the io ports

#### Returns

The manual valves states

**11.121.3.2 GetSync()** `uint32_t GetSync ( )`

Gets the output from the sync port

**Returns**

The sync state

**11.121.3.3 SetADC()** `void SetADC (   
uint32_t onoff )`

Sets the ADC port of the tester

**Parameters**

<i>onoff</i>	The port state
--------------	----------------

**11.121.3.4 SetIO()** `void SetIO (   
uint32_t io )`

Sets the input to the io ports

**Parameters**

<i>io</i>	The manual valves states
-----------	--------------------------

**11.121.3.5 SetIODirection()** `void SetIODirection (   
int32_t direction )`

Sets the direction of the IO test ports

**Parameters**

<i>direction</i>	The 16bit direction map: 1=IN 0=OUT
------------------	-------------------------------------

**11.121.3.6 SetTrigger()** `void SetTrigger (   
uint32_t trigger )`

Sets the input to the trigger port

## Parameters

<i>trigger</i>	The trigger state
----------------	-------------------

**11.121.3.7 SetTriggerSyncDirection()** `void SetTriggerSyncDirection ( uint32_t direction )`

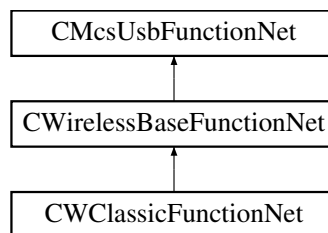
Sets the direction of the trigger/sync test port

## Parameters

<i>direction</i>	The direction: 1=IN 0=OUT
------------------	---------------------------

## 11.122 CWClassicFunctionNet Class Reference

Inheritance diagram for CWClassicFunctionNet:



### Public Member Functions

- [CWClassicFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> wClassicFuntionPointerContainer)
- [CWClassicFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- `uint32_t` [ResetChannelmap](#) (unsigned int virtualDevice)
- `uint32_t` [SetChannelmap](#) (unsigned char position, unsigned char channel, unsigned int Device)
- `void` [SetHWSelectedChannels](#) (array< bool ><sup>^</sup> channels, unsigned int Device)
- `void` [SetRFLostBehaviour](#) (uint8\_t stoponfailure, unsigned int Device)
- `void` [SetHeadstageOnOff](#) (uint16\_t onoff)
- `USHORT` [GetHeadstageOnOff](#) ()
- `void` [SetRFFrequencyHeadstage](#) (uint8\_t receiver\_nb, unsigned short frequency)
- `unsigned short` [GetRFFrequencyHeadstage](#) (uint8\_t receiver\_nb)
- `void` [SetRFFrequencyReceiver](#) (uint8\_t receiver\_nb, uint8\_t configuration, unsigned short frequency)
- `void` [SetRFFrequencyReceiverEeprom](#) (uint8\_t receiver\_nb, uint8\_t configuration, unsigned short frequency)
- `unsigned short` [GetRFFrequencyReceiver](#) (uint8\_t receiver\_nb, uint8\_t configuration)
- `void` [SetSerialNumberHeadstage](#) (unsigned short number)
- `unsigned short` [GetSerialNumberHeadstage](#) ()
- `void` [SetSelectedHeadstage](#) (uint8\_t number)
- `uint8_t` [GetSelectedHeadstage](#) ()
- `void` [ScanForHeadstages](#) ()
- `uint8_t` [GetScanHeadstagesResult](#) (int max\_wait\_for\_ms)

- void [SetFilterParametersHeadstage](#) (unsigned short index, array< int > ^ buffer)
- array< int > ^ [GetFilterParametersHeadstage](#) (unsigned short index)
- bool [GetHasRedLedHeadstage](#) ()
- void [SetHasChecksum](#) (unsigned int has, unsigned int Device)
- unsigned int [GetHasChecksum](#) (unsigned int Device)
- void [SetResetFilter](#) (unsigned int reset, unsigned int Device)
- unsigned int [GetResetFilter](#) (unsigned int Device)
- void [SetWPAType](#) (unsigned short type, unsigned int Device)
- unsigned short [GetWPAType](#) (unsigned int Device)
- void [SetWPADebugMode](#) (unsigned int mode, unsigned int Device)
- unsigned int [GetWPADebugMode](#) (unsigned int Device)
- void [SetRFPower](#) (unsigned short power)
- unsigned short [GetRFPower](#) ()
- unsigned int [GetRFConnectionStatus](#) ()

## Additional Inherited Members

### 11.122.1 Constructor & Destructor Documentation

**11.122.1.1 CWClassicFunctionNet()** [1/2] [CWClassicFunctionNet](#) (  
[CMcsUsbNet](#) ^ *mcsusb*,  
[CMcsUsbFunctionPointerContainer](#) ^ *wClassicFuntionPointerContainer* )

**11.122.1.2 CWClassicFunctionNet()** [2/2] [CWClassicFunctionNet](#) (  
[CMcsUsbNet](#) ^ *mcsusb* )

### 11.122.2 Member Function Documentation

**11.122.2.1 GetFilterParametersHeadstage()** array<int> ^ [GetFilterParametersHeadstage](#) (  
 unsigned short *index* )

**11.122.2.2 GetHasChecksum()** unsigned int [GetHasChecksum](#) (  
 unsigned int *Device* )

**11.122.2.3 GetHasRedLedHeadstage()** bool [GetHasRedLedHeadstage](#) ( )

**11.122.2.4 GetHeadstageOnOff()** USHORT GetHeadstageOnOff ( )

**11.122.2.5 GetResetFilter()** unsigned int GetResetFilter (   
 unsigned int *Device* )

**11.122.2.6 GetRFConnectionStatus()** unsigned int GetRFConnectionStatus ( )

**11.122.2.7 GetRFFrequencyHeadstage()** unsigned short GetRFFrequencyHeadstage (   
 uint8\_t *receiver\_nb* )

**11.122.2.8 GetRFFrequencyReceiver()** unsigned short GetRFFrequencyReceiver (   
 uint8\_t *receiver\_nb*,   
 uint8\_t *configuration* )

**11.122.2.9 GetRFPower()** unsigned short GetRFPower ( )

**11.122.2.10 GetScanHeadstagesResult()** uint8\_t GetScanHeadstagesResult (   
 int *max\_wait\_for\_ms* )

**11.122.2.11 GetSelectedHeadstage()** uint8\_t GetSelectedHeadstage ( )

**11.122.2.12 GetSerialNumberHeadstage()** unsigned short GetSerialNumberHeadstage ( )

**11.122.2.13 GetWPADebugMode()** unsigned int GetWPADebugMode (   
 unsigned int *Device* )

**11.122.2.14 GetWPAType()** unsigned short GetWPAType (   
 unsigned int *Device* )

**11.122.2.15 ResetChannelmap()** uint32\_t ResetChannelmap (   
 unsigned int *virtualDevice* )

**11.122.2.16 ScanForHeadstages()** void ScanForHeadstages ( )

**11.122.2.17 SetChannelmap()** uint32\_t SetChannelmap (   
 unsigned char *position*,   
 unsigned char *channel*,   
 unsigned int *Device* )

**11.122.2.18 SetFilterParametersHeadstage()** void SetFilterParametersHeadstage (   
 unsigned short *index*,   
 array< int >^ *buffer* )

**11.122.2.19 SetHasChecksum()** void SetHasChecksum (   
 unsigned int *has*,   
 unsigned int *Device* )

**11.122.2.20 SetHeadstageOnOff()** void SetHeadstageOnOff (   
 uint16\_t *onoff* )

**11.122.2.21 SetHWSelectedChannels()** void SetHWSelectedChannels (   
 array< bool >^ *channels*,   
 unsigned int *Device* )

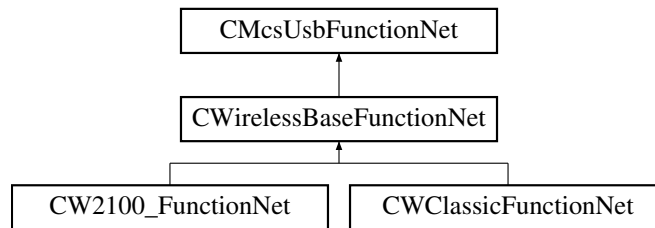
**11.122.2.22 SetResetFilter()** void SetResetFilter (   
 unsigned int *reset*,   
 unsigned int *Device* )



- 11.122.2.23 SetRFFrequencyHeadstage()** void SetRFFrequencyHeadstage (   
     uint8\_t *receiver\_nb*,   
     unsigned short *frequency* )
- 11.122.2.24 SetRFFrequencyReceiver()** void SetRFFrequencyReceiver (   
     uint8\_t *receiver\_nb*,   
     uint8\_t *configuration*,   
     unsigned short *frequency* )
- 11.122.2.25 SetRFFrequencyReceiverEeprom()** void SetRFFrequencyReceiverEeprom (   
     uint8\_t *receiver\_nb*,   
     uint8\_t *configuration*,   
     unsigned short *frequency* )
- 11.122.2.26 SetRFLostBehaviour()** void SetRFLostBehaviour (   
     uint8\_t *stoponfailure*,   
     unsigned int *Device* )
- 11.122.2.27 SetRFPower()** void SetRFPower (   
     unsigned short *power* )
- 11.122.2.28 SetSelectedHeadstage()** void SetSelectedHeadstage (   
     uint8\_t *number* )
- 11.122.2.29 SetSerialNumberHeadstage()** void SetSerialNumberHeadstage (   
     unsigned short *number* )
- 11.122.2.30 SetWPADebugMode()** void SetWPADebugMode (   
     unsigned int *mode*,   
     unsigned int *Device* )
- 11.122.2.31 SetWPAType()** void SetWPAType (   
     unsigned short *type*,   
     unsigned int *Device* )

## 11.123 CWirelessBaseFunctionNet Class Reference

Inheritance diagram for CWirelessBaseFunctionNet:



### Public Member Functions

- [CWirelessBaseFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> mcsusbfunction)

### Static Public Member Functions

- static String <sup>^</sup> [CreateWirelessHeadstageSerialNumberString](#) (unsigned short ID)

### Additional Inherited Members

#### 11.123.1 Constructor & Destructor Documentation

**11.123.1.1 CWirelessBaseFunctionNet()** [CWirelessBaseFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb,  
[CMcsUsbFunctionPointerContainer](#)<sup>^</sup> mcsusbfunction )

#### 11.123.2 Member Function Documentation

**11.123.2.1 CreateWirelessHeadstageSerialNumberString()** static String <sup>^</sup> [CreateWirelessHeadstageSerialNumberString](#) (  
 unsigned short ID ) [static]

## 11.124 DeviceIdNet Struct Reference

Device Id.

## Public Member Functions

- [DeviceIdNet](#) ()
- [DeviceIdNet](#) (VendorIdEnumNet vendor, ProductIdEnumNet product, int bcd, McsBusTypeEnumNet bustype)
- [DeviceIdNet](#) ([DeviceIdNet](#)% deviceId)
- [DeviceIdNet](#) operator= ([DeviceIdNet](#)% deviceId)

## Public Attributes

- VendorIdEnumNet [IdVendor](#)
- ProductIdEnumNet [IdProduct](#)
- int [BcdDevice](#)
- McsBusTypeEnumNet [BusType](#)

### 11.124.1 Detailed Description

Device Id.

### 11.124.2 Constructor & Destructor Documentation

#### 11.124.2.1 DeviceIdNet() [1/3] [DeviceIdNet](#) ( )

#### 11.124.2.2 DeviceIdNet() [2/3] [DeviceIdNet](#) ( VendorIdEnumNet *vendor*, ProductIdEnumNet *product*, int *bcd*, McsBusTypeEnumNet *bustype* )

#### 11.124.2.3 DeviceIdNet() [3/3] [DeviceIdNet](#) ( [DeviceIdNet](#)% *deviceId* )

### 11.124.3 Member Function Documentation

#### 11.124.3.1 operator=() [DeviceIdNet](#) operator= ( [DeviceIdNet](#)% *deviceId* )

#### 11.124.4 Member Data Documentation

**11.124.4.1 BcdDevice** `int BcdDevice`

**11.124.4.2 BusType** `McsBusTypeEnumNet BusType`

**11.124.4.3 IdProduct** `ProductIdEnumNet IdProduct`

**11.124.4.4 IdVendor** `VendorIdEnumNet IdVendor`

### 11.125 DigitalSource< digitalsourceenum > Class Template Reference

#### Public Member Functions

- [DigitalSource](#) (digitalsourceenum source)
- int [MaxBitNumber](#) ()

#### Static Public Member Functions

- static int [size](#) ()

#### Properties

- digitalsourceenum [Source](#) [get]

#### 11.125.1 Constructor & Destructor Documentation

**11.125.1.1 DigitalSource()** `DigitalSource (digitalsourceenum source )`

#### 11.125.2 Member Function Documentation

**11.125.2.1 MaxBitNumber()** `int MaxBitNumber ( )`

**11.125.2.2 size()** `static int size ( ) [static]`

### 11.125.3 Property Documentation

**11.125.3.1 Source** `digitalsourceenum Source [get]`

## 11.126 DriverVersionNet Class Reference

Class gives firmware versions of the device's firmware destinations.

### Public Member Functions

- [DriverVersionNet](#) ()  
*Constructor.*
- [~DriverVersionNet](#) ()  
*Destructor.*
- unsigned int [GetStatus](#) (CFirmwareDestinationNet dest)  
*Get status of firmware destination.*
- unsigned int [GetStatus](#) (unsigned int index)  
*Get status of firmware destination.*
- unsigned int [GetVersionInt](#) (CFirmwareDestinationNet dest)  
*Get the version number of firmware destination (major in high word, minor in low word)*
- unsigned int [GetVersionInt](#) (unsigned int index)  
*Get the version number of firmware destination (major in high word, minor in low word)*
- unsigned int [GetMajor](#) (CFirmwareDestinationNet dest)  
*Get the major version number of firmware destination.*
- unsigned int [GetMajor](#) (unsigned int index)  
*Get the major version number of firmware destination.*
- unsigned int [GetMinor](#) (CFirmwareDestinationNet dest)  
*Get the minor version number of firmware destination.*
- unsigned int [GetMinor](#) (unsigned int index)  
*Get the minor version number of firmware destination.*
- unsigned int [GetNumEntries](#) ()  
*Get the number of available firmware destinations.*
- String ^ [GetVersionString](#) (CFirmwareDestinationNet dest)  
*Get the version as a string in the format Major.Minor.*
- String ^ [GetVersionString](#) (unsigned int index)  
*Get the version as a string in the format Major.Minor.*
- CFirmwareDestinationNet [GetDestinationCode](#) (unsigned int index)  
*Get CFirmwareDestinationNet.*
- String ^ [GetDestinationName](#) (CFirmwareDestinationNet dest)  
*Get firmware destination name.*
- String ^ [GetDestinationName](#) (unsigned int index)  
*Get firmware destination name.*

**Static Public Member Functions**

- static String ^ [DriverVersionNet::FormatVersion](#) (unsigned int v)

**11.126.1 Detailed Description**

Class gives firmware versions of the device's firmware destinations.

**11.126.2 Constructor & Destructor Documentation****11.126.2.1 DriverVersionNet()** [DriverVersionNet](#) ( )

Constructor.

**11.126.2.2 ~DriverVersionNet()** [~DriverVersionNet](#) ( )

Destructor.

**11.126.3 Member Function Documentation****11.126.3.1 DriverVersionNet::FormatVersion()** static String ^ [DriverVersionNet::FormatVersion](#) ( unsigned int v ) [static]**11.126.3.2 GetDestinationCode()** CFirmwareDestinationNet [GetDestinationCode](#) ( unsigned int index )

Get CFirmwareDestinationNet.

**Parameters**

<i>index</i>	by index of firmware destination
--------------	----------------------------------

**11.126.3.3 GetDestinationName()** [1/2] String ^ [GetDestinationName](#) ( CFirmwareDestinationNet dest )

Get firmware destination name.

## Parameters

<i>dest</i>	by CFirmwareDestiationNet
-------------	---------------------------

**11.126.3.4 GetDestinationName()** [2/2] `String ^ GetDestinationName ( unsigned int index )`

Get firmware destination name.

## Parameters

<i>index</i>	by index of firmware destination
--------------	----------------------------------

**11.126.3.5 GetMajor()** [1/2] `unsigned int GetMajor ( CFirmwareDestiationNet dest )`

Get the major version number of firmware destination.

## Parameters

<i>dest</i>	by CFirmwareDestiationNet
-------------	---------------------------

**11.126.3.6 GetMajor()** [2/2] `unsigned int GetMajor ( unsigned int index )`

Get the major version number of firmware destination.

## Parameters

<i>index</i>	by index of firmware destination
--------------	----------------------------------

**11.126.3.7 GetMinor()** [1/2] `unsigned int GetMinor ( CFirmwareDestiationNet dest )`

Get the minor version number of firmware destination.

## Parameters

<i>dest</i>	by CFirmwareDestiationNet
-------------	---------------------------



**11.126.3.8 GetMinor()** [2/2] unsigned int GetMinor (   
 unsigned int *index* )

Get the minor version number of firmware destination.

Parameters

<i>index</i>	by index of firmware destination
--------------	----------------------------------

**11.126.3.9 GetNumEntries()** unsigned int GetNumEntries ( )

Get the number of available firmware destinations.

**11.126.3.10 GetStatus()** [1/2] unsigned int GetStatus (   
 CFirmwareDestinationNet *dest* )

Get status of firmware destination.

Parameters

<i>dest</i>	by CFirmwareDestinationNet
-------------	----------------------------

**11.126.3.11 GetStatus()** [2/2] unsigned int GetStatus (   
 unsigned int *index* )

Get status of firmware destination.

Parameters

<i>index</i>	by index of firmware destination
--------------	----------------------------------

**11.126.3.12 GetVersionInt()** [1/2] unsigned int GetVersionInt (   
 CFirmwareDestinationNet *dest* )

Get the version number of firmware destination (major in high word, minor in low word)

## Parameters

<i>dest</i>	by CFirmwareDestinationNet
-------------	----------------------------

**11.126.3.13 GetVersionInt()** [2/2] unsigned int GetVersionInt (   
 unsigned int *index* )

Get the version number of firmware destination (major in high word, minor in low word)

## Parameters

<i>index</i>	by index of firmware destination
--------------	----------------------------------

**11.126.3.14 GetVersionString()** [1/2] String ^ GetVersionString (   
 CFirmwareDestinationNet *dest* )

Get the version as a string in the format Major.Minor.

## Parameters

<i>dest</i>	by CFirmwareDestinationNet
-------------	----------------------------

**11.126.3.15 GetVersionString()** [2/2] String ^ GetVersionString (   
 unsigned int *index* )

Get the version as a string in the format Major.Minor.

## Parameters

<i>index</i>	by index of firmware
--------------	----------------------

## 11.127 FirmwareDestinationNames Class Reference

### Static Public Attributes

- static String ^ **DSP** = gcnw String( "DSP" )
- static String ^ **USB** = gcnw String( "USB" )
- static String ^ **MCU1** = gcnw String( "MCU1" )
- static String ^ **Bootstrap** = gcnw String( "Bootstrap" )
- static String ^ **MCSBUS1** = gcnw String( "McsBus1" )

- static String ^ [MCSBUS2](#) = gcnew String( "McsBus2" )
- static String ^ [MCSBUS3](#) = gcnew String( "McsBus3" )
- static String ^ [MCSBUS4](#) = gcnew String( "McsBus4" )
- static String ^ [MCSBUS5](#) = gcnew String( "McsBus5" )
- static String ^ [MCSBUS6](#) = gcnew String( "McsBus6" )
- static String ^ [MCSBUS7](#) = gcnew String( "McsBus7" )
- static String ^ [MCSBUS8](#) = gcnew String( "McsBus8" )
- static String ^ [MCSBUS9](#) = gcnew String( "McsBus9" )
- static String ^ [MCSBUS10](#) = gcnew String( "McsBus10" )
- static String ^ [MCSBUS11](#) = gcnew String( "McsBus11" )
- static String ^ [MCSBUS12](#) = gcnew String( "McsBus12" )
- static String ^ [MCSBUS13](#) = gcnew String( "McsBus13" )
- static String ^ [BUS1\\_MCSBUS1](#) = gcnew String( "Bus1McsBus1" )
- static String ^ [BUS1\\_MCSBUS2](#) = gcnew String( "Bus1McsBus2" )
- static String ^ [PIC](#) = gcnew String( "PIC" )
- static String ^ [PIC2](#) = gcnew String( "PIC2" )
- static String ^ [PIC3](#) = gcnew String( "PIC3" )
- static String ^ [PIC4](#) = gcnew String( "PIC4" )
- static String ^ [Altera](#) = gcnew String( "Altera" )
- static String ^ [FPGA2](#) = gcnew String( "FPGA2" )
- static String ^ [FPGA3](#) = gcnew String( "FPGA3" )
- static String ^ [FPGA4](#) = gcnew String( "FPGA4" )
- static String ^ [FPGA5](#) = gcnew String( "FPGA5" )
- static String ^ [FPGA6](#) = gcnew String( "FPGA6" )

### 11.127.1 Member Data Documentation

**11.127.1.1 Altera** String ^ Altera = gcnew String( "Altera" ) [static]

**11.127.1.2 Bootstrap** String ^ Bootstrap = gcnew String( "Bootstrap" ) [static]

**11.127.1.3 BUS1\_MCSBUS1** String ^ BUS1\_MCSBUS1 = gcnew String( "Bus1McsBus1" ) [static]

**11.127.1.4 BUS1\_MCSBUS2** String ^ BUS1\_MCSBUS2 = gcnew String( "Bus1McsBus2" ) [static]

**11.127.1.5 DSP** String ^ DSP = gcnew String( "DSP" ) [static]

**11.127.1.6 FPGA2** `String ^ FPGA2 = gcnew String( "FPGA2" ) [static]`

**11.127.1.7 FPGA3** `String ^ FPGA3 = gcnew String( "FPGA3" ) [static]`

**11.127.1.8 FPGA4** `String ^ FPGA4 = gcnew String( "FPGA4" ) [static]`

**11.127.1.9 FPGA5** `String ^ FPGA5 = gcnew String( "FPGA5" ) [static]`

**11.127.1.10 FPGA6** `String ^ FPGA6 = gcnew String( "FPGA6" ) [static]`

**11.127.1.11 MCSBUS1** `String ^ MCSBUS1 = gcnew String( "McsBus1" ) [static]`

**11.127.1.12 MCSBUS10** `String ^ MCSBUS10 = gcnew String( "McsBus10" ) [static]`

**11.127.1.13 MCSBUS11** `String ^ MCSBUS11 = gcnew String( "McsBus11" ) [static]`

**11.127.1.14 MCSBUS12** `String ^ MCSBUS12 = gcnew String( "McsBus12" ) [static]`

**11.127.1.15 MCSBUS13** `String ^ MCSBUS13 = gcnew String( "McsBus13" ) [static]`

**11.127.1.16 MCSBUS2** `String ^ MCSBUS2 = gcnew String( "McsBus2" ) [static]`

**11.127.1.17 MCSBUS3** `String ^ MCSBUS3 = gcnew String( "McsBus3" ) [static]`

**11.127.1.18 MCSBUS4** `String ^ MCSBUS4 = gcnew String( "McsBus4" ) [static]`

**11.127.1.19 MCSBUS5** `String ^ MCSBUS5 = gcnew String( "McsBus5" ) [static]`

**11.127.1.20 MCSBUS6** `String ^ MCSBUS6 = gcnew String( "McsBus6" ) [static]`

**11.127.1.21 MCSBUS7** `String ^ MCSBUS7 = gcnew String( "McsBus7" ) [static]`

**11.127.1.22 MCSBUS8** `String ^ MCSBUS8 = gcnew String( "McsBus8" ) [static]`

**11.127.1.23 MCSBUS9** `String ^ MCSBUS9 = gcnew String( "McsBus9" ) [static]`

**11.127.1.24 MCU1** `String ^ MCU1 = gcnew String( "MCU1" ) [static]`

**11.127.1.25 PIC** `String ^ PIC = gcnew String( "PIC" ) [static]`

**11.127.1.26 PIC2** `String ^ PIC2 = gcnew String( "PIC2" ) [static]`

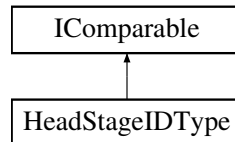
**11.127.1.27 PIC3** `String ^ PIC3 = gcnew String( "PIC3" ) [static]`

**11.127.1.28 PIC4** `String ^ PIC4 = gcnew String( "PIC4" ) [static]`

**11.127.1.29 USB** `String ^ USB = gcnew String( "USB" ) [static]`

## 11.128 HeadStageIDType Class Reference

Inheritance diagram for HeadStageIDType:



### Public Types

- enum [HeadstageTypeEnum](#) {  
[Unknown](#),  
[MeasuringOnly](#),  
[OpticalStimulation](#),  
[ElectricalStimulation](#) }

### Public Member Functions

- [HeadStageIDType](#) (unsigned int entry, [CW2100\\_FunctionNet](#)^ device)
- virtual `System::String ^ ToString ()` override
- virtual `bool Equals (Object^ obj)` override
- virtual `Int32 CompareTo (Object^ obj)`

### Properties

- `bool Valid` [get]
- `unsigned int Entry` [get]
- `unsigned short ID` [get]
- `System::String^ SN` [get]
- `unsigned int TypeValue` [get]
- `System::String^ Type` [get]
- [HeadstageTypeEnum](#) [HeadstageType](#) [get]
- `System::String^ UserDefinedName` [get]
- `int NumberOfAnalogChannels` [get]
- `int NumberOfStimulationChannels` [get]
- [W2100\\_StimulusParametersNet](#)^ [StimulusParameters](#) [get]
- `bool HasIMU` [get]
- `bool W16IsW14` [get]
- `bool HasOptoCurrentMeasurement` [get]

## 11.128.1 Member Enumeration Documentation

**11.128.1.1 HeadstageTypeEnum** enum [HeadstageTypeEnum](#) [strong]

## Enumerator

Unknown	
MeasuringOnly	
OpticalStimulation	
ElectricalStimulation	

## 11.128.2 Constructor &amp; Destructor Documentation

**11.128.2.1 HeadStageIDType()** `HeadStageIDType ( unsigned int entry, CW2100_FunctionNet^ device )`

## 11.128.3 Member Function Documentation

**11.128.3.1 CompareTo()** `virtual Int32 CompareTo ( Object^ obj ) [virtual]`

**11.128.3.2 Equals()** `virtual bool Equals ( Object^ obj ) [override], [virtual]`

**11.128.3.3 ToString()** `virtual System::String ^ ToString ( ) [override], [virtual]`

## 11.128.4 Property Documentation

**11.128.4.1 Entry** `unsigned int Entry [get]`

**11.128.4.2 HasIMU** `bool HasIMU [get]`

**11.128.4.3 HasOptoCurrentMessurement** bool HasOptoCurrentMessurement [get]

**11.128.4.4 HeadstageType** [HeadstageTypeEnum](#) HeadstageType [get]

**11.128.4.5 ID** unsigned short ID [get]

**11.128.4.6 NumberOfAnalogChannels** int NumberOfAnalogChannels [get]

**11.128.4.7 NumberOfStimulationChannels** int NumberOfStimulationChannels [get]

**11.128.4.8 SN** System:: String^ SN [get]

**11.128.4.9 StimulusParameters** [W2100\\_StimulusParametersNet](#)^ StimulusParameters [get]

**11.128.4.10 Type** System:: String^ Type [get]

**11.128.4.11 TypeValue** unsigned int TypeValue [get]

**11.128.4.12 UserDefinedName** System:: String^ UserDefinedName [get]

**11.128.4.13 Valid** bool Valid [get]

**11.128.4.14 W16IsW14** bool W16IsW14 [get]



## 11.129 HeadstageIDTypeObject Class Reference

### Public Member Functions

- [HeadstageIDTypeObject](#) ([HeadStageIDType](#)^ *idType*)
- virtual [String](#) ^ [ToString](#) () override
- virtual bool [Equals](#) ([Object](#)^ *obj*) override
- virtual int [GetHashCode](#) () override

### Public Attributes

- [HeadStageIDType](#) ^ [\\_IdType](#)
- [String](#) ^ [\\_AdditionalText](#)

### Properties

- [HeadStageIDType](#)^ [IdType](#) [get]
- [String](#)^ [AdditionalText](#) [get, set]

### 11.129.1 Constructor & Destructor Documentation

**11.129.1.1 [HeadstageIDTypeObject\(\)](#)** [HeadstageIDTypeObject](#) (  
    [HeadStageIDType](#)^ *idType* )

### 11.129.2 Member Function Documentation

**11.129.2.1 [Equals\(\)](#)** virtual bool [Equals](#) (  
    [Object](#)^ *obj* ) [override], [virtual]

**11.129.2.2 [GetHashCode\(\)](#)** virtual int [GetHashCode](#) ( ) [override], [virtual]

**11.129.2.3 [ToString\(\)](#)** virtual [String](#) ^ [ToString](#) ( ) [override], [virtual]

### 11.129.3 Member Data Documentation

**11.129.3.1** `_AdditionalText` `String` ^ `_AdditionalText`

**11.129.3.2** `_IdType` `HeadStageIDType` ^ `_IdType`

#### 11.129.4 Property Documentation

**11.129.4.1** `AdditionalText` `String`^ `AdditionalText` `[get]`, `[set]`

**11.129.4.2** `IdType` `HeadStageIDType`^ `IdType` `[get]`

### 11.130 HeadStageIDTypeState Class Reference

#### Properties

- unsigned int `State` `[get]`
- `HeadStageIDType`^ `IdType` `[get]`
- bool `ControlState` `[get]`
- bool `DataState` `[get]`

#### 11.130.1 Property Documentation

**11.130.1.1** `ControlState` `bool` `ControlState` `[get]`

**11.130.1.2** `DataState` `bool` `DataState` `[get]`

**11.130.1.3** `IdType` `HeadStageIDType`^ `IdType` `[get]`

**11.130.1.4** `State` `unsigned int` `State` `[get]`

## 11.131 mkfilterNet Class Reference

### Static Public Member Functions

- static int [mkfilter](#) (String<sup>^</sup> *filtertype*, double *value*, String<sup>^</sup> *passtype*, int *order*, double *alpha1*, double *alpha2*, [System::Runtime::InteropServices::Out] array< double ><sup>^</sup>% *xcoeffs*, [System::Runtime::InteropServices::Out] array< double ><sup>^</sup>% *ycoeffs*)
- static int [mkfilter\\_MCS](#) (int *SamplesPerSecond*, double *R1*, double *R2*, double *C*, double *Amplification*, double *Correction*, [System::Runtime::InteropServices::Out] array< double ><sup>^</sup>% *xcoeffs*, [System::Runtime::InteropServices::Out] array< double ><sup>^</sup>% *ycoeffs*)
- static int [mkfilter\\_MCS](#) (int *SamplesPerSecond*, double *R1*, double *R2*, double *C*, double *Correction*, [System::Runtime::InteropServices::Out] array< double ><sup>^</sup>% *xcoeffs*, [System::Runtime::InteropServices::Out] array< double ><sup>^</sup>% *ycoeffs*)
- static int [mkfilter\\_MCS\\_k](#) (int *SamplesPerSecond*, double *R1*, double *R2*, double *C*, double *Amplification*, double *Correction*, [System::Runtime::InteropServices::Out] array< double ><sup>^</sup>% *coeffs*)
- static int [mkfilter\\_MCS\\_k](#) (int *SamplesPerSecond*, double *R1*, double *R2*, double *C*, double *Correction*, [System::Runtime::InteropServices::Out] array< double ><sup>^</sup>% *coeffs*)
- static void [mkfilter\\_coef\\_in\\_one\\_set](#) (int *n*, [System::Runtime::InteropServices::In] array< double ><sup>^</sup> *xcoeffs*, [System::Runtime::InteropServices::In] array< double ><sup>^</sup> *ycoeffs*, [System::Runtime::InteropServices::Out] array< double ><sup>^</sup>% *out\_coeffs*)
- static void [mkfilter\\_scale\\_coef\\_in\\_one\\_set](#) (int *n*, double *scale*, [System::Runtime::InteropServices::In] array< double ><sup>^</sup> *xcoeffs*, [System::Runtime::InteropServices::In] array< double ><sup>^</sup> *ycoeffs*, [System::Runtime::InteropServices::Out] array< double ><sup>^</sup>% *out\_coeffs*)
- static void [mkfilter\\_normalize\\_coeffs\\_short](#) (short *maxvalue*, [System::Runtime::InteropServices::In] array< double ><sup>^</sup> *coeffs*, [System::Runtime::InteropServices::Out] array< short ><sup>^</sup>% *out\_coeffs*)
- static void [mkfilter\\_normalize\\_coeffs\\_int](#) (int *maxvalue*, [System::Runtime::InteropServices::In] array< double ><sup>^</sup> *coeffs*, [System::Runtime::InteropServices::Out] array< int ><sup>^</sup>% *out\_coeffs*)
- static void [mkfilter\\_normalize\\_scale\\_coeffs\\_int](#) (int *maxvalue*, [System::Runtime::InteropServices::In] array< double ><sup>^</sup> *coeffs*, [System::Runtime::InteropServices::Out] array< int ><sup>^</sup>% *out\_coeffs*)
- static double [mkfilter\\_highpass\\_coeff](#) (int *SamplesPerSecond*, double *Frequency*)
- static double [mkfilter\\_highpass\\_k](#) (int *SamplesPerSecond*, double *Frequency*)
- static double [mkfilter\\_highpass\\_frequency\\_from\\_coeff](#) (int *SamplesPerSecond*, double *coeff*)
- static double [mkfilter\\_highpass\\_frequency\\_from\\_k](#) (int *SamplesPerSecond*, double *k*)

### 11.131.1 Member Function Documentation

**11.131.1.1 [mkfilter\(\)](#)** static int [mkfilter](#) (  
String<sup>^</sup> *filtertype*,  
double *value*,  
String<sup>^</sup> *passtype*,  
int *order*,  
double *alpha1*,  
double *alpha2*,  
[System::Runtime::InteropServices::Out] array< double ><sup>^</sup>% *xcoeffs*,  
[System::Runtime::InteropServices::Out] array< double ><sup>^</sup>% *ycoeffs* ) [static]

**11.131.1.2 mkfilter\_coef\_in\_one\_set()** static void mkfilter\_coef\_in\_one\_set (   
     int *n*,   
     [System::Runtime::InteropServices::In] array< double >^ *xcoeffs*,   
     [System::Runtime::InteropServices::In] array< double >^ *ycoeffs*,   
     [System::Runtime::InteropServices::Out] array< double >^% *out\_coeffs* ) [static]

**11.131.1.3 mkfilter\_highpass\_coeff()** static double mkfilter\_highpass\_coeff (   
     int *SamplesPerSecond*,   
     double *Frequency* ) [static]

**11.131.1.4 mkfilter\_highpass\_frequency\_from\_coeff()** static double mkfilter\_highpass\_frequency\_from↵  
     from\_coeff (   
         int *SamplesPerSecond*,   
         double *coeff* ) [static]

**11.131.1.5 mkfilter\_highpass\_frequency\_from\_k()** static double mkfilter\_highpass\_frequency\_from↵  
     \_k (   
         int *SamplesPerSecond*,   
         double *k* ) [static]

**11.131.1.6 mkfilter\_highpass\_k()** static double mkfilter\_highpass\_k (   
     int *SamplesPerSecond*,   
     double *Frequency* ) [static]

**11.131.1.7 mkfilter\_MCS() [1/2]** static int mkfilter\_MCS (   
     int *SamplesPerSecond*,   
     double *R1*,   
     double *R2*,   
     double *C*,   
     double *Amplification*,   
     double *Correction*,   
     [System::Runtime::InteropServices::Out] array< double >^% *xcoeffs*,   
     [System::Runtime::InteropServices::Out] array< double >^% *ycoeffs* ) [static]

**11.131.1.8 mkfilter\_MCS() [2/2]** static int mkfilter\_MCS (   
     int *SamplesPerSecond*,   
     double *R1*,   
     double *R2*,   
     double *C*,   
     double *Correction*,   
     [System::Runtime::InteropServices::Out] array< double >^% *xcoeffs*,   
     [System::Runtime::InteropServices::Out] array< double >^% *ycoeffs* ) [static]

- 11.131.1.9 mkfilter\_MCS\_k()** [1/2] static int mkfilter\_MCS\_k (   
int *SamplesPerSecond*,  
double *R1*,  
double *R2*,  
double *C*,  
double *Amplification*,  
double *Correction*,  
[System::Runtime::InteropServices::Out] array< double >^% *coeffs* ) [static]
- 11.131.1.10 mkfilter\_MCS\_k()** [2/2] static int mkfilter\_MCS\_k (   
int *SamplesPerSecond*,  
double *R1*,  
double *R2*,  
double *C*,  
double *Correction*,  
[System::Runtime::InteropServices::Out] array< double >^% *coeffs* ) [static]
- 11.131.1.11 mkfilter\_normalize\_coeffs\_int()** static void mkfilter\_normalize\_coeffs\_int (   
int *maxvalue*,  
[System::Runtime::InteropServices::In] array< double >^ *coeffs*,  
[System::Runtime::InteropServices::Out] array< int >^% *out\_coeffs* ) [static]
- 11.131.1.12 mkfilter\_normalize\_coeffs\_short()** static void mkfilter\_normalize\_coeffs\_short (   
short *maxvalue*,  
[System::Runtime::InteropServices::In] array< double >^ *coeffs*,  
[System::Runtime::InteropServices::Out] array< short >^% *out\_coeffs* ) [static]
- 11.131.1.13 mkfilter\_normalize\_scale\_coeffs\_int()** static void mkfilter\_normalize\_scale\_coeffs\_int (   
int *maxvalue*,  
[System::Runtime::InteropServices::In] array< double >^ *coeffs*,  
[System::Runtime::InteropServices::Out] array< int >^% *out\_coeffs* ) [static]
- 11.131.1.14 mkfilter\_scale\_coef\_in\_one\_set()** static void mkfilter\_scale\_coef\_in\_one\_set (   
int *n*,  
double *scale*,  
[System::Runtime::InteropServices::In] array< double >^ *xcoeffs*,  
[System::Runtime::InteropServices::In] array< double >^ *ycoeffs*,  
[System::Runtime::InteropServices::Out] array< double >^% *out\_coeffs* ) [static]

## 11.132 CRoboDeviceNet::RoboMainLowLevelCommands Class Reference

### Public Member Functions

- void [SetParameter](#) (unsigned short command, unsigned short index, unsigned int value)
- void [SetParameter](#) (unsigned short command, unsigned short index, unsigned int value1, unsigned int value2)
- void [SetUserParameter](#) (unsigned short index, unsigned int value)  
*Stores persistently 32 bit integer values on motor controller*
- void [SetUserParameter](#) (unsigned short index, int value)  
*Stores persistently 32 bit integer values on motor controller*
- void [GetParameter](#) (unsigned short command, unsigned short index, [System::Runtime::InteropServices::Out]unsigned int% value)
- void [GetParameter](#) (unsigned short command, unsigned short index, [System::Runtime::InteropServices::Out]unsigned int% value1, [System::Runtime::InteropServices::Out]unsigned int% value2)
- void [GetUserParameter](#) (unsigned short index, [System::Runtime::InteropServices::Out]unsigned int% value)  
*Reads 32 bit integer values stored persistently on motor controller*
- void [GetUserParameter](#) (unsigned short index, [System::Runtime::InteropServices::Out]int% value)  
*Reads 32 bit integer values stored persistently on motor controller*
- void [FindReferencePhase0](#) (unsigned char busaddress, char axes)
- void [FindReferencePhase0](#) (unsigned char busaddress, char axes, int timeout)
- unsigned char [HasRef](#) (unsigned char busaddress, char axes)
- void [SetHWRevision](#) (unsigned int revision)
- unsigned int [GetHWRevision](#) ()
- void [SetHWConfig](#) (unsigned int config)
- unsigned int [GetHWConfig](#) ()
- void [SetMinPressureWaitTime](#) (unsigned int t)
- unsigned int [GetMinPressureWaitTime](#) ()
- void [SetMinPressure](#) (unsigned int pressure)
- unsigned int [GetMinPressure](#) ()
- void [SetMaxPressureWaitTime](#) (unsigned int t)
- unsigned int [GetMaxPressureWaitTime](#) ()
- void [SetMinNoPressureWaitTime](#) (unsigned int t)
- unsigned int [GetMinNoPressureWaitTime](#) ()
- void [SetMaxNoPressure](#) (unsigned int pressure)
- unsigned int [GetMaxNoPressure](#) ()
- void [SetMaxNoPressureWaitTime](#) (unsigned int t)
- unsigned int [GetMaxNoPressureWaitTime](#) ()
- void [SetSearchReferenceMethod](#) (unsigned char busaddress, char axes, unsigned int method)
- unsigned int [GetSearchReferenceMethod](#) (unsigned char busaddress, char axes)
- void [SetSearchReferenceOffsetPos](#) (unsigned char busaddress, char axes, int offsetpos)
- int [GetSearchReferenceOffsetPos](#) (unsigned char busaddress, char axes)
- void [SetSearchReferenceFastSpeed](#) (unsigned char busaddress, char axes, unsigned short speed)
- unsigned short [GetSearchReferenceFastSpeed](#) (unsigned char busaddress, char axes)
- void [SetSearchReferenceFastAccel](#) (unsigned char busaddress, char axes, unsigned short accel)
- unsigned short [GetSearchReferenceFastAccel](#) (unsigned char busaddress, char axes)
- void [SetSearchReferenceFineSpeed](#) (unsigned char busaddress, char axes, unsigned short speed)
- unsigned short [GetSearchReferenceFineSpeed](#) (unsigned char busaddress, char axes)
- void [SetSearchReferenceFineAccel](#) (unsigned char busaddress, char axes, unsigned short accel)
- unsigned short [GetSearchReferenceFineAccel](#) (unsigned char busaddress, char axes)
- void [SetSearchReferenceMoveOut](#) (unsigned char busaddress, char axes, int move)
- int [GetSearchReferenceMoveOut](#) (unsigned char busaddress, char axes)
- void [SetAxisConfig](#) (unsigned char busaddress, char axes, unsigned int config)
- unsigned int [GetAxisConfig](#) (unsigned char busaddress, char axes)
- void [GetPhases](#) (unsigned char busaddress, char axes, [System::Runtime::InteropServices::Out] unsigned short% phase0, [System::Runtime::InteropServices::Out] unsigned short% lastphase)

### 11.132.1 Member Function Documentation

**11.132.1.1 FindReferencePhase0()** [1/2] void FindReferencePhase0 (   
 unsigned char *busaddress*,  
 char *axes* )

**11.132.1.2 FindReferencePhase0()** [2/2] void FindReferencePhase0 (   
 unsigned char *busaddress*,  
 char *axes*,  
 int *timeout* )

**11.132.1.3 GetAxisConfig()** unsigned int GetAxisConfig (   
 unsigned char *busaddress*,  
 char *axes* )

**11.132.1.4 GetHWConfig()** unsigned int GetHWConfig ( )

**11.132.1.5 GetHWRevision()** unsigned int GetHWRevision ( )

**11.132.1.6 GetMaxNoPressure()** unsigned int GetMaxNoPressure ( )

**11.132.1.7 GetMaxNoPressureWaitTime()** unsigned int GetMaxNoPressureWaitTime ( )

**11.132.1.8 GetMaxPressureWaitTime()** unsigned int GetMaxPressureWaitTime ( )

**11.132.1.9 GetMinNoPressureWaitTime()** unsigned int GetMinNoPressureWaitTime ( )

**11.132.1.10 GetMinPressure()** unsigned int GetMinPressure ( )

**11.132.1.11 GetMinPressureWaitTime()** unsigned int GetMinPressureWaitTime ( )

**11.132.1.12 GetParameter() [1/2]** void GetParameter (   
 unsigned short *command*,   
 unsigned short *index*,   
 [System::Runtime::InteropServices::Out] unsigned int% *value* )

**11.132.1.13 GetParameter() [2/2]** void GetParameter (   
 unsigned short *command*,   
 unsigned short *index*,   
 [System::Runtime::InteropServices::Out] unsigned int% *value1*,   
 [System::Runtime::InteropServices::Out] unsigned int% *value2* )

**11.132.1.14 GetPhases()** void GetPhases (   
 unsigned char *busaddress*,   
 char *axes*,   
 [System::Runtime::InteropServices::Out] unsigned short% *phase0*,   
 [System::Runtime::InteropServices::Out] unsigned short% *lastphase* )

**11.132.1.15 GetSearchReferenceFastAccel()** unsigned short GetSearchReferenceFastAccel (   
 unsigned char *busaddress*,   
 char *axes* )

**11.132.1.16 GetSearchReferenceFastSpeed()** unsigned short GetSearchReferenceFastSpeed (   
 unsigned char *busaddress*,   
 char *axes* )

**11.132.1.17 GetSearchReferenceFineAccel()** unsigned short GetSearchReferenceFineAccel (   
 unsigned char *busaddress*,   
 char *axes* )



**11.132.1.18 GetSearchReferenceFineSpeed()** unsigned short GetSearchReferenceFineSpeed ( unsigned char *busaddress*, char *axes* )

**11.132.1.19 GetSearchReferenceMethod()** unsigned int GetSearchReferenceMethod ( unsigned char *busaddress*, char *axes* )

**11.132.1.20 GetSearchReferenceMoveOut()** int GetSearchReferenceMoveOut ( unsigned char *busaddress*, char *axes* )

**11.132.1.21 GetSearchReferenceOffsetPos()** int GetSearchReferenceOffsetPos ( unsigned char *busaddress*, char *axes* )

**11.132.1.22 GetUserParameter() [1/2]** void GetUserParameter ( unsigned short *index*, [System::Runtime::InteropServices::Out] int% *value* )

Reads 32 bit integer values stored persistently on motor controller

intention: provide free persistent user memory space on motor controller

#### Parameters

<i>index</i>	address offset of parameter; range: 0..15
<i>value</i>	data buffer

**11.132.1.23 GetUserParameter() [2/2]** void GetUserParameter ( unsigned short *index*, [System::Runtime::InteropServices::Out] unsigned int% *value* )

Reads 32 bit integer values stored persistently on motor controller

intention: provide free persistent user memory space on motor controller

#### Parameters

<i>index</i>	address offset of parameter; range: 0..15
<i>value</i>	data buffer

**11.132.1.24 HasRef()** unsigned char HasRef (   
 unsigned char *busaddress*,   
 char *axes* )

**11.132.1.25 SetAxisConfig()** void SetAxisConfig (   
 unsigned char *busaddress*,   
 char *axes*,   
 unsigned int *config* )

**11.132.1.26 SetHWConfig()** void SetHWConfig (   
 unsigned int *config* )

**11.132.1.27 SetHWRevision()** void SetHWRevision (   
 unsigned int *revision* )

**11.132.1.28 SetMaxNoPressure()** void SetMaxNoPressure (   
 unsigned int *pressure* )

**11.132.1.29 SetMaxNoPressureWaitTime()** void SetMaxNoPressureWaitTime (   
 unsigned int *t* )

**11.132.1.30 SetMaxPressureWaitTime()** void SetMaxPressureWaitTime (   
 unsigned int *t* )

**11.132.1.31 SetMinNoPressureWaitTime()** void SetMinNoPressureWaitTime (   
 unsigned int *t* )

**11.132.1.32 SetMinPressure()** void SetMinPressure (   
 unsigned int *pressure* )

**11.132.1.33 SetMinPressureWaitTime()** void SetMinPressureWaitTime (   
 unsigned int *t* )

**11.132.1.34 SetParameter()** [1/2] void SetParameter (   
 unsigned short *command*,   
 unsigned short *index*,   
 unsigned int *value* )

**11.132.1.35 SetParameter()** [2/2] void SetParameter (   
 unsigned short *command*,   
 unsigned short *index*,   
 unsigned int *value1*,   
 unsigned int *value2* )

**11.132.1.36 SetSearchReferenceFastAccel()** void SetSearchReferenceFastAccel (   
 unsigned char *busaddress*,   
 char *axes*,   
 unsigned short *accel* )

**11.132.1.37 SetSearchReferenceFastSpeed()** void SetSearchReferenceFastSpeed (   
 unsigned char *busaddress*,   
 char *axes*,   
 unsigned short *speed* )

**11.132.1.38 SetSearchReferenceFineAccel()** void SetSearchReferenceFineAccel (   
 unsigned char *busaddress*,   
 char *axes*,   
 unsigned short *accel* )

**11.132.1.39 SetSearchReferenceFineSpeed()** void SetSearchReferenceFineSpeed (   
 unsigned char *busaddress*,   
 char *axes*,   
 unsigned short *speed* )

**11.132.1.40 SetSearchReferenceMethod()** void SetSearchReferenceMethod (   
     unsigned char *busaddress*,   
     char *axes*,   
     unsigned int *method* )

**11.132.1.41 SetSearchReferenceMoveOut()** void SetSearchReferenceMoveOut (   
     unsigned char *busaddress*,   
     char *axes*,   
     int *move* )

**11.132.1.42 SetSearchReferenceOffsetPos()** void SetSearchReferenceOffsetPos (   
     unsigned char *busaddress*,   
     char *axes*,   
     int *offsetpos* )

**11.132.1.43 SetUserParameter() [1/2]** void SetUserParameter (   
     unsigned short *index*,   
     int *value* )

Stores *persistently* 32 bit integer values on motor controller

intention: provide free persistent user memory space on motor controller

#### Parameters

<i>index</i>	address offset of parameter; range: 0..15
<i>value</i>	data to be stored

**11.132.1.44 SetUserParameter() [2/2]** void SetUserParameter (   
     unsigned short *index*,   
     unsigned int *value* )

Stores *persistently* 32 bit integer values on motor controller

intention: provide free persistent user memory space on motor controller

#### Parameters

<i>index</i>	address offset of parameter; range: 0..15
<i>value</i>	data to be stored

## 11.133 CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands Class Reference

### Public Member Functions

- void [FindReferencePhase0XY](#) ()
- void [FindReferencePhase0XY](#) (int timeout)

#### 11.133.1 Member Function Documentation

**11.133.1.1 FindReferencePhase0XY()** [1/2] `void FindReferencePhase0XY ( )`

**11.133.1.2 FindReferencePhase0XY()** [2/2] `void FindReferencePhase0XY (   
int timeout )`

## 11.134 CMeaAudioFunctionNet::s\_setaudionet Struct Reference

### Public Attributes

- int [channel](#)
- int [amplification](#)

#### 11.134.1 Member Data Documentation

**11.134.1.1 amplification** `int amplification`

**11.134.1.2 channel** `int channel`

## 11.135 CStimulusFunctionNet::SidebandData Class Reference

### Public Member Functions

- [SidebandData](#) ()
- [~SidebandData](#) ()  
*Destructor: called by Dispose()*
- [!SidebandData](#) ()  
*Finalizer: called by GC before collecting*

## Properties

- `array< int32_t >^ Sideband` [get]
- `array< uint64_t >^ Duration` [get]

## 11.135.1 Constructor & Destructor Documentation

**11.135.1.1 SidebandData()** `SidebandData ( )`

**11.135.1.2 ~SidebandData()** `~SidebandData ( )`

Destructor: called by Dispose()

**11.135.1.3 "!SidebandData()** `!SidebandData ( )`

Finalizer: called by GC before collecting

## 11.135.2 Property Documentation

**11.135.2.1 Duration** `array< uint64_t >^ Duration` [get]

**11.135.2.2 Sideband** `array< int32_t >^ Sideband` [get]

## 11.136 StgStatusNet Class Reference

### Static Public Member Functions

- static `StgStatusNet ^ FromIntPtr` (IntPtr stgstatus)
- static `StgStatusNet ^ FromPtr` (stgstatus\_t \*stgstatus)

### Public Attributes

- `array< Stg200xTriggerStatusEnumNet > ^ TiggerStatus`
- `array< uint32_t > ^ ListOfChangedTriggers`

### 11.136.1 Member Function Documentation

**11.136.1.1 FromIntPtr()** static [StgStatusNet](#) ^ FromIntPtr (   
 IntPtr stgstatus ) [static]

**11.136.1.2 FromPtr()** static [StgStatusNet](#) ^ FromPtr (   
 stgstatus\_t \* stgstatus ) [static]

### 11.136.2 Member Data Documentation

**11.136.2.1 ListOfChangedTriggers** array<uint32\_t> ^ ListOfChangedTriggers

**11.136.2.2 TiggerStatus** array<Stg200xTriggerStatusEnumNet> ^ TiggerStatus

## 11.137 CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData Class Reference

### Public Member Functions

- [StimulusDeviceDataAndUnrolledData](#) ()
- [~StimulusDeviceDataAndUnrolledData](#) ()  
*Destructor: called by Dispose()*
- [!StimulusDeviceDataAndUnrolledData](#) ()  
*Finalizer: called by GC before collecting*

### Properties

- array< uint8\_t >^ [DeviceData](#) [get]
- int [DeviceDataLength](#) [get]
- array< int32\_t >^ [UnrolledAmplitude](#) [get]
- array< uint32\_t >^ [UnrolledSync](#) [get]
- array< uint64\_t >^ [UnrolledDuration](#) [get]

### 11.137.1 Constructor & Destructor Documentation

**11.137.1.1 StimulusDeviceDataAndUnrolledData()** [StimulusDeviceDataAndUnrolledData \( \)](#)

**11.137.1.2 ~StimulusDeviceDataAndUnrolledData()** [~StimulusDeviceDataAndUnrolledData \( \)](#)

Destructor: called by Dispose()

**11.137.1.3 ~!StimulusDeviceDataAndUnrolledData()** [!StimulusDeviceDataAndUnrolledData \( \)](#)

Finalizer: called by GC before collecting

## 11.137.2 Property Documentation

**11.137.2.1 DeviceData** [array< uint8\\_t>^ DeviceData \[get\]](#)

**11.137.2.2 DeviceDataLength** [int DeviceDataLength \[get\]](#)

**11.137.2.3 UnrolledAmplitude** [array< int32\\_t>^ UnrolledAmplitude \[get\]](#)

**11.137.2.4 UnrolledDuration** [array< uint64\\_t>^ UnrolledDuration \[get\]](#)

**11.137.2.5 UnrolledSync** [array< uint32\\_t>^ UnrolledSync \[get\]](#)

## 11.138 usbSetupPacket\_t Class Reference

### Public Attributes

- [uint8\\_t bmRequestType](#)
- [uint8\\_t bRequest](#)
- [uint16\\_t wValue](#)
- [uint16\\_t wIndex](#)
- [uint16\\_t wLength](#)



### 11.138.1 Member Data Documentation

**11.138.1.1 bmRequestType** `uint8_t bmRequestType`

**11.138.1.2 bRequest** `uint8_t bRequest`

**11.138.1.3 wIndex** `uint16_t wIndex`

**11.138.1.4 wLength** `uint16_t wLength`

**11.138.1.5 wValue** `uint16_t wValue`

## 11.139 W2100\_StimulusParametersNet Struct Reference

### Public Attributes

- int [DACResolution](#)
- int [TimeResolutionInNanoSeconds](#)
- int [VoltageRangeInMicroVolt](#)
- int [VoltageResolutionInMicroVolt](#)
- int [CurrentRangeInNanoAmp](#)
- int [CurrentResolutionInNanoAmp](#)

### 11.139.1 Member Data Documentation

**11.139.1.1 CurrentRangeInNanoAmp** `int CurrentRangeInNanoAmp`

**11.139.1.2 CurrentResolutionInNanoAmp** `int CurrentResolutionInNanoAmp`

**11.139.1.3 DACResolution** `int DACResolution`

**11.139.1.4 TimeResolutionInNanoSeconds** `int TimeResolutionInNanoSeconds`

**11.139.1.5 VoltageRangeInMicroVolt** `int VoltageRangeInMicroVolt`

**11.139.1.6 VoltageResolutionInMicroVolt** `int VoltageResolutionInMicroVolt`

## Index

!CDacCalibrationFunctionNet  
    CDacCalibrationFunctionNet, [50](#)  
!CDigOutStimulatorFunctionNet  
    CDigOutStimulatorFunctionNet, [59](#)  
!CExternDTesterDeviceNet  
    CExternDTesterDeviceNet, [64](#)  
!CInterfaceboardFunctionNet  
    CInterfaceboardFunctionNet, [108](#)  
!CLIH3DeviceNet  
    CLIH3DeviceNet, [111](#)  
!CMEA2100x256FunctionNet  
    CMEA2100x256FunctionNet, [248](#)  
!CMcsUsbFunctionNet  
    CMcsUsbFunctionNet, [218](#)  
!CMcsUsbListNet  
    CMcsUsbListNet, [224](#)  
!CMcsUsbNet  
    CMcsUsbNet, [230](#)  
!CMeFunctionNet  
    CMeFunctionNet, [282](#)  
!CMeaCleanDeviceNet  
    CMeaCleanDeviceNet, [254](#)  
!CMeaCoatDeviceNet  
    CMeaCoatDeviceNet, [258](#)  
!CMultiBatteryChargerDeviceNet  
    CMultiBatteryChargerDeviceNet, [284](#)  
!CMultiwellCallbackFunctionNet  
    CMultiwellCallbackFunctionNet, [291](#)  
!CMultiwellDeviceNet  
    CMultiwellDeviceNet, [295](#)  
!CMultiwellOptoStimFunctionNet  
    CMultiwellOptoStimFunctionNet, [300](#)  
!CPPCFunctionNet  
    CPPCFunctionNet, [327](#)  
!CPedoterDeviceNet  
    CPedoterDeviceNet, [314](#)  
!CPositionIIDeviceNet  
    CPositionIIDeviceNet, [319](#)  
!CPositionImpDeviceNet  
    CPositionImpDeviceNet, [322](#)  
!CProgramPressureCurveNet  
    CProgramPressureCurveNet, [339](#)  
!CPulseGeneratorFunctionNet  
    CPulseGeneratorFunctionNet, [341](#)  
!CRFFunctionNet  
    CRFFunctionNet, [349](#)  
!CSCUFunctionNet  
    CSCUFunctionNet, [401](#)  
!CTEERFunctionNet  
    CTEERFunctionNet, [491](#)  
!CWarnerUssingDeviceNet  
    CWarnerUssingDeviceNet, [517](#)  
!CWarnerUssingFunctionNet  
    CWarnerUssingFunctionNet, [520](#)  
!CWarnerValveControllerDeviceNet  
    CWarnerValveControllerDeviceNet, [538](#)  
!CWarnerValveControllerDeviceTesterFunctionNet  
    CWarnerValveControllerDeviceTesterFunctionNet,  
    [558](#)  
!SidebandData  
    CStimulusFunctionNet::SidebandData, [593](#)  
!StimulusDeviceDataAndUnrolledData  
    CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData,  
    [595](#)  
\_AdditionalText  
    HeadstageIDTypeObject, [580](#)  
\_IdType  
    HeadstageIDTypeObject, [581](#)  
~CCMOSMeaDeviceNet  
    CCMOSMeaDeviceNet, [45](#)  
~CChannelTestDeviceNet  
    CChannelTestDeviceNet, [33](#)  
~CCreateFilterNet  
    CCreateFilterNet, [48](#)  
~CDacCalibrationFunctionNet  
    CDacCalibrationFunctionNet, [50](#)  
~CDigOutStimulatorFunctionNet  
    CDigOutStimulatorFunctionNet, [59](#)  
~CExternDTesterDeviceNet  
    CExternDTesterDeviceNet, [64](#)  
~CFilterCoefficientsNet  
    CFilterCoefficientsNet, [66](#)  
~CFilterPropertyNet  
    CFilterPropertyNet, [72](#)  
~CFluidControlDeviceNet  
    CFluidControlDeviceNet, [74](#)  
~CGenericDevelopDeviceNet  
    CGenericDevelopDeviceNet, [88](#)  
~CGilsonDeviceNet  
    CGilsonDeviceNet, [99](#)  
~CInterfaceboardFunctionNet  
    CInterfaceboardFunctionNet, [108](#)  
~CLIH3DeviceNet  
    CLIH3DeviceNet, [111](#)  
~CMEA2100x256FunctionNet  
    CMEA2100x256FunctionNet, [248](#)  
~CMcsBusNet  
    CMcsBusNet, [155](#)  
~CMcsBus\_AxisParametersNet  
    CMcsBus\_AxisParametersNet, [117](#)  
~CMcsBus\_ExtensionNet  
    CMcsBus\_ExtensionNet, [118](#)  
~CMcsBus\_FYIExtensionNet  
    CMcsBus\_FYIExtensionNet, [119](#)  
~CMcsBus\_MotorControlNet  
    CMcsBus\_MotorControlNet, [124](#)  
~CMcsBus\_SensorNet  
    CMcsBus\_SensorNet, [140](#)  
~CMcsBus\_TempSensorNet  
    CMcsBus\_TempSensorNet, [149](#)

- ~CMcsBus\_VoltageModeNet
  - CMcsBus\_VoltageModeNet, [151](#)
- ~CMcsUsbDacqNet
  - CMcsUsbDacqNet, [165](#)
- ~CMcsUsbFactoryNet
  - CMcsUsbFactoryNet, [211](#)
- ~CMcsUsbFunctionNet
  - CMcsUsbFunctionNet, [218](#)
- ~CMcsUsbListEntryNet
  - CMcsUsbListEntryNet, [220](#)
- ~CMcsUsbListNet
  - CMcsUsbListNet, [224](#)
- ~CMcsUsbNet
  - CMcsUsbNet, [230](#)
- ~CMeFunctionNet
  - CMeFunctionNet, [282](#)
- ~CMeaCleanDeviceNet
  - CMeaCleanDeviceNet, [253](#)
- ~CMeaCoatDeviceNet
  - CMeaCoatDeviceNet, [258](#)
- ~CMeaDeviceNet
  - CMeaDeviceNet, [264](#)
- ~CMealImpedanceDeviceNet
  - CMealImpedanceDeviceNet, [276](#)
- ~CMeaSwitchDeviceNet
  - CMeaSwitchDeviceNet, [279](#)
- ~CMeaUSBDeviceNet
  - CMeaUSBDeviceNet, [281](#)
- ~CMultiBatteryChargerDeviceNet
  - CMultiBatteryChargerDeviceNet, [284](#)
- ~CMultiwellCallbackFunctionNet
  - CMultiwellCallbackFunctionNet, [291](#)
- ~CMultiwellDeviceNet
  - CMultiwellDeviceNet, [295](#)
- ~CMultiwellOptoStimFunctionNet
  - CMultiwellOptoStimFunctionNet, [300](#)
- ~CNF\_GenDeviceNet
  - CNF\_GenDeviceNet, [304](#)
- ~CokuvisionStimulatorDeviceNet
  - CokuvisionStimulatorDeviceNet, [309](#)
- ~CPPCFunctionNet
  - CPPCFunctionNet, [327](#)
- ~CPathIdentDeviceNet
  - CPathIdentDeviceNet, [313](#)
- ~CPedoterDeviceNet
  - CPedoterDeviceNet, [314](#)
- ~CPeristalticPumpDeviceNet
  - CPeristalticPumpDeviceNet, [316](#)
- ~CPgaDeviceNet
  - CPgaDeviceNet, [317](#)
- ~CPositionIIDeviceNet
  - CPositionIIDeviceNet, [319](#)
- ~CPositionImpDeviceNet
  - CPositionImpDeviceNet, [322](#)
- ~CProgramPressureCurveNet
  - CProgramPressureCurveNet, [339](#)
- ~CPulseGeneratorFunctionNet
  - CPulseGeneratorFunctionNet, [341](#)
- ~CRFFFunctionNet
  - CRFFFunctionNet, [349](#)
- ~CRetinaLedDeviceNet
  - CRetinaLedDeviceNet, [346](#)
- ~CRoboDeviceNet
  - CRoboDeviceNet, [372](#)
- ~CRoboFluidDeviceNet
  - CRoboFluidDeviceNet, [384](#)
- ~CSCUFunctionNet
  - CSCUFunctionNet, [401](#)
- ~CSafeISDeviceNet
  - CSafeISDeviceNet, [396](#)
- ~CStg200xBasicNet
  - CStg200xBasicNet, [420](#)
- ~CStg200xDownloadNet
  - CStg200xDownloadNet, [456](#)
- ~CSw2to64DeviceNet
  - CSw2to64DeviceNet, [474](#)
- ~CTEERFunctionNet
  - CTEERFunctionNet, [491](#)
- ~CTEERMachineDeviceNet
  - CTEERMachineDeviceNet, [499](#)
- ~CTcxDeviceNet
  - CTcxDeviceNet, [478](#)
- ~CWarnerUssingDeviceNet
  - CWarnerUssingDeviceNet, [517](#)
- ~CWarnerUssingFunctionNet
  - CWarnerUssingFunctionNet, [520](#)
- ~CWarnerValveControllerDeviceNet
  - CWarnerValveControllerDeviceNet, [538](#)
- ~CWarnerValveControllerDeviceTesterFunctionNet
  - CWarnerValveControllerDeviceTesterFunctionNet, [558](#)
- ~DriverVersionNet
  - DriverVersionNet, [569](#)
- ~SidebandData
  - CStimulusFunctionNet::SidebandData, [593](#)
- ~StimulusDeviceDataAndUnrolledData
  - CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, [595](#)
- A
  - CFilterCoefficientsNet, [67](#)
- A1
  - CFilterCoefficientsNet, [67](#)
- A2
  - CFilterCoefficientsNet, [67](#)
- AdditionalText
  - HeadstageIDTypeObject, [581](#)
- AddLoopEntry
  - CRetinaLedDeviceNet, [346](#)
- AddSelectedChannelsQueue
  - CMcsUsbDacqNet, [165–167](#)
- AddSoftwareKey
  - CMcsUsbNet, [230](#)
- AddTableEntry
  - CRetinaLedDeviceNet, [346](#)
- Altera
  - FirmwareDestinationNames, [574](#)

- amplification
  - CMeaAudioFunctionNet::s\_setaudionet, [592](#)
  - CW2100\_FunctionNet::AudioChannelsNet, [29](#)
- AmplifierSettle
  - CIntanMea\_FunctionNet, [106](#)
- AnalogGain
  - CMeaDeviceNet, [269](#)
- ApplyGains
  - CPgaDeviceNet, [317](#)
- AreTransistorVoltagesSet
  - CCMOSMea\_FunctionNet, [36](#)
- AssociateToThis
  - CMcsUsbNet, [230](#)
- AutomaticAnalogOut
  - CSCUFunctionNet, [401](#)
- Axes\_I
  - CRoboDeviceNet, [379](#)
- Axes\_X
  - CRoboDeviceNet, [379](#)
- Axes\_Y
  - CRoboDeviceNet, [379](#)
- Axes\_Z
  - CRoboDeviceNet, [379](#)
- Axis\_I
  - CRoboDeviceNet, [379](#)
- Axis\_X
  - CRoboDeviceNet, [379](#)
- Axis\_Y
  - CRoboDeviceNet, [379](#)
- Axis\_Z
  - CRoboDeviceNet, [380](#)
- B
  - CFilterCoefficientsNet, [67](#)
- B0
  - CFilterCoefficientsNet, [67](#)
- B1
  - CFilterCoefficientsNet, [67](#)
- B2
  - CFilterCoefficientsNet, [67](#)
- BatteryState, [29](#)
  - Charge, [29](#)
  - ChargeRegionString, [29](#)
  - ChargeString, [29](#)
  - Voltage, [30](#)
  - VoltageString, [30](#)
- BcdDevice
  - DeviceldNet, [567](#)
- BeginImpedanceCheck
  - CIntanMea\_FunctionNet, [106](#)
- BesselFilterHighPassNet, [30](#)
  - BesselFilterHighPassNet, [30](#)
- BesselFilterLowPassNet, [30](#)
  - BesselFilterLowPassNet, [31](#)
- bmRequestType
  - usbSetupPacket\_t, [596](#)
- BOOST\_BIT
  - CW2100\_StimulatorFunctionNet, [515](#)
- Bootstrap
  - FirmwareDestinationNames, [574](#)
- bRequest
  - usbSetupPacket\_t, [596](#)
- BurnAdcOffset
  - COctoPotDeviceNet, [305](#)
- BurnDacOffset
  - CDacCalibrationFunctionNet, [50](#)
  - COctoPotDeviceNet, [306](#)
- BUS1\_MCSBUS1
  - FirmwareDestinationNames, [574](#)
- BUS1\_MCSBUS2
  - FirmwareDestinationNames, [574](#)
- BusType
  - DeviceldNet, [567](#)
- ButterworthFilterHighPassNet, [31](#)
  - ButterworthFilterHighPassNet, [31](#)
- ButterworthFilterLowPassNet, [32](#)
  - ButterworthFilterLowPassNet, [32](#)
- CalibrateThermocouple
  - CFluidControlDeviceNet, [75](#)
  - CTcxDeviceNet, [478](#)
- CancelInternalCalibration
  - CTEERFunctionNet, [491](#)
- CancelPoolLoop
  - CRoboDeviceNet, [373](#)
- CancelPoolLoopAndStopMovement
  - CRoboDeviceNet, [373](#)
- CancelTableLoop
  - CRoboDacqNet, [358](#)
- CancelTableLoopAndStopTable
  - CRoboDacqNet, [358](#)
- CapacityTest
  - CMultiBatteryChargerDeviceNet, [284](#)
- CatchAmpGetAdcMean
  - CMcsBus\_SensorNet, [140](#)
- CatchAmpGetAdcValue
  - CMcsBus\_SensorNet, [140](#)
- CatchAmpGetAdcValueH
  - CMcsBus\_SensorNet, [140](#)
- CatchAmpGetAdcValueL
  - CMcsBus\_SensorNet, [140](#)
- CatchAmpGetDacAmplitude
  - CMcsBus\_SensorNet, [141](#)
- CatchAmpGetDacEnable
  - CMcsBus\_SensorNet, [141](#)
- CatchAmpGetDacOffset
  - CMcsBus\_SensorNet, [141](#)
- CatchAmpGetPwmEnable
  - CMcsBus\_SensorNet, [141](#)
- CatchAmpSetDacAmplitude
  - CMcsBus\_SensorNet, [141](#)
- CatchAmpSetDacEnable
  - CMcsBus\_SensorNet, [141](#)
- CatchAmpSetDacOffset
  - CMcsBus\_SensorNet, [141](#)
- CatchAmpSetPwmEnable
  - CMcsBus\_SensorNet, [141](#)
- CChannelTestDeviceNet, [32](#)

- ~CChannelTestDeviceNet, [33](#)
- CChannelTestDeviceNet, [33](#)
- SetAmplitude, [33](#)
- SetAttenuation, [33](#)
- SetFrequency, [33](#)
- SetWaveform, [33](#)
- CCMOSMea\_FunctionNet, [33](#)
  - AreTransistorVoltagesSet, [36](#)
  - CCMOSMea\_FunctionNet, [35, 36](#)
  - ClearSTGOutput, [36](#)
  - DetectChipType, [36](#)
  - EnableChannelsInGroup, [36](#)
  - GetADCInputOffset, [36](#)
  - GetBath, [36](#)
  - GetBathMode, [36](#)
  - GetEnabledChannelsInGroup, [37](#)
  - GetGate, [37](#)
  - GetGNDI, [37](#)
  - GetGroupADCBits, [37](#)
  - GetGroupChannelBitmaskBySelect, [37](#)
  - GetGroupChannelBitmaskHS1NCBathCurrent, [37, 38](#)
  - GetGroupChannelBitmaskHS1NCCol2Current, [38](#)
  - GetGroupChannelBitmaskHS1NChipTemp, [38](#)
  - GetGroupChannelBitmaskHS1Sidebands, [38](#)
  - GetGroupChannelBitmaskHS1TriggerStatus, [38, 39](#)
  - GetGroupChannelBitmaskIFDigChannels, [39](#)
  - GetGroupChannelBitmaskInterfaceADC, [39](#)
  - GetGroupChannelBitmaskPacketFrameContext, [39](#)
  - GetGroupChannelBitmaskSTG1DACSignal, [39, 40](#)
  - GetGroupDCOffset, [40](#)
  - GetGroupID, [40](#)
  - GetGroupNumberOfChannels, [40](#)
  - GetGroupResolutionPerDigit, [40](#)
  - GetGroupSampleSize, [41](#)
  - GetGroupType, [41](#)
  - GetGroupUnit, [41](#)
  - GetMaxNumOfColumns, [41](#)
  - GetNeurochipMemoryData, [41](#)
  - GetNeurochipMemorySize, [42](#)
  - GetNumberOfSupportedGroups, [42](#)
  - GetSourceBulk, [42](#)
  - GetSourceDrain, [42](#)
  - GetSourceGate, [42](#)
  - GetStimulusSites, [42](#)
  - GetVDD3I, [42](#)
  - GetVDDI, [42](#)
  - IsChipPowered, [42](#)
  - IsGateFloating, [42](#)
  - PowerChip, [43](#)
  - SetADCInputOffset, [43](#)
  - SetBath, [43](#)
  - SetBathMode, [43](#)
  - SetGate, [43](#)
  - SetGateFloating, [43](#)
  - SetGateToVOP, [43](#)
  - SetNeurochipMemoryData, [43](#)
  - SetSourceBulk, [43](#)
  - SetSourceDrain, [44](#)
  - SetSourceGate, [44](#)
  - SetStimulusSites, [44](#)
  - UpdateTransistorVoltages, [44](#)
  - VOPSTimerSetResetTimes, [44](#)
- CCMOSMeaDeviceNet, [44](#)
  - ~CCMOSMeaDeviceNet, [45](#)
  - CCMOSMeaDeviceNet, [45](#)
  - CMosMea, [47](#)
  - GetAvailableBaseSamplerates, [45](#)
  - GetBaseSamplerate, [46](#)
  - GetChannelDataI16, [46](#)
  - GetChannelDataI32, [46](#)
  - GetChannelDataUI16, [46](#)
  - GetChannelDataUI32, [46](#)
  - GetCMOSDataDictionary, [46](#)
  - SetBaseSamplerate, [46](#)
  - SetRegionOfInterests, [46](#)
  - Stimulus, [47](#)
  - UpdateChannelBlock, [47](#)
- CCMOSMeaDeviceNet::CRegionOfInterestRect, [345](#)
  - CRegionOfInterestRect, [345](#)
  - DeepCopy, [345](#)
  - m\_Bottom, [345](#)
  - m\_Left, [345](#)
  - m\_Right, [345](#)
  - m\_Top, [345](#)
- CCreateFilterNet, [47](#)
  - ~CCreateFilterNet, [48](#)
  - CCreateFilterNet, [48](#)
  - CutoffFrequency, [49](#)
  - FindFilter, [48](#)
  - GetBiQuad, [48](#)
  - GetBiQuads, [48](#)
  - NumCoefSets, [49](#)
  - Order, [49](#)
  - SampleRate, [49](#)
  - Scale, [49](#)
- CDacCalibrationFunctionNet, [49](#)
  - !CDacCalibrationFunctionNet, [50](#)
  - ~CDacCalibrationFunctionNet, [50](#)
  - BurnDacOffset, [50](#)
  - CDacCalibrationFunctionNet, [50](#)
  - GetDacOffset, [50](#)
  - SetDacOffset, [51](#)
- CDacqGroupChannelGenericSelectionNet, [51](#)
  - CDacqGroupChannelGenericSelectionNet, [51](#)
- CDacqGroupChannelSelectionNet, [52](#)
  - CDacqGroupChannelSelectionNet, [52](#)
- CDacqGroupChannelSelectionTemplateNet
  - CDacqGroupChannelSelectionTemplateNet< Dac-qGroupChannelEnumTemplateNet, Dac-qGroupChannelEnumTemplate, CDevice-GroupChannelInfoTemplateNet >, [53](#)
- CDacqGroupChannelSelectionTemplateNet< Dac-qGroupChannelEnumTemplateNet, Dac-qGroupChannelEnumTemplate, CDevice-

- GroupChannelInfoTemplateNet >, 52
- CDacqGroupChannelSelectionTemplateNet, 53
- EnableChannelsInGroup, 53
- GetDeviceGroupChannelInfos, 53
- GetEnabledChannelsInGroup, 53, 54
- GetGroupID, 54
- GetGroupNumberOfChannels, 54
- GetGroupSampleSize, 54
- GetGroupType, 54
- GetNumberOfSupportedGroups, 55
- CDeviceGroupChannelInfoGenericNet, 55
  - CDeviceGroupChannelInfoGenericNet, 55
- CDeviceGroupChannelInfoNet, 55
  - CDeviceGroupChannelInfoNet, 56
- CDeviceGroupChannelInfoSCUNet, 56
  - CDeviceGroupChannelInfoSCUNet, 56
- CDeviceGroupChannelInfoTemplateNet
  - CDeviceGroupChannelInfoTemplateNet< Dacq-GroupChannelEnumTemplateNet >, 57
- CDeviceGroupChannelInfoTemplateNet< Dacq-GroupChannelEnumTemplateNet >, 57
  - CDeviceGroupChannelInfoTemplateNet, 57
  - GroupID, 57
  - GroupType, 57
  - NumberOfChannels, 57
- CDeviceGroupChannelInfoW2100Net, 57
  - CDeviceGroupChannelInfoW2100Net, 58
- CDigOutStimulatorFunctionNet, 58
  - !CDigOutStimulatorFunctionNet, 59
  - ~CDigOutStimulatorFunctionNet, 59
  - CDigOutStimulatorFunctionNet, 59
  - ClearChannel, 59
  - GetGlobalRepeat, 59
  - GetNumberOfChannels, 60
  - GetStartTriggerSlope, 60
  - GetStopTriggerSlope, 60
  - PrepareChannelData, 61
  - SendChannelData, 61
  - SetGlobalRepeat, 61
  - SetStartTriggerSlope, 61
  - SetStopTriggerSlope, 62
- CEncapsulatorDeviceNet, 62
  - CEncapsulatorDeviceNet, 63
  - GetRoboFluidDevice, 63
- CExternDTesterDeviceNet, 63
  - !CExternDTesterDeviceNet, 64
  - ~CExternDTesterDeviceNet, 64
  - CExternDTesterDeviceNet, 64
  - Read, 64
  - Read2, 64
  - Write, 64
  - Write2, 65
- CFilterCoefficientsNet, 65
  - ~CFilterCoefficientsNet, 66
  - A, 67
  - A1, 67
  - A2, 67
  - B, 67
  - B0, 67
  - B1, 67
  - B2, 67
  - CFilterCoefficientsNet, 65, 66
  - Equals, 66
  - SetAFormat, 66
  - SetBFormat, 67
  - UIntA1, 67
  - UIntA2, 68
  - UIntB0, 68
  - UIntB1, 68
  - UIntB2, 68
- CFilterConfigurationNet, 68
  - CFilterConfigurationNet, 68
  - EraseFilterParameterPermanent, 69
  - GetHighpassFilterEnable, 69
  - ResetHighpassFilter, 69
  - SetFilterParameter, 69
  - SetFilterParameterPermanent, 69
  - SetHighpassFilterEnable, 69
- CFilterConfigurationRegisterNet, 70
  - CFilterConfigurationRegisterNet, 70
  - EraseFilterParameterPermanent, 70
  - SetFilterParameter, 71
  - SetFilterParameterPermanent, 71
- CFilterPropertyNet, 71
  - ~CFilterPropertyNet, 72
  - CFilterPropertyNet, 72
  - CornerFrequency, 72
  - CornerFrequencycymHz, 72
  - FilterActive, 72
  - FilterBand, 73
  - FilterFamily, 73
  - FilterType, 73
  - Order, 73
  - ToString, 72
- CFluidControlDeviceNet, 73
  - ~CFluidControlDeviceNet, 74
  - CalibrateThermocouple, 75
  - CFluidControlDeviceNet, 74
  - GetAdc, 76
  - GetDigin, 76
  - GetDigout, 76
  - GetPWM, 76
  - GetReferenceTemperature, 76
  - GetSingleValve, 77
  - GetThermocoupleCalibration, 77
  - GetThermocoupleNanovoltPerKelvin, 77
  - GetThermocoupleTemperature, 78
  - GetValve, 78
  - McsBus\_VoltageMode, 80
  - SetDigout, 78
  - SetPWM, 78
  - SetSingleValve, 79
  - SetThermocoupleNanovoltPerKelvin, 79
  - SetValve, 79
- CFYIDeviceNet, 80
  - CFYIDeviceNet, 80

- FYIProgram, [81](#)
- FYITemp, [81](#)
- Sensor, [81](#)
- CGenericDevelopDeviceNet, [81](#)
  - ~CGenericDevelopDeviceNet, [88](#)
  - CGenericDevelopDeviceNet, [88](#)
  - ClosePipe, [88](#)
  - GetBuffer, [88](#)
  - GetByteBuffer, [89](#)
  - GetIntBuffer, [89](#)
  - GetShortBuffer, [90](#)
  - GetUByteBuffer, [91](#)
  - GetUIntBuffer, [91](#)
  - GetUShortBuffer, [92](#)
  - OpenPipe, [93](#)
  - ReadPipe, [93](#)
  - ResetPipe, [93](#)
  - SetBuffer, [94](#)
  - SetByteBuffer, [94](#)
  - SetIntBuffer, [94](#)
  - SetShortBuffer, [95](#)
  - SetUByteBuffer, [96](#)
  - SetUIntBuffer, [96](#)
  - SetUShortBuffer, [97](#)
  - SetValue, [97](#)
  - WritePipe, [98](#)
- CGilsonDeviceNet, [98](#)
  - ~CGilsonDeviceNet, [99](#)
  - CGilsonDeviceNet, [99](#)
  - ConnectSlave, [99](#)
  - GetLastAnswer, [100](#)
  - m\_pGilsonDevice, [100](#)
  - SendBuffered, [100](#)
  - SendImmediate, [100](#)
  - SendImmediateGetResponse, [100](#)
- ChangeSerialNumber
  - CMcsUsbFactoryNet, [211](#)
- channel
  - CMeaAudioFunctionNet::s\_setaudionet, [592](#)
  - CW2100\_FunctionNet::AudioChannelsNet, [29](#)
- ChannelBlock\_AvailFrames
  - CMcsUsbDacqNet, [168](#)
- ChannelBlock\_ReadAsFrameArrayI16
  - CMcsUsbDacqNet, [168](#), [169](#)
- ChannelBlock\_ReadAsFrameArrayI32
  - CMcsUsbDacqNet, [170](#)
- ChannelBlock\_ReadAsFrameArrayUI16
  - CMcsUsbDacqNet, [171](#), [172](#)
- ChannelBlock\_ReadAsFrameArrayUI32
  - CMcsUsbDacqNet, [172](#), [173](#)
- ChannelBlock\_ReadFramesDictI16
  - CMcsUsbDacqNet, [174](#)
- ChannelBlock\_ReadFramesDictI32
  - CMcsUsbDacqNet, [174](#)
- ChannelBlock\_ReadFramesDictUI16
  - CMcsUsbDacqNet, [175](#)
- ChannelBlock\_ReadFramesDictUI32
  - CMcsUsbDacqNet, [175](#)
- ChannelBlock\_ReadFramesI16
  - CMcsUsbDacqNet, [176](#), [177](#)
- ChannelBlock\_ReadFramesI32
  - CMcsUsbDacqNet, [177](#), [178](#)
- ChannelBlock\_ReadFramesUI16
  - CMcsUsbDacqNet, [179](#)
- ChannelBlock\_ReadFramesUI32
  - CMcsUsbDacqNet, [180](#), [181](#)
- ChannelDataEvent
  - CMcsUsbDacqNet, [207](#)
- ChannelReset
  - CMultiBatteryChargerDeviceNet, [285](#)
- Charge
  - BatteryState, [29](#)
- ChargeRegionString
  - BatteryState, [29](#)
- ChargeString
  - BatteryState, [29](#)
- CHiClampDeviceNet, [100](#)
  - CHiClampDeviceNet, [101](#)
  - RoboDacq, [101](#)
- CHLADacqNet, [101](#)
  - CHLADacqNet, [102](#)
- CHLADeviceNet, [102](#)
  - CHLADeviceNet, [102](#)
  - HLADacq, [103](#)
  - SerialPort, [103](#)
- CHWInfo
  - CMcsUsbDacqNet::CHWInfo, [103](#)
- CIntanMea\_FunctionNet, [105](#)
  - AmplifierSettle, [106](#)
  - BeginImpedanceCheck, [106](#)
  - CIntanMea\_FunctionNet, [106](#)
  - GetDSPHighPassByIndex, [106](#)
  - GetImpedanceResult, [106](#)
  - GetIntanRegister, [106](#)
  - GetLowerFrequencyByIndex, [106](#)
  - GetUpperFrequencyByIndex, [107](#)
  - SetBandwidthByIndex, [107](#)
  - SetDiagnosticMode, [107](#)
  - SetDSPHighPassByIndex, [107](#)
  - SetIntanRegister, [107](#)
- CInterfaceboardFunctionNet, [107](#)
  - !CInterfaceboardFunctionNet, [108](#)
  - ~CInterfaceboardFunctionNet, [108](#)
  - CInterfaceboardFunctionNet, [108](#)
  - GetCardinalDacqSamplerate, [109](#)
  - GetCardinalStgOutputrate, [109](#)
  - SetCardinalDacqSamplerate, [109](#)
  - SetCardinalStgOutputrate, [109](#)
- ClampAmpRestart
  - CRoboDacqNet, [359](#)
- ClearBuffers
  - CMcsUsbDacqNet, [181](#)
- ClearChannel
  - CDigOutStimulatorFunctionNet, [59](#)
- ClearChannel\_PrepareAndSendData
  - CStg200xDownloadNet, [457](#)



- CStimulusFunctionNet, 464
- ClearChannelData
  - CStg200xDownloadBasicNet, 448
  - CStimulusFunctionNet, 464
  - CW2100\_StimulatorFunctionNet, 510
- ClearMultiplexedData
  - CStimulusFunctionNet, 464
- ClearSTGOutput
  - CCMOSMea\_FunctionNet, 36
- ClearStimulusParametersCache
  - CW2100\_FunctionNet, 503
- ClearSyncData
  - CStg200xDownloadBasicNet, 448
  - CStimulusFunctionNet, 464
- ClearTable
  - CRetinalLedDeviceNet, 347
- ClearTableName
  - CWarnerValveControllerDeviceNet, 538
- ClearUserDefinedNameCache
  - CW2100\_FunctionNet, 503
- ClearValveTable
  - CWarnerValveControllerDeviceNet, 538
- CLIH3DeviceNet, 109
  - !CLIH3DeviceNet, 111
  - ~CLIH3DeviceNet, 111
  - CLIH3DeviceNet, 111
  - DummyCommand, 111
  - EnableUserTrigger, 112
  - GetADCOffset, 112
  - GetDacIdleValue, 112
  - GetDacqRunStatus, 112
  - GetDacUseIdleValue, 113
  - GetDigInState, 113
  - GetEEPROMPage, 113
  - GetSampleInterval, 114
  - IsUserTriggerEnabled, 114
  - ReadClipping, 114
  - SendCommand, 114
  - SetADCOffset, 115
  - SetDacIdleValue, 115
  - SetDacUseIdleValue, 115
  - SetDigOutState, 115
  - SetEEPROMPage, 116
  - SetSampleInterval, 116
  - StimulusFunction, 116
- CloseAllValves
  - CRoboFluidDeviceNet, 384
- ClosePipe
  - CGenericDevelopDeviceNet, 88
- ClosePlateClamp
  - CMultiwellDeviceNet, 295
- CMcsBus\_AxisParametersNet, 116
  - ~CMcsBus\_AxisParametersNet, 117
  - CMcsBus\_AxisParametersNet, 117
  - GetAxisParametersSignedEeprom, 117
  - GetAxisParametersUnsignedEeprom, 117
  - SetAxisParametersEeprom, 117, 118
- CMcsBus\_ExtensionNet, 118
  - ~CMcsBus\_ExtensionNet, 118
  - CMcsBus\_ExtensionNet, 118
  - GetLEDSwitch, 119
  - SetLEDSwitch, 119
- CMcsBus\_FYIExtensionNet, 119
  - ~CMcsBus\_FYIExtensionNet, 119
  - CMcsBus\_FYIExtensionNet, 119
  - GetDIO, 120
  - GetSingleHeater, 120
  - GetValves, 120
  - SetDIO, 120
  - SetSingleHeater, 120
  - SetValves, 120
- CMcsBus\_MotorControlNet, 121
  - ~CMcsBus\_MotorControlNet, 124
  - CMcsBus\_MotorControlNet, 124
  - GetMCAcceleration, 124
  - GetMCAccelerationEeprom, 124
  - GetMCAccelerationShortCommand, 124
  - GetMCRevisionEeprom, 124
  - GetMCBreakCurrent, 124
  - GetMCBreakCurrentEeprom, 125
  - GetMCConfig, 125
  - GetMCConfigEeprom, 125
  - GetMCCurrent, 125
  - GetMCCurrentEeprom, 125
  - GetMCCurrentMode, 125
  - GetMCCurrentModeEeprom, 125
  - GetMCCurrentModeShortCommand, 126
  - GetMCCurrentPosition, 126
  - GetMCCurrentShortCommand, 126
  - GetMCCurrentSpeed, 126
  - GetMCMaxAcceleration, 126
  - GetMCMaxAccelerationEeprom, 126
  - GetMCMaxCurrent, 126
  - GetMCMaxCurrentEeprom, 127
  - GetMCMaxSpeed, 127
  - GetMCMaxSpeedEeprom, 127
  - GetMCMaxTravel, 127
  - GetMCMaxTravelEeprom, 127
  - GetMCMaxTravelShortCommand, 127
  - GetMCMovement, 127
  - GetMCNewPosition, 128
  - GetMCOutputOnOff, 128
  - GetMCPhase, 128
  - GetMCPhaseOffset, 128
  - GetMCReference, 128
  - GetMCReferenceCurrent, 128
  - GetMCReferenceCurrentEeprom, 128
  - GetMCRegulatorGain, 129
  - GetMCRegulatorGainEeprom, 129
  - GetMCScalingFactor, 129
  - GetMCScalingFactorEeprom, 129
  - GetMCSpeed, 129
  - GetMCSpeedEeprom, 129
  - GetMCSpeedShortCommand, 129
  - GetMCSpeedUnitEeprom, 130
  - GetMCStandbyCurrent, 130

- GetMCStandbyCurrentEeprom, 130
- GetMCStandbyTime, 130
- GetMCStandbyTimeEeprom, 130
- GetSubChannel, 130
- SetMCAcceleration, 130
- SetMCAccelerationEeprom, 131
- SetMCAccelerationShortCommand, 131
- SetMCAxisRevisionEeprom, 131
- SetMCBreakCurrent, 131
- SetMCBreakCurrentEeprom, 131
- SetMCConfig, 131
- SetMCConfigEeprom, 132
- SetMCCurrent, 132
- SetMCCurrentEeprom, 132
- SetMCCurrentMode, 132
- SetMCCurrentModeEeprom, 132
- SetMCCurrentModeShortCommand, 132
- SetMCCurrentPosition, 133
- SetMCCurrentShortCommand, 133
- SetMCMaxAcceleration, 133
- SetMCMaxAccelerationEeprom, 133
- SetMCMaxCurrent, 133
- SetMCMaxCurrentEeprom, 133
- SetMCMaxSpeed, 134
- SetMCMaxSpeedEeprom, 134
- SetMCMaxTravel, 134
- SetMCMaxTravelEeprom, 134
- SetMCMaxTravelShortCommand, 134
- SetMCNewPosition, 134
- SetMCOutputOnOff, 135
- SetMCReference, 135
- SetMCReferenceCurrent, 135
- SetMCReferenceCurrentEeprom, 135
- SetMCRegulatorGain, 135
- SetMCRegulatorGainEeprom, 135
- SetMCRotation, 136
- SetMCScalingFactor, 136
- SetMCScalingFactorEeprom, 136
- SetMCSpeed, 136
- SetMCSpeedEeprom, 136
- SetMCSpeedShortCommand, 136
- SetMCSpeedUnitEeprom, 137
- SetMCStandbyCurrent, 137
- SetMCStandbyCurrentEeprom, 137
- SetMCStandbyTime, 137
- SetMCStandbyTimeEeprom, 137
- SetSubChannel, 137
- StartMCMovement, 138
- StopMCMovement, 138
- CMcsBus\_SensorNet, 138
  - ~CMcsBus\_SensorNet, 140
  - CatchAmpGetAdcMean, 140
  - CatchAmpGetAdcValue, 140
  - CatchAmpGetAdcValueH, 140
  - CatchAmpGetAdcValueL, 140
  - CatchAmpGetDacAmplitude, 141
  - CatchAmpGetDacEnable, 141
  - CatchAmpGetDacOffset, 141
  - CatchAmpGetPwmEnable, 141
  - CatchAmpSetDacAmplitude, 141
  - CatchAmpSetDacEnable, 141
  - CatchAmpSetDacOffset, 141
  - CatchAmpSetPwmEnable, 141
  - CMcsBus\_SensorNet, 140
  - Get2AnalogInput, 142
  - Get2DigitalInput, 142
  - Get4ADC, 142
  - Get4ADCAverage, 142
  - Get4ADCCatchampAverageShift, 142
  - Get4ADCMode, 142
  - Get4DAC, 142
  - GetADCs, 142
  - GetADCsLoop, 143
  - GetBubbleStatus, 143
  - GetDACs, 143
  - GetDetectionThreshold, 143
  - GetDetectorValue, 143
  - GetLatency, 143
  - GetLatencyCounter, 143
  - GetMinimalThreshold, 143
  - GetMovePump, 144
  - GetPiezoState, 144
  - GetPressure, 144
  - GetPressureOffset, 144
  - GetRegulationTimeouts, 144
  - GetRegulatorFactor, 145
  - GetRegulatorOnOff, 145
  - GetRegulatorStatus, 145
  - GetRotatePump, 145
  - GetSamplePeriode, 145
  - GetSollPressure, 145
  - GetSyncState, 145
  - Set4ADCCatchampAverageShift, 146
  - Set4ADCMode, 146
  - Set4DAC, 146
  - SetDACs, 146
  - SetDetectionThreshold, 146
  - SetLatency, 146
  - SetMinimalThreshold, 146
  - SetMovePump, 147
  - SetPiezoState, 147
  - SetPressureOffset, 147
  - SetRegulationTimeouts, 147
  - SetRegulatorFactor, 147
  - SetRegulatorOnOff, 147
  - SetRotatePump, 147
  - SetSamplePeriode, 148
  - SetSollPressure, 148
  - StartSync, 148
  - TactSwitchGetState, 148
  - TactSwitchSetDisplay, 148
- CMcsBus\_TempSensorNet, 148
  - ~CMcsBus\_TempSensorNet, 149
  - CMcsBus\_TempSensorNet, 149
  - GetNanoVoltsPerKelvin, 149
  - GetTemperatur, 149

- GetThermoOffset, 149
- GetThermoTemp, 150
- GetThermoVoltage, 150
- SetNanoVoltsPerKelvin, 150
- SetThermoOffset, 150
- CMcsBus\_VoltageModeNet, 150
  - ~CMcsBus\_VoltageModeNet, 151
- CMcsBus\_VoltageModeNet, 151
- GetVMMMaxNegativeCurrent, 152
- GetVMMMaxNegativeCurrentEeprom, 152
- GetVMMMaxNegativeVoltage, 152
- GetVMMMaxNegativeVoltageEeprom, 152
- GetVMMMaxPositiveCurrent, 152
- GetVMMMaxPositiveCurrentEeprom, 152
- GetVMMMaxPositiveVoltage, 152
- GetVMMMaxPositiveVoltageEeprom, 153
- GetVMOOutputOnOff, 153
- GetVMVoltage, 153
- SetVMMMaxNegativeCurrent, 153
- SetVMMMaxNegativeCurrentEeprom, 153
- SetVMMMaxNegativeVoltage, 153
- SetVMMMaxNegativeVoltageEeprom, 153
- SetVMMMaxPositiveCurrent, 154
- SetVMMMaxPositiveCurrentEeprom, 154
- SetVMMMaxPositiveVoltage, 154
- SetVMMMaxPositiveVoltageEeprom, 154
- SetVMOOutputOnOff, 154
- SetVMVoltage, 154
- CMcsBusNet, 155
  - ~CMcsBusNet, 155
- CMcsBusNet, 155
- CMcsBusNet::GetMode, 156
- CMcsBusNet::GetModeEeprom, 156
- CMcsBusNet::SetMode, 156
- CMcsBusNet::SetModeEeprom, 156
- GetBusAddress, 156
- GetBusAddressEeprom, 156
- GetCommand, 156, 157
- GetHWRevisionEeprom, 157
- SetBusAddress, 157
- SetBusAddressEeprom, 157
- SetCommand, 158
- SetHWRevisionEeprom, 158
- CMcsBusNet::GetMode
  - CMcsBusNet, 156
- CMcsBusNet::GetModeEeprom
  - CMcsBusNet, 156
- CMcsBusNet::SetMode
  - CMcsBusNet, 156
- CMcsBusNet::SetModeEeprom
  - CMcsBusNet, 156
- CMcsUsbDacqNet, 159
  - ~CMcsUsbDacqNet, 165
  - AddSelectedChannelsQueue, 165–167
  - ChannelBlock\_AvailFrames, 168
  - ChannelBlock\_ReadAsFrameArrayI16, 168, 169
  - ChannelBlock\_ReadAsFrameArrayI32, 170
  - ChannelBlock\_ReadAsFrameArrayUI16, 171, 172
  - ChannelBlock\_ReadAsFrameArrayUI32, 172, 173
  - ChannelBlock\_ReadFramesDictI16, 174
  - ChannelBlock\_ReadFramesDictI32, 174
  - ChannelBlock\_ReadFramesDictUI16, 175
  - ChannelBlock\_ReadFramesDictUI32, 175
  - ChannelBlock\_ReadFramesI16, 176, 177
  - ChannelBlock\_ReadFramesI32, 177, 178
  - ChannelBlock\_ReadFramesUI16, 179
  - ChannelBlock\_ReadFramesUI32, 180, 181
  - ChannelDataEvent, 207
  - ClearBuffers, 181
  - CMcsUsbDacqNet, 165
  - CMcsUsbDacqNet::GetFilterProperties, 181
  - Error\_Callback\_Aquisition\_Stopped, 206
  - Error\_Callback\_Data\_lost, 206
  - Error\_Callback\_Frames\_Lost, 206
  - Error\_Callback\_Packet\_Error, 206
  - Error\_Callback\_Queue\_Full, 206
  - Error\_Callback\_RingQueue\_Full, 206
  - ErrorEvent, 207
  - GetAdapterType, 182
  - GetAdcDataFormat, 182
  - GetAdcZero, 182
  - GetAnalogValueUnit, 182
  - GetChannelDataFillSize, 182
  - GetChannelLayout, 182
  - GetChannelsInBlock, 183
  - GetDataFormat, 183
  - GetDataMode, 183
  - GetDigitalSource, 183–185
  - GetFilterProperty, 185
  - GetGroupChannelDataI16, 185
  - GetGroupChannelDataI32, 186
  - GetGroupChannelDataUI16, 186
  - GetGroupChannelDataUI32, 187
  - GetHardwareMaxRange, 188
  - GetHardwareMinRange, 188
  - GetMaxSamplingFrequency, 188
  - GetMeaLayout, 188
  - GetMinSamplingFrequencyStepsize, 188
  - GetNumberOfDataBits, 189
  - GetPoti, 189
  - GetResolutionPerDigit, 189
  - GetSamplerate, 189
  - GetVoltageRangeIndex, 189
  - GetVoltageRangeInMicroVolt, 189
  - GetVoltageRangeInMilliVolt, 190
  - HWInfo, 190
  - Samplerate, 206
  - SendStartDacq, 190
  - SendStartStgAndDacq, 190
  - SendStopDacq, 191
  - SendStopStgAndDacq, 191
  - SendStopStgAndDacqWithOptions, 191
  - SetDataMode, 192
  - SetDigitalSource, 192–194
  - SetPoti, 194
  - SetSamplerate, 194

- SetSelectedChannels, [194–196](#)
- SetSelectedChannelsQueue, [197–199](#)
- SetSelectedData, [199–201](#)
- SetupGroupDacqQueue, [201](#)
- SetVoltageRangeByIndex, [201](#)
- SetVoltageRangeInMicroVolt, [201](#)
- StartDacq, [201–203](#)
- StartLoop, [203, 204](#)
- StopDacq, [205](#)
- StopLoop, [206](#)
- CMcsUsbDacqNet::CHWInfo, [103](#)
  - CHWInfo, [103](#)
  - GetAvailableSampleRates, [104](#)
  - GetAvailableVoltageRangesInMicroVolt, [104](#)
  - GetAvailableVoltageRangesInMicroVoltAndStringsInMilliVolt, [104](#)
  - GetNumberOfHWADCCChannels, [104](#)
  - GetNumberOfHWDigitalChannels, [104](#)
  - IsDigitalChannelDedicated, [105](#)
- CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet, [501](#)
  - CVoltageRangeInfoNet, [501](#)
  - VoltageRangeDisplayStringMilliVolt, [501](#)
  - VoltageRangeInMicroVolt, [501](#)
- CMcsUsbDacqNet::GetFilterProperties
  - CMcsUsbDacqNet, [181](#)
- CMcsUsbDeviceStatePushFunctionNet, [207](#)
  - CMcsUsbDeviceStatePushFunctionNet, [207](#)
  - McsUsbDeviceStateEvent, [208](#)
  - TriggerStatus, [208](#)
- CMcsUsbDeviceStatePushNet, [208](#)
  - CMcsUsbDeviceStatePushNet, [208](#)
  - McsUsbDeviceStateEvent, [209](#)
  - TriggerStatus, [209](#)
- CMcsUsbFactoryNet, [209](#)
  - ~CMcsUsbFactoryNet, [211](#)
  - ChangeSerialNumber, [211](#)
  - CMcsUsbFactoryNet, [211](#)
  - Coldstart, [211](#)
  - CompareFirmware, [211](#)
  - DownloadFirmware, [211](#)
  - FindFirmwareVersionMagicInBuffer, [211](#)
  - FX3MCSDDataAddress, [217](#)
  - FX3MCSDDataDeviceIdOffset, [217](#)
  - FX3MCSDDataVersionOffset, [217](#)
  - GetChecksumFromFX3Image, [212](#)
  - GetDestination, [212](#)
  - GetDestinationDisplayLabel, [212](#)
  - GetDestinationName, [212](#)
  - GetDestinationSerialNumber, [212](#)
  - GetDestinationTargetAddress, [212](#)
  - GetFirmwareVersionFromFile, [213](#)
  - GetFirmwareVersionFromHexFile, [213](#)
  - GetNumDestinations, [213](#)
  - GetUSBDeviceIDFromFX3Image, [213](#)
  - GetUserCodeFromBitFile, [213](#)
  - GetUserCodeFromFlash, [213](#)
  - GetXilinxFlashOffset, [214](#)
  - GetXilinxFlashReadCommand, [214](#)
  - LoadUserFirmware, [214](#)
  - ReadBlockFromFlash, [214](#)
  - ReadBlockFromIFBGlobalEEPROM, [215](#)
  - ReadBlockFromNVMEM, [215](#)
  - SetDestinationSerialNumber, [215](#)
  - UpdateFirmware, [215, 216](#)
- CMcsUsbFunctionNet, [217](#)
  - !CMcsUsbFunctionNet, [218](#)
  - ~CMcsUsbFunctionNet, [218](#)
  - CMcsUsbFunctionNet, [218](#)
  - m\_pMcsUsb, [219](#)
  - m\_pMcsUsbFunction, [219](#)
  - ThrowCUsbExceptionNetOnError, [218](#)
- CMcsUsbFunctionPointerContainer, [219](#)
- CMcsUsbListEntryNet, [219](#)
  - ~CMcsUsbListEntryNet, [220](#)
  - DeviceId, [222](#)
  - DeviceName, [222](#)
  - Equals, [220](#)
  - GetEntry, [220, 221](#)
  - GetEntryCount, [221](#)
  - HwVersion, [222](#)
  - Manufacturer, [222](#)
  - Product, [223](#)
  - SerialNumber, [223](#)
  - SetStringFormat, [222](#)
  - ToString, [222](#)
- CMcsUsbListNet, [223](#)
  - !CMcsUsbListNet, [224](#)
  - ~CMcsUsbListNet, [224](#)
  - CMcsUsbListNet, [224](#)
  - Count, [225](#)
  - DeviceArrival, [225](#)
  - DeviceRemoval, [225](#)
  - GetNumberOfDevices, [224](#)
  - GetUsbListEntries, [224](#)
  - GetUsbListEntry, [224](#)
  - IsDeviceTypeOf, [225](#)
  - SetStringFormat, [225](#)
- CMcsUsbNet, [226](#)
  - !CMcsUsbNet, [230](#)
  - ~CMcsUsbNet, [230](#)
  - AddSoftwareKey, [230](#)
  - AssociateToThis, [230](#)
  - CMcsUsbNet, [230](#)
  - Connect, [230, 231](#)
  - Disconnect, [232](#)
  - EmptyKey, [232](#)
  - EnableExceptions, [232](#)
  - EraseEEPROMRegisterPreconfig, [232](#)
  - GetConfiguration, [233](#)
  - GetDeviceCannotStallOutRequests, [233](#)
  - GetDeviceCapableSpeed, [233](#)
  - GetDeviceEnum, [233](#)
  - GetDeviceId, [233](#)
  - GetDeviceRootHubVendorEnum, [233](#)
  - GetDeviceRootHubVendorID, [233](#)

- GetDeviceRootHubVendorName, 233
- GetDeviceSpeed, 234
- GetErrorText, 234
- GetFirmwareVersion, 234
- GetHardwareRevision, 234
- GetHeadstageActive, 235
- GetHeadstageID, 235
- GetHeadstagePresent, 235
- GetIdent, 235
- GetLastUSBError, 236
- GetMea21UsbPort, 236
- GetNumConfigurations, 236
- GetSerialNumber, 236
- GetSoftwareKey, 236
- GetSoftwareKeyString, 236
- GetStatus, 236
- GetStatusOfLastCommand, 237
- GetUsbListEntry, 237
- GetVersion, 237
- HasSoftwareKey, 237
- IsConnected, 237
- IsDeviceHighSpeed, 237
- IsDeviceHighSpeedCapable, 238
- IsExceptionsEnabled, 238
- MultibootGetCypressImageld, 238
- MultibootGetImageld, 238
- MultibootGetSelectedImage, 238
- MultibootSelectImage, 238
- ReadEepromRegisterPreconfig, 239
- ReadRegister, 239
- ReadRegister32, 239
- ReadRegisterTimeSlot, 239
- RemoveSoftwareKey, 240
- RescanHeadstage, 240
- SerialNumber, 247
- SetConfiguration, 240
- SetSoftwareKey, 240
- Status\_AlreadyConfigured, 242
- Status\_BadStartFrame, 243
- Status\_Btstuff, 243
- Status\_BufferOverrun, 243
- Status\_BufferUnderrun, 243
- Status\_Canceled, 243
- Status\_Canceling, 243
- Status\_ConnectedPipes, 243
- Status\_ControlNotOwned, 243
- Status\_Crc, 243
- Status\_DataOverrun, 243
- Status\_DataToggleMismatch, 243
- Status\_DataUnderrun, 244
- Status\_DeviceLocked, 244
- Status\_DeviceNotFound, 244
- Status\_DeviceRemoved, 244
- Status\_DevNotResponding, 244
- Status\_EndpointHalted, 244
- Status\_ErrorBusy, 244
- Status\_ErrorShortTransfer, 244
- Status\_Fifo, 244
- Status\_FrameControlOwned, 244
- Status\_InternalHcError, 244
- Status\_InvalidParameter, 245
- Status\_InvalidPipeHandle, 245
- Status\_InvalidUrbFunction, 245
- Status\_IoPending, 245
- Status\_IoTimeout, 245
- Status\_IsochRequestFailed, 245
- Status\_LastUsbErrorMismatch, 245
- Status\_NoBandwidth, 245
- Status\_NoMemory, 245
- Status\_NoSuchDevice, 245
- Status\_NotAccessed, 245
- Status\_NotSupported, 246
- Status\_PidCheckFailure, 246
- Status\_PipeNotLinked, 246
- Status\_RequestFailed, 246
- Status\_RequestMutexFailed, 246
- Status\_RequestMutexTimeout, 246
- Status\_Stall, 246
- Status\_Unconfigured, 246
- Status\_UnexpectedPid, 246
- ThrowCUsbExceptionNetOnError, 240
- TxnGetSerialNumber, 240
- TxnSetSerialNumber, 240
- TxnTestMemoryReadAndCheck, 240
- TxnTestMemoryWrite, 241
- ValidKey, 241
- WPAError\_ScanningIsPending, 246
- WriteEepromRegisterPreconfig, 241
- WriteRegister, 241, 242
- WriteRegister32, 242
- WriteRegisterArray, 242
- WriteRegisterTimeSlot, 242
- WriteRegisterValue, 242
- CMcsUsbPointerContainer, 247
- CMEA2100x256FunctionNet, 247
  - !CMEA2100x256FunctionNet, 248
  - ~CMEA2100x256FunctionNet, 248
  - CMEA2100x256FunctionNet, 247, 248
  - GetLayoutConfiguration, 248
  - SetLayoutConfiguration, 248
- CMeaAudioFunctionNet, 249
  - CMeaAudioFunctionNet, 249
  - GetAudioChannels, 250, 251
  - GetNumberOfAudioChannels, 251
  - SetAudioChannels, 251, 252
- CMeaAudioFunctionNet::s\_setaudionet, 592
  - amplification, 592
  - channel, 592
- CMeaCleanDeviceNet, 252
  - !CMeaCleanDeviceNet, 254
  - ~CMeaCleanDeviceNet, 253
  - CMeaCleanDeviceNet, 253
  - GetCycle, 254
  - GetCycles, 254
  - GetMaxVoltage, 254
  - GetMinVoltage, 254

- GetOutputVoltage, [254](#)
- GetSlope, [255](#)
- IsRunning, [255](#)
- SetCycles, [255](#)
- SetMaxVoltage, [255](#)
- SetMinVoltage, [256](#)
- SetSlope, [256](#)
- Start, [256](#)
- Stop, [256](#)
- CMeaCoatDeviceNet, [256](#)
  - !CMeaCoatDeviceNet, [258](#)
  - ~CMeaCoatDeviceNet, [258](#)
  - CMeaCoatDeviceNet, [258](#)
  - GetCurrentCycle, [258](#)
  - GetCycles, [258](#)
  - GetDuration, [258](#)
  - GetMaxCurrent, [258](#)
  - GetOffsetCurrent, [259](#)
  - GetOutputCurrent, [259](#)
  - GetPauseDuration, [259](#)
  - GetSlope, [259](#)
  - GetTimeInPause, [259](#)
  - GetTimeInPlateau, [260](#)
  - IsRunning, [260](#)
  - SetCycles, [260](#)
  - SetDuration, [260](#)
  - SetMaxCurrent, [261](#)
  - SetOffsetCurrent, [261](#)
  - SetPauseDuration, [261](#)
  - SetSlope, [261](#)
  - Start, [261](#)
  - Stop, [262](#)
- CMeaDeviceNet, [262](#)
  - ~CMeaDeviceNet, [264](#)
  - AnalogGain, [269](#)
  - CMeaDeviceNet, [263](#)
  - EnableChecksum, [264](#)
  - EnableDigitalIn, [264](#), [265](#)
  - EnableTimestamp, [265](#)
  - Gain, [269](#)
  - GetAnalogGain, [266](#)
  - GetEnumerationSpeed, [266](#)
  - GetGain, [266](#)
  - MeaAudioFunctionNet, [269](#)
  - MeaDigitalDataFunctionNet, [269](#)
  - MeaFeedbackFunctionNet, [269](#)
  - MeFunctionNet, [270](#)
  - SetDigitalOut, [266](#)
  - SetNumberOfAnalogChannels, [266](#)
  - SetNumberOfChannels, [267](#), [268](#)
  - SetTriggerMaskValue, [268](#)
  - SetTriggerPeriod, [269](#)
  - W2100\_FunctionNet, [270](#)
  - WClassicFunctionNet, [270](#)
- CMeaDigitalDataFunctionNet, [270](#)
  - CMeaDigitalDataFunctionNet, [270](#)
  - GetDigitalData, [271](#)
  - SetDigitalData, [271](#)
- CMeaFeedbackFunctionNet, [272](#)
  - CMeaFeedbackFunctionNet, [273](#)
  - FeedbackGetSampleTimerCount, [273](#)
  - FeedbackSetAnalogSource, [273](#)
  - FeedbackSetChannelFilter, [273](#)
  - FeedbackSetDigitalMapping, [273](#)
  - FeedbackSetFeedback, [273](#)
  - FeedbackSetFilterOff, [274](#)
  - FeedbackSetFilterParameter, [274](#)
  - FeedbackSetFilterParameter32, [274](#)
  - FeedbackSetGlobalChannelFilter, [274](#)
  - FeedbackSetIIRFilterParameter, [274](#)
  - FeedbackSetLogic, [274](#)
  - FeedbackSetMkFilter, [274](#)
  - FeedbackSetNumberOfLogics, [275](#)
  - FeedbackSetNumberOfRateCounter, [275](#)
  - FeedbackSetNumberOfRateDetectors, [275](#)
  - FeedbackSetNumberOfSpikeDetectors, [275](#)
  - FeedbackSetNumberOfTriggers, [275](#)
  - FeedbackSetRateCounter, [275](#)
  - FeedbackSetRateDetector, [275](#)
  - FeedbackSetSpikeDetectorThreshold, [275](#)
  - FeedbackSetTrigger, [276](#)
- CMeaImpedanceDeviceNet, [276](#)
  - ~CMeaImpedanceDeviceNet, [276](#)
  - CMeaImpedanceDeviceNet, [276](#)
  - GetAdapterCode, [277](#)
  - GetArraySize, [277](#)
  - GetImpedanceTestFrequency, [277](#)
  - GetReady, [277](#)
  - GetResult, [277](#)
  - SetImpedanceTestFrequency, [277](#)
  - StartMeasurement, [277](#)
- CMeasureTableDeviceNet, [277](#)
  - CMeasureTableDeviceNet, [278](#)
  - Sensor, [278](#)
- CMeaSwitchDeviceNet, [278](#)
  - ~CMeaSwitchDeviceNet, [279](#)
  - CMeaSwitchDeviceNet, [279](#)
  - GetNumber, [279](#)
  - GetPattern, [279](#)
  - GetPatternBool, [280](#)
  - SetPattern, [280](#)
  - SetPatternBool, [280](#)
- CMeaUSBDeviceNet, [280](#)
  - ~CMeaUSBDeviceNet, [281](#)
  - CMeaUSBDeviceNet, [281](#)
- CMeFunctionNet, [281](#)
  - !CMeFunctionNet, [282](#)
  - ~CMeFunctionNet, [282](#)
  - CMeFunctionNet, [282](#)
  - SetTransformer, [282](#)
- CMosMea
  - CCMOSMeaDeviceNet, [47](#)
- CMultiBatteryChargerDeviceNet, [283](#)
  - !CMultiBatteryChargerDeviceNet, [284](#)
  - ~CMultiBatteryChargerDeviceNet, [284](#)
  - CapacityTest, [284](#)



- ChannelReset, [285](#)
- CMultiBatteryChargerDeviceNet, [284](#)
- GetBatteryVoltage, [285](#)
- GetChannels, [285](#)
- GetChannelState, [285](#)
- GetChargeCapacity, [286](#)
- GetChargeCurrent, [286](#)
- GetChargingMode, [286](#)
- GetChargingPCoefficient, [286](#)
- GetDischargeCapacity, [287](#)
- GetDischargeCurrent, [287](#)
- GetDischargeCurrentSetPoint, [287](#)
- GetFinalDischargeVoltage, [288](#)
- GetRatedCapacity, [288](#)
- SetChargingMode, [288](#)
- SetChargingPCoefficient, [288](#)
- SetDischargeCurrentSetPoint, [289](#)
- SetFinalDischargeVoltage, [289](#)
- SetRatedCapacity, [289](#)
- SetRatedCapacityVolatile, [289](#)
- CMultiwellCallbackFunctionNet, [290](#)
  - !CMultiwellCallbackFunctionNet, [291](#)
  - ~CMultiwellCallbackFunctionNet, [291](#)
  - CMultiwellCallbackFunctionNet, [291](#)
  - GetPlateClampStateByHeadstage, [291](#)
  - GetPlateClampStateByHeadstageEvent, [293](#)
  - GetPlateTypeByHeadstage, [292](#)
  - GetPlateTypeByHeadstageEvent, [293](#)
  - IsPlateTypeValidByHeadstage, [292](#)
  - IsPlateTypeValidByHeadstageEvent, [293](#)
  - OnGetPlateClampStateByHeadstage, [292](#)
  - OnGetPlateTypeByHeadstage, [292](#)
  - OnIsPlateTypeValidByHeadstage, [292](#)
- CMultiwellDeviceNet, [293](#)
  - !CMultiwellDeviceNet, [295](#)
  - ~CMultiwellDeviceNet, [295](#)
  - ClosePlateClamp, [295](#)
  - CMultiwellDeviceNet, [295](#)
  - GetPlateClampLockState, [295](#)
  - GetPlateClampState, [295](#)
  - GetPlateClampStateByHeadstage, [295](#)
  - GetPlateMux, [296](#)
  - GetPlateMuxByHeadstage, [296](#)
  - GetPlateType, [296](#)
  - GetPlateTypeByHeadstage, [296](#)
  - IsPlateTypeValid, [297](#)
  - IsPlateTypeValidByHeadstage, [297](#)
  - LockPlateClamp, [297](#)
  - OpenPlateClamp, [297](#)
  - SetPlateMux, [297](#)
  - SetPlateMuxByHeadstage, [298](#)
  - SetPlateType, [298](#)
  - SetPlateTypeByHeadstage, [298](#)
  - StopPlateClamp, [298](#)
  - UnlockPlateClamp, [298](#)
- CMultiwellOptoStimFunctionNet, [299](#)
  - !CMultiwellOptoStimFunctionNet, [300](#)
  - ~CMultiwellOptoStimFunctionNet, [300](#)
  - CMultiwellOptoStimFunctionNet, [299](#), [300](#)
  - GetAbsMaxCurrentInMicroAmp, [300](#)
  - GetColorRgb, [300](#)
  - GetColorStr, [301](#)
  - GetMaxDurationHighCurrentInMicroSec, [301](#)
  - GetMaxDutyCycleHighCurrent, [301](#)
  - GetPermanentCurrentInMicroAmp, [301](#)
  - GetWaveLengthInNanometer, [302](#)
  - SetAbsMaxCurrentInMicroAmp, [302](#)
  - SetColorRgb, [302](#)
  - SetColorStr, [302](#)
  - SetMaxDurationHighCurrentInMicroSec, [303](#)
  - SetMaxDutyCycleHighCurrent, [303](#)
  - SetPermanentCurrentInMicroAmp, [303](#)
  - SetWaveLengthInNanometer, [303](#)
- CNF\_GenDeviceNet, [304](#)
  - ~CNF\_GenDeviceNet, [304](#)
  - CNF\_GenDeviceNet, [304](#)
  - Set\_Values, [304](#)
- COctoPotDeviceNet, [304](#)
  - BurnAdcOffset, [305](#)
  - BurnDacOffset, [306](#)
  - COctoPotDeviceNet, [305](#)
  - EnableChecksum, [306](#)
  - EnableDigitalIn, [306](#)
  - EnableTimestamp, [306](#)
  - GetAdcOffset, [306](#)
  - GetDacOffset, [306](#)
  - PatternListStart, [306](#)
  - RampStart, [306](#)
  - ResetAdcOffset, [306](#)
  - ResetDacOffset, [306](#)
  - SetAdcOffset, [307](#)
  - SetAmplificationSwitch, [307](#)
  - SetBathclamp, [307](#)
  - SetChannelSwitch, [307](#)
  - SetDacAutoControl, [307](#)
  - SetDacOffset, [307](#)
  - SetDacValue, [307](#)
  - SetNumberOfChannels, [307](#)
  - SetOutputRate, [307](#)
  - SetPatternListEntry, [308](#)
  - SetPidParameter, [308](#)
  - SetRampParameter, [308](#)
  - SetSineParameter, [308](#)
  - SineStart, [308](#)
- CokuvisionStimulatorDeviceNet, [308](#)
  - ~CokuvisionStimulatorDeviceNet, [309](#)
  - CokuvisionStimulatorDeviceNet, [309](#)
  - GetCheckVoltage, [309](#)
  - GetCurrentFactor, [309](#)
  - GetDACOffset, [310](#)
  - GetMaxPower, [310](#)
  - GetMaxVoltage, [310](#)
  - GetPulseform, [310](#)
  - GetRTC, [310](#)
  - GetStimulatorStatus, [310](#)
  - GetVoltage, [310](#)

- SetCheckVoltage, 311
- SetCurrentFactor, 311
- SetDACOffset, 311
- SetMaxPower, 311
- SetMaxVoltage, 311
- SetPulseform, 311
- SetRTC, 311
- Coldstart
  - CMcsUsbFactoryNet, 211
- CompareFirmware
  - CMcsUsbFactoryNet, 211
- CompareTo
  - HeadStageIDType, 578
- CompensateElectrodeOffset
  - CWarnerUssingFunctionNet, 520
- Connect
  - CMcsUsbNet, 230, 231
  - CRFFunctionNet, 349
- ConnectDevice
  - CRadioControlledDevicesNet, 344
- ConnectedImp
  - CPositionImpDeviceNet, 323
- ConnectImp
  - CPositionImpDeviceNet, 323
- ConnectSlave
  - CGilsonDeviceNet, 99
- ControlState
  - HeadStageIDTypeState, 581
- CornerFrequency
  - CFilterPropertyNet, 72
- CornerFrequencycmHz
  - CFilterPropertyNet, 72
- Count
  - CMcsUsbListNet, 225
- CPatchServerDeviceNet, 312
  - CPatchServerDeviceNet, 312
  - Sensor, 312
- CPathIdentDeviceNet, 313
  - ~CPathIdentDeviceNet, 313
  - CPathIdentDeviceNet, 313
  - Measure, 313
  - Set\_Values, 313
- CPedoterDeviceNet, 314
  - !CPedoterDeviceNet, 314
  - ~CPedoterDeviceNet, 314
  - CPedoterDeviceNet, 314
  - GetCommand, 314
  - SetCommand, 315
- CPeristalticPumpDeviceNet, 315
  - ~CPeristalticPumpDeviceNet, 316
  - CPeristalticPumpDeviceNet, 316
  - McsBus\_MotorControl, 316
- CPgaDeviceNet, 316
  - ~CPgaDeviceNet, 317
  - ApplyGains, 317
  - CPgaDeviceNet, 317
  - DefineAmplification, 317
  - DefineFrequencyRange, 317
  - DefineNumAmplifications, 317
  - DefineNumFrequencyRanges, 317
  - GetAmplification, 317
  - GetFrequencyRange, 318
  - GetGain, 318
  - GetNumAmplifications, 318
  - GetNumFrequencyRanges, 318
  - SetGain, 318
- CPositionIIDeviceNet, 318
  - !CPositionIIDeviceNet, 319
  - ~CPositionIIDeviceNet, 319
  - CPositionIIDeviceNet, 319
  - GetCoilCommunication, 319
  - GetImplantCurrentSetpoint, 320
  - GetImplantState, 320
  - GetOnOff, 320
  - RFFunction, 321
  - SetImplantCurrentSetpoint, 321
  - SwitchOnOff, 321
- CPositionImpDeviceNet, 321
  - !CPositionImpDeviceNet, 322
  - ~CPositionImpDeviceNet, 322
  - ConnectedImp, 323
  - ConnectImp, 323
  - CPositionImpDeviceNet, 322
  - GetDeviceList, 323
  - GetImpld, 323
  - GetRFFrequency, 323
  - SetDeviceList, 324
  - SetImpld, 324
  - SetRFFrequency, 324
- CPPCDeviceNet, 324
  - CPPCDeviceNet, 325
  - McsBus, 325
  - McsBus\_MotorControl, 325
  - McsBus\_Sensor, 325
  - PPCFunction, 325
- CPPPCFunctionNet, 326
  - !CPPPCFunctionNet, 327
  - ~CPPPCFunctionNet, 327
  - CPPPCFunctionNet, 327
  - FirePressurePulse, 327
  - GetAnalogVoltage, 328
  - GetAnalogVoltageRange, 328
  - GetDigitalIn, 328
  - GetPressureRange, 329
  - GetPumpModeType, 329
  - GetPumpSpeedUnit, 329
  - GetSupplyVoltage, 329
  - GetValveActive, 330
  - IsBusy, 330
  - LoadPressure, 330
  - MeasureReservoir, 331
  - SetAnalogVoltageRange, 331
  - SetPressureOffset, 331
  - SetPressureRange, 331
  - SetPumpModeType, 331
  - SetPumpSpeedUnit, 332



- SetValveActive, [332](#)
- CPPS\_DeviceNet, [332](#)
  - CPPS\_DeviceNet, [333](#)
  - McsBus, [333](#)
  - McsBus\_MotorControl, [333](#)
  - McsBus\_Sensor, [333](#)
  - PPS\_Function, [333](#)
- CPPS\_FunctionNet, [333](#)
  - CPPS\_FunctionNet, [334](#)
  - GetAnalogVoltage, [335](#)
  - GetAnalogVoltages, [335](#)
  - GetBubbleState, [335](#)
  - GetDigitalIn, [335](#)
  - GetPumpCouple, [335](#)
  - GetPumpEnableSpeedRatio, [335](#)
  - GetPumpFastOnOff, [335](#)
  - GetPumpFastSpeed, [335](#)
  - GetPumpFunctionSpeeds, [335](#)
  - GetPumpManualOnOff, [335](#)
  - GetPumpMaxSpeed, [336](#)
  - GetPumpModeType, [336](#)
  - GetPumpSpeedRatio, [336](#)
  - GetPumpSpeedUnit, [336](#)
  - GetSupplyVoltage, [336](#)
  - GetUseBubble, [336](#)
  - SetAnalogVoltages, [336](#)
  - SetPumpCouple, [336](#)
  - SetPumpEnableSpeedRatio, [336](#)
  - SetPumpFastOnOff, [336](#)
  - SetPumpFastSpeed, [337](#)
  - SetPumpFunctionSpeeds, [337](#)
  - SetPumpManualOnOff, [337](#)
  - SetPumpMaxSpeed, [337](#)
  - SetPumpModeType, [337](#)
  - SetPumpSpeedRatio, [337](#)
  - SetPumpSpeedUnit, [337](#)
  - SetUseBubble, [337](#)
- CPPSDeviceNet, [338](#)
  - CPPSDeviceNet, [338](#)
- CProgramPressureCurveNet, [338](#)
  - !CProgramPressureCurveNet, [339](#)
  - ~CProgramPressureCurveNet, [339](#)
  - CProgramPressureCurveNet, [339](#)
  - GetRepeats, [339](#)
  - Program, [339](#)
  - SetRepeats, [340](#)
- CPulseGeneratorFunctionNet, [340](#)
  - !CPulseGeneratorFunctionNet, [341](#)
  - ~CPulseGeneratorFunctionNet, [341](#)
  - CPulseGeneratorFunctionNet, [341](#)
  - GetModeSelect, [341](#)
  - GetPeriod, [342](#)
  - GetPulseLength, [342](#)
  - SetModeSelect, [342](#)
  - SetPeriod, [342](#)
  - SetPulseLength, [343](#)
- CRadioControlledDevicesNet, [343](#)
  - ConnectDevice, [344](#)
  - CRadioControlledDevicesNet, [344](#)
  - DisconnectDevice, [344](#)
  - GetDeviceNames, [344](#)
  - GetFrequency, [344](#)
  - HasRadioControl, [344](#)
  - SetFrequency, [344](#)
  - StillConnected, [344](#)
- CreateSideband
  - CStimulusFunctionNet, [464](#)
- CreateWirelessHeadstageSerialNumberString
  - CWirelessBaseFunctionNet, [565](#)
- CRegionOfInterestRect
  - CCMOSMeaDeviceNet::CRegionOfInterestRect, [345](#)
- CRetinaLedDeviceNet, [346](#)
  - ~CRetinaLedDeviceNet, [346](#)
  - AddLoopEntry, [346](#)
  - AddTableEntry, [346](#)
  - ClearTable, [347](#)
  - CRetinaLedDeviceNet, [346](#)
  - GetTablepointer, [347](#)
  - SetLED, [347](#)
  - SetLumi, [347](#)
  - SetPersistency, [347](#)
  - SetRepeat, [347](#)
  - SetTablepointer, [347](#)
  - SetTrigger, [347](#)
- CRFFFunctionNet, [348](#)
  - !CRFFFunctionNet, [349](#)
  - ~CRFFFunctionNet, [349](#)
  - Connect, [349](#)
  - CRFFFunctionNet, [349](#)
  - GetAvailableDeviceList, [349](#)
  - GetAvailableDeviceListEx, [349](#)
  - GetAvailableStateList, [350](#)
  - GetAvailableStateListEx, [350](#)
  - GetBaseFrequency, [350](#)
  - GetConnectedDevice, [351](#)
  - GetState, [351](#)
  - GetTestMode, [351](#)
  - GetWorkingFrequency, [351](#)
  - SetBaseFrequency, [351](#)
  - SetTestMode, [352](#)
  - SetWorkingFrequency, [352](#)
- CRobo\_FYIProgram\_FunctionNet, [352](#)
  - CRobo\_FYIProgram\_FunctionNet, [353](#)
  - GetLength, [353](#)
  - GetState, [353](#)
  - GetValve1, [353](#)
  - GetValve2, [353](#)
  - SetLength, [353](#)
  - SetValve1, [353](#)
  - SetValve2, [354](#)
  - Start, [354](#)
- CRobo\_FYITemp\_FunctionNet, [354](#)
  - CRobo\_FYITemp\_FunctionNet, [354](#)
  - GetICoeff, [355](#)
  - GetMaxPower, [355](#)

- GetPCoeff, 355
- GetRegulatorOnOff, 355
- GetSollTemp, 355
- SetICoeff, 355
- SetMaxPower, 355
- SetPCoeff, 355
- SetRegulatorOnOff, 355
- SetSollTemp, 356
- CRoboDacqNet, 356
  - CancelTableLoop, 358
  - CancelTableLoopAndStopTable, 358
  - ClampAmpRestart, 359
  - CRoboDacqNet, 358
  - DoRamp, 359
  - Emu\_GetCellCapacity, 359
  - Emu\_GetCellPotential, 359
  - Emu\_GetCellResists, 359
  - Emu\_GetElectrodeResists, 359
  - Emu\_GetNoise, 359
  - Emu\_SetCellCapacity, 359
  - Emu\_SetCellPotential, 359
  - Emu\_SetCellResists, 359
  - Emu\_SetElectrodeResists, 360
  - Emu\_SetNoise, 360
  - GetAllDigout, 360
  - GetCapacityC, 360
  - GetCapacityV, 360
  - GetCapacityX, 360
  - GetClampAmpSerialNumber, 360
  - GetCommand, 360
  - GetConfigurationBit, 360
  - GetConfigurationBitAxc, 360
  - GetConfigurationBitBlu\_Led, 361
  - GetConfigurationBitBlu\_LedToggleFast, 361
  - GetConfigurationBitBlu\_LedToggleSlow, 361
  - GetConfigurationBitCC\_Gen, 361
  - GetConfigurationBitCV\_Gen, 361
  - GetConfigurationBitRC\_Gen, 361
  - GetConfigurationBitRed\_Led, 361
  - GetConfigurationBitRed\_LedSaturation, 361
  - GetConfigurationBitRed\_LedToggleFast, 361
  - GetConfigurationBitRed\_LedToggleSlow, 361
  - GetConfigurationBitRelais, 361
  - GetConfigurationBitRV\_Gen, 362
  - GetConfigurationBits, 362
  - GetConfigurationBitStream, 362
  - GetConfigurationBitSupply, 362
  - GetCrossTalkOffset, 362
  - GetCrossTalkOptimum, 362
  - GetDigout, 362
  - GetDisplayText, 362
  - GetDownsampleFactor, 362
  - GetFilter, 362
  - GetFilterCoeffs, 362
  - GetIC, 363
  - GetIClamp, 363
  - GetICOffset, 363
  - GetIGain, 363
  - GetNIC\_MS, 363
  - GetNUC\_MS, 363
  - GetNUV\_MS, 363
  - GetPGain, 363
  - GetRecordingNumber, 363
  - GetResistanceC, 363
  - GetResistanceV, 363
  - GetScreen, 364
  - GetSimulation, 364
  - GetUC, 364
  - GetUClamp, 364
  - GetUCOffset, 364
  - GetUpdateDisplay, 364
  - GetUV, 364
  - GetUVOffset, 364
  - GetXGain, 364
  - RunTable, 364
  - SetAllDigout, 365
  - SetCommand, 365
  - SetConfigurationBit, 365
  - SetConfigurationBitAxc, 365
  - SetConfigurationBitBlu\_Led, 365
  - SetConfigurationBitBlu\_LedToggleFast, 365
  - SetConfigurationBitBlu\_LedToggleSlow, 365
  - SetConfigurationBitCC\_Gen, 365
  - SetConfigurationBitCV\_Gen, 365
  - SetConfigurationBitRC\_Gen, 366
  - SetConfigurationBitRed\_Led, 366
  - SetConfigurationBitRed\_LedSaturation, 366
  - SetConfigurationBitRed\_LedToggleFast, 366
  - SetConfigurationBitRed\_LedToggleSlow, 366
  - SetConfigurationBitRelais, 366
  - SetConfigurationBitRV\_Gen, 366
  - SetConfigurationBitStream, 366
  - SetConfigurationBitSupply, 366
  - SetCrossTalkOffset, 366
  - SetCrossTalkOptimum, 367
  - SetDigout, 367
  - SetDisplayText, 367
  - SetDownsampleFactor, 367
  - SetFilter, 367
  - SetFilterCoeffs, 367
  - SetIClamp, 367
  - SetICOffset, 367
  - SetIGain, 367
  - SetPGain, 368
  - SetRecordingNumber, 368
  - SetScreen, 368
  - SetSimulation, 368
  - SetUClamp, 368
  - SetUCOffset, 368
  - SetUVOffset, 368
  - SetXGain, 368
  - StopTable, 368
  - Table\_Wait, 369
  - TableDefBegin, 369
  - TableDefEnd, 369
  - UpdateDisplay, 369

- CRoboDeviceNet, [369](#)
  - ~CRoboDeviceNet, [372](#)
  - Axes\_I, [379](#)
  - Axes\_X, [379](#)
  - Axes\_Y, [379](#)
  - Axes\_Z, [379](#)
  - Axis\_I, [379](#)
  - Axis\_X, [379](#)
  - Axis\_Y, [379](#)
  - Axis\_Z, [380](#)
  - CancelPoolLoop, [373](#)
  - CancelPoolLoopAndStopMovement, [373](#)
  - CRoboDeviceNet, [372](#)
  - FindReference, [373](#)
  - GetAirpressure, [373](#)
  - GetAirpressureLimit, [373](#)
  - GetAirValve, [373](#)
  - GetCurrentAirvalve, [373](#)
  - GetCurrentAirvalveLimit, [374](#)
  - GetCurrentPosition, [374](#)
  - GetErrorAirpressure, [374](#)
  - GetErrorCurrentAirvalve, [374](#)
  - GetErrorVoltage12V, [374](#)
  - GetErrorVoltage5V, [374](#)
  - GetErrorVoltageAirvalve, [374](#)
  - GetErrorVoltageRs485A, [374](#)
  - GetErrorVoltageRs485B, [375](#)
  - GetErrorVoltageValves, [375](#)
  - GetInMovement, [375](#)
  - GetMinPressure, [375](#)
  - GetMovementError, [375](#)
  - GetVoltage12V, [375](#)
  - GetVoltage12VLimit, [375](#)
  - GetVoltage5V, [375](#)
  - GetVoltage5VLimit, [375](#)
  - GetVoltageAirvalve, [375](#)
  - GetVoltageAirvalveLimit, [376](#)
  - GetVoltageRs485A, [376](#)
  - GetVoltageRs485ALimit, [376](#)
  - GetVoltageRs485B, [376](#)
  - GetVoltageRs485BLimit, [376](#)
  - GetVoltageValves, [376](#)
  - GetVoltageValvesLimit, [376](#)
  - McsBus, [383](#)
  - McsBus\_MotorControl, [383](#)
  - McsBus\_XY, [380](#)
  - McsBus\_ZI, [380](#)
  - MoveAbs, [376](#)
  - RoboError\_AnotherMaster, [380](#)
  - RoboError\_Base, [380](#)
  - RoboError\_CannotEscapeEndSwitch, [380](#)
  - RoboError\_CommandAlreadyInProgress, [380](#)
  - RoboError\_CommandNotPossible, [380](#)
  - RoboError\_CommunicationTimeout, [380](#)
  - RoboError\_DacqNotReady, [381](#)
  - RoboError\_DLLMovementTimeout, [381](#)
  - RoboError\_FindReferenceMethod, [381](#)
  - RoboError\_GilsonCommandPending, [381](#)
  - RoboError\_GilsonTimeout, [381](#)
  - RoboError\_GilsonWrondID, [381](#)
  - RoboError\_McsBus\_UnknownCommand, [381](#)
  - RoboError\_NoEndSwitch, [381](#)
  - RoboError\_NoMoreData, [381](#)
  - RoboError\_NoReference, [381](#)
  - RoboError\_NoSpeedOrAcceleration, [382](#)
  - RoboError\_OverPressure, [382](#)
  - RoboError\_ParameterNotAllowed, [382](#)
  - RoboError\_PeristalticTimeout, [382](#)
  - RoboError\_Phase0OutOfRange, [382](#)
  - RoboError\_PollLoopCanceled, [382](#)
  - RoboError\_PollLoopCanceledAndStopMovement, [382](#)
  - RoboError\_Pressure, [382](#)
  - RoboError\_RangeExceeded, [382](#)
  - RoboError\_StateChangeNotPossible, [382](#)
  - RoboError\_Timeout, [383](#)
  - RoboError\_UnknownCommand, [383](#)
  - RoboMainLowLevelCommand, [383](#)
  - RoboStatusEvent, [383](#)
  - SetAirpressureLimit, [377](#)
  - SetAirValve, [377](#)
  - SetCurrentAirvalveLimit, [377](#)
  - SetCurrentAndAir, [377](#)
  - SetInMovement, [377](#)
  - SetMinPressure, [377](#)
  - SetVoltage12VLimit, [378](#)
  - SetVoltage5VLimit, [378](#)
  - SetVoltageAirvalveLimit, [378](#)
  - SetVoltageRs485ALimit, [378](#)
  - SetVoltageRs485BLimit, [378](#)
  - SetVoltageValvesLimit, [378](#)
  - StopMovement, [378](#)
- CRoboDeviceNet::RoboMainLowLevelCommands, [585](#)
  - FindReferencePhase0, [586](#)
  - GetAxisConfig, [586](#)
  - GetHWConfig, [586](#)
  - GetHWRevision, [586](#)
  - GetMaxNoPressure, [586](#)
  - GetMaxNoPressureWaitTime, [586](#)
  - GetMaxPressureWaitTime, [586](#)
  - GetMinNoPressureWaitTime, [586](#)
  - GetMinPressure, [586](#)
  - GetMinPressureWaitTime, [587](#)
  - GetParameter, [587](#)
  - GetPhases, [587](#)
  - GetSearchReferenceFastAccel, [587](#)
  - GetSearchReferenceFastSpeed, [587](#)
  - GetSearchReferenceFineAccel, [587](#)
  - GetSearchReferenceFineSpeed, [587](#)
  - GetSearchReferenceMethod, [588](#)
  - GetSearchReferenceMoveOut, [588](#)
  - GetSearchReferenceOffsetPos, [588](#)
  - GetUserParameter, [588](#)
  - HasRef, [589](#)
  - SetAxisConfig, [589](#)
  - SetHWConfig, [589](#)

- SetHWRevision, [589](#)
- SetMaxNoPressure, [589](#)
- SetMaxNoPressureWaitTime, [589](#)
- SetMaxPressureWaitTime, [589](#)
- SetMinNoPressureWaitTime, [589](#)
- SetMinPressure, [589](#)
- SetMinPressureWaitTime, [589](#)
- SetParameter, [590](#)
- SetSearchReferenceFastAccel, [590](#)
- SetSearchReferenceFastSpeed, [590](#)
- SetSearchReferenceFineAccel, [590](#)
- SetSearchReferenceFineSpeed, [590](#)
- SetSearchReferenceMethod, [590](#)
- SetSearchReferenceMoveOut, [591](#)
- SetSearchReferenceOffsetPos, [591](#)
- SetUserParameter, [591](#)
- CRoboFluidDeviceNet, [383](#)
  - ~CRoboFluidDeviceNet, [384](#)
  - CloseAllValves, [384](#)
  - CRoboFluidDeviceNet, [384](#)
  - GetPumpSpeed, [385](#)
  - GetSingleValve, [385](#)
  - GetValve, [385](#)
  - IsPumpMotorOn, [385](#)
  - m\_pMcsBus\_MotorControlNet, [386](#)
  - m\_pRoboFluidDevice, [386](#)
  - McsBus\_MotorControl, [386](#)
  - PumpOff, [385](#)
  - PumpOn, [385](#)
  - SetPumpSpeed, [385](#)
  - SetSingleValve, [386](#)
  - SetValve, [386](#)
- CRoboInjectDeviceNet, [387](#)
  - CRoboInjectDeviceNet, [387](#)
- CRoboocyte2DeviceNet, [387](#)
  - CRoboocyte2DeviceNet, [388](#)
  - GetAxisLED, [388](#)
  - GetGilsonDevice, [388](#)
  - GetMcsBus\_Extension, [388](#)
  - GetRoboDacq, [388](#)
  - GetRoboFluidDevice, [388](#)
  - SetAxisLED, [389](#)
- CRoboStatorDeviceNet, [389](#)
  - CRoboStatorDeviceNet, [390](#)
  - FindReferenceI, [390](#)
  - FindReferenceXY, [390](#), [391](#)
  - FindReferenceZ, [391](#)
  - GetCurrentPositionI, [391](#)
  - GetCurrentPositionXY, [391](#)
  - GetCurrentPositionZ, [391](#)
  - HasRefI, [391](#)
  - HasRefXY, [391](#)
  - HasRefZ, [391](#)
  - MoveAbsI, [391](#), [392](#)
  - MoveAbsXY, [392](#)
  - MoveAbsZ, [392](#)
  - RoboMainStatorLowLevelCommand, [395](#)
  - SetAccelerationI, [392](#)
  - SetAccelerationNativeI, [392](#)
  - SetAccelerationNativeXY, [392](#)
  - SetAccelerationNativeZ, [392](#)
  - SetAccelerationXY, [393](#)
  - SetAccelerationZ, [393](#)
  - SetCurrentAndAirXY, [393](#)
  - SetSpeedI, [393](#)
  - SetSpeedNativeI, [393](#)
  - SetSpeedNativeXY, [393](#)
  - SetSpeedNativeZ, [393](#)
  - SetSpeedXY, [393](#)
  - SetSpeedZ, [394](#)
  - SetVelocityI, [394](#)
  - SetVelocityXY, [394](#)
  - SetVelocityZ, [394](#)
  - StopMovementI, [394](#)
  - StopMovementXY, [394](#)
  - StopMovementZ, [394](#)
- CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands, [592](#)
  - FindReferencePhase0XY, [592](#)
- CSafeISDeviceNet, [395](#)
  - ~CSafeISDeviceNet, [396](#)
  - CSafeISDeviceNet, [396](#)
  - DacqDevice, [397](#)
  - FluidControlDevice, [398](#)
  - RoboDevice, [398](#)
  - SetAdcChannels, [396](#)
  - SetAdcSamplePos, [396](#)
  - SetDacMode, [396](#)
  - SetDacPeriode, [397](#)
  - SetDacPulseform, [397](#)
  - SetSwitches, [397](#)
- CSCUDacqGroupChannelSelectionNet, [398](#)
  - CSCUDacqGroupChannelSelectionNet, [398](#)
- CSCUFunctionNet, [399](#)
  - !CSCUFunctionNet, [401](#)
  - ~CSCUFunctionNet, [401](#)
  - AutomaticAnalogOut, [401](#)
  - CSCUFunctionNet, [401](#)
  - EnableAnalogOut, [402](#)
  - GetAnalogOutADCRange, [402](#)
  - GetAnalogOutChannels, [402](#)
  - GetAnalogOutDACRange, [402](#)
  - GetAvailableHeadstages, [402](#)
  - GetAvailableHeadstagesEvent, [414](#)
  - GetFilterProperties, [403](#)
  - GetFilterProperty, [403](#)
  - GetHeadstageAdcBits, [403](#)
  - GetHeadstageAdcRangeInMicroVolt, [404](#)
  - GetHeadstageDacBits, [404](#)
  - GetHeadstageDacCurrentRangeInMicroAmpere, [404](#)
  - GetHeadstageDacCurrentResolutionInNanoAmpere, [405](#)
  - GetHeadstageDacVoltageRangeInMilliVolt, [405](#)
  - GetHeadstageDacVoltageResolutionInMicroVolt, [405](#)

- GetHeadstageGainInPer mille, 406
- GetHeadstageID, 406
- GetHeadstageNumberOfAnalogChannels, 406
- GetHeadstageNumberOfStimulationChannels, 407
- GetHeadstageSamplerate, 407
- GetHeadstageSerialNumber, 407
- GetMaxNumberOfHeadstages, 408
- GetMaxStimulusChannelsPerHeadstage, 408
- GetReferenceElectrodeMode, 408
- GetReferenceElectrodeSwitchState, 408
- HasAnalogOut, 409
- HasGalvanicIsolation, 409
- HasHSPowerSwitch, 409
- IsAnalogOutEnabled, 409
- IsAutomaticAnalogOut, 410
- IsHeadstageAvailable, 410
- IsHeadstageAvailableEvent, 414
- IsHSPowered, 410
- IsInDacqLegacyMode, 410
- OnGetAvailableHeadstages, 411
- OnIsHeadstageAvailable, 411
- PowerHS, 411
- SetAnalogOutADCRange, 411
- SetAnalogOutChannels, 411
- SetAnalogOutDACRange, 413
- SetDacqLegacyMode, 413
- SetReferenceElectrodeMode, 413
- SetReferenceElectrodeSwitchState, 413
- CSerialPortNet, 414
  - CSerialPortNet, 414
  - GetBytesAvailable, 415
  - Receive, 415
  - ReceiveString, 415
  - Send, 415
- CStg200xBasicNet, 416
  - ~CStg200xBasicNet, 420
  - GetAnalogRanges, 420
  - GetAnalogResolution, 420
  - GetAutocalibrationDisabled, 421
  - GetAvailableMemory, 421
  - GetBlankingEnable, 421, 422
  - GetCurrentRangeInNanoAmp, 422
  - GetCurrentResolutionInNanoAmp, 422
  - GetDacAmplificationFactor, 423
  - GetDACResolution, 423
  - GetDiginValue, 423
  - GetDigoutMode, 423
  - GetDigoutValue, 424
  - GetElectrodeDacMux, 424
  - GetElectrodeEnable, 425
  - GetElectrodeMode, 426
  - GetEnableAmplifierProtectionSwitch, 426, 427
  - GetExternalElectrodeEnable, 427
  - GetFAAmplification, 428
  - GetHeadstage, 428
  - GetListModelIndexRange, 428
  - GetListmodeTriggerSource, 428
  - GetNumberOfAnalogChannels, 428
  - GetNumberOfHWDACPaths, 428
  - GetNumberOfStimulationElectrodes, 428
  - GetNumberOfStimulationSourcesPerElectrode, 429
  - GetNumberOfSyncoutChannels, 429
  - GetNumberOfTriggerInputs, 429
  - GetOutputRate, 429
  - GetStgProgramInfo, 429, 430
  - GetStgVersionInfo, 430
  - GetSyncoutMap, 430
  - GetTotalMemory, 431
  - GetTriggerSource, 431
  - GetVoltageRangeInMicroVolt, 431
  - GetVoltageResolutionInMicroVolt, 431
  - ListModeSendStart, 432
  - ListModeSendStop, 432
  - SendStart, 432
  - SendStop, 432
  - SetAutocalibrationDisabled, 432
  - SetBlankingEnable, 433, 434
  - SetCurrentMode, 434
  - SetDacAmplificationFactor, 434
  - SetDigoutMode, 435
  - SetDigoutValue, 435
  - SetElectrodeDacMux, 435–437
  - SetElectrodeEnable, 437–439
  - SetElectrodeMode, 440, 441
  - SetEnableAmplifierProtectionSwitch, 441, 442
  - SetExternalElectrodeEnable, 443
  - SetFAAmplification, 444
  - SetHeadstage, 444
  - SetListModelIndexRange, 444
  - SetListmodeTriggerSource, 444
  - SetMeasurementMode, 444
  - SetOutputRate, 445
  - SetStgProgramInfo, 445
  - SetSyncoutMap, 445
  - SetTriggerSource, 445, 446
  - SetVoltageMode, 446
- CStg200xDownloadBasicNet, 446
  - ClearChannelData, 448
  - ClearSyncData, 448
  - DisableAutoReset, 448
  - EnableAutoReset, 449
  - ForceStatusEvent, 449
  - GetMemoryUsageDAC, 449
  - GetMemoryUsageSyncout, 449
  - GetSweepCount, 450
  - GetTrigger, 450
  - ResetStatus, 450
  - SendChannelData, 452
  - SendSyncData, 452
  - SetupRetriggerMode, 453
  - SetupTrigger, 453
  - SetupTriggerSingle, 454
  - Stimulus, 455
- CStg200xDownloadNet, 455
  - ~CStg200xDownloadNet, 456

- ClearChannel\_PrepareAndSendData, 457
- CStg200xDownloadNet, 456
- DisableMultiFileMode, 457
- EnableMultiFileMode, 457
- GetModuleCurrent, 458
- GetModuleTemp, 458
- MwPollStatusEvent, 461
- PrepareAndAppendData, 458
- PrepareAndSendData, 459
- QueryTriggerstatus, 460
- SendSegmentDefine, 460
- SendSegmentSelect, 460
- SendSegmentStart, 461
- SetOutputMap, 461
- Stg200xPollStatusEvent, 461
- CStimulusFunctionNet, 462
  - ClearChannel\_PrepareAndSendData, 464
  - ClearChannelData, 464
  - ClearMultiplexedData, 464
  - ClearSyncData, 464
  - CreateSideband, 464
  - CStimulusFunctionNet, 463
  - ForceStatusEvent, 465
  - GetAvailableMemory, 465
  - GetCurrentRangeInNanoAmp, 466
  - GetCurrentResolutionInNanoAmp, 466
  - GetDACResolution, 466
  - GetMultiplexedDataChannelsInBlock, 466
  - GetNumberOfAnalogChannels, 467
  - GetTotalMemory, 467
  - GetVoltageRangeInMicroVolt, 467
  - GetVoltageResolutionInMicroVolt, 467
  - PollStatusEvent, 473
  - PrepareAndAppendData, 468
  - PrepareAndSendData, 469
  - PrepareData, 469
  - SendMultiplexedData, 470
  - SendPreparedData, 470
  - SendStart, 470
  - SendStop, 470
  - SetupTrigger, 471
  - SetupTriggerSingle, 471
  - StartPoll, 473
  - StopPoll, 473
- CStimulusFunctionNet::SidebandData, 592
  - !SidebandData, 593
  - ~SidebandData, 593
  - Duration, 593
  - Sideband, 593
  - SidebandData, 593
- CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, 594
  - !StimulusDeviceDataAndUnrolledData, 595
  - ~StimulusDeviceDataAndUnrolledData, 595
  - DeviceData, 595
  - DeviceDataLength, 595
  - StimulusDeviceDataAndUnrolledData, 594
  - UnrolledAmplitude, 595
  - UnrolledDuration, 595
  - UnrolledSync, 595
- Csw2to64DeviceNet, 473
  - ~Csw2to64DeviceNet, 474
  - Csw2to64DeviceNet, 474
  - GetChannel, 474
  - GetChannels, 475
  - GetNumber, 475
  - SetChannel, 475
  - SetChannels, 475
- CTcxDeviceNet, 476
  - ~CTcxDeviceNet, 478
  - CalibrateThermocouple, 478
  - CTcxDeviceNet, 478
  - FactoryReset, 478
  - GetBoardTemp, 479
  - GetCalibration, 479
  - GetCalibrationDecp, 479
  - GetCalibrationMax, 479
  - GetCalibrationMin, 479
  - GetCurrent, 479
  - GetD, 479
  - GetDDecp, 479
  - GetDevice, 479
  - GetDeviceType, 480
  - GetDevname, 480
  - GetDMax, 480
  - GetDMin, 480
  - GetDuty, 480
  - GetEnableHeaterLimit, 480
  - GetEnableThermocouple, 480
  - GetHasThermocouple, 480
  - GetHeaterLimit, 481
  - GetHeaterTemp, 481
  - GetI, 481
  - GetIDecp, 481
  - GetIMax, 481
  - GetIMin, 481
  - GetIOut, 481
  - GetMaxHeaterPowerMultiwell, 482
  - GetMaxP, 482
  - GetMaxpDecp, 482
  - GetMaxpMax, 482
  - GetMaxpMin, 482
  - GetNumControlChannels, 482
  - GetNumDevices, 482
  - GetNumMeasureChannels, 482
  - GetOnOff, 483
  - GetP, 483
  - GetPDecp, 483
  - GetPMax, 483
  - GetPMin, 483
  - GetPOut, 483
  - GetPwrOut, 483
  - GetPwrSet, 484
  - GetRes1, 484
  - GetRes2, 484
  - GetResS, 484



- GetResX, [484](#)
- GetROut, [484](#)
- GetSensorType, [484](#)
- GetSetpoint, [484](#)
- GetSetpointDecp, [485](#)
- GetSetpointMax, [485](#)
- GetSetpointMin, [485](#)
- GetThermocoupleCalibration, [485](#)
- GetThermocoupleNanovoltPerKelvin, [485](#)
- GetThermocoupleReferenceTemp, [485](#)
- GetThermocoupleTemp, [485](#)
- GetThermocoupleTempAbs, [486](#)
- GetUnit, [486](#)
- GetUOut, [486](#)
- GetValue, [486](#)
- GetValueHires, [486](#)
- GetVoli, [486](#)
- SetCalibration, [486](#)
- SetD, [487](#)
- SetDevice, [487](#)
- SetDeviceType, [487](#)
- SetDevname, [487](#)
- SetEnableHeaterLimit, [487](#)
- SetEnableThermocouple, [487](#)
- SetHeaterLimit, [487](#)
- SetI, [487](#)
- SetMaxHeaterPowerMultiwell, [488](#)
- SetMaxP, [488](#)
- SetOnOff, [488](#)
- SetP, [488](#)
- SetSensorType, [488](#)
- SetSetpoint, [488](#)
- SetThermocoupleNanovoltPerKelvin, [489](#)
- CTEERFunctionNet, [489](#)
  - !CTEERFunctionNet, [491](#)
  - ~CTEERFunctionNet, [491](#)
- CancelInternalCalibration, [491](#)
- CTEERFunctionNet, [491](#)
- GetAdapterCode, [491](#)
- GetAdcOffsetU1, [492](#)
- GetAdcOffsetU2, [492](#)
- GetAmplitude\_nA, [492](#)
- GetBytesPerSample, [492](#)
- GetClampMode, [492](#)
- GetControllerParams, [492](#)
- GetCurrentEnable, [493](#)
- GetDacZero, [493](#)
- GetLiquidResistance, [493](#)
- GetMaxChunkSize\_Byte, [493](#)
- GetNumberOfAvailableSamples, [493](#)
- GetPeriod\_us, [494](#)
- GetRotaryPositionCode, [494](#)
- GetSampleBufferChunk, [494](#)
- GetSampleVoltageBuffer\_uV, [494](#)
- GetScaleFactorU1, [495](#)
- GetScaleFactorU2, [495](#)
- GetUptimeSeconds, [495](#)
- GetWaveform, [495](#)
- IsInternalCalibrationFinished, [495](#)
- IsSamplingFinished, [496](#)
- SetAmplitude\_nA, [496](#)
- SetBufferIndex, [496](#)
- SetClampMode, [496](#)
- SetControllerParams, [497](#)
- SetCurrentEnable, [497](#)
- SetExternalLED, [497](#)
- SetLiquidResistance, [497](#)
- SetPeriod\_us, [498](#)
- SetWaveform, [498](#)
- StartInternalCalibration, [498](#)
- StartSampling, [498](#)
- StopSampling, [498](#)
- CTEERMachineDeviceNet, [499](#)
  - ~CTEERMachineDeviceNet, [499](#)
- CTEERMachineDeviceNet, [499](#)
- TEERFunctionNet, [499](#)
- CurrentRangeInNanoAmp
  - W2100\_StimulusParametersNet, [596](#)
- CurrentResolutionInNanoAmp
  - W2100\_StimulusParametersNet, [596](#)
- CUsbExceptionNet, [500](#)
  - CUsbExceptionNet, [500](#)
  - Status, [501](#)
- CutoffFrequency
  - CCreateFilterNet, [49](#)
- CVoltageRangeInfoNet
  - CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet, [501](#)
- CW2100\_FunctionNet, [501](#)
  - ClearStimulusParametersCache, [503](#)
  - ClearUserDefinedNameCache, [503](#)
  - CW2100\_FunctionNet, [503](#)
  - DeselectAllHeadstages, [504](#)
  - DeselectHeadstage, [504](#)
  - GetAccelGyroCurrentRate, [504](#)
  - GetAccelGyroDesiredRate, [504](#)
  - GetAccelGyroEnabled, [504](#)
  - GetAccelRange, [504](#)
  - GetAnalogOutChannel, [504](#)
  - GetAnalogOutFilter, [504](#)
  - GetAudioChannels, [504](#)
  - GetAvailableHeadstages, [504](#)
  - GetBatteryState, [505](#)
  - GetDacRange, [505](#)
  - GetFilterProperties, [505](#)
  - GetFilterProperty, [505](#)
  - GetFPGA FirmwareType, [505](#)
  - GetGyroRange, [505](#)
  - GetHeadstageOnOff, [505](#)
  - GetHeadstageSamplingActive, [505](#)
  - GetMultiHeadstageMode, [505](#)
  - GetPicFirmwareType, [505](#)
  - GetSelectedChannels, [506](#)
  - GetSelectedHeadstageState, [506](#)
  - GetStimulusParametersCache, [506](#)
  - GetStimulusParametersFromSelectedHS, [506](#)

- GetStimulusParameters, [506](#)
- GetUserDefinedName, [506](#)
- GetUserDefinedNameCache, [506](#)
- GetUserDefinedNameFromSelectedHS, [507](#)
- PulseGenerator, [508](#)
- SelectHeadstage, [507](#)
- SetAccelGyroDesiredRate, [507](#)
- SetAccelGyroEnabled, [507](#)
- SetAccelRange, [507](#)
- SetAnalogOutChannel, [507](#)
- SetAnalogOutFilter, [507](#)
- SetAudioChannels, [507](#)
- SetDacRange, [508](#)
- SetGyroRange, [508](#)
- SetHeadstageOnOff, [508](#)
- SetHeadstageSamplingActive, [508](#)
- SetHeadstageToSleep, [508](#)
- SetMultiHeadstageMode, [508](#)
- SetSelectedChannels, [508](#)
- Stimulator, [509](#)
- CW2100\_FunctionNet::AudioChannelsNet, [29](#)
  - amplification, [29](#)
  - channel, [29](#)
  - dacqgroup, [29](#)
- CW2100\_StimulatorFunctionNet, [509](#)
  - BOOST\_BIT, [515](#)
  - ClearChannelData, [510](#)
  - CW2100\_StimulatorFunctionNet, [510](#)
  - GetBoostAlwaysOnMode, [511](#)
  - GetBoostPreTime, [511](#)
  - GetCurrentRangeInNanoAmp, [511](#)
  - GetCurrentResolutionInNanoAmp, [511](#)
  - GetDACResolution, [511](#)
  - GetDigitalStimulatorTrigger, [511](#)
  - GetDigitalStimulatorTriggerSlope, [512](#)
  - GetNumberOfAnalogChannels, [512](#)
  - GetNumberOfSyncoutChannels, [512](#)
  - GetNumberOfTriggerInputs, [512](#)
  - GetStimulationPatternMemory, [512](#)
  - GetTimeResolutionInNanoSeconds, [512](#)
  - GetTimeSlot, [512](#)
  - GetVoltageRangeInMicroVolt, [512](#)
  - GetVoltageResolutionInMicroVolt, [513](#)
  - GND\_SWITCH\_BIT, [515](#)
  - PollStatusEvent, [515](#)
  - PrepareData, [513](#)
  - PrepareDataSync, [513](#)
  - SelectTimeSlot, [513](#)
  - SendPreparedData, [513](#)
  - SendStart, [514](#)
  - SendStop, [514](#)
  - SetBoostAlwaysOnMode, [514](#)
  - SetDigitalStimulatorTrigger, [514](#)
  - SetDigitalStimulatorTriggerSlope, [514](#)
  - StartPoll, [514](#)
  - StopPoll, [515](#)
  - SYNC\_BIT0, [515](#)
  - SYNC\_BIT1, [515](#)
- CW2100DacqGroupChannelSelectionNet, [515](#)
  - CW2100DacqGroupChannelSelectionNet, [516](#)
- CWarnerUssingDeviceNet, [516](#)
  - !CWarnerUssingDeviceNet, [517](#)
  - ~CWarnerUssingDeviceNet, [517](#)
  - CWarnerUssingDeviceNet, [517](#)
  - WarnerUssingFunction, [517](#)
- CWarnerUssingFunctionNet, [517](#)
  - !CWarnerUssingFunctionNet, [520](#)
  - ~CWarnerUssingFunctionNet, [520](#)
  - CompensateElectrodeOffset, [520](#)
  - CWarnerUssingFunctionNet, [519](#)
  - GetAvailableChambers, [520](#)
  - GetChannelsCountOfChamber, [520](#)
  - GetClampMode, [521](#)
  - GetComplianceVoltage, [521](#)
  - GetDacNampsPerDigitHighCurrentRange, [521](#)
  - GetDacPampsPerDigitLowCurrentRange, [522](#)
  - GetDacZero, [522](#)
  - GetHighCurrentRange, [522](#)
  - GetIdleModeOffset, [523](#)
  - GetLiquidResistance, [523](#)
  - GetLowCurrentRange, [523](#)
  - GetNumberOfAvailableChambers, [524](#)
  - GetNumberOfHardwareSlotsForChambers, [524](#)
  - GetU1Offset, [524](#)
  - GetU1Reference, [524](#)
  - GetU2Offset, [525](#)
  - GetU2Reference, [525](#)
  - GetUnitDescription, [525](#)
  - GetUnitExponent, [526](#)
  - GetUnitName, [526](#)
  - GetUnitsPerDigit, [526](#)
  - GetUptimeSeconds, [528](#)
  - GetVoltageClampControllerParam\_D, [528](#)
  - GetVoltageClampControllerParam\_I, [528](#)
  - GetVoltageClampControllerParam\_P, [529](#)
  - IsChamberAvailable, [529](#)
  - IsHighCurrentMode, [529](#)
  - IsInternalCalibrationFinished, [530](#)
  - IsPulseEnabled, [530](#)
  - SetClampMode, [530](#)
  - SetEnablePulse, [531](#)
  - SetHighCurrentMode, [531](#)
  - SetIdleModeOffset, [531](#)
  - SetLiquidResistance, [531](#)
  - SetLowCurrentMode, [532](#)
  - SetPulse, [532](#)
  - SetVoltageClampControllerParam\_D, [532](#)
  - SetVoltageClampControllerParam\_I, [533](#)
  - SetVoltageClampControllerParam\_P, [533](#)
  - WaitForAllChambers, [533](#)
  - WaitForChamber, [533](#)
- CWarnerValveControllerDeviceNet, [534](#)
  - !CWarnerValveControllerDeviceNet, [538](#)
  - ~CWarnerValveControllerDeviceNet, [538](#)
  - ClearTableName, [538](#)
  - ClearValveTable, [538](#)



- CWarnerValveControllerDeviceNet, 538
- GetActiveRunningTableNumber, 538
- GetActiveRunningTableNumberEvent, 554
- GetAnalogThresholdHigh, 539
- GetAnalogThresholdHighEvent, 554
- GetAnalogThresholdLow, 539
- GetAnalogThresholdLowEvent, 554
- GetAnalogVoltage, 539
- GetAnalogVoltageEvent, 554
- GetCurrentEditTableNumber, 539
- GetCurrentNumberOfValves, 540
- GetCurrentNumberOfValvesEvent, 554
- GetDigitalOutPortValve, 540
- GetDigitalOutPortValveEvent, 555
- GetDigitalPortDirection, 540
- GetDigitalPortDirectionEvent, 555
- GetDisplayMode, 540
- GetDisplayModeEvent, 555
- GetTableName, 541
- GetTableNamebyIndex, 541
- GetTableNamebyIndexEvent, 555
- GetTotalNumberOfDigitalPorts, 541
- GetTotalNumberOfTables, 541
- GetTotalNumberOfValves, 541
- GetTotalTableSize, 542
- GetValveActive, 542
- GetValveActiveEvent, 555
- GetValveBoardRevision, 542
- GetValveBoardRevisionEvent, 555
- GetValveBoardRevisionString, 542
- GetValveDigitalInPort, 542
- GetValveDigitalInPortEvent, 555
- GetValveManualGroup, 543
- GetValveManualGroupEvent, 556
- GetValveManualState, 543
- GetValveManualStateEvent, 556
- GetValveMode, 543
- GetValveModeEvent, 556
- GetValvesActiveMap, 544
- GetValvesManualStateMap, 544
- GetValveTableEntry, 544
- IsDigitalOutPortInverted, 544
- IsDigitalOutPortInvertedEvent, 556
- IsValveDigitalInInverted, 545
- IsValveDigitalInInvertedEvent, 556
- IsValveOpen, 545
- IsValveOpenEvent, 556
- IsValveOpenInAnalogMode, 545
- IsValveOpenInAnalogModeEvent, 556
- IsValveOpenInDigitalMode, 546
- IsValveOpenInDigitalModeEvent, 557
- LoadValveTable, 546
- OnGetActiveRunningTableNumber, 546
- OnGetAnalogThresholdHigh, 546
- OnGetAnalogThresholdLow, 546
- OnGetAnalogVoltage, 546
- OnGetCurrentNumberOfValves, 547
- OnGetDigitalOutPortValve, 547
- OnGetDigitalPortDirection, 547
- OnGetDisplayMode, 547
- OnGetTableNamebyIndex, 547
- OnGetValveActive, 547
- OnGetValveBoardRevision, 547
- OnGetValveDigitalInPort, 547
- OnGetValveManualGroup, 547
- OnGetValveManualState, 548
- OnGetValveMode, 548
- OnIsDigitalOutPortInverted, 548
- OnIsValveDigitalInInverted, 548
- OnIsValveOpen, 548
- OnIsValveOpenInAnalogMode, 548
- OnIsValveOpenInDigitalMode, 548
- OnTableEntryChanged, 548
- SetActiveRunningTableNumber, 548
- SetAnalogThresholdHigh, 549
- SetAnalogThresholdLow, 549
- SetCurrentEditTableNumber, 549
- SetDefault, 549
- SetDigitalOutPortInvert, 549
- SetDigitalOutPortValve, 550
- SetDigitalPortDirection, 550
- SetDisplayMode, 550
- SetTableName, 551
- SetTableStep, 551
- SetTableStepAll, 551
- SetValveActive, 551
- SetValveDigitalInInvert, 552
- SetValveDigitalInPort, 552
- SetValveManualGroup, 552
- SetValveManualState, 552
- SetValveMode, 553
- SetValvesActiveMap, 553
- SetValvesManualStateMap, 553
- SetValveTableEntry, 553
- StoreValveTable, 554
- TableEntryChangedEvent, 557
- CWarnerValveControllerDeviceTesterFunctionNet, 557
  - !CWarnerValveControllerDeviceTesterFunctionNet, 558
  - ~CWarnerValveControllerDeviceTesterFunctionNet, 558
  - CWarnerValveControllerDeviceTesterFunctionNet, 558
  - GetIO, 558
  - GetSync, 558
  - SetADC, 559
  - SetIO, 559
  - SetIODirection, 559
  - SetTrigger, 559
  - SetTriggerSyncDirection, 560
- CWClassicFunctionNet, 560
  - CWClassicFunctionNet, 561
  - GetFilterParametersHeadstage, 561
  - GetHasChecksum, 561
  - GetHasRedLedHeadstage, 561
  - GetHeadstageOnOff, 561

- GetResetFilter, [562](#)
- GetRFConnectionStatus, [562](#)
- GetRFFrequencyHeadstage, [562](#)
- GetRFFrequencyReceiver, [562](#)
- GetRFPower, [562](#)
- GetScanHeadstagesResult, [562](#)
- GetSelectedHeadstage, [562](#)
- GetSerialNumberHeadstage, [562](#)
- GetWPADebugMode, [562](#)
- GetWPAType, [562](#)
- ResetChannelmap, [563](#)
- ScanForHeadstages, [563](#)
- SetChannelmap, [563](#)
- SetFilterParametersHeadstage, [563](#)
- SetHasChecksum, [563](#)
- SetHeadstageOnOff, [563](#)
- SetHWSelectedChannels, [563](#)
- SetResetFilter, [563](#)
- SetRFFrequencyHeadstage, [563](#)
- SetRFFrequencyReceiver, [564](#)
- SetRFFrequencyReceiverEeprom, [564](#)
- SetRFLostBehaviour, [564](#)
- SetRFPower, [564](#)
- SetSelectedHeadstage, [564](#)
- SetSerialNumberHeadstage, [564](#)
- SetWPADebugMode, [564](#)
- SetWPAType, [564](#)
- CWirelessBaseFunctionNet, [565](#)
  - CreateWirelessHeadstageSerialNumberString, [565](#)
  - CWirelessBaseFunctionNet, [565](#)
- DacqDevice
  - CSafeISDeviceNet, [397](#)
- dacqgroup
  - CW2100\_FunctionNet::AudioChannelsNet, [29](#)
- DACResolution
  - W2100\_StimulusParametersNet, [596](#)
- DataState
  - HeadStageIDTypeState, [581](#)
- DeepCopy
  - CCMOSMeaDeviceNet::CRegionOfInterestRect, [345](#)
- DefineAmplification
  - CPgaDeviceNet, [317](#)
- DefineFrequencyRange
  - CPgaDeviceNet, [317](#)
- DefineNumAmplifications
  - CPgaDeviceNet, [317](#)
- DefineNumFrequencyRanges
  - CPgaDeviceNet, [317](#)
- DeselectAllHeadstages
  - CW2100\_FunctionNet, [504](#)
- DeselectHeadstage
  - CW2100\_FunctionNet, [504](#)
- DetectChipType
  - CCMOSMea\_FunctionNet, [36](#)
- DEVICE\_NOT\_FOUND
  - Mcs::Usb, [27](#)
- DeviceArrival
  - CMcsUsbListNet, [225](#)
- DeviceData
  - CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, [595](#)
- DeviceDataLength
  - CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, [595](#)
- DeviceId
  - CMcsUsbListEntryNet, [222](#)
- DeviceIdNet, [565](#)
  - BcdDevice, [567](#)
  - BusType, [567](#)
  - DeviceIdNet, [566](#)
  - IdProduct, [567](#)
  - IdVendor, [567](#)
  - operator=, [566](#)
- DeviceName
  - CMcsUsbListEntryNet, [222](#)
- DeviceRemoval
  - CMcsUsbListNet, [225](#)
- DigitalSource
  - DigitalSource< digitalsourceenum >, [567](#)
- DigitalSource< digitalsourceenum >, [567](#)
  - DigitalSource, [567](#)
  - MaxBitNumber, [567](#)
  - size, [568](#)
  - Source, [568](#)
- DisableAutoReset
  - CStg200xDownloadBasicNet, [448](#)
- DisableMultiFileMode
  - CStg200xDownloadNet, [457](#)
- Disconnect
  - CMcsUsbNet, [232](#)
- DisConnectDevice
  - CRadioControlledDevicesNet, [344](#)
- DoRamp
  - CRoboDacqNet, [359](#)
- DownloadFirmware
  - CMcsUsbFactoryNet, [211](#)
- DriverVersionNet, [568](#)
  - ~DriverVersionNet, [569](#)
  - DriverVersionNet, [569](#)
  - DriverVersionNet::FormatVersion, [569](#)
  - GetDestinationCode, [569](#)
  - GetDestinationName, [569](#), [571](#)
  - GetMajor, [571](#)
  - GetMinor, [571](#), [572](#)
  - GetNumEntries, [572](#)
  - GetStatus, [572](#)
  - GetVersionInt, [572](#), [573](#)
  - GetVersionString, [573](#)
- DriverVersionNet::FormatVersion
  - DriverVersionNet, [569](#)
- DSP
  - FirmwareDestinationNames, [574](#)
- DummyCommand
  - CLIH3DeviceNet, [111](#)

- Duration
  - CStimulusFunctionNet::SidebandData, [593](#)
- ElectricalStimulation
  - HeadStageIDType, [578](#)
- EmptyKey
  - CMcsUsbNet, [232](#)
- Emu\_GetCellCapacity
  - CRoboDacqNet, [359](#)
- Emu\_GetCellPotential
  - CRoboDacqNet, [359](#)
- Emu\_GetCellResists
  - CRoboDacqNet, [359](#)
- Emu\_GetElectrodeResists
  - CRoboDacqNet, [359](#)
- Emu\_GetNoise
  - CRoboDacqNet, [359](#)
- Emu\_SetCellCapacity
  - CRoboDacqNet, [359](#)
- Emu\_SetCellPotential
  - CRoboDacqNet, [359](#)
- Emu\_SetCellResists
  - CRoboDacqNet, [359](#)
- Emu\_SetElectrodeResists
  - CRoboDacqNet, [360](#)
- Emu\_SetNoise
  - CRoboDacqNet, [360](#)
- EnableAnalogOut
  - CSCUFunctionNet, [402](#)
- EnableAutoReset
  - CStg200xDownloadBasicNet, [449](#)
- EnableChannelsInGroup
  - CCMOSMea\_FunctionNet, [36](#)
  - CDacqGroupChannelSelectionTemplateNet< Dac-  
qGroupChannelEnumTemplateNet, Dac-  
qGroupChannelEnumTemplate, CDevice-  
GroupChannelInfoTemplateNet >, [53](#)
- EnableChecksum
  - CMeaDeviceNet, [264](#)
  - COctoPotDeviceNet, [306](#)
- EnableDigitalIn
  - CMeaDeviceNet, [264](#), [265](#)
  - COctoPotDeviceNet, [306](#)
- EnableExceptions
  - CMcsUsbNet, [232](#)
- EnableMultiFileMode
  - CStg200xDownloadNet, [457](#)
- EnableTimestamp
  - CMeaDeviceNet, [265](#)
  - COctoPotDeviceNet, [306](#)
- EnableUserTrigger
  - CLIH3DeviceNet, [112](#)
- enCMosMeaChipType
  - Mcs::Usb, [26](#)
- EnSTG200x\_STATUS
  - Mcs::Usb, [27](#)
- Entry
  - HeadStageIDType, [578](#)
- Equals
  - CFilterCoefficientsNet, [66](#)
  - CMcsUsbListEntryNet, [220](#)
  - HeadStageIDType, [578](#)
  - HeadstageIDTypeObject, [580](#)
  - EraseEepromRegisterPreconfig
    - CMcsUsbNet, [232](#)
  - EraseFilterParameterPermanent
    - CFilterConfigurationNet, [69](#)
    - CFilterConfigurationRegisterNet, [70](#)
  - Error\_Callback\_Aquisition\_Stopped
    - CMcsUsbDacqNet, [206](#)
  - Error\_Callback\_Data\_lost
    - CMcsUsbDacqNet, [206](#)
  - Error\_Callback\_Frames\_Lost
    - CMcsUsbDacqNet, [206](#)
  - Error\_Callback\_Packet\_Error
    - CMcsUsbDacqNet, [206](#)
  - Error\_Callback\_Queue\_Full
    - CMcsUsbDacqNet, [206](#)
  - Error\_Callback\_RingQueue\_Full
    - CMcsUsbDacqNet, [206](#)
  - ErrorEvent
    - CMcsUsbDacqNet, [207](#)
  - FactoryReset
    - CTcxDeviceNet, [478](#)
  - FeedbackGetSampleTimerCount
    - CMeaFeedbackFunctionNet, [273](#)
  - FeedbackSetAnalogSource
    - CMeaFeedbackFunctionNet, [273](#)
  - FeedbackSetChannelFilter
    - CMeaFeedbackFunctionNet, [273](#)
  - FeedbackSetDigitalMapping
    - CMeaFeedbackFunctionNet, [273](#)
  - FeedbackSetFeedback
    - CMeaFeedbackFunctionNet, [273](#)
  - FeedbackSetFilterOff
    - CMeaFeedbackFunctionNet, [274](#)
  - FeedbackSetFilterParameter
    - CMeaFeedbackFunctionNet, [274](#)
  - FeedbackSetFilterParameter32
    - CMeaFeedbackFunctionNet, [274](#)
  - FeedbackSetGlobalChannelFilter
    - CMeaFeedbackFunctionNet, [274](#)
  - FeedbackSetIIRFilterParameter
    - CMeaFeedbackFunctionNet, [274](#)
  - FeedbackSetLogic
    - CMeaFeedbackFunctionNet, [274](#)
  - FeedbackSetMkFilter
    - CMeaFeedbackFunctionNet, [274](#)
  - FeedbackSetNumberOfLogics
    - CMeaFeedbackFunctionNet, [275](#)
  - FeedbackSetNumberOfRateCounter
    - CMeaFeedbackFunctionNet, [275](#)
  - FeedbackSetNumberOfRateDetectors
    - CMeaFeedbackFunctionNet, [275](#)
  - FeedbackSetNumberOfSpikeDetectors
    - CMeaFeedbackFunctionNet, [275](#)
  - FeedbackSetNumberOfTriggers

- CMeaFeedbackFunctionNet, [275](#)
- FeedbackSetRateCounter
  - CMeaFeedbackFunctionNet, [275](#)
- FeedbackSetRateDetector
  - CMeaFeedbackFunctionNet, [275](#)
- FeedbackSetSpikeDetectorThreshold
  - CMeaFeedbackFunctionNet, [275](#)
- FeedbackSetTrigger
  - CMeaFeedbackFunctionNet, [276](#)
- FilterActive
  - CFilterPropertyNet, [72](#)
- FilterBand
  - CFilterPropertyNet, [73](#)
- FilterFamily
  - CFilterPropertyNet, [73](#)
- FilterType
  - CFilterPropertyNet, [73](#)
- FindFilter
  - CCreateFilterNet, [48](#)
- FindFirmwareVersionMagicInBuffer
  - CMcsUsbFactoryNet, [211](#)
- FindReference
  - CRoboDeviceNet, [373](#)
- FindReferencel
  - CRoboStatorDeviceNet, [390](#)
- FindReferencePhase0
  - CRoboDeviceNet::RoboMainLowLevelCommands, [586](#)
- FindReferencePhase0XY
  - CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands, [592](#)
- FindReferenceXY
  - CRoboStatorDeviceNet, [390](#), [391](#)
- FindReferenceZ
  - CRoboStatorDeviceNet, [391](#)
- FirePressurePulse
  - CPPCFunctionNet, [327](#)
- FirmwareDestinationNames, [573](#)
  - Altera, [574](#)
  - Bootstrap, [574](#)
  - BUS1\_MCSBUS1, [574](#)
  - BUS1\_MCSBUS2, [574](#)
  - DSP, [574](#)
  - FPGA2, [574](#)
  - FPGA3, [575](#)
  - FPGA4, [575](#)
  - FPGA5, [575](#)
  - FPGA6, [575](#)
  - MCSBUS1, [575](#)
  - MCSBUS10, [575](#)
  - MCSBUS11, [575](#)
  - MCSBUS12, [575](#)
  - MCSBUS13, [575](#)
  - MCSBUS2, [575](#)
  - MCSBUS3, [575](#)
  - MCSBUS4, [576](#)
  - MCSBUS5, [576](#)
  - MCSBUS6, [576](#)
  - MCSBUS7, [576](#)
  - MCSBUS8, [576](#)
  - MCSBUS9, [576](#)
  - MCU1, [576](#)
  - PIC, [576](#)
  - PIC2, [576](#)
  - PIC3, [576](#)
  - PIC4, [576](#)
  - USB, [577](#)
- FluidControlDevice
  - CSafeISDeviceNet, [398](#)
- ForceStatusEvent
  - CStg200xDownloadBasicNet, [449](#)
  - CStimulusFunctionNet, [465](#)
- FPGA2
  - FirmwareDestinationNames, [574](#)
- FPGA3
  - FirmwareDestinationNames, [575](#)
- FPGA4
  - FirmwareDestinationNames, [575](#)
- FPGA5
  - FirmwareDestinationNames, [575](#)
- FPGA6
  - FirmwareDestinationNames, [575](#)
- FromIntPtr
  - StgStatusNet, [594](#)
- FromPtr
  - StgStatusNet, [594](#)
- FX3MCSDataAddress
  - CMcsUsbFactoryNet, [217](#)
- FX3MCSDataDeviceIdOffset
  - CMcsUsbFactoryNet, [217](#)
- FX3MCSDataVersionOffset
  - CMcsUsbFactoryNet, [217](#)
- FYIProgram
  - CFYIDeviceNet, [81](#)
- FYITemp
  - CFYIDeviceNet, [81](#)
- Gain
  - CMeaDeviceNet, [269](#)
- Get2AnalogInput
  - CMcsBus\_SensorNet, [142](#)
- Get2DigitalInput
  - CMcsBus\_SensorNet, [142](#)
- Get4ADC
  - CMcsBus\_SensorNet, [142](#)
- Get4ADCAverage
  - CMcsBus\_SensorNet, [142](#)
- Get4ADCCatchampAverageShift
  - CMcsBus\_SensorNet, [142](#)
- Get4ADCMode
  - CMcsBus\_SensorNet, [142](#)
- Get4DAC
  - CMcsBus\_SensorNet, [142](#)
- GetAbsMaxCurrentInMicroAmp
  - CMultiwellOptoStimFunctionNet, [300](#)
- GetAccelGyroCurrentRate
  - CW2100\_FunctionNet, [504](#)

- GetAccelGyroDesiredRate
  - CW2100\_FunctionNet, [504](#)
- GetAccelGyroEnabled
  - CW2100\_FunctionNet, [504](#)
- GetAccelRange
  - CW2100\_FunctionNet, [504](#)
- GetActiveRunningTableNumber
  - CWarnerValveControllerDeviceNet, [538](#)
- GetActiveRunningTableNumberEvent
  - CWarnerValveControllerDeviceNet, [554](#)
- GetAdapterCode
  - CMealImpedanceDeviceNet, [277](#)
  - CTEERFunctionNet, [491](#)
- GetAdapterType
  - CMcsUsbDacqNet, [182](#)
- GetAdc
  - CFluidControlDeviceNet, [76](#)
- GetAdcDataFormat
  - CMcsUsbDacqNet, [182](#)
- GetADCInputOffset
  - CCMOSMea\_FunctionNet, [36](#)
- GetADCOffset
  - CLIH3DeviceNet, [112](#)
- GetAdcOffset
  - COctoPotDeviceNet, [306](#)
- GetAdcOffsetU1
  - CTEERFunctionNet, [492](#)
- GetAdcOffsetU2
  - CTEERFunctionNet, [492](#)
- GetADCs
  - CMcsBus\_SensorNet, [142](#)
- GetADCsLoop
  - CMcsBus\_SensorNet, [143](#)
- GetAdcZero
  - CMcsUsbDacqNet, [182](#)
- GetAirpressure
  - CRoboDeviceNet, [373](#)
- GetAirpressureLimit
  - CRoboDeviceNet, [373](#)
- GetAirValve
  - CRoboDeviceNet, [373](#)
- GetAllDigout
  - CRoboDacqNet, [360](#)
- GetAmplification
  - CPgaDeviceNet, [317](#)
- GetAmplitude\_nA
  - CTEERFunctionNet, [492](#)
- GetAnalogGain
  - CMeaDeviceNet, [266](#)
- GetAnalogOutADCRange
  - CSCUFunctionNet, [402](#)
- GetAnalogOutChannel
  - CW2100\_FunctionNet, [504](#)
- GetAnalogOutChannels
  - CSCUFunctionNet, [402](#)
- GetAnalogOutDACRange
  - CSCUFunctionNet, [402](#)
- GetAnalogOutFilter
  - CW2100\_FunctionNet, [504](#)
- GetAnalogRanges
  - CStg200xBasicNet, [420](#)
- GetAnalogResolution
  - CStg200xBasicNet, [420](#)
- GetAnalogThresholdHigh
  - CWarnerValveControllerDeviceNet, [539](#)
- GetAnalogThresholdHighEvent
  - CWarnerValveControllerDeviceNet, [554](#)
- GetAnalogThresholdLow
  - CWarnerValveControllerDeviceNet, [539](#)
- GetAnalogThresholdLowEvent
  - CWarnerValveControllerDeviceNet, [554](#)
- GetAnalogValueUnit
  - CMcsUsbDacqNet, [182](#)
- GetAnalogVoltage
  - CPPCFunctionNet, [328](#)
  - CPPS\_FunctionNet, [335](#)
  - CWarnerValveControllerDeviceNet, [539](#)
- GetAnalogVoltageEvent
  - CWarnerValveControllerDeviceNet, [554](#)
- GetAnalogVoltageRange
  - CPPCFunctionNet, [328](#)
- GetAnalogVoltages
  - CPPS\_FunctionNet, [335](#)
- GetArraySize
  - CMealImpedanceDeviceNet, [277](#)
- GetAudioChannels
  - CMeaAudioFunctionNet, [250](#), [251](#)
  - CW2100\_FunctionNet, [504](#)
- GetAutocalibrationDisabled
  - CStg200xBasicNet, [421](#)
- GetAvailableBaseSamplerates
  - CCMOSMeaDeviceNet, [45](#)
- GetAvailableChambers
  - CWarnerUssingFunctionNet, [520](#)
- GetAvailableDeviceList
  - CRFFFunctionNet, [349](#)
- GetAvailableDeviceListEx
  - CRFFFunctionNet, [349](#)
- GetAvailableHeadstages
  - CSCUFunctionNet, [402](#)
  - CW2100\_FunctionNet, [504](#)
- GetAvailableHeadstagesEvent
  - CSCUFunctionNet, [414](#)
- GetAvailableMemory
  - CStg200xBasicNet, [421](#)
  - CStimulusFunctionNet, [465](#)
- GetAvailableSampleRates
  - CMcsUsbDacqNet::CHWInfo, [104](#)
- GetAvailableStateList
  - CRFFFunctionNet, [350](#)
- GetAvailableStateListEx
  - CRFFFunctionNet, [350](#)
- GetAvailableVoltageRangesInMicroVolt
  - CMcsUsbDacqNet::CHWInfo, [104](#)
- GetAvailableVoltageRangesInMicroVoltAndStringsInMilliVolt
  - CMcsUsbDacqNet::CHWInfo, [104](#)

- GetAxisConfig
  - CRoboDeviceNet::RoboMainLowLevelCommands, [586](#)
- GetAxisLED
  - CRoboocyte2DeviceNet, [388](#)
- GetAxisParametersSignedEeprom
  - CMcsBus\_AxisParametersNet, [117](#)
- GetAxisParametersUnsignedEeprom
  - CMcsBus\_AxisParametersNet, [117](#)
- GetBaseFrequency
  - CRFFunctionNet, [350](#)
- GetBaseSamplerate
  - CCMOSMeaDeviceNet, [46](#)
- GetBath
  - CCMOSMea\_FunctionNet, [36](#)
- GetBathMode
  - CCMOSMea\_FunctionNet, [36](#)
- GetBatteryState
  - CW2100\_FunctionNet, [505](#)
- GetBatteryVoltage
  - CMultiBatteryChargerDeviceNet, [285](#)
- GetBiQuad
  - CCreateFilterNet, [48](#)
- GetBiQuads
  - CCreateFilterNet, [48](#)
- GetBlankingEnable
  - CStg200xBasicNet, [421](#), [422](#)
- GetBoardTemp
  - CTcxDeviceNet, [479](#)
- GetBoostAlwaysOnMode
  - CW2100\_StimulatorFunctionNet, [511](#)
- GetBoostPreTime
  - CW2100\_StimulatorFunctionNet, [511](#)
- GetBubbleState
  - CPPS\_FunctionNet, [335](#)
- GetBubbleStatus
  - CMcsBus\_SensorNet, [143](#)
- GetBuffer
  - CGenericDevelopDeviceNet, [88](#)
- GetBusAddress
  - CMcsBusNet, [156](#)
- GetBusAddressEeprom
  - CMcsBusNet, [156](#)
- GetByteBuffer
  - CGenericDevelopDeviceNet, [89](#)
- GetBytesAvailable
  - CSerialPortNet, [415](#)
- GetBytesPerSample
  - CTEERFunctionNet, [492](#)
- GetCalibration
  - CTcxDeviceNet, [479](#)
- GetCalibrationDecp
  - CTcxDeviceNet, [479](#)
- GetCalibrationMax
  - CTcxDeviceNet, [479](#)
- GetCalibrationMin
  - CTcxDeviceNet, [479](#)
- GetCapacityC
  - CRoboDacqNet, [360](#)
- GetCapacityV
  - CRoboDacqNet, [360](#)
- GetCapacityX
  - CRoboDacqNet, [360](#)
- GetCardinalDacqSamplerate
  - CInterfaceboardFunctionNet, [109](#)
- GetCardinalStgOutputrate
  - CInterfaceboardFunctionNet, [109](#)
- GetChannel
  - CSw2to64DeviceNet, [474](#)
- GetChannelDataFillSize
  - CMcsUsbDacqNet, [182](#)
- GetChannelDataI16
  - CCMOSMeaDeviceNet, [46](#)
- GetChannelDataI32
  - CCMOSMeaDeviceNet, [46](#)
- GetChannelDataUI16
  - CCMOSMeaDeviceNet, [46](#)
- GetChannelDataUI32
  - CCMOSMeaDeviceNet, [46](#)
- GetChannelLayout
  - CMcsUsbDacqNet, [182](#)
- GetChannels
  - CMultiBatteryChargerDeviceNet, [285](#)
  - CSw2to64DeviceNet, [475](#)
- GetChannelsCountOfChamber
  - CWarnerUssingFunctionNet, [520](#)
- GetChannelsInBlock
  - CMcsUsbDacqNet, [183](#)
- GetChannelState
  - CMultiBatteryChargerDeviceNet, [285](#)
- GetChargeCapacity
  - CMultiBatteryChargerDeviceNet, [286](#)
- GetChargeCurrent
  - CMultiBatteryChargerDeviceNet, [286](#)
- GetChargingMode
  - CMultiBatteryChargerDeviceNet, [286](#)
- GetChargingPCoefficient
  - CMultiBatteryChargerDeviceNet, [286](#)
- GetChecksumFromFX3Image
  - CMcsUsbFactoryNet, [212](#)
- GetCheckVoltage
  - CokuvisionStimulatorDeviceNet, [309](#)
- GetClampAmpSerialNumber
  - CRoboDacqNet, [360](#)
- GetClampMode
  - CTEERFunctionNet, [492](#)
  - CWarnerUssingFunctionNet, [521](#)
- GetCMOSDataDictionary
  - CCMOSMeaDeviceNet, [46](#)
- GetCoilCommunication
  - CPositionIIDeviceNet, [319](#)
- GetColorRgb
  - CMultiwellOptoStimFunctionNet, [300](#)
- GetColorStr
  - CMultiwellOptoStimFunctionNet, [301](#)
- GetCommand



- CMcsBusNet, [156](#), [157](#)
- CPedoterDeviceNet, [314](#)
- CRoboDacqNet, [360](#)
- GetComplianceVoltage
  - CWarnerUssingFunctionNet, [521](#)
- GetConfiguration
  - CMcsUsbNet, [233](#)
- GetConfigurationBit
  - CRoboDacqNet, [360](#)
- GetConfigurationBitAxc
  - CRoboDacqNet, [360](#)
- GetConfigurationBitBlu\_Led
  - CRoboDacqNet, [361](#)
- GetConfigurationBitBlu\_LedToggleFast
  - CRoboDacqNet, [361](#)
- GetConfigurationBitBlu\_LedToggleSlow
  - CRoboDacqNet, [361](#)
- GetConfigurationBitCC\_Gen
  - CRoboDacqNet, [361](#)
- GetConfigurationBitCV\_Gen
  - CRoboDacqNet, [361](#)
- GetConfigurationBitRC\_Gen
  - CRoboDacqNet, [361](#)
- GetConfigurationBitRed\_Led
  - CRoboDacqNet, [361](#)
- GetConfigurationBitRed\_LedSaturation
  - CRoboDacqNet, [361](#)
- GetConfigurationBitRed\_LedToggleFast
  - CRoboDacqNet, [361](#)
- GetConfigurationBitRed\_LedToggleSlow
  - CRoboDacqNet, [361](#)
- GetConfigurationBitRelais
  - CRoboDacqNet, [361](#)
- GetConfigurationBitRV\_Gen
  - CRoboDacqNet, [362](#)
- GetConfigurationBits
  - CRoboDacqNet, [362](#)
- GetConfigurationBitStream
  - CRoboDacqNet, [362](#)
- GetConfigurationBitSupply
  - CRoboDacqNet, [362](#)
- GetConnectedDevice
  - CRFFFunctionNet, [351](#)
- GetControllerParams
  - CTEERFunctionNet, [492](#)
- GetCrossTalkOffset
  - CRoboDacqNet, [362](#)
- GetCrossTalkOptimum
  - CRoboDacqNet, [362](#)
- GetCurrent
  - CTcxDeviceNet, [479](#)
- GetCurrentAirvalve
  - CRoboDeviceNet, [373](#)
- GetCurrentAirvalveLimit
  - CRoboDeviceNet, [374](#)
- GetCurrentCycle
  - CMeaCoatDeviceNet, [258](#)
- GetCurrentEditTableNumber
  - CWarnerValveControllerDeviceNet, [539](#)
- GetCurrentEnable
  - CTEERFunctionNet, [493](#)
- GetCurrentFactor
  - COkuvisionStimulatorDeviceNet, [309](#)
- GetCurrentNumberOfValves
  - CWarnerValveControllerDeviceNet, [540](#)
- GetCurrentNumberOfValvesEvent
  - CWarnerValveControllerDeviceNet, [554](#)
- GetCurrentPosition
  - CRoboDeviceNet, [374](#)
- GetCurrentPositionI
  - CRoboStatorDeviceNet, [391](#)
- GetCurrentPositionXY
  - CRoboStatorDeviceNet, [391](#)
- GetCurrentPositionZ
  - CRoboStatorDeviceNet, [391](#)
- GetCurrentRangeInNanoAmp
  - CStg200xBasicNet, [422](#)
  - CStimulusFunctionNet, [466](#)
  - CW2100\_StimulatorFunctionNet, [511](#)
- GetCurrentResolutionInNanoAmp
  - CStg200xBasicNet, [422](#)
  - CStimulusFunctionNet, [466](#)
  - CW2100\_StimulatorFunctionNet, [511](#)
- GetCycle
  - CMeaCleanDeviceNet, [254](#)
- GetCycles
  - CMeaCleanDeviceNet, [254](#)
  - CMeaCoatDeviceNet, [258](#)
- GetD
  - CTcxDeviceNet, [479](#)
- GetDacAmplificationFactor
  - CStg200xBasicNet, [423](#)
- GetDacIdleValue
  - CLIH3DeviceNet, [112](#)
- GetDacNampsPerDigitHighCurrentRange
  - CWarnerUssingFunctionNet, [521](#)
- GetDACOffset
  - COkuvisionStimulatorDeviceNet, [310](#)
- GetDacOffset
  - CDacCalibrationFunctionNet, [50](#)
  - COctoPotDeviceNet, [306](#)
- GetDacPampsPerDigitLowCurrentRange
  - CWarnerUssingFunctionNet, [522](#)
- GetDacqRunStatus
  - CLIH3DeviceNet, [112](#)
- GetDacRange
  - CW2100\_FunctionNet, [505](#)
- GetDACResolution
  - CStg200xBasicNet, [423](#)
  - CStimulusFunctionNet, [466](#)
  - CW2100\_StimulatorFunctionNet, [511](#)
- GetDACs
  - CMcsBus\_SensorNet, [143](#)
- GetDacUseldleValue
  - CLIH3DeviceNet, [113](#)
- GetDacZero

- CTEERFunctionNet, 493
- CWarnerUssingFunctionNet, 522
- GetDataFormat
  - CMcsUsbDacqNet, 183
- GetDataMode
  - CMcsUsbDacqNet, 183
- GetDDecp
  - CTcxDeviceNet, 479
- GetDestination
  - CMcsUsbFactoryNet, 212
- GetDestinationCode
  - DriverVersionNet, 569
- GetDestinationDisplayLabel
  - CMcsUsbFactoryNet, 212
- GetDestinationName
  - CMcsUsbFactoryNet, 212
  - DriverVersionNet, 569, 571
- GetDestinationSerialNumber
  - CMcsUsbFactoryNet, 212
- GetDestinationTargetAddress
  - CMcsUsbFactoryNet, 212
- GetDetectionThreshold
  - CMcsBus\_SensorNet, 143
- GetDetectorValue
  - CMcsBus\_SensorNet, 143
- GetDevice
  - CTcxDeviceNet, 479
- GetDeviceCannotStallOutRequests
  - CMcsUsbNet, 233
- GetDeviceCapableSpeed
  - CMcsUsbNet, 233
- GetDeviceEnum
  - CMcsUsbNet, 233
- GetDeviceGroupChannelInfos
  - CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet >, 53
- GetDeviceId
  - CMcsUsbNet, 233
- GetDeviceList
  - CPositionImpDeviceNet, 323
- GetDeviceNames
  - CRadioControlledDevicesNet, 344
- GetDeviceRootHubVendorEnum
  - CMcsUsbNet, 233
- GetDeviceRootHubVendorID
  - CMcsUsbNet, 233
- GetDeviceRootHubVendorName
  - CMcsUsbNet, 233
- GetDeviceSpeed
  - CMcsUsbNet, 234
- GetDeviceType
  - CTcxDeviceNet, 480
- GetDevname
  - CTcxDeviceNet, 480
- GetDigin
  - CFluidControlDeviceNet, 76
- GetDiginState
  - CLIH3DeviceNet, 113
- GetDiginValue
  - CStg200xBasicNet, 423
- GetDigitalData
  - CMeaDigitalDataFunctionNet, 271
- GetDigitalIn
  - CPPCFunctionNet, 328
  - CPPS\_FunctionNet, 335
- GetDigitalOutPortValve
  - CWarnerValveControllerDeviceNet, 540
- GetDigitalOutPortValveEvent
  - CWarnerValveControllerDeviceNet, 555
- GetDigitalPortDirection
  - CWarnerValveControllerDeviceNet, 540
- GetDigitalPortDirectionEvent
  - CWarnerValveControllerDeviceNet, 555
- GetDigitalSource
  - CMcsUsbDacqNet, 183–185
- GetDigitalStimulatorTrigger
  - CW2100\_StimulatorFunctionNet, 511
- GetDigitalStimulatorTriggerSlope
  - CW2100\_StimulatorFunctionNet, 512
- GetDigout
  - CFluidControlDeviceNet, 76
  - CRoboDacqNet, 362
- GetDigoutMode
  - CStg200xBasicNet, 423
- GetDigoutValue
  - CStg200xBasicNet, 424
- GetDIO
  - CMcsBus\_FYIExtensionNet, 120
- GetDischargeCapacity
  - CMultiBatteryChargerDeviceNet, 287
- GetDischargeCurrent
  - CMultiBatteryChargerDeviceNet, 287
- GetDischargeCurrentSetPoint
  - CMultiBatteryChargerDeviceNet, 287
- GetDisplayMode
  - CWarnerValveControllerDeviceNet, 540
- GetDisplayModeEvent
  - CWarnerValveControllerDeviceNet, 555
- GetDisplayText
  - CRoboDacqNet, 362
- GetDMax
  - CTcxDeviceNet, 480
- GetDMin
  - CTcxDeviceNet, 480
- GetDownsampleFactor
  - CRoboDacqNet, 362
- GetDSPHighPassByIndex
  - CIntanMea\_FunctionNet, 106
- GetDuration
  - CMeaCoatDeviceNet, 258
- GetDuty
  - CTcxDeviceNet, 480
- GetEEPromPage
  - CLIH3DeviceNet, 113



- GetElectrodeDacMux
  - CStg200xBasicNet, [424](#)
- GetElectrodeEnable
  - CStg200xBasicNet, [425](#)
- GetElectrodeMode
  - CStg200xBasicNet, [426](#)
- GetEnableAmplifierProtectionSwitch
  - CStg200xBasicNet, [426](#), [427](#)
- GetEnabledChannelsInGroup
  - CCMOSMea\_FunctionNet, [37](#)
  - CDacqGroupChannelSelectionTemplateNet< Dac-  
qGroupChannelEnumTemplateNet, Dac-  
qGroupChannelEnumTemplate, CDevice-  
GroupChannelInfoTemplateNet >, [53](#), [54](#)
- GetEnableHeaterLimit
  - CTcxDeviceNet, [480](#)
- GetEnableThermocouple
  - CTcxDeviceNet, [480](#)
- GetEntry
  - CMcsUsbListEntryNet, [220](#), [221](#)
- GetEntryCount
  - CMcsUsbListEntryNet, [221](#)
- GetEnumerationSpeed
  - CMeaDeviceNet, [266](#)
- GetErrorAirpressure
  - CRoboDeviceNet, [374](#)
- GetErrorCurrentAirvalve
  - CRoboDeviceNet, [374](#)
- GetErrorText
  - CMcsUsbNet, [234](#)
- GetErrorVoltage12V
  - CRoboDeviceNet, [374](#)
- GetErrorVoltage5V
  - CRoboDeviceNet, [374](#)
- GetErrorVoltageAirvalve
  - CRoboDeviceNet, [374](#)
- GetErrorVoltageRs485A
  - CRoboDeviceNet, [374](#)
- GetErrorVoltageRs485B
  - CRoboDeviceNet, [375](#)
- GetErrorVoltageValves
  - CRoboDeviceNet, [375](#)
- GetExternalElectrodeEnable
  - CStg200xBasicNet, [427](#)
- GetFAAmplification
  - CStg200xBasicNet, [428](#)
- GetFilter
  - CRoboDacqNet, [362](#)
- GetFilterCoeffs
  - CRoboDacqNet, [362](#)
- GetFilterParametersHeadstage
  - CWClassicFunctionNet, [561](#)
- GetFilterProperties
  - CSCUFunctionNet, [403](#)
  - CW2100\_FunctionNet, [505](#)
- GetFilterProperty
  - CMcsUsbDacqNet, [185](#)
  - CSCUFunctionNet, [403](#)
  - CW2100\_FunctionNet, [505](#)
- GetFinalDischargeVoltage
  - CMultiBatteryChargerDeviceNet, [288](#)
- GetFirmwareVersion
  - CMcsUsbNet, [234](#)
- GetFirmwareVersionFromFile
  - CMcsUsbFactoryNet, [213](#)
- GetFirmwareVersionFromHexFile
  - CMcsUsbFactoryNet, [213](#)
- GetFPGA FirmwareType
  - CW2100\_FunctionNet, [505](#)
- GetFrequency
  - CRadioControlledDevicesNet, [344](#)
- GetFrequencyRange
  - CPgaDeviceNet, [318](#)
- GetGain
  - CMeaDeviceNet, [266](#)
  - CPgaDeviceNet, [318](#)
- GetGate
  - CCMOSMea\_FunctionNet, [37](#)
- GetGilsonDevice
  - CRoboocyte2DeviceNet, [388](#)
- GetGlobalRepeat
  - CDigOutStimulatorFunctionNet, [59](#)
- GetGNDI
  - CCMOSMea\_FunctionNet, [37](#)
- GetGroupADCBits
  - CCMOSMea\_FunctionNet, [37](#)
- GetGroupChannelBitmaskBySelect
  - CCMOSMea\_FunctionNet, [37](#)
- GetGroupChannelBitmaskHS1NCBathCurrent
  - CCMOSMea\_FunctionNet, [37](#), [38](#)
- GetGroupChannelBitmaskHS1NCCol2Current
  - CCMOSMea\_FunctionNet, [38](#)
- GetGroupChannelBitmaskHS1NChipTemp
  - CCMOSMea\_FunctionNet, [38](#)
- GetGroupChannelBitmaskHS1Sidebands
  - CCMOSMea\_FunctionNet, [38](#)
- GetGroupChannelBitmaskHS1TriggerStatus
  - CCMOSMea\_FunctionNet, [38](#), [39](#)
- GetGroupChannelBitmaskIFDigChannels
  - CCMOSMea\_FunctionNet, [39](#)
- GetGroupChannelBitmaskInterfaceADC
  - CCMOSMea\_FunctionNet, [39](#)
- GetGroupChannelBitmaskPacketFrameContext
  - CCMOSMea\_FunctionNet, [39](#)
- GetGroupChannelBitmaskSTG1DACSignal
  - CCMOSMea\_FunctionNet, [39](#), [40](#)
- GetGroupChannelData16
  - CMcsUsbDacqNet, [185](#)
- GetGroupChannelData132
  - CMcsUsbDacqNet, [186](#)
- GetGroupChannelDataUI16
  - CMcsUsbDacqNet, [186](#)
- GetGroupChannelDataUI32
  - CMcsUsbDacqNet, [187](#)
- GetGroupDCOffset
  - CCMOSMea\_FunctionNet, [40](#)

- GetGroupID
  - CCMOSMea\_FunctionNet, [40](#)
  - CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet >, [54](#)
- GetGroupNumberOfChannels
  - CCMOSMea\_FunctionNet, [40](#)
  - CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet >, [54](#)
- GetGroupResolutionPerDigit
  - CCMOSMea\_FunctionNet, [40](#)
- GetGroupSampleSize
  - CCMOSMea\_FunctionNet, [41](#)
  - CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet >, [54](#)
- GetGroupType
  - CCMOSMea\_FunctionNet, [41](#)
  - CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet >, [54](#)
- GetGroupUnit
  - CCMOSMea\_FunctionNet, [41](#)
- GetGyroRange
  - CW2100\_FunctionNet, [505](#)
- GetHardwareMaxRange
  - CMcsUsbDacqNet, [188](#)
- GetHardwareMinRange
  - CMcsUsbDacqNet, [188](#)
- GetHardwareRevision
  - CMcsUsbNet, [234](#)
- GetHasChecksum
  - CWClassicFunctionNet, [561](#)
- GetHashCode
  - HeadstageIDTypeObject, [580](#)
- GetHasRedLedHeadstage
  - CWClassicFunctionNet, [561](#)
- GetHasThermocouple
  - CTcxDeviceNet, [480](#)
- GetHeadstage
  - CStg200xBasicNet, [428](#)
- GetHeadstageActive
  - CMcsUsbNet, [235](#)
- GetHeadstageAdcBits
  - CSCUFunctionNet, [403](#)
- GetHeadstageAdcRangeInMicroVolt
  - CSCUFunctionNet, [404](#)
- GetHeadstageDacBits
  - CSCUFunctionNet, [404](#)
- GetHeadstageDacCurrentRangeInMicroAmpere
  - CSCUFunctionNet, [404](#)
- GetHeadstageDacCurrentResolutionInNanoAmpere
  - CSCUFunctionNet, [405](#)
- GetHeadstageDacVoltageRangeInMilliVolt
  - CSCUFunctionNet, [405](#)
- GetHeadstageDacVoltageResolutionInMicroVolt
  - CSCUFunctionNet, [405](#)
- GetHeadstageGainInPer mille
  - CSCUFunctionNet, [406](#)
- GetHeadstageID
  - CMcsUsbNet, [235](#)
  - CSCUFunctionNet, [406](#)
- GetHeadstageNumberOfAnalogChannels
  - CSCUFunctionNet, [406](#)
- GetHeadstageNumberOfStimulationChannels
  - CSCUFunctionNet, [407](#)
- GetHeadstageOnOff
  - CW2100\_FunctionNet, [505](#)
  - CWClassicFunctionNet, [561](#)
- GetHeadstagePresent
  - CMcsUsbNet, [235](#)
- GetHeadstageSamplerate
  - CSCUFunctionNet, [407](#)
- GetHeadstageSamplingActive
  - CW2100\_FunctionNet, [505](#)
- GetHeadstageSerialNumber
  - CSCUFunctionNet, [407](#)
- GetHeaterLimit
  - CTcxDeviceNet, [481](#)
- GetHeaterTemp
  - CTcxDeviceNet, [481](#)
- GetHighCurrentRange
  - CWarnerUssingFunctionNet, [522](#)
- GetHighpassFilterEnable
  - CFilterConfigurationNet, [69](#)
- GetHWConfig
  - CRoboDeviceNet::RoboMainLowLevelCommands, [586](#)
- GetHWRevision
  - CRoboDeviceNet::RoboMainLowLevelCommands, [586](#)
- GetHWRevisionEeprom
  - CMcsBusNet, [157](#)
- GetI
  - CTcxDeviceNet, [481](#)
- GetIC
  - CRoboDacqNet, [363](#)
- GetIClamp
  - CRoboDacqNet, [363](#)
- GetICoeff
  - CRobo\_FYITemp\_FunctionNet, [355](#)
- GetICOffset
  - CRoboDacqNet, [363](#)
- GetIDecp
  - CTcxDeviceNet, [481](#)
- GetIDent
  - CMcsUsbNet, [235](#)
- GetIdleModeOffset
  - CWarnerUssingFunctionNet, [523](#)
- GetIGain
  - CRoboDacqNet, [363](#)

- GetIMax
  - CTcxDeviceNet, [481](#)
- GetIMin
  - CTcxDeviceNet, [481](#)
- GetImpedanceResult
  - CIntanMea\_FunctionNet, [106](#)
- GetImpedanceTestFrequency
  - CMealImpedanceDeviceNet, [277](#)
- GetImpId
  - CPositionImpDeviceNet, [323](#)
- GetImplantCurrentSetpoint
  - CPositionIIDeviceNet, [320](#)
- GetImplantState
  - CPositionIIDeviceNet, [320](#)
- GetInMovement
  - CRoboDeviceNet, [375](#)
- GetIntanRegister
  - CIntanMea\_FunctionNet, [106](#)
- GetIntBuffer
  - CGenericDevelopDeviceNet, [89](#)
- GetIO
  - CWarnerValveControllerDeviceTesterFunctionNet, [558](#)
- GetIOut
  - CTcxDeviceNet, [481](#)
- GetLastAnswer
  - CGilsonDeviceNet, [100](#)
- GetLastUSBError
  - CMcsUsbNet, [236](#)
- GetLatency
  - CMcsBus\_SensorNet, [143](#)
- GetLatencyCounter
  - CMcsBus\_SensorNet, [143](#)
- GetLayoutConfiguration
  - CMEA2100x256FunctionNet, [248](#)
- GetLEDSwitch
  - CMcsBus\_ExtensionNet, [119](#)
- GetLength
  - CRobo\_FYIProgram\_FunctionNet, [353](#)
- GetLiquidResistance
  - CTEERFunctionNet, [493](#)
  - CWarnerUssingFunctionNet, [523](#)
- GetListModelIndexRange
  - CStg200xBasicNet, [428](#)
- GetListmodeTriggerSource
  - CStg200xBasicNet, [428](#)
- GetLowCurrentRange
  - CWarnerUssingFunctionNet, [523](#)
- GetLowerFrequencyByIndex
  - CIntanMea\_FunctionNet, [106](#)
- GetMajor
  - DriverVersionNet, [571](#)
- GetMaxChunkSize\_Byte
  - CTEERFunctionNet, [493](#)
- GetMaxCurrent
  - CMeaCoatDeviceNet, [258](#)
- GetMaxDurationHighCurrentInMicroSec
  - CMultiwellOptoStimFunctionNet, [301](#)
- GetMaxDutyCycleHighCurrent
  - CMultiwellOptoStimFunctionNet, [301](#)
- GetMaxHeaterPowerMultiwell
  - CTcxDeviceNet, [482](#)
- GetMaxNoPressure
  - CRoboDeviceNet::RoboMainLowLevelCommands, [586](#)
- GetMaxNoPressureWaitTime
  - CRoboDeviceNet::RoboMainLowLevelCommands, [586](#)
- GetMaxNumberOfHeadstages
  - CSCUFunctionNet, [408](#)
- GetMaxNumOfColumns
  - CCMOSMea\_FunctionNet, [41](#)
- GetMaxP
  - CTcxDeviceNet, [482](#)
- GetMaxpDecp
  - CTcxDeviceNet, [482](#)
- GetMaxpMax
  - CTcxDeviceNet, [482](#)
- GetMaxpMin
  - CTcxDeviceNet, [482](#)
- GetMaxPower
  - COKuvisionStimulatorDeviceNet, [310](#)
  - CRobo\_FYITemp\_FunctionNet, [355](#)
- GetMaxPressureWaitTime
  - CRoboDeviceNet::RoboMainLowLevelCommands, [586](#)
- GetMaxSamplingFrequency
  - CMcsUsbDacqNet, [188](#)
- GetMaxStimulusChannelsPerHeadstage
  - CSCUFunctionNet, [408](#)
- GetMaxVoltage
  - CMeaCleanDeviceNet, [254](#)
  - COKuvisionStimulatorDeviceNet, [310](#)
- GetMCAcceleration
  - CMcsBus\_MotorControlNet, [124](#)
- GetMCAccelerationEeprom
  - CMcsBus\_MotorControlNet, [124](#)
- GetMCAccelerationShortCommand
  - CMcsBus\_MotorControlNet, [124](#)
- GetMCAXisRevisionEeprom
  - CMcsBus\_MotorControlNet, [124](#)
- GetMCBreakCurrent
  - CMcsBus\_MotorControlNet, [124](#)
- GetMCBreakCurrentEeprom
  - CMcsBus\_MotorControlNet, [125](#)
- GetMCCConfig
  - CMcsBus\_MotorControlNet, [125](#)
- GetMCCConfigEeprom
  - CMcsBus\_MotorControlNet, [125](#)
- GetMCCurrent
  - CMcsBus\_MotorControlNet, [125](#)
- GetMCCurrentEeprom
  - CMcsBus\_MotorControlNet, [125](#)
- GetMCCurrentMode
  - CMcsBus\_MotorControlNet, [125](#)
- GetMCCurrentModeEeprom

- CMcsBus\_MotorControlNet, [125](#)
- GetMCCurrentModeShortCommand
  - CMcsBus\_MotorControlNet, [126](#)
- GetMCCurrentPosition
  - CMcsBus\_MotorControlNet, [126](#)
- GetMCCurrentShortCommand
  - CMcsBus\_MotorControlNet, [126](#)
- GetMCCurrentSpeed
  - CMcsBus\_MotorControlNet, [126](#)
- GetMCMaxAcceleration
  - CMcsBus\_MotorControlNet, [126](#)
- GetMCMaxAccelerationEeprom
  - CMcsBus\_MotorControlNet, [126](#)
- GetMCMaxCurrent
  - CMcsBus\_MotorControlNet, [126](#)
- GetMCMaxCurrentEeprom
  - CMcsBus\_MotorControlNet, [127](#)
- GetMCMaxSpeed
  - CMcsBus\_MotorControlNet, [127](#)
- GetMCMaxSpeedEeprom
  - CMcsBus\_MotorControlNet, [127](#)
- GetMCMaxTravel
  - CMcsBus\_MotorControlNet, [127](#)
- GetMCMaxTravelEeprom
  - CMcsBus\_MotorControlNet, [127](#)
- GetMCMaxTravelShortCommand
  - CMcsBus\_MotorControlNet, [127](#)
- GetMCMovement
  - CMcsBus\_MotorControlNet, [127](#)
- GetMCNewPosition
  - CMcsBus\_MotorControlNet, [128](#)
- GetMCOutputOnOff
  - CMcsBus\_MotorControlNet, [128](#)
- GetMCPhase
  - CMcsBus\_MotorControlNet, [128](#)
- GetMCPhaseOffset
  - CMcsBus\_MotorControlNet, [128](#)
- GetMCReference
  - CMcsBus\_MotorControlNet, [128](#)
- GetMCReferenceCurrent
  - CMcsBus\_MotorControlNet, [128](#)
- GetMCReferenceCurrentEeprom
  - CMcsBus\_MotorControlNet, [128](#)
- GetMCRegulatorGain
  - CMcsBus\_MotorControlNet, [129](#)
- GetMCRegulatorGainEeprom
  - CMcsBus\_MotorControlNet, [129](#)
- GetMcsBus\_Extension
  - CRobocyte2DeviceNet, [388](#)
- GetMCScalingFactor
  - CMcsBus\_MotorControlNet, [129](#)
- GetMCScalingFactorEeprom
  - CMcsBus\_MotorControlNet, [129](#)
- GetMCSpeed
  - CMcsBus\_MotorControlNet, [129](#)
- GetMCSpeedEeprom
  - CMcsBus\_MotorControlNet, [129](#)
- GetMCSpeedShortCommand
  - CMcsBus\_MotorControlNet, [129](#)
- CMcsBus\_MotorControlNet, [129](#)
- GetMCSpeedUnitEeprom
  - CMcsBus\_MotorControlNet, [130](#)
- GetMCStandbyCurrent
  - CMcsBus\_MotorControlNet, [130](#)
- GetMCStandbyCurrentEeprom
  - CMcsBus\_MotorControlNet, [130](#)
- GetMCStandbyTime
  - CMcsBus\_MotorControlNet, [130](#)
- GetMCStandbyTimeEeprom
  - CMcsBus\_MotorControlNet, [130](#)
- GetMea21UsbPort
  - CMcsUsbNet, [236](#)
- GetMeaLayout
  - CMcsUsbDacqNet, [188](#)
- GetMemoryUsageDAC
  - CStg200xDownloadBasicNet, [449](#)
- GetMemoryUsageSyncout
  - CStg200xDownloadBasicNet, [449](#)
- GetMinimalThreshold
  - CMcsBus\_SensorNet, [143](#)
- GetMinNoPressureWaitTime
  - CRoboDeviceNet::RoboMainLowLevelCommands, [586](#)
- GetMinor
  - DriverVersionNet, [571](#), [572](#)
- GetMinPressure
  - CRoboDeviceNet, [375](#)
  - CRoboDeviceNet::RoboMainLowLevelCommands, [586](#)
- GetMinPressureWaitTime
  - CRoboDeviceNet::RoboMainLowLevelCommands, [587](#)
- GetMinSamplingFrequencyStepsize
  - CMcsUsbDacqNet, [188](#)
- GetMinVoltage
  - CMeaCleanDeviceNet, [254](#)
- GetModeSelect
  - CPulseGeneratorFunctionNet, [341](#)
- GetModuleCurrent
  - CStg200xDownloadNet, [458](#)
- GetModuleTemp
  - CStg200xDownloadNet, [458](#)
- GetMovementError
  - CRoboDeviceNet, [375](#)
- GetMovePump
  - CMcsBus\_SensorNet, [144](#)
- GetMultiHeadstageMode
  - CW2100\_FunctionNet, [505](#)
- GetMultiplexedDataChannelsInBlock
  - CStimulusFunctionNet, [466](#)
- GetNanoVoltsPerKelvin
  - CMcsBus\_TempSensorNet, [149](#)
- GetNeurochipMemoryData
  - CCMOSMea\_FunctionNet, [41](#)
- GetNeurochipMemorySize
  - CCMOSMea\_FunctionNet, [42](#)
- GetNIC\_MS

- CRoboDacqNet, 363
- GetNUC\_MS
  - CRoboDacqNet, 363
- GetNumAmplifications
  - CPgaDeviceNet, 318
- GetNumber
  - CMeaSwitchDeviceNet, 279
  - CSw2to64DeviceNet, 475
- GetNumberOfAnalogChannels
  - CStg200xBasicNet, 428
  - CStimulusFunctionNet, 467
  - CW2100\_StimulatorFunctionNet, 512
- GetNumberOfAudioChannels
  - CMeaAudioFunctionNet, 251
- GetNumberOfAvailableChambers
  - CWarnerUssingFunctionNet, 524
- GetNumberOfAvailableSamples
  - CTEERFunctionNet, 493
- GetNumberOfChannels
  - CDigOutStimulatorFunctionNet, 60
- GetNumberOfDataBits
  - CMcsUsbDacqNet, 189
- GetNumberOfDevices
  - CMcsUsbListNet, 224
- GetNumberOfHardwareSlotsForChambers
  - CWarnerUssingFunctionNet, 524
- GetNumberOfHWADCCChannels
  - CMcsUsbDacqNet::CHWInfo, 104
- GetNumberOfHWDACPaths
  - CStg200xBasicNet, 428
- GetNumberOfHWDigitalChannels
  - CMcsUsbDacqNet::CHWInfo, 104
- GetNumberOfStimulationElectrodes
  - CStg200xBasicNet, 428
- GetNumberOfStimulationSourcesPerElectrode
  - CStg200xBasicNet, 429
- GetNumberOfSupportedGroups
  - CCMOSMea\_FunctionNet, 42
  - CDacqGroupChannelSelectionTemplateNet< Dac-  
qGroupChannelEnumTemplateNet, Dac-  
qGroupChannelEnumTemplate, CDevice-  
GroupChannelInfoTemplateNet >, 55
- GetNumberOfSyncoutChannels
  - CStg200xBasicNet, 429
  - CW2100\_StimulatorFunctionNet, 512
- GetNumberOfTriggerInputs
  - CStg200xBasicNet, 429
  - CW2100\_StimulatorFunctionNet, 512
- GetNumConfigurations
  - CMcsUsbNet, 236
- GetNumControlChannels
  - CTcxDeviceNet, 482
- GetNumDestinations
  - CMcsUsbFactoryNet, 213
- GetNumDevices
  - CTcxDeviceNet, 482
- GetNumEntries
  - DriverVersionNet, 572
- GetNumFrequencyRanges
  - CPgaDeviceNet, 318
- GetNumMeasureChannels
  - CTcxDeviceNet, 482
- GetNUV\_MS
  - CRoboDacqNet, 363
- GetOffsetCurrent
  - CMeaCoatDeviceNet, 259
- GetOnOff
  - CPositionIIDeviceNet, 320
  - CTcxDeviceNet, 483
- GetOutputCurrent
  - CMeaCoatDeviceNet, 259
- GetOutputRate
  - CStg200xBasicNet, 429
- GetOutputVoltage
  - CMeaCleanDeviceNet, 254
- GetP
  - CTcxDeviceNet, 483
- GetParameter
  - CRoboDeviceNet::RoboMainLowLevelCommands,  
587
- GetPattern
  - CMeaSwitchDeviceNet, 279
- GetPatternBool
  - CMeaSwitchDeviceNet, 280
- GetPauseDuration
  - CMeaCoatDeviceNet, 259
- GetPCoeff
  - CRobo\_FYITemp\_FunctionNet, 355
- GetPDecp
  - CTcxDeviceNet, 483
- GetPeriod
  - CPulseGeneratorFunctionNet, 342
- GetPeriod\_us
  - CTEERFunctionNet, 494
- GetPermanentCurrentInMicroAmp
  - CMultiwellOptoStimFunctionNet, 301
- GetPGain
  - CRoboDacqNet, 363
- GetPhases
  - CRoboDeviceNet::RoboMainLowLevelCommands,  
587
- GetPicFirmwareType
  - CW2100\_FunctionNet, 505
- GetPiezoState
  - CMcsBus\_SensorNet, 144
- GetPlateClampLockState
  - CMultiwellDeviceNet, 295
- GetPlateClampState
  - CMultiwellDeviceNet, 295
- GetPlateClampStateByHeadstage
  - CMultiwellCallbackFunctionNet, 291
  - CMultiwellDeviceNet, 295
- GetPlateClampStateByHeadstageEvent
  - CMultiwellCallbackFunctionNet, 293
- GetPlateMux
  - CMultiwellDeviceNet, 296

- GetPlateMuxByHeadstage
  - CMultiwellDeviceNet, [296](#)
- GetPlateType
  - CMultiwellDeviceNet, [296](#)
- GetPlateTypeByHeadstage
  - CMultiwellCallbackFunctionNet, [292](#)
  - CMultiwellDeviceNet, [296](#)
- GetPlateTypeByHeadstageEvent
  - CMultiwellCallbackFunctionNet, [293](#)
- GetPMax
  - CTcxDeviceNet, [483](#)
- GetPMin
  - CTcxDeviceNet, [483](#)
- GetPoti
  - CMcsUsbDacqNet, [189](#)
- GetPOut
  - CTcxDeviceNet, [483](#)
- GetPressure
  - CMcsBus\_SensorNet, [144](#)
- GetPressureOffset
  - CMcsBus\_SensorNet, [144](#)
- GetPressureRange
  - CPPCFunctionNet, [329](#)
- GetPulseform
  - CokuvisionStimulatorDeviceNet, [310](#)
- GetPulseLength
  - CPulseGeneratorFunctionNet, [342](#)
- GetPumpCouple
  - CPPS\_FunctionNet, [335](#)
- GetPumpEnableSpeedRatio
  - CPPS\_FunctionNet, [335](#)
- GetPumpFastOnOff
  - CPPS\_FunctionNet, [335](#)
- GetPumpFastSpeed
  - CPPS\_FunctionNet, [335](#)
- GetPumpFunctionSpeeds
  - CPPS\_FunctionNet, [335](#)
- GetPumpManualOnOff
  - CPPS\_FunctionNet, [335](#)
- GetPumpMaxSpeed
  - CPPS\_FunctionNet, [336](#)
- GetPumpModeType
  - CPPCFunctionNet, [329](#)
  - CPPS\_FunctionNet, [336](#)
- GetPumpSpeed
  - CRoboFluidDeviceNet, [385](#)
- GetPumpSpeedRatio
  - CPPS\_FunctionNet, [336](#)
- GetPumpSpeedUnit
  - CPPCFunctionNet, [329](#)
  - CPPS\_FunctionNet, [336](#)
- GetPWM
  - CFluidControlDeviceNet, [76](#)
- GetPwrOut
  - CTcxDeviceNet, [483](#)
- GetPwrSet
  - CTcxDeviceNet, [484](#)
- GetRatedCapacity
  - CMultiBatteryChargerDeviceNet, [288](#)
- GetReady
  - CMealImpedanceDeviceNet, [277](#)
- GetRecordingNumber
  - CRoboDacqNet, [363](#)
- GetReferenceElectrodeMode
  - CSCUFunctionNet, [408](#)
- GetReferenceElectrodeSwitchState
  - CSCUFunctionNet, [408](#)
- GetReferenceTemperature
  - CFluidControlDeviceNet, [76](#)
- GetRegulationTimeouts
  - CMcsBus\_SensorNet, [144](#)
- GetRegulatorFactor
  - CMcsBus\_SensorNet, [145](#)
- GetRegulatorOnOff
  - CMcsBus\_SensorNet, [145](#)
  - CRobo\_FYITemp\_FunctionNet, [355](#)
- GetRegulatorStatus
  - CMcsBus\_SensorNet, [145](#)
- GetRepeats
  - CProgramPressureCurveNet, [339](#)
- GetRes1
  - CTcxDeviceNet, [484](#)
- GetRes2
  - CTcxDeviceNet, [484](#)
- GetResetFilter
  - CWClassicFunctionNet, [562](#)
- GetResistanceC
  - CRoboDacqNet, [363](#)
- GetResistanceV
  - CRoboDacqNet, [363](#)
- GetResolutionPerDigit
  - CMcsUsbDacqNet, [189](#)
- GetResS
  - CTcxDeviceNet, [484](#)
- GetResult
  - CMealImpedanceDeviceNet, [277](#)
- GetResX
  - CTcxDeviceNet, [484](#)
- GetRFConnectionStatus
  - CWClassicFunctionNet, [562](#)
- GetRFFrequency
  - CPositionImpDeviceNet, [323](#)
- GetRFFrequencyHeadstage
  - CWClassicFunctionNet, [562](#)
- GetRFFrequencyReceiver
  - CWClassicFunctionNet, [562](#)
- GetRFPower
  - CWClassicFunctionNet, [562](#)
- GetRoboDacq
  - CRoboocyte2DeviceNet, [388](#)
- GetRoboFluidDevice
  - CEncapsulatorDeviceNet, [63](#)
  - CRoboocyte2DeviceNet, [388](#)
- GetRotaryPositionCode
  - CTEERFunctionNet, [494](#)
- GetRotatePump



- CMcsBus\_SensorNet, [145](#)
- GetROut
  - CTcxDeviceNet, [484](#)
- GetRTC
  - CokuvisionStimulatorDeviceNet, [310](#)
- GetSampleBufferChunk
  - CTEERFunctionNet, [494](#)
- GetSampleInterval
  - CLIH3DeviceNet, [114](#)
- GetSamplePeriode
  - CMcsBus\_SensorNet, [145](#)
- GetSamplerate
  - CMcsUsbDacqNet, [189](#)
- GetSampleVoltageBuffer\_uV
  - CTEERFunctionNet, [494](#)
- GetScaleFactorU1
  - CTEERFunctionNet, [495](#)
- GetScaleFactorU2
  - CTEERFunctionNet, [495](#)
- GetScanHeadstagesResult
  - CWClassicFunctionNet, [562](#)
- GetScreen
  - CRoboDacqNet, [364](#)
- GetSearchReferenceFastAccel
  - CRoboDeviceNet::RoboMainLowLevelCommands, [587](#)
- GetSearchReferenceFastSpeed
  - CRoboDeviceNet::RoboMainLowLevelCommands, [587](#)
- GetSearchReferenceFineAccel
  - CRoboDeviceNet::RoboMainLowLevelCommands, [587](#)
- GetSearchReferenceFineSpeed
  - CRoboDeviceNet::RoboMainLowLevelCommands, [587](#)
- GetSearchReferenceMethod
  - CRoboDeviceNet::RoboMainLowLevelCommands, [588](#)
- GetSearchReferenceMoveOut
  - CRoboDeviceNet::RoboMainLowLevelCommands, [588](#)
- GetSearchReferenceOffsetPos
  - CRoboDeviceNet::RoboMainLowLevelCommands, [588](#)
- GetSelectedChannels
  - CW2100\_FunctionNet, [506](#)
- GetSelectedHeadstage
  - CWClassicFunctionNet, [562](#)
- GetSelectedHeadstageState
  - CW2100\_FunctionNet, [506](#)
- GetSensorType
  - CTcxDeviceNet, [484](#)
- GetSerialNumber
  - CMcsUsbNet, [236](#)
- GetSerialNumberHeadstage
  - CWClassicFunctionNet, [562](#)
- GetSetpoint
  - CTcxDeviceNet, [484](#)
- GetSetpointDecp
  - CTcxDeviceNet, [485](#)
- GetSetpointMax
  - CTcxDeviceNet, [485](#)
- GetSetpointMin
  - CTcxDeviceNet, [485](#)
- GetShortBuffer
  - CGenericDevelopDeviceNet, [90](#)
- GetSimulation
  - CRoboDacqNet, [364](#)
- GetSingleHeater
  - CMcsBus\_FYIExtensionNet, [120](#)
- GetSingleValve
  - CFluidControlDeviceNet, [77](#)
  - CRoboFluidDeviceNet, [385](#)
- GetSlope
  - CMeaCleanDeviceNet, [255](#)
  - CMeaCoatDeviceNet, [259](#)
- GetSoftwareKey
  - CMcsUsbNet, [236](#)
- GetSoftwareKeyString
  - CMcsUsbNet, [236](#)
- GetSoilPressure
  - CMcsBus\_SensorNet, [145](#)
- GetSoilTemp
  - CRobo\_FYITemp\_FunctionNet, [355](#)
- GetSourceBulk
  - CCMOSMea\_FunctionNet, [42](#)
- GetSourceDrain
  - CCMOSMea\_FunctionNet, [42](#)
- GetSourceGate
  - CCMOSMea\_FunctionNet, [42](#)
- GetStartTriggerSlope
  - CDigOutStimulatorFunctionNet, [60](#)
- GetState
  - CRFFFunctionNet, [351](#)
  - CRobo\_FYIProgram\_FunctionNet, [353](#)
- GetStatus
  - CMcsUsbNet, [236](#)
  - DriverVersionNet, [572](#)
- GetStatusOfLastCommand
  - CMcsUsbNet, [237](#)
- GetStgProgramInfo
  - CStg200xBasicNet, [429](#), [430](#)
- GetStgVersionInfo
  - CStg200xBasicNet, [430](#)
- GetStimulationPatternMemory
  - CW2100\_StimulatorFunctionNet, [512](#)
- GetStimulatorStatus
  - CokuvisionStimulatorDeviceNet, [310](#)
- GetStimulusParametersCache
  - CW2100\_FunctionNet, [506](#)
- GetStimulusParametersFromSelectedHS
  - CW2100\_FunctionNet, [506](#)
- GetStimulusSites
  - CCMOSMea\_FunctionNet, [42](#)
- GetStiumlusParameters
  - CW2100\_FunctionNet, [506](#)

- GetStopTriggerSlope
  - CDigOutStimulatorFunctionNet, 60
- GetSubChannel
  - CMcsBus\_MotorControlNet, 130
- GetSupplyVoltage
  - CPPCFunctionNet, 329
  - CPPS\_FunctionNet, 336
- GetSweepCount
  - CStg200xDownloadBasicNet, 450
- GetSync
  - CWarnerValveControllerDeviceTesterFunctionNet, 558
- GetSyncoutMap
  - CStg200xBasicNet, 430
- GetSyncState
  - CMcsBus\_SensorNet, 145
- GetTableName
  - CWarnerValveControllerDeviceNet, 541
- GetTableNamebyIndex
  - CWarnerValveControllerDeviceNet, 541
- GetTableNamebyIndexEvent
  - CWarnerValveControllerDeviceNet, 555
- GetTablepointer
  - CRetinalLedDeviceNet, 347
- GetTemperatur
  - CMcsBus\_TempSensorNet, 149
- GetTestMode
  - CRFFFunctionNet, 351
- GetThermocoupleCalibration
  - CFluidControlDeviceNet, 77
  - CTcxDeviceNet, 485
- GetThermocoupleNanovoltPerKelvin
  - CFluidControlDeviceNet, 77
  - CTcxDeviceNet, 485
- GetThermocoupleReferenceTemp
  - CTcxDeviceNet, 485
- GetThermocoupleTemp
  - CTcxDeviceNet, 485
- GetThermocoupleTempAbs
  - CTcxDeviceNet, 486
- GetThermocoupleTemperature
  - CFluidControlDeviceNet, 78
- GetThermoOffset
  - CMcsBus\_TempSensorNet, 149
- GetThermoTemp
  - CMcsBus\_TempSensorNet, 150
- GetThermoVoltage
  - CMcsBus\_TempSensorNet, 150
- GetTimeInPause
  - CMeaCoatDeviceNet, 259
- GetTimeInPlateau
  - CMeaCoatDeviceNet, 260
- GetTimeResolutionInNanoSeconds
  - CW2100\_StimulatorFunctionNet, 512
- GetTimeSlot
  - CW2100\_StimulatorFunctionNet, 512
- GetTotalMemory
  - CStg200xBasicNet, 431
- CStimulusFunctionNet, 467
- GetTotalNumberOfDigitalPorts
  - CWarnerValveControllerDeviceNet, 541
- GetTotalNumberOfTables
  - CWarnerValveControllerDeviceNet, 541
- GetTotalNumberOfValves
  - CWarnerValveControllerDeviceNet, 541
- GetTotalTableSize
  - CWarnerValveControllerDeviceNet, 542
- GetTrigger
  - CStg200xDownloadBasicNet, 450
- GetTriggerSource
  - CStg200xBasicNet, 431
- GetU1Offset
  - CWarnerUssingFunctionNet, 524
- GetU1Reference
  - CWarnerUssingFunctionNet, 524
- GetU2Offset
  - CWarnerUssingFunctionNet, 525
- GetU2Reference
  - CWarnerUssingFunctionNet, 525
- GetUByteBuffer
  - CGenericDevelopDeviceNet, 91
- GetUC
  - CRoboDacqNet, 364
- GetUClamp
  - CRoboDacqNet, 364
- GetUCOffset
  - CRoboDacqNet, 364
- GetUIntBuffer
  - CGenericDevelopDeviceNet, 91
- GetUnit
  - CTcxDeviceNet, 486
- GetUnitDescription
  - CWarnerUssingFunctionNet, 525
- GetUnitExponent
  - CWarnerUssingFunctionNet, 526
- GetUnitName
  - CWarnerUssingFunctionNet, 526
- GetUnitsPerDigit
  - CWarnerUssingFunctionNet, 526
- GetUOut
  - CTcxDeviceNet, 486
- GetUpdateDisplay
  - CRoboDacqNet, 364
- GetUpperFrequencyByIndex
  - CIntanMea\_FunctionNet, 107
- GetUptimeSeconds
  - CTEERFunctionNet, 495
  - CWarnerUssingFunctionNet, 528
- GetUSBDeviceIDFromFX3Image
  - CMcsUsbFactoryNet, 213
- GetUsbListEntries
  - CMcsUsbListNet, 224
- GetUsbListEntry
  - CMcsUsbListNet, 224
  - CMcsUsbNet, 237
- GetUseBubble



- CPPS\_FunctionNet, [336](#)
- GetUserCodeFromBitFile
  - CMcsUsbFactoryNet, [213](#)
- GetUserCodeFromFlash
  - CMcsUsbFactoryNet, [213](#)
- GetUserDefinedName
  - CW2100\_FunctionNet, [506](#)
- GetUserDefinedNameCache
  - CW2100\_FunctionNet, [506](#)
- GetUserDefinedNameFromSelectedHS
  - CW2100\_FunctionNet, [507](#)
- GetUserParameter
  - CRoboDeviceNet::RoboMainLowLevelCommands, [588](#)
- GetUShortBuffer
  - CGenericDevelopDeviceNet, [92](#)
- GetUV
  - CRoboDacqNet, [364](#)
- GetUOffset
  - CRoboDacqNet, [364](#)
- GetValue
  - CTcxDeviceNet, [486](#)
- GetValueHires
  - CTcxDeviceNet, [486](#)
- GetValve
  - CFluidControlDeviceNet, [78](#)
  - CRoboFluidDeviceNet, [385](#)
- GetValve1
  - CRobo\_FYIProgram\_FunctionNet, [353](#)
- GetValve2
  - CRobo\_FYIProgram\_FunctionNet, [353](#)
- GetValveActive
  - CPPCFunctionNet, [330](#)
  - CWarnerValveControllerDeviceNet, [542](#)
- GetValveActiveEvent
  - CWarnerValveControllerDeviceNet, [555](#)
- GetValveBoardRevision
  - CWarnerValveControllerDeviceNet, [542](#)
- GetValveBoardRevisionEvent
  - CWarnerValveControllerDeviceNet, [555](#)
- GetValveBoardRevisionString
  - CWarnerValveControllerDeviceNet, [542](#)
- GetValveDigitalInPort
  - CWarnerValveControllerDeviceNet, [542](#)
- GetValveDigitalInPortEvent
  - CWarnerValveControllerDeviceNet, [555](#)
- GetValveManualGroup
  - CWarnerValveControllerDeviceNet, [543](#)
- GetValveManualGroupEvent
  - CWarnerValveControllerDeviceNet, [556](#)
- GetValveManualState
  - CWarnerValveControllerDeviceNet, [543](#)
- GetValveManualStateEvent
  - CWarnerValveControllerDeviceNet, [556](#)
- GetValveMode
  - CWarnerValveControllerDeviceNet, [543](#)
- GetValveModeEvent
  - CWarnerValveControllerDeviceNet, [556](#)
- GetValves
  - CMcsBus\_FYIExtensionNet, [120](#)
- GetValvesActiveMap
  - CWarnerValveControllerDeviceNet, [544](#)
- GetValvesManualStateMap
  - CWarnerValveControllerDeviceNet, [544](#)
- GetValveTableEntry
  - CWarnerValveControllerDeviceNet, [544](#)
- GetVDD3I
  - CCMOSMea\_FunctionNet, [42](#)
- GetVDDI
  - CCMOSMea\_FunctionNet, [42](#)
- GetVersion
  - CMcsUsbNet, [237](#)
- GetVersionInt
  - DriverVersionNet, [572](#), [573](#)
- GetVersionString
  - DriverVersionNet, [573](#)
- GetVMMaxNegativeCurrent
  - CMcsBus\_VoltageModeNet, [152](#)
- GetVMMaxNegativeCurrentEeprom
  - CMcsBus\_VoltageModeNet, [152](#)
- GetVMMaxNegativeVoltage
  - CMcsBus\_VoltageModeNet, [152](#)
- GetVMMaxNegativeVoltageEeprom
  - CMcsBus\_VoltageModeNet, [152](#)
- GetVMMaxPositiveCurrent
  - CMcsBus\_VoltageModeNet, [152](#)
- GetVMMaxPositiveCurrentEeprom
  - CMcsBus\_VoltageModeNet, [152](#)
- GetVMMaxPositiveVoltage
  - CMcsBus\_VoltageModeNet, [152](#)
- GetVMMaxPositiveVoltageEeprom
  - CMcsBus\_VoltageModeNet, [153](#)
- GetVMOuputOnOff
  - CMcsBus\_VoltageModeNet, [153](#)
- GetVMVoltage
  - CMcsBus\_VoltageModeNet, [153](#)
- GetVoltage
  - COKuvisionStimulatorDeviceNet, [310](#)
- GetVoltage12V
  - CRoboDeviceNet, [375](#)
- GetVoltage12VLimit
  - CRoboDeviceNet, [375](#)
- GetVoltage5V
  - CRoboDeviceNet, [375](#)
- GetVoltage5VLimit
  - CRoboDeviceNet, [375](#)
- GetVoltageAirvalve
  - CRoboDeviceNet, [375](#)
- GetVoltageAirvalveLimit
  - CRoboDeviceNet, [376](#)
- GetVoltageClampControllerParam\_D
  - CWarnerUssingFunctionNet, [528](#)
- GetVoltageClampControllerParam\_I
  - CWarnerUssingFunctionNet, [528](#)
- GetVoltageClampControllerParam\_P
  - CWarnerUssingFunctionNet, [529](#)

- GetVoltageRangeIndex
  - CMcsUsbDacqNet, [189](#)
- GetVoltageRangeInMicroVolt
  - CMcsUsbDacqNet, [189](#)
  - CStg200xBasicNet, [431](#)
  - CStimulusFunctionNet, [467](#)
  - CW2100\_StimulatorFunctionNet, [512](#)
- GetVoltageRangeInMilliVolt
  - CMcsUsbDacqNet, [190](#)
- GetVoltageResolutionInMicroVolt
  - CStg200xBasicNet, [431](#)
  - CStimulusFunctionNet, [467](#)
  - CW2100\_StimulatorFunctionNet, [513](#)
- GetVoltageRs485A
  - CRoboDeviceNet, [376](#)
- GetVoltageRs485ALimit
  - CRoboDeviceNet, [376](#)
- GetVoltageRs485B
  - CRoboDeviceNet, [376](#)
- GetVoltageRs485BLimit
  - CRoboDeviceNet, [376](#)
- GetVoltageValves
  - CRoboDeviceNet, [376](#)
- GetVoltageValvesLimit
  - CRoboDeviceNet, [376](#)
- GetVolti
  - CTcxDeviceNet, [486](#)
- GetWaveform
  - CTEERFunctionNet, [495](#)
- GetWaveLengthInNanometer
  - CMultiwellOptoStimFunctionNet, [302](#)
- GetWorkingFrequency
  - CRFFFunctionNet, [351](#)
- GetWPADebugMode
  - CWClassicFunctionNet, [562](#)
- GetWPAType
  - CWClassicFunctionNet, [562](#)
- GetXGain
  - CRoboDacqNet, [364](#)
- GetXilinxFlashOffset
  - CMcsUsbFactoryNet, [214](#)
- GetXilinxFlashReadCommand
  - CMcsUsbFactoryNet, [214](#)
- GND\_SWITCH\_BIT
  - CW2100\_StimulatorFunctionNet, [515](#)
- GroupID
  - CDeviceGroupChannelInfoTemplateNet< Dacq-GroupChannelEnumTemplateNet >, [57](#)
- GroupType
  - CDeviceGroupChannelInfoTemplateNet< Dacq-GroupChannelEnumTemplateNet >, [57](#)
- HasAnalogOut
  - CSCUFunctionNet, [409](#)
- HasGalvanicIsolation
  - CSCUFunctionNet, [409](#)
- HasHSPowerSwitch
  - CSCUFunctionNet, [409](#)
- HasIMU
  - HeadStageIDType, [578](#)
- HasOptoCurrentMeasurement
  - HeadStageIDType, [578](#)
- HasRadioControl
  - CRadioControlledDevicesNet, [344](#)
- HasRef
  - CRoboDeviceNet::RoboMainLowLevelCommands, [589](#)
- HasRefI
  - CRoboStatorDeviceNet, [391](#)
- HasRefXY
  - CRoboStatorDeviceNet, [391](#)
- HasRefZ
  - CRoboStatorDeviceNet, [391](#)
- HasSoftwareKey
  - CMcsUsbNet, [237](#)
- HeadStageIDType, [577](#)
  - CompareTo, [578](#)
  - ElectricalStimulation, [578](#)
  - Entry, [578](#)
  - Equals, [578](#)
  - HasIMU, [578](#)
  - HasOptoCurrentMeasurement, [578](#)
  - HeadStageIDType, [578](#)
  - HeadstageType, [579](#)
  - HeadstageTypeEnum, [577](#)
  - ID, [579](#)
  - MeasuringOnly, [578](#)
  - NumberOfAnalogChannels, [579](#)
  - NumberOfStimulationChannels, [579](#)
  - OpticalStimulation, [578](#)
  - SN, [579](#)
  - StimulusParameters, [579](#)
  - ToString, [578](#)
  - Type, [579](#)
  - TypeValue, [579](#)
  - Unknown, [578](#)
  - UserDefinedName, [579](#)
  - Valid, [579](#)
  - W16IsW14, [579](#)
- HeadstageIDTypeObject, [580](#)
  - \_AdditionalText, [580](#)
  - \_IdType, [581](#)
  - AdditionalText, [581](#)
  - Equals, [580](#)
  - GetHashCode, [580](#)
  - HeadstageIDTypeObject, [580](#)
  - IdType, [581](#)
  - ToString, [580](#)
- HeadStageIDTypeState, [581](#)
  - ControlState, [581](#)
  - DataState, [581](#)
  - IdType, [581](#)
  - State, [581](#)
- HeadstageType
  - HeadStageIDType, [579](#)
- HeadstageTypeEnum
  - HeadStageIDType, [577](#)

- HLADacq
  - CHLADeviceNet, [103](#)
- HWInfo
  - CMcsUsbDacqNet, [190](#)
- HwVersion
  - CMcsUsbListEntryNet, [222](#)
- ID
  - HeadStageIDType, [579](#)
- IdProduct
  - DeviceIdNet, [567](#)
- IdType
  - HeadstageIDTypeObject, [581](#)
  - HeadstageIDTypeState, [581](#)
- IdVendor
  - DeviceIdNet, [567](#)
- IsAnalogOutEnabled
  - CSCUFunctionNet, [409](#)
- IsAutomaticAnalogOut
  - CSCUFunctionNet, [410](#)
- IsBusy
  - CPPCFunctionNet, [330](#)
- IsChamberAvailable
  - CWarnerUssingFunctionNet, [529](#)
- IsChipPowered
  - CCMOSMea\_FunctionNet, [42](#)
- IsConnected
  - CMcsUsbNet, [237](#)
- IsDeviceHighSpeed
  - CMcsUsbNet, [237](#)
- IsDeviceHighSpeedCapable
  - CMcsUsbNet, [238](#)
- IsDeviceTypeOf
  - CMcsUsbListNet, [225](#)
- IsDigitalChannelDedicated
  - CMcsUsbDacqNet::CHWInfo, [105](#)
- IsDigitalOutPortInverted
  - CWarnerValveControllerDeviceNet, [544](#)
- IsDigitalOutPortInvertedEvent
  - CWarnerValveControllerDeviceNet, [556](#)
- IsExceptionsEnabled
  - CMcsUsbNet, [238](#)
- IsGateFloating
  - CCMOSMea\_FunctionNet, [42](#)
- IsHeadstageAvailable
  - CSCUFunctionNet, [410](#)
- IsHeadstageAvailableEvent
  - CSCUFunctionNet, [414](#)
- IsHighCurrentMode
  - CWarnerUssingFunctionNet, [529](#)
- IsHSPowered
  - CSCUFunctionNet, [410](#)
- IsInDacqLegacyMode
  - CSCUFunctionNet, [410](#)
- IsInternalCalibrationFinished
  - CTEERFunctionNet, [495](#)
  - CWarnerUssingFunctionNet, [530](#)
- IsPlateTypeValid
  - CMultiwellDeviceNet, [297](#)
- IsPlateTypeValidByHeadstage
  - CMultiwellCallbackFunctionNet, [292](#)
  - CMultiwellDeviceNet, [297](#)
- IsPlateTypeValidByHeadstageEvent
  - CMultiwellCallbackFunctionNet, [293](#)
- IsPulseEnabled
  - CWarnerUssingFunctionNet, [530](#)
- IsPumpMotorOn
  - CRoboFluidDeviceNet, [385](#)
- IsRunning
  - CMeaCleanDeviceNet, [255](#)
  - CMeaCoatDeviceNet, [260](#)
- IsSamplingFinished
  - CTEERFunctionNet, [496](#)
- IsUserTriggerEnabled
  - CLIH3DeviceNet, [114](#)
- IsValveDigitalInInverted
  - CWarnerValveControllerDeviceNet, [545](#)
- IsValveDigitalInInvertedEvent
  - CWarnerValveControllerDeviceNet, [556](#)
- IsValveOpen
  - CWarnerValveControllerDeviceNet, [545](#)
- IsValveOpenEvent
  - CWarnerValveControllerDeviceNet, [556](#)
- IsValveOpenInAnalogMode
  - CWarnerValveControllerDeviceNet, [545](#)
- IsValveOpenInAnalogModeEvent
  - CWarnerValveControllerDeviceNet, [556](#)
- IsValveOpenInDigitalMode
  - CWarnerValveControllerDeviceNet, [546](#)
- IsValveOpenInDigitalModeEvent
  - CWarnerValveControllerDeviceNet, [557](#)
- ListModeSendStart
  - CStg200xBasicNet, [432](#)
- ListModeSendStop
  - CStg200xBasicNet, [432](#)
- ListOfChangedTriggers
  - StgStatusNet, [594](#)
- LoadPressure
  - CPPCFunctionNet, [330](#)
- LoadUserFirmware
  - CMcsUsbFactoryNet, [214](#)
- LoadValveTable
  - CWarnerValveControllerDeviceNet, [546](#)
- LockPlateClamp
  - CMultiwellDeviceNet, [297](#)
- m\_Bottom
  - CCMOSMeaDeviceNet::CRegionOfInterestRect, [345](#)
- m\_Left
  - CCMOSMeaDeviceNet::CRegionOfInterestRect, [345](#)
- m\_pGilsonDevice
  - CGilsonDeviceNet, [100](#)
- m\_pMcsBus\_MotorControlNet
  - CRoboFluidDeviceNet, [386](#)
- m\_pMcsUsb

- CMcsUsbFunctionNet, [219](#)
- m\_pMcsUsbFunction
  - CMcsUsbFunctionNet, [219](#)
- m\_pRoboFluidDevice
  - CRoboFluidDeviceNet, [386](#)
- m\_Right
  - CCMOSMeaDeviceNet::CRegionOfInterestRect, [345](#)
- m\_Top
  - CCMOSMeaDeviceNet::CRegionOfInterestRect, [345](#)
- Manufacturer
  - CMcsUsbListEntryNet, [222](#)
- MaxBitNumber
  - DigitalSource< digitalsourceenum >, [567](#)
- Mcs, [22](#)
- Mcs::Usb, [22](#)
  - DEVICE\_NOT\_FOUND, [27](#)
  - enCMosMeaChipType, [26](#)
  - EnSTG200x\_STATUS, [27](#)
  - nMos16LV, [27](#)
  - nMos32LV, [27](#)
  - nMos36LN, [27](#)
  - nMos64LN, [27](#)
  - NOT\_CONNECTED, [27](#)
  - OK, [27](#)
  - OnChannelData, [27](#)
  - OnDeviceArrivalRemoval, [27](#)
  - OnError, [27](#)
  - OnMcsUsbDeviceState, [27](#)
  - OnMcsUsbDeviceStateCallback, [27](#)
  - OnMwPollStatus, [28](#)
  - OnStg200xDataHandler, [28](#)
  - OnStg200xErrorHandler, [28](#)
  - OnStgPollStatus, [28](#)
  - OnUpdateFirmwareProgress, [28](#)
  - OnUpdateFirmwareStatusChange, [28](#)
  - RoboStatusEventDelegate, [28](#)
  - unknown, [27](#)
- McsBus
  - CPPCDeviceNet, [325](#)
  - CPPS\_DeviceNet, [333](#)
  - CRoboDeviceNet, [383](#)
- MCSBUS1
  - FirmwareDestinationNames, [575](#)
- MCSBUS10
  - FirmwareDestinationNames, [575](#)
- MCSBUS11
  - FirmwareDestinationNames, [575](#)
- MCSBUS12
  - FirmwareDestinationNames, [575](#)
- MCSBUS13
  - FirmwareDestinationNames, [575](#)
- MCSBUS2
  - FirmwareDestinationNames, [575](#)
- MCSBUS3
  - FirmwareDestinationNames, [575](#)
- MCSBUS4
  - FirmwareDestinationNames, [576](#)
- MCSBUS5
  - FirmwareDestinationNames, [576](#)
- MCSBUS6
  - FirmwareDestinationNames, [576](#)
- MCSBUS7
  - FirmwareDestinationNames, [576](#)
- MCSBUS8
  - FirmwareDestinationNames, [576](#)
- MCSBUS9
  - FirmwareDestinationNames, [576](#)
- McsBus\_MotorControl
  - CPeristalticPumpDeviceNet, [316](#)
  - CPPCDeviceNet, [325](#)
  - CPPS\_DeviceNet, [333](#)
  - CRoboDeviceNet, [383](#)
  - CRoboFluidDeviceNet, [386](#)
- McsBus\_Sensor
  - CPPCDeviceNet, [325](#)
  - CPPS\_DeviceNet, [333](#)
- McsBus\_VoltageMode
  - CFluidControlDeviceNet, [80](#)
- McsBus\_XY
  - CRoboDeviceNet, [380](#)
- McsBus\_ZI
  - CRoboDeviceNet, [380](#)
- McsUsbDeviceStateEvent
  - CMcsUsbDeviceStatePushFunctionNet, [208](#)
  - CMcsUsbDeviceStatePushNet, [209](#)
- MCU1
  - FirmwareDestinationNames, [576](#)
- MeaAudioFunctionNet
  - CMeaDeviceNet, [269](#)
- MeaDigitalDataFunctionNet
  - CMeaDeviceNet, [269](#)
- MeaFeedbackFunctionNet
  - CMeaDeviceNet, [269](#)
- Measure
  - CPathIdentDeviceNet, [313](#)
- MeasureReservoir
  - CPPCFunctionNet, [331](#)
- MeasuringOnly
  - HeadStageIDType, [578](#)
- MeFunctionNet
  - CMeaDeviceNet, [270](#)
- mkfilter
  - mkfilterNet, [582](#)
  - mkfilter\_coef\_in\_one\_set
    - mkfilterNet, [582](#)
  - mkfilter\_highpass\_coeff
    - mkfilterNet, [583](#)
  - mkfilter\_highpass\_frequency\_from\_coeff
    - mkfilterNet, [583](#)
  - mkfilter\_highpass\_frequency\_from\_k
    - mkfilterNet, [583](#)
  - mkfilter\_highpass\_k
    - mkfilterNet, [583](#)
  - mkfilter\_MCS

- mkfilterNet, 583
- mkfilter\_MCS\_k
  - mkfilterNet, 583, 584
- mkfilter\_normalize\_coeffs\_int
  - mkfilterNet, 584
- mkfilter\_normalize\_coeffs\_short
  - mkfilterNet, 584
- mkfilter\_normalize\_scale\_coeffs\_int
  - mkfilterNet, 584
- mkfilter\_scale\_coef\_in\_one\_set
  - mkfilterNet, 584
- mkfilterNet, 582
  - mkfilter, 582
  - mkfilter\_coef\_in\_one\_set, 582
  - mkfilter\_highpass\_coeff, 583
  - mkfilter\_highpass\_frequency\_from\_coeff, 583
  - mkfilter\_highpass\_frequency\_from\_k, 583
  - mkfilter\_highpass\_k, 583
  - mkfilter\_MCS, 583
  - mkfilter\_MCS\_k, 583, 584
  - mkfilter\_normalize\_coeffs\_int, 584
  - mkfilter\_normalize\_coeffs\_short, 584
  - mkfilter\_normalize\_scale\_coeffs\_int, 584
  - mkfilter\_scale\_coef\_in\_one\_set, 584
- MoveAbs
  - CRoboDeviceNet, 376
- MoveAbsI
  - CRoboStatorDeviceNet, 391, 392
- MoveAbsXY
  - CRoboStatorDeviceNet, 392
- MoveAbsZ
  - CRoboStatorDeviceNet, 392
- MultibootGetCypressImgeld
  - CMcsUsbNet, 238
- MultibootGetImgeld
  - CMcsUsbNet, 238
- MultibootGetSelectedImage
  - CMcsUsbNet, 238
- MultibootSelectImage
  - CMcsUsbNet, 238
- MwPollStatusEvent
  - CStg200xDownloadNet, 461
- nMos16LV
  - Mcs::Usb, 27
- nMos32LV
  - Mcs::Usb, 27
- nMos36LN
  - Mcs::Usb, 27
- nMos64LN
  - Mcs::Usb, 27
- NOT\_CONNECTED
  - Mcs::Usb, 27
- NumberOfAnalogChannels
  - HeadStageIDType, 579
- NumberOfChannels
  - CDeviceGroupChannelInfoTemplateNet < Dacq-GroupChannelEnumTemplateNet >, 57
- NumberOfStimulationChannels
  - HeadStageIDType, 579
- NumCoefSets
  - CCreateFilterNet, 49
- OK
  - Mcs::Usb, 27
- OnChannelData
  - Mcs::Usb, 27
- OnDeviceArrivalRemoval
  - Mcs::Usb, 27
- OnError
  - Mcs::Usb, 27
- OnGetActiveRunningTableNumber
  - CWarnerValveControllerDeviceNet, 546
- OnGetAnalogThresholdHigh
  - CWarnerValveControllerDeviceNet, 546
- OnGetAnalogThresholdLow
  - CWarnerValveControllerDeviceNet, 546
- OnGetAnalogVoltage
  - CWarnerValveControllerDeviceNet, 546
- OnGetAvailableHeadstages
  - CSCUFunctionNet, 411
- OnGetCurrentNumberOfValves
  - CWarnerValveControllerDeviceNet, 547
- OnGetDigitalOutPortValve
  - CWarnerValveControllerDeviceNet, 547
- OnGetDigitalPortDirection
  - CWarnerValveControllerDeviceNet, 547
- OnGetDisplayMode
  - CWarnerValveControllerDeviceNet, 547
- OnGetPlateClampStateByHeadstage
  - CMultiwellCallbackFunctionNet, 292
- OnGetPlateTypeByHeadstage
  - CMultiwellCallbackFunctionNet, 292
- OnGetTableNamebyIndex
  - CWarnerValveControllerDeviceNet, 547
- OnGetValveActive
  - CWarnerValveControllerDeviceNet, 547
- OnGetValveBoardRevision
  - CWarnerValveControllerDeviceNet, 547
- OnGetValveDigitalInPort
  - CWarnerValveControllerDeviceNet, 547
- OnGetValveManualGroup
  - CWarnerValveControllerDeviceNet, 547
- OnGetValveManualState
  - CWarnerValveControllerDeviceNet, 548
- OnGetValveMode
  - CWarnerValveControllerDeviceNet, 548
- OnIsDigitalOutPortInverted
  - CWarnerValveControllerDeviceNet, 548
- OnIsHeadstageAvailable
  - CSCUFunctionNet, 411
- OnIsPlateTypeValidByHeadstage
  - CMultiwellCallbackFunctionNet, 292
- OnIsValveDigitalInInverted
  - CWarnerValveControllerDeviceNet, 548
- OnIsValveOpen
  - CWarnerValveControllerDeviceNet, 548
- OnIsValveOpenInAnalogMode

- CWarnerValveControllerDeviceNet, 548
- OnIsValveOpenInDigitalMode
  - CWarnerValveControllerDeviceNet, 548
- OnMcsUsbDeviceState
  - Mcs::Usb, 27
- OnMcsUsbDeviceStateCallback
  - Mcs::Usb, 27
- OnMwPollStatus
  - Mcs::Usb, 28
- OnStg200xDataHandler
  - Mcs::Usb, 28
- OnStg200xErrorHandler
  - Mcs::Usb, 28
- OnStgPollStatus
  - Mcs::Usb, 28
- OnTableEntryChanged
  - CWarnerValveControllerDeviceNet, 548
- OnUpdateFirmwareProgress
  - Mcs::Usb, 28
- OnUpdateFirmwareStatusChange
  - Mcs::Usb, 28
- OpenPipe
  - CGenericDevelopDeviceNet, 93
- OpenPlateClamp
  - CMultiwellDeviceNet, 297
- operator=
  - DeviceIdNet, 566
- OpticalStimulation
  - HeadStageIDType, 578
- Order
  - CCreateFilterNet, 49
  - CFilterPropertyNet, 73
- PatternListStart
  - COctoPotDeviceNet, 306
- PIC
  - FirmwareDestinationNames, 576
- PIC2
  - FirmwareDestinationNames, 576
- PIC3
  - FirmwareDestinationNames, 576
- PIC4
  - FirmwareDestinationNames, 576
- PollStatusEvent
  - CStimulusFunctionNet, 473
  - CW2100\_StimulatorFunctionNet, 515
- PowerChip
  - CCMOSMea\_FunctionNet, 43
- PowerHS
  - CSCUFunctionNet, 411
- PPCFunction
  - CPPCDeviceNet, 325
- PPS\_Function
  - CPPS\_DeviceNet, 333
- PrepareAndAppendData
  - CStg200xDownloadNet, 458
  - CStimulusFunctionNet, 468
- PrepareAndSendData
  - CStg200xDownloadNet, 459
- CStimulusFunctionNet, 469
- PrepareChannelData
  - CDigOutStimulatorFunctionNet, 61
- PrepareData
  - CStimulusFunctionNet, 469
  - CW2100\_StimulatorFunctionNet, 513
- PrepareDataSync
  - CW2100\_StimulatorFunctionNet, 513
- Product
  - CMcsUsbListEntryNet, 223
- Program
  - CProgramPressureCurveNet, 339
- PulseGenerator
  - CW2100\_FunctionNet, 508
- PumpOff
  - CRoboFluidDeviceNet, 385
- PumpOn
  - CRoboFluidDeviceNet, 385
- QueryTriggerstatus
  - CStg200xDownloadNet, 460
- RampStart
  - COctoPotDeviceNet, 306
- Read
  - CExternDTesterDeviceNet, 64
- Read2
  - CExternDTesterDeviceNet, 64
- ReadBlockFromFlash
  - CMcsUsbFactoryNet, 214
- ReadBlockFromIFBGlobalEEPROM
  - CMcsUsbFactoryNet, 215
- ReadBlockFromNVMEM
  - CMcsUsbFactoryNet, 215
- ReadClipping
  - CLIH3DeviceNet, 114
- ReadEEPROMRegisterPreconfig
  - CMcsUsbNet, 239
- ReadPipe
  - CGenericDevelopDeviceNet, 93
- ReadRegister
  - CMcsUsbNet, 239
- ReadRegister32
  - CMcsUsbNet, 239
- ReadRegisterTimeSlot
  - CMcsUsbNet, 239
- Receive
  - CSerialPortNet, 415
- ReceiveString
  - CSerialPortNet, 415
- RemoveSoftwareKey
  - CMcsUsbNet, 240
- RescanHeadstage
  - CMcsUsbNet, 240
- ResetAdcOffset
  - COctoPotDeviceNet, 306
- ResetChannelmap
  - CWClassicFunctionNet, 563
- ResetDacOffset



- COctoPotDeviceNet, [306](#)
- ResetHighpassFilter
  - CFilterConfigurationNet, [69](#)
- ResetPipe
  - CGenericDevelopDeviceNet, [93](#)
- ResetStatus
  - CStg200xDownloadBasicNet, [450](#)
- RFFunction
  - CPositionIIDeviceNet, [321](#)
- RoboDacq
  - CHiClampDeviceNet, [101](#)
- RoboDevice
  - CSafeISDeviceNet, [398](#)
- RoboError\_AnotherMaster
  - CRoboDeviceNet, [380](#)
- RoboError\_Base
  - CRoboDeviceNet, [380](#)
- RoboError\_CannotEscapeEndSwitch
  - CRoboDeviceNet, [380](#)
- RoboError\_CommandAlreadyInProgress
  - CRoboDeviceNet, [380](#)
- RoboError\_CommandNotPossible
  - CRoboDeviceNet, [380](#)
- RoboError\_CommunicationTimeout
  - CRoboDeviceNet, [380](#)
- RoboError\_DacqNotReady
  - CRoboDeviceNet, [381](#)
- RoboError\_DLLMovementTimeout
  - CRoboDeviceNet, [381](#)
- RoboError\_FindReferenceMethod
  - CRoboDeviceNet, [381](#)
- RoboError\_GilsonCommandPending
  - CRoboDeviceNet, [381](#)
- RoboError\_GilsonTimeout
  - CRoboDeviceNet, [381](#)
- RoboError\_GilsonWrondID
  - CRoboDeviceNet, [381](#)
- RoboError\_McsBus\_UnknownCommand
  - CRoboDeviceNet, [381](#)
- RoboError\_NoEndSwitch
  - CRoboDeviceNet, [381](#)
- RoboError\_NoMoreData
  - CRoboDeviceNet, [381](#)
- RoboError\_NoReference
  - CRoboDeviceNet, [381](#)
- RoboError\_NoSpeedOrAcceleration
  - CRoboDeviceNet, [382](#)
- RoboError\_OverPressure
  - CRoboDeviceNet, [382](#)
- RoboError\_ParameterNotAllowed
  - CRoboDeviceNet, [382](#)
- RoboError\_PeristalticTimeout
  - CRoboDeviceNet, [382](#)
- RoboError\_Phase0OutOfRange
  - CRoboDeviceNet, [382](#)
- RoboError\_PollLoopCanceled
  - CRoboDeviceNet, [382](#)
- RoboError\_PollLoopCanceledAndStopMovement
  - CRoboDeviceNet, [382](#)
- RoboError\_Pressure
  - CRoboDeviceNet, [382](#)
- RoboError\_RangeExceeded
  - CRoboDeviceNet, [382](#)
- RoboError\_StateChangeNotPossible
  - CRoboDeviceNet, [382](#)
- RoboError\_Timeout
  - CRoboDeviceNet, [383](#)
- RoboError\_UnknownCommand
  - CRoboDeviceNet, [383](#)
- RoboMainLowLevelCommand
  - CRoboDeviceNet, [383](#)
- RoboMainStatorLowLevelCommand
  - CRoboStatorDeviceNet, [395](#)
- RoboStatusEvent
  - CRoboDeviceNet, [383](#)
- RoboStatusEventDelegate
  - Mcs::Usb, [28](#)
- RunTable
  - CRoboDacqNet, [364](#)
- SampleRate
  - CCreateFilterNet, [49](#)
- Samplerate
  - CMcsUsbDacqNet, [206](#)
- Scale
  - CCreateFilterNet, [49](#)
- ScanForHeadstages
  - CWClassicFunctionNet, [563](#)
- SelectHeadstage
  - CW2100\_FunctionNet, [507](#)
- SelectTimeSlot
  - CW2100\_StimulatorFunctionNet, [513](#)
- Send
  - CSerialPortNet, [415](#)
- SendBuffered
  - CGilsonDeviceNet, [100](#)
- SendChannelData
  - CDigOutStimulatorFunctionNet, [61](#)
  - CStg200xDownloadBasicNet, [452](#)
- SendCommand
  - CLIH3DeviceNet, [114](#)
- SendImmediate
  - CGilsonDeviceNet, [100](#)
- SendImmediateGetResponse
  - CGilsonDeviceNet, [100](#)
- SendMultiplexedData
  - CStimulusFunctionNet, [470](#)
- SendPreparedData
  - CStimulusFunctionNet, [470](#)
  - CW2100\_StimulatorFunctionNet, [513](#)
- SendSegmentDefine
  - CStg200xDownloadNet, [460](#)
- SendSegmentSelect
  - CStg200xDownloadNet, [460](#)
- SendSegmentStart
  - CStg200xDownloadNet, [461](#)
- SendStart

- CStg200xBasicNet, [432](#)
- CStimulusFunctionNet, [470](#)
- CW2100\_StimulatorFunctionNet, [514](#)
- SendStartDacq
  - CMcsUsbDacqNet, [190](#)
- SendStartStgAndDacq
  - CMcsUsbDacqNet, [190](#)
- SendStop
  - CStg200xBasicNet, [432](#)
  - CStimulusFunctionNet, [470](#)
  - CW2100\_StimulatorFunctionNet, [514](#)
- SendStopDacq
  - CMcsUsbDacqNet, [191](#)
- SendStopStgAndDacq
  - CMcsUsbDacqNet, [191](#)
- SendStopStgAndDacqWithOptions
  - CMcsUsbDacqNet, [191](#)
- SendSyncData
  - CStg200xDownloadBasicNet, [452](#)
- Sensor
  - CFYIDeviceNet, [81](#)
  - CMeasureTableDeviceNet, [278](#)
  - CPatchServerDeviceNet, [312](#)
- SerialNumber
  - CMcsUsbListEntryNet, [223](#)
  - CMcsUsbNet, [247](#)
- SerialPort
  - CHLADeviceNet, [103](#)
- Set4ADCCatchampAverageShift
  - CMcsBus\_SensorNet, [146](#)
- Set4ADCMode
  - CMcsBus\_SensorNet, [146](#)
- Set4DAC
  - CMcsBus\_SensorNet, [146](#)
- Set\_Values
  - CNF\_GenDeviceNet, [304](#)
  - CPathIdentDeviceNet, [313](#)
- SetAbsMaxCurrentInMicroAmp
  - CMultiwellOptoStimFunctionNet, [302](#)
- SetAccelerationI
  - CRoboStatorDeviceNet, [392](#)
- SetAccelerationNativeI
  - CRoboStatorDeviceNet, [392](#)
- SetAccelerationNativeXY
  - CRoboStatorDeviceNet, [392](#)
- SetAccelerationNativeZ
  - CRoboStatorDeviceNet, [392](#)
- SetAccelerationXY
  - CRoboStatorDeviceNet, [393](#)
- SetAccelerationZ
  - CRoboStatorDeviceNet, [393](#)
- SetAccelGyroDesiredRate
  - CW2100\_FunctionNet, [507](#)
- SetAccelGyroEnabled
  - CW2100\_FunctionNet, [507](#)
- SetAccelRange
  - CW2100\_FunctionNet, [507](#)
- SetActiveRunningTableNumber
  - CWarnerValveControllerDeviceNet, [548](#)
- SetADC
  - CWarnerValveControllerDeviceTesterFunctionNet, [559](#)
- SetAdcChannels
  - CSafeISDeviceNet, [396](#)
- SetADCInputOffset
  - CCMOSMea\_FunctionNet, [43](#)
- SetADCOffset
  - CLIH3DeviceNet, [115](#)
- SetAdcOffset
  - COctoPotDeviceNet, [307](#)
- SetAdcSamplePos
  - CSafeISDeviceNet, [396](#)
- SetAFormat
  - CFilterCoefficientsNet, [66](#)
- SetAirpressureLimit
  - CRoboDeviceNet, [377](#)
- SetAirValve
  - CRoboDeviceNet, [377](#)
- SetAllDigout
  - CRoboDacqNet, [365](#)
- SetAmplificationSwitch
  - COctoPotDeviceNet, [307](#)
- SetAmplitude
  - CChannelTestDeviceNet, [33](#)
- SetAmplitude\_nA
  - CTEERFunctionNet, [496](#)
- SetAnalogOutADCRange
  - CSCUFunctionNet, [411](#)
- SetAnalogOutChannel
  - CW2100\_FunctionNet, [507](#)
- SetAnalogOutChannels
  - CSCUFunctionNet, [411](#)
- SetAnalogOutDACRange
  - CSCUFunctionNet, [413](#)
- SetAnalogOutFilter
  - CW2100\_FunctionNet, [507](#)
- SetAnalogThresholdHigh
  - CWarnerValveControllerDeviceNet, [549](#)
- SetAnalogThresholdLow
  - CWarnerValveControllerDeviceNet, [549](#)
- SetAnalogVoltageRange
  - CPPCFunctionNet, [331](#)
- SetAnalogVoltages
  - CPPS\_FunctionNet, [336](#)
- SetAttenuation
  - CChannelTestDeviceNet, [33](#)
- SetAudioChannels
  - CMeaAudioFunctionNet, [251](#), [252](#)
  - CW2100\_FunctionNet, [507](#)
- SetAutocalibrationDisabled
  - CStg200xBasicNet, [432](#)
- SetAxisConfig
  - CRoboDeviceNet::RoboMainLowLevelCommands, [589](#)
- SetAxisLED
  - CRoboocyte2DeviceNet, [389](#)



- SetAxisParametersEeprom
  - CMcsBus\_AxisParametersNet, 117, 118
- SetBandwidthByIndex
  - CLntanMea\_FunctionNet, 107
- SetBaseFrequency
  - CRFFunctionNet, 351
- SetBaseSamplerate
  - CCMOSMeaDeviceNet, 46
- SetBath
  - CCMOSMea\_FunctionNet, 43
- SetBathclamp
  - COctoPotDeviceNet, 307
- SetBathMode
  - CCMOSMea\_FunctionNet, 43
- SetBFormat
  - CFilterCoefficientsNet, 67
- SetBlankingEnable
  - CStg200xBasicNet, 433, 434
- SetBoostAlwaysOnMode
  - CW2100\_StimulatorFunctionNet, 514
- SetBuffer
  - CGenericDevelopDeviceNet, 94
- SetBufferIndex
  - CTEERFunctionNet, 496
- SetBusAddress
  - CMcsBusNet, 157
- SetBusAddressEeprom
  - CMcsBusNet, 157
- SetByteBuffer
  - CGenericDevelopDeviceNet, 94
- SetCalibration
  - CTcxDeviceNet, 486
- SetCardinalDacqSamplerate
  - CInterfaceboardFunctionNet, 109
- SetCardinalStgOutputrate
  - CInterfaceboardFunctionNet, 109
- SetChannel
  - CSw2to64DeviceNet, 475
- SetChannelmap
  - CWClassicFunctionNet, 563
- SetChannels
  - CSw2to64DeviceNet, 475
- SetChannelSwitch
  - COctoPotDeviceNet, 307
- SetChargingMode
  - CMultiBatteryChargerDeviceNet, 288
- SetChargingPCoefficient
  - CMultiBatteryChargerDeviceNet, 288
- SetCheckVoltage
  - CokuvisionStimulatorDeviceNet, 311
- SetClampMode
  - CTEERFunctionNet, 496
  - CWarnerUssingFunctionNet, 530
- SetColorRgb
  - CMultiwellOptoStimFunctionNet, 302
- SetColorStr
  - CMultiwellOptoStimFunctionNet, 302
- SetCommand
  - CMcsBusNet, 158
  - CPedoterDeviceNet, 315
  - CRoboDacqNet, 365
- SetConfiguration
  - CMcsUsbNet, 240
- SetConfigurationBit
  - CRoboDacqNet, 365
- SetConfigurationBitAxc
  - CRoboDacqNet, 365
- SetConfigurationBitBlu\_Led
  - CRoboDacqNet, 365
- SetConfigurationBitBlu\_LedToggleFast
  - CRoboDacqNet, 365
- SetConfigurationBitBlu\_LedToggleSlow
  - CRoboDacqNet, 365
- SetConfigurationBitCC\_Gen
  - CRoboDacqNet, 365
- SetConfigurationBitCV\_Gen
  - CRoboDacqNet, 365
- SetConfigurationBitRC\_Gen
  - CRoboDacqNet, 366
- SetConfigurationBitRed\_Led
  - CRoboDacqNet, 366
- SetConfigurationBitRed\_LedSaturation
  - CRoboDacqNet, 366
- SetConfigurationBitRed\_LedToggleFast
  - CRoboDacqNet, 366
- SetConfigurationBitRed\_LedToggleSlow
  - CRoboDacqNet, 366
- SetConfigurationBitRelais
  - CRoboDacqNet, 366
- SetConfigurationBitRV\_Gen
  - CRoboDacqNet, 366
- SetConfigurationBitStream
  - CRoboDacqNet, 366
- SetConfigurationBitSupply
  - CRoboDacqNet, 366
- SetControllerParams
  - CTEERFunctionNet, 497
- SetCrossTalkOffset
  - CRoboDacqNet, 366
- SetCrossTalkOptimum
  - CRoboDacqNet, 367
- SetCurrentAirvalveLimit
  - CRoboDeviceNet, 377
- SetCurrentAndAir
  - CRoboDeviceNet, 377
- SetCurrentAndAirXY
  - CRoboStatorDeviceNet, 393
- SetCurrentEditTableNumber
  - CWarnerValveControllerDeviceNet, 549
- SetCurrentEnable
  - CTEERFunctionNet, 497
- SetCurrentFactor
  - CokuvisionStimulatorDeviceNet, 311
- SetCurrentMode
  - CStg200xBasicNet, 434
- SetCycles

- CMeaCleanDeviceNet, 255
- CMeaCoatDeviceNet, 260
- SetD
  - CTcxDeviceNet, 487
- SetDacAmplificationFactor
  - CStg200xBasicNet, 434
- SetDacAutoControl
  - COctoPotDeviceNet, 307
- SetDacIdleValue
  - CLIH3DeviceNet, 115
- SetDacMode
  - CSafeISDeviceNet, 396
- SetDACOffset
  - CokuvisionStimulatorDeviceNet, 311
- SetDacOffset
  - CDacCalibrationFunctionNet, 51
  - COctoPotDeviceNet, 307
- SetDacPeriode
  - CSafeISDeviceNet, 397
- SetDacPulseform
  - CSafeISDeviceNet, 397
- SetDacqLegacyMode
  - CSCUFunctionNet, 413
- SetDacRange
  - CW2100\_FunctionNet, 508
- SetDACs
  - CMcsBus\_SensorNet, 146
- SetDacUseIdleValue
  - CLIH3DeviceNet, 115
- SetDacValue
  - COctoPotDeviceNet, 307
- SetDataMode
  - CMcsUsbDacqNet, 192
- SetDefault
  - CWarnerValveControllerDeviceNet, 549
- SetDestinationSerialNumber
  - CMcsUsbFactoryNet, 215
- SetDetectionThreshold
  - CMcsBus\_SensorNet, 146
- SetDevice
  - CTcxDeviceNet, 487
- SetDeviceList
  - CPositionImpDeviceNet, 324
- SetDeviceType
  - CTcxDeviceNet, 487
- SetDevname
  - CTcxDeviceNet, 487
- SetDiagnosticMode
  - CIntanMea\_FunctionNet, 107
- SetDigitalData
  - CMeaDigitalDataFunctionNet, 271
- SetDigitalOut
  - CMeaDeviceNet, 266
- SetDigitalOutPortInvert
  - CWarnerValveControllerDeviceNet, 549
- SetDigitalOutPortValve
  - CWarnerValveControllerDeviceNet, 550
- SetDigitalPortDirection
  - CWarnerValveControllerDeviceNet, 550
- SetDigitalSource
  - CMcsUsbDacqNet, 192–194
- SetDigitalStimulatorTrigger
  - CW2100\_StimulatorFunctionNet, 514
- SetDigitalStimulatorTriggerSlope
  - CW2100\_StimulatorFunctionNet, 514
- SetDigout
  - CFluidControlDeviceNet, 78
  - CRoboDacqNet, 367
- SetDigoutMode
  - CStg200xBasicNet, 435
- SetDigOutState
  - CLIH3DeviceNet, 115
- SetDigoutValue
  - CStg200xBasicNet, 435
- SetDIO
  - CMcsBus\_FYIExtensionNet, 120
- SetDischargeCurrentSetPoint
  - CMultiBatteryChargerDeviceNet, 289
- SetDisplayMode
  - CWarnerValveControllerDeviceNet, 550
- SetDisplayText
  - CRoboDacqNet, 367
- SetDownsampleFactor
  - CRoboDacqNet, 367
- SetDSPHighPassByIndex
  - CIntanMea\_FunctionNet, 107
- SetDuration
  - CMeaCoatDeviceNet, 260
- SetEEPromPage
  - CLIH3DeviceNet, 116
- SetElectrodeDacMux
  - CStg200xBasicNet, 435–437
- SetElectrodeEnable
  - CStg200xBasicNet, 437–439
- SetElectrodeMode
  - CStg200xBasicNet, 440, 441
- SetEnableAmplifierProtectionSwitch
  - CStg200xBasicNet, 441, 442
- SetEnableHeaterLimit
  - CTcxDeviceNet, 487
- SetEnablePulse
  - CWarnerUssingFunctionNet, 531
- SetEnableThermocouple
  - CTcxDeviceNet, 487
- SetExternalElectrodeEnable
  - CStg200xBasicNet, 443
- SetExternalLED
  - CTEERFunctionNet, 497
- SetFAAmplification
  - CStg200xBasicNet, 444
- SetFilter
  - CRoboDacqNet, 367
- SetFilterCoeffs
  - CRoboDacqNet, 367
- SetFilterParameter
  - CFilterConfigurationNet, 69

- CFilterConfigurationRegisterNet, 71
- SetFilterParameterPermanent
  - CFilterConfigurationNet, 69
  - CFilterConfigurationRegisterNet, 71
- SetFilterParametersHeadstage
  - CWClassicFunctionNet, 563
- SetFinalDischargeVoltage
  - CMultiBatteryChargerDeviceNet, 289
- SetFrequency
  - CChannelTestDeviceNet, 33
  - CRadioControlledDevicesNet, 344
- SetGain
  - CPgaDeviceNet, 318
- SetGate
  - CCMOSMea\_FunctionNet, 43
- SetGateFloating
  - CCMOSMea\_FunctionNet, 43
- SetGateToVOP
  - CCMOSMea\_FunctionNet, 43
- SetGlobalRepeat
  - CDigOutStimulatorFunctionNet, 61
- SetGyroRange
  - CW2100\_FunctionNet, 508
- SetHasChecksum
  - CWClassicFunctionNet, 563
- SetHeadstage
  - CStg200xBasicNet, 444
- SetHeadstageOnOff
  - CW2100\_FunctionNet, 508
  - CWClassicFunctionNet, 563
- SetHeadstageSamplingActive
  - CW2100\_FunctionNet, 508
- SetHeadstageToSleep
  - CW2100\_FunctionNet, 508
- SetHeaterLimit
  - CTcxDeviceNet, 487
- SetHighCurrentMode
  - CWarnerUssingFunctionNet, 531
- SetHighpassFilterEnable
  - CFilterConfigurationNet, 69
- SetHWConfig
  - CRoboDeviceNet::RoboMainLowLevelCommands, 589
- SetHWRevision
  - CRoboDeviceNet::RoboMainLowLevelCommands, 589
- SetHWRevisionEeprom
  - CMcsBusNet, 158
- SetHWSelectedChannels
  - CWClassicFunctionNet, 563
- SetI
  - CTcxDeviceNet, 487
- SetIClamp
  - CRoboDacqNet, 367
- SetICoeff
  - CRobo\_FYITemp\_FunctionNet, 355
- SetICOffset
  - CRoboDacqNet, 367
- SetIdleModeOffset
  - CWarnerUssingFunctionNet, 531
- SetIGain
  - CRoboDacqNet, 367
- SetImpedanceTestFrequency
  - CMealImpedanceDeviceNet, 277
- SetImpId
  - CPositionImpDeviceNet, 324
- SetImplantCurrentSetpoint
  - CPositionIIDeviceNet, 321
- SetInMovement
  - CRoboDeviceNet, 377
- SetIntanRegister
  - CIntanMea\_FunctionNet, 107
- SetIntBuffer
  - CGenericDevelopDeviceNet, 94
- SetIO
  - CWarnerValveControllerDeviceTesterFunctionNet, 559
- SetIODirection
  - CWarnerValveControllerDeviceTesterFunctionNet, 559
- SetLatency
  - CMcsBus\_SensorNet, 146
- SetLayoutConfiguration
  - CMEA2100x256FunctionNet, 248
- SetLED
  - CRetinaLedDeviceNet, 347
- SetLEDSwitch
  - CMcsBus\_ExtensionNet, 119
- SetLength
  - CRobo\_FYIProgram\_FunctionNet, 353
- SetLiquidResistance
  - CTEERFunctionNet, 497
  - CWarnerUssingFunctionNet, 531
- SetListmodeIndexRange
  - CStg200xBasicNet, 444
- SetListmodeTriggerSource
  - CStg200xBasicNet, 444
- SetLowCurrentMode
  - CWarnerUssingFunctionNet, 532
- SetLumi
  - CRetinaLedDeviceNet, 347
- SetMaxCurrent
  - CMeaCoatDeviceNet, 261
- SetMaxDurationHighCurrentInMicroSec
  - CMultiwellOptoStimFunctionNet, 303
- SetMaxDutyCycleHighCurrent
  - CMultiwellOptoStimFunctionNet, 303
- SetMaxHeaterPowerMultiwell
  - CTcxDeviceNet, 488
- SetMaxNoPressure
  - CRoboDeviceNet::RoboMainLowLevelCommands, 589
- SetMaxNoPressureWaitTime
  - CRoboDeviceNet::RoboMainLowLevelCommands, 589
- SetMaxP

- CTcxDeviceNet, [488](#)
- SetMaxPower
  - COkuvisionStimulatorDeviceNet, [311](#)
  - CRobo\_FYITemp\_FunctionNet, [355](#)
- SetMaxPressureWaitTime
  - CRoboDeviceNet::RoboMainLowLevelCommands, [589](#)
- SetMaxVoltage
  - CMeaCleanDeviceNet, [255](#)
  - COkuvisionStimulatorDeviceNet, [311](#)
- SetMCAcceleration
  - CMcsBus\_MotorControlNet, [130](#)
- SetMCAccelerationEeprom
  - CMcsBus\_MotorControlNet, [131](#)
- SetMCAccelerationShortCommand
  - CMcsBus\_MotorControlNet, [131](#)
- SetMCAXisRevisionEeprom
  - CMcsBus\_MotorControlNet, [131](#)
- SetMCBreakCurrent
  - CMcsBus\_MotorControlNet, [131](#)
- SetMCBreakCurrentEeprom
  - CMcsBus\_MotorControlNet, [131](#)
- SetMCConfig
  - CMcsBus\_MotorControlNet, [131](#)
- SetMCConfigEeprom
  - CMcsBus\_MotorControlNet, [132](#)
- SetMCCurrent
  - CMcsBus\_MotorControlNet, [132](#)
- SetMCCurrentEeprom
  - CMcsBus\_MotorControlNet, [132](#)
- SetMCCurrentMode
  - CMcsBus\_MotorControlNet, [132](#)
- SetMCCurrentModeEeprom
  - CMcsBus\_MotorControlNet, [132](#)
- SetMCCurrentModeShortCommand
  - CMcsBus\_MotorControlNet, [132](#)
- SetMCCurrentPosition
  - CMcsBus\_MotorControlNet, [133](#)
- SetMCCurrentShortCommand
  - CMcsBus\_MotorControlNet, [133](#)
- SetMCMaxAcceleration
  - CMcsBus\_MotorControlNet, [133](#)
- SetMCMaxAccelerationEeprom
  - CMcsBus\_MotorControlNet, [133](#)
- SetMCMaxCurrent
  - CMcsBus\_MotorControlNet, [133](#)
- SetMCMaxCurrentEeprom
  - CMcsBus\_MotorControlNet, [133](#)
- SetMCMaxSpeed
  - CMcsBus\_MotorControlNet, [134](#)
- SetMCMaxSpeedEeprom
  - CMcsBus\_MotorControlNet, [134](#)
- SetMCMaxTravel
  - CMcsBus\_MotorControlNet, [134](#)
- SetMCMaxTravelEeprom
  - CMcsBus\_MotorControlNet, [134](#)
- SetMCMaxTravelShortCommand
  - CMcsBus\_MotorControlNet, [134](#)
- SetMCNewPosition
  - CMcsBus\_MotorControlNet, [134](#)
- SetMCOOutputOnOff
  - CMcsBus\_MotorControlNet, [135](#)
- SetMCReference
  - CMcsBus\_MotorControlNet, [135](#)
- SetMCReferenceCurrent
  - CMcsBus\_MotorControlNet, [135](#)
- SetMCReferenceCurrentEeprom
  - CMcsBus\_MotorControlNet, [135](#)
- SetMCRegulatorGain
  - CMcsBus\_MotorControlNet, [135](#)
- SetMCRegulatorGainEeprom
  - CMcsBus\_MotorControlNet, [135](#)
- SetMCRotation
  - CMcsBus\_MotorControlNet, [136](#)
- SetMCScalingFactor
  - CMcsBus\_MotorControlNet, [136](#)
- SetMCScalingFactorEeprom
  - CMcsBus\_MotorControlNet, [136](#)
- SetMCSpeed
  - CMcsBus\_MotorControlNet, [136](#)
- SetMCSpeedEeprom
  - CMcsBus\_MotorControlNet, [136](#)
- SetMCSpeedShortCommand
  - CMcsBus\_MotorControlNet, [136](#)
- SetMCSpeedUnitEeprom
  - CMcsBus\_MotorControlNet, [137](#)
- SetMCStandbyCurrent
  - CMcsBus\_MotorControlNet, [137](#)
- SetMCStandbyCurrentEeprom
  - CMcsBus\_MotorControlNet, [137](#)
- SetMCStandbyTime
  - CMcsBus\_MotorControlNet, [137](#)
- SetMCStandbyTimeEeprom
  - CMcsBus\_MotorControlNet, [137](#)
- SetMeasurementMode
  - CStg200xBasicNet, [444](#)
- SetMinimalThreshold
  - CMcsBus\_SensorNet, [146](#)
- SetMinNoPressureWaitTime
  - CRoboDeviceNet::RoboMainLowLevelCommands, [589](#)
- SetMinPressure
  - CRoboDeviceNet, [377](#)
  - CRoboDeviceNet::RoboMainLowLevelCommands, [589](#)
- SetMinPressureWaitTime
  - CRoboDeviceNet::RoboMainLowLevelCommands, [589](#)
- SetMinVoltage
  - CMeaCleanDeviceNet, [256](#)
- SetModeSelect
  - CPulseGeneratorFunctionNet, [342](#)
- SetMovePump
  - CMcsBus\_SensorNet, [147](#)
- SetMultiHeadstageMode
  - CW2100\_FunctionNet, [508](#)

- SetNanoVoltsPerKelvin
  - CMcsBus\_TempSensorNet, 150
- SetNeurochipMemoryData
  - CCMOSMea\_FunctionNet, 43
- SetNumberOfAnalogChannels
  - CMeaDeviceNet, 266
- SetNumberOfChannels
  - CMeaDeviceNet, 267, 268
  - COctoPotDeviceNet, 307
- SetOffsetCurrent
  - CMeaCoatDeviceNet, 261
- SetOnOff
  - CTcxDeviceNet, 488
- SetOutputMap
  - CStg200xDownloadNet, 461
- SetOutputRate
  - COctoPotDeviceNet, 307
  - CStg200xBasicNet, 445
- SetP
  - CTcxDeviceNet, 488
- SetParameter
  - CRoboDeviceNet::RoboMainLowLevelCommands, 590
- SetPattern
  - CMeaSwitchDeviceNet, 280
- SetPatternBool
  - CMeaSwitchDeviceNet, 280
- SetPatternListEntry
  - COctoPotDeviceNet, 308
- SetPauseDuration
  - CMeaCoatDeviceNet, 261
- SetPCoeff
  - CRobo\_FYITemp\_FunctionNet, 355
- SetPeriod
  - CPulseGeneratorFunctionNet, 342
- SetPeriod\_us
  - CTEERFunctionNet, 498
- SetPermanentCurrentInMicroAmp
  - CMultiwellOptoStimFunctionNet, 303
- SetPersistency
  - CRetinaLedDeviceNet, 347
- SetPGain
  - CRoboDacqNet, 368
- SetPidParameter
  - COctoPotDeviceNet, 308
- SetPiezoState
  - CMcsBus\_SensorNet, 147
- SetPlateMux
  - CMultiwellDeviceNet, 297
- SetPlateMuxByHeadstage
  - CMultiwellDeviceNet, 298
- SetPlateType
  - CMultiwellDeviceNet, 298
- SetPlateTypeByHeadstage
  - CMultiwellDeviceNet, 298
- SetPoti
  - CMcsUsbDacqNet, 194
- SetPressureOffset
  - CMcsBus\_SensorNet, 147
- CPPCFunctionNet, 331
- SetPressureRange
  - CPPCFunctionNet, 331
- SetPulse
  - CWarnerUssingFunctionNet, 532
- SetPulseform
  - COkuvisionStimulatorDeviceNet, 311
- SetPulseLength
  - CPulseGeneratorFunctionNet, 343
- SetPumpCouple
  - CPPS\_FunctionNet, 336
- SetPumpEnableSpeedRatio
  - CPPS\_FunctionNet, 336
- SetPumpFastOnOff
  - CPPS\_FunctionNet, 336
- SetPumpFastSpeed
  - CPPS\_FunctionNet, 337
- SetPumpFunctionSpeeds
  - CPPS\_FunctionNet, 337
- SetPumpManualOnOff
  - CPPS\_FunctionNet, 337
- SetPumpMaxSpeed
  - CPPS\_FunctionNet, 337
- SetPumpModeType
  - CPPCFunctionNet, 331
  - CPPS\_FunctionNet, 337
- SetPumpSpeed
  - CRoboFluidDeviceNet, 385
- SetPumpSpeedRatio
  - CPPS\_FunctionNet, 337
- SetPumpSpeedUnit
  - CPPCFunctionNet, 332
  - CPPS\_FunctionNet, 337
- SetPWM
  - CFluidControlDeviceNet, 78
- SetRampParameter
  - COctoPotDeviceNet, 308
- SetRatedCapacity
  - CMultiBatteryChargerDeviceNet, 289
- SetRatedCapacityVolatile
  - CMultiBatteryChargerDeviceNet, 289
- SetRecordingNumber
  - CRoboDacqNet, 368
- SetReferenceElectrodeMode
  - CSCUFunctionNet, 413
- SetReferenceElectrodeSwitchState
  - CSCUFunctionNet, 413
- SetRegionOfInterests
  - CCMOSMeaDeviceNet, 46
- SetRegulationTimeouts
  - CMcsBus\_SensorNet, 147
- SetRegulatorFactor
  - CMcsBus\_SensorNet, 147
- SetRegulatorOnOff
  - CMcsBus\_SensorNet, 147
  - CRobo\_FYITemp\_FunctionNet, 355
- SetRepeat

- CRetinalLedDeviceNet, [347](#)
- SetRepeats
  - CProgramPressureCurveNet, [340](#)
- SetResetFilter
  - CWClassicFunctionNet, [563](#)
- SetRFFrequency
  - CPositionImpDeviceNet, [324](#)
- SetRFFrequencyHeadstage
  - CWClassicFunctionNet, [563](#)
- SetRFFrequencyReceiver
  - CWClassicFunctionNet, [564](#)
- SetRFFrequencyReceiverEeprom
  - CWClassicFunctionNet, [564](#)
- SetRFLostBehaviour
  - CWClassicFunctionNet, [564](#)
- SetRFPower
  - CWClassicFunctionNet, [564](#)
- SetRotatePump
  - CMcsBus\_SensorNet, [147](#)
- SetRTC
  - CokuvisionStimulatorDeviceNet, [311](#)
- SetSampleInterval
  - CLIH3DeviceNet, [116](#)
- SetSamplePeriode
  - CMcsBus\_SensorNet, [148](#)
- SetSamplerate
  - CMcsUsbDacqNet, [194](#)
- SetScreen
  - CRoboDacqNet, [368](#)
- SetSearchReferenceFastAccel
  - CRoboDeviceNet::RoboMainLowLevelCommands, [590](#)
- SetSearchReferenceFastSpeed
  - CRoboDeviceNet::RoboMainLowLevelCommands, [590](#)
- SetSearchReferenceFineAccel
  - CRoboDeviceNet::RoboMainLowLevelCommands, [590](#)
- SetSearchReferenceFineSpeed
  - CRoboDeviceNet::RoboMainLowLevelCommands, [590](#)
- SetSearchReferenceMethod
  - CRoboDeviceNet::RoboMainLowLevelCommands, [590](#)
- SetSearchReferenceMoveOut
  - CRoboDeviceNet::RoboMainLowLevelCommands, [591](#)
- SetSearchReferenceOffsetPos
  - CRoboDeviceNet::RoboMainLowLevelCommands, [591](#)
- SetSelectedChannels
  - CMcsUsbDacqNet, [194–196](#)
  - CW2100\_FunctionNet, [508](#)
- SetSelectedChannelsQueue
  - CMcsUsbDacqNet, [197–199](#)
- SetSelectedData
  - CMcsUsbDacqNet, [199–201](#)
- SetSelectedHeadstage
  - CWClassicFunctionNet, [564](#)
- SetSensorType
  - CTcxDeviceNet, [488](#)
- SetSerialNumberHeadstage
  - CWClassicFunctionNet, [564](#)
- SetSetpoint
  - CTcxDeviceNet, [488](#)
- SetShortBuffer
  - CGenericDevelopDeviceNet, [95](#)
- SetSimulation
  - CRoboDacqNet, [368](#)
- SetSineParameter
  - COctoPotDeviceNet, [308](#)
- SetSingleHeater
  - CMcsBus\_FYIExtensionNet, [120](#)
- SetSingleValve
  - CFluidControlDeviceNet, [79](#)
  - CRoboFluidDeviceNet, [386](#)
- SetSlope
  - CMeaCleanDeviceNet, [256](#)
  - CMeaCoatDeviceNet, [261](#)
- SetSoftwareKey
  - CMcsUsbNet, [240](#)
- SetSollPressure
  - CMcsBus\_SensorNet, [148](#)
- SetSollTemp
  - CRobo\_FYITemp\_FunctionNet, [356](#)
- SetSourceBulk
  - CCMOSMea\_FunctionNet, [43](#)
- SetSourceDrain
  - CCMOSMea\_FunctionNet, [44](#)
- SetSourceGate
  - CCMOSMea\_FunctionNet, [44](#)
- SetSpeedI
  - CRoboStatorDeviceNet, [393](#)
- SetSpeedNativeI
  - CRoboStatorDeviceNet, [393](#)
- SetSpeedNativeXY
  - CRoboStatorDeviceNet, [393](#)
- SetSpeedNativeZ
  - CRoboStatorDeviceNet, [393](#)
- SetSpeedXY
  - CRoboStatorDeviceNet, [393](#)
- SetSpeedZ
  - CRoboStatorDeviceNet, [394](#)
- SetStartTriggerSlope
  - CDigOutStimulatorFunctionNet, [61](#)
- SetStgProgramInfo
  - CStg200xBasicNet, [445](#)
- SetStimulusSites
  - CCMOSMea\_FunctionNet, [44](#)
- SetStopTriggerSlope
  - CDigOutStimulatorFunctionNet, [62](#)
- SetStringFormat
  - CMcsUsbListEntryNet, [222](#)
  - CMcsUsbListNet, [225](#)
- SetSubChannel
  - CMcsBus\_MotorControlNet, [137](#)



- SetSwitches
  - CSafeISDeviceNet, [397](#)
- SetSyncoutMap
  - CStg200xBasicNet, [445](#)
- SetTableName
  - CWarnerValveControllerDeviceNet, [551](#)
- SetTablepointer
  - CRetinaLedDeviceNet, [347](#)
- SetTableStep
  - CWarnerValveControllerDeviceNet, [551](#)
- SetTableStepAll
  - CWarnerValveControllerDeviceNet, [551](#)
- SetTestMode
  - CRFFunctionNet, [352](#)
- SetThermocoupleNanovoltPerKelvin
  - CFluidControlDeviceNet, [79](#)
  - CTcxDeviceNet, [489](#)
- SetThermoOffset
  - CMcsBus\_TempSensorNet, [150](#)
- SetTransformer
  - CMeFunctionNet, [282](#)
- SetTrigger
  - CRetinaLedDeviceNet, [347](#)
  - CWarnerValveControllerDeviceTesterFunctionNet, [559](#)
- SetTriggerMaskValue
  - CMeaDeviceNet, [268](#)
- SetTriggerPeriod
  - CMeaDeviceNet, [269](#)
- SetTriggerSource
  - CStg200xBasicNet, [445](#), [446](#)
- SetTriggerSyncDirection
  - CWarnerValveControllerDeviceTesterFunctionNet, [560](#)
- SetUByteBuffer
  - CGenericDevelopDeviceNet, [96](#)
- SetUClamp
  - CRoboDacqNet, [368](#)
- SetUOffset
  - CRoboDacqNet, [368](#)
- SetUIntBuffer
  - CGenericDevelopDeviceNet, [96](#)
- SetupGroupDacqQueue
  - CMcsUsbDacqNet, [201](#)
- SetupRetriggerMode
  - CStg200xDownloadBasicNet, [453](#)
- SetupTrigger
  - CStg200xDownloadBasicNet, [453](#)
  - CStimulusFunctionNet, [471](#)
- SetupTriggerSingle
  - CStg200xDownloadBasicNet, [454](#)
  - CStimulusFunctionNet, [471](#)
- SetUseBubble
  - CPPS\_FunctionNet, [337](#)
- SetUserParameter
  - CRoboDeviceNet::RoboMainLowLevelCommands, [591](#)
- SetUShortBuffer
  - CGenericDevelopDeviceNet, [97](#)
- SetUOffset
  - CRoboDacqNet, [368](#)
- SetValue
  - CGenericDevelopDeviceNet, [97](#)
- SetValve
  - CFluidControlDeviceNet, [79](#)
  - CRoboFluidDeviceNet, [386](#)
- SetValve1
  - CRobo\_FYIProgram\_FunctionNet, [353](#)
- SetValve2
  - CRobo\_FYIProgram\_FunctionNet, [354](#)
- SetValveActive
  - CPPCFunctionNet, [332](#)
  - CWarnerValveControllerDeviceNet, [551](#)
- SetValveDigitalInInvert
  - CWarnerValveControllerDeviceNet, [552](#)
- SetValveDigitalInPort
  - CWarnerValveControllerDeviceNet, [552](#)
- SetValveManualGroup
  - CWarnerValveControllerDeviceNet, [552](#)
- SetValveManualState
  - CWarnerValveControllerDeviceNet, [552](#)
- SetValveMode
  - CWarnerValveControllerDeviceNet, [553](#)
- SetValves
  - CMcsBus\_FYIExtensionNet, [120](#)
- SetValvesActiveMap
  - CWarnerValveControllerDeviceNet, [553](#)
- SetValvesManualStateMap
  - CWarnerValveControllerDeviceNet, [553](#)
- SetValveTableEntry
  - CWarnerValveControllerDeviceNet, [553](#)
- SetVelocityI
  - CRoboStatorDeviceNet, [394](#)
- SetVelocityXY
  - CRoboStatorDeviceNet, [394](#)
- SetVelocityZ
  - CRoboStatorDeviceNet, [394](#)
- SetVMMaxNegativeCurrent
  - CMcsBus\_VoltageModeNet, [153](#)
- SetVMMaxNegativeCurrentEeprom
  - CMcsBus\_VoltageModeNet, [153](#)
- SetVMMaxNegativeVoltage
  - CMcsBus\_VoltageModeNet, [153](#)
- SetVMMaxNegativeVoltageEeprom
  - CMcsBus\_VoltageModeNet, [153](#)
- SetVMMaxPositiveCurrent
  - CMcsBus\_VoltageModeNet, [154](#)
- SetVMMaxPositiveCurrentEeprom
  - CMcsBus\_VoltageModeNet, [154](#)
- SetVMMaxPositiveVoltage
  - CMcsBus\_VoltageModeNet, [154](#)
- SetVMMaxPositiveVoltageEeprom
  - CMcsBus\_VoltageModeNet, [154](#)
- SetVMOutputOnOff
  - CMcsBus\_VoltageModeNet, [154](#)
- SetVMVoltage

- CMcsBus\_VoltageModeNet, 154
- SetVoltage12VLimit
  - CRoboDeviceNet, 378
- SetVoltage5VLimit
  - CRoboDeviceNet, 378
- SetVoltageAirvalveLimit
  - CRoboDeviceNet, 378
- SetVoltageClampControllerParam\_D
  - CWarnerUssingFunctionNet, 532
- SetVoltageClampControllerParam\_I
  - CWarnerUssingFunctionNet, 533
- SetVoltageClampControllerParam\_P
  - CWarnerUssingFunctionNet, 533
- SetVoltageMode
  - CStg200xBasicNet, 446
- SetVoltageRangeByIndex
  - CMcsUsbDacqNet, 201
- SetVoltageRangeInMicroVolt
  - CMcsUsbDacqNet, 201
- SetVoltageRs485ALimit
  - CRoboDeviceNet, 378
- SetVoltageRs485BLimit
  - CRoboDeviceNet, 378
- SetVoltageValvesLimit
  - CRoboDeviceNet, 378
- SetWaveform
  - CChannelTestDeviceNet, 33
  - CTEERFunctionNet, 498
- SetWaveLengthInNanometer
  - CMultiwellOptoStimFunctionNet, 303
- SetWorkingFrequency
  - CRFFFunctionNet, 352
- SetWPADebugMode
  - CWClassicFunctionNet, 564
- SetWPAType
  - CWClassicFunctionNet, 564
- SetXGain
  - CRoboDacqNet, 368
- Sideband
  - CStimulusFunctionNet::SidebandData, 593
- SidebandData
  - CStimulusFunctionNet::SidebandData, 593
- SineStart
  - COctoPotDeviceNet, 308
- size
  - DigitalSource< digitalsourceenum >, 568
- SN
  - HeadStageIDType, 579
- Source
  - DigitalSource< digitalsourceenum >, 568
- Start
  - CMeaCleanDeviceNet, 256
  - CMeaCoatDeviceNet, 261
  - CRobo\_FYIPProgram\_FunctionNet, 354
- StartDacq
  - CMcsUsbDacqNet, 201–203
- StartInternalCalibration
  - CTEERFunctionNet, 498
- StartLoop
  - CMcsUsbDacqNet, 203, 204
- StartMCMovement
  - CMcsBus\_MotorControlNet, 138
- StartMeasurement
  - CMeaImpedanceDeviceNet, 277
- StartPoll
  - CStimulusFunctionNet, 473
  - CW2100\_StimulatorFunctionNet, 514
- StartSampling
  - CTEERFunctionNet, 498
- StartSync
  - CMcsBus\_SensorNet, 148
- State
  - HeadStageIDTypeState, 581
- Status
  - CUsbExceptionNet, 501
- Status\_AlreadyConfigured
  - CMcsUsbNet, 242
- Status\_BadStartFrame
  - CMcsUsbNet, 243
- Status\_Btstuff
  - CMcsUsbNet, 243
- Status\_BufferOverrun
  - CMcsUsbNet, 243
- Status\_BufferUnderrun
  - CMcsUsbNet, 243
- Status\_Canceled
  - CMcsUsbNet, 243
- Status\_Canceling
  - CMcsUsbNet, 243
- Status\_ConnectedPipes
  - CMcsUsbNet, 243
- Status\_ControlNotOwned
  - CMcsUsbNet, 243
- Status\_Crc
  - CMcsUsbNet, 243
- Status\_DataOverrun
  - CMcsUsbNet, 243
- Status\_DataToggleMismatch
  - CMcsUsbNet, 243
- Status\_DataUnderrun
  - CMcsUsbNet, 244
- Status\_DeviceLocked
  - CMcsUsbNet, 244
- Status\_DeviceNotFound
  - CMcsUsbNet, 244
- Status\_DeviceRemoved
  - CMcsUsbNet, 244
- Status\_DevNotResponding
  - CMcsUsbNet, 244
- Status\_EndpointHalted
  - CMcsUsbNet, 244
- Status\_ErrorBusy
  - CMcsUsbNet, 244
- Status\_ErrorShortTransfer
  - CMcsUsbNet, 244
- Status\_Fifo



- CMcsUsbNet, [244](#)
- Status\_FrameControlOwned
  - CMcsUsbNet, [244](#)
- Status\_InternalHcError
  - CMcsUsbNet, [244](#)
- Status\_InvalidParameter
  - CMcsUsbNet, [245](#)
- Status\_InvalidPipeHandle
  - CMcsUsbNet, [245](#)
- Status\_InvalidUrbFunction
  - CMcsUsbNet, [245](#)
- Status\_IoPending
  - CMcsUsbNet, [245](#)
- Status\_IoTimeout
  - CMcsUsbNet, [245](#)
- Status\_IsochRequestFailed
  - CMcsUsbNet, [245](#)
- Status\_LastUsbErrorMismatch
  - CMcsUsbNet, [245](#)
- Status\_NoBandwidth
  - CMcsUsbNet, [245](#)
- Status\_NoMemory
  - CMcsUsbNet, [245](#)
- Status\_NoSuchDevice
  - CMcsUsbNet, [245](#)
- Status\_NotAccessed
  - CMcsUsbNet, [245](#)
- Status\_NotSupported
  - CMcsUsbNet, [246](#)
- Status\_PidCheckFailure
  - CMcsUsbNet, [246](#)
- Status\_PipeNotLinked
  - CMcsUsbNet, [246](#)
- Status\_RequestFailed
  - CMcsUsbNet, [246](#)
- Status\_RequestMutexFailed
  - CMcsUsbNet, [246](#)
- Status\_RequestMutexTimeout
  - CMcsUsbNet, [246](#)
- Status\_Stall
  - CMcsUsbNet, [246](#)
- Status\_Unconfigured
  - CMcsUsbNet, [246](#)
- Status\_UnexpectedPid
  - CMcsUsbNet, [246](#)
- Stg200xPollStatusEvent
  - CStg200xDownloadNet, [461](#)
- StgStatusNet, [593](#)
  - FromIntPtr, [594](#)
  - FromPtr, [594](#)
  - ListOfChangedTriggers, [594](#)
  - TiggerStatus, [594](#)
- StillConnected
  - CRadioControlledDevicesNet, [344](#)
- Stimulator
  - CW2100\_FunctionNet, [509](#)
- Stimulus
  - CCMOSMeaDeviceNet, [47](#)
  - CStg200xDownloadBasicNet, [455](#)
- StimulusDeviceDataAndUnrolledData
  - CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, [594](#)
- StimulusFunction
  - CLIH3DeviceNet, [116](#)
- StimulusParameters
  - HeadStageIDType, [579](#)
- Stop
  - CMeaCleanDeviceNet, [256](#)
  - CMeaCoatDeviceNet, [262](#)
- StopDacq
  - CMcsUsbDacqNet, [205](#)
- StopLoop
  - CMcsUsbDacqNet, [206](#)
- StopMCMovement
  - CMcsBus\_MotorControlNet, [138](#)
- StopMovement
  - CRoboDeviceNet, [378](#)
- StopMovementI
  - CRoboStatorDeviceNet, [394](#)
- StopMovementXY
  - CRoboStatorDeviceNet, [394](#)
- StopMovementZ
  - CRoboStatorDeviceNet, [394](#)
- StopPlateClamp
  - CMultiwellDeviceNet, [298](#)
- StopPoll
  - CStimulusFunctionNet, [473](#)
  - CW2100\_StimulatorFunctionNet, [515](#)
- StopSampling
  - CTEERFunctionNet, [498](#)
- StopTable
  - CRoboDacqNet, [368](#)
- StoreValveTable
  - CWarnerValveControllerDeviceNet, [554](#)
- SwitchOnOff
  - CPositionIIDeviceNet, [321](#)
- SYNC\_BIT0
  - CW2100\_StimulatorFunctionNet, [515](#)
- SYNC\_BIT1
  - CW2100\_StimulatorFunctionNet, [515](#)
- Table\_Wait
  - CRoboDacqNet, [369](#)
- TableDefBegin
  - CRoboDacqNet, [369](#)
- TableDefEnd
  - CRoboDacqNet, [369](#)
- TableEntryChangedEvent
  - CWarnerValveControllerDeviceNet, [557](#)
- TactSwitchGetState
  - CMcsBus\_SensorNet, [148](#)
- TactSwitchSetDisplay
  - CMcsBus\_SensorNet, [148](#)
- TEERFunctionNet
  - CTEERMachineDeviceNet, [499](#)
- ThrowCUsbExceptionNetOnError
  - CMcsUsbFunctionNet, [218](#)

- CMcsUsbNet, 240
- TiggerStatus
  - StgStatusNet, 594
- TimeResolutionInNanoSeconds
  - W2100\_StimulusParametersNet, 597
- ToString
  - CFilterPropertyNet, 72
  - CMcsUsbListEntryNet, 222
  - HeadStageIDType, 578
  - HeadstageIDTypeObject, 580
- TriggerStatus
  - CMcsUsbDeviceStatePushFunctionNet, 208
  - CMcsUsbDeviceStatePushNet, 209
- TxnGetSerialNumber
  - CMcsUsbNet, 240
- TxnSetSerialNumber
  - CMcsUsbNet, 240
- TxnTestMemoryReadAndCheck
  - CMcsUsbNet, 240
- TxnTestMemoryWrite
  - CMcsUsbNet, 241
- Type
  - HeadStageIDType, 579
- TypeValue
  - HeadStageIDType, 579
- UIntA1
  - CFilterCoefficientsNet, 67
- UIntA2
  - CFilterCoefficientsNet, 68
- UIntB0
  - CFilterCoefficientsNet, 68
- UIntB1
  - CFilterCoefficientsNet, 68
- UIntB2
  - CFilterCoefficientsNet, 68
- Unknown
  - HeadStageIDType, 578
- unknown
  - Mcs::Usb, 27
- UnlockPlateClamp
  - CMultiwellDeviceNet, 298
- UnrolledAmplitude
  - CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, 595
- UnrolledDuration
  - CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, 595
- UnrolledSync
  - CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, 595
- UpdateChannelBlock
  - CCMOSMeaDeviceNet, 47
- UpdateDisplay
  - CRoboDacqNet, 369
- UpdateFirmware
  - CMcsUsbFactoryNet, 215, 216
- UpdateTransistorVoltages
  - CCMOSMea\_FunctionNet, 44
- USB
  - FirmwareDestinationNames, 577
  - usbSetupPacket\_t, 595
    - bmRequestType, 596
    - bRequest, 596
    - wIndex, 596
    - wLength, 596
    - wValue, 596
  - UserDefinedName
    - HeadStageIDType, 579
  - Valid
    - HeadStageIDType, 579
  - ValidKey
    - CMcsUsbNet, 241
  - Voltage
    - BatteryState, 30
  - VoltageRangeDisplayStringMilliVolt
    - CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet, 501
  - VoltageRangeInMicroVolt
    - CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet, 501
    - W2100\_StimulusParametersNet, 597
  - VoltageResolutionInMicroVolt
    - W2100\_StimulusParametersNet, 597
  - VoltageString
    - BatteryState, 30
  - VOPSTimerSetResetTimes
    - CCMOSMea\_FunctionNet, 44
  - W16lsW14
    - HeadStageIDType, 579
  - W2100\_FunctionNet
    - CMeaDeviceNet, 270
  - W2100\_StimulusParametersNet, 596
    - CurrentRangeInNanoAmp, 596
    - CurrentResolutionInNanoAmp, 596
    - DACResolution, 596
    - TimeResolutionInNanoSeconds, 597
    - VoltageRangeInMicroVolt, 597
    - VoltageResolutionInMicroVolt, 597
  - WaitForAllChambers
    - CWarnerUssingFunctionNet, 533
  - WaitForChamber
    - CWarnerUssingFunctionNet, 533
  - WaitForUssingFunction
    - CWarnerUssingDeviceNet, 517
  - WClassicFunctionNet
    - CMeaDeviceNet, 270
  - wIndex
    - usbSetupPacket\_t, 596
  - wLength
    - usbSetupPacket\_t, 596
  - WPAError\_ScanningIsPending
    - CMcsUsbNet, 246
  - Write
    - CExternDTesterDeviceNet, 64
  - Write2

- CExternDTesterDeviceNet, [65](#)
- WriteEepromRegisterPreconfig
  - CMcsUsbNet, [241](#)
- WritePipe
  - CGenericDevelopDeviceNet, [98](#)
- WriteRegister
  - CMcsUsbNet, [241](#), [242](#)
- WriteRegister32
  - CMcsUsbNet, [242](#)
- WriteRegisterArray
  - CMcsUsbNet, [242](#)
- WriteRegisterTimeSlot
  - CMcsUsbNet, [242](#)
- WriteRegisterValue
  - CMcsUsbNet, [242](#)
- wValue
  - usbSetupPacket\_t, [596](#)