

# multichannel\* systems

McsUsbNet.dll

Version 5.1.13

---

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	1
1.3 Connecting to an MCS device	1
<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	2
2.2 MCS-USB-Sw2to64 device	3
<b>3 Function Classes</b>	<b>3</b>
<b>4 Data ACQuisition (DACQ) Devices</b>	<b>4</b>
<b>5 The MCS Robo Device</b>	<b>5</b>
5.1 Introduction	5
<b>6 STG200x &amp; STG400x STimulus Generator</b>	<b>5</b>
6.1 Introduction	5
6.2 Download mode	6
6.2.1 Memory Layout and Trigger Setup	6
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	27
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	30
11.3 BesselFilterHighPassNet Class Reference	30

11.3.1 Constructor & Destructor Documentation	30
11.4 BesselFilterLowPassNet Class Reference	31
11.4.1 Constructor & Destructor Documentation	31
11.5 ButterworthFilterHighPassNet Class Reference	31
11.5.1 Constructor & Destructor Documentation	32
11.6 ButterworthFilterLowPassNet Class Reference	32
11.6.1 Constructor & Destructor Documentation	32
11.7 CChannelTestDeviceNet Class Reference	33
11.7.1 Constructor & Destructor Documentation	33
11.7.2 Member Function Documentation	33
11.8 CCMOSMea_FunctionNet Class Reference	34
11.8.1 Constructor & Destructor Documentation	36
11.8.2 Member Function Documentation	36
11.9 CCMOSMeaDeviceNet Class Reference	45
11.9.1 Constructor & Destructor Documentation	46
11.9.2 Member Function Documentation	46
11.9.3 Property Documentation	47
11.10 CCreateFilterNet Class Reference	48
11.10.1 Constructor & Destructor Documentation	48
11.10.2 Member Function Documentation	49
11.10.3 Property Documentation	49
11.11 CDacCalibrationFunctionNet Class Reference	50
11.11.1 Detailed Description	50
11.11.2 Constructor & Destructor Documentation	50
11.11.3 Member Function Documentation	51
11.12 CDacqGroupChannelGenericSelectionNet Class Reference	52
11.12.1 Constructor & Destructor Documentation	52
11.13 CDacqGroupChannelSelectionNet Class Reference	52
11.13.1 Constructor & Destructor Documentation	53
11.14 CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet > Class Template Reference	53
11.14.1 Constructor & Destructor Documentation	54
11.14.2 Member Function Documentation	54
11.15 CDeviceGroupChannelInfoGenericNet Class Reference	56
11.15.1 Constructor & Destructor Documentation	56
11.16 CDeviceGroupChannelInfoMEA2100_256Net Class Reference	56
11.16.1 Constructor & Destructor Documentation	57
11.17 CDeviceGroupChannelInfoNet Class Reference	57
11.17.1 Constructor & Destructor Documentation	57
11.18 CDeviceGroupChannelInfoSCUNet Class Reference	57
11.18.1 Constructor & Destructor Documentation	58
11.19 CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumTemplateNet > Class Template Reference	58

---

11.19.1 Constructor & Destructor Documentation	58
11.19.2 Member Data Documentation	58
11.20 CDeviceGroupChannelInfoW2100Net Class Reference	59
11.20.1 Constructor & Destructor Documentation	59
11.21 CDigOutStimulatorFunctionNet Class Reference	59
11.21.1 Detailed Description	60
11.21.2 Constructor & Destructor Documentation	60
11.21.3 Member Function Documentation	61
11.22 CEncapsulatorDeviceNet Class Reference	64
11.22.1 Detailed Description	64
11.22.2 Constructor & Destructor Documentation	64
11.22.3 Member Function Documentation	64
11.23 CExternDTesterDeviceNet Class Reference	64
11.23.1 Detailed Description	65
11.23.2 Constructor & Destructor Documentation	65
11.23.3 Member Function Documentation	65
11.24 CFilterCoefficientsNet Class Reference	66
11.24.1 Constructor & Destructor Documentation	67
11.24.2 Member Function Documentation	68
11.24.3 Property Documentation	68
11.25 CFilterConfigurationNet Class Reference	68
11.25.1 Constructor & Destructor Documentation	69
11.25.2 Member Function Documentation	69
11.26 CFilterConfigurationRegisterNet Class Reference	70
11.26.1 Constructor & Destructor Documentation	71
11.26.2 Member Function Documentation	71
11.27 CFilterPropertyNet Class Reference	72
11.27.1 Constructor & Destructor Documentation	73
11.27.2 Member Function Documentation	73
11.27.3 Property Documentation	73
11.28 CFluidControlDeviceNet Class Reference	74
11.28.1 Detailed Description	75
11.28.2 Constructor & Destructor Documentation	75
11.28.3 Member Function Documentation	75
11.28.4 Property Documentation	80
11.29 CFYIDeviceNet Class Reference	80
11.29.1 Detailed Description	80
11.29.2 Constructor & Destructor Documentation	80
11.29.3 Property Documentation	81
11.30 CGenericDevelopDeviceNet Class Reference	81
11.30.1 Detailed Description	88
11.30.2 Constructor & Destructor Documentation	88

---

11.30.3 Member Function Documentation	88
11.31 CGilsonDeviceNet Class Reference	98
11.31.1 Detailed Description	99
11.31.2 Constructor & Destructor Documentation	99
11.31.3 Member Function Documentation	99
11.31.4 Member Data Documentation	100
11.32 CGrapheneFunctionNet Class Reference	100
11.32.1 Detailed Description	102
11.32.2 Constructor & Destructor Documentation	102
11.32.3 Member Function Documentation	102
11.33 CHiClampDeviceNet Class Reference	109
11.33.1 Detailed Description	110
11.33.2 Constructor & Destructor Documentation	110
11.33.3 Property Documentation	110
11.34 CHLADacqNet Class Reference	110
11.34.1 Constructor & Destructor Documentation	111
11.35 CHLADeviceNet Class Reference	111
11.35.1 Detailed Description	111
11.35.2 Constructor & Destructor Documentation	111
11.35.3 Property Documentation	112
11.36 CMcsUsbDacqNet::CHWInfo Class Reference	112
11.36.1 Detailed Description	112
11.36.2 Constructor & Destructor Documentation	113
11.36.3 Member Function Documentation	113
11.37 CIntanMea_FunctionNet Class Reference	114
11.37.1 Constructor & Destructor Documentation	115
11.37.2 Member Function Documentation	115
11.38 CInterfaceboard2FunctionNet Class Reference	116
11.38.1 Detailed Description	117
11.38.2 Constructor & Destructor Documentation	117
11.38.3 Member Function Documentation	117
11.39 CInterfaceboardFunctionNet Class Reference	118
11.39.1 Detailed Description	119
11.39.2 Constructor & Destructor Documentation	119
11.39.3 Member Function Documentation	119
11.40 CLIH3DeviceNet Class Reference	120
11.40.1 Detailed Description	122
11.40.2 Constructor & Destructor Documentation	122
11.40.3 Member Function Documentation	122
11.40.4 Property Documentation	129
11.41 CMcsBus_AxisParametersNet Class Reference	129
11.41.1 Constructor & Destructor Documentation	129

11.41.2 Member Function Documentation . . . . .	130
11.42 CMcsBus_ExtensionNet Class Reference . . . . .	130
11.42.1 Constructor & Destructor Documentation . . . . .	131
11.42.2 Member Function Documentation . . . . .	131
11.43 CMcsBus_FYIExtensionNet Class Reference . . . . .	131
11.43.1 Constructor & Destructor Documentation . . . . .	132
11.43.2 Member Function Documentation . . . . .	132
11.44 CMcsBus_MotorControlNet Class Reference . . . . .	133
11.44.1 Constructor & Destructor Documentation . . . . .	136
11.44.2 Member Function Documentation . . . . .	136
11.45 CMcsBus_SensorNet Class Reference . . . . .	150
11.45.1 Constructor & Destructor Documentation . . . . .	152
11.45.2 Member Function Documentation . . . . .	152
11.46 CMcsBus_TempSensorNet Class Reference . . . . .	161
11.46.1 Constructor & Destructor Documentation . . . . .	161
11.46.2 Member Function Documentation . . . . .	162
11.47 CMcsBus_VoltageModeNet Class Reference . . . . .	163
11.47.1 Constructor & Destructor Documentation . . . . .	164
11.47.2 Member Function Documentation . . . . .	164
11.48 CMcsBusNet Class Reference . . . . .	167
11.48.1 Constructor & Destructor Documentation . . . . .	168
11.48.2 Member Function Documentation . . . . .	168
11.49 CMcsUsbDacqNet Class Reference . . . . .	171
11.49.1 Detailed Description . . . . .	177
11.49.2 Constructor & Destructor Documentation . . . . .	177
11.49.3 Member Function Documentation . . . . .	177
11.49.4 Member Data Documentation . . . . .	218
11.49.5 Property Documentation . . . . .	219
11.49.6 Event Documentation . . . . .	219
11.50 CMcsUsbDeviceStatePushFunctionNet Class Reference . . . . .	219
11.50.1 Constructor & Destructor Documentation . . . . .	220
11.50.2 Member Function Documentation . . . . .	220
11.50.3 Event Documentation . . . . .	220
11.51 CMcsUsbDeviceStatePushNet Class Reference . . . . .	220
11.51.1 Constructor & Destructor Documentation . . . . .	221
11.51.2 Member Function Documentation . . . . .	221
11.51.3 Event Documentation . . . . .	221
11.52 CMcsUsbFactoryNet Class Reference . . . . .	221
11.52.1 Constructor & Destructor Documentation . . . . .	223
11.52.2 Member Function Documentation . . . . .	223
11.52.3 Member Data Documentation . . . . .	229
11.53 CMcsUsbFunctionNet Class Reference . . . . .	230

11.53.1 Constructor & Destructor Documentation	230
11.53.2 Member Function Documentation	231
11.53.3 Member Data Documentation	231
11.54 CMcsUsbFunctionPointerContainer Class Reference	231
11.55 CMcsUsbListEntryNet Class Reference	231
11.55.1 Detailed Description	232
11.55.2 Constructor & Destructor Documentation	232
11.55.3 Member Function Documentation	233
11.55.4 Property Documentation	236
11.56 CMcsUsbListNet Class Reference	237
11.56.1 Detailed Description	237
11.56.2 Constructor & Destructor Documentation	237
11.56.3 Member Function Documentation	238
11.56.4 Property Documentation	239
11.56.5 Event Documentation	239
11.57 CMcsUsbNet Class Reference	240
11.57.1 Detailed Description	244
11.57.2 Constructor & Destructor Documentation	244
11.57.3 Member Function Documentation	244
11.57.4 Member Data Documentation	256
11.57.5 Property Documentation	261
11.58 CMcsUsbPointerContainer Class Reference	261
11.59 CMEA2100_256DacqGroupChannelSelectionNet Class Reference	261
11.59.1 Constructor & Destructor Documentation	261
11.60 CMEA2100x256FunctionNet Class Reference	262
11.60.1 Detailed Description	262
11.60.2 Constructor & Destructor Documentation	262
11.60.3 Member Function Documentation	263
11.61 CMeaAudioFunctionNet Class Reference	263
11.61.1 Constructor & Destructor Documentation	264
11.61.2 Member Function Documentation	264
11.62 CMeaCleanDeviceNet Class Reference	266
11.62.1 Detailed Description	267
11.62.2 Constructor & Destructor Documentation	267
11.62.3 Member Function Documentation	268
11.63 CMeaCoatDeviceNet Class Reference	270
11.63.1 Detailed Description	271
11.63.2 Constructor & Destructor Documentation	272
11.63.3 Member Function Documentation	272
11.64 CMeaDeviceNet Class Reference	276
11.64.1 Detailed Description	277
11.64.2 Constructor & Destructor Documentation	277

---

11.64.3 Member Function Documentation . . . . .	278
11.64.4 Property Documentation . . . . .	283
11.65 CMeaDigitalDataFunctionNet Class Reference . . . . .	284
11.65.1 Constructor & Destructor Documentation . . . . .	284
11.65.2 Member Function Documentation . . . . .	285
11.66 CMeaFeedbackFunctionNet Class Reference . . . . .	286
11.66.1 Constructor & Destructor Documentation . . . . .	287
11.66.2 Member Function Documentation . . . . .	287
11.67 CMeaImpedanceDeviceNet Class Reference . . . . .	290
11.67.1 Constructor & Destructor Documentation . . . . .	290
11.67.2 Member Function Documentation . . . . .	291
11.68 CMeasureTableDeviceNet Class Reference . . . . .	291
11.68.1 Detailed Description . . . . .	292
11.68.2 Constructor & Destructor Documentation . . . . .	292
11.68.3 Property Documentation . . . . .	292
11.69 CMeaSwitchDeviceNet Class Reference . . . . .	292
11.69.1 Detailed Description . . . . .	293
11.69.2 Constructor & Destructor Documentation . . . . .	293
11.69.3 Member Function Documentation . . . . .	293
11.70 CMeaUSBDeviceNet Class Reference . . . . .	294
11.70.1 Detailed Description . . . . .	295
11.70.2 Constructor & Destructor Documentation . . . . .	295
11.71 CMeFunctionNet Class Reference . . . . .	295
11.71.1 Detailed Description . . . . .	296
11.71.2 Constructor & Destructor Documentation . . . . .	296
11.71.3 Member Function Documentation . . . . .	296
11.72 CMultiBatteryChargerDeviceNet Class Reference . . . . .	297
11.72.1 Detailed Description . . . . .	298
11.72.2 Constructor & Destructor Documentation . . . . .	298
11.72.3 Member Function Documentation . . . . .	298
11.73 CMultiwellCallbackFunctionNet Class Reference . . . . .	304
11.73.1 Detailed Description . . . . .	304
11.73.2 Constructor & Destructor Documentation . . . . .	305
11.73.3 Member Function Documentation . . . . .	305
11.73.4 Event Documentation . . . . .	306
11.74 CMultiwellDeviceNet Class Reference . . . . .	306
11.74.1 Detailed Description . . . . .	307
11.74.2 Constructor & Destructor Documentation . . . . .	307
11.74.3 Member Function Documentation . . . . .	308
11.75 CMultiwellOptoStimFunctionNet Class Reference . . . . .	312
11.75.1 Detailed Description . . . . .	313
11.75.2 Constructor & Destructor Documentation . . . . .	313



11.75.3 Member Function Documentation	313
11.76 CNF_GenDeviceNet Class Reference	317
11.76.1 Constructor & Destructor Documentation	317
11.76.2 Member Function Documentation	317
11.77 COctoPotDeviceNet Class Reference	318
11.77.1 Constructor & Destructor Documentation	318
11.77.2 Member Function Documentation	319
11.78 COkuvisionStimulatorDeviceNet Class Reference	322
11.78.1 Constructor & Destructor Documentation	323
11.78.2 Member Function Documentation	323
11.79 CPatchServerDeviceNet Class Reference	325
11.79.1 Detailed Description	326
11.79.2 Constructor & Destructor Documentation	326
11.79.3 Property Documentation	326
11.80 CPathIdentDeviceNet Class Reference	326
11.80.1 Constructor & Destructor Documentation	327
11.80.2 Member Function Documentation	327
11.81 CPedoterDeviceNet Class Reference	327
11.81.1 Detailed Description	328
11.81.2 Constructor & Destructor Documentation	328
11.81.3 Member Function Documentation	328
11.82 CPeristalticPumpDeviceNet Class Reference	329
11.82.1 Detailed Description	329
11.82.2 Constructor & Destructor Documentation	329
11.82.3 Property Documentation	330
11.83 CPgaDeviceNet Class Reference	330
11.83.1 Constructor & Destructor Documentation	330
11.83.2 Member Function Documentation	331
11.84 CPositionIIDeviceNet Class Reference	332
11.84.1 Detailed Description	334
11.84.2 Constructor & Destructor Documentation	334
11.84.3 Member Function Documentation	334
11.84.4 Property Documentation	341
11.85 CPositionImpDeviceNet Class Reference	341
11.85.1 Detailed Description	341
11.85.2 Constructor & Destructor Documentation	342
11.85.3 Member Function Documentation	342
11.86 CPPCDeviceNet Class Reference	344
11.86.1 Constructor & Destructor Documentation	344
11.86.2 Property Documentation	345
11.87 CPPCFunctionNet Class Reference	345
11.87.1 Detailed Description	346

---

11.87.2 Constructor & Destructor Documentation . . . . .	346
11.87.3 Member Function Documentation . . . . .	347
11.88 CPPS_DeviceNet Class Reference . . . . .	354
11.88.1 Constructor & Destructor Documentation . . . . .	354
11.88.2 Property Documentation . . . . .	354
11.89 CPPS_FunctionNet Class Reference . . . . .	355
11.89.1 Constructor & Destructor Documentation . . . . .	355
11.89.2 Member Function Documentation . . . . .	356
11.90 CPPSDeviceNet Class Reference . . . . .	359
11.90.1 Detailed Description . . . . .	359
11.90.2 Constructor & Destructor Documentation . . . . .	360
11.91 CProgramPressureCurveNet Class Reference . . . . .	360
11.91.1 Detailed Description . . . . .	360
11.91.2 Constructor & Destructor Documentation . . . . .	360
11.91.3 Member Function Documentation . . . . .	361
11.92 CPulseGeneratorFunctionNet Class Reference . . . . .	361
11.92.1 Detailed Description . . . . .	362
11.92.2 Constructor & Destructor Documentation . . . . .	362
11.92.3 Member Function Documentation . . . . .	363
11.93 CRadioControlledDevicesNet Class Reference . . . . .	364
11.93.1 Constructor & Destructor Documentation . . . . .	365
11.93.2 Member Function Documentation . . . . .	365
11.94 CCMOSMeaDeviceNet::CRegionOfInterestRect Class Reference . . . . .	366
11.94.1 Constructor & Destructor Documentation . . . . .	366
11.94.2 Member Function Documentation . . . . .	366
11.94.3 Member Data Documentation . . . . .	367
11.95 CRetinaLedDeviceNet Class Reference . . . . .	367
11.95.1 Constructor & Destructor Documentation . . . . .	368
11.95.2 Member Function Documentation . . . . .	368
11.96 CRFFFunctionNet Class Reference . . . . .	369
11.96.1 Detailed Description . . . . .	370
11.96.2 Constructor & Destructor Documentation . . . . .	370
11.96.3 Member Function Documentation . . . . .	370
11.97 CRobo_FYIPProgram_FunctionNet Class Reference . . . . .	374
11.97.1 Constructor & Destructor Documentation . . . . .	374
11.97.2 Member Function Documentation . . . . .	374
11.98 CRobo_FYITemp_FunctionNet Class Reference . . . . .	375
11.98.1 Constructor & Destructor Documentation . . . . .	376
11.98.2 Member Function Documentation . . . . .	376
11.99 CRoboDacqNet Class Reference . . . . .	377
11.99.1 Constructor & Destructor Documentation . . . . .	380
11.99.2 Member Function Documentation . . . . .	380

11.100 CRoboDeviceNet Class Reference . . . . .	391
11.100.1 Detailed Description . . . . .	394
11.100.2 Constructor & Destructor Documentation . . . . .	394
11.100.3 Member Function Documentation . . . . .	394
11.100.4 Member Data Documentation . . . . .	401
11.100.5 Property Documentation . . . . .	405
11.100.6 Event Documentation . . . . .	405
11.101 CRoboFluidDeviceNet Class Reference . . . . .	405
11.101.1 Constructor & Destructor Documentation . . . . .	406
11.101.2 Member Function Documentation . . . . .	406
11.101.3 Member Data Documentation . . . . .	408
11.101.4 Property Documentation . . . . .	408
11.102 CRoboInjectDeviceNet Class Reference . . . . .	409
11.102.1 Detailed Description . . . . .	409
11.102.2 Constructor & Destructor Documentation . . . . .	409
11.103 CRoboocyte2DeviceNet Class Reference . . . . .	409
11.103.1 Detailed Description . . . . .	410
11.103.2 Constructor & Destructor Documentation . . . . .	410
11.103.3 Member Function Documentation . . . . .	410
11.104 CRoboStatorDeviceNet Class Reference . . . . .	411
11.104.1 Constructor & Destructor Documentation . . . . .	412
11.104.2 Member Function Documentation . . . . .	412
11.104.3 Property Documentation . . . . .	417
11.105 CSafeISDeviceNet Class Reference . . . . .	417
11.105.1 Detailed Description . . . . .	418
11.105.2 Constructor & Destructor Documentation . . . . .	418
11.105.3 Member Function Documentation . . . . .	418
11.105.4 Property Documentation . . . . .	419
11.106 CSCUDacqGroupChannelSelectionNet Class Reference . . . . .	420
11.106.1 Constructor & Destructor Documentation . . . . .	420
11.107 CSCUFunctionNet Class Reference . . . . .	421
11.107.1 Detailed Description . . . . .	423
11.107.2 Constructor & Destructor Documentation . . . . .	423
11.107.3 Member Function Documentation . . . . .	423
11.107.4 Event Documentation . . . . .	435
11.108 CSerialPortNet Class Reference . . . . .	436
11.108.1 Constructor & Destructor Documentation . . . . .	436
11.108.2 Member Function Documentation . . . . .	436
11.109 CStg200xBasicNet Class Reference . . . . .	437
11.109.1 Detailed Description . . . . .	441
11.109.2 Constructor & Destructor Documentation . . . . .	441
11.109.3 Member Function Documentation . . . . .	441

11.110 CStg200xDownloadBasicNet Class Reference . . . . .	467
11.110.1 Detailed Description . . . . .	469
11.110.2 Member Function Documentation . . . . .	469
11.110.3 Property Documentation . . . . .	476
11.111 CStg200xDownloadNet Class Reference . . . . .	476
11.111.1 Detailed Description . . . . .	477
11.111.2 Constructor & Destructor Documentation . . . . .	477
11.111.3 Member Function Documentation . . . . .	478
11.111.4 Event Documentation . . . . .	482
11.112 CStimulusFunctionNet Class Reference . . . . .	483
11.112.1 Constructor & Destructor Documentation . . . . .	484
11.112.2 Member Function Documentation . . . . .	485
11.112.3 Event Documentation . . . . .	494
11.113 CSw2to64DeviceNet Class Reference . . . . .	494
11.113.1 Detailed Description . . . . .	495
11.113.2 Constructor & Destructor Documentation . . . . .	495
11.113.3 Member Function Documentation . . . . .	495
11.114 CTcxDeviceNet Class Reference . . . . .	497
11.114.1 Detailed Description . . . . .	499
11.114.2 Constructor & Destructor Documentation . . . . .	499
11.114.3 Member Function Documentation . . . . .	499
11.115 CTEERFunctionNet Class Reference . . . . .	510
11.115.1 Detailed Description . . . . .	512
11.115.2 Constructor & Destructor Documentation . . . . .	512
11.115.3 Member Function Documentation . . . . .	512
11.116 CTEERMachineDeviceNet Class Reference . . . . .	520
11.116.1 Constructor & Destructor Documentation . . . . .	521
11.116.2 Property Documentation . . . . .	521
11.117 CUsbDeviceConfigurationFunctionNet Class Reference . . . . .	521
11.117.1 Detailed Description . . . . .	522
11.117.2 Constructor & Destructor Documentation . . . . .	522
11.117.3 Member Function Documentation . . . . .	522
11.118 CUsbExceptionNet Class Reference . . . . .	523
11.118.1 Detailed Description . . . . .	523
11.118.2 Constructor & Destructor Documentation . . . . .	523
11.118.3 Property Documentation . . . . .	524
11.119 CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet Class Reference . . . . .	524
11.119.1 Constructor & Destructor Documentation . . . . .	524
11.119.2 Member Data Documentation . . . . .	524
11.120 CW2100_FunctionNet Class Reference . . . . .	525
11.120.1 Constructor & Destructor Documentation . . . . .	526
11.120.2 Member Function Documentation . . . . .	527

11.120.3 Property Documentation . . . . .	532
11.121 CW2100_StimulatorFunctionNet Class Reference . . . . .	532
11.121.1 Constructor & Destructor Documentation . . . . .	534
11.121.2 Member Function Documentation . . . . .	534
11.121.3 Member Data Documentation . . . . .	539
11.121.4 Event Documentation . . . . .	540
11.122 CW2100DacqGroupChannelSelectionNet Class Reference . . . . .	540
11.122.1 Constructor & Destructor Documentation . . . . .	540
11.123 CWarnerUssingDeviceNet Class Reference . . . . .	541
11.123.1 Detailed Description . . . . .	541
11.123.2 Constructor & Destructor Documentation . . . . .	541
11.123.3 Property Documentation . . . . .	542
11.124 CWarnerUssingFunctionNet Class Reference . . . . .	542
11.124.1 Detailed Description . . . . .	544
11.124.2 Constructor & Destructor Documentation . . . . .	544
11.124.3 Member Function Documentation . . . . .	544
11.125 CWarnerValveControllerDeviceNet Class Reference . . . . .	558
11.125.1 Detailed Description . . . . .	562
11.125.2 Constructor & Destructor Documentation . . . . .	562
11.125.3 Member Function Documentation . . . . .	563
11.125.4 Event Documentation . . . . .	580
11.126 CWarnerValveControllerDeviceTesterFunctionNet Class Reference . . . . .	583
11.126.1 Detailed Description . . . . .	584
11.126.2 Constructor & Destructor Documentation . . . . .	584
11.126.3 Member Function Documentation . . . . .	584
11.127 CWClassicFunctionNet Class Reference . . . . .	586
11.127.1 Constructor & Destructor Documentation . . . . .	587
11.127.2 Member Function Documentation . . . . .	587
11.128 CWirelessBaseFunctionNet Class Reference . . . . .	591
11.128.1 Constructor & Destructor Documentation . . . . .	591
11.128.2 Member Function Documentation . . . . .	591
11.129 DeviceIdNet Struct Reference . . . . .	591
11.129.1 Detailed Description . . . . .	592
11.129.2 Constructor & Destructor Documentation . . . . .	592
11.129.3 Member Function Documentation . . . . .	592
11.129.4 Member Data Documentation . . . . .	593
11.130 DigitalSource< digitalsourceenum > Class Template Reference . . . . .	593
11.130.1 Constructor & Destructor Documentation . . . . .	593
11.130.2 Member Function Documentation . . . . .	594
11.130.3 Property Documentation . . . . .	594
11.131 DigitalSourceGeneral Class Reference . . . . .	594
11.131.1 Constructor & Destructor Documentation . . . . .	595

11.131.2 Member Function Documentation . . . . .	595
11.131.3 Property Documentation . . . . .	595
11.132 DriverVersionNet Class Reference . . . . .	596
11.132.1 Detailed Description . . . . .	597
11.132.2 Constructor & Destructor Documentation . . . . .	597
11.132.3 Member Function Documentation . . . . .	597
11.133 FirmwareDestinationNames Class Reference . . . . .	601
11.133.1 Member Data Documentation . . . . .	601
11.134 HeadStageIDType Class Reference . . . . .	604
11.134.1 Member Enumeration Documentation . . . . .	605
11.134.2 Constructor & Destructor Documentation . . . . .	605
11.134.3 Member Function Documentation . . . . .	605
11.134.4 Property Documentation . . . . .	606
11.135 HeadstageIDTypeObject Class Reference . . . . .	607
11.135.1 Constructor & Destructor Documentation . . . . .	607
11.135.2 Member Function Documentation . . . . .	608
11.135.3 Member Data Documentation . . . . .	608
11.135.4 Property Documentation . . . . .	608
11.136 HeadStageIDTypeState Class Reference . . . . .	608
11.136.1 Property Documentation . . . . .	609
11.137 mkfilterNet Class Reference . . . . .	609
11.137.1 Member Function Documentation . . . . .	610
11.138 CRoboDeviceNet::RoboMainLowLevelCommands Class Reference . . . . .	612
11.138.1 Member Function Documentation . . . . .	613
11.139 CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands Class Reference . . . . .	619
11.139.1 Member Function Documentation . . . . .	619
11.140 CFilterCoefficientsNet::s_FilterAttributesNet Struct Reference . . . . .	619
11.140.1 Constructor & Destructor Documentation . . . . .	620
11.140.2 Member Function Documentation . . . . .	620
11.140.3 Member Data Documentation . . . . .	620
11.141 CMeaAudioFunctionNet::s_setaudionet Struct Reference . . . . .	621
11.141.1 Member Data Documentation . . . . .	621
11.142 CStimulusFunctionNet::SidebandData Class Reference . . . . .	621
11.142.1 Constructor & Destructor Documentation . . . . .	621
11.142.2 Property Documentation . . . . .	622
11.143 StgStatusNet Class Reference . . . . .	622
11.143.1 Member Function Documentation . . . . .	622
11.143.2 Member Data Documentation . . . . .	623
11.144 CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData Class Reference . . . . .	623
11.144.1 Constructor & Destructor Documentation . . . . .	623
11.144.2 Property Documentation . . . . .	624
11.145 usbSetupPacket_t Class Reference . . . . .	624

11.145.1 Member Data Documentation . . . . .	624
11.146 W2100_StimulusParametersNet Struct Reference . . . . .	625
11.146.1 Member Data Documentation . . . . .	625
<b>Index</b>	<b>626</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.Disonnect();
```

## 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 divided 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:

<a href="#">CW2100_FunctionNet::AudioChannelsNet</a>	<a href="#">29</a>
<a href="#">BatteryState</a>	<a href="#">29</a>
<a href="#">CCreateFilterNet</a>	<a href="#">48</a>
<a href="#">BesselFilterHighPassNet</a>	<a href="#">30</a>
<a href="#">BesselFilterLowPassNet</a>	<a href="#">31</a>
<a href="#">ButterworthFilterHighPassNet</a>	<a href="#">31</a>
<a href="#">ButterworthFilterLowPassNet</a>	<a href="#">32</a>
<a href="#">CDeviceGroupChannelInfoTemplateNet&lt; DacqGroupChannelEnumTemplateNet &gt;</a>	<a href="#">58</a>
<a href="#">CDeviceGroupChannelInfoTemplateNet&lt; DacqGroupChannelEnumNet &gt;</a>	<a href="#">58</a>
<a href="#">CDeviceGroupChannelInfoNet</a>	<a href="#">57</a>
<a href="#">CDeviceGroupChannelInfoTemplateNet&lt; int &gt;</a>	<a href="#">58</a>
<a href="#">CDeviceGroupChannelInfoGenericNet</a>	<a href="#">56</a>

CDeviceGroupChannelInfoTemplateNet< MEA2100_256DacqGroupChannelEnumNet >	58
CDeviceGroupChannelInfoMEA2100_256Net	56
CDeviceGroupChannelInfoTemplateNet< SCUDacqGroupChannelEnumNet >	58
CDeviceGroupChannelInfoSCUNet	57
CDeviceGroupChannelInfoTemplateNet< W2100DacqGroupChannelEnumNet >	58
CDeviceGroupChannelInfoW2100Net	59
CFilterCoefficientsNet	66
CFilterPropertyNet	72
CMcsUsbDacqNet::CHWInfo	112
CMcsUsbFunctionNet	230
CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumNet, DacqGroupChannelEnum, CDeviceGroupChannelInfoNet >	53
CDacqGroupChannelSelectionNet	52
CDacqGroupChannelSelectionTemplateNet< int, int, CDeviceGroupChannelInfoGenericNet >	53
CDacqGroupChannelGenericSelectionNet	52
CDacqGroupChannelSelectionTemplateNet< MEA2100_256DacqGroupChannelEnumNet, MEA2100_256DacqGroupChannelEnum, CDeviceGroupChannelInfoMEA2100_256Net >	53
CMEA2100_256DacqGroupChannelSelectionNet	261
CDacqGroupChannelSelectionTemplateNet< SCUDacqGroupChannelEnumNet, SCUDacqGroupChannelEnum, CDeviceGroupChannelInfoSCUNet >	53
CSCUDacqGroupChannelSelectionNet	420
CDacqGroupChannelSelectionTemplateNet< W2100DacqGroupChannelEnumNet, W2100DacqGroupChannelEnum, CDeviceGroupChannelInfoW2100Net >	53
CW2100DacqGroupChannelSelectionNet	540
CCMOSMea_FunctionNet	34
CDacCalibrationFunctionNet	50
CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet >	53
CDigOutStimulatorFunctionNet	59
CFilterConfigurationNet	68
CFilterConfigurationRegisterNet	70
CGrapheneFunctionNet	100
CIntanMea_FunctionNet	114
CInterfaceboardFunctionNet	118

CInterfaceboard2FunctionNet	116
CMcsBus_AxisParametersNet	129
CMcsBus_ExtensionNet	130
CMcsBus_FYIExtensionNet	131
CMcsBus_MotorControlNet	133
CMcsBus_SensorNet	150
CMcsBus_TempSensorNet	161
CMcsBus_VoltageModeNet	163
CMcsBusNet	167
CMcsUsbDeviceStatePushFunctionNet	219
CMultiwellCallbackFunctionNet	304
CSCUFunctionNet	421
CMEA2100x256FunctionNet	262
CMeaAudioFunctionNet	263
CMeaDigitalDataFunctionNet	284
CMeaFeedbackFunctionNet	286
CMeFunctionNet	295
CMultiwellOptoStimFunctionNet	312
CPPCFunctionNet	345
CPPS_FunctionNet	355
CProgramPressureCurveNet	360
CPulseGeneratorFunctionNet	361
CRFFunctionNet	369
CRobo_FYIProgram_FunctionNet	374
CRobo_FYITemp_FunctionNet	375
CStimulusFunctionNet	483
CTEERFunctionNet	510
CUsbDeviceConfigurationFunctionNet	521
CW2100_StimulatorFunctionNet	532
CWarnerUssingFunctionNet	542
CWarnerValveControllerDeviceTesterFunctionNet	583
CWirelessBaseFunctionNet	591

CW2100_FunctionNet	525
CWClassicFunctionNet	586
CMcsUsbFunctionPointerContainer	231
CMcsUsbListEntryNet	231
CMcsUsbListNet	237
CMcsUsbNet	240
CExternDTesterDeviceNet	64
CFluidControlDeviceNet	74
CGenericDevelopDeviceNet	81
CGilsonDeviceNet	98
CMcsUsbDacqNet	171
CMeaDeviceNet	276
CMeaUSBDeviceNet	294
CCMOSMeaDeviceNet	45
CHLADacqNet	110
CLIH3DeviceNet	120
CMultiwellDeviceNet	306
CWarnerUssingDeviceNet	541
COctoPotDeviceNet	318
CRoboDacqNet	377
CMcsUsbDeviceStatePushNet	220
CWarnerValveControllerDeviceNet	558
CMcsUsbFactoryNet	221
CMeaCleanDeviceNet	266
CMeaCoatDeviceNet	270
CMeaImpedanceDeviceNet	290
CMeaSwitchDeviceNet	292
CChannelTestDeviceNet	33
CMultiBatteryChargerDeviceNet	297
CNF_GenDeviceNet	317
CokuvisionStimulatorDeviceNet	322
CPathIdentDeviceNet	326

CPedoterDeviceNet	327
CPeristalticPumpDeviceNet	329
CPgaDeviceNet	330
CPositionIIDeviceNet	332
CPositionImpDeviceNet	341
CPPCDeviceNet	344
CPPS_DeviceNet	354
CRadioControlledDevicesNet	364
CRetinaLedDeviceNet	367
CRoboDeviceNet	391
CEncapsulatorDeviceNet	64
CFYIDeviceNet	80
CHiClampDeviceNet	109
CHLADeviceNet	111
CMeasureTableDeviceNet	291
CPatchServerDeviceNet	325
CPPSDeviceNet	359
CRoboStatorDeviceNet	411
CRoboInjectDeviceNet	409
CRoboocyte2DeviceNet	409
CTEERMachineDeviceNet	520
CRoboFluidDeviceNet	405
CSafeISDeviceNet	417
CSerialPortNet	436
CStg200xBasicNet	437
CStg200xDownloadBasicNet	467
CStg200xDownloadNet	476
CSw2to64DeviceNet	494
CTcxDeviceNet	497
CMcsUsbPointerContainer	261
CCMOSMeaDeviceNet::CRegionOfInterestRect	366
CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet	524

<b>DeviceIdNet</b>	<b>591</b>
<b>DigitalSource</b> < digitalsourceenum >	<b>593</b>
<b>DigitalSourceGeneral</b>	<b>594</b>
<b>DriverVersionNet</b>	<b>596</b>
Exception	
<b>CUsbExceptionNet</b>	<b>523</b>
<b>FirmwareDestinationNames</b>	<b>601</b>
<b>HeadstageIDTypeObject</b>	<b>607</b>
<b>HeadStageIDTypeState</b>	<b>608</b>
IComparable	
<b>HeadStageIDType</b>	<b>604</b>
<b>mkfilterNet</b>	<b>609</b>
<b>CRoboDeviceNet::RoboMainLowLevelCommands</b>	<b>612</b>
<b>CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands</b>	<b>619</b>
<b>CFilterCoefficientsNet::s_FilterAttributesNet</b>	<b>619</b>
<b>CMeaAudioFunctionNet::s_setaudionet</b>	<b>621</b>
<b>CStimulusFunctionNet::SidebandData</b>	<b>621</b>
<b>StgStatusNet</b>	<b>622</b>
stgstreaming	
<b>CStg200xBasicNet</b>	<b>437</b>
<b>CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData</b>	<b>623</b>
<b>usbSetupPacket_t</b>	<b>624</b>
<b>W2100_StimulusParametersNet</b>	<b>625</b>

## 9 Class Index

### 9.1 Class List

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

<b>CW2100_FunctionNet::AudioChannelsNet</b>	<b>29</b>
<b>BatteryState</b>	<b>29</b>
<b>BesselFilterHighPassNet</b>	<b>30</b>
<b>BesselFilterLowPassNet</b>	<b>31</b>
<b>ButterworthFilterHighPassNet</b>	<b>31</b>

ButterworthFilterLowPassNet	32
CChannelTestDeviceNet	33
CCMOSMea_FunctionNet	34
CCMOSMeaDeviceNet	45
CCreateFilterNet	48
CDacCalibrationFunctionNet 50	
CDacqGroupChannelGenericSelectionNet	52
CDacqGroupChannelSelectionNet	52
CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplateNet, 53	
CDeviceGroupChannelInfoGenericNet	56
CDeviceGroupChannelInfoMEA2100_256Net	56
CDeviceGroupChannelInfoNet	57
CDeviceGroupChannelInfoSCUNet	57
CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumTemplateNet >	58
CDeviceGroupChannelInfoW2100Net	59
CDigOutStimulatorFunctionNet CDigOutStimulatorFunctionNet is the class of the DigOut stimulator function class	59
CEncapsulatorDeviceNet CEncapsulatorDeviceNet is the to control the MCS HiClamp device	64
CExternDTesterDeviceNet CExternDTesterDeviceNet is the class to access the ExternD Tester (Handheld Device Tester D)	64
CFilterCoefficientsNet	66
CFilterConfigurationNet	68
CFilterConfigurationRegisterNet	70
CFilterPropertyNet	72
CFluidControlDeviceNet CFluidControlDeviceNet is the class to control MCS FluidControl (FCB and FCX) device	74
CFYIDeviceNet CFYIDeviceNet is the class to control the MCS FYI device	80
CGenericDevelopDeviceNet CGenericDevelopDeviceNet is the class to use during development of a new device	81
CGilsonDeviceNet CGilsonDeviceNet is the class to control a Gilson device	98
CGrapheneFunctionNet CGrapheneFunctionNet is the class to control the TEER device	100

<a href="#">CHiClampDeviceNet</a>	
<a href="#">CHiClampDeviceNet</a> is the to control the MCS HiClamp device	109
<a href="#">CHLADacqNet</a>	110
<a href="#">CHLADeviceNet</a>	
<a href="#">CHLADeviceNet</a> is the to control the MCS HLA device	111
<a href="#">CMcsUsbDacqNet::CHWInfo</a>	
Class to provide hardware information about the device	112
<a href="#">CIntanMea_FunctionNet</a>	114
<a href="#">CInterfaceboard2FunctionNet</a>	
<a href="#">CInterfaceboard2FunctionNet</a> is the class to control the Interfaceboard	116
<a href="#">CInterfaceboardFunctionNet</a>	
<a href="#">CInterfaceboardFunctionNet</a> is the class to control the Interfaceboard	118
<a href="#">CLIH3DeviceNet</a>	
<a href="#">CLIH3DeviceNet</a> is the class to access the HEKA LIH3 device	120
<a href="#">CMcsBus_AxisParametersNet</a>	129
<a href="#">CMcsBus_ExtensionNet</a>	130
<a href="#">CMcsBus_FYIExtensionNet</a>	131
<a href="#">CMcsBus_MotorControlNet</a>	133
<a href="#">CMcsBus_SensorNet</a>	150
<a href="#">CMcsBus_TempSensorNet</a>	161
<a href="#">CMcsBus_VoltageModeNet</a>	163
<a href="#">CMcsBusNet</a>	167
<a href="#">CMcsUsbDacqNet</a>	
Base class for data acquisition devices	171
<a href="#">CMcsUsbDeviceStatePushFunctionNet</a>	219
<a href="#">CMcsUsbDeviceStatePushNet</a>	220
<a href="#">CMcsUsbFactoryNet</a>	221
<a href="#">CMcsUsbFunctionNet</a>	230
<a href="#">CMcsUsbFunctionPointerContainer</a>	231
<a href="#">CMcsUsbListEntryNet</a>	
<a href="#">McsUsbListEntryNet</a> identifies a connected device	231
<a href="#">CMcsUsbListNet</a>	
Class to handle a list of connected MCS USB devices	237
<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	240
<a href="#">CMcsUsbPointerContainer</a>	261



<a href="#">CMEA2100_256DacqGroupChannelSelectionNet</a>	261
<a href="#">CMEA2100x256FunctionNet</a>	
<a href="#">CMEA2100x256FunctionNet</a> is the class to control the MEA2100-256 device needs #include " <a href="#">Stg200xNet.h</a> " to resolve documentation reference	262
<a href="#">CMeaAudioFunctionNet</a>	263
<a href="#">CMeaCleanDeviceNet</a>	
<a href="#">CMeaCleanDeviceNet</a> is the class to access the MEA Clean device	266
<a href="#">CMeaCoatDeviceNet</a>	
<a href="#">CMeaCoatDeviceNet</a> is the class to access the MEA Coat device	270
<a href="#">CMeaDeviceNet</a>	
Base class for MEA data acquisition devices	276
<a href="#">CMeaDigitalDataFunctionNet</a>	284
<a href="#">CMeaFeedbackFunctionNet</a>	286
<a href="#">CMeaImpedanceDeviceNet</a>	290
<a href="#">CMeasureTableDeviceNet</a>	
<a href="#">CMeasureTableDeviceNet</a> is the to control the MCS HLA device	291
<a href="#">CMeaSwitchDeviceNet</a>	
The class to control the USB-MEA-Switch	292
<a href="#">CMeaUSBDeviceNet</a>	
Class for data acquisition via ME and MEA USB amplifiers	294
<a href="#">CMeFunctionNet</a>	295
<a href="#">CMultiBatteryChargerDeviceNet</a>	
<a href="#">CMultiBatteryChargerDeviceNet</a> is the class to access the MBC-08 device	297
<a href="#">CMultiwellCallbackFunctionNet</a>	
<a href="#">CMultiwellCallbackFunctionNet</a> is the class to access the Multiwell-Mini-Stimulator	304
<a href="#">CMultiwellDeviceNet</a>	
<a href="#">CMultiwellDeviceNet</a> is the class to access the Multiwell device	306
<a href="#">CMultiwellOptoStimFunctionNet</a>	
<a href="#">CMultiwellOptoStimFunctionNet</a> is the class to access the optical properties of the Multiwell Optostim device	312
<a href="#">CNF_GenDeviceNet</a>	317
<a href="#">COctoPotDeviceNet</a>	318
<a href="#">CokuvisionStimulatorDeviceNet</a>	322
<a href="#">CPatchServerDeviceNet</a>	
<a href="#">CPatchServerDeviceNet</a> is the class to control the MCS PatchServer device	325
<a href="#">CPathIdentDeviceNet</a>	326
<a href="#">CPedoterDeviceNet</a>	327

<a href="#">CPeristalticPumpDeviceNet</a>	
<a href="#">CPeristalticPumpDeviceNet</a> is the class to control a Persistaltic Pump	329
<a href="#">CPgaDeviceNet</a>	330
<a href="#">CPositionIIDeviceNet</a>	
<a href="#">CPositionIIDeviceNet</a> is the class to control PositionII devices	332
<a href="#">CPositionImpDeviceNet</a>	
<a href="#">CPositionImpDeviceNet</a> is the class to access the Position/Imp devices	341
<a href="#">CPPCDeviceNet</a>	344
<a href="#">CPPCFunctionNet</a>	
<a href="#">CPPCFunctionNet</a> is the class to access the PPC (high precision Patch Peristaltic patch Pump	345
<a href="#">CPPS_DeviceNet</a>	354
<a href="#">CPPS_FunctionNet</a>	355
<a href="#">CPPSDeviceNet</a>	
<a href="#">CPPS4plus1DeviceNet</a> is the to control the MCS HLA device	359
<a href="#">CProgramPressureCurveNet</a>	
<a href="#">CProgramPressureCurveNet</a> is the class to program pressure curves	360
<a href="#">CPulseGeneratorFunctionNet</a>	
<a href="#">CPulseGeneratorFunctionNet</a> is the class to control the pulse generator for video tracking	361
<a href="#">CRadioControlledDevicesNet</a>	364
<a href="#">CCMOSMeaDeviceNet::CRegionOfInterestRect</a>	366
<a href="#">CRetinaLedDeviceNet</a>	367
<a href="#">CRFFFunctionNet</a>	
<a href="#">CRFFFunctionNet</a> is the class to control RF devices	369
<a href="#">CRobo_FYIPProgram_FunctionNet</a>	374
<a href="#">CRobo_FYITemp_FunctionNet</a>	375
<a href="#">CRoboDacqNet</a>	377
<a href="#">CRoboDeviceNet</a>	
<a href="#">CRoboDeviceNet</a> is the base class for all Robo platform based devices	391
<a href="#">CRoboFluidDeviceNet</a>	405
<a href="#">CRoboInjectDeviceNet</a>	
<a href="#">CRoboInjectDeviceNet</a> is the to control the MCS RoboInject device	409
<a href="#">CRoboocyte2DeviceNet</a>	
<a href="#">CRoboocyte2DeviceNet</a> is the class to control the MCS Roboocyte2 device	409
<a href="#">CRoboStatorDeviceNet</a>	411
<a href="#">CSafeISDeviceNet</a>	
417	
<a href="#">CSCUDacqGroupChannelSelectionNet</a>	420

<a href="#">CSCUFunctionNet</a>	
<a href="#">CSCUFunctionNet</a> is the class to control the SCU device	421
<a href="#">CSerialPortNet</a>	436
<a href="#">CStg200xBasicNet</a>	
Base class for the Stg200x	437
<a href="#">CStg200xDownloadBasicNet</a>	
<a href="#">CStg200xDownloadBasicNet</a> is the base class to control the download mode of the MCS STG device	467
<a href="#">CStg200xDownloadNet</a>	
Main class for the STG download mode This class implements the STG download mode interface.	476
<a href="#">CStimulusFunctionNet</a>	483
<a href="#">CSw2to64DeviceNet</a>	
The class to control the MCS-USB-Sw2to64 device	494
<a href="#">CTcxDeviceNet</a>	
Class to control a Temperature Controller (TCX)	497
<a href="#">CTEERFunctionNet</a>	
<a href="#">CTEERFunctionNet</a> is the class to control the TEER device	510
<a href="#">CTEERMachineDeviceNet</a>	520
<a href="#">CUsbDeviceConfigurationFunctionNet</a>	
<a href="#">CUsbDeviceConfigurationFunctionNet</a> is the class to configure the USB firmware	521
<a href="#">CUsbExceptionNet</a>	
Exception class that is thrown in case of an USB error	523
<a href="#">CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet</a>	524
<a href="#">CW2100_FunctionNet</a>	525
<a href="#">CW2100_StimulatorFunctionNet</a>	532
<a href="#">CW2100DacqGroupChannelSelectionNet</a>	540
<a href="#">CWarnerUssingDeviceNet</a>	
<a href="#">CWarnerUssingDeviceNet</a> is the class to control the Ussing device	541
<a href="#">CWarnerUssingFunctionNet</a>	
<a href="#">CWarnerUssingFunctionNet</a> is the class to control the Ussing device	542
<a href="#">CWarnerValveControllerDeviceNet</a>	
<a href="#">CWarnerValveControllerDeviceNet</a> is the class to access the Warner Valve Controller	558
<a href="#">CWarnerValveControllerDeviceTesterFunctionNet</a>	
<a href="#">CWarnerValveControllerDeviceTesterFunctionNet</a> is the class to access the functions for the Warner Valve Controller Device Tester	583
<a href="#">CWClassicFunctionNet</a>	586
<a href="#">CWirelessBaseFunctionNet</a>	591
<a href="#">DeviceIdNet</a>	
Device Id	591

<a href="#">DigitalSource&lt; digitalsourceenum &gt;</a>	593
<a href="#">DigitalSourceGeneral</a>	594
<a href="#">DriverVersionNet</a>	
Class gives firmware versions of the device's firmware destinations	596
<a href="#">FirmwareDestinationNames</a>	601
<a href="#">HeadStageIDType</a>	604
<a href="#">HeadstageIDTypeObject</a>	607
<a href="#">HeadStageIDTypeState</a>	608
<a href="#">mkfilterNet</a>	609
<a href="#">CRoboDeviceNet::RoboMainLowLevelCommands</a>	612
<a href="#">CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands</a>	619
<a href="#">CFilterCoefficientsNet::s_FilterAttributesNet</a>	619
<a href="#">CMeaAudioFunctionNet::s_setaudionet</a>	621
<a href="#">CStimulusFunctionNet::SidebandData</a>	621
<a href="#">StgStatusNet</a>	622
<a href="#">CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData</a>	623
<a href="#">usbSetupPacket_t</a>	624
<a href="#">W2100_StimulusParametersNet</a>	625

## 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 [CDeviceGroupChannelInfoMEA2100\\_256Net](#)
- 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 [CGrapheneFunctionNet](#)  
*CGrapheneFunctionNet is the class to control the TEER 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 [CInterfaceboard2FunctionNet](#)  
*CInterfaceboard2FunctionNet is the class to control the Interfaceboard*
- 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 [CMEA2100\\_256DacqGroupChannelSelectionNet](#)
- 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 Multiwell 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\\_FYIProgram\\_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 [CUsbDeviceConfigurationFunctionNet](#)  
*CUsbDeviceConfigurationFunctionNet is the class to configure the USB firmware*

- 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 [DeviceldNet](#)  
*Device Id.*
- class [DigitalSource](#)
- class [DigitalSourceGeneral](#)
- 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](#)<sup>^</sup> stgStatusNet, array< int ><sup>^</sup> index\_list)
- public delegate void [OnMwPollStatus](#) (unsigned int CurrentTemp, unsigned int PlateState, unsigned int SwitchState)
- public delegate void [RoboStatusEventDelegate](#) (array< unsigned char ><sup>^</sup> buffer)
- public delegate void [OnStg200xDataHandler](#) (uint32\_t trigger)
- public delegate void [OnStg200xErrorHandler](#) ()
- public delegate void [OnChannelData](#) ([CMcsUsbDacqNet](#)<sup>^</sup> dacq, int CbHandle, int numFrames)
- public delegate void [OnError](#) (String<sup>^</sup> 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](#)<sup>^</sup> stgStatusNet,   
 array< int ><sup>^</sup> 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]

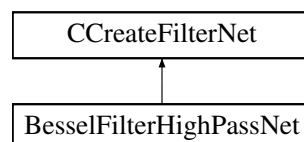
**11.2.1.3 ChargeString** System:: String^ ChargeString [get]

**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

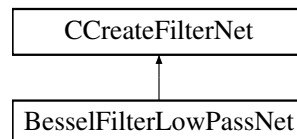
### 11.3.1 Constructor & Destructor Documentation

**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

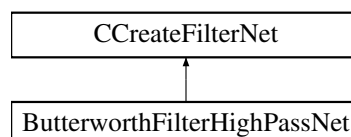
#### 11.4.1 Constructor & Destructor Documentation

**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)

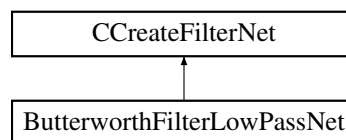
## Additional Inherited Members

### 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`)

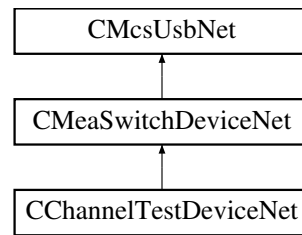
## Additional Inherited Members

### 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* )

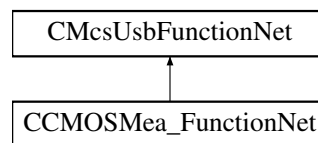
**11.7.2.2 SetAttenuation()** void SetAttenuation (   
 unsigned int *Attenuation* )

**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 >^ MemData)
- uint32\_t [GetNeurochipMemoryData](#) (uint16\_t MemAddress)
- array< uint32\_t > ^ [GetNeurochipMemoryData](#) (uint16\_t MemAddress, uint32\_t RequestLength)
- uint32\_t [GetNeurochipMemorySize](#) ()
- uint32\_t [GetMaxNumOfColumns](#) (uint32\_t Samplerate)
- void [SetStimulusSites](#) (List< int16\_t >^ SwitchPosition)
- List< int16\_t > ^ [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 virtualDevice)
- void [EnableChannelsInGroup](#) (DacqGroupChannelEnumNet GroupID, List< bool >^ EnabledChannelsBitMap)
- void [EnableChannelsInGroup](#) (DacqGroupChannelEnumNet GroupID, List< bool >^ EnabledChannelsBitMap, uint32\_t virtualDevice)
- List< bool > ^ [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 ChannelNumber)
- uint32\_t [GetGroupChannelBitmaskBySelect](#) (DacqGroupChannelEnumNet GroupID, uint32\_t ChannelNumber, 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 ChannelNumber)
- CMOSMeaHeadstage1NChipTempEnumNet [GetGroupChannelBitmaskHS1NChipTemp](#) (uint32\_t ChannelNumber, 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](#)<sup>^</sup> *mcsusb*,   
[CMcsUsbFunctionPointerContainer](#)<sup>^</sup> *cCMOSMea\_FunctionPointerContainer* )

**11.8.1.2 CCMOSMea\_FunctionNet()** [2/2] [CCMOSMea\\_FunctionNet](#) (   
[CMcsUsbNet](#)<sup>^</sup> *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 *RequestLength* )

**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 )

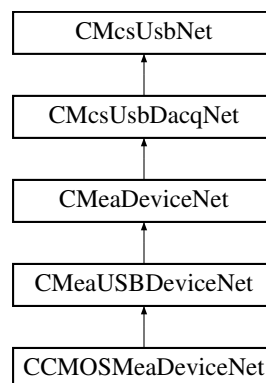
**11.8.2.76 UpdateTransistorVoltages()** void UpdateTransistorVoltages ( )

**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 )`

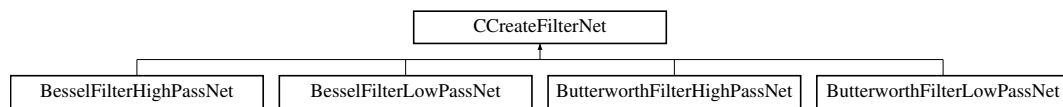
### 11.9.3 Property Documentation

**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)
- static int [FindFilter](#) (array< array< uint64\_t > ^>^ coef, array< [CCreateFilterNet](#) ^>^ param, [CFilterCoefficientsNet::s\\_FilterAttributesNet](#) ^ FiltAttr, bool DoMCSLegacyCompare)

### 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()` [1/2] `static int FindFilter (`  
`array< array< uint64_t >>^ coef,`  
`array< CCreateFilterNet^>^ param,`  
`CFilterCoefficientsNet::s_FilterAttributesNet^ FiltAttr,`  
`bool DoMCSLegacyCompare )` [static]

**11.10.2.2** `FindFilter()` [2/2] `static int FindFilter (`  
`array< CFilterCoefficientsNet^>^ coef,`  
`array< CCreateFilterNet^>^ param )` [static]

**11.10.2.3** `GetBiQuad()` `CFilterCoefficientsNet^ GetBiQuad (`  
`int index )`

**11.10.2.4** `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]

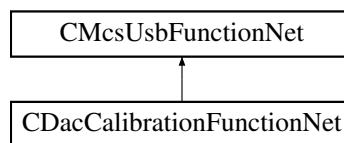
**11.10.3.3 Order** `int Order [get]`

**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

<i>dacChannel</i>	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 )`

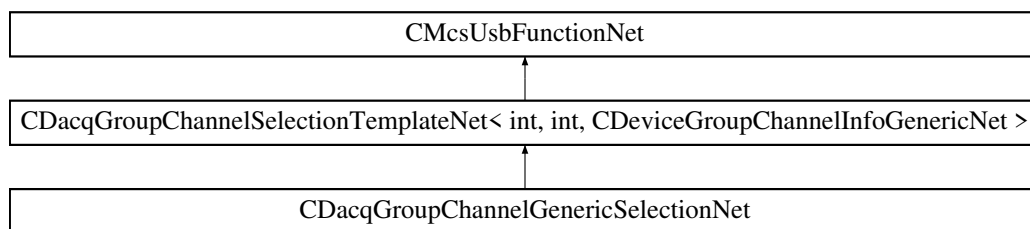
Sets the offset of a DAC channel.

#### 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)

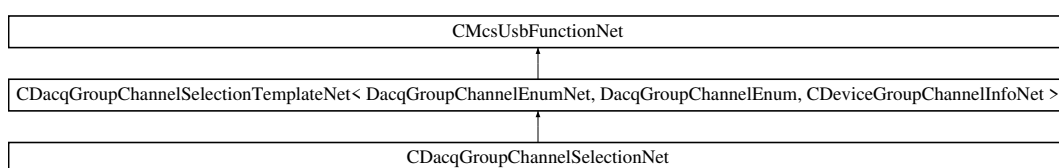
### Additional Inherited Members

#### 11.12.1 Constructor & Destructor Documentation

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

## 11.13 CDacqGroupChannelSelectionNet Class Reference

Inheritance diagram for CDacqGroupChannelSelectionNet:



## 11.14 CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet > Class Template Reference 53

### Public Member Functions

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

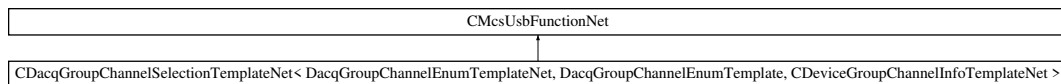
### Additional Inherited Members

#### 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)
- [void EnableChannelsInGroup](#) ([DacqGroupChannelEnumTemplateNet](#) GroupID, [List< bool >](#) EnabledChannelsBitMap)
- [void EnableChannelsInGroup](#) ([DacqGroupChannelEnumTemplateNet](#) GroupID, [List< bool >](#) EnabledChannelsBitMap, [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^> ^ Get↔ DeviceGroupChannelInfos ( )`

**11.14.2.4 GetDeviceGroupChannelInfos()** [2/2] `List<CDeviceGroupChannelInfoTemplateNet^> ^ Get↔ DeviceGroupChannelInfos ( 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* )

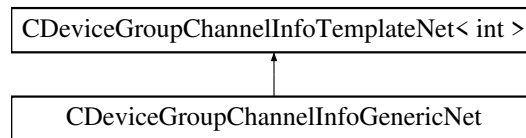
**11.14.2.14 GetGroupType()** [2/2] DacqMeaGroupTypeEnumNet GetGroupType (   
DacqGroupChannelEnumTemplateNet *GroupID*,  
uint32\_t *virtualDevice* )

**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)

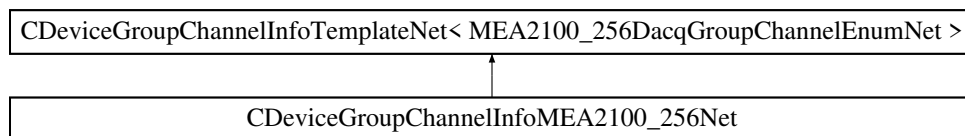
### Additional Inherited Members

#### 11.15.1 Constructor & Destructor Documentation

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

## 11.16 CDeviceGroupChannelInfoMEA2100\_256Net Class Reference

Inheritance diagram for CDeviceGroupChannelInfoMEA2100\_256Net:



### Public Member Functions

- [CDeviceGroupChannelInfoMEA2100\\_256Net](#) (MEA2100\_256DacqGroupChannelEnumNet id, int channels, DacqMeaGroupTypeEnumNet type)

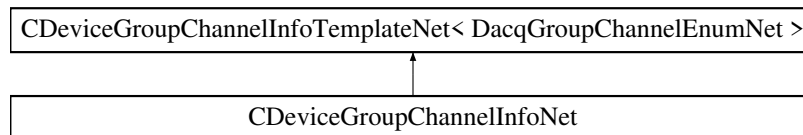
## Additional Inherited Members

### 11.16.1 Constructor & Destructor Documentation

**11.16.1.1 CDeviceGroupChannelInfoMEA2100\_256Net()** [CDeviceGroupChannelInfoMEA2100\\_256Net](#) (   
 MEA2100\_256DacqGroupChannelEnumNet *id*,  
 int *channels*,  
 DacqMeaGroupTypeEnumNet *type* )

## 11.17 CDeviceGroupChannelInfoNet Class Reference

Inheritance diagram for CDeviceGroupChannelInfoNet:



## Public Member Functions

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

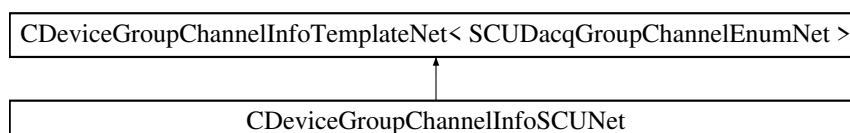
## Additional Inherited Members

### 11.17.1 Constructor & Destructor Documentation

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

## 11.18 CDeviceGroupChannelInfoSCUNet Class Reference

Inheritance diagram for CDeviceGroupChannelInfoSCUNet:



## Public Member Functions

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

## Additional Inherited Members

### 11.18.1 Constructor & Destructor Documentation

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

## 11.19 CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumTemplateNet > Class Template Reference

## Public Member Functions

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

## Public Attributes

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

### 11.19.1 Constructor & Destructor Documentation

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

### 11.19.2 Member Data Documentation

**11.19.2.1 GroupID** DacqGroupChannelEnumTemplateNet GroupID

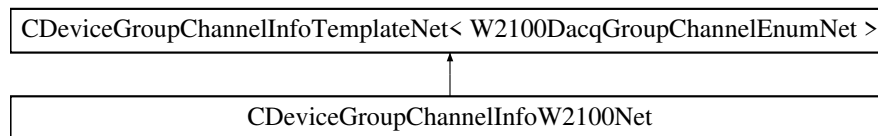


**11.19.2.2 GroupType** `DacqMeaGroupTypeEnumNet GroupType`

**11.19.2.3 NumberOfChannels** `int NumberOfChannels`

## 11.20 CDeviceGroupChannelInfoW2100Net Class Reference

Inheritance diagram for CDeviceGroupChannelInfoW2100Net:



### Public Member Functions

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

### Additional Inherited Members

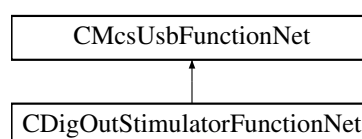
#### 11.20.1 Constructor & Destructor Documentation

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

## 11.21 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](#) ()
- [ICDigOutStimulatorFunctionNet](#) ()
- void [ClearChannel](#) (int32\_t NrChannel)  
*clear stimulation pattern*
- [CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData](#) ^ [PrepareChannelData](#) (array< int32\_t > ^ Amplitude, array< uint64\_t > ^ Duration)  
*prepares the channel data for the device and unrolles the data for the GUI*
- void [SendChannelData](#) (int32\_t NrChannel, [CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData](#)<sup>^</sup> device\_data\_and\_unrolled)  
*send or append stimulation pattern*
- int32\_t [GetNumberOfChannels](#) ()  
*get the number of channels available on the device*
- 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

### 11.21.1 Detailed Description

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

### 11.21.2 Constructor & Destructor Documentation

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

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

**11.21.2.2 CDigOutStimulatorFunctionNet()** [2/2] `CDigOutStimulatorFunctionNet ( CMcsUsbNet^ mcsusb )`

**11.21.2.3 ~CDigOutStimulatorFunctionNet()** `virtual ~CDigOutStimulatorFunctionNet ( )` [virtual]

**11.21.2.4 "!CDigOutStimulatorFunctionNet()** `!CDigOutStimulatorFunctionNet ( )`

### 11.21.3 Member Function Documentation

**11.21.3.1 ClearChannel()** `void ClearChannel ( int32_t NrChannel )`

clear stimulation pattern

#### Parameters

<i>NrChannel</i>	the channel to clear
------------------	----------------------

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

get repeat whole stimulation pattern

#### Parameters

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

#### Returns

current value

**11.21.3.3 GetNumberOfChannels()** `int32_t GetNumberOfChannels ( )`

get the number of channels available on the device

#### Returns

the number of channels

**11.21.3.4 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)

**11.21.3.5 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)

**11.21.3.6 PrepareChannelData()** `CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData ^ PrepareChannelData ( array< int32_t >^ Amplitude, array< uint64_t >^ Duration )`

prepares the channel data for the device and unrolles the data for the GUI

#### Parameters

<i>Amplitude</i>	array of amplitudes
<i>Duration</i>	array of durations

#### Returns

**11.21.3.7 SendChannelData()** void SendChannelData (  
     int32\_t *NrChannel*,  
     CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData^ *device\_data\_and\_unrolled*  
 )

send or append stimulation pattern

#### Parameters

<i>NrChannel</i>	the channel to send data to
<i>device_data_and_unrolled</i>	internal, use value obtained from PrepareChannelData

**11.21.3.8 SetGlobalRepeat()** void SetGlobalRepeat (  
     int32\_t *NrChannel*,  
     bool *value* )

set repeat whole stimulation pattern

#### Parameters

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

**11.21.3.9 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)

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

sets stop condition of digital out stimulator

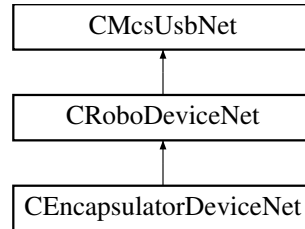
#### Parameters

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

## 11.22 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

#### 11.22.1 Detailed Description

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

#### 11.22.2 Constructor & Destructor Documentation

**11.22.2.1 CEncapsulatorDeviceNet()** [CEncapsulatorDeviceNet](#) (   
 void )

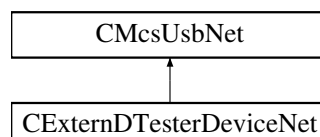
#### 11.22.3 Member Function Documentation

**11.22.3.1 GetRoboFluidDevice()** [CRoboFluidDeviceNet](#) ^ [GetRoboFluidDevice](#) ( )

## 11.23 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](#) ()
- [IExternDTesterDeviceNet](#) ()
- 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****11.23.1 Detailed Description**

[CExternDTesterDeviceNet](#) is the class to access the ExternD Tester (Handheld Device Tester D)

**11.23.2 Constructor & Destructor Documentation****11.23.2.1 CExternDTesterDeviceNet()** [CExternDTesterDeviceNet](#) ( )

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

**11.23.2.2 ~CExternDTesterDeviceNet()** virtual [~CExternDTesterDeviceNet](#) ( ) [virtual]**11.23.2.3 !IExternDTesterDeviceNet()** [!CExternDTesterDeviceNet](#) ( )**11.23.3 Member Function Documentation****11.23.3.1 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.

**11.23.3.2 Read2()** `String ^ Read2 ( )`

Reads the config string from the device.

## Returns

The config string.

**11.23.3.3 Write()** `void Write ( array< uint8_t >^ configString )`

Reads the config string from the device.

## Parameters

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

**11.23.3.4 Write2()** `void Write2 ( String^ configString )`

Reads the config string from the device.

## Parameters

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

**11.24 CFilterCoefficientsNet Class Reference**

## Classes

- struct [s\\_FilterAttributesNet](#)



**Public Member Functions**

- [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](#) ()
- bool [IsEqual](#) (array< uint64\_t >^ coefficients, [s\\_FilterAttributesNet](#)^ FiltAttr)
- bool [IsEqual](#) (array< uint64\_t >^ coefficients, [s\\_FilterAttributesNet](#)^ FiltAttr, bool DoMCSLegacyCompare)
- uint64\_t [GetUIntB](#) (int index, [s\\_FilterAttributesNet](#)^ FiltAttr)
- uint64\_t [GetUIntA](#) (int index, [s\\_FilterAttributesNet](#)^ FiltAttr)

**Properties**

- array< double >^ [A](#) [get]
- array< double >^ [B](#) [get]

**11.24.1 Constructor & Destructor Documentation**

**11.24.1.1 CFilterCoefficientsNet()** [1/4] [CFilterCoefficientsNet](#) ( )

**11.24.1.2 CFilterCoefficientsNet()** [2/4] [CFilterCoefficientsNet](#) (   
double b0,   
double b1,   
double b2,   
double a1,   
double a2 )

**11.24.1.3 CFilterCoefficientsNet()** [3/4] [CFilterCoefficientsNet](#) (   
double b0,   
double b1,   
double a1 )

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

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

### 11.24.2 Member Function Documentation

**11.24.2.1 GetUintA()** `uint64_t GetUintA (`  
     `int index,`  
     `s_FilterAttributesNet^ FiltAttr )`

**11.24.2.2 GetUintB()** `uint64_t GetUintB (`  
     `int index,`  
     `s_FilterAttributesNet^ FiltAttr )`

**11.24.2.3 IsEqual() [1/2]** `bool IsEqual (`  
     `array< uint64_t >^ coefficients,`  
     `s_FilterAttributesNet^ FiltAttr )`

**11.24.2.4 IsEqual() [2/2]** `bool IsEqual (`  
     `array< uint64_t >^ coefficients,`  
     `s_FilterAttributesNet^ FiltAttr,`  
     `bool DoMCSLegacyCompare )`

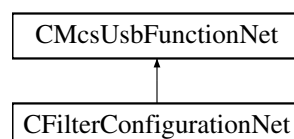
### 11.24.3 Property Documentation

**11.24.3.1 A** `array< double>^ A [get]`

**11.24.3.2 B** `array< double>^ B [get]`

## 11.25 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](#) ()
- uint32\_t [GetFilterAttributes](#) (DacqGroupChannelEnumNet GroupID, uint32\_t FilterNumber, FilterAttributeEnumNet index)
- [CFilterCoefficientsNet::s\\_FilterAttributesNet](#) <sup>^</sup> [GetFilterAttributes](#) (DacqGroupChannelEnumNet GroupID, uint32\_t FilterNumber)

**Additional Inherited Members****11.25.1 Constructor & Destructor Documentation**

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

**11.25.2 Member Function Documentation**

**11.25.2.1 EraseFilterParameterPermanent()** void EraseFilterParameterPermanent (  
DacqGroupChannelEnumNet GroupID,  
uint32\_t FilterNumber )

**11.25.2.2 GetFilterAttributes()** [1/2] [CFilterCoefficientsNet::s\\_FilterAttributesNet](#) <sup>^</sup> GetFilterAttributes (  
DacqGroupChannelEnumNet GroupID,  
uint32\_t FilterNumber )

**11.25.2.3 GetFilterAttributes()** [2/2] uint32\_t GetFilterAttributes (  
DacqGroupChannelEnumNet GroupID,  
uint32\_t FilterNumber,  
FilterAttributeEnumNet index )

**11.25.2.4 GetHighpassFilterEnable()** `bool GetHighpassFilterEnable ( )`

**11.25.2.5 ResetHighpassFilter()** `void ResetHighpassFilter ( )`

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

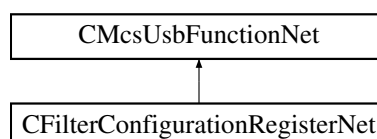
**11.25.2.7 SetFilterParameter() [2/2]** `void SetFilterParameter (`  
`DacqGroupChannelEnumNet GroupID,`  
`uint32_t FilterNumber,`  
`CFilterCoefficientsNet^ CoefficientsSet1,`  
`CFilterCoefficientsNet^ CoefficientsSet2,`  
`CFilterPropertyNet^ FilterProp )`

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

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

## 11.26 CFilterConfigurationRegisterNet Class Reference

Inheritance diagram for CFilterConfigurationRegisterNet:



**Public Member Functions**

- [CFilterConfigurationRegisterNet](#) ([CMcsUsbNet](#)^ mcsusb)
- void [SetFilterParameter](#) (uint32\_t FilterCoefRegBase, [CFilterCoefficientsNet](#)^ Coefficients, uint32\_t FilterInfoRegBase, [CFilterPropertyNet](#)^ FilterProp)
- void [SetFilterParameter](#) (uint32\_t FilterCoefSet1RegBase, [CFilterCoefficientsNet](#)^ CoefficientsSet1, uint32\_t FilterCoefSet2RegBase, [CFilterCoefficientsNet](#)^ CoefficientsSet2, uint32\_t FilterInfoRegBase, [CFilterPropertyNet](#)^ 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****11.26.1 Constructor & Destructor Documentation**

**11.26.1.1 CFilterConfigurationRegisterNet()** [CFilterConfigurationRegisterNet](#) (  
[CMcsUsbNet](#)^ mcsusb )

**11.26.2 Member Function Documentation**

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

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

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

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

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

**11.26.2.6 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 )

## 11.27 CFilterPropertyNet Class Reference

### Public Member Functions

- CFilterPropertyNet (uint32\_t CornerFrequencyMHz, 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]

### 11.27.1 Constructor & Destructor Documentation

**11.27.1.1 CFilterPropertyNet()** `CFilterPropertyNet (`  
    uint32\_t *CornerFrequenzymHz*,  
    uint32\_t *Order*,  
    FilterBandEnumNet *FilterBand*,  
    FilterFamilyEnumNet *FilterFamily*,  
    FilterTypeEnumNet *FilterType*,  
    bool *Active* )

**11.27.1.2 ~CFilterPropertyNet()** `~CFilterPropertyNet ( )`

### 11.27.2 Member Function Documentation

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

### 11.27.3 Property Documentation

**11.27.3.1 CornerFrequency** `double CornerFrequency [get]`

**11.27.3.2 CornerFrequencymHz** `uint32_t CornerFrequencymHz [get]`

**11.27.3.3 FilterActive** `bool FilterActive [get]`

**11.27.3.4 FilterBand** `FilterBandEnumNet FilterBand [get]`

**11.27.3.5 FilterFamily** `FilterFamilyEnumNet FilterFamily [get]`

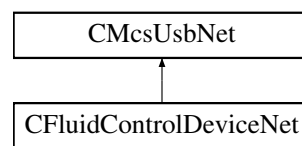
**11.27.3.6 FilterType** `FilterTypeEnumNet FilterType [get]`

**11.27.3.7 Order** `uint32_t Order [get]`

## 11.28 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 [GetValve](#) ()  
*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

### 11.28.1 Detailed Description

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

### 11.28.2 Constructor & Destructor Documentation

#### 11.28.2.1 CFluidControlDeviceNet() [CFluidControlDeviceNet](#) ( )

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

#### 11.28.2.2 ~CFluidControlDeviceNet() [~CFluidControlDeviceNet](#) ( )

Default destructor.

### 11.28.3 Member Function Documentation

#### 11.28.3.1 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.
----------------	------------------------------

**11.28.3.2 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.

**11.28.3.3 GetDigin()** `unsigned int GetDigin ( )`

Reads the digital input.

## Returns

The bit pattern of the state of the digital inputs.

**11.28.3.4 GetDigout()** `unsigned int GetDigout ( )`

Gets the state of the digital output.

## Returns

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

**11.28.3.5 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.

**11.28.3.6 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.

**11.28.3.7 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

**11.28.3.8 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.

**11.28.3.9 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.

**11.28.3.10 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.

**11.28.3.11 GetValve()** `unsigned int GetValve ( )`

Gets the state of the valves.

**Returns**

The current state of the valves as a bit pattern.

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

Define the pattern on the Digital Output.

**Parameters**

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

**11.28.3.13 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.

**11.28.3.14 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.
--------------	--------------------------

**11.28.3.15 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.

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

Open or Close valves.

## Parameters

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

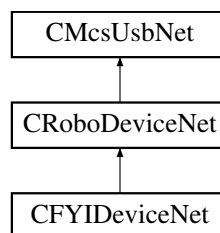
### 11.28.4 Property Documentation

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

## 11.29 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

### 11.29.1 Detailed Description

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

### 11.29.2 Constructor & Destructor Documentation

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

### 11.29.3 Property Documentation

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

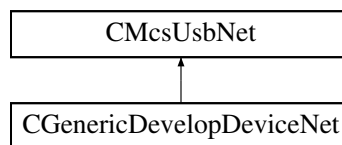
**11.29.3.2 FYITemp** `CRobo_FYITemp_FunctionNet^ FYITemp [get]`

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

## 11.30 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****11.30.1 Detailed Description**

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

**11.30.2 Constructor & Destructor Documentation**

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

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

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

**11.30.3 Member Function Documentation**

**11.30.3.1 ClosePipe()** void ClosePipe (  
    IntPtr *pipeHandle* )

Close a Pipe to an USB Endpoint.

*Parameters*

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

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

**11.30.3.3 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.

**11.30.3.4 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.

**11.30.3.5 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.

**11.30.3.6 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.

**11.30.3.7 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.

**11.30.3.8 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.

**11.30.3.9 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.

**11.30.3.10 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.

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

Reset a Pipe to an USB Endpoint.

**Parameters**

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

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

**11.30.3.13 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.
---------------	---------------------

**11.30.3.14 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.
---------------	---------------------

**11.30.3.15 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.
---------------	---------------------

**11.30.3.16 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.
---------------	---------------------

**11.30.3.17 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.
---------------	---------------------

**11.30.3.18 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.
---------------	---------------------

**11.30.3.19 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.
--------------	---------------------------

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

Write data to an USB Endpoint.

Parameters

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

Parameters

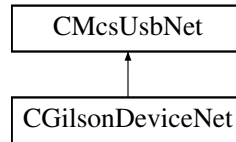
<i>buffer</i>	An array of data to write.
---------------	----------------------------

## 11.31 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

#### 11.31.1 Detailed Description

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

#### 11.31.2 Constructor & Destructor Documentation

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

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

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

#### 11.31.3 Member Function Documentation

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

**11.31.3.2 GetLastAnswer()** `String ^ GetLastAnswer ( )`

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

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

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

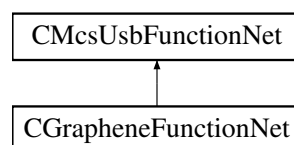
#### 11.31.4 Member Data Documentation

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

## 11.32 CGrapheneFunctionNet Class Reference

[CGrapheneFunctionNet](#) is the class to control the TEER device

Inheritance diagram for CGrapheneFunctionNet:



## Public Member Functions

- [CGrapheneFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> pGrapheneFunctionPointerContainer)  
*Initializes a new instance of the [CGrapheneFunctionNet](#) class.*
- [CGrapheneFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- virtual [~CGrapheneFunctionNet](#) ()
- [ICGrapheneFunctionNet](#) ()
- void [GetVdVsDAC](#) ([System::Runtime::InteropServices::Out]int16\_t% Vd, [System::Runtime::InteropServices::Out]int16\_t% Vs)  
*Gets Vd and Vs*
- void [GetVdVsDAC](#) (uint32\_t Headstage, [System::Runtime::InteropServices::Out]int16\_t% Vd, [System::Runtime::InteropServices::Out]int16\_t% Vs)  
*Gets Vd and Vs*
- void [SetVdVsDAC](#) (int16\_t Vd, int16\_t Vs)  
*Sets Vd and Vs*
- void [SetVdVsDAC](#) (uint32\_t Headstage, int16\_t Vd, int16\_t Vs)  
*Sets Vd and VS*
- bool [GetVoltageReached](#) ()  
*Gets the reached voltage*
- bool [GetVoltageReached](#) (uint32\_t Headstage)  
*Gets the reached voltage*
- int32\_t [GetVoltageRange](#) ()  
*Gets the voltage range*
- int32\_t [GetVoltageRange](#) (uint32\_t Headstage)  
*Gets the voltage range*
- void [SetVoltageRange](#) (int32\_t range)  
*Sets the voltage range*
- void [SetVoltageRange](#) (uint32\_t Headstage, int32\_t range)  
*Sets the voltage range*
- int32\_t [GetVoltageResolution](#) ()  
*Gets the voltage resolution*
- int32\_t [GetVoltageResolution](#) (uint32\_t Headstage)  
*Gets the voltage resolution*
- void [SetVoltageResolution](#) (int32\_t resolution)  
*Sets the voltage resolution*
- void [SetVoltageResolution](#) (uint32\_t Headstage, int32\_t resolution)  
*Sets the voltage resolution*
- void [GetDACOffset](#) ([System::Runtime::InteropServices::Out]int16\_t% offset\_vd, [System::Runtime::InteropServices::Out]int16\_t% offset\_vs)  
*Gets the DAC offset*
- void [GetDACOffset](#) (uint32\_t Headstage, [System::Runtime::InteropServices::Out]int16\_t% offset\_vd, [System::Runtime::InteropServices::Out]int16\_t% offset\_vs)  
*Gets the DAC offset*
- void [SetDACOffset](#) (int16\_t offset\_vd, int16\_t offset\_vs)  
*Sets the DAC offset*
- void [SetDACOffset](#) (uint32\_t Headstage, int16\_t offset\_vd, int16\_t offset\_vs)  
*Set the DAC offset*
- void [GetVdVs](#) ([System::Runtime::InteropServices::Out]int32\_t% Vd, [System::Runtime::InteropServices::Out]int32\_t% Vs)  
*Gets Vd and Vs*

- void [GetVdVs](#) (uint32\_t Headstage, [System::Runtime::InteropServices::Out]int32\_t% Vd, [System::Runtime::InteropServices::Out]int32\_t% Vs)  
*Gets Vd and Vs*
- void [SetVdVs](#) (int32\_t Vd, int32\_t Vs)  
*Sets Vd and Vs*
- void [SetVdVs](#) (uint32\_t Headstage, int32\_t Vd, int32\_t Vs)  
*Sets Vd and Vs*
- void [SetVgs](#) (int32\_t Vgs)  
*Sets Vgs*
- void [SetVgs](#) (uint32\_t Headstage, int32\_t Vgs)  
*Sets Vgs*
- void [SetVds](#) (int32\_t Vds)  
*Sets Vds*
- void [SetVds](#) (uint32\_t Headstage, int32\_t Vds)  
*Sets Vds*

## Additional Inherited Members

### 11.32.1 Detailed Description

[CGrapheneFunctionNet](#) is the class to control the TEER device

### 11.32.2 Constructor & Destructor Documentation

**11.32.2.1 CGrapheneFunctionNet()** [1/2] [CGrapheneFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> *mcsusb*,  
[CMcsUsbFunctionPointerContainer](#)<sup>^</sup> *pGrapheneFunctionPointerContainer* )

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

**11.32.2.2 CGrapheneFunctionNet()** [2/2] [CGrapheneFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> *mcsusb* )

**11.32.2.3 ~CGrapheneFunctionNet()** virtual [~CGrapheneFunctionNet](#) ( ) [virtual]

**11.32.2.4 "!CGrapheneFunctionNet()** [!CGrapheneFunctionNet](#) ( )

### 11.32.3 Member Function Documentation

**11.32.3.1 GetDACOffset()** [1/2] void GetDACOffset (  
[System::Runtime::InteropServices::Out] int16\_t% *offset\_vd*,  
[System::Runtime::InteropServices::Out] int16\_t% *offset\_vs* )

Gets the DAC offset

## Parameters

<i>offset_vd</i>	Vd offset
<i>offset_vs</i>	Vs offset

**11.32.3.2 GetDACOffset()** [2/2] void GetDACOffset (   
     uint32\_t *Headstage*,   
     [System::Runtime::InteropServices::Out] int16\_t% *offset\_vd*,   
     [System::Runtime::InteropServices::Out] int16\_t% *offset\_vs* )

Gets the DAC offset

## Parameters

<i>Headstage</i>	The headstage to query.
<i>offset_vd</i>	Vd offset
<i>offset_vs</i>	Vs offset

**11.32.3.3 GetVdVs()** [1/2] void GetVdVs (   
     [System::Runtime::InteropServices::Out] int32\_t% *Vd*,   
     [System::Runtime::InteropServices::Out] int32\_t% *Vs* )

Gets Vd and Vs

## Parameters

<i>Vd</i>	Vd
<i>Vs</i>	Vs

**11.32.3.4 GetVdVs()** [2/2] void GetVdVs (   
     uint32\_t *Headstage*,   
     [System::Runtime::InteropServices::Out] int32\_t% *Vd*,   
     [System::Runtime::InteropServices::Out] int32\_t% *Vs* )

Gets Vd and Vs

## Parameters

<i>Headstage</i>	The headstage to query.
<i>Vd</i>	Vd
<i>Vs</i>	Vs

**11.32.3.5 GetVdVsDAC()** [1/2] void GetVdVsDAC (   
[System::Runtime::InteropServices::Out] int16\_t% Vd,   
[System::Runtime::InteropServices::Out] int16\_t% Vs )

Gets Vd and Vs

Parameters

<i>Vd</i>	Vd
<i>Vs</i>	Vs

**11.32.3.6 GetVdVsDAC()** [2/2] void GetVdVsDAC (   
uint32\_t Headstage,   
[System::Runtime::InteropServices::Out] int16\_t% Vd,   
[System::Runtime::InteropServices::Out] int16\_t% Vs )

Gets Vd and Vs

Parameters

<i>Headstage</i>	The headstage to query.
<i>Vd</i>	Vd
<i>Vs</i>	Vs

**11.32.3.7 GetVoltageRange()** [1/2] int32\_t GetVoltageRange ( )

Gets the voltage range

Returns

The voltage range

**11.32.3.8 GetVoltageRange()** [2/2] int32\_t GetVoltageRange (   
uint32\_t Headstage )

Gets the voltage range

Parameters

<i>Headstage</i>	The headstage to query.
------------------	-------------------------

**Returns**

The voltage range

**11.32.3.9 GetVoltageReached() [1/2]** `bool GetVoltageReached ( )`

Gets the reached voltage

**Returns**

the reached voltage

**11.32.3.10 GetVoltageReached() [2/2]** `bool GetVoltageReached (   
uint32_t Headstage )`

Gets the reached voltage

**Parameters**

<i>Headstage</i>	The headstage to query.
------------------	-------------------------

**Returns**

The reached voltage

**11.32.3.11 GetVoltageResolution() [1/2]** `int32_t GetVoltageResolution ( )`

Gets the voltage resolution

**Returns**

The voltage resolution

**11.32.3.12 GetVoltageResolution() [2/2]** `int32_t GetVoltageResolution (   
uint32_t Headstage )`

Gets the voltage resolution

**Parameters**

<i>Headstage</i>	The headstage to query.
------------------	-------------------------

**Returns**

The voltage resolution

**11.32.3.13 SetDACOffset() [1/2]** `void SetDACOffset (`  
`int16_t offset_vd,`  
`int16_t offset_vs )`

Sets the DAC offset

**Parameters**

<i>offset_vd</i>	Vd
<i>offset_vs</i>	Vs

**11.32.3.14 SetDACOffset() [2/2]** `void SetDACOffset (`  
`uint32_t Headstage,`  
`int16_t offset_vd,`  
`int16_t offset_vs )`

Set the DAC offset

**Parameters**

<i>Headstage</i>	The headstage to query.
<i>offset_vd</i>	Vd
<i>offset_vs</i>	Vs

**11.32.3.15 SetVds() [1/2]** `void SetVds (`  
`int32_t Vds )`

Sets Vds

**Parameters**

<i>Vds</i>	Vds
------------	-----

**11.32.3.16 SetVds() [2/2]** `void SetVds (`  
`uint32_t Headstage,`  
`int32_t Vds )`

Sets Vds



## Parameters

<i>Headstage</i>	The headstage to query.
<i>Vds</i>	Vds

**11.32.3.17 SetVdVs() [1/2]** `void SetVdVs (`  
`int32_t Vd,`  
`int32_t Vs )`

Sets Vd and Vs

## Parameters

<i>Vd</i>	Vd
<i>Vs</i>	Vs

**11.32.3.18 SetVdVs() [2/2]** `void SetVdVs (`  
`uint32_t Headstage,`  
`int32_t Vd,`  
`int32_t Vs )`

Sets Vd and Vs

## Parameters

<i>Headstage</i>	The headstage to query.
<i>Vd</i>	Vd
<i>Vs</i>	Vs

**11.32.3.19 SetVdVsDAC() [1/2]** `void SetVdVsDAC (`  
`int16_t Vd,`  
`int16_t Vs )`

Sets Vd and Vs

## Parameters

<i>Vd</i>	Vd
<i>Vs</i>	Vs

**11.32.3.20 SetVdVsDAC()** [2/2] `void SetVdVsDAC (`  
     `uint32_t Headstage,`  
     `int16_t Vd,`  
     `int16_t Vs )`

Sets Vd and VS

Parameters

<i>Headstage</i>	The headstage to query.
<i>Vd</i>	Vd
<i>Vs</i>	Vs

**11.32.3.21 SetVgs()** [1/2] `void SetVgs (`  
     `int32_t Vgs )`

Sets Vgs

Parameters

<i>Vgs</i>	Vgs
------------	-----

**11.32.3.22 SetVgs()** [2/2] `void SetVgs (`  
     `uint32_t Headstage,`  
     `int32_t Vgs )`

Sets Vgs

Parameters

<i>Headstage</i>	The headstage to query.
<i>Vgs</i>	Vgs

**11.32.3.23 SetVoltageRange()** [1/2] `void SetVoltageRange (`  
     `int32_t range )`

Sets the voltage range

Parameters

<i>range</i>	The voltage range
--------------	-------------------

**11.32.3.24 SetVoltageRange()** [2/2] `void SetVoltageRange (`  
     `uint32_t Headstage,`  
     `int32_t range )`

Sets the voltage range

Parameters

<i>Headstage</i>	The headstage to query.
<i>range</i>	The voltage range

**11.32.3.25 SetVoltageResolution()** [1/2] `void SetVoltageResolution (`  
     `int32_t resolution )`

Sets the voltage resolution

Parameters

<i>resolution</i>	The voltage resolution
-------------------	------------------------

**11.32.3.26 SetVoltageResolution()** [2/2] `void SetVoltageResolution (`  
     `uint32_t Headstage,`  
     `int32_t resolution )`

Sets the voltage resolution

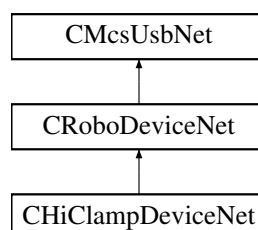
Parameters

<i>Headstage</i>	The headstage to query.
<i>resolution</i>	The voltage resolution

## 11.33 CHiClampDeviceNet Class Reference

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

Inheritance diagram for CHiClampDeviceNet:



## Public Member Functions

- **CHiClampDeviceNet** (void)

**Public Member Functions**

- [CHLADacqNet](#) (void)

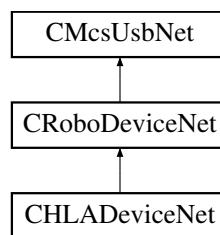
**Additional Inherited Members****11.34.1 Constructor & Destructor Documentation**

**11.34.1.1 CHLADacqNet()** [CHLADacqNet](#) (  
void )

**11.35 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****11.35.1 Detailed Description**

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

**11.35.2 Constructor & Destructor Documentation**

**11.35.2.1 CHLADeviceNet()** [CHLADeviceNet](#) (  
void )

### 11.35.3 Property Documentation

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

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

## 11.36 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)

### 11.36.1 Detailed Description

Class to provide hardware information about the device.

## 11.36.2 Constructor & Destructor Documentation

**11.36.2.1 CHWInfo()** [CHWInfo](#) (  
[CMcsUsbDacqNet](#)<sup>^</sup> *device* )

## 11.36.3 Member Function Documentation

**11.36.3.1 GetAvailableSampleRates()** [virtual uint32\\_t GetAvailableSampleRates](#) (  
[System::Runtime::InteropServices::Out] [System::Collections::Generic::List](#)< [int32](#)↔  
\_t >^% *sampleRates* ) [virtual]

**11.36.3.2 GetAvailableVoltageRangesInMicroVolt()** [virtual uint32\\_t GetAvailableVoltageRangesIn](#)↔  
[MicroVolt](#) (  
[System::Runtime::InteropServices::Out] [System::Collections::Generic::List](#)< [int32](#)↔  
\_t >^% *voltageRanges* ) [virtual]

**11.36.3.3 GetAvailableVoltageRangesInMicroVoltAndStringsInMilliVolt()** [virtual uint32\\_t GetAvailable](#)↔  
[VoltageRangesInMicroVoltAndStringsInMilliVolt](#) (  
[System::Runtime::InteropServices::Out] [System::Collections::Generic::List](#)< [CVoltageRangeInfoNet](#)  
*voltageRanges* ) [virtual]

**11.36.3.4 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.

**11.36.3.5 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.

**11.36.3.6 IsDigitalChannelDedicated()** `virtual bool IsDigitalChannelDedicated ( ) [virtual]`

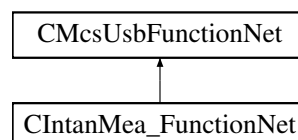
Query if the digital channel replaces an analog channel when enabled (e.g. on MC\_Card) or adds a channel link on USB devices.

#### 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.

## 11.37 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)
- [GetUpperFrequencyByIndex](#) (unsigned short index)
- [GetLowerFrequencyByIndex](#) (unsigned short index)
- [GetDSPHighPassByIndex](#) (unsigned short index)
- [GetIntanRegister](#) (unsigned short chip, unsigned short registernumber)
- [GetImpedanceResult](#) (unsigned short channel)
- [SetBandwidthByIndex](#) (int upper\_index, int lower\_index)
- [SetDSPHighPassByIndex](#) (int index)
- [AmplifierSettle](#) ()
- [SetIntanRegister](#) (unsigned short register\_number, int value)
- [SetDiagnosticMode](#) (unsigned char onoff)
- [BeginImpedanceCheck](#) (array< int ><sup>^</sup> config\_values)



## Additional Inherited Members

### 11.37.1 Constructor & Destructor Documentation

**11.37.1.1 CIntanMea\_FunctionNet()** [1/2] CIntanMea\_FunctionNet (   
 CMcsUsbNet^ *mcsusb*,   
 CMcsUsbFunctionPointerContainer^ *intalMea\_FunctionPointerContainer* )

**11.37.1.2 CIntanMea\_FunctionNet()** [2/2] CIntanMea\_FunctionNet (   
 CMcsUsbNet^ *mcsusb* )

### 11.37.2 Member Function Documentation

**11.37.2.1 AmplifierSettle()** void AmplifierSettle ( )

**11.37.2.2 BeginImpedanceCheck()** void BeginImpedanceCheck (   
 array< int >^ *config\_values* )

**11.37.2.3 GetDSPHighPassByIndex()** int64\_t GetDSPHighPassByIndex (   
 unsigned short *index* )

**11.37.2.4 GetImpedanceResult()** int GetImpedanceResult (   
 unsigned short *channel* )

**11.37.2.5 GetIntanRegister()** int GetIntanRegister (   
 unsigned short *chip*,   
 unsigned short *registernumber* )

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

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

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

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

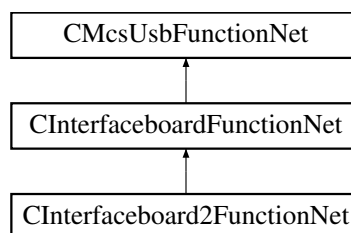
**11.37.2.10 SetDSPHighPassByIndex()** `void SetDSPHighPassByIndex (`  
`int index )`

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

## 11.38 CInterfaceboard2FunctionNet Class Reference

[CInterfaceboard2FunctionNet](#) is the class to control the Interfaceboard

Inheritance diagram for CInterfaceboard2FunctionNet:



## Public Member Functions

- [CInterfaceboard2FunctionNet](#) ([CMcsUsbNet](#)^ mcsusb, [CMcsUsbFunctionPointerContainer](#)^ pInterfaceboard2FunctionPointerContainer)  
*Initializes a new instance of the [CInterfaceboard2FunctionNet](#) class.*
- [CInterfaceboard2FunctionNet](#) ([CMcsUsbNet](#)^ mcsusb)
- virtual [~CInterfaceboard2FunctionNet](#) ()
- [!CInterfaceboard2FunctionNet](#) ()
- void [SetIoVoltage](#) (IoVoltageEnumNet ioVoltage)  
*Sets the I/O Voltage level for the IFB2 digital and AUX ports, default is 3.3V.*
- IoVoltageEnumNet [GetIoVoltage](#) ()  
*Gets the I/O Voltage level for the IFB2 digital and AUX ports, default is 3.3V.*

## Additional Inherited Members

### 11.38.1 Detailed Description

[CInterfaceboard2FunctionNet](#) is the class to control the Interfaceboard

### 11.38.2 Constructor & Destructor Documentation

**11.38.2.1 CInterfaceboard2FunctionNet()** [1/2] [CInterfaceboard2FunctionNet](#) (  
[CMcsUsbNet](#)^ mcsusb,  
[CMcsUsbFunctionPointerContainer](#)^ pInterfaceboard2FunctionPointerContainer )

Initializes a new instance of the [CInterfaceboard2FunctionNet](#) class.

**11.38.2.2 CInterfaceboard2FunctionNet()** [2/2] [CInterfaceboard2FunctionNet](#) (  
[CMcsUsbNet](#)^ mcsusb )

**11.38.2.3 ~CInterfaceboard2FunctionNet()** virtual [~CInterfaceboard2FunctionNet](#) ( ) [virtual]

**11.38.2.4 !CInterfaceboard2FunctionNet()** [!CInterfaceboard2FunctionNet](#) ( )

### 11.38.3 Member Function Documentation

**11.38.3.1 GetIoVoltage()** `IoVoltageEnumNet GetIoVoltage ( )`

Gets the I/O Voltage level for the IFB2 digital and AUX ports, default is 3.3V.

**Returns**

Enum for the IO Voltage (3.3V or 5.0V).

**11.38.3.2 SetIoVoltage()** `void SetIoVoltage (   
IoVoltageEnumNet ioVoltage )`

Sets the I/O Voltage level for the IFB2 digital and AUX ports, default is 3.3V.

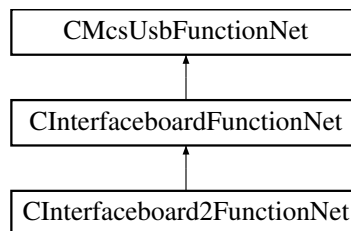
**Parameters**

<i>ioVoltage</i>	Enum for the I/O Voltage (3.3V or 5.0V).
------------------	--

**11.39 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> pInterfaceboard↔  
FunctionPointerContainer)  
*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

### 11.39.1 Detailed Description

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

### 11.39.2 Constructor & Destructor Documentation

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

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

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

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

**11.39.2.4 !"CInterfaceboardFunctionNet()** ![CInterfaceboardFunctionNet](#) ( )

### 11.39.3 Member Function Documentation

**11.39.3.1 GetCardinalDacqSamplerate()** uint32\_t GetCardinalDacqSamplerate ( )

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

Returns

The samplerate in Hz.

**11.39.3.2 GetCardinalStgOutputrate()** uint32\_t GetCardinalStgOutputrate ( )

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

Returns

The output rate in Hz.

**11.39.3.3 SetCardinalDacqSamplerate()** void SetCardinalDacqSamplerate (  
    uint32\_t *samplerate* )

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

## Parameters

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

**11.39.3.4 SetCardinalStgOutputrate()** `void SetCardinalStgOutputrate ( uint32_t outputrate )`

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

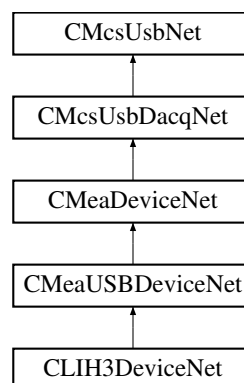
## Parameters

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

## 11.40 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, int32\_t Offset)
- Sets the ADC offset of the DACQ for a single channel*
- int32\_t [GetAdcOffset](#) (LIH30\_ADC\_Channel\_EnumNet AdcChannel)
- Gets the ADC offset of the DACQ for a single channel*
- void [SetAdcOffsetPermanent](#) (LIH30\_ADC\_Channel\_EnumNet AdcChannel)
- Writes the ADC offset of the DACQ for a single channel to permanent EEPROM memory*
- void [ErasePermanentAdcOffset](#) (LIH30\_ADC\_Channel\_EnumNet AdcChannel)
- Deletes the ADC offset of the DACQ for a single channel in permanent EEPROM memory*
- 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*
- void [SetDacOffset](#) (LIH30\_DAC\_Channel\_EnumNet DacChannel, int32\_t Offset)
- Sets the offset of a DAC channel.*
- int32\_t [GetDacOffset](#) (LIH30\_DAC\_Channel\_EnumNet DacChannel)
- Gets the offset of a DAC channel.*
- void [SetDacOffsetPermanent](#) (LIH30\_DAC\_Channel\_EnumNet DacChannel)
- Writes the DAC offset of the STG for a single channel to permanent EEPROM memory*
- void [ErasePermanentDacOffset](#) (LIH30\_DAC\_Channel\_EnumNet DacChannel)
- Deletes the DAC offset of the STG for a single channel in permanent EEPROM memory*
- void [SetAudioOutDacParameter](#) (uint32\_t Frequency, uint32\_t Amplification)
- Sets the parameter of the audio DAC output.*
- void [GetAudioOutDacParameter](#) ([System::Runtime::InteropServices::Out]uint32\_t% Frequency, [System::Runtime::InteropServices::Out]uint32\_t% Amplification)
- Gets the parameter of the audio DAC output.*

## Properties

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

## Additional Inherited Members

### 11.40.1 Detailed Description

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

### 11.40.2 Constructor & Destructor Documentation

#### 11.40.2.1 CLIH3DeviceNet() [CLIH3DeviceNet](#) ( )

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

#### 11.40.2.2 ~CLIH3DeviceNet() [virtual](#) [~CLIH3DeviceNet](#) ( ) [\[virtual\]](#)

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

### 11.40.3 Member Function Documentation

#### 11.40.3.1 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
-----------------------	---------------------------------

#### 11.40.3.2 EnableUserTrigger() [void](#) [EnableUserTrigger](#) ( [bool](#) *enable* )

Enables the User Trigger

##### Parameters

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



**11.40.3.3 ErasePermanentAdcOffset()** `void ErasePermanentAdcOffset (`  
`LIH30_ADC_Channel_EnumNet AdcChannel )`

Deletes the ADC offset of the DACQ for a single channel in permanent EEPROM memory

Parameters

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

**11.40.3.4 ErasePermanentDacOffset()** `void ErasePermanentDacOffset (`  
`LIH30_DAC_Channel_EnumNet DacChannel )`

Deletes the DAC offset of the STG for a single channel in permanent EEPROM memory

Parameters

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

**11.40.3.5 GetAdcOffset()** `int32_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

**11.40.3.6 GetAudioOutDacParameter()** `void GetAudioOutDacParameter (`  
`[System::Runtime::InteropServices::Out] uint32_t% Frequency,`  
`[System::Runtime::InteropServices::Out] uint32_t% Amplification )`

Gets the parameter of the audio DAC output.

Parameters

<i>Frequency</i>	Frequency(1 - 25000 Hz)
<i>Amplification</i>	Amplification(0 - 0xFFFF)

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

Gets the DAC Idle value

**Parameters**

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

**Returns**

The idle value

**11.40.3.8 GetDacOffset()** `int32_t GetDacOffset (   
 LIH30_DAC_Channel_EnumNet DacChannel )`

Gets the offset of a DAC channel.

**Parameters**

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

**Returns**

The offset for the given channel number

**11.40.3.9 GetDacqRunStatus()** `uint16_t GetDacqRunStatus ( )`

Gets the data acquisition running status

**Returns**

The status (1: running / 0: stopped)

**11.40.3.10 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

**11.40.3.11 GetDigInState()** `uint16_t GetDigInState ( )`

Reads from the LIH30 digital input

## Returns

The bit mask defining the digital input state

**11.40.3.12 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

**11.40.3.13 GetSampleInterval()** `uint32_t GetSampleInterval ( )`

Gets the Sample Interval for the DACQ and Stimulation

## Returns

Sample Interval configured on the device

**11.40.3.14 IsUserTriggerEnabled()** `bool IsUserTriggerEnabled ( )`

Is the User Trigger enabled

**Returns**

Enabled

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

Gets the clipping information

**Parameters**

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

**Returns**

The clipping value

**11.40.3.16 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

**11.40.3.17 SetAdcOffset()** `void SetAdcOffset (`  
`LIH30_ADC_Channel_EnumNet AdcChannel,`  
`int32_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

**11.40.3.18 SetAdcOffsetPermanent()** `void SetAdcOffsetPermanent (`  
`LIH30_ADC_Channel_EnumNet AdcChannel )`

Writes the ADC offset of the DACQ for a single channel to permanent EEPROM memory

**Parameters**

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

**11.40.3.19 SetAudioOutDacParameter()** `void SetAudioOutDacParameter (`  
`uint32_t Frequency,`  
`uint32_t Amplification )`

Sets the parameter of the audio DAC output.

**Parameters**

<i>Frequency</i>	Frequency(1 - 25000 Hz)
<i>Amplification</i>	Amplification(0 - 0xFFFF)

**11.40.3.20 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

**11.40.3.21 SetDacOffset()** `void SetDacOffset (`  
`LIH30_DAC_Channel_EnumNet DacChannel,`  
`int32_t Offset )`

Sets the offset of a DAC channel.

**Parameters**

<i>DacChannel</i>	The DAC channel
<i>Offset</i>	The offset for the given channel number

**11.40.3.22 SetDacOffsetPermanent()** `void SetDacOffsetPermanent (`  
`LIH30_DAC_Channel_EnumNet DacChannel )`

Writes the DAC offset of the STG for a single channel to permanent EEPROM memory

Parameters

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

**11.40.3.23 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

**11.40.3.24 SetDigOutState()** `void SetDigOutState (`  
`uint16_t DigOutState )`

Writes to the LIH30 digital output

Parameters

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

**11.40.3.25 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

**11.40.3.26 SetSampleInterval()** `void SetSampleInterval (`  
`uint32_t SampleInterval )`

Sets the Sample Interval for the DACQ and Stimulation

Parameters

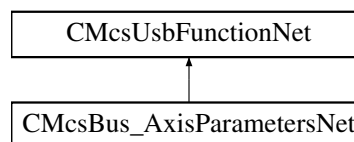
<i>SampleInterval</i>	between the samples, Sample interval is available from 1 to 4194303
-----------------------	---

#### 11.40.4 Property Documentation

**11.40.4.1 StimulusFunction** `CStimulusFunctionNet^ StimulusFunction [get]`

### 11.41 CMcsBus\_AxisParametersNet Class Reference

Inheritance diagram for CMcsBus\_AxisParametersNet:



#### Public Member Functions

- `CMcsBus_AxisParametersNet (CMcsUsbNet^ 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

##### 11.41.1 Constructor & Destructor Documentation

**11.41.1.1 CMcsBus\_AxisParametersNet()** `CMcsBus_AxisParametersNet ( CMcsUsbNet^ device )`

**11.41.1.2 ~CMcsBus\_AxisParametersNet()** `~CMcsBus_AxisParametersNet ( void )`

## 11.41.2 Member Function Documentation

**11.41.2.1 GetAxisParametersSignedEeprom()** `int GetAxisParametersSignedEeprom ( unsigned char busnumber, unsigned char busaddress, unsigned char axis, unsigned short index )`

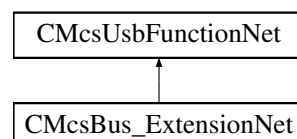
**11.41.2.2 GetAxisParametersUnsignedEeprom()** `unsigned int GetAxisParametersUnsignedEeprom ( unsigned char busnumber, unsigned char busaddress, unsigned char axis, unsigned short index )`

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

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

## 11.42 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****11.42.1 Constructor & Destructor Documentation**

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

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

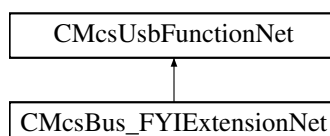
**11.42.2 Member Function Documentation**

**11.42.2.1 GetLEDSwitch()** unsigned short [GetLEDSwitch](#) (  
unsigned char *busnumber*,  
unsigned char *busaddress* )

**11.42.2.2 SetLEDSwitch()** void [SetLEDSwitch](#) (  
unsigned char *busnumber*,  
unsigned char *busaddress*,  
unsigned short *LEDSwitch* )

**11.43 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****11.43.1 Constructor & Destructor Documentation**

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

**11.43.1.2 [~CMcsBus\\_FYIExtensionNet\(\)](#)** [~CMcsBus\\_FYIExtensionNet](#) (  
void )

**11.43.2 Member Function Documentation**

**11.43.2.1 [GetDIO\(\)](#)** unsigned short [GetDIO](#) (  
unsigned char *busnumber*,  
unsigned char *busaddress* )

**11.43.2.2 [GetSingleHeater\(\)](#)** unsigned short [GetSingleHeater](#) (  
unsigned char *busnumber*,  
unsigned char *busaddress*,  
short *index* )

**11.43.2.3 [GetValves\(\)](#)** unsigned int [GetValves](#) (  
unsigned char *busnumber*,  
unsigned char *busaddress* )

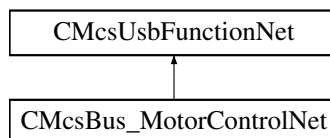
**11.43.2.4 SetDIO()** void SetDIO (   
     unsigned char *busnumber*,   
     unsigned char *busaddress*,   
     unsigned short *io* )

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

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

## 11.44 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 [SetMCOOutputOnOff](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, bool OnOff\_status)
- bool [GetMCOOutputOnOff](#) (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

### 11.44.1 Constructor & Destructor Documentation

**11.44.1.1 CMcsBus\_MotorControlNet()** [CMcsBus\\_MotorControlNet](#) ( [CMcsUsbNet](#)^ device )

**11.44.1.2 ~CMcsBus\_MotorControlNet()** [~CMcsBus\\_MotorControlNet](#) ( void )

### 11.44.2 Member Function Documentation

**11.44.2.1 GetMCAcceleration()** unsigned short [GetMCAcceleration](#) ( unsigned char *busnumber*, unsigned char *busaddress*, unsigned char *axis* )

**11.44.2.2 GetMCAccelerationEeprom()** unsigned short GetMCAccelerationEeprom (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

**11.44.2.3 GetMCAccelerationShortCommand()** unsigned short GetMCAccelerationShortCommand (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

**11.44.2.4 GetMCAxisRevisionEeprom()** unsigned short GetMCAxisRevisionEeprom (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

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

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

**11.44.2.7 GetMCConfig()** unsigned short GetMCConfig (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

**11.44.2.8 GetMCConfigEeprom()** unsigned short GetMCConfigEeprom (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

**11.44.2.9 GetMCCurrent()** short GetMCCurrent (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

**11.44.2.10 GetMCCurrentEeprom()** short GetMCCurrentEeprom (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

**11.44.2.11 GetMCCurrentMode()** RoboCurrentModeEnumNet GetMCCurrentMode (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

**11.44.2.12 GetMCCurrentModeEeprom()** RoboCurrentModeEnumNet GetMCCurrentModeEeprom (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

**11.44.2.13 GetMCCurrentModeShortCommand()** RoboCurrentModeEnumNet GetMCCurrentModeShort↔  
Command (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

**11.44.2.14 GetMCCurrentPosition()** int GetMCCurrentPosition (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

**11.44.2.15 GetMCCurrentShortCommand()** short GetMCCurrentShortCommand (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )



**11.44.2.16 GetMCCurrentSpeed()** short GetMCCurrentSpeed (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )

**11.44.2.17 GetMCMaxAcceleration()** unsigned short GetMCMaxAcceleration (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )

**11.44.2.18 GetMCMaxAccelerationEeprom()** unsigned short GetMCMaxAccelerationEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )

**11.44.2.19 GetMCMaxCurrent()** short GetMCMaxCurrent (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )

**11.44.2.20 GetMCMaxCurrentEeprom()** short GetMCMaxCurrentEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )

**11.44.2.21 GetMCMaxSpeed()** unsigned short GetMCMaxSpeed (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )

**11.44.2.22 GetMCMaxSpeedEeprom()** unsigned short GetMCMaxSpeedEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )

- 11.44.2.23 GetMCMaxTravel()** `int GetMCMaxTravel ( unsigned char busnumber, unsigned char busaddress, unsigned char axis )`
- 11.44.2.24 GetMCMaxTravelEeprom()** `int GetMCMaxTravelEeprom ( unsigned char busnumber, unsigned char busaddress, unsigned char axis )`
- 11.44.2.25 GetMCMaxTravelShortCommand()** `int GetMCMaxTravelShortCommand ( unsigned char busnumber, unsigned char busaddress, unsigned char axis )`
- 11.44.2.26 GetMCMovement()** `unsigned short GetMCMovement ( unsigned char busnumber, unsigned char busaddress, unsigned char axis )`
- 11.44.2.27 GetMCNewPosition()** `int GetMCNewPosition ( unsigned char busnumber, unsigned char busaddress, unsigned char axis )`
- 11.44.2.28 GetMCOutputOnOff()** `bool GetMCOutputOnOff ( unsigned char busnumber, unsigned char busaddress, unsigned char axis )`
- 11.44.2.29 GetMCPhase()** `unsigned short GetMCPhase ( unsigned char busnumber, unsigned char busaddress, unsigned char axis )`

**11.44.2.30 GetMCPhaseOffset()** unsigned short GetMCPhaseOffset (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

**11.44.2.31 GetMCReference()** unsigned char GetMCReference (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis*,  
 [System::Runtime::InteropServices::Out] unsigned char% *switch\_port* )

**11.44.2.32 GetMCReferenceCurrent()** short GetMCReferenceCurrent (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

**11.44.2.33 GetMCReferenceCurrentEeprom()** short GetMCReferenceCurrentEeprom (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

**11.44.2.34 GetMCRegulatorGain()** short GetMCRegulatorGain (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

**11.44.2.35 GetMCRegulatorGainEeprom()** short GetMCRegulatorGainEeprom (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

**11.44.2.36 GetMCScalingFactor()** int GetMCScalingFactor (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

**11.44.2.37 GetMCScalingFactorEeprom()** int GetMCScalingFactorEeprom (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis* )

**11.44.2.38 GetMCSpeed()** short GetMCSpeed (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis* )

**11.44.2.39 GetMCSpeedEeprom()** unsigned short GetMCSpeedEeprom (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis* )

**11.44.2.40 GetMCSpeedShortCommand()** short GetMCSpeedShortCommand (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis* )

**11.44.2.41 GetMCSpeedUnitEeprom()** int32\_t GetMCSpeedUnitEeprom (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis* )

**11.44.2.42 GetMCStandbyCurrent()** short GetMCStandbyCurrent (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis* )

**11.44.2.43 GetMCStandbyCurrentEeprom()** short GetMCStandbyCurrentEeprom (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis* )

**11.44.2.44 GetMCStandbyTime()** short GetMCStandbyTime (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )

**11.44.2.45 GetMCStandbyTimeEeprom()** short GetMCStandbyTimeEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )

**11.44.2.46 GetSubChannel()** unsigned short GetSubChannel (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )

**11.44.2.47 SetMCAcceleration()** void SetMCAcceleration (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 unsigned short *acceleration* )

**11.44.2.48 SetMCAccelerationEeprom()** void SetMCAccelerationEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 unsigned short *acceleration* )

**11.44.2.49 SetMCAccelerationShortCommand()** void SetMCAccelerationShortCommand (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 unsigned short *acceleration* )

**11.44.2.50 SetMCAxisRevisionEeprom()** void SetMCAxisRevisionEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 unsigned short *revision* )

**11.44.2.51 SetMCBreakCurrent()** void SetMCBreakCurrent (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis*,  
    short *current* )

**11.44.2.52 SetMCBreakCurrentEeprom()** void SetMCBreakCurrentEeprom (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis*,  
    short *current* )

**11.44.2.53 SetMCConfig()** void SetMCConfig (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis*,  
    unsigned short *config* )

**11.44.2.54 SetMCConfigEeprom()** void SetMCConfigEeprom (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis*,  
    unsigned short *config* )

**11.44.2.55 SetMCCurrent()** void SetMCCurrent (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis*,  
    short *current* )

**11.44.2.56 SetMCCurrentEeprom()** void SetMCCurrentEeprom (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis*,  
    short *current* )

**11.44.2.57 SetMCCurrentMode()** void SetMCCurrentMode (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 RoboCurrentModeEnumNet *mode* )

**11.44.2.58 SetMCCurrentModeEeprom()** void SetMCCurrentModeEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 RoboCurrentModeEnumNet *mode* )

**11.44.2.59 SetMCCurrentModeShortCommand()** void SetMCCurrentModeShortCommand (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 RoboCurrentModeEnumNet *mode* )

**11.44.2.60 SetMCCurrentPosition()** void SetMCCurrentPosition (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 int *position* )

**11.44.2.61 SetMCCurrentShortCommand()** void SetMCCurrentShortCommand (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 short *current* )

**11.44.2.62 SetMCMaxAcceleration()** void SetMCMaxAcceleration (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 unsigned short *acceleration* )

**11.44.2.63 SetMCMaxAccelerationEeprom()** void SetMCMaxAccelerationEeprom (   
    unsigned char *busnumber*,   
    unsigned char *busaddress*,   
    unsigned char *axis*,   
    unsigned short *acceleration* )

**11.44.2.64 SetMCMaxCurrent()** void SetMCMaxCurrent (   
    unsigned char *busnumber*,   
    unsigned char *busaddress*,   
    unsigned char *axis*,   
    short *current* )

**11.44.2.65 SetMCMaxCurrentEeprom()** void SetMCMaxCurrentEeprom (   
    unsigned char *busnumber*,   
    unsigned char *busaddress*,   
    unsigned char *axis*,   
    short *current* )

**11.44.2.66 SetMCMaxSpeed()** void SetMCMaxSpeed (   
    unsigned char *busnumber*,   
    unsigned char *busaddress*,   
    unsigned char *axis*,   
    unsigned short *speed* )

**11.44.2.67 SetMCMaxSpeedEeprom()** void SetMCMaxSpeedEeprom (   
    unsigned char *busnumber*,   
    unsigned char *busaddress*,   
    unsigned char *axis*,   
    unsigned short *speed* )

**11.44.2.68 SetMCMaxTravel()** void SetMCMaxTravel (   
    unsigned char *busnumber*,   
    unsigned char *busaddress*,   
    unsigned char *axis*,   
    int *travel* )



**11.44.2.69 SetMCMaxTravelEeprom()** void SetMCMaxTravelEeprom (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis*,  
    int *travel* )

**11.44.2.70 SetMCMaxTravelShortCommand()** void SetMCMaxTravelShortCommand (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis*,  
    int *travel* )

**11.44.2.71 SetMCNewPosition()** void SetMCNewPosition (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis*,  
    int *position* )

**11.44.2.72 SetMCOutputOnOff()** void SetMCOutputOnOff (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis*,  
    bool *OnOff\_status* )

**11.44.2.73 SetMCReference()** void SetMCReference (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis*,  
    unsigned char *switch\_enable*,  
    unsigned char *switch\_polarity* )

**11.44.2.74 SetMCReferenceCurrent()** void SetMCReferenceCurrent (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis*,  
    short *current* )

**11.44.2.75 SetMCReferenceCurrentEeprom()** void SetMCReferenceCurrentEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 short *current* )

**11.44.2.76 SetMCRegulatorGain()** void SetMCRegulatorGain (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 short *gain* )

**11.44.2.77 SetMCRegulatorGainEeprom()** void SetMCRegulatorGainEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 short *gain* )

**11.44.2.78 SetMCRotation()** void SetMCRotation (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 unsigned char *onoff* )

**11.44.2.79 SetMCScalingFactor()** void SetMCScalingFactor (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 int *factor* )

**11.44.2.80 SetMCScalingFactorEeprom()** void SetMCScalingFactorEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 int *factor* )

**11.44.2.81 SetMCSpeed()** void SetMCSpeed (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 short *speed* )

**11.44.2.82 SetMCSpeedEeprom()** void SetMCSpeedEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 unsigned short *speed* )

**11.44.2.83 SetMCSpeedShortCommand()** void SetMCSpeedShortCommand (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 short *speed* )

**11.44.2.84 SetMCSpeedUnitEeprom()** void SetMCSpeedUnitEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 int32\_t *speedunit* )

**11.44.2.85 SetMCStandbyCurrent()** void SetMCStandbyCurrent (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 short *percent* )

**11.44.2.86 SetMCStandbyCurrentEeprom()** void SetMCStandbyCurrentEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 short *percent* )

**11.44.2.87 SetMCStandbyTime()** void SetMCStandbyTime (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 short *t* )

**11.44.2.88 SetMCStandbyTimeEeprom()** void SetMCStandbyTimeEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 short *t* )

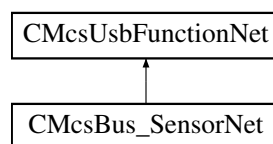
**11.44.2.89 SetSubChannel()** void SetSubChannel (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 unsigned short *subchannel* )

**11.44.2.90 StartMCMovement()** void StartMCMovement (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )

**11.44.2.91 StopMCMovement()** void StopMCMovement (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )

## 11.45 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 > <sup>^</sup> [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 > <sup>^</sup> [Get4ADC](#) (unsigned char busnumber, unsigned char busaddress)
- array< int > <sup>^</sup> [Get4ADCAverage](#) (unsigned char busnumber, unsigned char busaddress)
- void [Set4DAC](#) (unsigned char busnumber, unsigned char busaddress, array< unsigned short > <sup>^</sup> dac)
- array< unsigned short > <sup>^</sup> [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 > <sup>^</sup> [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

### 11.45.1 Constructor & Destructor Documentation

**11.45.1.1 CMcsBus\_SensorNet()** [CMcsBus\\_SensorNet](#) (   
 [CMcsUsbNet](#) ^ device )

**11.45.1.2 ~CMcsBus\_SensorNet()** [~CMcsBus\\_SensorNet](#) (   
 void )

### 11.45.2 Member Function Documentation

**11.45.2.1 CatchAmpGetAdcMean()** int CatchAmpGetAdcMean (   
 unsigned char *busnumber*,   
 unsigned char *busaddress* )

**11.45.2.2 CatchAmpGetAdcValue()** int CatchAmpGetAdcValue (   
 unsigned char *busnumber*,   
 unsigned char *busaddress* )

**11.45.2.3 CatchAmpGetAdcValueH()** int CatchAmpGetAdcValueH (   
 unsigned char *busnumber*,   
 unsigned char *busaddress* )

**11.45.2.4 CatchAmpGetAdcValueL()** int CatchAmpGetAdcValueL (   
 unsigned char *busnumber*,   
 unsigned char *busaddress* )

**11.45.2.5 CatchAmpGetDacAmplitude()** unsigned short CatchAmpGetDacAmplitude (   
 unsigned char *busnumber*,   
 unsigned char *busaddress* )

**11.45.2.6 CatchAmpGetDacEnable()** bool CatchAmpGetDacEnable (   
 unsigned char *busnumber*,   
 unsigned char *busaddress* )

**11.45.2.7 CatchAmpGetDacOffset()** short CatchAmpGetDacOffset (   
 unsigned char *busnumber*,   
 unsigned char *busaddress* )

**11.45.2.8 CatchAmpGetPwmEnable()** bool CatchAmpGetPwmEnable (   
 unsigned char *busnumber*,   
 unsigned char *busaddress* )

**11.45.2.9 CatchAmpSetDacAmplitude()** void CatchAmpSetDacAmplitude (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned short *dacAmplitude* )

**11.45.2.10 CatchAmpSetDacEnable()** void CatchAmpSetDacEnable (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    bool *dacEnable* )

**11.45.2.11 CatchAmpSetDacOffset()** void CatchAmpSetDacOffset (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    short *dacOffset* )

**11.45.2.12 CatchAmpSetPwmEnable()** void CatchAmpSetPwmEnable (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    bool *pwmEnable* )

**11.45.2.13 Get2AnalogInput()** array<unsigned short> ^ Get2AnalogInput (   
    unsigned char *busnumber*,  
    unsigned char *busaddress* )

**11.45.2.14 Get2DigitalInput()** unsigned short Get2DigitalInput (   
    unsigned char *busnumber*,  
    unsigned char *busaddress* )

**11.45.2.15 Get4ADC()** array<int> ^ Get4ADC (   
    unsigned char *busnumber*,  
    unsigned char *busaddress* )

**11.45.2.16 Get4ADCAverage()** array<int> ^ Get4ADCAverage (   
    unsigned char *busnumber*,  
    unsigned char *busaddress* )



**11.45.2.17 Get4ADCCatchampAverageShift()** unsigned int Get4ADCCatchampAverageShift (   
 unsigned char *busnumber*,   
 unsigned char *busaddress* )

**11.45.2.18 Get4ADCMode()** PatchServAdcModeEnumNet Get4ADCMode (   
 unsigned char *busnumber*,   
 unsigned char *busaddress* )

**11.45.2.19 Get4DAC()** array<unsigned short> ^ Get4DAC (   
 unsigned char *busnumber*,   
 unsigned char *busaddress* )

**11.45.2.20 GetADCs()** array<unsigned short> ^ GetADCs (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 int *n* )

**11.45.2.21 GetADCsLoop()** array<unsigned short> ^ GetADCsLoop (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 int *n* )

**11.45.2.22 GetBubbleStatus()** unsigned short GetBubbleStatus (   
 unsigned char *busnumber*,   
 unsigned char *busaddress* )

**11.45.2.23 GetDACs()** array<unsigned short> ^ GetDACs (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned short *index* )

**11.45.2.24 GetDetectionThreshold()** unsigned short GetDetectionThreshold (   
 unsigned char *busnumber*,   
 unsigned char *busaddress* )

**11.45.2.25 GetDetectorValue()** unsigned short GetDetectorValue (   
 unsigned char *busnumber*,   
 unsigned char *busaddress* )

**11.45.2.26 GetLatency()** unsigned short GetLatency (   
 unsigned char *busnumber*,   
 unsigned char *busaddress* )

**11.45.2.27 GetLatencyCounter()** unsigned short GetLatencyCounter (   
 unsigned char *busnumber*,   
 unsigned char *busaddress* )

**11.45.2.28 GetMinimalThreshold()** unsigned short GetMinimalThreshold (   
 unsigned char *busnumber*,   
 unsigned char *busaddress* )

**11.45.2.29 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* )

**11.45.2.30 GetPiezoState()** void GetPiezoState (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 [System::Runtime::InteropServices::Out] int% *state*,   
 [System::Runtime::InteropServices::Out] int% *reason* )

**11.45.2.31 GetPressure()** [1/2] array<int> ^ GetPressure (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 int *n* )

**11.45.2.32 GetPressure()** [2/2] int GetPressure (   
    unsigned char *busnumber*,   
    unsigned char *busaddress*,   
    unsigned short *index* )

**11.45.2.33 GetPressureOffset()** [1/2] array<unsigned short> ^ GetPressureOffset (   
    unsigned char *busnumber*,   
    unsigned char *busaddress* )

**11.45.2.34 GetPressureOffset()** [2/2] int GetPressureOffset (   
    unsigned char *busnumber*,   
    unsigned char *busaddress*,   
    unsigned short *index* )

**11.45.2.35 GetRegulationTimeouts()** void GetRegulationTimeouts (   
    unsigned char *busnumber*,   
    unsigned char *busaddress*,   
    [System::Runtime::InteropServices::Out] unsigned short% *MaxSpeedWait*,   
    [System::Runtime::InteropServices::Out] unsigned short% *MaxSignChange* )

**11.45.2.36 GetRegulatorFactor()** int GetRegulatorFactor (   
    unsigned char *busnumber*,   
    unsigned char *busaddress*,   
    unsigned short *index* )

**11.45.2.37 GetRegulatorOnOff()** unsigned char GetRegulatorOnOff (   
    unsigned char *busnumber*,   
    unsigned char *busaddress*,   
    unsigned short *index* )

**11.45.2.38 GetRegulatorStatus()** unsigned int GetRegulatorStatus (   
    unsigned char *busnumber*,   
    unsigned char *busaddress*,   
    unsigned short *index* )

- 11.45.2.39 GetRotatePump()** short GetRotatePump (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned short *index* )
- 11.45.2.40 GetSamplePeriode()** unsigned short GetSamplePeriode (   
 unsigned char *busnumber*,   
 unsigned char *busaddress* )

- 11.45.2.41 GetSollPressure()** int GetSollPressure (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned short *index* )

- 11.45.2.42 GetSyncState()** unsigned short GetSyncState (   
 unsigned char *busnumber*,   
 unsigned char *busaddress* )

- 11.45.2.43 Set4ADCCatchampAverageShift()** void Set4ADCCatchampAverageShift (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned int *shift* )

- 11.45.2.44 Set4ADCMode()** void Set4ADCMode (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 PatchServAdcModeEnumNet *mode* )

- 11.45.2.45 Set4DAC()** void Set4DAC (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 array< unsigned short >^ *dac* )

**11.45.2.46 SetDACs()** void SetDACs (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned short *index*,  
    array< unsigned short >^ *dac\_times\_voltages* )

**11.45.2.47 SetDetectionThreshold()** void SetDetectionThreshold (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned short *threshold* )

**11.45.2.48 SetLatency()** void SetLatency (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned short *latency* )

**11.45.2.49 SetMinimalThreshold()** void SetMinimalThreshold (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned short *threshold* )

**11.45.2.50 SetMovePump()** void SetMovePump (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned short *index*,  
    unsigned short *speed*,  
    int *position* )

**11.45.2.51 SetPiezoState()** void SetPiezoState (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    int *state* )

**11.45.2.52 SetPressureOffset()** void SetPressureOffset (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned short *index* )

**11.45.2.53 SetRegulationTimeouts()** void SetRegulationTimeouts (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned short *MaxSpeedWait*,  
    unsigned short *MaxSignChange* )

**11.45.2.54 SetRegulatorFactor()** void SetRegulatorFactor (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned short *index*,  
    int *factor* )

**11.45.2.55 SetRegulatorOnOff()** void SetRegulatorOnOff (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned short *index*,  
    unsigned char *onoff* )

**11.45.2.56 SetRotatePump()** void SetRotatePump (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned short *index*,  
    short *speed* )

**11.45.2.57 SetSamplePeriode()** void SetSamplePeriode (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned short *periode* )

**11.45.2.58 SetSoilPressure()** void SetSoilPressure (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned short *index*,  
    int *pressure* )

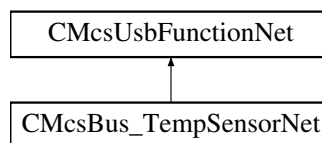
**11.45.2.59 StartSync()** void StartSync (   
    unsigned char *busnumber*,  
    unsigned char *busaddress* )

**11.45.2.60 TactSwitchGetState()** `int TactSwitchGetState (`  
     `unsigned char busnumber,`  
     `unsigned char busaddress )`

**11.45.2.61 TactSwitchSetDisplay()** `void TactSwitchSetDisplay (`  
     `unsigned char busnumber,`  
     `unsigned char busaddress,`  
     `int Melody )`

## 11.46 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

#### 11.46.1 Constructor & Destructor Documentation

**11.46.1.1 CMcsBus\_TempSensorNet()** `CMcsBus_TempSensorNet (`  
     `CMcsUsbNet^ device )`

**11.46.1.2 ~CMcsBus\_TempSensorNet()** `~CMcsBus_TempSensorNet (`  
     `void )`

## 11.46.2 Member Function Documentation

**11.46.2.1 GetNanoVoltsPerKelvin()** `int GetNanoVoltsPerKelvin ( unsigned char busnumber, unsigned char busaddress )`

**11.46.2.2 GetTemperatur()** `[1/2] short GetTemperatur ( unsigned char busnumber, unsigned char busaddress )`

**11.46.2.3 GetTemperatur()** `[2/2] short GetTemperatur ( unsigned char busnumber, unsigned char busaddress, short index )`

**11.46.2.4 GetThermoOffset()** `short GetThermoOffset ( unsigned char busnumber, unsigned char busaddress, short index )`

**11.46.2.5 GetThermoTemp()** `short GetThermoTemp ( unsigned char busnumber, unsigned char busaddress, short index )`

**11.46.2.6 GetThermoVoltage()** `short GetThermoVoltage ( unsigned char busnumber, unsigned char busaddress, short index )`

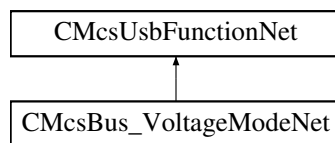
**11.46.2.7 SetNanoVoltsPerKelvin()** `void SetNanoVoltsPerKelvin ( unsigned char busnumber, unsigned char busaddress, int nanovoltsperkelvin )`



**11.46.2.8 SetThermoOffset()** void SetThermoOffset (  
     unsigned char *busnumber*,  
     unsigned char *busaddress*,  
     short *index*,  
     short *offset* )

## 11.47 CMcsBus\_VoltageModeNet Class Reference

Inheritance diagram for CMcsBus\_VoltageModeNet:



### Public Member Functions

- `CMcsBus_VoltageModeNet` (`CMcsUsbNet`^ device)
- `~CMcsBus_VoltageModeNet` (void)
- void `SetVMMaxPositiveCurrentEeprom` (unsigned char busnumber, unsigned char busaddress, unsigned char channel, short current)
- short `GetVMMaxPositiveCurrentEeprom` (unsigned char busnumber, unsigned char busaddress, unsigned char channel)
- void `SetVMMaxPositiveCurrent` (unsigned char busnumber, unsigned char busaddress, unsigned char channel, short current)
- short `GetVMMaxPositiveCurrent` (unsigned char busnumber, unsigned char busaddress, unsigned char channel)
- void `SetVMMaxNegativeCurrentEeprom` (unsigned char busnumber, unsigned char busaddress, unsigned char channel, short current)
- short `GetVMMaxNegativeCurrentEeprom` (unsigned char busnumber, unsigned char busaddress, unsigned char channel)
- void `SetVMMaxNegativeCurrent` (unsigned char busnumber, unsigned char busaddress, unsigned char channel, short current)
- short `GetVMMaxNegativeCurrent` (unsigned char busnumber, unsigned char busaddress, unsigned char channel)
- void `SetVMMaxPositiveVoltageEeprom` (unsigned char busnumber, unsigned char busaddress, unsigned char channel, short voltage)
- short `GetVMMaxPositiveVoltageEeprom` (unsigned char busnumber, unsigned char busaddress, unsigned char channel)
- void `SetVMMaxPositiveVoltage` (unsigned char busnumber, unsigned char busaddress, unsigned char channel, short voltage)
- short `GetVMMaxPositiveVoltage` (unsigned char busnumber, unsigned char busaddress, unsigned char channel)
- void `SetVMMaxNegativeVoltageEeprom` (unsigned char busnumber, unsigned char busaddress, unsigned char channel, short voltage)
- short `GetVMMaxNegativeVoltageEeprom` (unsigned char busnumber, unsigned char busaddress, unsigned char channel)
- void `SetVMMaxNegativeVoltage` (unsigned char busnumber, unsigned char busaddress, unsigned char channel, short voltage)
- short `GetVMMaxNegativeVoltage` (unsigned char busnumber, unsigned char busaddress, unsigned char channel)

- void [SetVMOutputOnOff](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel, unsigned short status)
- unsigned short [GetVMOutputOnOff](#) (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

### 11.47.1 Constructor & Destructor Documentation

**11.47.1.1 CMcsBus\_VoltageModeNet()** `CMcsBus_VoltageModeNet ( CMcsUsbNet^ device )`

**11.47.1.2 ~CMcsBus\_VoltageModeNet()** `~CMcsBus_VoltageModeNet ( void )`

### 11.47.2 Member Function Documentation

**11.47.2.1 GetVMMaxNegativeCurrent()** `short GetVMMaxNegativeCurrent ( unsigned char busnumber, unsigned char busaddress, unsigned char channel )`

**11.47.2.2 GetVMMaxNegativeCurrentEeprom()** `short GetVMMaxNegativeCurrentEeprom ( unsigned char busnumber, unsigned char busaddress, unsigned char channel )`

**11.47.2.3 GetVMMaxNegativeVoltage()** `short GetVMMaxNegativeVoltage ( unsigned char busnumber, unsigned char busaddress, unsigned char channel )`

**11.47.2.4 GetVMMaxNegativeVoltageEeprom()** short GetVMMaxNegativeVoltageEeprom (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *channel* )

**11.47.2.5 GetVMMaxPositiveCurrent()** short GetVMMaxPositiveCurrent (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *channel* )

**11.47.2.6 GetVMMaxPositiveCurrentEeprom()** short GetVMMaxPositiveCurrentEeprom (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *channel* )

**11.47.2.7 GetVMMaxPositiveVoltage()** short GetVMMaxPositiveVoltage (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *channel* )

**11.47.2.8 GetVMMaxPositiveVoltageEeprom()** short GetVMMaxPositiveVoltageEeprom (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *channel* )

**11.47.2.9 GetVMOutputOnOff()** unsigned short GetVMOutputOnOff (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *channel* )

**11.47.2.10 GetVMVoltage()** short GetVMVoltage (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *channel* )

- 11.47.2.11 SetVMMaxNegativeCurrent()** void SetVMMaxNegativeCurrent (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *channel*,  
    short *current* )
- 11.47.2.12 SetVMMaxNegativeCurrentEeprom()** void SetVMMaxNegativeCurrentEeprom (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *channel*,  
    short *current* )
- 11.47.2.13 SetVMMaxNegativeVoltage()** void SetVMMaxNegativeVoltage (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *channel*,  
    short *voltage* )
- 11.47.2.14 SetVMMaxNegativeVoltageEeprom()** void SetVMMaxNegativeVoltageEeprom (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *channel*,  
    short *voltage* )
- 11.47.2.15 SetVMMaxPositiveCurrent()** void SetVMMaxPositiveCurrent (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *channel*,  
    short *current* )
- 11.47.2.16 SetVMMaxPositiveCurrentEeprom()** void SetVMMaxPositiveCurrentEeprom (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *channel*,  
    short *current* )

**11.47.2.17 SetVMMaxPositiveVoltage()** `void SetVMMaxPositiveVoltage (`  
     `unsigned char busnumber,`  
     `unsigned char busaddress,`  
     `unsigned char channel,`  
     `short voltage )`

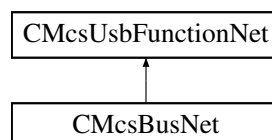
**11.47.2.18 SetVMMaxPositiveVoltageEeprom()** `void SetVMMaxPositiveVoltageEeprom (`  
     `unsigned char busnumber,`  
     `unsigned char busaddress,`  
     `unsigned char channel,`  
     `short voltage )`

**11.47.2.19 SetVMOutputOnOff()** `void SetVMOutputOnOff (`  
     `unsigned char busnumber,`  
     `unsigned char busaddress,`  
     `unsigned char channel,`  
     `unsigned short status )`

**11.47.2.20 SetVMVoltage()** `void SetVMVoltage (`  
     `unsigned char busnumber,`  
     `unsigned char busaddress,`  
     `unsigned char channel,`  
     `short voltage )`

## 11.48 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

### 11.48.1 Constructor & Destructor Documentation

**11.48.1.1 CMcsBusNet()** [CMcsBusNet](#) (   
 [CMcsUsbNet](#)<sup>^</sup> device )

**11.48.1.2 ~CMcsBusNet()** virtual [~CMcsBusNet](#) (   
 void ) [virtual]

### 11.48.2 Member Function Documentation

**11.48.2.1 CMcsBusNet::GetMode()** unsigned short [CMcsBusNet::GetMode](#) (   
 unsigned char *busnumber*,   
 unsigned char *busaddress* )

**11.48.2.2 CMcsBusNet::GetModeEeprom()** unsigned short CMcsBusNet::GetModeEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress* )

**11.48.2.3 CMcsBusNet::SetMode()** void CMcsBusNet::SetMode (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned short *mode* )

**11.48.2.4 CMcsBusNet::SetModeEeprom()** void CMcsBusNet::SetModeEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned short *mode* )

**11.48.2.5 GetBusAddress()** unsigned short GetBusAddress (   
 unsigned char *busnumber*,   
 unsigned char *busaddress* )

**11.48.2.6 GetBusAddressEeprom()** unsigned short GetBusAddressEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress* )

**11.48.2.7 GetCommand() [1/4]** void GetCommand (   
 unsigned char *command*,   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 [System::Runtime::InteropServices::Out] int% *value* )

**11.48.2.8 GetCommand() [2/4]** void GetCommand (   
 unsigned char *command*,   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 [System::Runtime::InteropServices::Out] short% *value* )

**11.48.2.9 GetCommand() [3/4]** void GetCommand (   
    unsigned char *command*,   
    unsigned char *busnumber*,   
    unsigned char *busaddress*,   
    unsigned char *axis*,   
    [System::Runtime::InteropServices::Out] unsigned int% *value* )

**11.48.2.10 GetCommand() [4/4]** void GetCommand (   
    unsigned char *command*,   
    unsigned char *busnumber*,   
    unsigned char *busaddress*,   
    unsigned char *axis*,   
    [System::Runtime::InteropServices::Out] unsigned short% *value* )

**11.48.2.11 GetHWRevisionEeprom()** unsigned short GetHWRevisionEeprom (   
    unsigned char *busnumber*,   
    unsigned char *busaddress* )

**11.48.2.12 SetBusAddress()** void SetBusAddress (   
    unsigned char *busnumber*,   
    unsigned char *busaddress*,   
    unsigned short *newaddress* )

**11.48.2.13 SetBusAddressEeprom()** void SetBusAddressEeprom (   
    unsigned char *busnumber*,   
    unsigned char *busaddress*,   
    unsigned short *newaddress* )

**11.48.2.14 SetCommand() [1/4]** void SetCommand (   
    unsigned char *command*,   
    unsigned char *busnumber*,   
    unsigned char *busaddress*,   
    unsigned char *axis*,   
    int *value* )

**11.48.2.15 SetCommand() [2/4]** void SetCommand (   
    unsigned char *command*,   
    unsigned char *busnumber*,   
    unsigned char *busaddress*,   
    unsigned char *axis*,   
    short *value* )



**11.48.2.16 SetCommand() [3/4]** void SetCommand (   
     unsigned char *command*,   
     unsigned char *busnumber*,   
     unsigned char *busaddress*,   
     unsigned char *axis*,   
     unsigned int *value* )

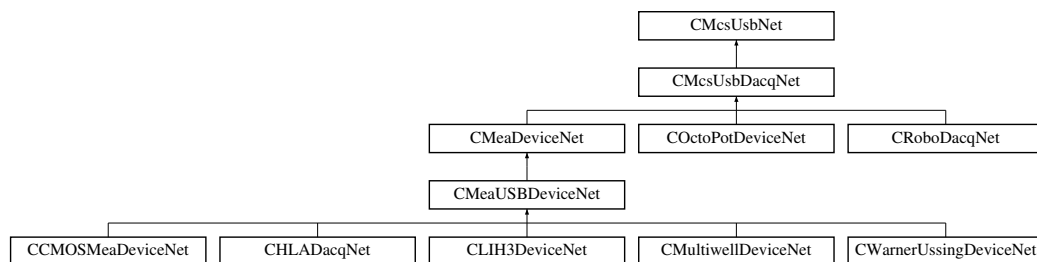
**11.48.2.17 SetCommand() [4/4]** void SetCommand (   
     unsigned char *command*,   
     unsigned char *busnumber*,   
     unsigned char *busaddress*,   
     unsigned char *axis*,   
     unsigned short *value* )

**11.48.2.18 SetHWRevisionEeprom()** void SetHWRevisionEeprom (   
     unsigned char *busnumber*,   
     unsigned char *busaddress*,   
     unsigned short *revision* )

## 11.49 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](#)< digitalsourceenum >^ source, int bitnumber\_offset)  
*Sets the function/source of an digital output bit.*
- 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.*
- 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::InteropServices::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::InteropServices::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 int32\_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 int32\_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)
- void [SetupGroupDacqQueue](#) (int queuesize, int threshold, unsigned int virtualDevice)
- [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

#### 11.49.1 Detailed Description

Base class for data acquisition devices.

#### 11.49.2 Constructor & Destructor Documentation

**11.49.2.1** [CMcsUsbDacqNet\(\)](#) [CMcsUsbDacqNet](#) ( )

**11.49.2.2** [~CMcsUsbDacqNet\(\)](#) [~CMcsUsbDacqNet](#) ( )

#### 11.49.3 Member Function Documentation

**11.49.3.1 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

<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>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.
-------------------	--



**Returns**

The handle to the Queue.

**11.49.3.2 AddSelectedChannelsQueue() [2/4]** virtual int AddSelectedChannelsQueue (   
     int *nByteOffset*,   
     int *nChannelOffset*,   
     array< bool >^ *selectedChannels*,   
     int *queuesize*,   
     int *threshold*,   
     SampleSizeNet *samplesize*,   
     SampleDstSizeNet *sampleDstSize* ) [virtual]

**11.49.3.3 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.

**11.49.3.4 AddSelectedChannelsQueue() [4/4]** virtual int AddSelectedChannelsQueue (  
     int *nByteOffset*,  
     int *nChannelOffset*,  
     int *nChannels*,  
     int *queuesize*,  
     int *threshold*,  
     SampleSizeNet *samplesize*,  
     SampleDstSizeNet *sampleDstSize* ) [virtual]

**11.49.3.5 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.

**11.49.3.6 ChannelBlock\_AvailFrames()** [2/2] virtual uint32\_t ChannelBlock\_AvailFrames (   
     int *handle*,  
     int *queue* ) [virtual]

**11.49.3.7 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.

**11.49.3.8 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.

#### 11.49.3.9 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.

### 11.49.3.10 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.

**11.49.3.11 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.

**11.49.3.12 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.

**11.49.3.13 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.

**11.49.3.14 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.

**11.49.3.15 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.

**11.49.3.16 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.

**11.49.3.17 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.

**11.49.3.18 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.

**11.49.3.19 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.

**11.49.3.20 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.
-------------------	--

**11.49.3.21 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.

**11.49.3.22 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.
-------------------	--

**11.49.3.23 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.
---------------	---

## 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.

**11.49.3.24 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.

**11.49.3.25 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.

**11.49.3.26 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.
-------------------	--

**11.49.3.27 ClearBuffers()** `virtual void ClearBuffers ( ) [virtual]`

**11.49.3.28 CMcsUsbDacqNet::GetFilterProperties()** `virtual array<CFilterPropertyNet^> ^ CMcsUsbDacqNet::GetFilterProperties ( DacqGroupChannelEnumNet GroupID ) [virtual]`

**11.49.3.29 GetAdapterType()** `virtual AdapterTypeEnumNet GetAdapterType ( ) [virtual]`

Gets the adapter which is connected to the MEA2100 device.

## Returns

AdapterTypeEnumNet which enumerates the possible adapters.

**11.49.3.30 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.



**11.49.3.31 GetAdcZero()** virtual uint32\_t GetAdcZero (   
 uint32\_t *virtualDevice*,   
 DacqGroupChannelEnumNet *group*,   
 [System::Runtime::InteropServices::Out] int% *adcZ* ) [virtual]

**11.49.3.32 GetAnalogValueUnit()** virtual uint32\_t GetAnalogValueUnit (   
 uint32\_t *virtualDevice*,   
 DacqGroupChannelEnumNet *group*,   
 [System::Runtime::InteropServices::Out] AnalogUnitEnumNet% *unit* ) [virtual]

**11.49.3.33 GetChannelDataFillSize()** int GetChannelDataFillSize ( )

**11.49.3.34 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]

**11.49.3.35 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.

**11.49.3.36 GetDataFormat()** virtual uint32\_t GetDataFormat (   
 uint32\_t *virtualDevice*,   
 DacqGroupChannelEnumNet *group*,   
 [System::Runtime::InteropServices::Out] int% *numberOfBits* ) [virtual]

**11.49.3.37 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.

**11.49.3.38 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.

**11.49.3.39 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.

**11.49.3.40 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.

**11.49.3.41 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.

**11.49.3.42 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.

**11.49.3.43 GetFilterProperty()** virtual `CFilterPropertyNet` ^ GetFilterProperty (   
     DacqGroupChannelEnumNet *GroupID*,   
     unsigned int *index* ) [virtual]

**11.49.3.44 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.

**11.49.3.45 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.

**11.49.3.46 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.

**11.49.3.47 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.

**11.49.3.48 GetHardwareMaxRange()** virtual uint32\_t GetHardwareMaxRange (   
uint32\_t virtualDevice,   
DacqGroupChannelEnumNet group,   
[System::Runtime::InteropServices::Out] int% r,   
[System::Runtime::InteropServices::Out] int% rUnit ) [virtual]

**11.49.3.49 GetHardwareMinRange()** virtual uint32\_t GetHardwareMinRange (   
uint32\_t virtualDevice,   
DacqGroupChannelEnumNet group,   
[System::Runtime::InteropServices::Out] int% r,   
[System::Runtime::InteropServices::Out] int% rUnit ) [virtual]

**11.49.3.50 GetMaxSamplingFrequency()** virtual uint32\_t GetMaxSamplingFrequency (   
int virtualDevice ) [virtual]

Gets the maximal sampling frequency of the device.

#### Returns

Sampling frequency in Hz.

**11.49.3.51 GetMeaLayout()** virtual MeaLayoutEnumNet GetMeaLayout ( ) [virtual]

Gets the MEA layout which is connected to the MEA2100 device.

Returns

MeaLayoutEnumNet which enumerates the MEA types.

**11.49.3.52 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.

**11.49.3.53 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

**11.49.3.54 GetPoti()** virtual uint32\_t GetPoti ( uint32\_t *channel*, [System::Runtime::InteropServices::Out] uint32\_t% *value* ) [virtual]

**11.49.3.55 GetResolutionPerDigit()** virtual uint32\_t GetResolutionPerDigit ( uint32\_t *virtualDevice*, DacqGroupChannelEnumNet *group*, [System::Runtime::InteropServices::Out] int% *res*, [System::Runtime::InteropServices::Out] int% *resUnit* ) [virtual]

**11.49.3.56 GetSamplerate()** virtual int32\_t GetSamplerate ( unsigned int *virtualDevice* ) [virtual]

Gets the sampling frequency of the device.

Returns

Sampling frequency in Hz.

**11.49.3.57 GetVoltageRangeIndex()** `virtual uint32_t GetVoltageRangeIndex ( unsigned int virtualDevice ) [virtual]`

**11.49.3.58 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.

**11.49.3.59 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.

**11.49.3.60 HWInfo()** `CHWInfo ^ HWInfo ( )`

**11.49.3.61 SendStartDacq()** [1/2] `virtual void SendStartDacq ( ) [virtual]`

Start sampling.

**11.49.3.62 SendStartDacq()** [2/2] `virtual void SendStartDacq ( int VirtualDacqMap ) [virtual]`

Start sampling.

#### Parameters

<i>VirtualDacqMap</i>	
-----------------------	--



**11.49.3.63 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>	

**11.49.3.64 SendStopDacq()** [1/2] virtual void SendStopDacq ( ) [virtual]

Stop sampling.

**11.49.3.65 SendStopDacq()** [2/2] virtual void SendStopDacq (   
     int *VirtualDacqMap* ) [virtual]

Stop sampling.

Parameters

<i>VirtualDacqMap</i>	
-----------------------	--

**11.49.3.66 SendStopStgAndDacq()** virtual void SendStopStgAndDacq (   
     uint32\_t *trigger\_map*,   
     int *VirtualDacqMap* ) [virtual]

Stop sampling together with the STG.

Parameters

<i>trigger_map</i>	
--------------------	--

**11.49.3.67 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>	
-----------------------	--

**11.49.3.68 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.

**11.49.3.69 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.

**11.49.3.70 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.

**11.49.3.71 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.

**11.49.3.72 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.

**11.49.3.73 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.

**11.49.3.74 SetPoti()** `virtual uint32_t SetPoti (`  
`uint32_t channel,`  
`uint32_t value,`  
`bool write_nvram ) [virtual]`

**11.49.3.75 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.
-------------	---------------------------

**11.49.3.76 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.
------------------------	---

**11.49.3.77 SetSelectedChannels() [2/4]** virtual void SetSelectedChannels (   
     array< bool >^ *selectedChannels*,   
     int *queuesize*,   
     int *threshold*,   
     SampleSizeNet *samplesize*,   
     SampleDstSizeNet *sampleDstSize*,   
     int *ChannelsInBlock* ) [virtual]

**11.49.3.78 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.

**11.49.3.79 SetSelectedChannels()** [4/4] virtual void SetSelectedChannels (

```

    int nChannels,
    int queuesize,
    int threshold,
    SampleSizeNet samplesize,
    SampleDstSizeNet sampleDstSize,
    int ChannelsInBlock ) [virtual]

```

**11.49.3.80 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.
------------------------	---

**11.49.3.81 SetSelectedChannelsQueue() [2/4]** virtual void SetSelectedChannelsQueue (   
     array< bool >^ *selectedChannels*,  
     int *queuesize*,  
     int *threshold*,  
     SampleSizeNet *samplesize*,  
     SampleDstSizeNet *sampleDstSize*,  
     int *ChannelsInBlock* ) [virtual]

**11.49.3.82 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.
------------------------	---

**11.49.3.83 SetSelectedChannelsQueue() [4/4]** virtual void SetSelectedChannelsQueue (   
     int *nChannels*,  
     int *queuesize*,  
     int *threshold*,  
     SampleSizeNet *samplesize*,  
     SampleDstSizeNet *sampleDstSize*,  
     int *ChannelsInBlock* ) [virtual]

**11.49.3.84 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.

**11.49.3.85 SetSelectedData() [2/4]** virtual void SetSelectedData (   
     array< bool >^ *selectedChannels*,   
     int *queuesize*,   
     int *threshold*,   
     SampleSizeNet *samplesize*,   
     SampleDstSizeNet *sampleDstSize*,   
     int *ChannelsInBlock* ) [virtual]

**11.49.3.86 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.

**11.49.3.87 SetSelectedData()** [4/4] virtual void SetSelectedData (   
     int *nChannels*,   
     int *queuesize*,   
     int *threshold*,   
     SampleSizeNet *samplesize*,   
     SampleDstSizeNet *sampleDstSize*,   
     int *ChannelsInBlock* ) [virtual]

**11.49.3.88 SetupGroupDacqQueue()** [1/2] void SetupGroupDacqQueue (   
     int *queuesize*,   
     int *threshold* )

**11.49.3.89 SetupGroupDacqQueue()** [2/2] void SetupGroupDacqQueue (   
     int *queuesize*,   
     int *threshold*,   
     unsigned int *virtualDevice* )

**11.49.3.90 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.
--------------------------	--

**11.49.3.91 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!

**11.49.3.92 StartDacq()** `[1/4] virtual void StartDacq ( ) [virtual]`

Start the data acquisition thread and sampling.

**11.49.3.93 StartDacq()** `[2/4] virtual void StartDacq (   
 int32_t timeout ) [virtual]`

Start the data acquisition thread and sampling.

**Parameters**

<i>timeout</i>	Timeout in ms.
----------------	----------------

**11.49.3.94 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.
---------------------	----------------------

**11.49.3.95 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.
----------------------	--------------------------

**11.49.3.96 StartLoop()** [1/4] `virtual void StartLoop ( ) [virtual]`

Start the data acquisition thread.

**11.49.3.97 StartLoop()** [2/4] `virtual void StartLoop (   
int32_t timeout ) [virtual]`

Start the data acquisition thread.

## Parameters

<i>timeout</i>	Timeout in ms.
----------------	----------------

**11.49.3.98 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.
---------------------	----------------------

**11.49.3.99 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.
----------------------	--------------------------

**11.49.3.100 StopDacq()** [1/2] `virtual void StopDacq ( ) [virtual]`

Stop the data acquisition thread and sampling.

**11.49.3.101 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.
----------------------	--------------------------

**11.49.3.102 StopLoop()** `virtual void StopLoop ( ) [virtual]`

## 11.49.4 Member Data Documentation

**11.49.4.1 Error\_Callback\_Aquisition\_Stopped** `const int Error_Callback_Aquisition_Stopped = 0x200 [static]`

**11.49.4.2 Error\_Callback\_Data\_lost** `const int Error_Callback_Data_lost = 5 [static]`

**11.49.4.3 Error\_Callback\_Frames\_Lost** `const int Error_Callback_Frames_Lost = 4 [static]`



**11.49.4.4 Error\_Callback\_Packet\_Error** `const int Error_Callback_Packet_Error = 1 [static]`

**11.49.4.5 Error\_Callback\_Queue\_Full** `const int Error_Callback_Queue_Full = 0x100 [static]`

**11.49.4.6 Error\_Callback\_RingQueue\_Full** `const int Error_Callback_RingQueue_Full = 3 [static]`

### 11.49.5 Property Documentation

**11.49.5.1 Samplerate** `virtual int Samplerate [get], [set]`

The sampling frequency of the device in Hz.

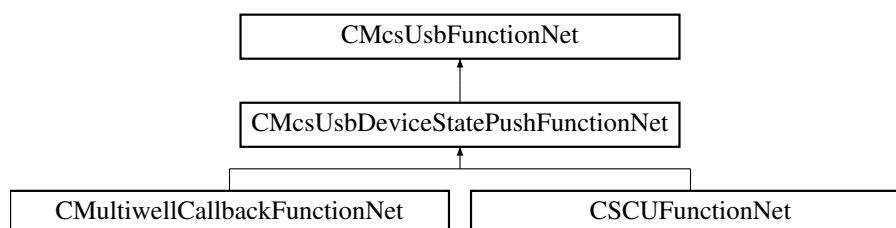
### 11.49.6 Event Documentation

**11.49.6.1 ChannelDataEvent** `OnChannelData^ ChannelDataEvent`

**11.49.6.2 ErrorEvent** `OnError^ ErrorEvent`

## 11.50 CMcsUsbDeviceStatePushFunctionNet Class Reference

Inheritance diagram for CMcsUsbDeviceStatePushFunctionNet:



### Public Member Functions

- void [TriggerStatus](#) ()

**Protected Member Functions**

- [CMcsUsbDeviceStatePushFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> pDevice)

**Events**

- [OnMcsUsbDeviceState](#)<sup>^</sup> [McsUsbDeviceStateEvent](#) [add, remove, raise]

**Additional Inherited Members****11.50.1 Constructor & Destructor Documentation**

**11.50.1.1 CMcsUsbDeviceStatePushFunctionNet()** [CMcsUsbDeviceStatePushFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> pDevice ) [protected]

**11.50.2 Member Function Documentation**

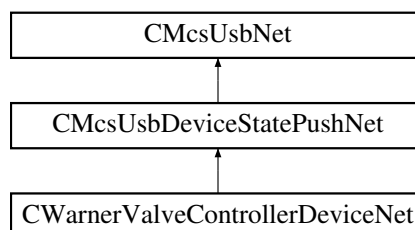
**11.50.2.1 TriggerStatus()** void TriggerStatus ( )

**11.50.3 Event Documentation**

**11.50.3.1 McsUsbDeviceStateEvent** [OnMcsUsbDeviceState](#)<sup>^</sup> McsUsbDeviceStateEvent [add], [remove], [raise]

**11.51 CMcsUsbDeviceStatePushNet Class Reference**

Inheritance diagram for CMcsUsbDeviceStatePushNet:



**Public Member Functions**

- void [TriggerStatus](#) ()

**Protected Member Functions**

- [CMcsUsbDeviceStatePushNet](#) ([CMcsUsbPointerContainer](#)<sup>^</sup> pDevice)

**Events**

- [OnMcsUsbDeviceState](#)<sup>^</sup> [McsUsbDeviceStateEvent](#) [add, remove, raise]

**Additional Inherited Members****11.51.1 Constructor & Destructor Documentation**

**11.51.1.1 CMcsUsbDeviceStatePushNet()** [CMcsUsbDeviceStatePushNet](#) ([CMcsUsbPointerContainer](#)<sup>^</sup> pDevice ) [protected]

**11.51.2 Member Function Documentation**

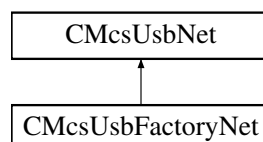
**11.51.2.1 TriggerStatus()** void [TriggerStatus](#) ( )

**11.51.3 Event Documentation**

**11.51.3.1 McsUsbDeviceStateEvent** [OnMcsUsbDeviceState](#)<sup>^</sup> [McsUsbDeviceStateEvent](#) [add], [remove], [raise]

**11.52 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
- static const uint32\_t [FX3MCSDDataIFB2ImageOffset](#) = 0xC
- static const uint32\_t [FX3MCSDDataIFB1ImageOffset](#) = 0x2C

**Additional Inherited Members****11.52.1 Constructor & Destructor Documentation**

**11.52.1.1 CMcsUsbFactoryNet()** [CMcsUsbFactoryNet](#) ( )

**11.52.1.2 ~CMcsUsbFactoryNet()** [~CMcsUsbFactoryNet](#) ( )

**11.52.2 Member Function Documentation**

**11.52.2.1 ChangeSerialNumber()** [uint32\\_t ChangeSerialNumber](#) (String^ serial )

**11.52.2.2 Coldstart()** uint32\_t Coldstart (   
 CFirmwareDestinationNet dest )

**11.52.2.3 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 )

**11.52.2.4 DownloadFirmware()** array<uint8\_t> ^ DownloadFirmware (   
 CFirmwareDestinationNet Dest,   
 uint32\_t Address,   
 uint32\_t length )

**11.52.2.5 FindFirmwareVersionMagicInBuffer()** static String ^ FindFirmwareVersionMagicInBuffer (   
 array< unsigned char >^ buffer,   
 int length,   
 [System::Runtime::InteropServices::Out] int% position ) [static]

**11.52.2.6 GetChecksumFromFX3Image()** static uint32\_t GetChecksumFromFX3Image (   
 String^ FirmwareFile ) [static]

**11.52.2.7 GetDestination()** [1/2] CFirmwareDestinationNet GetDestination (   
 String^ Key )

**11.52.2.8 GetDestination()** [2/2] CFirmwareDestinationNet GetDestination (   
 unsigned int index )

**11.52.2.9 GetDestinationDisplayLabel()** static String ^ GetDestinationDisplayLabel (   
 String^ RawLabel,   
 CFirmwareDestinationNet dest ) [static]

**11.52.2.10 GetDestinationName()** [1/2] `String ^ GetDestinationName ( CFirmwareDestinationNet dest )`

**11.52.2.11 GetDestinationName()** [2/2] `String ^ GetDestinationName ( unsigned int index )`

**11.52.2.12 GetDestinationSerialNumber()** `String ^ GetDestinationSerialNumber ( CFirmwareDestinationNet dest )`

**11.52.2.13 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.

**11.52.2.14 GetFirmwareVersionFromFile()** [1/2] `static bool GetFirmwareVersionFromFile ( String^ FirmwareFile, [System::Runtime::InteropServices::Out] uint32_t% Version ) [static]`

Retrives version info from a Firmware update file.

**11.52.2.15 GetFirmwareVersionFromFile()** [2/2] `static bool GetFirmwareVersionFromFile ( String^ FirmwareFile, [System::Runtime::InteropServices::Out] uint32_t% Version, [System::Runtime::InteropServices::Out] uint32_t% Position ) [static]`

**11.52.2.16 GetFirmwareVersionFromHexFile()** `static bool GetFirmwareVersionFromHexFile ( String^ FirmwareFile, [System::Runtime::InteropServices::Out] uint32_t% Version ) [static]`

**11.52.2.17 GetNumDestinations()** unsigned int GetNumDestinations ( )

**11.52.2.18 GetUSBDeviceIDFromFX3Image()** static uint32\_t GetUSBDeviceIDFromFX3Image ( String^ *FirmwareFile* ) [static]

**11.52.2.19 GetUsercodeFromBitFile()** static bool GetUsercodeFromBitFile ( String^ *FirmwareFile*, [System::Runtime::InteropServices::Out] unsigned int% *Usercode* ) [static]

**11.52.2.20 GetUsercodeFromFlash()** bool GetUsercodeFromFlash ( unsigned int *FPGA*, unsigned int *Address*, [System::Runtime::InteropServices::Out] unsigned int% *Usercode* )

**11.52.2.21 GetXilinxFlashOffset()** int32\_t GetXilinxFlashOffset ( CFirmwareDestinationNet *dest* )

**11.52.2.22 GetXilinxFlashReadCommand()** uint32\_t GetXilinxFlashReadCommand ( CFirmwareDestinationNet *dest* )

**11.52.2.23 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> .
------------------	--



**11.52.2.24 LoadUserFirmware()** [2/2] `bool LoadUserFirmware (`  
    `String^ FirmwareFile,`  
    `CMcsUsbListEntryNet^ listEntry,`  
    `uint32_t LockMask )`

**11.52.2.25 ReadBlockFromFlash()** [1/2] `array<unsigned char> ^ ReadBlockFromFlash (`  
    `unsigned int FPGA,`  
    `unsigned int Address )`

**11.52.2.26 ReadBlockFromFlash()** [2/2] `void ReadBlockFromFlash (`  
    `unsigned int FPGA,`  
    `unsigned int Address,`  
    `array< unsigned char >^ buffer,`  
    `int position )`

**11.52.2.27 ReadBlockFromIFBGlobalEEProm()** `array<unsigned char> ^ ReadBlockFromIFBGlobalEEProm (`  
    `unsigned int Address )`

**11.52.2.28 ReadBlockFromNVMEM()** `array<unsigned char> ^ ReadBlockFromNVMEM (`  
    `unsigned int FPGA,`  
    `unsigned int Offset,`  
    `unsigned int Address )`

**11.52.2.29 SetDestinationSerialNumber()** `void SetDestinationSerialNumber (`  
    `CFirmwareDestinationNet dest,`  
    `String^ serialnumber )`

**11.52.2.30 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.
------------------	-----------------------------------

**11.52.2.31 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.
------------------	-----------------------------------

**11.52.2.32 UpdateFirmware()** [3/5] `bool UpdateFirmware (`  
    `String^ FirmwareFile,`  
    `CMcsUsbListEntryNet^ listEntry,`  
    `CFirmwareDestinationNet dest,`  
    `bool SkipWait,`  
    `uint32_t LockMask )`

**11.52.2.33 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.
---------------------	--------------------------------

```

11.52.2.34 UpdateFirmware() [5/5] bool UpdateFirmware (
    String^ FirmwareFile,
    CMcsUsbListEntryNet^ listEntry,
    CFirmwareDestinationNet Dest,
    OnUpdateFirmwareStatusChange^ deleg,
    OnUpdateFirmwareProgress^ progress,
    bool SkipWait,
    unsigned int LockMask )

```

### 11.52.3 Member Data Documentation

```

11.52.3.1 FX3MCSDDataAddress const uint32_t FX3MCSDDataAddress = 0x40037E00 [static]

```

```

11.52.3.2 FX3MCSDDataDeviceIdOffset const uint32_t FX3MCSDDataDeviceIdOffset = 0x4 [static]

```

```

11.52.3.3 FX3MCSDDataIFB1ImageOffset const uint32_t FX3MCSDDataIFB1ImageOffset = 0x2C [static]

```

```

11.52.3.4 FX3MCSDDataIFB2ImageOffset const uint32_t FX3MCSDDataIFB2ImageOffset = 0xC [static]

```

```

11.52.3.5 FX3MCSDDataVersionOffset const uint32_t FX3MCSDDataVersionOffset = 0x8 [static]

```



**11.53.1.1 CMcsUsbFunctionNet()** [1/2] `CMcsUsbFunctionNet (`  
`CMcsUsbNet^ mcsusb )`

**11.53.1.2 ~CMcsUsbFunctionNet()** `virtual ~CMcsUsbFunctionNet (`  
`void ) [virtual]`

**11.53.1.3 ~!CMcsUsbFunctionNet()** `!CMcsUsbFunctionNet ( )`

**11.53.1.4 CMcsUsbFunctionNet()** [2/2] `CMcsUsbFunctionNet (`  
`CMcsUsbNet^ mcsusb,`  
`CMcsUsbFunctionPointerContainer^ mcsusbfunction ) [protected]`

## 11.53.2 Member Function Documentation

**11.53.2.1 ThrowCUsbExceptionNetOnError()** `void ThrowCUsbExceptionNetOnError (`  
`uint32_t status )`

## 11.53.3 Member Data Documentation

**11.53.3.1 m\_pMcsUsb** `CMcsUsbNet ^ m_pMcsUsb [protected]`

**11.53.3.2 m\_pMcsUsbFunction** `CMcsUsbFunction* m_pMcsUsbFunction [protected]`

## 11.54 CMcsUsbFunctionPointerContainer Class Reference

## 11.55 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]

### 11.55.1 Detailed Description

McsUsbListEntryNet identifies a connected device.

### 11.55.2 Constructor & Destructor Documentation

#### 11.55.2.1 ~CMcsUsbListEntryNet() ~CMcsUsbListEntryNet ( )

### 11.55.3 Member Function Documentation

**11.55.3.1 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.
------------	--

**11.55.3.2 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.

**11.55.3.3 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.

**11.55.3.4 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.

**11.55.3.5 GetEntryCount()** [1/2] `static unsigned int GetEntryCount ( ) [static]`

Returns the number of devices connected to the computer.

## Returns

The number of devices.

**11.55.3.6 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.

**11.55.3.7 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)".
---------------	--

**11.55.3.8 ToString()** `virtual String ^ ToString ( ) [override], [virtual]`

## 11.55.4 Property Documentation

**11.55.4.1 DeviceId** `DeviceIdNet^ DeviceId [get]`

**11.55.4.2 DeviceName** `String^ DeviceName [get]`

The device name of the device represented by this [CMcsUsbListEntryNet](#).

**11.55.4.3 HwVersion** `String^ HwVersion [get]`

The hardware revision of the device represented by this [CMcsUsbListEntryNet](#).

**11.55.4.4 Manufacturer** `String^ Manufacturer [get]`

The Manufacturer ID of the device represented by this [CMcsUsbListEntryNet](#).

**11.55.4.5 Product** `String^ Product [get]`

The Product ID of the device represented by this [CMcsUsbListEntryNet](#).

**11.55.4.6 SerialNumber** `String^ SerialNumber [get]`

The serial number of the device represented by this [CMcsUsbListEntryNet](#).

## 11.56 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](#)<sup>^</sup>> ^ 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](#)<sup>^</sup> [GetUsbListEntry](#) (unsigned int index)  
*Returns one [CMcsUsbListEntryNet](#) from the list of USB Devices connected to the computer.*
- array< [CMcsUsbListEntryNet](#)<sup>^</sup>> ^ [GetUsbListEntries](#) ()  
*Returns all entries from the list of USB Devices connected to the computer.*
- bool [IsDeviceTypeOf](#) ([CMcsUsbListEntryNet](#)<sup>^</sup> entry, DeviceEnumNet McsUsbDevice)

### Properties

- uint32\_t [Count](#) [get]  
*Gets the number of devices currently in the list.*

### Events

- [OnDeviceArrivalRemoval](#)<sup>^</sup> [DeviceArrival](#)
- [OnDeviceArrivalRemoval](#)<sup>^</sup> [DeviceRemoval](#)

#### 11.56.1 Detailed Description

Class to handle a list of connected MCS USB devices.

#### 11.56.2 Constructor & Destructor Documentation

##### 11.56.2.1 [CMcsUsbListNet](#)() [1/2] [CMcsUsbListNet](#) ( DeviceEnumNet McsUsbDevice )

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

**11.56.2.2 CMcsUsbListNet()** [2/2] `CMcsUsbListNet ( array< DeviceIdNet ^> ^ DeviceIdList )`

Initializes a new instance of `CMcsUsbListNet` class.

**11.56.2.3 ~CMcsUsbListNet()** `~CMcsUsbListNet ( )`

Destructor: called by `Dispose()`

**11.56.2.4 ~!CMcsUsbListNet()** `!CMcsUsbListNet ( )`

Finalizer: called by GC before collecting

### 11.56.3 Member Function Documentation

**11.56.3.1 GetNumberOfDevices()** `uint32_t GetNumberOfDevices ( )`

Gets the number of devices currently in the list.

#### Returns

The number of devices currently in the list.

**11.56.3.2 GetUsbListEntries()** `array<CMcsUsbListEntryNet ^> ^ GetUsbListEntries ( )`

Returns all entries from the list of USB Devices connected to the computer.

**11.56.3.3 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.
--------------	-----------------------------

**11.56.3.4 IsDeviceTypeOf()** `bool IsDeviceTypeOf (`  
    `CMcsUsbListEntryNet^ entry,`  
    `DeviceEnumNet McsUsbDevice )`

**11.56.3.5 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)".
---------------	--

## 11.56.4 Property Documentation

**11.56.4.1 Count** `uint32_t Count [get]`

Gets the number of devices currently in the list.

## 11.56.5 Event Documentation

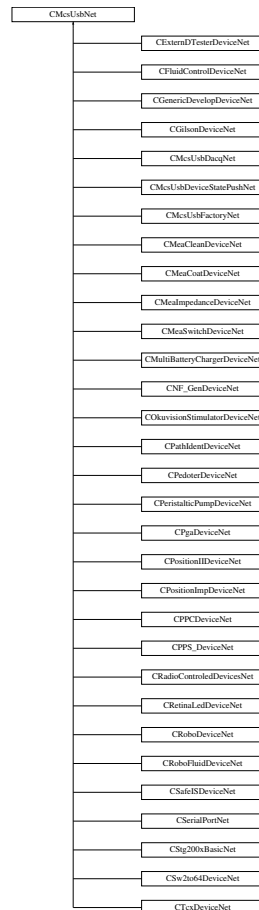
**11.56.5.1 DeviceArrival** `OnDeviceArrivalRemoval^ DeviceArrival`

**11.56.5.2 DeviceRemoval** `OnDeviceArrivalRemoval^ DeviceRemoval`

## 11.57 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)
- bool [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.*
- uint32\_t [CyclePort](#) ()
- 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\\_InvalidDeviceHandle](#) = (0xE0100013L)
- static const uint32\_t [Status\\_InvalidHandle](#) = (0xE0100012L)
- 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^ [SerialNumber](#) [get]

### 11.57.1 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.

### 11.57.2 Constructor & Destructor Documentation

#### 11.57.2.1 CMcsUsbNet() [1/2] `CMcsUsbNet ( )`

Initializes a new instance of the base class to handle MCS USB devices.

#### 11.57.2.2 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.
----------------	---

#### 11.57.2.3 ~CMcsUsbNet() `virtual ~CMcsUsbNet ( ) [virtual]`

#### 11.57.2.4 "!CMcsUsbNet() `!CMcsUsbNet ( )`

### 11.57.3 Member Function Documentation

#### 11.57.3.1 AddSoftwareKey() `void AddSoftwareKey ( String^ key )`

#### 11.57.3.2 AssociateToThis() `void AssociateToThis ( CMcsUsbNet^ device )`

**11.57.3.3 Connect()** [1/4] `virtual uint32_t Connect ( ) [virtual]`

Opens a connection to the device.

**Returns**

Error Status. 0 on success.

**11.57.3.4 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.

**11.57.3.5 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.

**11.57.3.6 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.

**11.57.3.7 CyclePort()** `uint32_t CyclePort ( )`

**11.57.3.8 Disconnect()** `virtual void Disconnect ( ) [virtual]`

Disconnect from a device.

**11.57.3.9 EmptyKey()** `bool EmptyKey ( String^ key )`

**11.57.3.10 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.
---------------	--

**11.57.3.11 EraseEepromRegisterPreconfig() [1/3]** `void EraseEepromRegisterPreconfig ( uint32_t EEPROMBase, uint32_t DMA_reg )`

**11.57.3.12 EraseEepromRegisterPreconfig() [2/3]** `void EraseEepromRegisterPreconfig ( uint32_t EEPROMBase, uint32_t DMA_reg, uint32_t EEPROMSize )`

**11.57.3.13 EraseEepromRegisterPreconfig()** [3/3] void EraseEepromRegisterPreconfig (   
    uint32\_t *EEPROMBase*,  
    uint32\_t *DMA\_reg*,  
    uint32\_t *EEPROMSize*,  
    uint32\_t *EepromStartAddress* )

**11.57.3.14 GetConfiguration()** uint8\_t GetConfiguration ( )

**11.57.3.15 GetDeviceCannotStallOutRequests()** bool GetDeviceCannotStallOutRequests ( )

**11.57.3.16 GetDeviceCapableSpeed()** McsUsbSpeedEnumNet GetDeviceCapableSpeed ( )

**11.57.3.17 GetDeviceEnum()** DeviceEnumNet GetDeviceEnum ( )

**11.57.3.18 GetDeviceId()** DeviceIdNet ^ GetDeviceId ( )

**11.57.3.19 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.

**11.57.3.20 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.

**11.57.3.21 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".

**11.57.3.22 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.

**11.57.3.23 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.

**11.57.3.24 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.

**11.57.3.25 GetHardwareRevision()** `String ^ GetHardwareRevision ( )`

**11.57.3.26 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

**11.57.3.27 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

**11.57.3.28 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

**11.57.3.29 GetIdent()** `uint32_t GetIdent ( [System::Runtime::InteropServices::Out] String^% Answer )`

**11.57.3.30 GetLastUSBError()** `unsigned int GetLastUSBError ( )`

**11.57.3.31 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

**11.57.3.32 GetNumConfigurations()** `uint8_t GetNumConfigurations ( )`

**11.57.3.33 GetSerialNumber()** `virtual String ^ GetSerialNumber ( ) [virtual]`

Query the Serial Number of the device.

**Returns**

The Serial Number.

**11.57.3.34 GetSoftwareKey()** `array<BYTE> ^ GetSoftwareKey ( unsigned int index )`

**11.57.3.35 GetSoftwareKeyString() [1/2]** `String ^ GetSoftwareKeyString ( SoftwareKeyProgramIdsNet::ProgramIdsNet ProgrammID, uint8_t majorversion )`

**11.57.3.36 GetSoftwareKeyString() [2/2]** `String ^ GetSoftwareKeyString ( uint8_t ProgrammID, uint8_t majorversion )`



**11.57.3.37 GetStatus()** virtual uint32\_t GetStatus (   
 [System::Runtime::InteropServices::Out] uint32\_t% iStatus ) [virtual]

**11.57.3.38 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.

**11.57.3.39 GetUsbListEntry()** CMcsUsbListEntryNet ^ GetUsbListEntry ( )

**11.57.3.40 GetVersion()** [1/2] DriverVersionNet ^ GetVersion ( )

**11.57.3.41 GetVersion()** [2/2] DriverVersionNet ^ GetVersion (   
 CFirmwareDestinationNet dest )

**11.57.3.42 HasSoftwareKey()** [1/2] bool HasSoftwareKey (   
 SoftwareKeyProgrammIdsNet::ProgrammIdsNet ProgrammID,   
 uint8\_t majorversion )

**11.57.3.43 HasSoftwareKey()** [2/2] bool HasSoftwareKey (   
 uint8\_t ProgrammID,   
 uint8\_t majorversion )

**11.57.3.44 IsConnected()** virtual bool IsConnected ( ) [virtual]

Check if a device is Connected.

#### Returns

true if the device is connected.

**11.57.3.45 IsDeviceHighSpeed()** `bool IsDeviceHighSpeed ( )`

**11.57.3.46 IsDeviceHighSpeedCapable()** `bool IsDeviceHighSpeedCapable ( )`

**11.57.3.47 IsExceptionsEnabled()** `bool IsExceptionsEnabled ( )`

**11.57.3.48 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.

**11.57.3.49 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.

**11.57.3.50 MultibootGetSelectedImage()** `uint32_t MultibootGetSelectedImage ( )`

Gets sector index of selected FPGA boot image on IFB

**Returns**

Sector index of image; range: 0..2

**11.57.3.51 MultibootSelectImage()** `void MultibootSelectImage ( unsigned int sector )`

Select the multiboot image specified by "sector" (range: 0..2) for IFB FPGA.

#### Returns

Throws exception on error.

**11.57.3.52 ReadEepromRegisterPreconfig()** [1/3] `bool ReadEepromRegisterPreconfig ( uint32_t EEPROMBase,  
uint32_t DMA_reg,  
[System::Runtime::InteropServices::Out] uint32_t% DMA_value )`

**11.57.3.53 ReadEepromRegisterPreconfig()** [2/3] `bool ReadEepromRegisterPreconfig ( uint32_t EEPROMBase,  
uint32_t DMA_reg,  
[System::Runtime::InteropServices::Out] uint32_t% DMA_value,  
uint32_t EEPROMSize )`

**11.57.3.54 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 )`

**11.57.3.55 ReadRegister()** [1/2] `unsigned int ReadRegister ( unsigned int reg )`

**11.57.3.56 ReadRegister()** [2/2] `array<uint32_t> ^ ReadRegister ( unsigned int reg,  
int length )`

**11.57.3.57 ReadRegister32()** `unsigned int ReadRegister32 ( unsigned int adr )`

**11.57.3.58 ReadRegisterTimeSlot()** unsigned int ReadRegisterTimeSlot (   
 unsigned int *reg*,   
 int *TimeSlot* )

**11.57.3.59 RemoveSoftwareKey()** void RemoveSoftwareKey (   
 unsigned int *index* )

**11.57.3.60 RescanHeadstage()** void RescanHeadstage (   
 uint32\_t *headstage* )

rescans and activates a headstage

#### Parameters

in	<i>headstage</i>	the headstage number (0 or 1)
----	------------------	-------------------------------

**11.57.3.61 SetConfiguration()** void SetConfiguration (   
 uint8\_t *config* )

**11.57.3.62 SetSoftwareKey()** void SetSoftwareKey (   
 unsigned int *index*,   
 array< BYTE >^ *buffer* )

**11.57.3.63 ThrowCUsbExceptionNetOnError()** void ThrowCUsbExceptionNetOnError (   
 uint32\_t *status* )

**11.57.3.64 TxnGetSerialNumber()** unsigned int TxnGetSerialNumber ( )

**11.57.3.65 TxnSetSerialNumber()** void TxnSetSerialNumber (   
 unsigned int *number* )

**11.57.3.66 TxnTestMemoryReadAndCheck()** unsigned int TxnTestMemoryReadAndCheck (   
 unsigned short *index* )

**11.57.3.67 TxnTestMemoryWrite()** unsigned int TxnTestMemoryWrite (   
 unsigned short *index* )

**11.57.3.68 ValidKey() [1/2]** bool ValidKey (   
 String^ *key*,   
 [System::Runtime::InteropServices::Out] String^% *serial\_number* )

**11.57.3.69 ValidKey() [2/2]** bool ValidKey (   
 String^ *key*,   
 uint8\_t *ProgrammID*,   
 uint8\_t *majorversion*,   
 [System::Runtime::InteropServices::Out] String^% *serial\_number* )

**11.57.3.70 WriteEepromRegisterPreconfig() [1/3]** void WriteEepromRegisterPreconfig (   
 uint32\_t *EEPROMBase*,   
 uint32\_t *DMA\_reg*,   
 uint32\_t *DMA\_value* )

**11.57.3.71 WriteEepromRegisterPreconfig() [2/3]** void WriteEepromRegisterPreconfig (   
 uint32\_t *EEPROMBase*,   
 uint32\_t *DMA\_reg*,   
 uint32\_t *DMA\_value*,   
 uint32\_t *EEPROMSize* )

**11.57.3.72 WriteEepromRegisterPreconfig() [3/3]** void WriteEepromRegisterPreconfig (   
 uint32\_t *EEPROMBase*,   
 uint32\_t *DMA\_reg*,   
 uint32\_t *DMA\_value*,   
 uint32\_t *EEPROMSize*,   
 uint32\_t *EepromStartAddress* )

**11.57.3.73 WriteRegister() [1/2]** void WriteRegister (   
 unsigned int *reg*,   
 array< unsigned int >^ *values* )

**11.57.3.74 WriteRegister() [2/2]** void WriteRegister (   
 unsigned int *reg*,   
 unsigned int *value* )

**11.57.3.75 WriteRegister32()** void WriteRegister32 (   
 unsigned int *adr*,   
 unsigned int *value* )

**11.57.3.76 WriteRegisterArray()** void WriteRegisterArray (   
 unsigned int *reg*,   
 array< unsigned int >^ *values* )

**11.57.3.77 WriteRegisterTimeSlot() [1/2]** void WriteRegisterTimeSlot (   
 unsigned int *reg*,   
 array< unsigned int >^ *values*,   
 int *TimeSlot* )

**11.57.3.78 WriteRegisterTimeSlot() [2/2]** void WriteRegisterTimeSlot (   
 unsigned int *reg*,   
 unsigned int *value*,   
 int *TimeSlot* )

**11.57.3.79 WriteRegisterValue()** void WriteRegisterValue (   
 unsigned int *reg*,   
 unsigned int *value* )

## 11.57.4 Member Data Documentation

**11.57.4.1 Status\_AlreadyConfigured** `const uint32_t Status_AlreadyConfigured = (0xE0110001L)`  
[static]

**11.57.4.2 Status\_BadStartFrame** `const uint32_t Status_BadStartFrame = (0xE0100A00L)` [static]

**11.57.4.3 Status\_Btstuff** `const uint32_t Status_Btstuff = (0xE0100002L)` [static]

**11.57.4.4 Status\_BufferOverrun** `const uint32_t Status_BufferOverrun = (0xE010000CL)` [static]

**11.57.4.5 Status\_BufferUnderrun** `const uint32_t Status_BufferUnderrun = (0xE010000DL)` [static]

**11.57.4.6 Status\_Canceled** `const uint32_t Status_Canceled = (0xE0110000L)` [static]

**11.57.4.7 Status\_Canceling** `const uint32_t Status_Canceling = (0xE0120000L)` [static]

**11.57.4.8 Status\_ConnectedPipes** `const uint32_t Status_ConnectedPipes = (0xE01F000AL)` [static]

**11.57.4.9 Status\_ControlNotOwned** `const uint32_t Status_ControlNotOwned = (0xE0100D00L)` [static]

**11.57.4.10 Status\_Crc** `const uint32_t Status_Crc = (0xE0100001L)` [static]

**11.57.4.11 Status\_DataOverrun** `const uint32_t Status_DataOverrun = (0xE0100008L)` [static]

**11.57.4.12 Status\_DataToggleMismatch** `const uint32_t Status_DataToggleMismatch = (0xE0100003L)`  
[static]

**11.57.4.13 Status\_DataUnderrun** `const uint32_t Status_DataUnderrun = (0xE0100009L)` [static]

**11.57.4.14 Status\_DeviceLocked** `const uint32_t Status_DeviceLocked = (0xE01F0010L)` [static]

**11.57.4.15 Status\_DeviceNotFound** `const uint32_t Status_DeviceNotFound = (0xE01F0003L)` [static]

**11.57.4.16 Status\_DeviceRemoved** `const uint32_t Status_DeviceRemoved = (0xE01F0008L)` [static]

**11.57.4.17 Status\_DevNotResponding** `const uint32_t Status_DevNotResponding = (0xE0100005L)`  
[static]

**11.57.4.18 Status\_EndpointHalted** `const uint32_t Status_EndpointHalted = (0xE0100030L)` [static]

**11.57.4.19 Status\_ErrorBusy** `const uint32_t Status_ErrorBusy = (0xE0100400L)` [static]

**11.57.4.20 Status\_ErrorShortTransfer** `const uint32_t Status_ErrorShortTransfer = (0xE0100900L)`  
[static]

**11.57.4.21 Status\_Fifo** `const uint32_t Status_Fifo = (0xE0100010L)` [static]

**11.57.4.22 Status\_FrameControlOwned** `const uint32_t Status_FrameControlOwned = (0xE0100C00L)`  
[static]



**11.57.4.23 Status\_InternalHcError** `const uint32_t Status_InternalHcError = (0xE0100800L) [static]`

**11.57.4.24 Status\_InvalidDeviceHandle** `const uint32_t Status_InvalidDeviceHandle = (0xE0100013L) [static]`

**11.57.4.25 Status\_InvalidHandle** `const uint32_t Status_InvalidHandle = (0xE0100012L) [static]`

**11.57.4.26 Status\_InvalidParameter** `const uint32_t Status_InvalidParameter = (0xE0100300L) [static]`

**11.57.4.27 Status\_InvalidPipeHandle** `const uint32_t Status_InvalidPipeHandle = (0xE0100600L) [static]`

**11.57.4.28 Status\_InvalidUrbFunction** `const uint32_t Status_InvalidUrbFunction = (0xE0100200L) [static]`

**11.57.4.29 Status\_IoPending** `const uint32_t Status_IoPending = (0xE01F0006L) [static]`

**11.57.4.30 Status\_IoTimeout** `const uint32_t Status_IoTimeout = (0xE01F0007L) [static]`

**11.57.4.31 Status\_IsochRequestFailed** `const uint32_t Status_IsochRequestFailed = (0xE0100B00L) [static]`

**11.57.4.32 Status\_LastUsbErrorMismatch** `const uint32_t Status_LastUsbErrorMismatch = (0xE01F0022L) [static]`

**11.57.4.33 Status\_NoBandwidth** `const uint32_t Status_NoBandwidth = (0xE0100700L) [static]`

**11.57.4.34 Status\_NoMemory** `const uint32_t Status_NoMemory = (0xE0100100L) [static]`

**11.57.4.35 Status\_NoSuchDevice** `const uint32_t Status_NoSuchDevice = (0xE01F0002L) [static]`

**11.57.4.36 Status\_NotAccessed** `const uint32_t Status_NotAccessed = (0xE010000FL) [static]`

**11.57.4.37 Status\_NotSupported** `const uint32_t Status_NotSupported = (0xE01F0005L) [static]`

**11.57.4.38 Status\_PidCheckFailure** `const uint32_t Status_PidCheckFailure = (0xE0100006L) [static]`

**11.57.4.39 Status\_PipeNotLinked** `const uint32_t Status_PipeNotLinked = (0xE01F0009L) [static]`

**11.57.4.40 Status\_RequestFailed** `const uint32_t Status_RequestFailed = (0xE0100500L) [static]`

**11.57.4.41 Status\_RequestMutexFailed** `const uint32_t Status_RequestMutexFailed = (0xE01F0021L) [static]`

**11.57.4.42 Status\_RequestMutexTimeout** `const uint32_t Status_RequestMutexTimeout = (0xE01F0020L) [static]`

**11.57.4.43 Status\_Stall** `const uint32_t Status_Stall = (0xE0100004L) [static]`

**11.57.4.44 Status\_Unconfigured** `const uint32_t Status_Unconfigured = (0xE0110002L) [static]`

**11.57.4.45 Status\_UnexpectedPid** `const uint32_t Status_UnexpectedPid = (0xE0100007L) [static]`

**11.57.4.46 WPAError\_ScanningIsPending** `const uint32_t WPAError_ScanningIsPending = ( (0x↵A0220000L) | 0x0036 ) [static]`

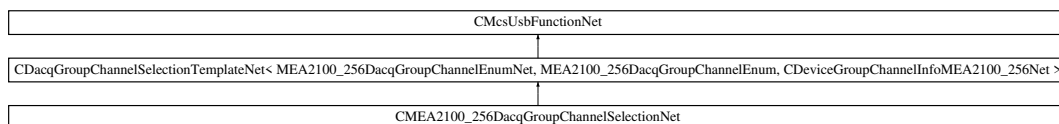
## 11.57.5 Property Documentation

**11.57.5.1 SerialNumber** `virtual String^ SerialNumber [get]`

## 11.58 CMcsUsbPointerContainer Class Reference

## 11.59 CMEA2100\_256DacqGroupChannelSelectionNet Class Reference

Inheritance diagram for CMEA2100\_256DacqGroupChannelSelectionNet:



### Public Member Functions

- [CMEA2100\\_256DacqGroupChannelSelectionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)

### Additional Inherited Members

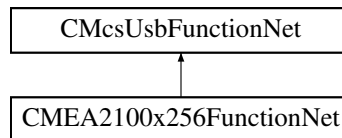
## 11.59.1 Constructor & Destructor Documentation

**11.59.1.1 CMEA2100\_256DacqGroupChannelSelectionNet()** [CMEA2100\\_256DacqGroupChannelSelectionNet](#) (  
     [CMcsUsbNet](#)<sup>^</sup> mcsusb )

## 11.60 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

#### 11.60.1 Detailed Description

[CMEA2100x256FunctionNet](#) is the class to control the MEA2100-256 device needs #include "Stg200xNet.h" to resolve documentation reference

#### 11.60.2 Constructor & Destructor Documentation

**11.60.2.1 CMEA2100x256FunctionNet()** [1/2] [CMEA2100x256FunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb,  
[CMcsUsbFunctionPointerContainer](#)<sup>^</sup> pMEA2100x256FunctionPointerContainer )

Initializes a new instance of the [CMEA2100x256FunctionNet](#) class.

**11.60.2.2 CMEA2100x256FunctionNet()** [2/2] [CMEA2100x256FunctionNet](#) ( [CMcsUsbNet](#)<sup>^</sup> *mcsusb* )

**11.60.2.3 ~CMEA2100x256FunctionNet()** [virtual ~CMEA2100x256FunctionNet](#) ( ) [virtual]

**11.60.2.4 "!CMEA2100x256FunctionNet()** [!CMEA2100x256FunctionNet](#) ( )

### 11.60.3 Member Function Documentation

**11.60.3.1 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.

**11.60.3.2 SetLayoutConfiguration()** [void](#) [SetLayoutConfiguration](#) ( [StimulationLayoutConfigurationEnumNet](#) *LayoutConfiguration* )

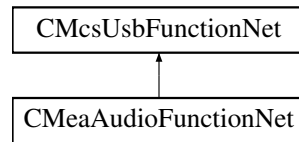
Sets the stimulation layout configuration. Can be single well, 6-well or 9-well. The number of D↔ AC channels available per well is [Mcs::Usb::CStg200xBasicNet::GetNumberOfAnalogChannels](#) divided by [Mcs::Usb::CStg200xBasicNet::GetNumberOfStimulationSourcesPerElectrode](#).

#### Parameters

<i>LayoutConfiguration</i>	The new stimulation layout configuration.
----------------------------	---

## 11.61 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

### 11.61.1 Constructor & Destructor Documentation

**11.61.1.1 CMeaAudioFunctionNet()** [1/2] [CMeaAudioFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb,  
[CMcsUsbFunctionPointerContainer](#)<sup>^</sup> meaAudioFunctionPointerContainer )

**11.61.1.2 CMeaAudioFunctionNet()** [2/2] [CMeaAudioFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb )

### 11.61.2 Member Function Documentation

**11.61.2.1 GetAudioChannels()** [1/2] virtual uint32\_t GetAudioChannels (  
[System::Runtime::InteropServices::Out] array< [s\\_setaudionet](#)<sup>^</sup>>% 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.

**11.61.2.2 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.

**11.61.2.3 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.

**11.61.2.4 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.

**11.61.2.5 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.
----------------------	------------------------

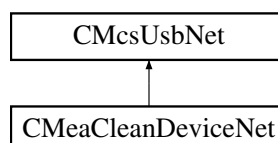
## Returns

Error Status. 0 on success.

## 11.62 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****11.62.1 Detailed Description**

[CMeaCleanDeviceNet](#) is the class to access the MEA Clean device.

**11.62.2 Constructor & Destructor Documentation****11.62.2.1 CMeaCleanDeviceNet() [CMeaCleanDeviceNet](#) ( )**

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

**11.62.2.2** `~CMeaCleanDeviceNet()` `virtual ~CMeaCleanDeviceNet ( ) [virtual]`

**11.62.2.3** `!CMeaCleanDeviceNet()` `!CMeaCleanDeviceNet ( )`

### 11.62.3 Member Function Documentation

**11.62.3.1** `GetCycle()` `int32_t GetCycle ( )`

Gets the current cycle.

#### Returns

The cycle number.

**11.62.3.2** `GetCycles()` `uint32_t GetCycles ( )`

Gets the number of cycles.

#### Returns

The number of cycles to run for.

**11.62.3.3** `GetMaxVoltage()` `int32_t GetMaxVoltage ( )`

Gets the upper voltage level

#### Returns

The upper voltage level in mV.

**11.62.3.4** `GetMinVoltage()` `int32_t GetMinVoltage ( )`

Gets the lower voltage level.

#### Returns

The lower voltage level in mV.

**11.62.3.5 GetOutputVoltage()** `int32_t GetOutputVoltage ( )`

Gets the output voltage.

**Returns**

The output voltage in mV.

**11.62.3.6 GetSlope()** `uint32_t GetSlope ( )`

Gets the voltage slope.

**Returns**

The voltage slope in mV/s.

**11.62.3.7 IsRunning()** `bool IsRunning ( )`

Gets if the MEA Clean device is running.

**Returns**

"true" when a run is in progress, otherwise "false".

**11.62.3.8 SetCycles()** `void SetCycles (   
uint32_t cycles )`

Sets the number of cycles.

**Parameters**

<i>cycles</i>	The number of cycles to run for (0 .. 99).
---------------	--

**11.62.3.9 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).
-------------------	--

**11.62.3.10 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).
-------------------	--

**11.62.3.11 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).
---------------------	--

**11.62.3.12 Start()** `void Start ( )`

Starts a MEA Clean run.

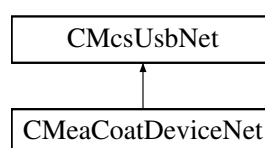
**11.62.3.13 Stop()** `void Stop ( )`

Stops a MEA Clean run.

## 11.63 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

### 11.63.1 Detailed Description

[CMeaCoatDeviceNet](#) is the class to access the MEA Coat device.

## 11.63.2 Constructor & Destructor Documentation

### 11.63.2.1 CMeaCoatDeviceNet() CMeaCoatDeviceNet ( )

Initializes a new instance of the CMeaCoatDeviceNet class.

### 11.63.2.2 ~CMeaCoatDeviceNet() virtual ~CMeaCoatDeviceNet ( ) [virtual]

### 11.63.2.3 "!CMeaCoatDeviceNet() !CMeaCoatDeviceNet ( )

## 11.63.3 Member Function Documentation

### 11.63.3.1 GetCurrentCycle() int32\_t GetCurrentCycle ( )

Gets the current cycle.

#### Returns

The cycle number.

### 11.63.3.2 GetCycles() uint32\_t GetCycles ( )

Gets the number of cycles.

#### Returns

The number of cycles to run for.

### 11.63.3.3 GetDuration() uint32\_t GetDuration ( )

Gets the duration of a MEA Coat run.

#### Returns

The duration in ms.

**11.63.3.4 GetMaxCurrent()** `uint32_t GetMaxCurrent ( )`

Gets the limit of the current ramp (absolute value).

**Returns**

The limit of the current ramp in pA (absolute value).

**11.63.3.5 GetOffsetCurrent()** `int32_t GetOffsetCurrent ( )`

Gets the offset of the current.

**Returns**

The offset of the current in pA.

**11.63.3.6 GetOutputCurrent()** `int32_t GetOutputCurrent ( )`

Gets the output current.

**Returns**

The output current in pA.

**11.63.3.7 GetPauseDuration()** `uint32_t GetPauseDuration ( )`

Gets the duration of the pause between MEA Coat pulses.

**Returns**

The duration in ms.

**11.63.3.8 GetSlope()** `int32_t GetSlope ( )`

Gets the current slope.

**Returns**

The current slope in pA/s.

**11.63.3.9 GetTimeInPause()** `int32_t GetTimeInPause ( )`

Gets the time in the pause.

**Returns**

The time in the pause in ms.

**11.63.3.10 GetTimeInPlateau()** `int32_t GetTimeInPlateau ( )`

Gets the time in the plateau.

**Returns**

The time in the plateau in ms.

**11.63.3.11 IsRunning()** `bool IsRunning ( )`

Gets if the MEA Clean device is running.

**Returns**

"true" when a run is in progress, otherwise "false".

**11.63.3.12 SetCycles()** `void SetCycles (   
uint32_t cycles )`

Sets the number of cycles.

**Parameters**

<i>cycles</i>	The number of cycles to run for (0 .. 99).
---------------	--

**11.63.3.13 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).
-----------------	---------------------------------------



**11.63.3.14 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).
-------------------	---

**11.63.3.15 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).
----------------------	---

**11.63.3.16 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).
----------------------	---------------------------------------

**11.63.3.17 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).
---------------------	---

**11.63.3.18 Start()** `void Start ( )`

Starts a MEA Coat run.

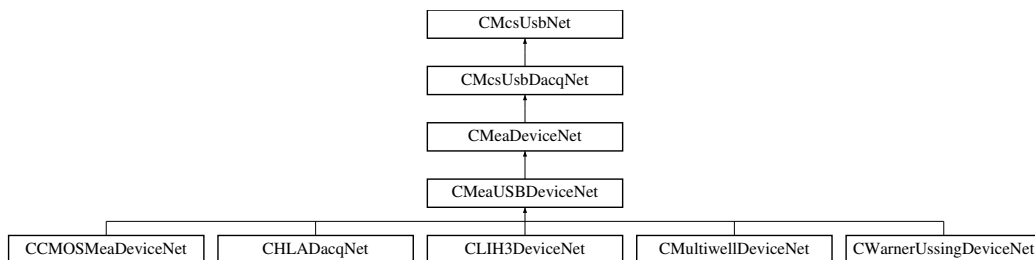
### 11.63.3.19 Stop() void Stop ( )

Stops a MEA Coat run.

## 11.64 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

### 11.64.1 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

### 11.64.2 Constructor & Destructor Documentation

#### 11.64.2.1 [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.
----------------	---

**11.64.2.2 CMeaDeviceNet()** [2/2] [CMeaDeviceNet](#) (   
 `McsBusTypeEnumNet bustype,`  
 `OnChannelData^ channelData,`  
 `OnError^ 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.
--------------	--

**11.64.2.3 ~CMeaDeviceNet()** [~CMeaDeviceNet](#) ( )

### 11.64.3 Member Function Documentation

**11.64.3.1 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.

**11.64.3.2 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.

**11.64.3.3 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.

**11.64.3.4 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.

**11.64.3.5 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.

**11.64.3.6 GetEnumerationSpeed()** `int32_t GetEnumerationSpeed ( )`**11.64.3.7 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.

**11.64.3.8 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.
--------------------	--------------------

**11.64.3.9 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.

**11.64.3.10 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.

**11.64.3.11 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.

**11.64.3.12 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.

**11.64.3.13 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.

**11.64.4 Property Documentation**

**11.64.4.1 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.

**11.64.4.2 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.

**11.64.4.3 MeaAudioFunctionNet** `CMeaAudioFunctionNet^ MeaAudioFunctionNet [get]`

**11.64.4.4 MeaDigitalDataFunctionNet** `CMeaDigitalDataFunctionNet^ MeaDigitalDataFunctionNet [get]`

**11.64.4.5 MeaFeedbackFunctionNet** [CMeaFeedbackFunctionNet](#)<sup>^</sup> MeaFeedbackFunctionNet [get]

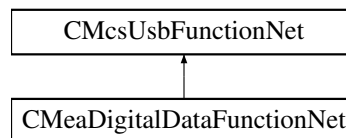
**11.64.4.6 MeFunctionNet** [CMeFunctionNet](#)<sup>^</sup> MeFunctionNet [get]

**11.64.4.7 W2100\_FunctionNet** [CW2100\\_FunctionNet](#)<sup>^</sup> W2100\_FunctionNet [get]

**11.64.4.8 WClassicFunctionNet** [CWClassicFunctionNet](#)<sup>^</sup> WClassicFunctionNet [get]

## 11.65 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

#### 11.65.1 Constructor & Destructor Documentation

**11.65.1.1 CMeaDigitalDataFunctionNet()** [1/2] [CMeaDigitalDataFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb,  
[CMcsUsbFunctionPointerContainer](#)<sup>^</sup> meaDigitalFunctionPointerContainer )

**11.65.1.2 CMeaDigitalDataFunctionNet()** [2/2] `CMeaDigitalDataFunctionNet ( CMcsUsbNet^ mcsusb )`

## 11.65.2 Member Function Documentation

**11.65.2.1 GetDigitalData()** `unsigned int GetDigitalData ( )`

Get the value of the digital output.

### Returns

Value on the digital data register.

**11.65.2.2 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.
--------------	------------

**11.65.2.3 SetDigitalData()** [2/2] `void SetDigitalData ( unsigned int digital_value, unsigned int digital_value_mask )`

Generate a value on the digital output.

### Parameters

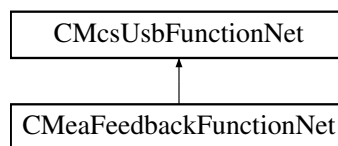
<i>digital_value</i>	Value to set.
----------------------	---------------

## Parameters

<i>digital_value_mask</i>	Mask for change.
---------------------------	------------------

## 11.66 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

### 11.66.1 Constructor & Destructor Documentation

**11.66.1.1 CMeaFeedbackFunctionNet()** [1/2] CMeaFeedbackFunctionNet (   
 CMcsUsbNet^ *mcsusb*,   
 CMcsUsbFunctionPointerContainer^ *meaFeedbackFunctionNet* )

**11.66.1.2 CMeaFeedbackFunctionNet()** [2/2] CMeaFeedbackFunctionNet (   
 CMcsUsbNet^ *mcsusb* )

### 11.66.2 Member Function Documentation

**11.66.2.1 FeedbackGetSampleTimerCount()** unsigned int FeedbackGetSampleTimerCount (   
 [System::Runtime::InteropServices::Out] unsigned int% *CurrentCount*,   
 [System::Runtime::InteropServices::Out] unsigned int% *LastKnownCount*,   
 [System::Runtime::InteropServices::Out] bool% *On* )

**11.66.2.2 FeedbackSetAnalogSource()** void FeedbackSetAnalogSource (   
 AnalogSourceEnumNet *AnalogSource*,   
 unsigned int *Channels*,   
 unsigned int *Offset* )

**11.66.2.3 FeedbackSetChannelFilter()** void FeedbackSetChannelFilter (   
 short *channel*,   
 char *filter* )

**11.66.2.4 FeedbackSetDigitalMapping()** void FeedbackSetDigitalMapping (   
 unsigned short *channel*,   
 unsigned short *outmapping*,   
 unsigned short *inmapping* )

**11.66.2.5 FeedbackSetFeedback()** void FeedbackSetFeedback (   
 unsigned char *on*,   
 unsigned short *digoutmask*,   
 unsigned short *diginmask* )

**11.66.2.6 FeedbackSetFilterOff()** void FeedbackSetFilterOff ( )

**11.66.2.7 FeedbackSetFilterParameter()** void FeedbackSetFilterParameter (   
 unsigned char *filter*,   
 array< short >^ *parameters* )

**11.66.2.8 FeedbackSetFilterParameter32()** void FeedbackSetFilterParameter32 (   
 unsigned char *filter*,   
 array< int >^ *parameters* )

**11.66.2.9 FeedbackSetGlobalChannelFilter()** void FeedbackSetGlobalChannelFilter (   
 char *filter*,   
 unsigned short *firstchannel*,   
 unsigned short *lastchannel* )

**11.66.2.10 FeedbackSetIIRFilterParameter()** void FeedbackSetIIRFilterParameter (   
 unsigned char *filter*,   
 int *length*,   
 array< double >^ *parameters* )

**11.66.2.11 FeedbackSetLogic()** void FeedbackSetLogic (   
 unsigned short *position*,   
 array< unsigned short >^ *sourcechannel*,   
 unsigned short *resultchannel*,   
 unsigned int *lookup* )

**11.66.2.12 FeedbackSetMkFilter()** void FeedbackSetMkFilter (   
 unsigned char *filter*,   
 String^ *filtertype*,   
 double *cheb\_ribble*,   
 String^ *passtype*,   
 int *order*,   
 double *alpha1*,   
 double *alpha2* )

**11.66.2.13 FeedbackSetNumberOfLogics()** void FeedbackSetNumberOfLogics (   
 unsigned short *number* )

**11.66.2.14 FeedbackSetNumberOfRateCounter()** void FeedbackSetNumberOfRateCounter (   
 unsigned short *number* )

**11.66.2.15 FeedbackSetNumberOfRateDetectors()** void FeedbackSetNumberOfRateDetectors (   
 unsigned short *number* )

**11.66.2.16 FeedbackSetNumberOfSpikeDetectors()** void FeedbackSetNumberOfSpikeDetectors (   
 unsigned short *number* )

**11.66.2.17 FeedbackSetNumberOfTriggers()** void FeedbackSetNumberOfTriggers (   
 unsigned short *number* )

**11.66.2.18 FeedbackSetRateCounter()** void FeedbackSetRateCounter (   
 unsigned short *position*,   
 unsigned short *sourcechannel*,   
 unsigned short *resultchannel* )

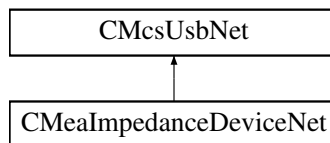
**11.66.2.19 FeedbackSetRateDetector()** void FeedbackSetRateDetector (   
 unsigned short *position*,   
 unsigned short *resultchannel*,   
 unsigned short *trigger*,   
 unsigned short *totzeit*,   
 unsigned short *pulses*,   
 unsigned int *duration1*,   
 unsigned int *duration2* )

**11.66.2.20 FeedbackSetSpikeDetectorThreshold()** `void FeedbackSetSpikeDetectorThreshold (`  
     `unsigned short position,`  
     `unsigned short sourcechannel,`  
     `unsigned short resultchannel,`  
     `unsigned short trigger,`  
     `unsigned short totzeit,`  
     `int threshold1,`  
     `int threshold2,`  
     `short slope )`

**11.66.2.21 FeedbackSetTrigger()** `void FeedbackSetTrigger (`  
     `unsigned short position,`  
     `unsigned short sourcechannel,`  
     `unsigned short resultchannel,`  
     `unsigned short trigger,`  
     `unsigned short totzeit )`

## 11.67 CMealImpedanceDeviceNet Class Reference

Inheritance diagram for CMealImpedanceDeviceNet:



### Public Member Functions

- [CMealImpedanceDeviceNet](#) ()
- [~CMealImpedanceDeviceNet](#) ()
- virtual void [StartMeasurement](#) (unsigned short channel)
- virtual unsigned short [GetReady](#) ()
- virtual unsigned short [GetArraySize](#) ()
- virtual array< unsigned short > ^ [GetResult](#) ()
- unsigned short [GetAdapterCode](#) ()
- virtual unsigned int [GetImpedanceTestFrequency](#) ()
- virtual void [SetImpedanceTestFrequency](#) (unsigned int TestFrequency\_Hertz)

### Additional Inherited Members

#### 11.67.1 Constructor & Destructor Documentation

**11.67.1.1 CMealImpedanceDeviceNet()** `CMealImpedanceDeviceNet ( )`



11.67.1.2 `~CMeaImpedanceDeviceNet()` `~CMeaImpedanceDeviceNet ( )`

## 11.67.2 Member Function Documentation

11.67.2.1 `GetAdapterCode()` `unsigned short GetAdapterCode ( )`

11.67.2.2 `GetArraySize()` `virtual unsigned short GetArraySize ( ) [virtual]`

11.67.2.3 `GetImpedanceTestFrequency()` `virtual unsigned int GetImpedanceTestFrequency ( ) [virtual]`

11.67.2.4 `GetReady()` `virtual unsigned short GetReady ( ) [virtual]`

11.67.2.5 `GetResult()` `virtual array<unsigned short> ^ GetResult ( ) [virtual]`

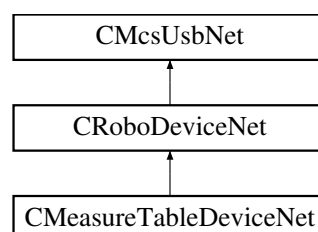
11.67.2.6 `SetImpedanceTestFrequency()` `virtual void SetImpedanceTestFrequency ( unsigned int TestFrequency_Hertz ) [virtual]`

11.67.2.7 `StartMeasurement()` `virtual void StartMeasurement ( unsigned short channel ) [virtual]`

## 11.68 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

#### 11.68.1 Detailed Description

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

#### 11.68.2 Constructor & Destructor Documentation

**11.68.2.1 CMeasureTableDeviceNet()** [CMeasureTableDeviceNet](#) (  
void )

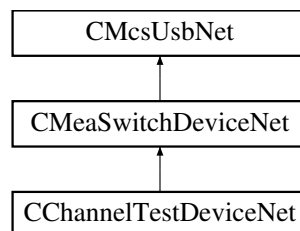
#### 11.68.3 Property Documentation

**11.68.3.1 Sensor** [CMcsBus\\_SensorNet^ Sensor](#) [get]

## 11.69 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****11.69.1 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.

**11.69.2 Constructor & Destructor Documentation****11.69.2.1 CMeaSwitchDeviceNet()** [CMeaSwitchDeviceNet](#) ( )

Constructor.

**11.69.2.2 ~CMeaSwitchDeviceNet()** [~CMeaSwitchDeviceNet](#) ( )

Destructor.

**11.69.3 Member Function Documentation****11.69.3.1 GetNumber()** unsigned short [GetNumber](#) ( )

Gets the number of boards in the device.

The MEA-Switch are delivered with 64 or 128 channels

**11.69.3.2 GetPattern()** `array<unsigned char> ^ GetPattern ( )`

Gets the pattern of the switches that are currently set in the device as char array.

**11.69.3.3 GetPatternBool()** `array<bool> ^ GetPatternBool ( )`

Gets the pattern of the switches that are currently set in the device as bools.

**11.69.3.4 SetPattern()** `void SetPattern ( array< unsigned char >^ pattern )`

Sets the pattern of switches from a char array.

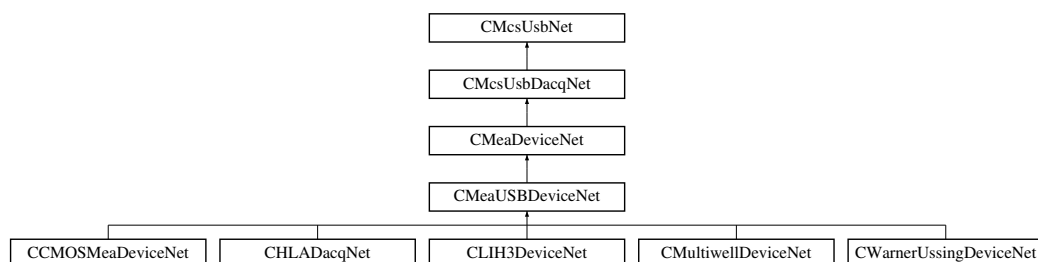
**11.69.3.5 SetPatternBool()** `void SetPatternBool ( array< bool >^ pattern )`

Sets the pattern of switches from a.

**11.70 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

### 11.70.1 Detailed Description

Class for data acquisition via ME and MEA USB amplifiers

### 11.70.2 Constructor & Destructor Documentation

**11.70.2.1 CMeaUSBDeviceNet()** [1/2] `CMeaUSBDeviceNet ( OnChannelData^ channelData, OnError^ error )`

Initializes a new instance of `CMeaDeviceNet` class.

#### Parameters

<i>channelData</i>	Handler to call when new data is available.
--------------------	---

#### Parameters

<i>error</i>	Handler to call when an error occurs.
--------------	---------------------------------------

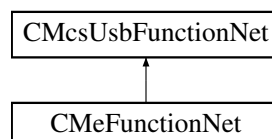
**11.70.2.2 CMeaUSBDeviceNet()** [2/2] `CMeaUSBDeviceNet ( )`

Initializes a new instance of `CMeaDeviceNet` class.

**11.70.2.3 ~CMeaUSBDeviceNet()** `~CMeaUSBDeviceNet ( )`

## 11.71 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

### 11.71.1 Detailed Description

### 11.71.2 Constructor & Destructor Documentation

**11.71.2.1 CMeFunctionNet()** [1/2] [CMeFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb,  
[CMcsUsbFunctionPointerContainer](#)<sup>^</sup> meFunctionPointerContainer )

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

**11.71.2.2 CMeFunctionNet()** [2/2] [CMeFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb )

**11.71.2.3 ~CMeFunctionNet()** virtual [~CMeFunctionNet](#) (  
void ) [virtual]

**11.71.2.4 !CMeFunctionNet()** [!CMeFunctionNet](#) (  
void )

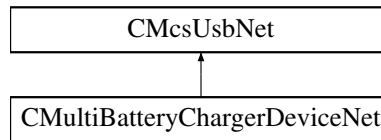
### 11.71.3 Member Function Documentation

**11.71.3.1 SetTransformer()** void [SetTransformer](#) (  
unsigned int index,  
bool onoff )

## 11.72 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](#) ()
- [ICMultiBatteryChargerDeviceNet](#) ()
- [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

### 11.72.1 Detailed Description

[CMultiBatteryChargerDeviceNet](#) is the class to access the MBC-08 device.

### 11.72.2 Constructor & Destructor Documentation

#### 11.72.2.1 CMultiBatteryChargerDeviceNet() [CMultiBatteryChargerDeviceNet](#) ( )

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

#### 11.72.2.2 ~CMultiBatteryChargerDeviceNet() [virtual ~CMultiBatteryChargerDeviceNet](#) ( ) [virtual]

#### 11.72.2.3 "!CMultiBatteryChargerDeviceNet() [!CMultiBatteryChargerDeviceNet](#) ( )

### 11.72.3 Member Function Documentation

#### 11.72.3.1 CapacityTest() [void CapacityTest](#) ( uint32\_t NrChannel )

start capacity test on channel

#### Parameters

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



**11.72.3.2 ChannelReset()** `void ChannelReset (`  
    `uint32_t NrChannel )`

cancel charging and capacity test functions; check if battery is connected

**Parameters**

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

**11.72.3.3 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

**11.72.3.4 GetChannels()** `uint32_t GetChannels ( )`

gets number of channels

**Returns**

number of channels

**11.72.3.5 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

**11.72.3.6 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

**11.72.3.7 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

**11.72.3.8 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

**11.72.3.9 GetChargingPCoefficient()** `uint32_t GetChargingPCoefficient ( )`

gets the p-coefficient for charging in mA/V / nominal charging current

**Returns**

the p-coefficient

**11.72.3.10 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

**11.72.3.11 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

**11.72.3.12 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

**11.72.3.13 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

**11.72.3.14 GetRatedCapacity()** `MbcRatedCapacityEnumNet GetRatedCapacity (`  
`uint32_t NrChannel )`

gets the rated capacity

**Parameters**

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

**Returns**

the capacity

**11.72.3.15 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

**11.72.3.16 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
---------------------	-------------------

**11.72.3.17 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

**11.72.3.18 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

**11.72.3.19 SetRatedCapacity()** void SetRatedCapacity (   
 uint32\_t *NrChannel*,   
 MbcRatedCapacityEnumNet *NewRatedCapacity* )

sets the rated capacity

Parameters

<i>NrChannel</i>	the channel number
<i>NewRatedCapacity</i>	the capacity

**11.72.3.20 SetRatedCapacityVolatile()** void SetRatedCapacityVolatile (   
 uint32\_t NrChannel,   
 MbcRatedCapacityEnumNet NewRatedCapacity )

sets the rated capacity (i.e. charge current) without storing it persistently

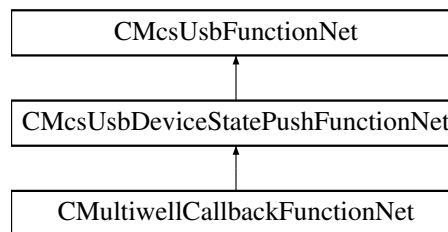
#### Parameters

<i>NrChannel</i>	the channel number
<i>NewRatedCapacity</i>	the capacity

## 11.73 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)
- [CMultiwellCallbackFunctionNet](#) (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ pMultiwellCallbackFunctionPointerContainer)
- *Initializes a new instance of the [CMultiwellCallbackFunctionNet](#) class.*
- [CMultiwellCallbackFunctionNet](#) (CMcsUsbNet^ mcsusb)
- virtual ~[CMultiwellCallbackFunctionNet](#) ()
- ![CMultiwellCallbackFunctionNet](#) ()
- PlateClampEnumNet [GetPlateClampStateByHeadstage](#) (uint32\_t Headstage)

*Gets the state of the plate*

### Events

- [OnGetPlateClampStateByHeadstage](#)^ [GetPlateClampStateByHeadstageEvent](#) [add, remove, raise]

*Event fires when the plate state for the headstage number has changed*

### Additional Inherited Members

#### 11.73.1 Detailed Description

[CMultiwellCallbackFunctionNet](#) is the class to access the Multiwell-Mini-Stimulator

## 11.73.2 Constructor & Destructor Documentation

**11.73.2.1 CMultiwellCallbackFunctionNet()** [1/2] `CMultiwellCallbackFunctionNet ( CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ pMultiwellCallbackFunctionPointerContainer )`

Initializes a new instance of the `CMultiwellCallbackFunctionNet` class.

**11.73.2.2 CMultiwellCallbackFunctionNet()** [2/2] `CMultiwellCallbackFunctionNet ( CMcsUsbNet^ mcsusb )`

**11.73.2.3 ~CMultiwellCallbackFunctionNet()** `virtual ~CMultiwellCallbackFunctionNet ( ) [virtual]`

**11.73.2.4 !CMultiwellCallbackFunctionNet()** `!CMultiwellCallbackFunctionNet ( )`

## 11.73.3 Member Function Documentation

**11.73.3.1 GetPlateClampStateByHeadstage()** `PlateClampEnumNet GetPlateClampStateByHeadstage ( uint32_t Headstage )`

Gets the state of the plate

### Parameters

<i>Headstage</i>	The headstage number
------------------	----------------------

### Returns

The plate state

**11.73.3.2 OnGetPlateClampStateByHeadstage()** `delegate void OnGetPlateClampStateByHeadstage ( uint32_t Headstage, PlateClampEnumNet plateState )`

### 11.73.4 Event Documentation

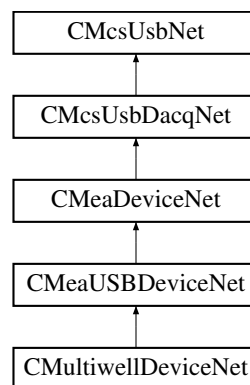
**11.73.4.1 GetPlateClampStateByHeadstageEvent** [OnGetPlateClampStateByHeadstage](#)<sup>^</sup> [GetPlateClampStateByHeadstageEvent](#) [add], [remove], [raise]

Event fires when the plate state for the headstage number has changed

## 11.74 CMultiwellDeviceNet Class Reference

[CMultiwellDeviceNet](#) is the class to access the Multiwell device.

Inheritance diagram for CMultiwellDeviceNet:



### Public Member Functions

- [CMultiwellDeviceNet](#) ()  
*Initializes a new instance of the [CMultiwellDeviceNet](#) class.*
- virtual [~CMultiwellDeviceNet](#) ()
- [!CMultiwellDeviceNet](#) ()
- PlateClampEnumNet [GetPlateClampState](#) ()  
*Gets the state of the Multiwell plate clamp.*
- PlateClampEnumNet [GetPlateClampState](#) (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 [GetPlateType](#) (uint32\_t Headstage)  
*Gets the plate type.*
- void [SetPlateType](#) (MultiwellPlateTypeEnumNet plateType)  
*Sets the plate type.*
- void [SetPlateType](#) (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 [SetPlateMux](#) (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 [GetPlateMux](#) (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 [IsPlateTypeValid](#) (uint32\_t Headstage)  
*Checks whether the plate type is valid, meaning all pins have contact.*
- void [SetPowerMuxPlate](#) (uint32\_t Headstage, bool powerOn)  
*On the Multiwell Mini device, turn Power to the MUX Plate On or Off.*
- bool [GetPowerMuxPlate](#) (uint32\_t Headstage)  
*On the Multiwell Mini device, Query if Power to the MUX Plate is On or Off.*

## Additional Inherited Members

### 11.74.1 Detailed Description

[CMultiwellDeviceNet](#) is the class to access the Multiwell device.

### 11.74.2 Constructor & Destructor Documentation

#### 11.74.2.1 CMultiwellDeviceNet() [CMultiwellDeviceNet](#) ( )

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

#### 11.74.2.2 ~CMultiwellDeviceNet() [virtual](#) [~CMultiwellDeviceNet](#) ( ) [[virtual](#)]

#### 11.74.2.3 ~!CMultiwellDeviceNet() [!CMultiwellDeviceNet](#) ( )

### 11.74.3 Member Function Documentation

#### 11.74.3.1 ClosePlateClamp() `void ClosePlateClamp ( )`

Closes the plate clamp.

#### 11.74.3.2 GetPlateClampLockState() `uint32_t GetPlateClampLockState ( )`

Gets the state of the plate clamp lock.

##### Returns

the state of the plate lock (unlocked/locked)

#### 11.74.3.3 GetPlateClampState() [1/2] `PlateClampEnumNet GetPlateClampState ( )`

Gets the state of the Multiwell plate clamp.

##### Returns

the state of the plate clamp (open/closed)

#### 11.74.3.4 GetPlateClampState() [2/2] `PlateClampEnumNet GetPlateClampState ( uint32_t Headstage )`

Gets the state of the plate

##### Parameters

<i>Headstage</i>	The headstage number
------------------	----------------------

##### Returns

The plate state

#### 11.74.3.5 GetPlateMux() [1/2] `uint32_t GetPlateMux ( )`

Gets the selected quarter of the electrodes on a high density Multiwell plate.

**Returns**

the selected quarter

**11.74.3.6 GetPlateMux()** [2/2] `uint32_t GetPlateMux (`  
`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

**11.74.3.7 GetPlateType()** [1/2] `MultiwellPlateTypeEnumNet GetPlateType ( )`

Gets the plate type.

**Returns**

the plate type

**11.74.3.8 GetPlateType()** [2/2] `MultiwellPlateTypeEnumNet GetPlateType (`  
`uint32_t Headstage )`

Gets the plate type.

**Parameters**

<i>Headstage</i>	The headstage to query.
------------------	-------------------------

**Returns**

the plate type

**11.74.3.9 GetPowerMuxPlate()** `bool GetPowerMuxPlate (`  
`uint32_t Headstage )`

On the Multiwell Mini device, Query if Power to the MUX Plate is On or Off.

## Parameters

<i>Headstage</i>	The headstage to query.
------------------	-------------------------

## Returns

"true" Power is On, "false" Power is Off

#### 11.74.3.10 IsPlateTypeValid() [1/2] `bool IsPlateTypeValid ( )`

Checks whether the plate type is valid, meaning all pins have contact.

## Returns

"true" when all pins have contact, otherwise "false".

#### 11.74.3.11 IsPlateTypeValid() [2/2] `bool IsPlateTypeValid ( 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".

#### 11.74.3.12 LockPlateClamp() `void LockPlateClamp ( )`

Locks the plate clamp.

#### 11.74.3.13 OpenPlateClamp() `void OpenPlateClamp ( )`

Opens the plate clamp.

#### 11.74.3.14 SetPlateMux() [1/2] `void SetPlateMux ( 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

**11.74.3.15 SetPlateMux()** [2/2] 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
---------------------	----------------------

**11.74.3.16 SetPlateType()** [1/2] void SetPlateType (   
MultiwellPlateTypeEnumNet *plateType* )

Sets the plate type.

## Parameters

<i>plateType</i>	the plate type
------------------	----------------

**11.74.3.17 SetPlateType()** [2/2] void SetPlateType (   
uint32\_t *Headstage*,   
MultiwellPlateTypeEnumNet *plateType* )

Sets the plate type.

## Parameters

<i>Headstage</i>	The headstage to query.
<i>plateType</i>	the plate type

**11.74.3.18 SetPowerMuxPlate()** void SetPowerMuxPlate (   
uint32\_t *Headstage*,   
bool *powerOn* )

On the Multiwell Mini device, turn Power to the MUX Plate On or Off.

## Parameters

<i>Headstage</i>	The headstage to query.
<i>powerOn</i>	"true" to turn Power On, "false" to turn Power Off

**11.74.3.19 StopPlateClamp()** void StopPlateClamp ( )

Stops the plate clamp movement.

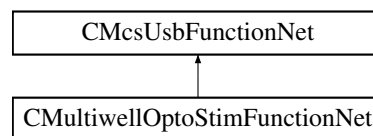
**11.74.3.20 UnlockPlateClamp()** void UnlockPlateClamp ( )

Unlocks the plate clamp.

**11.75 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

### 11.75.1 Detailed Description

[CMultiwellOptoStimFunctionNet](#) is the class to access the optical properties of the Multiwell Optostim device

### 11.75.2 Constructor & Destructor Documentation

**11.75.2.1 CMultiwellOptoStimFunctionNet()** [1/2] [CMultiwellOptoStimFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> *mcsusb*,  
[CMcsUsbFunctionPointerContainer](#)<sup>^</sup> *pMultiwellOptoStimFunctionPointerContainer* )

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

**11.75.2.2 CMultiwellOptoStimFunctionNet()** [2/2] [CMultiwellOptoStimFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> *mcsusb* )

**11.75.2.3 ~CMultiwellOptoStimFunctionNet()** `virtual ~CMultiwellOptoStimFunctionNet ( )` [virtual]

**11.75.2.4 ~!CMultiwellOptoStimFunctionNet()** `!CMultiwellOptoStimFunctionNet ( )`

### 11.75.3 Member Function Documentation

**11.75.3.1 GetAbsMaxCurrentInMicroAmp()** `uint32_t GetAbsMaxCurrentInMicroAmp (`  
`uint16_t channel )`

#### Parameters

<i>channel</i>	the (analog) channel number
----------------	-----------------------------

#### Returns

absolute max. current; unit: uA

**11.75.3.2 GetColorRgb()** `uint32_t GetColorRgb (`  
`uint16_t channel )`

Parameters

<i>channel</i>	the (analog) channel number
----------------	-----------------------------

Returns

RGB-value of LED color

**11.75.3.3 GetColorStr()** `String ^ GetColorStr (`  
`uint16_t channel )`

Parameters

<i>channel</i>	the (analog) channel number
----------------	-----------------------------

Returns

LED color as string

**11.75.3.4 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

**11.75.3.5 GetMaxDutyCycleHighCurrent()** `uint32_t GetMaxDutyCycleHighCurrent (`  
`uint16_t channel )`

Parameters

<i>channel</i>	the (analog) channel number
----------------	-----------------------------



**Returns**

max. duty cycle at max. current; unit: 100\*%

**11.75.3.6 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

**11.75.3.7 GetWaveLengthInNanometer()** `uint32_t GetWaveLengthInNanometer ( uint16_t channel )`

**Parameters**

<i>channel</i>	the (analog) channel number
----------------	-----------------------------

**Returns**

wavelength of this channel's LEDs; unit: nm

**11.75.3.8 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

**11.75.3.9 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

**11.75.3.10 SetColorStr()** void SetColorStr (   
    uint16\_t *channel*,   
    String^ *ColorString* )

## Parameters

<i>channel</i>	the (analog) channel number
<i>ColorString</i>	LED color as string

**11.75.3.11 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

**11.75.3.12 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*%

**11.75.3.13 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

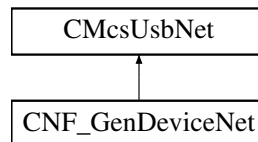
**11.75.3.14 SetWaveLengthInNanometer()** `void SetWaveLengthInNanometer (`  
     `uint16_t channel,`  
     `uint32_t WaveLength_nm )`

#### Parameters

<i>channel</i>	the (analog) channel number
<i>WaveLength_nm</i>	wavelength of this channel's LEDs; unit: nm

## 11.76 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

#### 11.76.1 Constructor & Destructor Documentation

**11.76.1.1 CNF\_GenDeviceNet()** `CNF_GenDeviceNet (`  
     `void )`

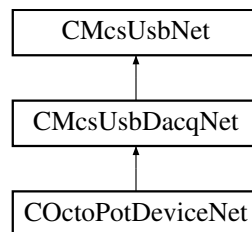
**11.76.1.2 ~CNF\_GenDeviceNet()** `~CNF_GenDeviceNet (`  
     `void )`

#### 11.76.2 Member Function Documentation

**11.76.2.1 Set\_Values()** void Set\_Values (   
     unsigned int *frequency*,   
     unsigned int *amplitude* )

## 11.77 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

#### 11.77.1 Constructor & Destructor Documentation

**11.77.1.1 COctoPotDeviceNet()** [1/2] [COctoPotDeviceNet](#) (   
 void )

**11.77.1.2 COctoPotDeviceNet()** [2/2] [COctoPotDeviceNet](#) (   
 [OnChannelData](#)^ *channelData*,   
 [OnError](#)^ *error* )

## 11.77.2 Member Function Documentation

**11.77.2.1 BurnAdcOffset()** uint32\_t BurnAdcOffset ( )

**11.77.2.2 BurnDacOffset()** uint32\_t BurnDacOffset ( )

**11.77.2.3 EnableChecksum()** uint32\_t EnableChecksum (   
 bool *enable* )

**11.77.2.4 EnableDigitalIn()** uint32\_t EnableDigitalIn (   
 bool *enable* )

**11.77.2.5 EnableTimestamp()** uint32\_t EnableTimestamp (   
 bool *enable* )

**11.77.2.6 GetAdcOffset()** uint32\_t GetAdcOffset (   
 unsigned int *channel*,   
 [System::Runtime::InteropServices::Out] int ^ *offset* )

**11.77.2.7 GetDacOffset()** uint32\_t GetDacOffset (   
 unsigned int *channel*,   
 [System::Runtime::InteropServices::Out] int ^ *offset* )

**11.77.2.8 PatternListStart()** uint32\_t PatternListStart (  
int *channelmap* )

**11.77.2.9 RampStart()** uint32\_t RampStart (  
int *channelmap* )

**11.77.2.10 ResetAdcOffset()** uint32\_t ResetAdcOffset (  
unsigned int *channel* )

**11.77.2.11 ResetDacOffset()** uint32\_t ResetDacOffset (  
unsigned int *channel* )

**11.77.2.12 SetAdcOffset()** uint32\_t SetAdcOffset (  
unsigned int *channel*,  
int *offset* )

**11.77.2.13 SetAmplificationSwitch()** uint32\_t SetAmplificationSwitch (  
unsigned int *channel*,  
unsigned int *state* )

**11.77.2.14 SetBathclamp()** uint32\_t SetBathclamp (  
unsigned int *block*,  
bool *enable* )

**11.77.2.15 SetChannelSwitch()** uint32\_t SetChannelSwitch (  
unsigned int *channel*,  
unsigned int *state* )

**11.77.2.16 SetDacAutoControl()** uint32\_t SetDacAutoControl (  
unsigned int *channel* )

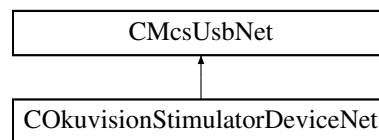
- 11.77.2.17 SetDacOffset()** uint32\_t SetDacOffset (   
 unsigned int *channel*,   
 int *offset* )
- 11.77.2.18 SetDacValue()** uint32\_t SetDacValue (   
 int *channel*,   
 int *value* )
- 11.77.2.19 SetNumberOfChannels()** uint32\_t SetNumberOfChannels (   
 unsigned int *NumberOfChannels* )
- 11.77.2.20 SetOutputRate()** uint32\_t SetOutputRate (   
 uint32\_t *rate* )
- 11.77.2.21 SetPatternListEntry()** uint32\_t SetPatternListEntry (   
 unsigned int *channel*,   
 unsigned int *position*,   
 unsigned int *duration*,   
 int *value* )
- 11.77.2.22 SetPidParameter()** uint32\_t SetPidParameter (   
 unsigned int *channel*,   
 int *const\_p*,   
 int *const\_i*,   
 int *shift\_p*,   
 int *shift\_i* )
- 11.77.2.23 SetRampParameter()** uint32\_t SetRampParameter (   
 unsigned int *channel*,   
 int *start*,   
 int *min*,   
 int *max*,   
 int *slope*,   
 int *slope2*,   
 int *pause*,   
 unsigned int *samples* )

**11.77.2.24 SetSineParameter()** uint32\_t SetSineParameter (   
     unsigned int *channel*,   
     int *amplitude* )

**11.77.2.25 SineStart()** uint32\_t SineStart (   
     int *channelmap* )

## 11.78 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

### 11.78.1 Constructor & Destructor Documentation

**11.78.1.1 COkuvisionStimulatorDeviceNet()** `COkuvisionStimulatorDeviceNet ( void )`

**11.78.1.2 ~COkuvisionStimulatorDeviceNet()** `~COkuvisionStimulatorDeviceNet ( void )`

### 11.78.2 Member Function Documentation

**11.78.2.1 GetCheckVoltage()** `int GetCheckVoltage ( int channel )`

**11.78.2.2 GetCurrentFactor()** `int GetCurrentFactor ( int channel )`

**11.78.2.3 GetDACOffset()** `int GetDACOffset ( int channel, int part )`

**11.78.2.4 GetMaxPower()** `int GetMaxPower ( int channel )`

**11.78.2.5 GetMaxVoltage()** `int GetMaxVoltage ( int channel )`

**11.78.2.6 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* )

**11.78.2.7 GetRTC()** [1/2] DateTime GetRTC ( )

**11.78.2.8 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* )

**11.78.2.9 GetStimulatorStatus()** void GetStimulatorStatus (   
[System::Runtime::InteropServices::Out] int% *startstop*,  
[System::Runtime::InteropServices::Out] int% *last\_error*,  
[System::Runtime::InteropServices::Out] int% *battery\_status* )

**11.78.2.10 GetVoltage()** int GetVoltage (   
int *channel* )

**11.78.2.11 SetCheckVoltage()** void SetCheckVoltage (   
int *channel*,  
int *voltage* )

**11.78.2.12 SetCurrentFactor()** void SetCurrentFactor (   
int *channel*,  
int *factor* )

**11.78.2.13 SetDACOffset()** `void SetDACOffset (`  
     `int channel,`  
     `int part,`  
     `int offset )`

**11.78.2.14 SetMaxPower()** `void SetMaxPower (`  
     `int channel,`  
     `int power )`

**11.78.2.15 SetMaxVoltage()** `void SetMaxVoltage (`  
     `int channel,`  
     `int voltage )`

**11.78.2.16 SetPulseform()** `void SetPulseform (`  
     `int channel,`  
     `int current,`  
     `int pulsewidth,`  
     `int periode,`  
     `int duration )`

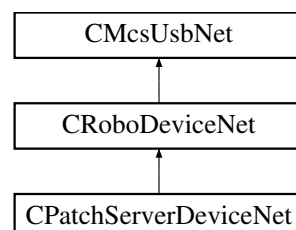
**11.78.2.17 SetRTC() [1/2]** `void SetRTC (`  
     `DateTime timestamp )`

**11.78.2.18 SetRTC() [2/2]** `void SetRTC (`  
     `uint8_t year,`  
     `uint8_t month,`  
     `uint8_t day,`  
     `uint8_t hour,`  
     `uint8_t minute,`  
     `uint8_t second )`

## 11.79 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****11.79.1 Detailed Description**

[CPatchServerDeviceNet](#) is the class to control the MCS PatchServer device

**11.79.2 Constructor & Destructor Documentation**

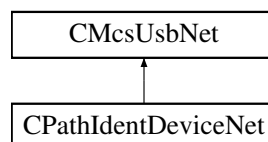
**11.79.2.1** [CPatchServerDeviceNet](#)() [CPatchServerDeviceNet](#) (  
void )

**11.79.3 Property Documentation**

**11.79.3.1** [Sensor](#) [CMcsBus\\_SensorNet](#)<sup>^</sup> [Sensor](#) [get]

**11.80 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

### 11.80.1 Constructor & Destructor Documentation

**11.80.1.1 CPathIdentDeviceNet()** `CPathIdentDeviceNet ( void )`

**11.80.1.2 ~CPathIdentDeviceNet()** `~CPathIdentDeviceNet ( void )`

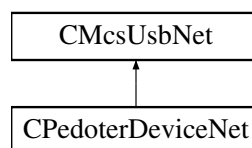
### 11.80.2 Member Function Documentation

**11.80.2.1 Measure()** `void Measure ( [System::Runtime::InteropServices::Out] unsigned int% phase, [System::Runtime::InteropServices::Out] unsigned int% amplitude )`

**11.80.2.2 Set\_Values()** `void Set_Values ( unsigned int frequency, unsigned int amplitude )`

## 11.81 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

### 11.81.1 Detailed Description

### 11.81.2 Constructor & Destructor Documentation

#### 11.81.2.1 CPedoterDeviceNet() `CPedoterDeviceNet ( )`

Initializes a new instance of the `CPedoterDeviceNet` class.

#### 11.81.2.2 ~CPedoterDeviceNet() `virtual ~CPedoterDeviceNet ( ) [virtual]`

#### 11.81.2.3 !CPedoterDeviceNet() `!CPedoterDeviceNet ( )`

### 11.81.3 Member Function Documentation

#### 11.81.3.1 GetCommand() `uint32_t GetCommand ( uint16_t Argument )`

Get value from the pedoter device

##### Parameters

<i>Argument</i>	argument
-----------------	----------

##### Returns

value

#### 11.81.3.2 SetCommand() `void SetCommand ( uint16_t Argument, uint32_t pData )`

Set value on the pedoter device

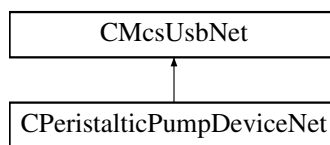
## Parameters

<i>Argument</i>	argument
<i>pData</i>	value

## 11.82 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](#)<sup>^</sup> [McsBus\\_MotorControl](#) [get]

### Additional Inherited Members

#### 11.82.1 Detailed Description

[CPeristalticPumpDeviceNet](#) is the class to control a Persistaltic Pump.

#### 11.82.2 Constructor & Destructor Documentation

##### 11.82.2.1 CPeristalticPumpDeviceNet() [CPeristalticPumpDeviceNet](#) ( void )

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

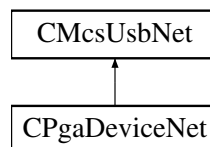
**11.82.2.2** `~CPeristalticPumpDeviceNet()` `~CPeristalticPumpDeviceNet` (  
     `void` )

### 11.82.3 Property Documentation

**11.82.3.1** `McsBus_MotorControl` `CMcsBus_MotorControlNet`<sup>^</sup> `McsBus_MotorControl` [get]

## 11.83 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

#### 11.83.1 Constructor & Destructor Documentation

**11.83.1.1** `CPgaDeviceNet()` `CPgaDeviceNet` ( )



### 11.83.1.2 ~CPgaDeviceNet() ~CPgaDeviceNet ( )

## 11.83.2 Member Function Documentation

### 11.83.2.1 ApplyGains() uint32\_t ApplyGains ( )

### 11.83.2.2 DefineAmplification() uint32\_t DefineAmplification ( int *index*, int *amplification*, int *potil*, int *poti2* )

### 11.83.2.3 DefineFrequencyRange() uint32\_t DefineFrequencyRange ( int *index*, int *low*, int *high*, int *channels*, int *gain* )

### 11.83.2.4 DefineNumAmplifications() uint32\_t DefineNumAmplifications ( int *number* )

### 11.83.2.5 DefineNumFrequencyRanges() uint32\_t DefineNumFrequencyRanges ( int *rnum* )

### 11.83.2.6 GetAmplification() uint32\_t GetAmplification ( int *index*, [System::Runtime::InteropServices::Out] int% *amplification*, [System::Runtime::InteropServices::Out] int% *potil*, [System::Runtime::InteropServices::Out] int% *poti2* )

**11.83.2.7 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* )

**11.83.2.8 GetGain()** uint32\_t GetGain (   
     int *channel*,   
     [System::Runtime::InteropServices::Out] int% *Gain*,   
     [System::Runtime::InteropServices::Out] int% *poti1*,   
     [System::Runtime::InteropServices::Out] int% *poti2* )

**11.83.2.9 GetNumAmplifications()** uint32\_t GetNumAmplifications (   
     [System::Runtime::InteropServices::Out] int% *number* )

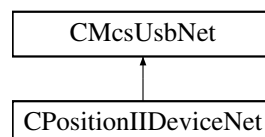
**11.83.2.10 GetNumFrequencyRanges()** uint32\_t GetNumFrequencyRanges (   
     [System::Runtime::InteropServices::Out] int% *numRanges* )

**11.83.2.11 SetGain()** uint32\_t SetGain (   
     int *channel*,   
     int *Gain*,   
     int *poti1*,   
     int *poti2* )

## 11.84 CPositionIIDeviceNet Class Reference

[CPositionIIDeviceNet](#) is the class to control PositionII devices

Inheritance diagram for CPositionIIDeviceNet:



## Public Member Functions

- [CPositionIODeviceNet](#) ()  
*Initializes a new instance of the [CPositionIODeviceNet](#) class.*
- virtual [~CPositionIODeviceNet](#) ()
- [ICPositionIODeviceNet](#) ()
- [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 coild 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*
- [uint32\\_t GetPowerStrength](#) ([uint16\\_t](#) coil)  
*sets the power for the trigger pulses*
- void [SetPowerStrength](#) ([uint16\\_t](#) coil, [uint32\\_t](#) power)  
*gets the power for the trigger pulses*
- [uint32\\_t GetImplantResult](#) ([uint16\\_t](#) coil)  
*gets the last result of the implant pulse trigger*
- 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)  
*Get the RTC*
- void [SetRTC](#) ([uint8\\_t](#) year, [uint8\\_t](#) month, [uint8\\_t](#) day, [uint8\\_t](#) hour, [uint8\\_t](#) minute, [uint8\\_t](#) second)  
*Set the RTC*
- [uint32\\_t GetStateDebugData](#) ([uint16\\_t](#) coil)  
*get the debug queue state*
- void [SetStateDebugData](#) ([uint16\\_t](#) coil, [uint32\\_t](#) state)  
*clears/starts/stops the debug queue for a certain coil*
- void [GetDebugData](#) ([uint16\\_t](#) coil, [[System::Runtime::InteropServices::Out](#)][uint16\\_t](#)% index, [[System::Runtime::InteropServices::Out](#)][uint16\\_t](#)% voltage, [[System::Runtime::InteropServices::Out](#)][uint16\\_t](#)% numberofpulses, [[System::Runtime::InteropServices::Out](#)][uint16\\_t](#)% mediantime)  
*get the oldest debug entry for a certain coil*
- [uint32\\_t GetStateEventData](#) ()  
*get the event queue state*
- void [SetStateEventData](#) ([uint32\\_t](#) state)  
*clears/starts/stops the event queue for a certain coil*
- void [GetEventData](#) ([[System::Runtime::InteropServices::Out](#)][uint16\\_t](#)% index, [[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, [[System::Runtime::InteropServices::Out](#)][uint16\\_t](#)% coil, [[System::Runtime::InteropServices::Out](#)][uint16\\_t](#)% type, [[System::Runtime::InteropServices::Out](#)][uint16\\_t](#)% value)  
*get the oldest event entry*

## Properties

- [CRFFunctionNet](#)<sup>^</sup> [RFFunction](#) [get]

## Additional Inherited Members

### 11.84.1 Detailed Description

[CPositionIODeviceNet](#) is the class to control PositionII devices

### 11.84.2 Constructor & Destructor Documentation

#### 11.84.2.1 CPositionIODeviceNet() [CPositionIODeviceNet](#) ( )

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

#### 11.84.2.2 ~CPositionIODeviceNet() [virtual](#) [~CPositionIODeviceNet](#) ( ) [virtual]

#### 11.84.2.3 "!CPositionIODeviceNet() [!CPositionIODeviceNet](#) ( )

### 11.84.3 Member Function Documentation

#### 11.84.3.1 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

**11.84.3.2 GetDebugData()** void GetDebugData (   
     uint16\_t *coil*,   
     [System::Runtime::InteropServices::Out] uint16\_t% *index*,   
     [System::Runtime::InteropServices::Out] uint16\_t% *voltage*,   
     [System::Runtime::InteropServices::Out] uint16\_t% *numberofpulses*,   
     [System::Runtime::InteropServices::Out] uint16\_t% *mediantime* )

get the oldest debug entry for a certain coil

#### Parameters

<i>coil</i>	the coil
<i>index</i>	the debug entry index number
<i>voltage</i>	the voltage applied
<i>numberofpulses</i>	the number of pulses detected
<i>mediantime</i>	the median time between pulses

**11.84.3.3 GetEventData()** void GetEventData (   
     [System::Runtime::InteropServices::Out] uint16\_t% *index*,   
     [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*,   
     [System::Runtime::InteropServices::Out] uint16\_t% *coil*,   
     [System::Runtime::InteropServices::Out] uint16\_t% *type*,   
     [System::Runtime::InteropServices::Out] uint16\_t% *value* )

get the oldest event entry

#### Parameters

<i>index</i>	the event index number
<i>year</i>	the year
<i>month</i>	the month
<i>day</i>	the day
<i>hour</i>	the hour
<i>minute</i>	the minute
<i>second</i>	the second
<i>coil</i>	the coil
<i>type</i>	the event type
<i>value</i>	the even value

**11.84.3.4 GetImplantCurrentSetpoint()** uint32\_t GetImplantCurrentSetpoint (   
     uint16\_t *coil* )

sets the implant current setpoint

## Parameters

<i>coil</i>	the coil
-------------	----------

## Returns

the current

**11.84.3.5 GetImplantResult()** uint32\_t GetImplantResult (   
uint16\_t *coil* )

gets the last result of the implant pulse trigger

## Parameters

<i>coil</i>	the coil
-------------	----------

## Returns

the result

**11.84.3.6 GetImplantState()** uint32\_t GetImplantState (   
uint16\_t *coil* )

gets the implantat state

## Parameters

<i>coil</i>	the coil
-------------	----------

## Returns

the state

**11.84.3.7 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

**11.84.3.8 GetPowerStrength()** `uint32_t GetPowerStrength (`  
     `uint16_t coil )`

sets the power for the trigger pulses

**Parameters**

<i>coil</i>	the coil
-------------	----------

**Returns**

the power

**11.84.3.9 GetRTC()** `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 )`

Get the RTC

**Parameters**

<i>year</i>	the year
<i>month</i>	the month
<i>day</i>	the day
<i>hour</i>	the hour
<i>minute</i>	the minute
<i>second</i>	the second

**11.84.3.10 GetStateDebugData()** `uint32_t GetStateDebugData (`  
     `uint16_t coil )`

get the debug queue state

**Parameters**

<i>coil</i>	the coil
-------------	----------

**Returns**

the state

**11.84.3.11 GetStateEventData()** `uint32_t GetStateEventData ( )`

get the event queue state

**Returns**

the state

**11.84.3.12 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

**11.84.3.13 SetPowerStrength()** `void SetPowerStrength (   
 uint16_t coil,   
 uint32_t power )`

gets the power for the trigger pulses

**Parameters**

<i>coil</i>	the coil
<i>power</i>	the power

**11.84.3.14 SetRTC()** `void SetRTC (   
 uint8_t year,   
 uint8_t month,   
 uint8_t day,   
 uint8_t hour,   
 uint8_t minute,   
 uint8_t second )`



Set the RTC

## Parameters

<i>year</i>	the year
<i>month</i>	the month
<i>day</i>	the day
<i>hour</i>	the hour
<i>minute</i>	the minute
<i>second</i>	the second

**11.84.3.15 SetStateDebugData()** `void SetStateDebugData (`  
    `uint16_t coil,`  
    `uint32_t state )`

clears/starts/stops the debug queue for a certain coil

## Parameters

<i>coil</i>	the coil
<i>state</i>	clear/start/stop

**11.84.3.16 SetStateEventData()** `void SetStateEventData (`  
    `uint32_t state )`

clears/starts/stops the event queue for a certain coil

## Parameters

<i>state</i>	clear/start/stop
--------------	------------------

**11.84.3.17 SwitchOnOff()** `void SwitchOnOff (`  
    `uint16_t coil,`  
    `uint32_t on )`

switched the coil on of

## Parameters

<i>coil</i>	the coil
<i>on</i>	0 = off, 1 = on

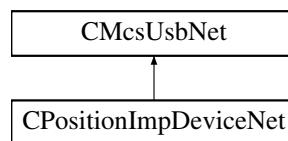
### 11.84.4 Property Documentation

**11.84.4.1 RFFunction** [CRFFunctionNet](#)<sup>^</sup> RFFunction [get]

## 11.85 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 [GetImplId](#) ()  
*Gets the ID of the impedance measure device*
- void [SetImplId](#) (uint32\_t id)  
*Sets the ID of the impedance measure device*

### Additional Inherited Members

#### 11.85.1 Detailed Description

[CPositionImpDeviceNet](#) is the class to access the Position/Imp devices

## 11.85.2 Constructor & Destructor Documentation

### 11.85.2.1 CPositionImpDeviceNet() CPositionImpDeviceNet ( )

Initializes a new instance of the CPositionImpDeviceNet class.

### 11.85.2.2 ~CPositionImpDeviceNet() virtual ~CPositionImpDeviceNet ( ) [virtual]

### 11.85.2.3 ~!CPositionImpDeviceNet() !CPositionImpDeviceNet ( )

## 11.85.3 Member Function Documentation

### 11.85.3.1 ConnectedImp() uint32\_t ConnectedImp ( )

The ID of the connected Imp device

#### Returns

The ID

### 11.85.3.2 ConnectImp() void ConnectImp ( uint32\_t id )

Connect to a Imp device with a certain ID

#### Parameters

<i>id</i>	The ID
-----------	--------

### 11.85.3.3 GetDeviceList() uint32\_t GetDeviceList ( int32\_t index )

Gets the device list

## Parameters

<i>index</i>	the index
--------------	-----------

## Returns

the ID

**11.85.3.4 GetImpId()** `uint32_t GetImpId ( )`

Gets the ID of the impedance measure device

## Returns

the ID

**11.85.3.5 GetRFFrequency()** `int32_t GetRFFrequency ( )`

Gets currently used RF frequency

## Returns

The frequency

**11.85.3.6 SetDeviceList()** `void SetDeviceList (   
int32_t index,   
uint32_t id )`

Sets the device list

## Parameters

<i>index</i>	the index
<i>id</i>	the ID

**11.85.3.7 SetImpId()** `void SetImpId (   
uint32_t id )`

Sets the ID of the impedance measure device

## Parameters

<i>id</i>	the ID
-----------	--------

**11.85.3.8 SetRFFrequency()** `void SetRFFrequency (`  
`int32_t frequency )`

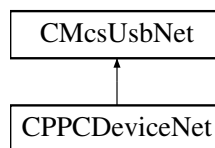
Sets the current RF frequency

## Parameters

<i>frequency</i>	The frequency
------------------	---------------

## 11.86 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

#### 11.86.1 Constructor & Destructor Documentation

**11.86.1.1 CPPCDeviceNet()** `CPPCDeviceNet (`  
`void )`

## 11.86.2 Property Documentation

**11.86.2.1 McsBus** [CMcsBusNet](#)<sup>^</sup> McsBus [get]

**11.86.2.2 McsBus\_MotorControl** [CMcsBus\\_MotorControlNet](#)<sup>^</sup> McsBus\_MotorControl [get]

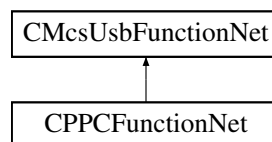
**11.86.2.3 McsBus\_Sensor** [CMcsBus\\_SensorNet](#)<sup>^</sup> McsBus\_Sensor [get]

**11.86.2.4 PPCFunction** [CPPCFunctionNet](#)<sup>^</sup> PPCFunction [get]

## 11.87 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> pPPCFunctionPointerContainer)
  - 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

### 11.87.1 Detailed Description

[CPPCFunctionNet](#) is the class to access the PPC (high precision Patch Peristaltic patch Pump

### 11.87.2 Constructor & Destructor Documentation

**11.87.2.1 CPPCFunctionNet()** [1/2] [CPPCFunctionNet](#) (  
     CMcsUsbNet^ mcsusb,  
     CMcsUsbFunctionPointerContainer^ pPPCFunctionPointerContainer )

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



**11.87.2.2 CPPCFunctionNet()** [2/2] `CPPCFunctionNet (   
CMcsUsbNet^ mcsusb )`

**11.87.2.3 ~CPPCFunctionNet()** `virtual ~CPPCFunctionNet ( ) [virtual]`

**11.87.2.4 "!CPPCFunctionNet()** `!CPPCFunctionNet ( )`

### 11.87.3 Member Function Documentation

**11.87.3.1 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

**11.87.3.2 GetAnalogVoltage()** `uint16_t GetAnalogVoltage (   
uint16_t channel )`

Reads the current analog voltage

#### Parameters

<i>channel</i>	The Channel Number
----------------	--------------------

#### Returns

The Analog Voltage

**11.87.3.3 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

**11.87.3.4 GetDigitalIn()** `uint16_t GetDigitalIn (`  
`uint16_t channel )`

Reads the digital input state

## Parameters

<i>channel</i>	The Channel Number
----------------	--------------------

## Returns

The Digital State

**11.87.3.5 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

**11.87.3.6 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.

**11.87.3.7 GetPumpSpeedUnit()** `int GetPumpSpeedUnit ( uint16_t channel )`

Reads the Pump Speed Unit

**Parameters**

<i>channel</i>	The Channel Number
----------------	--------------------

**Returns**

The Speed Unit

**11.87.3.8 GetSupplyVoltage()** `uint16_t GetSupplyVoltage ( )`

Reads the current supply voltage in mV

**Returns**

The supply voltage

**11.87.3.9 GetValveActive()** `int GetValveActive ( uint16_t valve )`

Gets the valve active/inactive state

**Parameters**

<i>valve</i>	The valve number
--------------	------------------

**Returns**

The valve state

**11.87.3.10 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

**11.87.3.11 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

**11.87.3.12 MeasureReservoir()** `int32_t MeasureReservoir ( )`

Measures the reservoir pressure

## Returns

The pressure

**11.87.3.13 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

**11.87.3.14 SetPressureOffset()** `void SetPressureOffset ( )`

Sets the pressure offset

**11.87.3.15 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

**11.87.3.16 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.

**11.87.3.17 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

**11.87.3.18 SetValveActive()** void SetValveActive (   
     uint16\_t *valve*,   
     int *valveActive* )

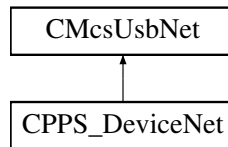
Sets the valve active/inactive state

Parameters

<i>valve</i>	The valve number
<i>valveActive</i>	The valve state

## 11.88 CPPS\_DeviceNet Class Reference

Inheritance diagram for CPPS\_DeviceNet:



### Public Member Functions

- [CPPS\\_DeviceNet](#) (void)

### Properties

- [CPPS\\_FunctionNet](#)<sup>^</sup> [PPS\\_Function](#) [get]
- [CMcsBusNet](#)<sup>^</sup> [McsBus](#) [get]
- [CMcsBus\\_MotorControlNet](#)<sup>^</sup> [McsBus\\_MotorControl](#) [get]
- [CMcsBus\\_SensorNet](#)<sup>^</sup> [McsBus\\_Sensor](#) [get]

### Additional Inherited Members

#### 11.88.1 Constructor & Destructor Documentation

**11.88.1.1** [CPPS\\_DeviceNet\(\)](#) [CPPS\\_DeviceNet](#) (  
void )

#### 11.88.2 Property Documentation

**11.88.2.1** [McsBus](#) [CMcsBusNet](#)<sup>^</sup> [McsBus](#) [get]

**11.88.2.2** [McsBus\\_MotorControl](#) [CMcsBus\\_MotorControlNet](#)<sup>^</sup> [McsBus\\_MotorControl](#) [get]

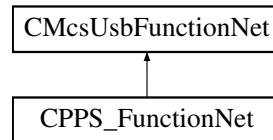
**11.88.2.3** [McsBus\\_Sensor](#) [CMcsBus\\_SensorNet](#)<sup>^</sup> [McsBus\\_Sensor](#) [get]



#### 11.88.2.4 PPS\_Function `CPPS_FunctionNet^ PPS_Function [get]`

### 11.89 CPPS\_FunctionNet Class Reference

Inheritance diagram for CPPS\_FunctionNet:



#### Public Member Functions

- `CPPS_FunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ cPPS_FunctionPointerContainer)`
- `CPPS_FunctionNet (CMcsUsbNet^ 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 GetDigitalln (unsigned short index)`
- `unsigned short GetBubbleState ()`

#### Additional Inherited Members

#### 11.89.1 Constructor & Destructor Documentation

**11.89.1.1 CPPS\_FunctionNet()** [1/2] `CPPS_FunctionNet (`  
    `CMcsUsbNet^ mcsusb,`  
    `CMcsUsbFunctionPointerContainer^ cPPS_FunctionPointerContainer )`

**11.89.1.2 CPPS\_FunctionNet()** [2/2] `CPPS_FunctionNet (`  
    `CMcsUsbNet^ mcsusb )`

## 11.89.2 Member Function Documentation

**11.89.2.1 GetAnalogVoltage()** `unsigned short GetAnalogVoltage (`  
    `unsigned short index )`

**11.89.2.2 GetAnalogVoltages()** `void GetAnalogVoltages (`  
    `unsigned short index,`  
    `[System::Runtime::InteropServices::Out] unsigned short% minvoltage,`  
    `[System::Runtime::InteropServices::Out] unsigned short% maxvoltage )`

**11.89.2.3 GetBubbleState()** `unsigned short GetBubbleState ( )`

**11.89.2.4 GetDigitalIn()** `unsigned short GetDigitalIn (`  
    `unsigned short index )`

**11.89.2.5 GetPumpCouple()** `unsigned int GetPumpCouple ( )`

**11.89.2.6 GetPumpEnableSpeedRatio()** `unsigned int GetPumpEnableSpeedRatio ( )`

**11.89.2.7 GetPumpFastOnOff()** `unsigned int GetPumpFastOnOff (`  
    `unsigned short index )`

**11.89.2.8 GetPumpFastSpeed()** short GetPumpFastSpeed (   
 unsigned short *index* )

**11.89.2.9 GetPumpFunctionSpeeds()** void GetPumpFunctionSpeeds (   
 unsigned short *index*,   
 [System::Runtime::InteropServices::Out] short% *offspeed*,   
 [System::Runtime::InteropServices::Out] short% *onspeed* )

**11.89.2.10 GetPumpManualOnOff()** unsigned int GetPumpManualOnOff (   
 unsigned short *index* )

**11.89.2.11 GetPumpMaxSpeed()** unsigned short GetPumpMaxSpeed (   
 unsigned short *index* )

**11.89.2.12 GetPumpModeType()** PP\_Pump\_Mode\_Type\_EnumNet GetPumpModeType (   
 unsigned short *index* )

**11.89.2.13 GetPumpSpeedRatio()** int GetPumpSpeedRatio ( )

**11.89.2.14 GetPumpSpeedUnit()** int GetPumpSpeedUnit (   
 unsigned short *index* )

**11.89.2.15 GetSupplyVoltage()** unsigned short GetSupplyVoltage ( )

**11.89.2.16 GetUseBubble()** unsigned int GetUseBubble (   
 unsigned short *index* )

**11.89.2.17 SetAnalogVoltages()** void SetAnalogVoltages (   
    unsigned short *index*,  
    unsigned short *minvoltage*,  
    unsigned short *maxvoltage* )

**11.89.2.18 SetPumpCouple()** void SetPumpCouple (   
    unsigned int *i* )

**11.89.2.19 SetPumpEnableSpeedRatio()** void SetPumpEnableSpeedRatio (   
    unsigned int *enable* )

**11.89.2.20 SetPumpFastOnOff()** void SetPumpFastOnOff (   
    unsigned short *index*,  
    unsigned int *onoff* )

**11.89.2.21 SetPumpFastSpeed()** void SetPumpFastSpeed (   
    unsigned short *index*,  
    short *fastspeed* )

**11.89.2.22 SetPumpFunctionSpeeds()** void SetPumpFunctionSpeeds (   
    unsigned short *index*,  
    short *offspeed*,  
    short *onspeed* )

**11.89.2.23 SetPumpManualOnOff()** void SetPumpManualOnOff (   
    unsigned short *index*,  
    unsigned int *onoff* )

**11.89.2.24 SetPumpMaxSpeed()** void SetPumpMaxSpeed (   
    unsigned short *index*,  
    unsigned short *maxspeed* )

**11.89.2.25 SetPumpModeType()** `void SetPumpModeType (`  
     `unsigned short index,`  
     `PP_Pump_Mode_Type_EnumNet type )`

**11.89.2.26 SetPumpSpeedRatio()** `void SetPumpSpeedRatio (`  
     `int ratio )`

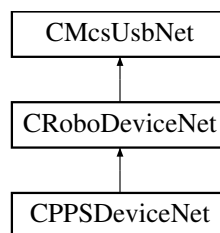
**11.89.2.27 SetPumpSpeedUnit()** `void SetPumpSpeedUnit (`  
     `unsigned short index,`  
     `int speedunit )`

**11.89.2.28 SetUseBubble()** `void SetUseBubble (`  
     `unsigned short index,`  
     `unsigned int usebubble )`

## 11.90 CPPSDeviceNet Class Reference

CPPS4plus1DeviceNet is the to control the MCS HLA device

Inheritance diagram for CPPSDeviceNet:



### Public Member Functions

- [CPPSDeviceNet](#) (void)

### Additional Inherited Members

#### 11.90.1 Detailed Description

CPPS4plus1DeviceNet is the to control the MCS HLA device

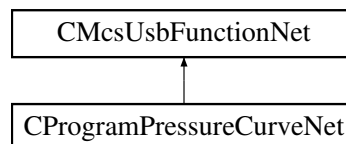
## 11.90.2 Constructor & Destructor Documentation

**11.90.2.1 CPPSDeviceNet()** `CPPSDeviceNet ( void )`

## 11.91 CProgramPressureCurveNet Class Reference

[CProgramPressureCurveNet](#) is the class to program pressure curves

Inheritance diagram for CProgramPressureCurveNet:



### Public Member Functions

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

#### 11.91.1 Detailed Description

[CProgramPressureCurveNet](#) is the class to program pressure curves

#### 11.91.2 Constructor & Destructor Documentation

**11.91.2.1 CProgramPressureCurveNet()** `CProgramPressureCurveNet ( CMcsUsbNet^ mcsusb )`

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

**11.91.2.2** `~CProgramPressureCurveNet()` `virtual ~CProgramPressureCurveNet (`  
`void ) [virtual]`

**11.91.2.3** `!CProgramPressureCurveNet()` `!CProgramPressureCurveNet (`  
`void )`

### 11.91.3 Member Function Documentation

**11.91.3.1** `GetRepeats()` `unsigned int GetRepeats (`  
`unsigned char busnumber,`  
`unsigned char busaddress,`  
`int32_t channel )`

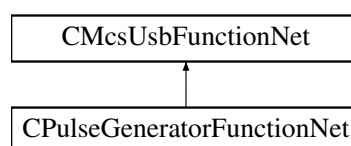
**11.91.3.2** `Program()` `void Program (`  
`unsigned char busnumber,`  
`unsigned char busaddress,`  
`int32_t channel,`  
`array< int32_t >^ pressures,`  
`array< int32_t >^ steps,`  
`array< int16_t >^ durations )`

**11.91.3.3** `SetRepeats()` `void SetRepeats (`  
`unsigned char busnumber,`  
`unsigned char busaddress,`  
`int32_t channel,`  
`uint32_t repeats )`

## 11.92 CPulseGeneratorFunctionNet Class Reference

[CPulseGeneratorFunctionNet](#) is the class to control the pulse generator for video tracking

Inheritance diagram for CPulseGeneratorFunctionNet:



## Public Member Functions

- [CPulseGeneratorFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> pPulseGeneratorFunctionPointerContainer)  
*Initializes a new instance of the [CPulseGeneratorFunctionNet](#) class.*
- [CPulseGeneratorFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- virtual [~CPulseGeneratorFunctionNet](#) ()
- [ICPulseGeneratorFunctionNet](#) ()
- [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]PulseGeneratorMode\_EnumNet% mode, [System::Runtime::InteropServices::Out][int32\\_t](#)% digitalchannel)  
*Reads the generator mode*
- void [SetModeSelect](#) ([int32\\_t](#) generator\_number, PulseGeneratorMode\_EnumNet mode, [int32\\_t](#) digitalchannel)  
*Writes the generator mode*

## Additional Inherited Members

### 11.92.1 Detailed Description

[CPulseGeneratorFunctionNet](#) is the class to control the pulse generator for video tracking

### 11.92.2 Constructor & Destructor Documentation

**11.92.2.1 [CPulseGeneratorFunctionNet](#)()** [1/2] [CPulseGeneratorFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb,  
[CMcsUsbFunctionPointerContainer](#)<sup>^</sup> pPulseGeneratorFunctionPointerContainer )

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

**11.92.2.2 [CPulseGeneratorFunctionNet](#)()** [2/2] [CPulseGeneratorFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb )

**11.92.2.3 [~CPulseGeneratorFunctionNet](#)()** virtual [~CPulseGeneratorFunctionNet](#) ( ) [virtual]



**11.92.2.4** `!CPulseGeneratorFunctionNet() !CPulseGeneratorFunctionNet ( )`

### 11.92.3 Member Function Documentation

**11.92.3.1 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

**11.92.3.2 GetPeriod()** `int32_t GetPeriod (`  
    `int32_t generator_number )`

Reads the generator period

#### Parameters

<i>generator_number</i>	The generator number
-------------------------	----------------------

#### Returns

The period

**11.92.3.3 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

**11.92.3.4 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

**11.92.3.5 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

**11.92.3.6 SetPulseLength()** void SetPulseLength (   
     int32\_t *generator\_number*,   
     int32\_t *pulselength\_in\_10us* )

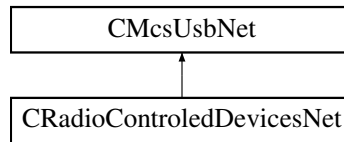
Writes the generator pulse length

**Parameters**

<i>generator_number</i>	The generator number
<i>pulselength_in_10us</i>	The pulse length

## 11.93 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

#### 11.93.1 Constructor & Destructor Documentation

**11.93.1.1 CRadioControlledDevicesNet()** [1/2] [CRadioControlledDevicesNet](#) ( void )

**11.93.1.2 CRadioControlledDevicesNet()** [2/2] [CRadioControlledDevicesNet](#) ( CRadioControlledDevices \* pRadioControlled ) [protected]

#### 11.93.2 Member Function Documentation

**11.93.2.1 ConnectDevice()** void [ConnectDevice](#) ( unsigned short *sn* )

**11.93.2.2 DisConnectDevice()** void [DisConnectDevice](#) ( )

**11.93.2.3 GetDeviceNames()** `array<unsigned short> ^ GetDeviceNames ( )`

**11.93.2.4 GetFrequency()** `unsigned short GetFrequency ( )`

**11.93.2.5 HasRadioControl()** `bool HasRadioControl ( )`

**11.93.2.6 SetFrequency()** `void SetFrequency (   
 unsigned short frequency )`

**11.93.2.7 StillConnected()** `bool StillConnected ( )`

## 11.94 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](#)

### 11.94.1 Constructor & Destructor Documentation

**11.94.1.1 CRegionOfInterestRect()** `CRegionOfInterestRect (   
 int left,   
 int top,   
 int right,   
 int bottom )`

### 11.94.2 Member Function Documentation

**11.94.2.1 DeepCopy()** [CRegionOfInterestRect](#) ^ DeepCopy ( )

### 11.94.3 Member Data Documentation

**11.94.3.1 m\_Bottom** `int m_Bottom`

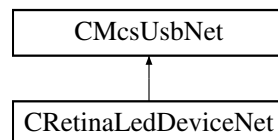
**11.94.3.2 m\_Left** `int m_Left`

**11.94.3.3 m\_Right** `int m_Right`

**11.94.3.4 m\_Top** `int m_Top`

## 11.95 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

### 11.95.1 Constructor & Destructor Documentation

**11.95.1.1 CRetinaLedDeviceNet()** `CRetinaLedDeviceNet ( )`

**11.95.1.2 ~CRetinaLedDeviceNet()** `~CRetinaLedDeviceNet ( )`

### 11.95.2 Member Function Documentation

**11.95.2.1 AddLoopEntry()** `unsigned int AddLoopEntry (   
 unsigned short repeats,   
 unsigned short steps_back )`

**11.95.2.2 AddTableEntry()** `unsigned int AddTableEntry (   
 unsigned long long pattern )`

**11.95.2.3 ClearTable()** `unsigned int ClearTable ( )`

**11.95.2.4 GetTablepointer()** `unsigned int GetTablepointer (   
 int % position )`

**11.95.2.5 SetLED()** `unsigned int SetLED (   
 unsigned long long pattern )`

**11.95.2.6 SetLumi()** `unsigned int SetLumi (   
 int lumi )`

**11.95.2.7 SetPersistency()** unsigned int SetPersistency (   
 unsigned int *persistency* )

**11.95.2.8 SetRepeat()** unsigned int SetRepeat (   
 int *repeat* )

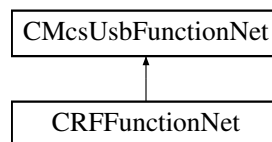
**11.95.2.9 SetTablepointer()** unsigned int SetTablepointer (   
 int *position* )

**11.95.2.10 SetTrigger()** unsigned int SetTrigger (   
 int *enable* )

## 11.96 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> pRFFunctionPointer↔ Container)
- Initializes a new instance of the [CRFFunctionNet](#) class.*
- [CRFFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- virtual [~CRFFunctionNet](#) ()
- [ICRFFunctionNet](#) ()
- 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 > ^ [GetAvailableDeviceListEx](#) (int list\_Length)
- get a list of available devices*
- array< uint32\_t > ^ [GetAvailableDeviceList](#) ()

- get a list of available devices*
- array< uint32\_t > ^ [GetAvailableStateListEx](#) (int list\_Length)
- get a list of the states of the available devices*
- array< uint32\_t > ^ [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

### 11.96.1 Detailed Description

[CRFFunctionNet](#) is the class to control RF devices

### 11.96.2 Constructor & Destructor Documentation

**11.96.2.1 CRFFunctionNet()** [1/2] [CRFFunctionNet](#) (  
[CMcsUsbNet](#)^ mcsusb,  
[CMcsUsbFunctionPointerContainer](#)^ pRFFunctionPointerContainer )

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

**11.96.2.2 CRFFunctionNet()** [2/2] [CRFFunctionNet](#) (  
[CMcsUsbNet](#)^ mcsusb )

**11.96.2.3 ~CRFFunctionNet()** virtual [~CRFFunctionNet](#) ( ) [virtual]

**11.96.2.4 "!CRFFunctionNet()** [!CRFFunctionNet](#) ( )

### 11.96.3 Member Function Documentation

**11.96.3.1 Connect()** void [Connect](#) (  
uint32\_t sn )

connect to a RF device, use 0 to disconnect



## Parameters

<i>sn</i>	the serial number
-----------	-------------------

**11.96.3.2 GetAvailableDeviceList()** `array<uint32_t> ^ GetAvailableDeviceList ( )`

get a list of available devices

## Returns

array of devices

**11.96.3.3 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

array of devices

**11.96.3.4 GetAvailableStateList()** `array<uint32_t> ^ GetAvailableStateList ( )`

get a list of the states of the available devices

## Returns

array of states

**11.96.3.5 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**

array of states

**11.96.3.6 GetBaseFrequency()** `uint32_t GetBaseFrequency ( CFirmwareDestinationNet destination )`

gets the base advertise frequency

**Parameters**

<i>destination</i>	the destination to query
--------------------	--------------------------

**Returns**

the frequency

**11.96.3.7 GetConnectedDevice()** `uint32_t GetConnectedDevice ( )`

get connect RF device, 0 = no device connected

**Returns**

the serial number

**11.96.3.8 GetState()** `uint32_t GetState ( )`

get connection state

**Returns**

the state

**11.96.3.9 GetTestMode()** `uint32_t GetTestMode ( )`

gets test mode

**Returns**

the mode

**11.96.3.10 GetWorkingFrequency()** `uint32_t GetWorkingFrequency ( )`

gets the working frequency

**Returns**

the frequency

**11.96.3.11 SetBaseFrequency()** `void SetBaseFrequency (`  
    `CFirmwareDestinationNet destination,`  
    `uint32_t frequency )`

sets the base advertise frequency

**Parameters**

<i>destination</i>	the destination to set
<i>frequency</i>	the frequency

**11.96.3.12 SetTestMode()** `void SetTestMode (`  
    `uint32_t mode )`

set test mode

**Parameters**

<i>mode</i>	the mode
-------------	----------

**11.96.3.13 SetWorkingFrequency()** `void SetWorkingFrequency (`  
    `uint32_t frequency )`

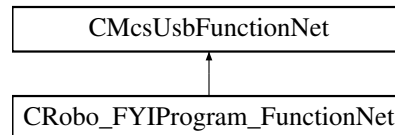
sets the working frequency

## Parameters

<i>frequency</i>	the frequency
------------------	---------------

## 11.97 CRobo\_FYIPProgram\_FunctionNet Class Reference

Inheritance diagram for CRobo\_FYIPProgram\_FunctionNet:



### Public Member Functions

- [CRobo\\_FYIPProgram\\_FunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> robo\_FY↔IProgram\_FunctionPointerContainer)
- [CRobo\\_FYIPProgram\\_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

#### 11.97.1 Constructor & Destructor Documentation

**11.97.1.1 CRobo\_FYIPProgram\_FunctionNet()** [1/2] [CRobo\\_FYIPProgram\\_FunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb,  
[CMcsUsbFunctionPointerContainer](#)<sup>^</sup> robo\_FYIPProgram\_FunctionPointerContainer )

**11.97.1.2 CRobo\_FYIPProgram\_FunctionNet()** [2/2] [CRobo\\_FYIPProgram\\_FunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb )

#### 11.97.2 Member Function Documentation

**11.97.2.1 GetLength()** `int GetLength (`  
`unsigned char index )`

**11.97.2.2 GetState()** `int GetState ( )`

**11.97.2.3 GetValve1()** `unsigned int GetValve1 (`  
`unsigned char index )`

**11.97.2.4 GetValve2()** `unsigned int GetValve2 (`  
`unsigned char index )`

**11.97.2.5 SetLength()** `void SetLength (`  
`unsigned char index,`  
`int length )`

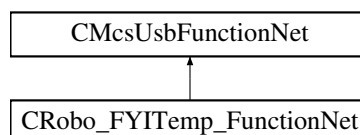
**11.97.2.6 SetValve1()** `void SetValve1 (`  
`unsigned char index,`  
`unsigned int valve1 )`

**11.97.2.7 SetValve2()** `void SetValve2 (`  
`unsigned char index,`  
`unsigned int valve2 )`

**11.97.2.8 Start()** `void Start ( )`

## 11.98 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

### 11.98.1 Constructor & Destructor Documentation

**11.98.1.1 CRobo\_FYITemp\_FunctionNet()** [CRobo\\_FYITemp\\_FunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb )

### 11.98.2 Member Function Documentation

**11.98.2.1 GetICoeff()** int GetICoeff (  
unsigned char index )

**11.98.2.2 GetMaxPower()** int GetMaxPower (  
unsigned char index )

**11.98.2.3 GetPCoeff()** int GetPCoeff (  
unsigned char index )

**11.98.2.4 GetRegulatorOnOff()** int GetRegulatorOnOff (  
unsigned char index )

**11.98.2.5 GetSollTemp()** `int GetSollTemp (`  
`unsigned char index )`

**11.98.2.6 SetICoeff()** `void SetICoeff (`  
`unsigned char index,`  
`int icoeff )`

**11.98.2.7 SetMaxPower()** `void SetMaxPower (`  
`unsigned char index,`  
`int power )`

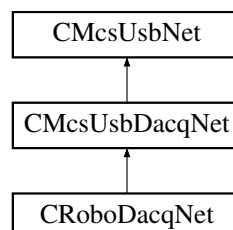
**11.98.2.8 SetPCoeff()** `void SetPCoeff (`  
`unsigned char index,`  
`int pcoeff )`

**11.98.2.9 SetRegulatorOnOff()** `void SetRegulatorOnOff (`  
`unsigned char index,`  
`int onoff )`

**11.98.2.10 SetSollTemp()** `void SetSollTemp (`  
`unsigned char index,`  
`int temp )`

## 11.99 CRoboDacqNet Class Reference

Inheritance diagram for CRoboDacqNet:



## Public Member Functions

- [CRoboDacqNet](#) (void)
- [CRoboDacqNet](#) ([CRoboDeviceNet](#)^ 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 [SetUVOffset](#) (int UVOffset)
- 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 [GetUVOffset](#) ()
- 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)
- void [SetNoFilterCoeffs](#) (int index)
- 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

### 11.99.1 Constructor & Destructor Documentation

**11.99.1.1 CRoboDacqNet()** [1/2] `CRoboDacqNet ( void )`

**11.99.1.2 CRoboDacqNet()** [2/2] `CRoboDacqNet ( CRoboDeviceNet^ robodevice )`

### 11.99.2 Member Function Documentation

**11.99.2.1 CancelTableLoop()** `void CancelTableLoop ( )`

**11.99.2.2 CancelTableLoopAndStopTable()** `void CancelTableLoopAndStopTable ( )`

**11.99.2.3 ClampAmpRestart()** `void ClampAmpRestart ( )`

**11.99.2.4 DoRamp()** `void DoRamp ( int startValue, int endValue, int duration, int mode )`

**11.99.2.5 Emu\_GetCellCapacity()** `int Emu_GetCellCapacity ( )`

**11.99.2.6 Emu\_GetCellPotential()** `int Emu_GetCellPotential ( )`

**11.99.2.7 Emu\_GetCellResists()** `int Emu_GetCellResists ( )`

**11.99.2.8 Emu\_GetElectrodeResists()** `int Emu_GetElectrodeResists ( )`

**11.99.2.9 Emu\_GetNoise()** `int Emu_GetNoise ( )`

**11.99.2.10 Emu\_SetCellCapacity()** `void Emu_SetCellCapacity (`  
`int emuCellCapacity )`

**11.99.2.11 Emu\_SetCellPotential()** `void Emu_SetCellPotential (`  
`int emuCellPotential )`

**11.99.2.12 Emu\_SetCellResists()** `void Emu_SetCellResists (`  
`int emuCellResist )`

**11.99.2.13 Emu\_SetElectrodeResists()** `void Emu_SetElectrodeResists (`  
`int emuElectrodeResist )`

**11.99.2.14 Emu\_SetNoise()** `void Emu_SetNoise (`  
`int emuNoise )`

**11.99.2.15 GetAllDigout()** `uint32_t GetAllDigout ( )`

**11.99.2.16 GetCapacityC()** `int GetCapacityC ( )`

**11.99.2.17 GetCapacityV()** `int GetCapacityV ( )`

**11.99.2.18 GetCapacityX()** `int GetCapacityX ( )`

**11.99.2.19 GetClampAmpSerialNumber()** `unsigned int GetClampAmpSerialNumber ( )`

**11.99.2.20 GetCommand()** `int GetCommand (`  
`unsigned char command )`

**11.99.2.21 GetConfigurationBit()** `bool GetConfigurationBit (`  
`unsigned short bit )`

**11.99.2.22 GetConfigurationBitAxc()** `bool GetConfigurationBitAxc ( )`

**11.99.2.23 GetConfigurationBitBlu\_Led()** `bool GetConfigurationBitBlu_Led ( )`

**11.99.2.24 GetConfigurationBitBlu\_LedToggleFast()** `bool GetConfigurationBitBlu_LedToggleFast ( )`

**11.99.2.25 GetConfigurationBitBlu\_LedToggleSlow()** `bool GetConfigurationBitBlu_LedToggleSlow ( )`

**11.99.2.26 GetConfigurationBitCC\_Gen()** `bool GetConfigurationBitCC_Gen ( )`

**11.99.2.27 GetConfigurationBitCV\_Gen()** `bool GetConfigurationBitCV_Gen ( )`

**11.99.2.28 GetConfigurationBitRC\_Gen()** `bool GetConfigurationBitRC_Gen ( )`

**11.99.2.29 GetConfigurationBitRed\_Led()** `bool GetConfigurationBitRed_Led ( )`

**11.99.2.30 GetConfigurationBitRed\_LedSaturation()** `bool GetConfigurationBitRed_LedSaturation ( )`

**11.99.2.31 GetConfigurationBitRed\_LedToggleFast()** `bool GetConfigurationBitRed_LedToggleFast ( )`

**11.99.2.32 GetConfigurationBitRed\_LedToggleSlow()** `bool GetConfigurationBitRed_LedToggleSlow ( )`

**11.99.2.33 GetConfigurationBitRelais()** `bool GetConfigurationBitRelais ( )`

**11.99.2.34 GetConfigurationBitRV\_Gen()** `bool GetConfigurationBitRV_Gen ( )`

**11.99.2.35 GetConfigurationBits()** `unsigned int GetConfigurationBits ( )`

**11.99.2.36 GetConfigurationBitStream()** `bool GetConfigurationBitStream ( )`

**11.99.2.37 GetConfigurationBitSupply()** `bool GetConfigurationBitSupply ( )`

**11.99.2.38 GetCrossTalkOffset()** `int GetCrossTalkOffset ( )`

**11.99.2.39 GetCrossTalkOptimum()** `int GetCrossTalkOptimum ( )`

**11.99.2.40 GetDigout()** `bool GetDigout (`  
`uint16_t index )`

**11.99.2.41 GetDisplayText()** `String ^ GetDisplayText (`  
`int index )`

**11.99.2.42 GetDownsampleFactor()** `int GetDownsampleFactor (`  
`int index )`

**11.99.2.43 GetFilter()** `int GetFilter ( )`

**11.99.2.44 GetFilterCoeffs()** `array<int> ^ GetFilterCoeffs (`  
`int index )`

**11.99.2.45 GetIC()** `int GetIC ( )`

**11.99.2.46 GetIClamp()** `int GetIClamp ( )`

**11.99.2.47 GetICOffset()** `int GetICOffset ( )`

**11.99.2.48 GetIGain()** `int GetIGain ( )`

**11.99.2.49 GetNIC\_MS()** `int GetNIC_MS ( )`

**11.99.2.50 GetNUC\_MS()** `int GetNUC_MS ( )`

**11.99.2.51 GetNUV\_MS()** `int GetNUV_MS ( )`

**11.99.2.52 GetPGain()** `int GetPGain ( )`

**11.99.2.53 GetRecordingNumber()** `unsigned int GetRecordingNumber ( )`

**11.99.2.54 GetResistanceC()** `int GetResistanceC ( )`

**11.99.2.55 GetResistanceV()** `int GetResistanceV ( )`

**11.99.2.56 GetScreen()** `int GetScreen ( )`

**11.99.2.57 GetSimulation()** `unsigned int GetSimulation ( )`

**11.99.2.58 GetUC()** `int GetUC ( )`

**11.99.2.59 GetUClamp()** `int GetUClamp ( )`

**11.99.2.60 GetUCOffset()** `int GetUCOffset ( )`

**11.99.2.61 GetUpdateDisplay()** `int GetUpdateDisplay ( )`

**11.99.2.62 GetUV()** `int GetUV ( )`

**11.99.2.63 GetUVOffset()** `int GetUVOffset ( )`

**11.99.2.64 GetXGain()** `int GetXGain ( )`

**11.99.2.65 RunTable()** [1/2] `void RunTable ( )`

**11.99.2.66 RunTable()** [2/2] `void RunTable (`  
`int timeout )`

**11.99.2.67 SetAllDigout()** `void SetAllDigout (`  
`uint32_t value )`

**11.99.2.68 SetCommand()** `void SetCommand (`  
`unsigned char command,`  
`int value )`

**11.99.2.69 SetConfigurationBit()** `void SetConfigurationBit (`  
`unsigned short bit,`  
`bool value )`

**11.99.2.70 SetConfigurationBitAxc()** `void SetConfigurationBitAxc (`  
`bool value )`



**11.99.2.71 SetConfigurationBitBlu\_Led()** void SetConfigurationBitBlu\_Led (  
bool value )

**11.99.2.72 SetConfigurationBitBlu\_LedToggleFast()** void SetConfigurationBitBlu\_LedToggleFast (  
bool value )

**11.99.2.73 SetConfigurationBitBlu\_LedToggleSlow()** void SetConfigurationBitBlu\_LedToggleSlow (  
bool value )

**11.99.2.74 SetConfigurationBitCC\_Gen()** void SetConfigurationBitCC\_Gen (  
bool value )

**11.99.2.75 SetConfigurationBitCV\_Gen()** void SetConfigurationBitCV\_Gen (  
bool value )

**11.99.2.76 SetConfigurationBitRC\_Gen()** void SetConfigurationBitRC\_Gen (  
bool value )

**11.99.2.77 SetConfigurationBitRed\_Led()** void SetConfigurationBitRed\_Led (  
bool value )

**11.99.2.78 SetConfigurationBitRed\_LedSaturation()** void SetConfigurationBitRed\_LedSaturation (  
bool value )

**11.99.2.79 SetConfigurationBitRed\_LedToggleFast()** void SetConfigurationBitRed\_LedToggleFast (  
bool value )

**11.99.2.80 SetConfigurationBitRed\_LedToggleSlow()** void SetConfigurationBitRed\_LedToggleSlow (  
bool value )

**11.99.2.81 SetConfigurationBitRelais()** void SetConfigurationBitRelais (  
    bool *value* )

**11.99.2.82 SetConfigurationBitRV\_Gen()** void SetConfigurationBitRV\_Gen (  
    bool *value* )

**11.99.2.83 SetConfigurationBitStream()** void SetConfigurationBitStream (  
    bool *value* )

**11.99.2.84 SetConfigurationBitSupply()** void SetConfigurationBitSupply (  
    bool *value* )

**11.99.2.85 SetCrossTalkOffset()** void SetCrossTalkOffset (  
    int *CrossTalk* )

**11.99.2.86 SetCrossTalkOptimum()** void SetCrossTalkOptimum (  
    int *cxOptimum* )

**11.99.2.87 SetDigout()** void SetDigout (  
    uint16\_t *index*,  
    bool *enable* )

**11.99.2.88 SetDisplayText()** void SetDisplayText (  
    int *index*,  
    String^ *displayText* )

**11.99.2.89 SetDownsampleFactor()** void SetDownsampleFactor (  
    int *index*,  
    int *downsample\_factor* )

**11.99.2.90 SetFilter()** void SetFilter (   
int *filter* )

**11.99.2.91 SetFilterCoeffs()** void SetFilterCoeffs (   
int *index*,   
array< int >^ *coeffs* )

**11.99.2.92 SetIClamp()** void SetIClamp (   
int *iClamp* )

**11.99.2.93 SetICOffset()** void SetICOffset (   
int *ICOffset* )

**11.99.2.94 SetIGain()** void SetIGain (   
int *iGain* )

**11.99.2.95 SetNoFilterCoeffs()** void SetNoFilterCoeffs (   
int *index* )

**11.99.2.96 SetPGain()** void SetPGain (   
int *pGain* )

**11.99.2.97 SetRecordingNumber()** void SetRecordingNumber (   
unsigned int *recordingNumber* )

**11.99.2.98 SetScreen()** void SetScreen (   
int *screen* )

**11.99.2.99 SetSimulation()** void SetSimulation (   
 unsigned int *enable* )

**11.99.2.100 SetUClamp()** void SetUClamp (   
 int *uClamp* )

**11.99.2.101 SetUCOffset()** void SetUCOffset (   
 int *UCOffset* )

**11.99.2.102 SetUVOffset()** void SetUVOffset (   
 int *UVOffset* )

**11.99.2.103 SetXGain()** void SetXGain (   
 int *xGain* )

**11.99.2.104 StopTable()** [1/2] void StopTable ( )

**11.99.2.105 StopTable()** [2/2] void StopTable (   
 int *timeout* )

**11.99.2.106 Table\_Wait()** void Table\_Wait (   
 unsigned int *tableWait* )

**11.99.2.107 TableDefBegin()** void TableDefBegin ( )

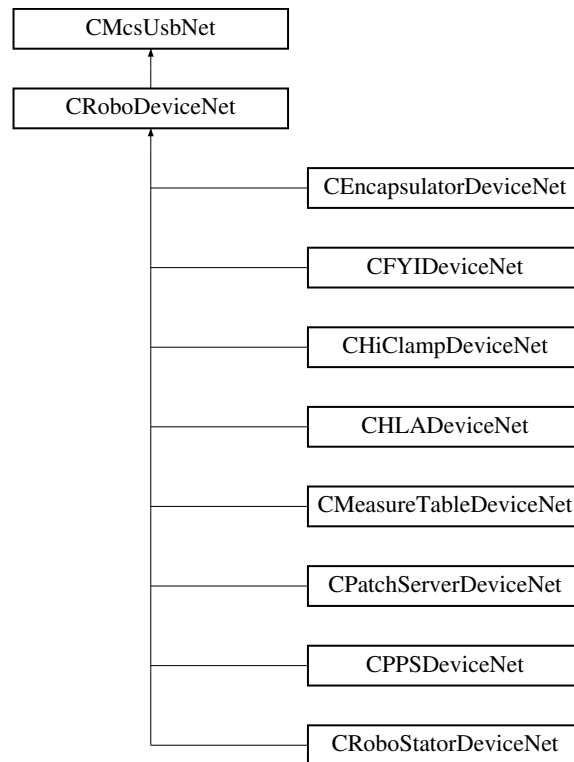
**11.99.2.108 TableDefEnd()** void TableDefEnd ( )

**11.99.2.109 UpdateDisplay()** `void UpdateDisplay ( )`

## 11.100 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****11.100.1 Detailed Description**

[CRoboDeviceNet](#) is the base class for all Robo platform based devices

**11.100.2 Constructor & Destructor Documentation**

**11.100.2.1 CRoboDeviceNet()** [CRoboDeviceNet](#) (  
void )

**11.100.2.2 ~CRoboDeviceNet()** [~CRoboDeviceNet](#) (  
void )

**11.100.3 Member Function Documentation**

**11.100.3.1 CancelPoolLoop()** void CancelPoolLoop ( )

**11.100.3.2 CancelPoolLoopAndStopMovement()** void CancelPoolLoopAndStopMovement ( )

**11.100.3.3 FindReference()** [1/2] void FindReference (  
unsigned char *busaddress*,  
char *axes* )

**11.100.3.4 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)

**11.100.3.5 GetAirpressure()** unsigned int GetAirpressure ( )

**11.100.3.6 GetAirpressureLimit()** void GetAirpressureLimit (   
[System::Runtime::InteropServices::Out] unsigned int% *lowerpressure*,   
[System::Runtime::InteropServices::Out] unsigned int% *upperpressure* )

**11.100.3.7 GetAirValve()** unsigned int GetAirValve ( )

**11.100.3.8 GetCurrentAirvalve()** unsigned int GetCurrentAirvalve ( )

**11.100.3.9 GetCurrentAirvalveLimit()** void GetCurrentAirvalveLimit (   
[System::Runtime::InteropServices::Out] unsigned int% *lowercurrent*,   
[System::Runtime::InteropServices::Out] unsigned int% *uppercurrent* )

**11.100.3.10 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

**11.100.3.11 GetErrorAirpressure()** unsigned int GetErrorAirpressure ( )

**11.100.3.12 GetErrorCurrentAirvalve()** unsigned int GetErrorCurrentAirvalve ( )

**11.100.3.13 GetErrorVoltage12V()** unsigned int GetErrorVoltage12V ( )

**11.100.3.14 GetErrorVoltage5V()** unsigned int GetErrorVoltage5V ( )

**11.100.3.15 GetErrorVoltageAirvalve()** unsigned int GetErrorVoltageAirvalve ( )

**11.100.3.16 GetErrorVoltageRs485A()** unsigned int GetErrorVoltageRs485A ( )

**11.100.3.17 GetErrorVoltageRs485B()** unsigned int GetErrorVoltageRs485B ( )

**11.100.3.18 GetErrorVoltageValves()** unsigned int GetErrorVoltageValves ( )

**11.100.3.19 GetInMovement()** bool GetInMovement ( )

Low level command, gets the internal state "In Movement"

**11.100.3.20 GetMinPressure()** int GetMinPressure ( )

**11.100.3.21 GetMovementError()** uint32\_t GetMovementError ( )

Low level command, gets the error of the last movement end

**11.100.3.22 GetVoltage12V()** unsigned int GetVoltage12V ( )**11.100.3.23 GetVoltage12VLimit()** void GetVoltage12VLimit (   
 [System::Runtime::InteropServices::Out] unsigned int% *lowervoltage*,   
 [System::Runtime::InteropServices::Out] unsigned int% *uppervoltage* )**11.100.3.24 GetVoltage5V()** unsigned int GetVoltage5V ( )**11.100.3.25 GetVoltage5VLimit()** void GetVoltage5VLimit (   
 [System::Runtime::InteropServices::Out] unsigned int% *lowervoltage*,   
 [System::Runtime::InteropServices::Out] unsigned int% *uppervoltage* )**11.100.3.26 GetVoltageAirvalve()** unsigned int GetVoltageAirvalve ( )**11.100.3.27 GetVoltageAirvalveLimit()** void GetVoltageAirvalveLimit (   
 [System::Runtime::InteropServices::Out] unsigned int% *lowervoltage*,   
 [System::Runtime::InteropServices::Out] unsigned int% *uppervoltage* )**11.100.3.28 GetVoltageRs485A()** unsigned int GetVoltageRs485A ( )**11.100.3.29 GetVoltageRs485ALimit()** void GetVoltageRs485ALimit (   
 [System::Runtime::InteropServices::Out] unsigned int% *lowervoltage*,   
 [System::Runtime::InteropServices::Out] unsigned int% *uppervoltage* )

**11.100.3.30 GetVoltageRs485B()** unsigned int GetVoltageRs485B ( )

**11.100.3.31 GetVoltageRs485BLimit()** void GetVoltageRs485BLimit (   
 [System::Runtime::InteropServices::Out] unsigned int% *lowervoltage*,   
 [System::Runtime::InteropServices::Out] unsigned int% *uppervoltage* )

**11.100.3.32 GetVoltageValves()** unsigned int GetVoltageValves ( )

**11.100.3.33 GetVoltageValvesLimit()** void GetVoltageValvesLimit (   
 [System::Runtime::InteropServices::Out] unsigned int% *lowervoltage*,   
 [System::Runtime::InteropServices::Out] unsigned int% *uppervoltage* )

**11.100.3.34 MoveAbs() [1/2]** void MoveAbs (   
 unsigned char *busaddress*,   
 char *axes*,   
 int *x*,   
 int *y* )

**11.100.3.35 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)

**11.100.3.36 SetAirpressureLimit()** void SetAirpressureLimit (

```
unsigned int lowerpressure,  
unsigned int upperpressure )
```

**11.100.3.37 SetAirValve()** void SetAirValve (   
unsigned int *onoff* )

**11.100.3.38 SetCurrentAirvalveLimit()** void SetCurrentAirvalveLimit (   
unsigned int *lowercurrent*,  
unsigned int *uppercurrent* )

**11.100.3.39 SetCurrentAndAir() [1/2]** void SetCurrentAndAir (   
unsigned char *busaddress*,  
char *axes*,  
unsigned short *onoff* )

**11.100.3.40 SetCurrentAndAir() [2/2]** void SetCurrentAndAir (   
unsigned char *busaddress*,  
char *axes*,  
unsigned short *onoff*,  
int *timeout* )

**11.100.3.41 SetInMovement()** void SetInMovement ( )

Low level command, sets the internal state to "In Movement"

**11.100.3.42 SetMinPressure()** void SetMinPressure (   
int *pressure* )

**11.100.3.43 SetVoltage12VLimit()** void SetVoltage12VLimit (   
unsigned int *lowervoltage*,  
unsigned int *uppervoltage* )

- 11.100.3.44 SetVoltage5VLimit()** void SetVoltage5VLimit (   
 unsigned int *lowervoltage*,   
 unsigned int *uppervoltage* )
- 11.100.3.45 SetVoltageAirvalveLimit()** void SetVoltageAirvalveLimit (   
 unsigned int *lowervoltage*,   
 unsigned int *uppervoltage* )
- 11.100.3.46 SetVoltageRs485ALimit()** void SetVoltageRs485ALimit (   
 unsigned int *lowervoltage*,   
 unsigned int *uppervoltage* )
- 11.100.3.47 SetVoltageRs485BLimit()** void SetVoltageRs485BLimit (   
 unsigned int *lowervoltage*,   
 unsigned int *uppervoltage* )
- 11.100.3.48 SetVoltageValvesLimit()** void SetVoltageValvesLimit (   
 unsigned int *lowervoltage*,   
 unsigned int *uppervoltage* )
- 11.100.3.49 StopMovement() [1/2]** void StopMovement (   
 unsigned char *busaddress*,   
 char *axes* )
- 11.100.3.50 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)

#### 11.100.4 Member Data Documentation

**11.100.4.1 Axes\_I** `const char Axes_I = 2 [static]`

Bit pattern for i axis for axes argument

**11.100.4.2 Axes\_X** `const char Axes_X = 1 [static]`

Bit pattern for x axis for axes argument

**11.100.4.3 Axes\_Y** `const char Axes_Y = 2 [static]`

Bit pattern for y axis for axes argument

**11.100.4.4 Axes\_Z** `const char Axes_Z = 1 [static]`

Bit pattern for z axis for axes argument

**11.100.4.5 Axis\_I** `const byte Axis_I = 1 [static]`

Axis number of i for axis argument

**11.100.4.6 Axis\_X** `const byte Axis_X = 0 [static]`

Axis number of x for axis argument

**11.100.4.7 Axis\_Y** `const byte Axis_Y = 1 [static]`

Axis number of y for axis argument

**11.100.4.8 Axis\_Z** `const byte Axis_Z = 0 [static]`

Axis number of z for axis argument

**11.100.4.9 McsBus\_XY** `const byte McsBus_XY = 1 [static]`

McsBus address for the xy-plane

**11.100.4.10 McsBus\_ZI** `const byte McsBus_ZI = 2 [static]`

McsBus address for the z and i axes

**11.100.4.11 RoboError\_AnotherMaster** `const uint32_t RoboError_AnotherMaster = ( (0xA0110000L) | 0x0005 ) [static]`

**11.100.4.12 RoboError\_Base** `const uint32_t RoboError_Base = (0xA0110000L) [static]`

**11.100.4.13 RoboError\_CannotEscapeEndSwitch** `const uint32_t RoboError_CannotEscapeEndSwitch = ( (0xA0110000L) | 0x0009 ) [static]`

**11.100.4.14 RoboError\_CommandAlreadyInProgress** `const uint32_t RoboError_CommandAlreadyInProgress = ( (0xA0110000L) | 0x000A ) [static]`

**11.100.4.15 RoboError\_CommandNotPossible** `const uint32_t RoboError_CommandNotPossible = ( (0xA0110000L) | 0x0014 ) [static]`

**11.100.4.16 RoboError\_CommunicationTimeout** `const uint32_t RoboError_CommunicationTimeout = ( (0xA0110000L) | 0x0004 ) [static]`



**11.100.4.17 RoboError\_DacqNotReady** `const uint32_t RoboError_DacqNotReady = ( (0xA0110000L) | 0x0015 ) [static]`

**11.100.4.18 RoboError\_DLLMovementTimeout** `const uint32_t RoboError_DLLMovementTimeout = ( (0xA0110000L) | 0x1001 ) [static]`

**11.100.4.19 RoboError\_FindReferenceMethod** `const uint32_t RoboError_FindReferenceMethod = ( (0xA0110000L) | 0x0006 ) [static]`

**11.100.4.20 RoboError\_GilsonCommandPending** `const uint32_t RoboError_GilsonCommandPending = ( (0xA0110000L) | 0x0011 ) [static]`

**11.100.4.21 RoboError\_GilsonTimeout** `const uint32_t RoboError_GilsonTimeout = ( (0xA0110000L) | 0x000F ) [static]`

**11.100.4.22 RoboError\_GilsonWrondID** `const uint32_t RoboError_GilsonWrondID = ( (0xA0110000L) | 0x0010 ) [static]`

**11.100.4.23 RoboError\_McsBus\_UnknownCommand** `const uint32_t RoboError_McsBus_UnknownCommand = ( (0xA0110000L) | 0x003F ) [static]`

**11.100.4.24 RoboError\_NoEndSwitch** `const uint32_t RoboError_NoEndSwitch = ( (0xA0110000L) | 0x0008 ) [static]`

**11.100.4.25 RoboError\_NoMoreData** `const uint32_t RoboError_NoMoreData = ( (0xA0110000L) | 0x0016 ) [static]`

**11.100.4.26 RoboError\_NoReference** `const uint32_t RoboError_NoReference = ( (0xA0110000L) | 0x000B ) [static]`

**11.100.4.27 RoboError\_NoSpeedOrAcceleration** `const uint32_t RoboError_NoSpeedOrAcceleration = ( (0xA0110000L) | 0x0007 ) [static]`

**11.100.4.28 RoboError\_OverPressure** `const uint32_t RoboError_OverPressure = ( (0xA0110000L) | 0x000C ) [static]`

**11.100.4.29 RoboError\_ParameterNotAllowed** `const uint32_t RoboError_ParameterNotAllowed = ( (0xA0110000L) | 0x0012 ) [static]`

**11.100.4.30 RoboError\_PeristalticTimeout** `const uint32_t RoboError_PeristalticTimeout = ( (0xA0110000L) | 0x000E ) [static]`

**11.100.4.31 RoboError\_Phase0OutOfRange** `const uint32_t RoboError_Phase0OutOfRange = ( (0xA0110000L) | 0x000D ) [static]`

**11.100.4.32 RoboError\_PollLoopCanceled** `const uint32_t RoboError_PollLoopCanceled = ( (0xA0110000L) | 0x1002 ) [static]`

**11.100.4.33 RoboError\_PollLoopCanceledAndStopMovement** `const uint32_t RoboError_PollLoopCanceledAndStopMovement = ( (0xA0110000L) | 0x1003 ) [static]`

**11.100.4.34 RoboError\_Pressure** `const uint32_t RoboError_Pressure = ( (0xA0110000L) | 0x0002 ) [static]`

**11.100.4.35 RoboError\_RangeExceeded** `const uint32_t RoboError_RangeExceeded = ( (0xA0110000L) | 0x0003 ) [static]`

**11.100.4.36 RoboError\_StateChangeNotPossible** `const uint32_t RoboError_StateChangeNotPossible = ( (0xA0110000L) | 0x0013 ) [static]`

**11.100.4.37 RoboError\_Timeout** `const uint32_t RoboError_Timeout = ( (0xA0110000L) | 0x0001 )`  
[static]

**11.100.4.38 RoboError\_UnknownCommand** `const uint32_t RoboError_UnknownCommand = ( (0xA0110000L) )` [static]

## 11.100.5 Property Documentation

**11.100.5.1 McsBus** `CMcsBusNet^ McsBus` [get]

**11.100.5.2 McsBus\_MotorControl** `CMcsBus_MotorControlNet^ McsBus_MotorControl` [get]

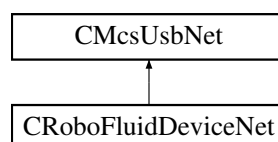
**11.100.5.3 RoboMainLowLevelCommand** `RoboMainLowLevelCommands^ RoboMainLowLevelCommand` [get]

## 11.100.6 Event Documentation

**11.100.6.1 RoboStatusEvent** `RoboStatusEventDelegate^ RoboStatusEvent`

## 11.101 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****11.101.1 Constructor & Destructor Documentation**

**11.101.1.1 CRoboFluidDeviceNet()** [CRoboFluidDeviceNet](#) (  
void )

**11.101.1.2 ~CRoboFluidDeviceNet()** [~CRoboFluidDeviceNet](#) (  
void )

**11.101.2 Member Function Documentation**

**11.101.2.1 CloseAllValves()** `void CloseAllValves ( )`

**11.101.2.2 GetPumpSpeed()** `short GetPumpSpeed (`  
`int index )`

**11.101.2.3 GetSingleValve()** `bool GetSingleValve (`  
`int valve )`

Query the state of a valve.

#### Parameters

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

#### Returns

state of the valve

**11.101.2.4 GetValve()** `int GetValve ( )`

Query the state of the valves.

#### Returns

the current state of the valves as a bit pattern.

**11.101.2.5 IsPumpMotorOn()** `bool IsPumpMotorOn (`  
`int index )`

**11.101.2.6 PumpOff()** `void PumpOff (`  
`int index )`

**11.101.2.7 PumpOn()** `void PumpOn (`  
`int index,`  
`short speed )`

**11.101.2.8 SetPumpSpeed()** `void SetPumpSpeed (`  
     `int index,`  
     `short speed )`

**11.101.2.9 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

**11.101.2.10 SetValve()** `void SetValve (`  
     `int value )`

Open or Close valves.

Parameters

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

### 11.101.3 Member Data Documentation

**11.101.3.1 m\_pMcsBus\_MotorControlNet** `CMcsBus_MotorControlNet ^ m_pMcsBus_MotorControlNet`  
 [protected]

**11.101.3.2 m\_pRoboFluidDevice** `CRoboFluidDevice* m_pRoboFluidDevice` [protected]

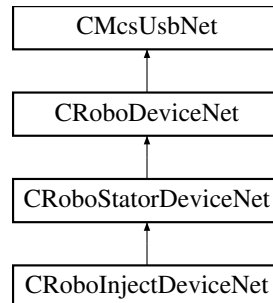
### 11.101.4 Property Documentation

**11.101.4.1 McsBus\_MotorControl** `CMcsBus_MotorControlNet^ McsBus_MotorControl` [get]

## 11.102 CRobolInjectDeviceNet Class Reference

[CRobolInjectDeviceNet](#) is the to control the MCS RobolInject device

Inheritance diagram for CRobolInjectDeviceNet:



### Public Member Functions

- [CRobolInjectDeviceNet](#) (void)

### Additional Inherited Members

#### 11.102.1 Detailed Description

[CRobolInjectDeviceNet](#) is the to control the MCS RobolInject device

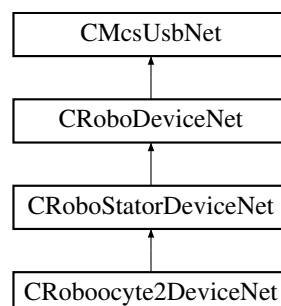
#### 11.102.2 Constructor & Destructor Documentation

**11.102.2.1 CRobolInjectDeviceNet()** [CRobolInjectDeviceNet](#) (  
void )

## 11.103 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

### 11.103.1 Detailed Description

[CRobocyte2DeviceNet](#) is the class to control the MCS Robocyte2 device

### 11.103.2 Constructor & Destructor Documentation

**11.103.2.1 CRobocyte2DeviceNet()** [CRobocyte2DeviceNet](#) (  
void )

### 11.103.3 Member Function Documentation

**11.103.3.1 GetAxisLED()** bool [GetAxisLED](#) ( )

**11.103.3.2 GetGilsonDevice()** [CGilsonDeviceNet](#) ^ [GetGilsonDevice](#) ( )

**11.103.3.3 GetMcsBus\_Extension()** [CMcsBus\\_ExtensionNet](#) ^ [GetMcsBus\\_Extension](#) ( )

**11.103.3.4 GetRoboDacq()** [CRoboDacqNet](#) ^ [GetRoboDacq](#) ( )

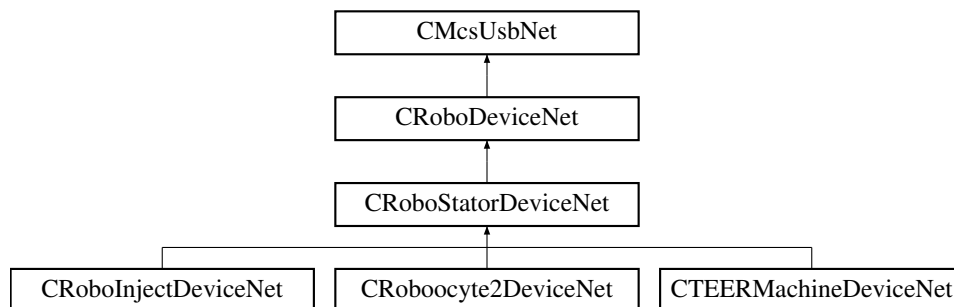


**11.103.3.5 GetRoboFluidDevice()** [CRoboFluidDeviceNet](#) ^ GetRoboFluidDevice ( )

**11.103.3.6 SetAxisLED()** void SetAxisLED (   
 bool *onoff* )

## 11.104 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

### 11.104.1 Constructor & Destructor Documentation

**11.104.1.1 CRoboStatorDeviceNet()** [CRoboStatorDeviceNet](#) (   
 void )

### 11.104.2 Member Function Documentation

**11.104.2.1 FindReferenceI()** [1/2] void FindReferenceI ( )

**11.104.2.2 FindReferenceI()** [2/2] void FindReferenceI (   
 int timeout )

**11.104.2.3 FindReferenceXY()** [1/2] void FindReferenceXY ( )

**11.104.2.4 FindReferenceXY()** [2/2] void FindReferenceXY (   
int timeout )

**11.104.2.5 FindReferenceZ()** [1/2] void FindReferenceZ ( )

**11.104.2.6 FindReferenceZ()** [2/2] void FindReferenceZ (   
int timeout )

**11.104.2.7 GetCurrentPositionI()** void GetCurrentPositionI (   
[System::Runtime::InteropServices::Out] int% i )

**11.104.2.8 GetCurrentPositionXY()** void GetCurrentPositionXY (   
[System::Runtime::InteropServices::Out] int% x,   
[System::Runtime::InteropServices::Out] int% y )

**11.104.2.9 GetCurrentPositionZ()** void GetCurrentPositionZ (   
[System::Runtime::InteropServices::Out] int% z )

**11.104.2.10 HasRefI()** unsigned char HasRefI ( )

**11.104.2.11 HasRefXY()** unsigned char HasRefXY ( )

**11.104.2.12 HasRefZ()** unsigned char HasRefZ ( )

**11.104.2.13 MoveAbsI()** [1/2] void MoveAbsI (  
int *i* )

**11.104.2.14 MoveAbsI()** [2/2] void MoveAbsI (  
int *i*,  
int *timeout* )

**11.104.2.15 MoveAbsXY()** [1/2] void MoveAbsXY (  
int *x*,  
int *y* )

**11.104.2.16 MoveAbsXY()** [2/2] void MoveAbsXY (  
int *x*,  
int *y*,  
int *timeout* )

**11.104.2.17 MoveAbsZ()** [1/2] void MoveAbsZ (  
int *z* )

**11.104.2.18 MoveAbsZ()** [2/2] void MoveAbsZ (  
int *z*,  
int *timeout* )

**11.104.2.19 SetAccelerationI()** void SetAccelerationI (  
int *a* )

**11.104.2.20 SetAccelerationNativeI()** void SetAccelerationNativeI (  
int *a* )

**11.104.2.21 SetAccelerationNativeXY()** void SetAccelerationNativeXY (  
int *a* )

**11.104.2.22 SetAccelerationNativeZ()** `void SetAccelerationNativeZ (`  
`int a )`

**11.104.2.23 SetAccelerationXY()** `void SetAccelerationXY (`  
`int a )`

**11.104.2.24 SetAccelerationZ()** `void SetAccelerationZ (`  
`int a )`

**11.104.2.25 SetCurrentAndAirXY() [1/2]** `void SetCurrentAndAirXY (`  
`unsigned short onoff )`

**11.104.2.26 SetCurrentAndAirXY() [2/2]** `void SetCurrentAndAirXY (`  
`unsigned short onoff,`  
`int timeout )`

**11.104.2.27 SetSpeedI()** `void SetSpeedI (`  
`int v )`

**11.104.2.28 SetSpeedNativeI()** `void SetSpeedNativeI (`  
`int v )`

**11.104.2.29 SetSpeedNativeXY()** `void SetSpeedNativeXY (`  
`int v )`

**11.104.2.30 SetSpeedNativeZ()** `void SetSpeedNativeZ (`  
`int v )`

**11.104.2.31 SetSpeedXY()** void SetSpeedXY (  
int v )

**11.104.2.32 SetSpeedZ()** void SetSpeedZ (  
int v )

**11.104.2.33 SetVelocityI()** void SetVelocityI (  
int v )

**11.104.2.34 SetVelocityXY()** void SetVelocityXY (  
int v )

**11.104.2.35 SetVelocityZ()** void SetVelocityZ (  
int v )

**11.104.2.36 StopMovementI()** [1/2] void StopMovementI ( )

**11.104.2.37 StopMovementI()** [2/2] void StopMovementI (  
int timeout )

**11.104.2.38 StopMovementXY()** [1/2] void StopMovementXY ( )

**11.104.2.39 StopMovementXY()** [2/2] void StopMovementXY (  
int timeout )

**11.104.2.40 StopMovementZ()** [1/2] void StopMovementZ ( )

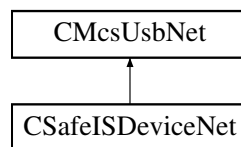
**11.104.2.41 StopMovementZ()** [2/2] void StopMovementZ (  
int timeout )

### 11.104.3 Property Documentation

**11.104.3.1 RoboMainStatorLowLevelCommand** [RoboMainStatorLowLevelCommands](#)<sup>^</sup> RoboMainStator↔  
LowLevelCommand [get]

## 11.105 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

### 11.105.1 Detailed Description

### 11.105.2 Constructor & Destructor Documentation

#### 11.105.2.1 CSafeISDeviceNet() [CSafeISDeviceNet](#) ( void )

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

#### 11.105.2.2 ~CSafeISDeviceNet() [~CSafeISDeviceNet](#) ( void )

Releases unmanaged resources and performs other cleanup operations before the [CSafeISDeviceNet](#) is reclaimed by garbage collection.

### 11.105.3 Member Function Documentation

#### 11.105.3.1 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
-----------------	--

#### 11.105.3.2 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.
------------------	----------------------------------



**11.105.3.3 SetDacMode()** `void SetDacMode (`  
`unsigned char mode )`

Sets the DAC mode.

Parameters

<i>mode</i>	The mode: 0 = Impedance ; 1 = Amperometry
-------------	---

**11.105.3.4 SetDacPeriode()** `void SetDacPeriode (`  
`unsigned int periode )`

Sets the DAC periode.

Parameters

<i>periode</i>	The periode in units of 10µs.
----------------	-------------------------------

**11.105.3.5 SetDacPulseform()** `void SetDacPulseform (`  
`array< short >^ pulseform )`

Sets the DAC pulseform.

Parameters

<i>pulseform</i>	The pulseform.
------------------	----------------

**11.105.3.6 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
-----------------	--

## 11.105.4 Property Documentation

**11.105.4.1 DacqDevice** `CMcsUsbDacqNet^ DacqDevice [get]`

Gets the `CMcsUsbDacqNet`. Use this to control the data acquisition.

**11.105.4.2 FluidControlDevice** `CFluidControlDeviceNet^ FluidControlDevice [get]`

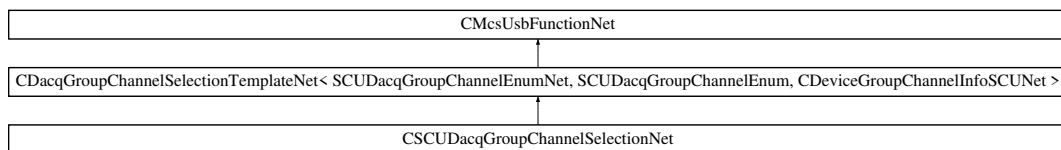
Gets the `CFluidControlDeviceNet`. Use this to control the valves. Only `SetSingleValve` is implemented for `CSafeISDeviceNet`.

**11.105.4.3 RoboDevice** `CRoboDeviceNet^ RoboDevice [get]`

Gets the `CRoboDeviceNet`. Use this to control the syringe.

**11.106 CSCUDacqGroupChannelSelectionNet Class Reference**

Inheritance diagram for `CSCUDacqGroupChannelSelectionNet`:

**Public Member Functions**

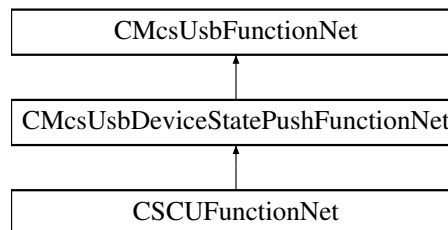
- `CSCUDacqGroupChannelSelectionNet (CMcsUsbNet^ mcsusb)`

**Additional Inherited Members****11.106.1 Constructor & Destructor Documentation****11.106.1.1 CSCUDacqGroupChannelSelectionNet()** `CSCUDacqGroupChannelSelectionNet ( CMcsUsbNet^ mcsusb )`

## 11.107 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 [GetHeadstagePowerStateAtStart](#) (uint32\_t Headstage)  
*Gets the Power Status at SCU Power on of a given headstage.*
- void [SetHeadstagePowerStateAtStart](#) (uint32\_t Headstage, bool Powerstatus)  
*Sets the Power Status at SCU Power on 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

### 11.107.1 Detailed Description

[CSCUFunctionNet](#) is the class to control the SCU device

### 11.107.2 Constructor & Destructor Documentation

**11.107.2.1 [CSCUFunctionNet\(\)](#) [1/2]** [CSCUFunctionNet](#) (   
 [CMcsUsbNet](#) ^ *mcsusb*,   
 [CMcsUsbFunctionPointerContainer](#) ^ *pSCUFunctionPointerContainer* )

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

**11.107.2.2 [CSCUFunctionNet\(\)](#) [2/2]** [CSCUFunctionNet](#) (   
 [CMcsUsbNet](#) ^ *mcsusb* )

**11.107.2.3 [~CSCUFunctionNet\(\)](#)** virtual [~CSCUFunctionNet](#) ( ) [virtual]

**11.107.2.4 ["!CSCUFunctionNet\(\)](#)** [!CSCUFunctionNet](#) ( )

### 11.107.3 Member Function Documentation

**11.107.3.1 [AutomaticAnalogOut\(\)](#)** void AutomaticAnalogOut (   
 bool *automatic* )

Sets automatic source channel selection

## Parameters

<i>automatic</i>	Automatic
------------------	-----------

**11.107.3.2 EnableAnalogOut()** `void EnableAnalogOut (`  
    `bool enable )`

Enables AnalogOut globally

## Parameters

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

**11.107.3.3 GetAnalogOutADCRange()** `uint32_t GetAnalogOutADCRange ( )`

Gets the analog out ADC range

## Returns

Range

**11.107.3.4 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

**11.107.3.5 GetAnalogOutDACRange()** `AnalogOut_DAC_Range_EnumNet GetAnalogOutDACRange ( )`

Gets the analog out DAC range

## Returns

Range

**11.107.3.6 GetAvailableHeadstages()** `uint32_t GetAvailableHeadstages ( )`

Gets a bitmap of available headstages.

**Returns**

The bitmap of available headstages.

**11.107.3.7 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

**11.107.3.8 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

**11.107.3.9 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.

**11.107.3.10 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.

**11.107.3.11 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.

**11.107.3.12 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.

**11.107.3.13 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.

**11.107.3.14 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.

**11.107.3.15 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.

**11.107.3.16 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.

**11.107.3.17 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.

**11.107.3.18 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.

**11.107.3.19 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.

**11.107.3.20 GetHeadstagePowerStateAtStart()** `bool GetHeadstagePowerStateAtStart ( uint32_t Headstage )`

Gets the Power Status at SCU Power on of a given headstage.

**Parameters**

<i>Headstage</i>	The headstage to query.
------------------	-------------------------

**Returns**

The Power State at startup for the given headstage: bool false -> off, bool true -> on.

**11.107.3.21 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.

**11.107.3.22 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.

**11.107.3.23 GetMaxNumberOfHeadstages()** `uint32_t GetMaxNumberOfHeadstages ( )`

Gets the maximal number of headstages.

## Returns

The maximal number of headstages.

**11.107.3.24 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.

**11.107.3.25 GetReferenceElectrodeMode()** `ReferenceElectrodeModeEnumNet GetReferenceElectrode↔ Mode ( uint32_t Headstage )`

Gets the mode for the reference electrode

## Parameters

<i>Headstage</i>	The headstage number
------------------	----------------------

## Returns

The mode

**11.107.3.26 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

**11.107.3.27 HasAnalogOut()** `bool HasAnalogOut ( )`

Has AnalogOut hardware

## Returns

Enabled

**11.107.3.28 HasGalvanicIsolation()** `bool HasGalvanicIsolation ( )`

Has galvanic isolated hardware

## Returns

Enabled

**11.107.3.29 HasHSPowerSwitch()** `bool HasHSPowerSwitch ( )`

Has SCU HS power switch

**Returns**

Has Switch

**11.107.3.30 IsAnalogOutEnabled()** `bool IsAnalogOutEnabled ( )`

Is AnalogOut enabled

**Returns**

Enabled

**11.107.3.31 IsAutomaticAnalogOut()** `bool IsAutomaticAnalogOut ( )`

Is Automatic source channel selection selected

**Returns**

Automatic

**11.107.3.32 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.

**11.107.3.33 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.

**11.107.3.34 IsInDacqLegacyMode()** `bool IsInDacqLegacyMode ( )`

Is the SCU in legacy mode

## Returns

Is Enabled

**11.107.3.35 OnGetAvailableHeadstages()** `delegate void OnGetAvailableHeadstages ( uint32_t AvailableHeadstages )`**11.107.3.36 OnIsHeadstageAvailable()** `delegate void OnIsHeadstageAvailable ( uint32_t Headstage, bool available )`**11.107.3.37 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.

**11.107.3.38 SetAnalogOutADCRange()** `void SetAnalogOutADCRange ( uint32_t range )`

Sets the analog out ADC range

## Parameters

<i>range</i>	Range
--------------	-------

**11.107.3.39 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

**11.107.3.40 SetAnalogOutDACRange()** void SetAnalogOutDACRange (   
     AnalogOut\_DAC\_Range\_EnumNet *range* )

Sets the analog out DAC range

## Parameters

<i>range</i>	Range
--------------	-------

**11.107.3.41 SetDacqLegacyMode()** void SetDacqLegacyMode (   
     bool *enable* )

Enable the SCU legacy mode

## Parameters

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

**11.107.3.42 SetHeadstagePowerStateAtStart()** void SetHeadstagePowerStateAtStart (   
     uint32\_t *Headstage*,   
     bool *Powerstatus* )

Sets the Power Status at SCU Power on of a given headstage.



## Parameters

<i>Headstage</i>	The headstage number
<i>Powerstatus</i>	The Power State at startup for the given headstage: bool false -> off, bool true -> on.

**11.107.3.43 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

**11.107.3.44 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

## 11.107.4 Event Documentation

**11.107.4.1 GetAvailableHeadstagesEvent** `OnGetAvailableHeadstages^` `GetAvailableHeadstagesEvent`  
     `[add], [remove], [raise]`

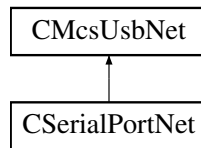
Event fires when the bitmap of available headstages has changed

**11.107.4.2 IsHeadstageAvailableEvent** `OnIsHeadstageAvailable^` `IsHeadstageAvailableEvent` `[add],`  
     `[remove], [raise]`

Event fires when 'true' if the headstage is connected for the headstage to query has changed

## 11.108 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

#### 11.108.1 Constructor & Destructor Documentation

**11.108.1.1 CSerialPortNet()** [CSerialPortNet](#) (  
void )

#### 11.108.2 Member Function Documentation

**11.108.2.1 GetBytesAvailable()** int [GetBytesAvailable](#) (  
void )

**11.108.2.2 Receive()** [1/2] array<byte> ^ [Receive](#) (  
int *length* )

**11.108.2.3 Receive()** [2/2] array<byte> ^ [Receive](#) (  
void )

**11.108.2.4 ReceiveString()** [1/2] `String ^ ReceiveString (`  
`int length )`

**11.108.2.5 ReceiveString()** [2/2] `String ^ ReceiveString (`  
`void )`

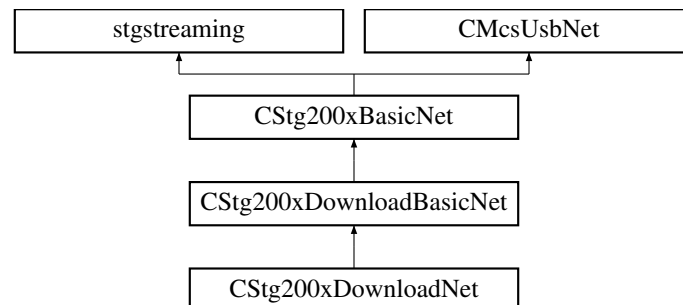
**11.108.2.6 Send()** [1/2] `void Send (`  
`array< byte >^ buffer )`

**11.108.2.7 Send()** [2/2] `void Send (`  
`String^ command )`

## 11.109 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 weather an electrode is enabled or disabled for stimulation.*
- virtual bool [GetElectrodeEnable](#) (uint32\_t Scu\_HS, uint32\_t electrode, uint32\_t listmodelIndex)
 

*Gets weather 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

### 11.109.1 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.

### 11.109.2 Constructor & Destructor Documentation

**11.109.2.1** `~CStg200xBasicNet()` `virtual ~CStg200xBasicNet ( ) [virtual]`

The destructor.

### 11.109.3 Member Function Documentation

**11.109.3.1** `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.

**11.109.3.2 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.

**11.109.3.3 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.

**11.109.3.4 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.

**11.109.3.5 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.

**11.109.3.6 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.

**11.109.3.7 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.

**11.109.3.8 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.

**11.109.3.9 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.

**11.109.3.10 GetDACResolution()** `virtual int32_t GetDACResolution ( ) [virtual]`

Gets number of bits of the DAC resolution.

**Returns**

The DAC resolution in bits.

**11.109.3.11 GetDiginValue()** `virtual uint32_t GetDiginValue ( ) [virtual]`

Gets the Value on the digital input port.

**Returns**

The current value on the digital inputs.

**11.109.3.12 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.

**11.109.3.13 GetDigoutValue()** `virtual uint32_t GetDigoutValue ( ) [virtual]`

Gets the Value on the digital output port.

**Returns**

The current value on the digital outputs.

**11.109.3.14 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.

**11.109.3.15 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.

**11.109.3.16 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.

**11.109.3.17 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.

**11.109.3.18 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.

**11.109.3.19 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.

**11.109.3.20 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.

**11.109.3.21 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.

**11.109.3.22 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.

**11.109.3.23 GetFAAmplification()** `virtual uint32_t GetFAAmplification ( ) [virtual]`

**11.109.3.24 GetHeadstage()** `virtual uint32_t GetHeadstage ( ) [virtual]`

**11.109.3.25 GetListmodeIndexRange()** `virtual void GetListmodeIndexRange ( unsigned int sideband, unsigned int & startIndex, unsigned int & endIndex, unsigned int & mode ) [virtual]`

**11.109.3.26 GetListmodeTriggerSource()** `virtual TriggerSourceEnumNet GetListmodeTriggerSource ( unsigned int sideband ) [virtual]`

**11.109.3.27 GetNumberOfAnalogChannels()** `virtual uint32_t GetNumberOfAnalogChannels ( ) [virtual]`

Gets the Number of available analog channels of the device.

**Returns**

The number of analog channels.

**11.109.3.28 GetNumberOfHWDACPaths()** `virtual uint32_t GetNumberOfHWDACPaths ( ) [virtual]`

Gets the Number of HW Stimulation DACs of the device.

**Returns**

The number of independent HW Stimulation outputs.

**11.109.3.29 GetNumberOfStimulationElectrodes()** virtual uint32\_t GetNumberOfStimulationElectrodes ( ) [virtual]

**11.109.3.30 GetNumberOfStimulationSourcesPerElectrode()** virtual uint32\_t GetNumberOfStimulationSourcesPerElectrode ( ) [virtual]

Gets the number of stimulation sources (DACs) per electrode.

#### Returns

The number of stimulation sources (DACs) per electrode.

**11.109.3.31 GetNumberOfSyncoutChannels()** virtual uint32\_t GetNumberOfSyncoutChannels ( ) [virtual]

Gets the Number of available syncout channels of the device.

#### Returns

The number of analog channels.

**11.109.3.32 GetNumberOfTriggerInputs()** virtual uint32\_t GetNumberOfTriggerInputs ( ) [virtual]

Gets the Number of trigger inputs of the device.

#### Returns

The number of trigger inputs.

**11.109.3.33 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.

**11.109.3.34 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 downloaded waveform.
<i>guid</i>	A GUID.

**11.109.3.35 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 downloaded waveform.

**11.109.3.36 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.

**11.109.3.37 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.

**11.109.3.38 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.

**11.109.3.39 GetTriggerSource()** `virtual TriggerSourceEnumNet GetTriggerSource ( unsigned int triggernum ) [virtual]`**11.109.3.40 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.

**11.109.3.41 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.

**11.109.3.42 ListModeSendStart()** virtual void ListModeSendStart (   
 unsigned int *sidebandMask* ) [virtual]

**11.109.3.43 ListModeSendStop()** virtual void ListModeSendStop (   
 unsigned int *sidebandMask* ) [virtual]

**11.109.3.44 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.
-------------------	---

**11.109.3.45 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.
-------------------	---

**11.109.3.46 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..

**11.109.3.47 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.

**11.109.3.48 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.

**11.109.3.49 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.

**11.109.3.50 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.

**11.109.3.51 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.

**11.109.3.52 SetCurrentMode()** [1/2] `virtual void SetCurrentMode ( ) [virtual]`

Sets all channels to current mode (STG3008-FA and STG400x only).

**11.109.3.53 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.
----------------	------------------------

**11.109.3.54 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.

**11.109.3.55 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.
-------------------	-------------------------

**11.109.3.56 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.
--------------------	---------------------------------------

**11.109.3.57 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).

**11.109.3.58 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).

**11.109.3.59 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).

**11.109.3.60 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).

**11.109.3.61 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.

**11.109.3.62 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.

**11.109.3.63 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.

**11.109.3.64 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.

**11.109.3.65 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.

**11.109.3.66 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.
-------------	--

**11.109.3.67 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.

**11.109.3.68 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.
-------------	--

**11.109.3.69 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.

**11.109.3.70 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.

**11.109.3.71 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.

**11.109.3.72 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.

**11.109.3.73 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.

**11.109.3.74 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.

**11.109.3.75 SetFAAmplification()** virtual void SetFAAmplification (   
 unsigned int *amplification* ) [virtual]

**11.109.3.76 SetHeadstage()** virtual void SetHeadstage (   
 unsigned int *headstage* ) [virtual]

**11.109.3.77 SetListmodeIndexRange()** virtual void SetListmodeIndexRange (   
 unsigned int *sideband*,   
 unsigned int *startIndex*,   
 unsigned int *endIndex*,   
 unsigned int *mode* ) [virtual]

**11.109.3.78 SetListmodeTriggerSource()** [1/2] virtual void SetListmodeTriggerSource (   
 unsigned int *sideband*,   
 TriggerSourceEnumNet *triggersource* ) [virtual]

**11.109.3.79 SetListmodeTriggerSource()** [2/2] virtual void SetListmodeTriggerSource (   
 unsigned int *sideband*,   
 TriggerSourceEnumNet *triggersource*,   
 int *bitnumOffset* ) [virtual]

**11.109.3.80 SetMeasurementMode()** virtual void SetMeasurementMode (   
 unsigned int *channel* ) [virtual]

Sets a channel to measurement mode (STG3008-FA).

## Parameters

<i>channel</i>	The channel to change.
----------------	------------------------

**11.109.3.81 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.
-------------	----------------------------

**11.109.3.82 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.

**11.109.3.83 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.

**11.109.3.84 SetTriggerSource()** [1/3] `virtual void SetTriggerSource (`  
`unsigned int triggernum,`



```
DigitalSource< digitalsourceenum >^ triggersource,
int bitnum_offset ) [virtual]
```

**11.109.3.85 SetTriggerSource() [2/3]** virtual void SetTriggerSource ( unsigned int *triggernum*, TriggerSourceEnumNet *triggersource* ) [virtual]

**11.109.3.86 SetTriggerSource() [3/3]** virtual void SetTriggerSource ( unsigned int *triggernum*, TriggerSourceEnumNet *triggersource*, int *bitnum\_offset* ) [virtual]

**11.109.3.87 SetVoltageMode() [1/2]** virtual void SetVoltageMode ( ) [virtual]

Sets all channels to voltage mode (STG3008-FA and STG400x only).

**11.109.3.88 SetVoltageMode() [2/2]** virtual void SetVoltageMode ( unsigned int *channel* ) [virtual]

Sets a channel to voltage mode (STG3008-FA and STG400x only).

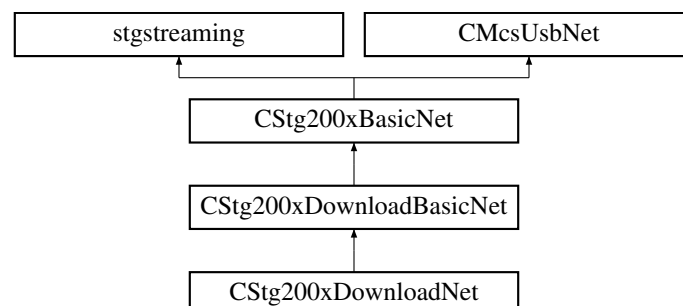
#### Parameters

<i>channel</i>	The channel to change.
----------------	------------------------

## 11.110 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

### 11.110.1 Detailed Description

[CStg200xDownloadBasicNet](#) is the base class to control the download mode of the MCS STG device.

### 11.110.2 Member Function Documentation

**11.110.2.1 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.
----------------	---------------------------------

**11.110.2.2 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.
----------------	---

**11.110.2.3 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()

#### 11.110.2.4 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.

#### 11.110.2.5 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.

#### 11.110.2.6 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.

#### 11.110.2.7 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.

**11.110.2.8 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.

**11.110.2.9 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.

**11.110.2.10 ResetStatus()** `void ResetStatus (`  
     `uint32_t triggermap )`

Reset the status flag.

## Parameters

<i>triggermap</i>	bitmap of trigger for which to reset the status.
-------------------	--

**11.110.2.11 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.

**11.110.2.12 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.

**11.110.2.13 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.

**11.110.2.14 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.

**11.110.2.15 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.

#### 11.110.2.16 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.

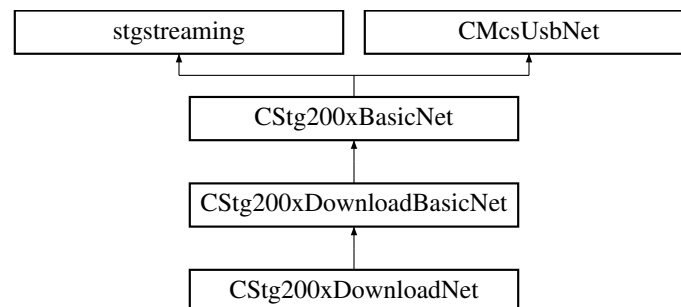
## 11.110.3 Property Documentation

11.110.3.1 Stimulus [CStimulusFunctionNet](#)<sup>^</sup> Stimulus [get]

## 11.111 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](#) (array< 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

### 11.111.1 Detailed Description

Main class for the STG download mode This class implements the STG download mode interface.

### 11.111.2 Constructor & Destructor Documentation

#### 11.111.2.1 CStg200xDownloadNet() [1/2] [CStg200xDownloadNet](#) ( )

Use this constructor if you do not want to use the status callback.

#### 11.111.2.2 CStg200xDownloadNet() [2/2] [CStg200xDownloadNet](#) ( [OnStgPollStatus](#) ^ *pollStatus* )

Use this constructor if you want to use the status callback.

#### 11.111.2.3 ~CStg200xDownloadNet() [~CStg200xDownloadNet](#) ( )

### 11.111.3 Member Function Documentation

**11.111.3.1 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.

**11.111.3.2 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.

**11.111.3.3 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.
----------------	--

**11.111.3.4 GetModuleCurrent()** `uint32_t GetModuleCurrent (`  
`unsigned int channel )`

**11.111.3.5 GetModuleTemp()** `int32_t GetModuleTemp (`  
`unsigned int channel )`

**11.111.3.6 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 $\text{nA}$ 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.

**11.111.3.7 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  $\text{nA}$ , 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 $\text{nA}$ 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.

**11.111.3.8 QueryTriggerstatus()** `StgStatusNet ^ QueryTriggerstatus ( )`**11.111.3.9 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).
---------------------	---

**11.111.3.10 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.
---------------------	---

**11.111.3.11 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.
---------------------	---

**11.111.3.12 SetOutputMap()** `void SetOutputMap (`  
`array< uint32_t >^ ChannelLayout )`

## 11.111.4 Event Documentation

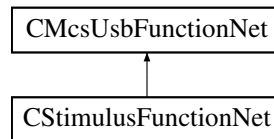
**11.111.4.1 MwPollStatusEvent** `OnMwPollStatus^ MwPollStatusEvent [add], [remove], [raise]`

**11.111.4.2 Stg200xPollStatusEvent** `OnStgPollStatus^ Stg200xPollStatusEvent [add], [remove], [raise]`



## 11.112 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

### 11.112.1 Constructor & Destructor Documentation

**11.112.1.1 CStimulusFunctionNet()** [1/2] [CStimulusFunctionNet](#) (   
[CMcsUsbNet](#)^ mcsusb,   
[CMcsUsbFunctionPointerContainer](#)^ stimulusFunctionPointerContainer )

**11.112.1.2 CStimulusFunctionNet()** [2/2] [CStimulusFunctionNet](#) (   
[CMcsUsbNet](#)^ mcsusb )

### 11.112.2 Member Function Documentation

**11.112.2.1 ClearChannel\_PrepareAndSendData()** `void ClearChannel_PrepareAndSendData (`  
     `uint32_t channel,`  
     `array< int32_t >^ amplitude,`  
     `array< uint64_t >^ duration,`  
     `STG_DestinationEnumNet destType,`  
     `bool doClear )`

**11.112.2.2 ClearChannelData()** `void ClearChannelData (`  
     `int channel )`

Delete a Stimulus Pattern from STG memory

#### Parameters

<i>channel</i>	specifies the channel to clear.
----------------	---------------------------------

**11.112.2.3 ClearMultiplexedData()** `void ClearMultiplexedData ( )`

Clears the Stimulation Memory in the STG device.

**11.112.2.4 ClearSyncData()** `void ClearSyncData (`  
     `int channel )`

Delete a Syncout Pattern from STG memory

#### Parameters

<i>channel</i>	specifies the channel to clear.
----------------	---------------------------------

**11.112.2.5 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.

#### 11.112.2.6 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.

**11.112.2.7 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.

**11.112.2.8 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.

**11.112.2.9 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.

**11.112.2.10 GetDACResolution()** `int GetDACResolution ( )`

Gets number of bits of the DAC resolution.

**Returns**

The DAC resolution in bits.

**11.112.2.11 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`

**11.112.2.12 GetNumberOfAnalogChannels()** `int GetNumberOfAnalogChannels ( )`

Get the number of STG channels.

**Returns**

The number of STG channels.

**11.112.2.13 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.

**11.112.2.14 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.

**11.112.2.15 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.

**11.112.2.16 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.

**11.112.2.17 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  $\mu$ V, 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.



**11.112.2.18 PrepareData()** `StimulusDeviceDataAndUnrolledData` ^ PrepareData (   
     int *channel*,  
     array< int32\_t >^ *amplitude*,  
     array< uint64\_t >^ *duration*,  
     STG\_DestinationEnumNet *destType* )

**11.112.2.19 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.
-------------	---------------------------

**11.112.2.20 SendPreparedData()** void SendPreparedData (   
     int *channel*,  
     `StimulusDeviceDataAndUnrolledData`^ *device\_data\_and\_unrolled*,  
     STG\_DestinationEnumNet *destType* )

**11.112.2.21 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.
-------------------	---

**11.112.2.22 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.
-------------------	---

**11.112.2.23 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..

**11.112.2.24 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.

**11.112.2.25 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.

**11.112.2.26 StartPoll()** `void StartPoll ( )`

Starts the interrupt fetching thread and delivers events

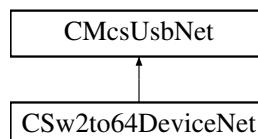
**11.112.2.27 StopPoll()** `void StopPoll ( )`

Stops the interrupt fetching thread and delivers events

**11.112.3 Event Documentation****11.112.3.1 PollStatusEvent** `OnStgPollStatus^ PollStatusEvent`**11.113 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****11.113.1 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.

**11.113.2 Constructor & Destructor Documentation**

**11.113.2.1 CSw2to64DeviceNet()** [CSw2to64DeviceNet](#) ( )

**11.113.2.2 ~CSw2to64DeviceNet()** [~CSw2to64DeviceNet](#) ( )

**11.113.3 Member Function Documentation**

**11.113.3.1 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

**11.113.3.2 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

**11.113.3.3 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

**11.113.3.4 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

**11.113.3.5 SetChannels()** `void SetChannels (   
    array< unsigned char >^ pattern )`

Sets the switch positions from a char array.

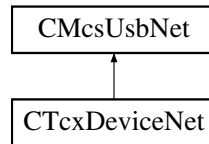
## Parameters

in	<i>pattern</i>	array of char with the size of the number of channels, each char has the setting of a channel
----	----------------	---

## 11.114 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

#### 11.114.1 Detailed Description

Class to control a Temperature Controller (TCX)

#### 11.114.2 Constructor & Destructor Documentation

##### 11.114.2.1 CTcxDeviceNet() [CTcxDeviceNet](#) ( )

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

##### 11.114.2.2 ~CTcxDeviceNet() [~CTcxDeviceNet](#) ( )

#### 11.114.3 Member Function Documentation

##### 11.114.3.1 CalibrateThermocouple() [void CalibrateThermocouple](#) ( unsigned int *channel* )

**11.114.3.2 FactoryReset()** `void FactoryReset ( )`

**11.114.3.3 GetBoardTemp()** `unsigned int GetBoardTemp ( )`

Gets the temperate of the mainboard in units of 0.1 °C.

**11.114.3.4 GetCalibration()** `int GetCalibration (`  
`unsigned int channel )`

**11.114.3.5 GetCalibrationDecp()** `int GetCalibrationDecp (`  
`unsigned int channel )`

**11.114.3.6 GetCalibrationMax()** `int GetCalibrationMax (`  
`unsigned int channel )`

**11.114.3.7 GetCalibrationMin()** `int GetCalibrationMin (`  
`unsigned int channel )`

**11.114.3.8 GetCurrent()** `int GetCurrent (`  
`unsigned int channel )`

**11.114.3.9 GetD()** `int GetD (`  
`unsigned int device )`

Gets the D-coefficient of the specified device.

**11.114.3.10 GetDDecp()** `int GetDDecp (`  
`unsigned int device )`

**11.114.3.11 GetDevice()** `int GetDevice (`  
`unsigned int channel )`

**11.114.3.12 GetDeviceType()** `TcxDeviceTypeEnumNet GetDeviceType ( )`

**11.114.3.13 GetDevname()** `String ^ GetDevname (`  
`unsigned int device )`

**11.114.3.14 GetDMax()** `int GetDMax (`  
`unsigned int device )`

**11.114.3.15 GetDMin()** `int GetDMin (`  
`unsigned int device )`

**11.114.3.16 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 %.

**11.114.3.17 GetEnableHeaterLimit()** `bool GetEnableHeaterLimit (`  
`unsigned int device )`

**11.114.3.18 GetEnableThermocouple()** `bool GetEnableThermocouple (`  
`unsigned int device )`

**11.114.3.19 GetHasThermocouple()** `bool GetHasThermocouple ( )`

Gets whether the device supports a thermocouple.

**11.114.3.20 GetHeaterLimit()** `int GetHeaterLimit (   
 unsigned int device )`

Gets the temperature limit of the specified heater in units of 0.1 °C.

**11.114.3.21 GetHeaterTemp()** `int GetHeaterTemp (   
 unsigned int channel )`

Gets the temperature of the specified heater in units of 0.1 °C.

**11.114.3.22 GetI()** `int GetI (   
 unsigned int device )`

Gets the I-coefficient of the specified device.

**11.114.3.23 GetIDecp()** `int GetIDecp (   
 unsigned int device )`**11.114.3.24 GetIMax()** `int GetIMax (   
 unsigned int device )`**11.114.3.25 GetIMin()** `int GetIMin (   
 unsigned int device )`**11.114.3.26 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.

**11.114.3.27 GetMaxHeaterPowerMultiwell()** `double GetMaxHeaterPowerMultiwell ( )`

queries the max. heater power that the Multiwell temperature controller will apply; unit: W; useful range: 5.2W..7.6W

**11.114.3.28 GetMaxP()** `int GetMaxP (   
 unsigned int device )`

Gets the maximum heater power of the specified device.

**11.114.3.29 GetMaxpDecp()** `int GetMaxpDecp (   
 unsigned int device )`

**11.114.3.30 GetMaxpMax()** `int GetMaxpMax (   
 unsigned int device )`

**11.114.3.31 GetMaxpMin()** `int GetMaxpMin (   
 unsigned int device )`

**11.114.3.32 GetNumControlChannels()** `unsigned int GetNumControlChannels ( )`

Gets the number of channels the device can control/regulate.

**11.114.3.33 GetNumDevices()** `unsigned int GetNumDevices ( )`

**11.114.3.34 GetNumMeasureChannels()** `unsigned int GetNumMeasureChannels ( )`

Gets the number of channels the device can measure.

**11.114.3.35 GetOnOff()** `int GetOnOff (   
 unsigned int channel )`

Gets if the specified channel is on or off.

**11.114.3.36 GetP()** `int GetP (   
 unsigned int device )`

Gets the P-coefficient of the specified device.

**11.114.3.37 GetPDecp()** `int GetPDecp (   
 unsigned int device )`**11.114.3.38 GetPMax()** `int GetPMax (   
 unsigned int device )`**11.114.3.39 GetPMin()** `int GetPMin (   
 unsigned int device )`**11.114.3.40 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.

**11.114.3.41 GetPwrOut()** `int GetPwrOut ( unsigned int channel )`

**11.114.3.42 GetPwrSet()** `int GetPwrSet ( unsigned int channel )`

**11.114.3.43 GetRes1()** `int GetRes1 ( unsigned int channel )`

**11.114.3.44 GetRes2()** `int GetRes2 ( unsigned int channel )`

**11.114.3.45 GetResS()** `int GetResS ( unsigned int channel )`

**11.114.3.46 GetResX()** `int GetResX ( unsigned int channel )`

**11.114.3.47 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.

**11.114.3.48 GetSensorType()** `TcxSensorTypeEnumNet GetSensorType ( unsigned int device )`

**11.114.3.49 GetSetpoint()** `int GetSetpoint ( unsigned int channel )`

Gets the target temperate of specified channel in units of 0.1 °C.

**11.114.3.50 GetSetpointDecp()** `int GetSetpointDecp ( unsigned int channel )`

**11.114.3.51 GetSetpointMax()** `int GetSetpointMax ( unsigned int channel )`

**11.114.3.52 GetSetpointMin()** `int GetSetpointMin ( unsigned int channel )`

**11.114.3.53 GetThermocoupleCalibration()** `int GetThermocoupleCalibration ( unsigned int channel )`

**11.114.3.54 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.

**11.114.3.55 GetThermocoupleReferenceTemp()** `int GetThermocoupleReferenceTemp ( unsigned int channel )`



**11.114.3.56 GetThermocoupleTemp()** `int GetThermocoupleTemp ( unsigned int channel )`

**11.114.3.57 GetThermocoupleTempAbs()** `int GetThermocoupleTempAbs ( unsigned int channel )`

**11.114.3.58 GetUnit()** `String ^ GetUnit ( unsigned int channel )`

**11.114.3.59 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.

**11.114.3.60 GetValue()** `int GetValue ( unsigned int channel )`

Gets the temperate of the specified channel in units of 0.1 °C.

**11.114.3.61 GetValueHires()** `int GetValueHires ( unsigned int channel )`

Gets the temperate of the specified channel in units of 0.01 °C.

**11.114.3.62 GetVolti()** `unsigned int GetVolti ( unsigned int channel )`

**11.114.3.63 SetCalibration()** void SetCalibration (   
 unsigned int *channel*,   
 int *calib* )

**11.114.3.64 SetD()** void SetD (   
 unsigned int *device*,   
 int *d\_coeff* )

Sets the D-coefficient of the specified device.

**11.114.3.65 SetDevice()** void SetDevice (   
 unsigned int *channel*,   
 int *device* )

**11.114.3.66 SetDeviceType()** void SetDeviceType (   
 TcxDeviceTypeEnumNet *devicetype* )

**11.114.3.67 SetDevname()** void SetDevname (   
 unsigned int *device*,   
 String^ *Devicename* )

**11.114.3.68 SetEnableHeaterLimit()** void SetEnableHeaterLimit (   
 unsigned int *device*,   
 bool *enable* )

**11.114.3.69 SetEnableThermocouple()** void SetEnableThermocouple (   
 unsigned int *device*,   
 bool *enable* )

**11.114.3.70 SetHeaterLimit()** void SetHeaterLimit (   
 unsigned int *device*,   
 int *heater\_limit* )

**11.114.3.71 SetI()** void SetI (   
 unsigned int *device*,   
 int *i\_coeff* )

Sets the I-coefficient of the specified device.

**11.114.3.72 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

**11.114.3.73 SetMaxP()** void SetMaxP (   
 unsigned int *device*,   
 int *maxp* )

Sets the maximum heater power of the specified device.

**11.114.3.74 SetOnOff()** void SetOnOff (   
 unsigned int *channel*,   
 bool *on* )

Switches the specified channel on or off.

#### Parameters

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

**11.114.3.75 SetP()** void SetP (   
 unsigned int *device*,   
 int *p\_coeff* )

Sets the P-coefficient of the specified device.

**11.114.3.76 SetSensorType()** void SetSensorType (   
 unsigned int *device*,   
 TcxSensorTypeEnumNet *type* )

**11.114.3.77 SetSetpoint()** `void SetSetpoint (`  
     `unsigned int channel,`  
     `int sp )`

Sets the target temperate of specified channel in units of 0.1 °C.

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

Sets the proportional constant for the thermocouple.

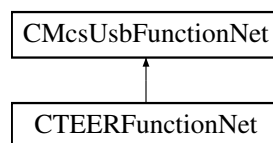
#### Parameters

<i>channel</i>	Thermocouple channel number.
<i>value</i>	Proportional constant in Nanovolt per Kelvin.

## 11.115 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](#) ()  
*gets the adapter code*
- uint32\_t [GetRotaryPositionCode](#) ()  
*gets the rotary position code*
- void [SetExternalLED](#) (uint32\_t NewState)  
*sets the external LED*
- 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)  
*sets the resistance of the liquid in ohms*

- `int32_t GetLiquidResistance ()`  
*gets the resitance of the liquid in ohms*
- `int GetScaleFactorU1 ()`  
*returns U1 scale factor times  $10^6$  (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)*
- `int GetFrameErrorCounter ()`  
*returns number of times (since bootup) sample memory got overwritten*
- `int GetSampleRate ()`  
*returns sample rate in Hz*

### Additional Inherited Members

#### 11.115.1 Detailed Description

[CTEERFunctionNet](#) is the class to control the TEER device

#### 11.115.2 Constructor & Destructor Documentation

**11.115.2.1 CTEERFunctionNet() [1/2]** `CTEERFunctionNet ( CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ pTEERFunctionPointerContainer )`

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

**11.115.2.2 CTEERFunctionNet() [2/2]** `CTEERFunctionNet ( CMcsUsbNet^ mcsusb )`

**11.115.2.3 ~CTEERFunctionNet()** `virtual ~CTEERFunctionNet ( ) [virtual]`

**11.115.2.4 "!CTEERFunctionNet()** `!CTEERFunctionNet ( )`

#### 11.115.3 Member Function Documentation

**11.115.3.1 CancelInternalCalibration()** `void CancelInternalCalibration ( )`

in case the internal calibration "hangs", this will cancel it

**11.115.3.2 GetAdapterCode()** `uint32_t GetAdapterCode ( )`

gets the adapter code

**Returns**

the adapter code

**11.115.3.3 GetAdcOffsetU1()** `int GetAdcOffsetU1 ( )`

returns ADC offset of U1 channel (result of internal calibration)

**Returns**

the ADC offset for U1

**11.115.3.4 GetAdcOffsetU2()** `int GetAdcOffsetU2 ( )`

returns ADC offset of U2 channel (result of internal calibration)

**Returns**

the ADC offset for U2

**11.115.3.5 GetAmplitude\_nA()** `uint32_t GetAmplitude_nA ( )`

gets TEER stimulation amplitude in nA

**Returns**

current stimulation amplitude in nA

**11.115.3.6 GetBytesPerSample()** `uint32_t GetBytesPerSample ( )`

private function to be called internally only

**Returns**

not documented

**11.115.3.7 GetClampMode()** `TeerClampModeEnumNet GetClampMode ( )`

gets TEER clamp mode (voltage/current)

**Returns**

current TEER clamp mode

**11.115.3.8 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>	the P value
<i>I</i>	the I value
<i>D</i>	the D value

**11.115.3.9 GetCurrentEnable()** `bool GetCurrentEnable ( )`

when disabled, no current will flow through chamber

**Returns**

false when disabled, true when enabled

**11.115.3.10 GetDacZero()** `int GetDacZero ( )`

returns DAC-offset (result of internal calibration); use to check for plausibility only

**Returns**

the DAC offset



**11.115.3.11 GetFrameErrorCounter()** `int GetFrameErrorCounter ( )`

returns number of times (since bootup) sample memory got overwritten

**Returns**

the number of errors

**11.115.3.12 GetLiquidResistance()** `int32_t GetLiquidResistance ( )`

gets the resistance of the liquid in ohms

**Returns**

the resistance in ohms

**11.115.3.13 GetMaxChunkSize\_Byte()** `uint32_t GetMaxChunkSize_Byte ( )`

private function to be called internally only

**Returns**

not documented

**11.115.3.14 GetNumberOfAvailableSamples()** `uint32_t GetNumberOfAvailableSamples ( )`

private function to be called internally only

**Returns**

not documented

**11.115.3.15 GetPeriod\_us()** `uint32_t GetPeriod_us ( )`

gets the period of TEER stimulation in us

**Returns**

the period in us

**11.115.3.16 GetRotaryPositionCode()** `uint32_t GetRotaryPositionCode ( )`

gets the rotary position code

**Returns**

the rotary position code

**11.115.3.17 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

not documented

**11.115.3.18 GetSampleRate()** `int GetSampleRate ( )`

returns sample rate in Hz

## Returns

the sample rate in Hz

**11.115.3.19 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

the voltage sample buffer

**11.115.3.20 GetScaleFactorU1()** `int GetScaleFactorU1 ( )`

returns U1 scale factor times  $10^6$  (result of internal calibration)

## Returns

the U1 scale factor

**11.115.3.21 GetScaleFactorU2()** `int GetScaleFactorU2 ( )`

returns U2 scale factor times  $10^6$  (result of internal calibration)

**Returns**

the U2 scale factor

**11.115.3.22 GetUptimeSeconds()** `int32_t GetUptimeSeconds ( )`

returns time in seconds since device was powered up

**Returns**

seconds since power-on

**11.115.3.23 GetWaveform()** `TeerWaveformEnumNet GetWaveform ( )`

gets TEER stimulation waveform (sine/rect)

**Returns**

waveform enum

**11.115.3.24 IsInternalCalibrationFinished()** `bool IsInternalCalibrationFinished ( )`

queries whether internal calibration has finished

**Returns**

true if calibration has finished

**11.115.3.25 IsSamplingFinished()** `uint32_t IsSamplingFinished ( )`

returns false iff stimulation/sampling is going on, otherwise true

**Returns**

true if sampling is finished

**11.115.3.26 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
---------------------	---------------------------------

**11.115.3.27 SetBufferIndex()** `void SetBufferIndex (`  
     `uint32_t NewBufferIndex )`

pre-selects sample buffer to be transferred by [GetSampleVoltageBuffer\\_uV\(\)](#)

## Parameters

<i>NewBufferIndex</i>	0 - chamber voltage; 1 - compliance voltage
-----------------------	---

**11.115.3.28 SetClampMode()** `void SetClampMode (`  
     `TeerClampModeEnumNet ClampMode )`

sets TEER clamp mode (voltage/current)

## Parameters

<i>ClampMode</i>	new TEER clamp mode
------------------	---------------------

**11.115.3.29 SetControllerParams()** `void SetControllerParams (`  
     `uint32_t P,`  
     `uint32_t I,`  
     `uint32_t D )`

sets PID controller parameters for voltage clamp mode

## Parameters

<i>P</i>	the P value
<i>I</i>	the I value
<i>D</i>	the D value

**11.115.3.30 SetCurrentEnable()** `void SetCurrentEnable (`  
     `bool NewCurrentEnable )`

when disabled, no current will flow through chamber

## Parameters

<i>NewCurrentEnable</i>	disabled when false, enabled when true
-------------------------	--

**11.115.3.31 SetExternalLED()** void SetExternalLED (   
 uint32\_t *NewState* )

sets the external LED

## Parameters

<i>NewState</i>	state
-----------------	-------

**11.115.3.32 SetLiquidResistance()** void SetLiquidResistance (   
 int32\_t *NewLiquidResistance\_Ohm* )

sets the resistance of the liquid in ohms

## Parameters

<i>NewLiquidResistance_Ohm</i>	the resistance in ohms
--------------------------------	------------------------

**11.115.3.33 SetPeriod\_us()** void SetPeriod\_us (   
 uint32\_t *period\_us* )

sets the period of TEER stimulation in us

## Parameters

<i>period_us</i>	the period in us
------------------	------------------

**11.115.3.34 SetWaveform()** void SetWaveform (   
 TeerWaveformEnumNet *Waveform* )

sets TEER stimulation waveform (sine/rect)

## Parameters

<i>Waveform</i>	waveform enum
-----------------	---------------

**11.115.3.35 StartInternalCalibration()** `void StartInternalCalibration ( )`

starts determination of internal DAC-offset; result is used internally; NON-BLOCKING call

**11.115.3.36 StartSampling()** `void StartSampling (`  
`uint32_t NumberOfCycles )`

starts TEER stimulation (duration: n cycles) and samples during last cycle

**Parameters**

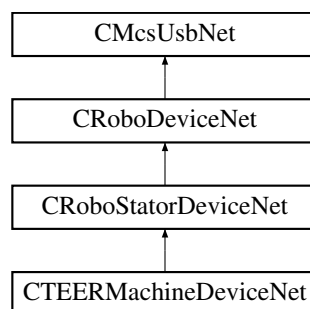
<i>NumberOfCycles</i>	number of cycles (sine or rect) to output (0 - loop forever)
-----------------------	--

**11.115.3.37 StopSampling()** `void StopSampling ( )`

stops TEER stimulation and sampling

**11.116 CTEERMachineDeviceNet Class Reference**

Inheritance diagram for CTEERMachineDeviceNet:

**Public Member Functions**

- [CTEERMachineDeviceNet \( \)](#)
- [~CTEERMachineDeviceNet \( \)](#)

**Properties**

- [CTEERFunctionNet^ TEERFunctionNet](#) [get]

## Additional Inherited Members

### 11.116.1 Constructor & Destructor Documentation

**11.116.1.1 CTEERMachineDeviceNet()** [CTEERMachineDeviceNet](#) ( )

**11.116.1.2 ~CTEERMachineDeviceNet()** [~CTEERMachineDeviceNet](#) ( )

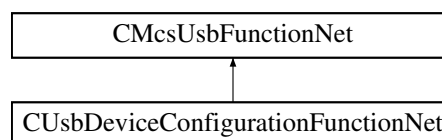
### 11.116.2 Property Documentation

**11.116.2.1 TEERFunctionNet** [CTEERFunctionNet](#)<sup>^</sup> TEERFunctionNet [get]

## 11.117 CUsbDeviceConfigurationFunctionNet Class Reference

[CUsbDeviceConfigurationFunctionNet](#) is the class to configure the USB firmware

Inheritance diagram for CUsbDeviceConfigurationFunctionNet:



## Public Member Functions

- [CUsbDeviceConfigurationFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> p↔  
UsbDeviceConfigurationFunctionPointerContainer)  
*Initializes a new instance of the [CUsbDeviceConfigurationFunctionNet](#) class.*
- [CUsbDeviceConfigurationFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- virtual [~CUsbDeviceConfigurationFunctionNet](#) ()
- [!CUsbDeviceConfigurationFunctionNet](#) ()
- void [SetDeviceName](#) (String<sup>^</sup> name)  
*sets the USB device name for configurable devices*
- void [SetDeviceId](#) (ProductIdEnumNet id)  
*sets the USB device name for configurable devices*

## Additional Inherited Members

### 11.117.1 Detailed Description

[CUsbDeviceConfigurationFunctionNet](#) is the class to configure the USB firmware

### 11.117.2 Constructor & Destructor Documentation

**11.117.2.1 CUsbDeviceConfigurationFunctionNet()** [1/2] [CUsbDeviceConfigurationFunctionNet](#) (   
 [CMcsUsbNet](#)<sup>^</sup> *mcsusb*,  
 [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> *pUsbDeviceConfigurationFunctionPointerContainer*  
 )

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

**11.117.2.2 CUsbDeviceConfigurationFunctionNet()** [2/2] [CUsbDeviceConfigurationFunctionNet](#) (   
 [CMcsUsbNet](#)<sup>^</sup> *mcsusb* )

**11.117.2.3 ~CUsbDeviceConfigurationFunctionNet()** virtual [~CUsbDeviceConfigurationFunctionNet](#) (   
 ) [virtual]

**11.117.2.4 "!CUsbDeviceConfigurationFunctionNet()** [!CUsbDeviceConfigurationFunctionNet](#) ( )

### 11.117.3 Member Function Documentation

**11.117.3.1 SetDeviceId()** void SetDeviceId (   
 [ProductIdEnumNet](#) *id* )

sets the USB device name for configurable devices

#### Parameters

<i>id</i>	
-----------	--



**11.117.3.2 SetDeviceName()** void SetDeviceName (   
 String^ name )

sets the USB device name for configurable devices

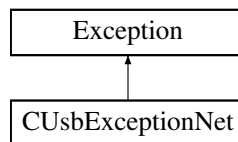
Parameters

<i>name</i>	
-------------	--

## 11.118 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]

### 11.118.1 Detailed Description

Exception class that is thrown in case of an USB error.

### 11.118.2 Constructor & Destructor Documentation

**11.118.2.1 CUsbExceptionNet()** [1/2] [CUsbExceptionNet](#) (   
 uint32\_t status )

Constructor of a CUsbException.

## Parameters

<i>status</i>	the status number
---------------	-------------------

**11.118.2.2 CUsbExceptionNet()** [2/2] [CUsbExceptionNet](#) (  
     uint32\_t *status*,  
     String^ *message* )

**11.118.3 Property Documentation**

**11.118.3.1 Status** uint32\_t Status [get]

**11.119 CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet Class Reference****Public Member Functions**

- [CVoltageRangeInfoNet](#) (int *vr*, String^ *vrString*)

**Public Attributes**

- int [VoltageRangeInMicroVolt](#)
- String^ [VoltageRangeDisplayStringMilliVolt](#)

**11.119.1 Constructor & Destructor Documentation**

**11.119.1.1 CVoltageRangeInfoNet()** [CVoltageRangeInfoNet](#) (  
     int *vr*,  
     String^ *vrString* )

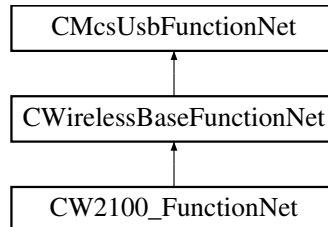
**11.119.2 Member Data Documentation**

**11.119.2.1 VoltageRangeDisplayStringMilliVolt** String^ VoltageRangeDisplayStringMilliVolt

### 11.119.2.2 VoltageRangeInMicroVolt `int VoltageRangeInMicroVolt`

## 11.120 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>> ^ 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 > ^ 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 > ^ [GetAnalogOutFilter](#) ([System::Runtime::InteropServices::Out]int % automatic)
- void [SetAnalogOutFilter](#) (int automatic, array< unsigned int > ^ Coeffs)
- AnalogOut\_DAC\_Range\_EnumNet [GetDacRange](#) ()
- void [SetDacRange](#) (AnalogOut\_DAC\_Range\_EnumNet range)
- CFilterPropertyNet ^ [GetFilterProperty](#) (W2100DacqGroupChannelEnumNet GroupID, unsigned int index)
- array< CFilterPropertyNet ^ > ^ [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](#) ^ > ^ channels)
- array< [AudioChannelsNet](#) ^ > ^ [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](#) ^ [Stimulator](#) [get]
- [CPulseGeneratorFunctionNet](#) ^ [PulseGenerator](#) [get]

### Additional Inherited Members

#### 11.120.1 Constructor & Destructor Documentation

**11.120.1.1 CW2100\_FunctionNet()** [1/2] [CW2100\\_FunctionNet](#) (   
[CMcsUsbNet](#) ^ *mcsusb*,   
[CMcsUsbFunctionPointerContainer](#) ^ *w2100\_FunctionPointerContainer* )

**11.120.1.2 CW2100\_FunctionNet()** [2/2] [CW2100\\_FunctionNet](#) (   
[CMcsUsbNet](#) ^ *mcsusb* )

## 11.120.2 Member Function Documentation

**11.120.2.1 ClearStimulusParametersCache()** [1/2] static void ClearStimulusParametersCache ( )  
[static]

**11.120.2.2 ClearStimulusParametersCache()** [2/2] static void ClearStimulusParametersCache (   
unsigned short *ID* ) [static]

**11.120.2.3 ClearUserDefinedNameCache()** [1/2] static void ClearUserDefinedNameCache ( ) [static]

**11.120.2.4 ClearUserDefinedNameCache()** [2/2] static void ClearUserDefinedNameCache (   
unsigned short *ID* ) [static]

**11.120.2.5 DeselectAllHeadstages()** void DeselectAllHeadstages ( )

**11.120.2.6 DeselectHeadstage()** void DeselectHeadstage (   
int *TimeSlotNr* )

**11.120.2.7 GetAccelGyroCurrentRate()** int GetAccelGyroCurrentRate (   
int *TimeSlotNr* )

**11.120.2.8 GetAccelGyroDesiredRate()** int GetAccelGyroDesiredRate (   
int *TimeSlotNr* )

**11.120.2.9 GetAccelGyroEnabled()** W2100\_Accel\_Gyro\_Select\_EnumNet GetAccelGyroEnabled (   
int *TimeSlotNr* )

- 11.120.2.10 GetAccelRange()** `int GetAccelRange ( int TimeSlotNr )`
- 11.120.2.11 GetAnalogOutChannel()** `unsigned int GetAnalogOutChannel ( [System::Runtime::InteropServices::Out] int % automatic, unsigned short index )`
- 11.120.2.12 GetAnalogOutFilter()** `array<unsigned int> ^ GetAnalogOutFilter ( [System::Runtime::InteropServices::Out] int % automatic )`
- 11.120.2.13 GetAudioChannels()** `array<AudioChannelsNet^> ^ GetAudioChannels ( )`
- 11.120.2.14 GetAvailableHeadstages()** `array<HeadStageIDType^> ^ GetAvailableHeadstages ( unsigned int max_length )`
- 11.120.2.15 GetBatteryState()** `BatteryState ^ GetBatteryState ( int TimeSlotNr )`
- 11.120.2.16 GetDacRange()** `AnalogOut_DAC_Range_EnumNet GetDacRange ( )`
- 11.120.2.17 GetFilterProperties()** `array<CFilterPropertyNet^> ^ GetFilterProperties ( W2100DacqGroupChannelEnumNet GroupID )`
- 11.120.2.18 GetFilterProperty()** `CFilterPropertyNet ^ GetFilterProperty ( W2100DacqGroupChannelEnumNet GroupID, unsigned int index )`
- 11.120.2.19 GetFPGA FirmwareType()** `unsigned int GetFPGA FirmwareType ( int TimeSlotNr )`

**11.120.2.20 GetGyroRange()** int GetGyroRange (   
 int *TimeSlotNr* )

**11.120.2.21 GetHeadstageOnOff()** unsigned short GetHeadstageOnOff (   
 int *TimeSlotNr* )

**11.120.2.22 GetHeadstageSamplingActive()** bool GetHeadstageSamplingActive (   
 int *TimeSlotNr* )

**11.120.2.23 GetMultiHeadstageMode()** bool GetMultiHeadstageMode ( )

**11.120.2.24 GetPicFirmwareType()** unsigned int GetPicFirmwareType (   
 int *TimeSlotNr* )

**11.120.2.25 GetSelectedChannels()** array<BYTE> ^ GetSelectedChannels (   
 int *TimeSlotNr* )

**11.120.2.26 GetSelectedHeadstageState()** HeadStageIDTypeState ^ GetSelectedHeadstageState (   
 int *TimeSlotNr* )

**11.120.2.27 GetStimulusParametersCache()** [1/2] W2100\_StimulusParametersNet ^ GetStimulus↔  
ParametersCache (   
 unsigned int *typeValue* )

**11.120.2.28 GetStimulusParametersCache()** [2/2] uint32\_t GetStimulusParametersCache (   
 unsigned int *typeValue*,   
 [System::Runtime::InteropServices::Out] W2100\_StimulusParametersNet^% *Stimulus↔*  
*Parameters* )

**11.120.2.29 GetStimulusParametersFromSelectedHS()** [W2100\\_StimulusParametersNet](#) ^ GetStimulusParametersFromSelectedHS (   
     int *TimeSlotNr* )

**11.120.2.30 GetStiumlusParameters()** [W2100\\_StimulusParametersNet](#) ^ GetStiumlusParameters (   
     unsigned short *ID* )

**11.120.2.31 GetUserDefinedName()** System::String ^ GetUserDefinedName (   
     unsigned short *ID* )

**11.120.2.32 GetUserDefinedNameCache() [1/2]** System::String ^ GetUserDefinedNameCache (   
     unsigned short *ID* )

**11.120.2.33 GetUserDefinedNameCache() [2/2]** uint32\_t GetUserDefinedNameCache (   
     unsigned short *ID*,   
     [System::Runtime::InteropServices::Out] System::String^% *Name* )

**11.120.2.34 GetUserDefinedNameFromSelectedHS()** System::String ^ GetUserDefinedNameFromSelectedHS (   
     int *TimeSlotNr* )

**11.120.2.35 SelectHeadstage()** void SelectHeadstage (   
     unsigned int *IDorEntry*,   
     int *TimeSlotNr* )

**11.120.2.36 SetAccelGyroDesiredRate()** void SetAccelGyroDesiredRate (   
     int *rate*,   
     int *TimeSlotNr* )

**11.120.2.37 SetAccelGyroEnabled()** void SetAccelGyroEnabled (   
     W2100\_Accel\_Gyro\_Select\_EnumNet *enable*,   
     int *TimeSlotNr* )



- 11.120.2.38 SetAccelRange()** void SetAccelRange (   
    int *range*,  
    int *TimeSlotNr* )
- 11.120.2.39 SetAnalogOutChannel()** void SetAnalogOutChannel (   
    int *automatic*,  
    unsigned short *index*,  
    unsigned int *Channel* )
- 11.120.2.40 SetAnalogOutFilter()** void SetAnalogOutFilter (   
    int *automatic*,  
    array< unsigned int >^ *Coeffs* )
- 11.120.2.41 SetAudioChannels()** void SetAudioChannels (   
    array< [AudioChannelsNet](#)^>^ *channels* )
- 11.120.2.42 SetDacRange()** void SetDacRange (   
    AnalogOut\_DAC\_Range\_EnumNet *range* )
- 11.120.2.43 SetGyroRange()** void SetGyroRange (   
    int *range*,  
    int *TimeSlotNr* )
- 11.120.2.44 SetHeadstageOnOff()** void SetHeadstageOnOff (   
    unsigned short *On*,  
    int *TimeSlotNr* )
- 11.120.2.45 SetHeadstageSamplingActive()** void SetHeadstageSamplingActive (   
    bool *Active*,  
    int *TimeSlotNr* )

**11.120.2.46 SetHeadstageToSleep()** `void SetHeadstageToSleep (`  
    `unsigned int Sleep16ms,`  
    `int TimeSlotNr )`

**11.120.2.47 SetMultiHeadstageMode()** `void SetMultiHeadstageMode (`  
    `bool Mode )`

**11.120.2.48 SetSelectedChannels()** `void SetSelectedChannels (`  
    `array< BYTE >^ channels,`  
    `int TimeSlotNr )`

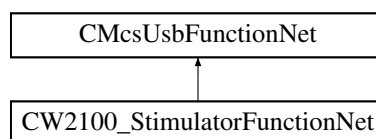
### 11.120.3 Property Documentation

**11.120.3.1 PulseGenerator** `CPulseGeneratorFunctionNet^ PulseGenerator [get]`

**11.120.3.2 Stimulator** `CW2100_StimulatorFunctionNet^ Stimulator [get]`

## 11.121 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](#) ^ [PrepareData](#) (int channel, array< int32\_t > ^ amplitude, array< uint64\_t > ^ duration, STG\_DestinationEnumNet destType, uint32\_t repeat)
- [CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData](#) ^ [PrepareDataSync](#) (int channel, array< int32\_t > ^ amplitude, array< uint32\_t > ^ Sync, array< uint64\_t > ^ duration, STG\_DestinationEnumNet destType, uint32\_t repeat)
- void [SendPreparedData](#) (int channel, [CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData](#) ^ 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](#) ()
- 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

### 11.121.1 Constructor & Destructor Documentation

**11.121.1.1 CW2100\_StimulatorFunctionNet()** [CW2100\\_StimulatorFunctionNet](#) ( [CMcsUsbNet](#)<sup>^</sup> *mcsusb* )

### 11.121.2 Member Function Documentation

**11.121.2.1 ClearChannelData()** `void ClearChannelData (   
int channel )`

Delete a Stimulus Pattern from STG memory

#### Parameters

<i>channel</i>	specifies the channel to clear.
----------------	---------------------------------

**11.121.2.2 GetBoostAlwaysOnMode()** `uint32_t GetBoostAlwaysOnMode ( )`

**11.121.2.3 GetBoostPreTime()** `uint32_t GetBoostPreTime ( )`

**11.121.2.4 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.

**11.121.2.5 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.

**11.121.2.6 GetDACResolution()** `int GetDACResolution ( )`

Gets number of bits of the DAC resolution.

**Returns**

The DAC resolution in bits.

**11.121.2.7 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 )`

**11.121.2.8 GetDigitalStimulatorTriggerSlope()** `DigitalStimulatorTriggerSlopeEnumNet GetDigital↔ StimulatorTriggerSlope ( int TimeSlotNr, DigitalStimulatorTriggerEventEnumNet trigger_event, int trigger_number )`

**11.121.2.9 GetNumberOfAnalogChannels()** uint32\_t GetNumberOfAnalogChannels ( )

**11.121.2.10 GetNumberOfSyncoutChannels()** uint32\_t GetNumberOfSyncoutChannels ( )

**11.121.2.11 GetNumberOfTriggerInputs()** uint32\_t GetNumberOfTriggerInputs ( )

**11.121.2.12 GetStimulationPatternMemory()** uint32\_t GetStimulationPatternMemory ( )

**11.121.2.13 GetTimeResolutionInNanoSeconds()** int GetTimeResolutionInNanoSeconds ( )

Gets number of bits of the DAC resolution.

#### Returns

The time resolution in ns.

**11.121.2.14 GetTimeSlot()** int GetTimeSlot ( )

**11.121.2.15 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.

**11.121.2.16 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.

**11.121.2.17 PrepareData()** `CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData` ^ Prepare↔  
Data (

```
int channel,
array< int32_t >^ amplitude,
array< uint64_t >^ duration,
STG_DestinationEnumNet destType,
uint32_t repeat )
```

**11.121.2.18 PrepareDataSync()** `CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData` ^ Prepare↔  
DataSync (

```
int channel,
array< int32_t >^ amplitude,
array< uint32_t >^ Sync,
array< uint64_t >^ duration,
STG_DestinationEnumNet destType,
uint32_t repeat )
```

**11.121.2.19 SelectTimeSlot()** void SelectTimeSlot (

```
int TimeSlotNr )
```

**11.121.2.20 SendPreparedData()** void SendPreparedData (

```
int channel,
CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData^ device_data_and_unrolled,
STG_DestinationEnumNet destType )
```

**11.121.2.21 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.
-------------------	---

**11.121.2.22 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.
-------------------	---

**11.121.2.23 SetDigitalStimulatorTrigger()** `void SetDigitalStimulatorTrigger (`  
     int TimeSlotNr,  
     DigitalStimulatorTriggerEventEnumNet *trigger\_event*,  
     int trigger\_number,  
     W2100DigitalSourceEnumNet *digstream\_source*,  
     int bitnumber\_offset `)`

**11.121.2.24 SetDigitalStimulatorTriggerSlope()** `void SetDigitalStimulatorTriggerSlope (`  
     int TimeSlotNr,  
     DigitalStimulatorTriggerEventEnumNet *trigger\_event*,  
     int trigger\_number,  
     DigitalStimulatorTriggerSlopeEnumNet *slope* `)`

**11.121.2.25 StartPoll()** `void StartPoll ( )`

**11.121.2.26 StopPoll()** `void StopPoll ( )`

**11.121.3 Member Data Documentation**

**11.121.3.1 BOOST\_BIT** `const uint32_t BOOST_BIT = (1 << 0) [static]`

**11.121.3.2 GND\_SWITCH\_BIT** `const uint32_t GND_SWITCH_BIT = (1 << 1) [static]`

**11.121.3.3 SYNC\_BIT0** `const uint32_t SYNC_BIT0 = (1 << 2) [static]`

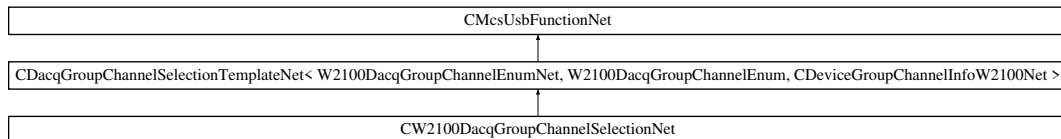
**11.121.3.4 SYNC\_BIT1** `const uint32_t SYNC_BIT1 = (1 << 3) [static]`

#### 11.121.4 Event Documentation

**11.121.4.1 PollStatusEvent** `OnStgPollStatus^ PollStatusEvent`

## 11.122 CW2100DacqGroupChannelSelectionNet Class Reference

Inheritance diagram for CW2100DacqGroupChannelSelectionNet:



#### Public Member Functions

- `CW2100DacqGroupChannelSelectionNet (CMcsUsbNet^ mcsusb)`

#### Additional Inherited Members

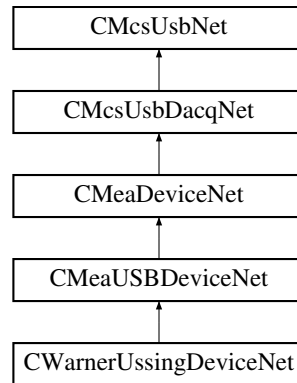
#### 11.122.1 Constructor & Destructor Documentation

**11.122.1.1 CW2100DacqGroupChannelSelectionNet()** `CW2100DacqGroupChannelSelectionNet ( CMcsUsbNet^ mcsusb )`

## 11.123 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](#) ()
- [!CWarnerUssingDeviceNet](#) ()

### Properties

- [CWarnerUssingFunctionNet](#)<sup>^</sup> [WarnerUssingFunction](#) [get]

### Additional Inherited Members

#### 11.123.1 Detailed Description

[CWarnerUssingDeviceNet](#) is the class to control the Ussing device

#### 11.123.2 Constructor & Destructor Documentation

##### 11.123.2.1 CWarnerUssingDeviceNet() [CWarnerUssingDeviceNet](#) ( )

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

##### 11.123.2.2 ~CWarnerUssingDeviceNet() virtual [~CWarnerUssingDeviceNet](#) ( ) [virtual]

11.123.2.3 `!CWarnerUssingDeviceNet() !CWarnerUssingDeviceNet ( )`

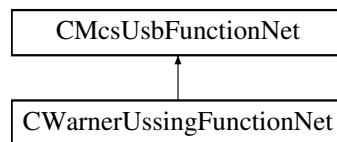
### 11.123.3 Property Documentation

11.123.3.1 **WarnerUssingFunction** `CWarnerUssingFunctionNet^ WarnerUssingFunction [get]`

## 11.124 CWarnerUssingFunctionNet Class Reference

`CWarnerUssingFunctionNet` is the class to control the Ussing device

Inheritance diagram for `CWarnerUssingFunctionNet`:



### Public Member Functions

- `CWarnerUssingFunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ pWarnerUssingFunctionPointerContainer)`  
*Initializes a new instance of the `CWarnerUssingFunctionNet` class.*
- `CWarnerUssingFunctionNet (CMcsUsbNet^ mcsusb)`
- `virtual ~CWarnerUssingFunctionNet ()`
- `!CWarnerUssingFunctionNet ()`
- `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 >^ Amplitudes, array< int >^ Durations, array< int >^ PulseMarker)`  
*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)  
*switch to high-current mode*
- void [SetLowCurrentMode](#) (int32\_t ChamberId)  
*switch to low-current mode*
- bool [IsHighCurrentMode](#) (int32\_t ChamberId)
- uint32\_t [GetLowCurrentRange](#) (int32\_t ChamberId)  
*query the range of the low current mode*
- uint32\_t [GetHighCurrentRange](#) (int32\_t ChamberId)  
*query the range of the high current mode*
- uint32\_t [GetDacPampsPerDigitLowCurrentRange](#) (int32\_t ChamberId)  
*get the resolution of the low current mode*
- uint32\_t [GetDacPampsPerDigitHighCurrentRange](#) (int32\_t ChamberId)  
*get the resolution of the high current mode*
- 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)  
*gets the unit exponent for specified chamber and channel*
- UssingUnitEnumNet [GetUnitName](#) (int32\_t ChamberId, int32\_t ChannelId)  
*gets the channel's unit name*
- String ^ [GetUnitDescription](#) (int32\_t ChamberId, int32\_t ChannelId)  
*gets the description for the unit*
- array< int > ^ [GetAvailableChambers](#) ()  
*returns array with (zero-based) ChamberIds of all available chambers*
- int32\_t [GetUptimeSeconds](#) (int32\_t ChamberId)  
*gets the uptime in seconds*
- 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)  
*sets the resistance of the liquid*

- `int32_t GetLiquidResistance (int32_t ChamberId)`  
*gets the resistance of the liquid*
- `int32_t GetComplianceVoltageThreshold (int32_t ChamberId)`  
*returns compliance voltage threshold in uV; when  $U_c$  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

### 11.124.1 Detailed Description

[CWarnerUssingFunctionNet](#) is the class to control the Ussing device

### 11.124.2 Constructor & Destructor Documentation

**11.124.2.1 CWarnerUssingFunctionNet() [1/2]** `CWarnerUssingFunctionNet ( CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ pWarnerUssingFunctionPointerContainer )`

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

**11.124.2.2 CWarnerUssingFunctionNet() [2/2]** `CWarnerUssingFunctionNet ( CMcsUsbNet^ mcsusb )`

**11.124.2.3 ~CWarnerUssingFunctionNet()** `virtual ~CWarnerUssingFunctionNet ( ) [virtual]`

**11.124.2.4 "!CWarnerUssingFunctionNet()** `!CWarnerUssingFunctionNet ( )`

### 11.124.3 Member Function Documentation

**11.124.3.1 CompensateElectrodeOffset()** `bool CompensateElectrodeOffset ( int32_t ChamberId )`

blocking call to compensate electrode offset of one chamber; returns true when successful

**Parameters**

<i>Chamber↔ Id</i>	index of hardware chamber slot (zero-based)
------------------------	---

**Returns**

true if compensation succeeded

**11.124.3.2 GetAvailableChambers()** `array<int> ^ GetAvailableChambers ( )`

returns array with (zero-based) ChamberIds of all available chambers

**11.124.3.3 GetChannelsCountOfChamber()** `int32_t GetChannelsCountOfChamber (   
int32_t ChamberId )`

gets number of channels in datastream from chamber amp with given index

**Parameters**

<i>Chamber↔ Id</i>	index of hardware chamber slot (zero-based)
------------------------	---

**Returns**

return value of zero means that amp is not placed

**11.124.3.4 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**

the current clamp mode

**11.124.3.5 GetComplianceVoltageThreshold()** `int32_t GetComplianceVoltageThreshold (int32_t ChamberId )`

returns compliance voltage threshold in uV; when Uc is above, current source is overloaded

#### Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based)
------------------	---

#### Returns

the compliance voltage threshold in uV

**11.124.3.6 GetDacPampsPerDigitHighCurrentRange()** `uint32_t GetDacPampsPerDigitHighCurrentRange (int32_t ChamberId )`

get the resolution of the high current mode

#### Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based)
------------------	---

#### Returns

unit: pA/digit in high current mode

**11.124.3.7 GetDacPampsPerDigitLowCurrentRange()** `uint32_t GetDacPampsPerDigitLowCurrentRange (int32_t ChamberId )`

get the resolution of the low current mode

#### Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based)
------------------	---

#### Returns

pA/digit in low current mode



**11.124.3.8 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

the zero value of the DAC

**11.124.3.9 GetHighCurrentRange()** `uint32_t GetHighCurrentRange (`  
`int32_t ChamberId )`

query the range of the high current mode

#### Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based)
------------------	---

#### Returns

low current range in nA

**11.124.3.10 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 uV

**11.124.3.11 GetLiquidResistance()** `int32_t GetLiquidResistance (`  
`int32_t ChamberId )`

gets the resistance of the liquid

#### Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based)
------------------	---

#### Returns

the liquid resistance in ohm

**11.124.3.12 GetLowCurrentRange()** `uint32_t GetLowCurrentRange (`  
`int32_t ChamberId )`

query the range of the low current mode

#### Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based)
------------------	---

#### Returns

low current range in nA

**11.124.3.13 GetNumberOfAvailableChambers()** `int32_t GetNumberOfAvailableChambers ( )`

gets number of actually connected chamber amps

#### Returns

the number of actually connected chambers

**11.124.3.14 GetNumberOfHardwareSlotsForChambers()** `int32_t GetNumberOfHardwareSlotsForChambers ( )`

gets number of physical hardware slots for chambers amps

#### Returns

the number of hardware chamber slots on the backplane

**11.124.3.15 GetU1Offset()** `int32_t GetU1Offset (`  
`int32_t ChamberId )`

- diagnostic function only -

## Parameters

<i>Chamber↔ Id</i>	index of hardware chamber slot (zero-based)
------------------------	---

## Returns

U1 offset

**11.124.3.16 GetU1Reference()** `int32_t GetU1Reference (`  
`int32_t ChamberId )`

- diagnostic function only -

## Parameters

<i>Chamber↔ Id</i>	index of hardware chamber slot (zero-based)
------------------------	---

## Returns

U1 reference

**11.124.3.17 GetU2Offset()** `int32_t GetU2Offset (`  
`int32_t ChamberId )`

- diagnostic function only -

## Parameters

<i>Chamber↔ Id</i>	index of hardware chamber slot (zero-based)
------------------------	---

## Returns

U2 offset

**11.124.3.18 GetU2Reference()** `int32_t GetU2Reference (`  
`int32_t ChamberId )`

- diagnostic function only -

## Parameters

<i>Chamber↔ Id</i>	index of hardware chamber slot (zero-based)
------------------------	---

## Returns

U2 reference

**11.124.3.19 GetUnitDescription()** `String ^ GetUnitDescription (`  
    `int32_t ChamberId,`  
    `int32_t ChannelId )`

gets the description for the unit

## Parameters

<i>Chamber↔ Id</i>	index of hardware chamber slot (zero-based)
<i>ChannelId</i>	index of channel (zero-based)

## Returns

the description of the unix

**11.124.3.20 GetUnitExponent()** `int32_t GetUnitExponent (`  
    `int32_t ChamberId,`  
    `int32_t ChannelId )`

gets the unit exponent 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

example: return value -9 means that amps/volts per digit is in nano

**11.124.3.21 GetUnitName()** `UssingUnitEnumNet GetUnitName (`  
    `int32_t ChamberId,`  
    `int32_t ChannelId )`

gets the channel's unit name

#### Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based)
<i>ChannelId</i>	index of channel (zero-based)

#### Returns

the name of the unit

**11.124.3.22 GetUnitsPerDigit()** `uint32_t GetUnitsPerDigit (`  
    `int32_t ChamberId,`  
    `int32_t ChannelId )`

gets amps/volts per digit for specified chamber and channel

#### Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based)
<i>ChannelId</i>	index of channel (zero-based)

#### Returns

amps/volts per digit

**11.124.3.23 GetUptimeSeconds()** `int32_t GetUptimeSeconds (`  
    `int32_t ChamberId )`

gets the uptime in seconds

#### Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based)
------------------	---

#### Returns

seconds since power-on

**11.124.3.24 GetVoltageClampControllerParam\_D()** `uint32_t GetVoltageClampControllerParam_D ( int32_t ChamberId )`

gets D value of PID controller;

#### Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based)
------------------	---

#### Returns

the D value

**11.124.3.25 GetVoltageClampControllerParam\_I()** `uint32_t GetVoltageClampControllerParam_I ( int32_t ChamberId )`

gets I value of PID controller;

#### Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based)
------------------	---

#### Returns

the I value

**11.124.3.26 GetVoltageClampControllerParam\_P()** `uint32_t GetVoltageClampControllerParam_P ( int32_t ChamberId )`

gets P value of PID controller;

#### Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based)
------------------	---

#### Returns

the P value

**11.124.3.27 IsChamberAvailable()** `bool IsChamberAvailable (`  
    `int32_t ChamberId )`

checks whether chamber amp is connected to slot

#### Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based)
------------------	---

#### Returns

true if the chamber is available

**11.124.3.28 IsHighCurrentMode()** `bool IsHighCurrentMode (`  
    `int32_t ChamberId )`

#### Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based)
------------------	---

#### Returns

true if in high current mode

**11.124.3.29 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

true if finished

**11.124.3.30 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

**11.124.3.31 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>	the clamp mode to use

**11.124.3.32 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

**11.124.3.33 SetHighCurrentMode()** void SetHighCurrentMode (  
    int32\_t *ChamberId* )

switch to high-current mode

## Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based)
------------------	---

**11.124.3.34 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 uV

**11.124.3.35 SetLiquidResistance()** void SetLiquidResistance (   
     int32\_t *ChamberId*,   
     int32\_t *NewLiquidResistance\_Ohm* )

sets the resistance of the liquid

## Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based)
<i>NewLiquidResistance_Ohm</i>	the liquid resistance in ohm

**11.124.3.36 SetLowCurrentMode()** void SetLowCurrentMode (   
     int32\_t *ChamberId* )

switch to low-current mode

## Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based)
------------------	---

**11.124.3.37 SetPulse()** void SetPulse (   
     int32\_t *ChamberId*,

```
UssingClampModeEnumNet StgMode,
int32_t NumberOfRepetitions,
array< int >^ Amplitudes,
array< int >^ Durations,
array< int >^ PulseMarker )
```

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: uV; 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
<i>PulseMarker</i>	defines values on digital channel for each step (positive: digital channel "01", neg: "10", zero: "00")

**11.124.3.38 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

**11.124.3.39 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

**11.124.3.40 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

**11.124.3.41 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...)

**11.124.3.42 WaitForChamber()** `bool WaitForChamber (`  
`int32_t ChamberId )`

blocking call that waits for chamber boot-up calibration to complete

#### Parameters

<i>ChamberId</i>	index of hardware chamber slot (zero-based)
------------------	---

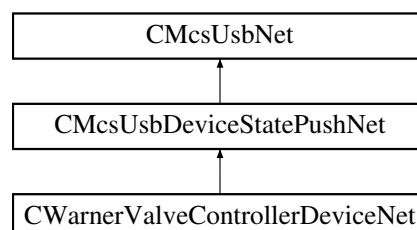
#### Returns

returns false when calibration fails (e.g. timeout...)

## 11.125 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 [OnGetValveLedOn](#) (bool ledon)
- delegate void [OnGetDisplayMode](#) (WvcDisplayModeEnumNet DisplayMode)
- [CWarnerValveControllerDeviceNet](#) ()  
*Initializes a new instance of the [CWarnerValveControllerDeviceNet](#) class.*
- virtual [~CWarnerValveControllerDeviceNet](#) ()
- [ICWarnerValveControllerDeviceNet](#) ()
- 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 table 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*
- bool [GetValveLedOn](#) ()
- Gets the LED state of the valve board*
- void [SetValveLedOn](#) (bool ledon)
- Gets the LED state 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](#)<sup>^</sup> [GetAnalogThresholdLowEvent](#) [add, remove, raise]  
*Event fires when the threshold in mV for the valve number has changed*
- [OnGetAnalogThresholdHigh](#)<sup>^</sup> [GetAnalogThresholdHighEvent](#) [add, remove, raise]  
*Event fires when the threshold in mV for the valve number has changed*
- [OnGetDigitalPortDirection](#)<sup>^</sup> [GetDigitalPortDirectionEvent](#) [add, remove, raise]  
*Event fires when the direction for the port number has changed*
- [OnIsValveDigitalInInverted](#)<sup>^</sup> [IsValveDigitalInInvertedEvent](#) [add, remove, raise]  
*Event fires when is inverted for the valve number has changed*
- [OnGetValveDigitalInPort](#)<sup>^</sup> [GetValveDigitalInPortEvent](#) [add, remove, raise]  
*Event fires when the digital in port for the valve number has changed*
- [OnIsDigitalOutPortInverted](#)<sup>^</sup> [IsDigitalOutPortInvertedEvent](#) [add, remove, raise]  
*Event fires when is inverted for the digital out port has changed*
- [OnGetDigitalOutPortValve](#)<sup>^</sup> [GetDigitalOutPortValveEvent](#) [add, remove, raise]  
*Event fires when the valve number for the digital out port has changed*
- [OnIsValveOpen](#)<sup>^</sup> [IsValveOpenEvent](#) [add, remove, raise]  
*Event fires when is open for the valve number has changed*
- [OnIsValveOpenInDigitalMode](#)<sup>^</sup> [IsValveOpenInDigitalModeEvent](#) [add, remove, raise]  
*Event fires when is open for the valve number has changed*
- [OnIsValveOpenInAnalogMode](#)<sup>^</sup> [IsValveOpenInAnalogModeEvent](#) [add, remove, raise]  
*Event fires when is open for the valve number has changed*
- [OnGetAnalogVoltage](#)<sup>^</sup> [GetAnalogVoltageEvent](#) [add, remove, raise]  
*Event fires when the voltage in mV has changed*
- [OnTableEntryChanged](#)<sup>^</sup> [TableEntryChangedEvent](#) [add, remove, raise]  
*Event fires when an entry of a table changed*
- [OnGetTableNamebyIndex](#)<sup>^</sup> [GetTableNamebyIndexEvent](#) [add, remove, raise]  
*Event fires when the name of the table for the table number has changed*
- [OnGetActiveRunningTableNumber](#)<sup>^</sup> [GetActiveRunningTableNumberEvent](#) [add, remove, raise]  
*Event fires when the table number has changed*
- [OnGetCurrentNumberOfValves](#)<sup>^</sup> [GetCurrentNumberOfValvesEvent](#) [add, remove, raise]  
*Event fires when the number of valves has changed*
- [OnGetValveBoardRevision](#)<sup>^</sup> [GetValveBoardRevisionEvent](#) [add, remove, raise]  
*Event fires when the revision code has changed*
- [OnGetValveLedOn](#)<sup>^</sup> [GetValveLedOnEvent](#) [add, remove, raise]  
*Event fires when the LED state has changed*
- [OnGetDisplayMode](#)<sup>^</sup> [GetDisplayModeEvent](#) [add, remove, raise]  
*Event fires when the display mode has changed*

## Additional Inherited Members

### 11.125.1 Detailed Description

[CWarnerValveControllerDeviceNet](#) is the class to access the Warner Valve Controller

### 11.125.2 Constructor & Destructor Documentation



**11.125.2.1 CWarnerValveControllerDeviceNet()** `CWarnerValveControllerDeviceNet ( )`

Initializes a new instance of the `CWarnerValveControllerDeviceNet` class.

**11.125.2.2 ~CWarnerValveControllerDeviceNet()** `virtual ~CWarnerValveControllerDeviceNet ( )`  
[virtual]**11.125.2.3 ~!CWarnerValveControllerDeviceNet()** `!CWarnerValveControllerDeviceNet ( )`**11.125.3 Member Function Documentation****11.125.3.1 ClearTableName()** `void ClearTableName ( )`

Clear the name of current protocol table

**11.125.3.2 ClearValveTable()** `void ClearValveTable (`  
`uint16_t valve )`

Clear the valve protocol table

**Parameters**

<i>valve</i>	The valve number
--------------	------------------

**11.125.3.3 GetActiveRunningTableNumber()** `uint32_t GetActiveRunningTableNumber ( )`

Gets the number of the table that is active for running

**Returns**

The table number

**11.125.3.4 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

**11.125.3.5 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

**11.125.3.6 GetAnalogVoltage()** `int32_t GetAnalogVoltage ( )`

Reads the voltage on the analog in port

## Returns

The voltage in mV

**11.125.3.7 GetCurrentEditTableNumber()** `uint32_t GetCurrentEditTableNumber ( )`

Gets the number of the table that is current for editing

## Returns

The table number

**11.125.3.8 GetCurrentNumberOfValves()** `int32_t GetCurrentNumberOfValves ( )`

Get the current number of valves connected to the system

**Returns**

The number of valves

**11.125.3.9 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

**11.125.3.10 GetDigitalPortDirection()** `PortDirectionEnumNet GetDigitalPortDirection ( uint16_t port )`

Gets the direction of a digital port

## Parameters

<i>port</i>	The port number
-------------	-----------------

## Returns

the direction

**11.125.3.11 GetDisplayMode()** `WvcDisplayModeEnumNet GetDisplayMode ( )`

Reads the display mode

## Returns

The display mode

**11.125.3.12 GetTableName()** `String ^ GetTableName ( )`

Get the name of the current protocol table

## Returns

The name of the table

**11.125.3.13 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

**11.125.3.14 GetTotalNumberOfDigitalPorts()** `int32_t GetTotalNumberOfDigitalPorts ( )`

Get the total number of digital ports in the system

## Returns

The number of digital ports

**11.125.3.15 GetTotalNumberOfTables()** `int32_t GetTotalNumberOfTables ( )`

Get the total number of tables in the system

## Returns

The number of tables

**11.125.3.16 GetTotalNumberOfValves()** `int32_t GetTotalNumberOfValves ( )`

Get the total number of valves in the system

## Returns

The number of valves

**11.125.3.17 GetTotalTableSize()** `int32_t GetTotalTableSize ( )`

Get the total table size in the system

## Returns

The table size

**11.125.3.18 GetValveActive()** `int GetValveActive ( uint16_t valve )`

Gets the valve active/inactive state

## Parameters

<i>valve</i>	The valve number
--------------	------------------

## Returns

The valve state

**11.125.3.19 GetValveBoardRevision()** `uint32_t GetValveBoardRevision ( )`

Gets the revision code of the valve board

## Returns

The revision code

**11.125.3.20 GetValveBoardRevisionString()** `String ^ GetValveBoardRevisionString ( )`

Gets the revision name of the valve board

## Returns

The revision name

**11.125.3.21 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

**11.125.3.22 GetValveLedOn()** `bool GetValveLedOn ( )`

Gets the LED state of the valve board

**Returns**

The LED state

**11.125.3.23 GetValveManualGroup()** `int32_t GetValveManualGroup (uint16_t valve )`

Gets the valve manual group

**Parameters**

<i>valve</i>	The valve number
--------------	------------------

**Returns**

The manual valve group

**11.125.3.24 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

**11.125.3.25 GetValveMode()** `WvcValveModeEnumNet GetValveMode (uint16_t valve )`

Reads the valve mode

**Parameters**

<i>valve</i>	The valve number
--------------	------------------

**Returns**

The valve mode

**11.125.3.26 GetValvesActiveMap()** `uint32_t GetValvesActiveMap ( )`

Gets the valves active/inactive states

**Returns**

The valves states

**11.125.3.27 GetValvesManualStateMap()** `uint32_t GetValvesManualStateMap ( )`

Gets the valves manual on/off states

**Returns**

The manual valves states

**11.125.3.28 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

**11.125.3.29 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



**11.125.3.30 IsValveDigitalInInverted()** `bool IsValveDigitalInInverted ( uint16_t valve )`

Is digital in inverted

#### Parameters

<i>valve</i>	The valve number
--------------	------------------

#### Returns

is inverted

**11.125.3.31 IsValveOpen()** `bool IsValveOpen ( uint16_t valve )`

Is valve open

#### Parameters

<i>valve</i>	The valve number
--------------	------------------

#### Returns

is open

**11.125.3.32 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

**11.125.3.33 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

**11.125.3.34 LoadValveTable()** `void LoadValveTable ( )`

Load the current table from permanent memory

**11.125.3.35 OnGetActiveRunningTableNumber()** `delegate void OnGetActiveRunningTableNumber ( uint32_t tableNumber )`**11.125.3.36 OnGetAnalogThresholdHigh()** `delegate void OnGetAnalogThresholdHigh ( uint16_t valve, int32_t threshold )`**11.125.3.37 OnGetAnalogThresholdLow()** `delegate void OnGetAnalogThresholdLow ( uint16_t valve, int32_t threshold )`**11.125.3.38 OnGetAnalogVoltage()** `delegate void OnGetAnalogVoltage ( int32_t voltage )`**11.125.3.39 OnGetCurrentNumberOfValves()** `delegate void OnGetCurrentNumberOfValves ( int32_t numberOfValves )`**11.125.3.40 OnGetDigitalOutPortValve()** `delegate void OnGetDigitalOutPortValve ( uint16_t digitalOutPort, uint32_t valve )`

**11.125.3.41 OnGetDigitalPortDirection()** delegate void OnGetDigitalPortDirection (   
 uint16\_t *port*,   
 PortDirectionEnumNet *direction* )

**11.125.3.42 OnGetDisplayMode()** delegate void OnGetDisplayMode (   
 WvcDisplayModeEnumNet *DisplayMode* )

**11.125.3.43 OnGetTableNamebyIndex()** delegate void OnGetTableNamebyIndex (   
 uint16\_t *tableNumber*,   
 String^ *tableName* )

**11.125.3.44 OnGetValveActive()** delegate void OnGetValveActive (   
 uint16\_t *valve*,   
 int *valveActive* )

**11.125.3.45 OnGetValveBoardRevision()** delegate void OnGetValveBoardRevision (   
 uint32\_t *revision* )

**11.125.3.46 OnGetValveDigitalInPort()** delegate void OnGetValveDigitalInPort (   
 uint16\_t *valve*,   
 uint32\_t *digitalInPort* )

**11.125.3.47 OnGetValveLedOn()** delegate void OnGetValveLedOn (   
 bool *ledon* )

**11.125.3.48 OnGetValveManualGroup()** delegate void OnGetValveManualGroup (   
 uint16\_t *valve*,   
 int32\_t *valveManualGroup* )

**11.125.3.49 OnGetValveManualState()** delegate void OnGetValveManualState (   
 uint16\_t *valve*,   
 int32\_t *valveManualState* )

**11.125.3.50 OnGetValveMode()** delegate void OnGetValveMode (   
 uint16\_t valve,   
 WvcValveModeEnumNet ValveMode )

**11.125.3.51 OnIsDigitalOutPortInverted()** delegate void OnIsDigitalOutPortInverted (   
 uint16\_t digitalOutPort,   
 bool isInverted )

**11.125.3.52 OnIsValveDigitalInInverted()** delegate void OnIsValveDigitalInInverted (   
 uint16\_t valve,   
 bool isInverted )

**11.125.3.53 OnIsValveOpen()** delegate void OnIsValveOpen (   
 uint16\_t valve,   
 bool valveOpen )

**11.125.3.54 OnIsValveOpenInAnalogMode()** delegate void OnIsValveOpenInAnalogMode (   
 uint16\_t valve,   
 bool valveOpen )

**11.125.3.55 OnIsValveOpenInDigitalMode()** delegate void OnIsValveOpenInDigitalMode (   
 uint16\_t valve,   
 bool valveOpen )

**11.125.3.56 OnTableEntryChanged()** delegate void OnTableEntryChanged (   
 uint16\_t tableNumber )

**11.125.3.57 SetActiveRunningTableNumber()** void SetActiveRunningTableNumber (   
 uint32\_t tableNumber )

Sets the number of the tanle that is active for running

#### Parameters

<i>tableNumber</i>	The table number
--------------------	------------------

**11.125.3.58 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

**11.125.3.59 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

**11.125.3.60 SetCurrentEditTableNumber()** `void SetCurrentEditTableNumber (`  
    `uint32_t tableNumber )`

Sets the number of the table that is current for editing

Parameters

<i>tableNumber</i>	The table number
--------------------	------------------

**11.125.3.61 SetDefault()** `void SetDefault ( )`

Sets the settings of the valve controller to default

**11.125.3.62 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

**11.125.3.63 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

**11.125.3.64 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

**11.125.3.65 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

**11.125.3.66 SetTableName()** void SetTableName (  
    String^ *tableName* )

Set the name of the current protocol table

#### Parameters

<i>tableName</i>	The name of the table
------------------	-----------------------

**11.125.3.67 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

**11.125.3.68 SetTableStepAll()** `void SetTableStepAll (`  
    `int32_t steps )`

Skips the table protocol for all valves by steps

#### Parameters

<i>steps</i>	Number of steps
--------------	-----------------

**11.125.3.69 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

**11.125.3.70 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

**11.125.3.71 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

**11.125.3.72 SetValveLedOn()** `void SetValveLedOn (`  
    `bool ledon )`

Gets the LED state of the valve board

#### Parameters

<i>ledon</i>	The LED state
--------------	---------------

**11.125.3.73 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

**11.125.3.74 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

**11.125.3.75 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

**11.125.3.76 SetValvesActiveMap()** void SetValvesActiveMap (   
     uint32\_t *valvesActive* )

Sets the valve active/inactive state

Parameters

<i>valvesActive</i>	The valves states
---------------------	-------------------

**11.125.3.77 SetValvesManualStateMap()** void SetValvesManualStateMap (   
     uint32\_t *valveaManualState* )

Sets the valve manual on/off state

Parameters

<i>valveaManualState</i>	The manual valves states
--------------------------	--------------------------

**11.125.3.78 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

#### 11.125.3.79 StoreValveTable() `void StoreValveTable ( )`

Store the current table in permanent memory

### 11.125.4 Event Documentation

#### 11.125.4.1 GetActiveRunningTableNumberEvent `OnGetActiveRunningTableNumber^ GetActiveRunningTableNumberEvent [add], [remove], [raise]`

Event fires when the table number has changed

#### 11.125.4.2 GetAnalogThresholdHighEvent `OnGetAnalogThresholdHigh^ GetAnalogThresholdHighEvent [add], [remove], [raise]`

Event fires when the threshold in mV for the valve number has changed

#### 11.125.4.3 GetAnalogThresholdLowEvent `OnGetAnalogThresholdLow^ GetAnalogThresholdLowEvent [add], [remove], [raise]`

Event fires when the threshold in mV for the valve number has changed

#### 11.125.4.4 GetAnalogVoltageEvent `OnGetAnalogVoltage^ GetAnalogVoltageEvent [add], [remove], [raise]`

Event fires when the voltage in mV has changed

**11.125.4.5 GetCurrentNumberOfValvesEvent** [OnGetCurrentNumberOfValves](#)<sup>^</sup> [GetCurrentNumberOfValvesEvent](#) [add], [remove], [raise]

Event fires when the number of valves has changed

**11.125.4.6 GetDigitalOutPortValveEvent** [OnGetDigitalOutPortValve](#)<sup>^</sup> [GetDigitalOutPortValveEvent](#) [add], [remove], [raise]

Event fires when the valve number for the digital out port has changed

**11.125.4.7 GetDigitalPortDirectionEvent** [OnGetDigitalPortDirection](#)<sup>^</sup> [GetDigitalPortDirectionEvent](#) [add], [remove], [raise]

Event fires when the direction for the port number has changed

**11.125.4.8 GetDisplayModeEvent** [OnGetDisplayMode](#)<sup>^</sup> [GetDisplayModeEvent](#) [add], [remove], [raise]

Event fires when the display mode has changed

**11.125.4.9 GetTableNamebyIndexEvent** [OnGetTableNamebyIndex](#)<sup>^</sup> [GetTableNamebyIndexEvent](#) [add], [remove], [raise]

Event fires when the name of the table for the table number has changed

**11.125.4.10 GetValveActiveEvent** [OnGetValveActive](#)<sup>^</sup> [GetValveActiveEvent](#) [add], [remove], [raise]

Event fires when the valve state for the valve number has changed

**11.125.4.11 GetValveBoardRevisionEvent** [OnGetValveBoardRevision](#)<sup>^</sup> [GetValveBoardRevisionEvent](#) [add], [remove], [raise]

Event fires when the revision code has changed

**11.125.4.12 GetValveDigitalInPortEvent** [OnGetValveDigitalInPort](#)<sup>^</sup> GetValveDigitalInPortEvent [add], [remove], [raise]

Event fires when the digital in port for the valve number has changed

**11.125.4.13 GetValveLedOnEvent** [OnGetValveLedOn](#)<sup>^</sup> GetValveLedOnEvent [add], [remove], [raise]

Event fires when the LED state has changed

**11.125.4.14 GetValveManualGroupEvent** [OnGetValveManualGroup](#)<sup>^</sup> GetValveManualGroupEvent [add], [remove], [raise]

Event fires when the manual valve group for the valve number has changed

**11.125.4.15 GetValveManualStateEvent** [OnGetValveManualState](#)<sup>^</sup> GetValveManualStateEvent [add], [remove], [raise]

Event fires when the manual valve state for the valve number has changed

**11.125.4.16 GetValveModeEvent** [OnGetValveMode](#)<sup>^</sup> GetValveModeEvent [add], [remove], [raise]

Event fires when the valve mode for the valve number has changed

**11.125.4.17 IsDigitalOutPortInvertedEvent** [OnIsDigitalOutPortInverted](#)<sup>^</sup> IsDigitalOutPortInverted↔ Event [add], [remove], [raise]

Event fires when is inverted for the digital out port has changed

**11.125.4.18 IsValveDigitalInInvertedEvent** [OnIsValveDigitalInInverted](#)<sup>^</sup> IsValveDigitalInInverted↔ Event [add], [remove], [raise]

Event fires when is inverted for the valve number has changed

**11.125.4.19 IsValveOpenEvent** [OnIsValveOpen](#)<sup>^</sup> IsValveOpenEvent [add], [remove], [raise]

Event fires when is open for the valve number has changed

**11.125.4.20 IsValveOpenInAnalogModeEvent** [OnIsValveOpenInAnalogMode](#)<sup>^</sup> [IsValveOpenInAnalogMode](#)↔  
Event [add], [remove], [raise]

Event fires when is open for the valve number has changed

**11.125.4.21 IsValveOpenInDigitalModeEvent** [OnIsValveOpenInDigitalMode](#)<sup>^</sup> [IsValveOpenInDigital](#)↔  
ModeEvent [add], [remove], [raise]

Event fires when is open for the valve number has changed

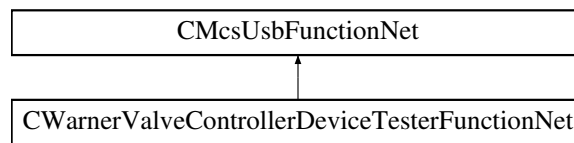
**11.125.4.22 TableEntryChangedEvent** [OnTableEntryChanged](#)<sup>^</sup> [TableEntryChangedEvent](#) [add], [remove], [raise]

Event fires when an entry of a table changed

## 11.126 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

### 11.126.1 Detailed Description

[CWarnerValveControllerDeviceTesterFunctionNet](#) is the class to access the functions for the Warner Valve Controller Device Tester

### 11.126.2 Constructor & Destructor Documentation

**11.126.2.1 CWarnerValveControllerDeviceTesterFunctionNet() [1/2]** [CWarnerValveControllerDeviceTesterFunctionNet](#)  
(  
    [CMcsUsbNet](#)<sup>^</sup> *mcsusb*,  
    [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> *pWarnerValveControllerDeviceTesterFunction*↔  
    *PointerContainer* )

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

**11.126.2.2 CWarnerValveControllerDeviceTesterFunctionNet() [2/2]** [CWarnerValveControllerDeviceTesterFunctionNet](#)  
(  
    [CMcsUsbNet](#)<sup>^</sup> *mcsusb* )

**11.126.2.3 ~CWarnerValveControllerDeviceTesterFunctionNet()** virtual [~CWarnerValveControllerDeviceTesterFunctionNet](#)  
( ) [virtual]

**11.126.2.4 "!CWarnerValveControllerDeviceTesterFunctionNet()** [!CWarnerValveControllerDeviceTesterFunctionNet](#)  
( )

### 11.126.3 Member Function Documentation

**11.126.3.1 GetIO()** uint32\_t GetIO ( )

Gets the output from the io ports

#### Returns

The manual valves states

**11.126.3.2 GetSync()** `uint32_t GetSync ( )`

Gets the output from the sync port

**Returns**

The sync state

**11.126.3.3 SetADC()** `void SetADC (   
uint32_t onoff )`

Sets the ADC port of the tester

**Parameters**

<i>onoff</i>	The port state
--------------	----------------

**11.126.3.4 SetIO()** `void SetIO (   
uint32_t io )`

Sets the input to the io ports

**Parameters**

<i>io</i>	The manual valves states
-----------	--------------------------

**11.126.3.5 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
------------------	-------------------------------------

**11.126.3.6 SetTrigger()** `void SetTrigger (   
uint32_t trigger )`

Sets the input to the trigger port

## Parameters

<i>trigger</i>	The trigger state
----------------	-------------------

**11.126.3.7 SetTriggerSyncDirection()** `void SetTriggerSyncDirection ( uint32_t direction )`

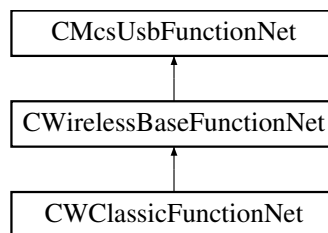
Sets the direction of the trigger/sync test port

## Parameters

<i>direction</i>	The direction: 1=IN 0=OUT
------------------	---------------------------

## 11.127 CWClassicFunctionNet Class Reference

Inheritance diagram for CWClassicFunctionNet:



### Public Member Functions

- [CWClassicFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> wClassicFunctionPointerContainer)
- [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

### 11.127.1 Constructor & Destructor Documentation

**11.127.1.1 CWClassicFunctionNet()** [1/2] [CWClassicFunctionNet](#) (  
[CMcsUsbNet](#) ^ *mcsusb*,  
[CMcsUsbFunctionPointerContainer](#) ^ *wClassicFuntionPointerContainer* )

**11.127.1.2 CWClassicFunctionNet()** [2/2] [CWClassicFunctionNet](#) (  
[CMcsUsbNet](#) ^ *mcsusb* )

### 11.127.2 Member Function Documentation

**11.127.2.1 GetFilterParametersHeadstage()** array<int> ^ [GetFilterParametersHeadstage](#) (  
unsigned short *index* )

**11.127.2.2 GetHasChecksum()** unsigned int [GetHasChecksum](#) (  
unsigned int *Device* )

**11.127.2.3 GetHasRedLedHeadstage()** bool [GetHasRedLedHeadstage](#) ( )

**11.127.2.4 GetHeadstageOnOff()** `USHORT GetHeadstageOnOff ( )`

**11.127.2.5 GetResetFilter()** `unsigned int GetResetFilter (`  
`unsigned int Device )`

**11.127.2.6 GetRFConnectionStatus()** `unsigned int GetRFConnectionStatus ( )`

**11.127.2.7 GetRFFrequencyHeadstage()** `unsigned short GetRFFrequencyHeadstage (`  
`uint8_t receiver_nb )`

**11.127.2.8 GetRFFrequencyReceiver()** `unsigned short GetRFFrequencyReceiver (`  
`uint8_t receiver_nb,`  
`uint8_t configuration )`

**11.127.2.9 GetRFPower()** `unsigned short GetRFPower ( )`

**11.127.2.10 GetScanHeadstagesResult()** `uint8_t GetScanHeadstagesResult (`  
`int max_wait_for_ms )`

**11.127.2.11 GetSelectedHeadstage()** `uint8_t GetSelectedHeadstage ( )`

**11.127.2.12 GetSerialNumberHeadstage()** `unsigned short GetSerialNumberHeadstage ( )`

**11.127.2.13 GetWPADebugMode()** `unsigned int GetWPADebugMode (`  
`unsigned int Device )`

**11.127.2.14 GetWPAType()** unsigned short GetWPAType (   
 unsigned int *Device* )

**11.127.2.15 ResetChannelmap()** uint32\_t ResetChannelmap (   
 unsigned int *virtualDevice* )

**11.127.2.16 ScanForHeadstages()** void ScanForHeadstages ( )

**11.127.2.17 SetChannelmap()** uint32\_t SetChannelmap (   
 unsigned char *position*,   
 unsigned char *channel*,   
 unsigned int *Device* )

**11.127.2.18 SetFilterParametersHeadstage()** void SetFilterParametersHeadstage (   
 unsigned short *index*,   
 array< int >^ *buffer* )

**11.127.2.19 SetHasChecksum()** void SetHasChecksum (   
 unsigned int *has*,   
 unsigned int *Device* )

**11.127.2.20 SetHeadstageOnOff()** void SetHeadstageOnOff (   
 uint16\_t *onoff* )

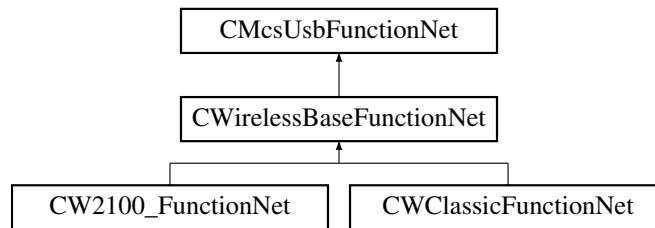
**11.127.2.21 SetHWSelectedChannels()** void SetHWSelectedChannels (   
 array< bool >^ *channels*,   
 unsigned int *Device* )

**11.127.2.22 SetResetFilter()** void SetResetFilter (   
 unsigned int *reset*,   
 unsigned int *Device* )

- 11.127.2.23 SetRFFrequencyHeadstage()** void SetRFFrequencyHeadstage (   
     uint8\_t *receiver\_nb*,   
     unsigned short *frequency* )
- 11.127.2.24 SetRFFrequencyReceiver()** void SetRFFrequencyReceiver (   
     uint8\_t *receiver\_nb*,   
     uint8\_t *configuration*,   
     unsigned short *frequency* )
- 11.127.2.25 SetRFFrequencyReceiverEeprom()** void SetRFFrequencyReceiverEeprom (   
     uint8\_t *receiver\_nb*,   
     uint8\_t *configuration*,   
     unsigned short *frequency* )
- 11.127.2.26 SetRFLostBehaviour()** void SetRFLostBehaviour (   
     uint8\_t *stoponfailure*,   
     unsigned int *Device* )
- 11.127.2.27 SetRFPower()** void SetRFPower (   
     unsigned short *power* )
- 11.127.2.28 SetSelectedHeadstage()** void SetSelectedHeadstage (   
     uint8\_t *number* )
- 11.127.2.29 SetSerialNumberHeadstage()** void SetSerialNumberHeadstage (   
     unsigned short *number* )
- 11.127.2.30 SetWPADebugMode()** void SetWPADebugMode (   
     unsigned int *mode*,   
     unsigned int *Device* )
- 11.127.2.31 SetWPAType()** void SetWPAType (   
     unsigned short *type*,   
     unsigned int *Device* )

## 11.128 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

#### 11.128.1 Constructor & Destructor Documentation

**11.128.1.1 CWirelessBaseFunctionNet()** [CWirelessBaseFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb,  
[CMcsUsbFunctionPointerContainer](#)<sup>^</sup> mcsusbfunction )

#### 11.128.2 Member Function Documentation

**11.128.2.1 CreateWirelessHeadstageSerialNumberString()** static String <sup>^</sup> [CreateWirelessHeadstageSerialNumberString](#) (  
 unsigned short ID ) [static]

## 11.129 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](#)

### 11.129.1 Detailed Description

Device Id.

### 11.129.2 Constructor & Destructor Documentation

**11.129.2.1 DeviceIdNet()** [1/3] [DeviceIdNet](#) ( )

**11.129.2.2 DeviceIdNet()** [2/3] [DeviceIdNet](#) (  
VendorIdEnumNet *vendor*,  
ProductIdEnumNet *product*,  
int *bcd*,  
McsBusTypeEnumNet *bustype* )

**11.129.2.3 DeviceIdNet()** [3/3] [DeviceIdNet](#) (  
[DeviceIdNet](#)% *deviceId* )

### 11.129.3 Member Function Documentation

**11.129.3.1 operator=()** [DeviceIdNet](#) operator= (  
[DeviceIdNet](#)% *deviceId* )

#### 11.129.4 Member Data Documentation

**11.129.4.1 BcdDevice** `int BcdDevice`

**11.129.4.2 BusType** `McsBusTypeEnumNet BusType`

**11.129.4.3 IdProduct** `ProductIdEnumNet IdProduct`

**11.129.4.4 IdVendor** `VendorIdEnumNet IdVendor`

### 11.130 DigitalSource< digitalsourceenum > Class Template Reference

#### Public Member Functions

- [DigitalSource](#) ()
- [DigitalSource](#) (digitalsourceenum source)
- int [MaxBitNumber](#) ()
- int [MaxBitNumber](#) (digitalsourceenum [Source](#))

#### Static Public Member Functions

- static int [MaxBitNumberStatic](#) (digitalsourceenum [Source](#))
- static int [size](#) ()

#### Properties

- digitalsourceenum [Source](#) [get, set]

#### 11.130.1 Constructor & Destructor Documentation

**11.130.1.1 DigitalSource()** [1/2] [DigitalSource](#) ( )

**11.130.1.2 DigitalSource()** [2/2] [DigitalSource](#) (  
digitalsourceenum *source* )

## 11.130.2 Member Function Documentation

**11.130.2.1 MaxBitNumber()** [1/2] int MaxBitNumber ( )

**11.130.2.2 MaxBitNumber()** [2/2] int MaxBitNumber (  
digitalsourceenum *Source* )

**11.130.2.3 MaxBitNumberStatic()** static int MaxBitNumberStatic (  
digitalsourceenum *Source* ) [static]

**11.130.2.4 size()** static int size ( ) [static]

## 11.130.3 Property Documentation

**11.130.3.1 Source** digitalsourceenum *Source* [get], [set]

## 11.131 DigitalSourceGeneral Class Reference

### Public Member Functions

- [DigitalSourceGeneral](#) (Type<sup>^</sup> type)
- [DigitalSourceGeneral](#) (Type<sup>^</sup> type, int [Source](#))
- int [MaxBitNumber](#) ()
- int [MaxBitNumber](#) (int [Source](#))

### Static Public Member Functions

- static int [MaxBitNumber](#) (Type<sup>^</sup> type, int [Source](#))
- static int [size](#) (Type<sup>^</sup> type)



## Properties

- int `Source` [get, set]

## 11.131.1 Constructor & Destructor Documentation

**11.131.1.1 DigitalSourceGeneral()** [1/2] `DigitalSourceGeneral (`  
    *Type^ type* )

**11.131.1.2 DigitalSourceGeneral()** [2/2] `DigitalSourceGeneral (`  
    *Type^ type*,  
    int *Source* )

## 11.131.2 Member Function Documentation

**11.131.2.1 MaxBitNumber()** [1/3] `int MaxBitNumber ( )`

**11.131.2.2 MaxBitNumber()** [2/3] `int MaxBitNumber (`  
    int *Source* )

**11.131.2.3 MaxBitNumber()** [3/3] `static int MaxBitNumber (`  
    *Type^ type*,  
    int *Source* ) [static]

**11.131.2.4 size()** `static int size (`  
    *Type^ type* ) [static]

## 11.131.3 Property Documentation

**11.131.3.1 Source** `int Source [get], [set]`

## 11.132 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.*
- String ^ [GetSerialNumber](#) (CFirmwareDestinationNet dest)  
*Get the serial number of the destination, when no serial number if found, return an empty string.*
- String ^ [GetSerialNumber](#) (unsigned int index)  
*Get the serial number of the destination, when no serial number if found, return an empty string.*

### Static Public Member Functions

- static String ^ [DriverVersionNet::FormatVersion](#) (unsigned int v)

### 11.132.1 Detailed Description

Class gives firmware versions of the device's firmware destinations.

### 11.132.2 Constructor & Destructor Documentation

#### 11.132.2.1 DriverVersionNet() `DriverVersionNet ( )`

Constructor.

#### 11.132.2.2 ~DriverVersionNet() `~DriverVersionNet ( )`

Destructor.

### 11.132.3 Member Function Documentation

#### 11.132.3.1 DriverVersionNet::FormatVersion() `static String ^ DriverVersionNet::FormatVersion ( unsigned int v ) [static]`

#### 11.132.3.2 GetDestinationCode() `CFirmwareDestinationNet GetDestinationCode ( unsigned int index )`

Get CFirmwareDestinationNet.

##### Parameters

<i>index</i>	by index of firmware destination
--------------	----------------------------------

#### 11.132.3.3 GetDestinationName() [1/2] `String ^ GetDestinationName ( CFirmwareDestinationNet dest )`

Get firmware destination name.

##### Parameters

<i>dest</i>	by CFirmwareDestinationNet
-------------	----------------------------

**11.132.3.4 GetDestinationName()** [2/2] `String ^ GetDestinationName ( unsigned int index )`

Get firmware destination name.

**Parameters**

<i>index</i>	by index of firmware destination
--------------	----------------------------------

**11.132.3.5 GetMajor()** [1/2] `unsigned int GetMajor ( CFirmwareDestinationNet dest )`

Get the major version number of firmware destination.

**Parameters**

<i>dest</i>	by CFirmwareDestinationNet
-------------	----------------------------

**11.132.3.6 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
--------------	----------------------------------

**11.132.3.7 GetMinor()** [1/2] `unsigned int GetMinor ( CFirmwareDestinationNet dest )`

Get the minor version number of firmware destination.

**Parameters**

<i>dest</i>	by CFirmwareDestinationNet
-------------	----------------------------

**11.132.3.8 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
--------------	----------------------------------

**11.132.3.9 GetNumEntries()** `unsigned int GetNumEntries ( )`

Get the number of available firmware destinations.

**11.132.3.10 GetSerialNumber()** [1/2] `String ^ GetSerialNumber ( CFirmwareDestinationNet dest )`

Get the serial number of the destination, when no serial number if found, return an empty string.

Parameters

<i>dest</i>	by CFirmwareDestinationNet
-------------	----------------------------

**11.132.3.11 GetSerialNumber()** [2/2] `String ^ GetSerialNumber ( unsigned int index )`

Get the serial number of the destination, when no serial number if found, return an empty string.

Parameters

<i>index</i>	by index of firmware destination
--------------	----------------------------------

**11.132.3.12 GetStatus()** [1/2] `unsigned int GetStatus ( CFirmwareDestinationNet dest )`

Get status of firmware destination.

Parameters

<i>dest</i>	by CFirmwareDestinationNet
-------------	----------------------------

**11.132.3.13 GetStatus()** [2/2] unsigned int GetStatus (   
 unsigned int *index* )

Get status of firmware destination.

Parameters

<i>index</i>	by index of firmware destination
--------------	----------------------------------

**11.132.3.14 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
-------------	----------------------------

**11.132.3.15 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
--------------	----------------------------------

**11.132.3.16 GetVersionString()** [1/2] String ^ GetVersionString (   
 CFirmwareDestinationNet *dest* )

Get the version as a string in the format Major.Minor.

Parameters

<i>dest</i>	by CFirmwareDestinationNet
-------------	----------------------------

**11.132.3.17 GetVersionString()** [2/2] String ^ GetVersionString (   
 unsigned int *index* )

Get the version as a string in the format Major.Minor.

## Parameters

<a href="#">index</a>	by index of firmware
-----------------------	----------------------

## 11.133 FirmwareDestinationNames Class Reference

## Static Public Attributes

- static String ^ **DSP** = gcnew String( "DSP" )
- static String ^ **USB** = gcnew String( "USB" )
- static String ^ **MCU1** = gcnew String( "MCU1" )
- static String ^ **Bootstrap** = gcnew String( "Bootstrap" )
- static String ^ **MCSBUS1** = gcnew 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" )

## 11.133.1 Member Data Documentation

**11.133.1.1 Altera** String ^ Altera = gcnew String( "Altera" ) [static]

**11.133.1.2 Bootstrap** String ^ Bootstrap = gcnew String( "Bootstrap" ) [static]

**11.133.1.3 BUS1\_MCSBUS1** `String ^ BUS1_MCSBUS1 = gcnew String( "Bus1McsBus1" ) [static]`

**11.133.1.4 BUS1\_MCSBUS2** `String ^ BUS1_MCSBUS2 = gcnew String( "Bus1McsBus2" ) [static]`

**11.133.1.5 DSP** `String ^ DSP = gcnew String( "DSP" ) [static]`

**11.133.1.6 FPGA2** `String ^ FPGA2 = gcnew String( "FPGA2" ) [static]`

**11.133.1.7 FPGA3** `String ^ FPGA3 = gcnew String( "FPGA3" ) [static]`

**11.133.1.8 FPGA4** `String ^ FPGA4 = gcnew String( "FPGA4" ) [static]`

**11.133.1.9 FPGA5** `String ^ FPGA5 = gcnew String( "FPGA5" ) [static]`

**11.133.1.10 FPGA6** `String ^ FPGA6 = gcnew String( "FPGA6" ) [static]`

**11.133.1.11 MCSBUS1** `String ^ MCSBUS1 = gcnew String( "McsBus1" ) [static]`

**11.133.1.12 MCSBUS10** `String ^ MCSBUS10 = gcnew String( "McsBus10" ) [static]`

**11.133.1.13 MCSBUS11** `String ^ MCSBUS11 = gcnew String( "McsBus11" ) [static]`



**11.133.1.14 MCSBUS12** `String ^ MCSBUS12 = gcnew String( "McsBus12" ) [static]`

**11.133.1.15 MCSBUS13** `String ^ MCSBUS13 = gcnew String( "McsBus13" ) [static]`

**11.133.1.16 MCSBUS2** `String ^ MCSBUS2 = gcnew String( "McsBus2" ) [static]`

**11.133.1.17 MCSBUS3** `String ^ MCSBUS3 = gcnew String( "McsBus3" ) [static]`

**11.133.1.18 MCSBUS4** `String ^ MCSBUS4 = gcnew String( "McsBus4" ) [static]`

**11.133.1.19 MCSBUS5** `String ^ MCSBUS5 = gcnew String( "McsBus5" ) [static]`

**11.133.1.20 MCSBUS6** `String ^ MCSBUS6 = gcnew String( "McsBus6" ) [static]`

**11.133.1.21 MCSBUS7** `String ^ MCSBUS7 = gcnew String( "McsBus7" ) [static]`

**11.133.1.22 MCSBUS8** `String ^ MCSBUS8 = gcnew String( "McsBus8" ) [static]`

**11.133.1.23 MCSBUS9** `String ^ MCSBUS9 = gcnew String( "McsBus9" ) [static]`

**11.133.1.24 MCU1** `String ^ MCU1 = gcnew String( "MCU1" ) [static]`

**11.133.1.25 PIC** `String ^ PIC = gnew String( "PIC" ) [static]`

**11.133.1.26 PIC2** `String ^ PIC2 = gnew String( "PIC2" ) [static]`

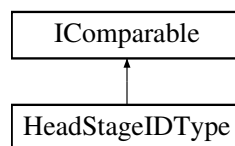
**11.133.1.27 PIC3** `String ^ PIC3 = gnew String( "PIC3" ) [static]`

**11.133.1.28 PIC4** `String ^ PIC4 = gnew String( "PIC4" ) [static]`

**11.133.1.29 USB** `String ^ USB = gnew String( "USB" ) [static]`

## 11.134 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]

## 11.134.1 Member Enumeration Documentation

### 11.134.1.1 HeadstageTypeEnum `enum HeadstageTypeEnum [strong]`

Enumerator

Unknown	
MeasuringOnly	
OpticalStimulation	
ElectricalStimulation	

## 11.134.2 Constructor & Destructor Documentation

### 11.134.2.1 HeadStageIDType() `HeadStageIDType ( unsigned int entry, CW2100_FunctionNet^ device )`

## 11.134.3 Member Function Documentation

### 11.134.3.1 CompareTo() `virtual Int32 CompareTo ( Object^ obj ) [virtual]`

**11.134.3.2 Equals()** virtual bool Equals (   
 Object^ *obj* ) [override], [virtual]

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

#### 11.134.4 Property Documentation

**11.134.4.1 Entry** unsigned int Entry [get]

**11.134.4.2 HasIMU** bool HasIMU [get]

**11.134.4.3 HasOptoCurrentMessurement** bool HasOptoCurrentMessurement [get]

**11.134.4.4 HeadstageType** [HeadstageTypeEnum](#) HeadstageType [get]

**11.134.4.5 ID** unsigned short ID [get]

**11.134.4.6 NumberOfAnalogChannels** int NumberOfAnalogChannels [get]

**11.134.4.7 NumberOfStimulationChannels** int NumberOfStimulationChannels [get]

**11.134.4.8 SN** System:: String^ SN [get]

**11.134.4.9 StimulusParameters** [W2100\\_StimulusParametersNet](#)<sup>^</sup> StimulusParameters [get]

**11.134.4.10 Type** System:: String<sup>^</sup> Type [get]

**11.134.4.11 TypeValue** unsigned int TypeValue [get]

**11.134.4.12 UserDefinedName** System:: String<sup>^</sup> UserDefinedName [get]

**11.134.4.13 Valid** bool Valid [get]

**11.134.4.14 W16IsW14** bool W16IsW14 [get]

## 11.135 HeadstageIDTypeObject Class Reference

### Public Member Functions

- [HeadstageIDTypeObject](#) ([HeadStageIDType](#)<sup>^</sup> idType)
- virtual String<sup>^</sup> [ToString](#) () override
- virtual bool [Equals](#) (Object<sup>^</sup> obj) override
- virtual int [GetHashCode](#) () override

### Public Attributes

- [HeadStageIDType](#)<sup>^</sup> [\\_IdType](#)
- String<sup>^</sup> [\\_AdditionalText](#)

### Properties

- [HeadStageIDType](#)<sup>^</sup> [IdType](#) [get]
- String<sup>^</sup> [AdditionalText](#) [get, set]

### 11.135.1 Constructor & Destructor Documentation

**11.135.1.1 HeadstageIDTypeObject()** `HeadstageIDTypeObject ( HeadStageIDType^ idType )`

## 11.135.2 Member Function Documentation

**11.135.2.1 Equals()** `virtual bool Equals ( Object^ obj ) [override], [virtual]`

**11.135.2.2 GetHashCode()** `virtual int GetHashCode ( ) [override], [virtual]`

**11.135.2.3 ToString()** `virtual String ^ ToString ( ) [override], [virtual]`

## 11.135.3 Member Data Documentation

**11.135.3.1 \_AdditionalText** `String ^ _AdditionalText`

**11.135.3.2 \_IdType** `HeadStageIDType ^ _IdType`

## 11.135.4 Property Documentation

**11.135.4.1 AdditionalText** `String^ AdditionalText [get], [set]`

**11.135.4.2 IdType** `HeadStageIDType^ IdType [get]`

## 11.136 HeadStageIDTypeState Class Reference

### Properties

- unsigned int `State` [get]
- `HeadStageIDType^ IdType` [get]
- bool `ControlState` [get]
- bool `DataState` [get]

### 11.136.1 Property Documentation

**11.136.1.1 ControlState** `bool ControlState [get]`

**11.136.1.2 DataState** `bool DataState [get]`

**11.136.1.3 IdType** `HeadStageIDType^ IdType [get]`

**11.136.1.4 State** `unsigned int State [get]`

## 11.137 mkfilterNet Class Reference

### Static Public Member Functions

- static int [mkfilter](#) (String^ filtertype, double value, String^ passtype, int order, double alpha1, double alpha2, [System::Runtime::InteropServices::Out] array< double >^% xcoeffs, [System::Runtime::InteropServices::Out] array< double >^% ycoeffs)
- 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 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 int [mkfilter\\_MCS\\_k](#) (int SamplesPerSecond, double R1, double R2, double C, double Amplification, double Correction, [System::Runtime::InteropServices::Out] array< double >^% coeffs)
- static int [mkfilter\\_MCS\\_k](#) (int SamplesPerSecond, double R1, double R2, double C, double Correction, [System::Runtime::InteropServices::Out] array< double >^% coeffs)
- 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 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 void [mkfilter\\_normalize\\_coeffs\\_short](#) (short maxvalue, [System::Runtime::InteropServices::In] array< double >^ coeffs, [System::Runtime::InteropServices::Out] array< short >^% out\_coeffs)
- static void [mkfilter\\_normalize\\_coeffs\\_int](#) (int maxvalue, [System::Runtime::InteropServices::In] array< double >^ coeffs, [System::Runtime::InteropServices::Out] array< int >^% out\_coeffs)
- 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 double [mkfilter\\_highpass\\_coef](#) (int SamplesPerSecond, double Frequency)
- static double [mkfilter\\_highpass\\_k](#) (int SamplesPerSecond, double Frequency)
- static double [mkfilter\\_highpass\\_frequency\\_from\\_coef](#) (int SamplesPerSecond, double coeff)
- static double [mkfilter\\_highpass\\_frequency\\_from\\_k](#) (int SamplesPerSecond, double k)

### 11.137.1 Member Function Documentation

**11.137.1.1 mkfilter()** static int mkfilter (   
String^ *filtertype*,  
double *value*,  
String^ *passtype*,  
int *order*,  
double *alpha1*,  
double *alpha2*,  
[System::Runtime::InteropServices::Out] array< double >^% *xcoeffs*,  
[System::Runtime::InteropServices::Out] array< double >^% *ycoeffs* ) [static]

**11.137.1.2 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]

**11.137.1.3 mkfilter\_highpass\_coeff()** static double mkfilter\_highpass\_coeff (   
int *SamplesPerSecond*,  
double *Frequency* ) [static]

**11.137.1.4 mkfilter\_highpass\_frequency\_from\_coeff()** static double mkfilter\_highpass\_frequency\_from↵  
from\_coeff (   
int *SamplesPerSecond*,  
double *coeff* ) [static]

**11.137.1.5 mkfilter\_highpass\_frequency\_from\_k()** static double mkfilter\_highpass\_frequency\_from↵  
\_k (   
int *SamplesPerSecond*,  
double *k* ) [static]

**11.137.1.6 mkfilter\_highpass\_k()** static double mkfilter\_highpass\_k (   
int *SamplesPerSecond*,  
double *Frequency* ) [static]



**11.137.1.7 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]

**11.137.1.8 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]

**11.137.1.9 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]

**11.137.1.10 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]

**11.137.1.11 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]

**11.137.1.12 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]

**11.137.1.13 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]

**11.137.1.14 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]

## 11.138 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 RoboMain*
- void [SetUserParameter](#) (unsigned short index, int value)
  - Stores persistently 32 bit integer values on RoboMain*
- 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 RoboMain*
- void [GetUserParameter](#) (unsigned short index, [System::Runtime::InteropServices::Out] int% value)
  - Reads 32 bit integer values stored persistently on RoboMain*
- 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)

### 11.138.1 Member Function Documentation

**11.138.1.1 FindReferencePhase0()** [1/2] void FindReferencePhase0 (   
 unsigned char *busaddress*,   
 char *axes* )

**11.138.1.2 FindReferencePhase0()** [2/2] void FindReferencePhase0 (   
 unsigned char *busaddress*,   
 char *axes*,   
 int *timeout* )

**11.138.1.3 GetAxisConfig()** unsigned int GetAxisConfig (   
 unsigned char *busaddress*,   
 char *axes* )

**11.138.1.4 GetHWConfig()** unsigned int GetHWConfig ( )

**11.138.1.5 GetHWRevision()** unsigned int GetHWRevision ( )

**11.138.1.6 GetMaxNoPressure()** unsigned int GetMaxNoPressure ( )

**11.138.1.7 GetMaxNoPressureWaitTime()** unsigned int GetMaxNoPressureWaitTime ( )

**11.138.1.8 GetMaxPressureWaitTime()** unsigned int GetMaxPressureWaitTime ( )

**11.138.1.9 GetMinNoPressureWaitTime()** unsigned int GetMinNoPressureWaitTime ( )

**11.138.1.10 GetMinPressure()** unsigned int GetMinPressure ( )

**11.138.1.11 GetMinPressureWaitTime()** unsigned int GetMinPressureWaitTime ( )

**11.138.1.12 GetParameter() [1/2]** void GetParameter (   
    unsigned short *command*,   
    unsigned short *index*,   
    [System::Runtime::InteropServices::Out] unsigned int% *value* )

**11.138.1.13 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* )

- 11.138.1.14 GetPhases()** void GetPhases (   
 unsigned char *busaddress*,   
 char *axes*,   
 [System::Runtime::InteropServices::Out] unsigned short% *phase0*,   
 [System::Runtime::InteropServices::Out] unsigned short% *lastphase* )
- 11.138.1.15 GetSearchReferenceFastAccel()** unsigned short GetSearchReferenceFastAccel (   
 unsigned char *busaddress*,   
 char *axes* )
- 11.138.1.16 GetSearchReferenceFastSpeed()** unsigned short GetSearchReferenceFastSpeed (   
 unsigned char *busaddress*,   
 char *axes* )
- 11.138.1.17 GetSearchReferenceFineAccel()** unsigned short GetSearchReferenceFineAccel (   
 unsigned char *busaddress*,   
 char *axes* )
- 11.138.1.18 GetSearchReferenceFineSpeed()** unsigned short GetSearchReferenceFineSpeed (   
 unsigned char *busaddress*,   
 char *axes* )
- 11.138.1.19 GetSearchReferenceMethod()** unsigned int GetSearchReferenceMethod (   
 unsigned char *busaddress*,   
 char *axes* )
- 11.138.1.20 GetSearchReferenceMoveOut()** int GetSearchReferenceMoveOut (   
 unsigned char *busaddress*,   
 char *axes* )
- 11.138.1.21 GetSearchReferenceOffsetPos()** int GetSearchReferenceOffsetPos (   
 unsigned char *busaddress*,   
 char *axes* )
- 11.138.1.22 GetUserParameter()** [1/2] void GetUserParameter (   
 unsigned short *index*,   
 [System::Runtime::InteropServices::Out] int% *value* )

Reads 32 bit integer values stored persistently on RoboMain

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

**11.138.1.23 GetUserParameter()** [2/2] void GetUserParameter (   
 unsigned short *index*,   
 [System::Runtime::InteropServices::Out] unsigned int% *value* )

Reads 32 bit integer values stored persistently on RoboMain

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

**11.138.1.24 HasRef()** unsigned char HasRef (   
 unsigned char *busaddress*,   
 char *axes* )

**11.138.1.25 SetAxisConfig()** void SetAxisConfig (   
 unsigned char *busaddress*,   
 char *axes*,   
 unsigned int *config* )

**11.138.1.26 SetHWConfig()** void SetHWConfig (   
 unsigned int *config* )

**11.138.1.27 SetHWRevision()** void SetHWRevision (   
 unsigned int *revision* )

**11.138.1.28 SetMaxNoPressure()** void SetMaxNoPressure (   
 unsigned int *pressure* )

**11.138.1.29 SetMaxNoPressureWaitTime()** void SetMaxNoPressureWaitTime ( unsigned int *t* )

**11.138.1.30 SetMaxPressureWaitTime()** void SetMaxPressureWaitTime ( unsigned int *t* )

**11.138.1.31 SetMinNoPressureWaitTime()** void SetMinNoPressureWaitTime ( unsigned int *t* )

**11.138.1.32 SetMinPressure()** void SetMinPressure ( unsigned int *pressure* )

**11.138.1.33 SetMinPressureWaitTime()** void SetMinPressureWaitTime ( unsigned int *t* )

**11.138.1.34 SetParameter() [1/2]** void SetParameter ( unsigned short *command*, unsigned short *index*, unsigned int *value* )

**11.138.1.35 SetParameter() [2/2]** void SetParameter ( unsigned short *command*, unsigned short *index*, unsigned int *value1*, unsigned int *value2* )

**11.138.1.36 SetSearchReferenceFastAccel()** void SetSearchReferenceFastAccel ( unsigned char *busaddress*, char *axes*, unsigned short *accel* )

**11.138.1.37 SetSearchReferenceFastSpeed()** void SetSearchReferenceFastSpeed (   
 unsigned char *busaddress*,   
 char *axes*,   
 unsigned short *speed* )

**11.138.1.38 SetSearchReferenceFineAccel()** void SetSearchReferenceFineAccel (   
 unsigned char *busaddress*,   
 char *axes*,   
 unsigned short *accel* )

**11.138.1.39 SetSearchReferenceFineSpeed()** void SetSearchReferenceFineSpeed (   
 unsigned char *busaddress*,   
 char *axes*,   
 unsigned short *speed* )

**11.138.1.40 SetSearchReferenceMethod()** void SetSearchReferenceMethod (   
 unsigned char *busaddress*,   
 char *axes*,   
 unsigned int *method* )

**11.138.1.41 SetSearchReferenceMoveOut()** void SetSearchReferenceMoveOut (   
 unsigned char *busaddress*,   
 char *axes*,   
 int *move* )

**11.138.1.42 SetSearchReferenceOffsetPos()** void SetSearchReferenceOffsetPos (   
 unsigned char *busaddress*,   
 char *axes*,   
 int *offsetpos* )

**11.138.1.43 SetUserParameter()** [1/2] void SetUserParameter (   
 unsigned short *index*,   
 int *value* )

Stores *persistently* 32 bit integer values on RoboMain

intention: provide free persistent user memory space on RoboMain



## Parameters

<i>index</i>	address offset of parameter; range: 0..15
<i>value</i>	data to be stored

**11.138.1.44 SetUserParameter()** [2/2] `void SetUserParameter (`  
     `unsigned short index,`  
     `unsigned int value )`

Stores *persistently* 32 bit integer values on RoboMain

intention: provide free persistent user memory space on RoboMain

## Parameters

<i>index</i>	address offset of parameter; range: 0..15
<i>value</i>	data to be stored

## 11.139 CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands Class Reference

### Public Member Functions

- void [FindReferencePhase0XY](#) ()
- void [FindReferencePhase0XY](#) (int timeout)

### 11.139.1 Member Function Documentation

**11.139.1.1 FindReferencePhase0XY()** [1/2] `void FindReferencePhase0XY ( )`

**11.139.1.2 FindReferencePhase0XY()** [2/2] `void FindReferencePhase0XY (`  
     `int timeout )`

## 11.140 CFilterCoefficientsNet::s\_FilterAttributesNet Struct Reference

### Public Member Functions

- [s\\_FilterAttributesNet](#) (s\_FilterAttributes attrib)
- s\_FilterAttributes [ToCpp](#) ()

## Public Attributes

- uint32\_t [PreCommaB](#)
- uint32\_t [PostCommaB](#)
- uint32\_t [CommaPositionB](#)
- uint32\_t [PreCommaA](#)
- uint32\_t [PostCommaA](#)
- uint32\_t [CommaPositionA](#)

## 11.140.1 Constructor & Destructor Documentation

**11.140.1.1** `s_FilterAttributesNet()` `s_FilterAttributesNet (`  
`s_FilterAttributes attrib )`

## 11.140.2 Member Function Documentation

**11.140.2.1** `ToCpp()` `s_FilterAttributes ToCpp ( )`

## 11.140.3 Member Data Documentation

**11.140.3.1** `CommaPositionA` `uint32_t CommaPositionA`

**11.140.3.2** `CommaPositionB` `uint32_t CommaPositionB`

**11.140.3.3** `PostCommaA` `uint32_t PostCommaA`

**11.140.3.4** `PostCommaB` `uint32_t PostCommaB`

**11.140.3.5 PreCommaA** `uint32_t PreCommaA`

**11.140.3.6 PreCommaB** `uint32_t PreCommaB`

## 11.141 CMeaAudioFunctionNet::s\_setaudionet Struct Reference

### Public Attributes

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

### 11.141.1 Member Data Documentation

**11.141.1.1 amplification** `int amplification`

**11.141.1.2 channel** `int channel`

## 11.142 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]

### 11.142.1 Constructor & Destructor Documentation

**11.142.1.1 SidebandData()** [SidebandData](#) ( )

**11.142.1.2 ~SidebandData()** [~SidebandData](#) ( )

Destructor: called by Dispose()

**11.142.1.3 ~!SidebandData()** [!SidebandData](#) ( )

Finalizer: called by GC before collecting

## 11.142.2 Property Documentation

**11.142.2.1 Duration** [array< uint64\\_t>^ Duration](#) [get]

**11.142.2.2 Sideband** [array< int32\\_t>^ Sideband](#) [get]

## 11.143 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](#)

### 11.143.1 Member Function Documentation

**11.143.1.1 FromIntPtr()** [static StgStatusNet ^ FromIntPtr](#) (   
 IntPtr stgstatus ) [static]

**11.143.1.2 FromPtr()** static [StgStatusNet](#) ^ FromPtr (   
           stgstatus\_t \* stgstatus ) [static]

### 11.143.2 Member Data Documentation

**11.143.2.1 ListOfChangedTriggers** array<uint32\_t> ^ ListOfChangedTriggers

**11.143.2.2 TiggerStatus** array<Stg200xTriggerStatusEnumNet> ^ TiggerStatus

## 11.144 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]

### 11.144.1 Constructor & Destructor Documentation

**11.144.1.1 StimulusDeviceDataAndUnrolledData()** [StimulusDeviceDataAndUnrolledData](#) ( )

**11.144.1.2 ~StimulusDeviceDataAndUnrolledData()** [~StimulusDeviceDataAndUnrolledData](#) ( )

Destructor: called by Dispose()

### 11.144.1.3 `!StimulusDeviceDataAndUnrolledData()` `!StimulusDeviceDataAndUnrolledData ( )`

Finalizer: called by GC before collecting

## 11.144.2 Property Documentation

**11.144.2.1 DeviceData** `array< uint8_t>^ DeviceData [get]`

**11.144.2.2 DeviceDataLength** `int DeviceDataLength [get]`

**11.144.2.3 UnrolledAmplitude** `array< int32_t>^ UnrolledAmplitude [get]`

**11.144.2.4 UnrolledDuration** `array< uint64_t>^ UnrolledDuration [get]`

**11.144.2.5 UnrolledSync** `array< uint32_t>^ UnrolledSync [get]`

## 11.145 usbSetupPacket\_t Class Reference

### Public Attributes

- `uint8_t bmRequestType`
- `uint8_t bRequest`
- `uint16_t wValue`
- `uint16_t wIndex`
- `uint16_t wLength`

## 11.145.1 Member Data Documentation

**11.145.1.1 bmRequestType** `uint8_t bmRequestType`

**11.145.1.2 bRequest** `uint8_t bRequest`

**11.145.1.3 wIndex** `uint16_t wIndex`

**11.145.1.4 wLength** `uint16_t wLength`

**11.145.1.5 wValue** `uint16_t wValue`

## 11.146 W2100\_StimulusParametersNet Struct Reference

### Public Attributes

- `int` [DACResolution](#)
- `int` [TimeResolutionInNanoSeconds](#)
- `int` [VoltageRangeInMicroVolt](#)
- `int` [VoltageResolutionInMicroVolt](#)
- `int` [CurrentRangeInNanoAmp](#)
- `int` [CurrentResolutionInNanoAmp](#)

### 11.146.1 Member Data Documentation

**11.146.1.1 CurrentRangeInNanoAmp** `int CurrentRangeInNanoAmp`

**11.146.1.2 CurrentResolutionInNanoAmp** `int CurrentResolutionInNanoAmp`

**11.146.1.3 DACResolution** `int DACResolution`

**11.146.1.4 TimeResolutionInNanoSeconds** `int TimeResolutionInNanoSeconds`

**11.146.1.5 VoltageRangeInMicroVolt** `int VoltageRangeInMicroVolt`

**11.146.1.6 VoltageResolutionInMicroVolt** `int VoltageResolutionInMicroVolt`

## Index

- !CDacCalibrationFunctionNet
  - CDacCalibrationFunctionNet, [51](#)
- !CDigOutStimulatorFunctionNet
  - CDigOutStimulatorFunctionNet, [61](#)
- !CExternDTesterDeviceNet
  - CExternDTesterDeviceNet, [65](#)
- !CGrapheneFunctionNet
  - CGrapheneFunctionNet, [102](#)
- !CInterfaceboard2FunctionNet
  - CInterfaceboard2FunctionNet, [117](#)
- !CInterfaceboardFunctionNet
  - CInterfaceboardFunctionNet, [119](#)
- !CLIH3DeviceNet
  - CLIH3DeviceNet, [122](#)
- !CMEA2100x256FunctionNet
  - CMEA2100x256FunctionNet, [263](#)
- !CMcsUsbFunctionNet
  - CMcsUsbFunctionNet, [231](#)
- !CMcsUsbListNet
  - CMcsUsbListNet, [238](#)
- !CMcsUsbNet
  - CMcsUsbNet, [244](#)
- !CMeFunctionNet
  - CMeFunctionNet, [296](#)
- !CMeaCleanDeviceNet
  - CMeaCleanDeviceNet, [268](#)
- !CMeaCoatDeviceNet
  - CMeaCoatDeviceNet, [272](#)
- !CMultiBatteryChargerDeviceNet
  - CMultiBatteryChargerDeviceNet, [298](#)
- !CMultiwellCallbackFunctionNet
  - CMultiwellCallbackFunctionNet, [305](#)
- !CMultiwellDeviceNet
  - CMultiwellDeviceNet, [307](#)
- !CMultiwellOptoStimFunctionNet
  - CMultiwellOptoStimFunctionNet, [313](#)
- !CPPCFunctionNet
  - CPPCFunctionNet, [347](#)
- !CPedoterDeviceNet
  - CPedoterDeviceNet, [328](#)
- !CPositionIIDeviceNet
  - CPositionIIDeviceNet, [334](#)
- !CPositionImpDeviceNet
  - CPositionImpDeviceNet, [342](#)
- !CProgramPressureCurveNet
  - CProgramPressureCurveNet, [361](#)
- !CPulseGeneratorFunctionNet
  - CPulseGeneratorFunctionNet, [362](#)
- !CRFFFunctionNet
  - CRFFFunctionNet, [370](#)
- !CSCUFunctionNet
  - CSCUFunctionNet, [423](#)
- !CTEERFunctionNet
  - CTEERFunctionNet, [512](#)
- !CUsbDeviceConfigurationFunctionNet, [522](#)
- !CWarnerUssingDeviceNet
  - CWarnerUssingDeviceNet, [541](#)
- !CWarnerUssingFunctionNet
  - CWarnerUssingFunctionNet, [544](#)
- !CWarnerValveControllerDeviceNet
  - CWarnerValveControllerDeviceNet, [563](#)
- !CWarnerValveControllerDeviceTesterFunctionNet
  - CWarnerValveControllerDeviceTesterFunctionNet, [584](#)
- !SidebandData
  - CStimulusFunctionNet::SidebandData, [622](#)
- !StimulusDeviceDataAndUnrolledData
  - CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, [623](#)
- \_AdditionalText
  - HeadstageIDTypeObject, [608](#)
- \_IdType
  - HeadstageIDTypeObject, [608](#)
- ~CCMOSMeaDeviceNet
  - CCMOSMeaDeviceNet, [46](#)
- ~CChannelTestDeviceNet
  - CChannelTestDeviceNet, [33](#)
- ~CCreateFilterNet
  - CCreateFilterNet, [48](#)
- ~CDacCalibrationFunctionNet
  - CDacCalibrationFunctionNet, [51](#)
- ~CDigOutStimulatorFunctionNet
  - CDigOutStimulatorFunctionNet, [61](#)
- ~CExternDTesterDeviceNet
  - CExternDTesterDeviceNet, [65](#)
- ~CFilterCoefficientsNet
  - CFilterCoefficientsNet, [67](#)
- ~CFilterPropertyNet
  - CFilterPropertyNet, [73](#)
- ~CFluidControlDeviceNet
  - CFluidControlDeviceNet, [75](#)
- ~CGenericDevelopDeviceNet
  - CGenericDevelopDeviceNet, [88](#)
- ~CGilsonDeviceNet
  - CGilsonDeviceNet, [99](#)
- ~CGrapheneFunctionNet
  - CGrapheneFunctionNet, [102](#)
- ~CInterfaceboard2FunctionNet
  - CInterfaceboard2FunctionNet, [117](#)
- ~CInterfaceboardFunctionNet
  - CInterfaceboardFunctionNet, [119](#)
- ~CLIH3DeviceNet
  - CLIH3DeviceNet, [122](#)
- ~CMEA2100x256FunctionNet
  - CMEA2100x256FunctionNet, [263](#)
- ~CMcsBusNet
  - CMcsBusNet, [168](#)
- ~CMcsBus\_AxisParametersNet
  - CMcsBus\_AxisParametersNet, [130](#)



- ~CMcsBus\_ExtensionNet
    - CMcsBus\_ExtensionNet, [131](#)
  - ~CMcsBus\_FYIExtensionNet
    - CMcsBus\_FYIExtensionNet, [132](#)
  - ~CMcsBus\_MotorControlNet
    - CMcsBus\_MotorControlNet, [136](#)
  - ~CMcsBus\_SensorNet
    - CMcsBus\_SensorNet, [152](#)
  - ~CMcsBus\_TempSensorNet
    - CMcsBus\_TempSensorNet, [161](#)
  - ~CMcsBus\_VoltageModeNet
    - CMcsBus\_VoltageModeNet, [164](#)
  - ~CMcsUsbDacqNet
    - CMcsUsbDacqNet, [177](#)
  - ~CMcsUsbFactoryNet
    - CMcsUsbFactoryNet, [223](#)
  - ~CMcsUsbFunctionNet
    - CMcsUsbFunctionNet, [231](#)
  - ~CMcsUsbListEntryNet
    - CMcsUsbListEntryNet, [232](#)
  - ~CMcsUsbListNet
    - CMcsUsbListNet, [238](#)
  - ~CMcsUsbNet
    - CMcsUsbNet, [244](#)
  - ~CMeFunctionNet
    - CMeFunctionNet, [296](#)
  - ~CMeaCleanDeviceNet
    - CMeaCleanDeviceNet, [267](#)
  - ~CMeaCoatDeviceNet
    - CMeaCoatDeviceNet, [272](#)
  - ~CMeaDeviceNet
    - CMeaDeviceNet, [278](#)
  - ~CMeaImpedanceDeviceNet
    - CMeaImpedanceDeviceNet, [290](#)
  - ~CMeaSwitchDeviceNet
    - CMeaSwitchDeviceNet, [293](#)
  - ~CMeaUSBDeviceNet
    - CMeaUSBDeviceNet, [295](#)
  - ~CMultiBatteryChargerDeviceNet
    - CMultiBatteryChargerDeviceNet, [298](#)
  - ~CMultiwellCallbackFunctionNet
    - CMultiwellCallbackFunctionNet, [305](#)
  - ~CMultiwellDeviceNet
    - CMultiwellDeviceNet, [307](#)
  - ~CMultiwellOptoStimFunctionNet
    - CMultiwellOptoStimFunctionNet, [313](#)
  - ~CNF\_GenDeviceNet
    - CNF\_GenDeviceNet, [317](#)
  - ~CokuvisionStimulatorDeviceNet
    - CokuvisionStimulatorDeviceNet, [323](#)
  - ~CPPCFunctionNet
    - CPPCFunctionNet, [347](#)
  - ~CPathIdentDeviceNet
    - CPathIdentDeviceNet, [327](#)
  - ~CPedoterDeviceNet
    - CPedoterDeviceNet, [328](#)
  - ~CPeristalticPumpDeviceNet
    - CPeristalticPumpDeviceNet, [329](#)
  - ~CPgaDeviceNet
    - CPgaDeviceNet, [330](#)
  - ~CPositionIIDeviceNet
    - CPositionIIDeviceNet, [334](#)
  - ~CPositionImpDeviceNet
    - CPositionImpDeviceNet, [342](#)
  - ~CProgramPressureCurveNet
    - CProgramPressureCurveNet, [360](#)
  - ~CPulseGeneratorFunctionNet
    - CPulseGeneratorFunctionNet, [362](#)
  - ~CRFFFunctionNet
    - CRFFFunctionNet, [370](#)
  - ~CRetinaLedDeviceNet
    - CRetinaLedDeviceNet, [368](#)
  - ~CRoboDeviceNet
    - CRoboDeviceNet, [394](#)
  - ~CRoboFluidDeviceNet
    - CRoboFluidDeviceNet, [406](#)
  - ~CSCUFunctionNet
    - CSCUFunctionNet, [423](#)
  - ~CSafeISDeviceNet
    - CSafeISDeviceNet, [418](#)
  - ~CStg200xBasicNet
    - CStg200xBasicNet, [441](#)
  - ~CStg200xDownloadNet
    - CStg200xDownloadNet, [477](#)
  - ~CSw2to64DeviceNet
    - CSw2to64DeviceNet, [495](#)
  - ~CTEERFunctionNet
    - CTEERFunctionNet, [512](#)
  - ~CTEERMachineDeviceNet
    - CTEERMachineDeviceNet, [521](#)
  - ~CTcxDeviceNet
    - CTcxDeviceNet, [499](#)
  - ~CUsbDeviceConfigurationFunctionNet
    - CUsbDeviceConfigurationFunctionNet, [522](#)
  - ~CWarnerUssingDeviceNet
    - CWarnerUssingDeviceNet, [541](#)
  - ~CWarnerUssingFunctionNet
    - CWarnerUssingFunctionNet, [544](#)
  - ~CWarnerValveControllerDeviceNet
    - CWarnerValveControllerDeviceNet, [563](#)
  - ~CWarnerValveControllerDeviceTesterFunctionNet
    - CWarnerValveControllerDeviceTesterFunctionNet, [584](#)
  - ~DriverVersionNet
    - DriverVersionNet, [597](#)
  - ~SidebandData
    - CStimulusFunctionNet::SidebandData, [622](#)
  - ~StimulusDeviceDataAndUnrolledData
    - CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, [623](#)
- A
- CFilterCoefficientsNet, [68](#)
- AdditionalText
- HeadstageIDTypeObject, [608](#)
- AddLoopEntry
- CRetinaLedDeviceNet, [368](#)

- AddSelectedChannelsQueue
  - CMcsUsbDacqNet, [177](#), [179](#), [180](#)
- AddSoftwareKey
  - CMcsUsbNet, [244](#)
- AddTableEntry
  - CRetinaLedDeviceNet, [368](#)
- Altera
  - FirmwareDestinationNames, [601](#)
- amplification
  - CMeaAudioFunctionNet::s\_setaudionet, [621](#)
  - CW2100\_FunctionNet::AudioChannelsNet, [29](#)
- AmplifierSettle
  - CIntanMea\_FunctionNet, [115](#)
- AnalogGain
  - CMeaDeviceNet, [283](#)
- ApplyGains
  - CPgaDeviceNet, [331](#)
- AreTransistorVoltagesSet
  - CCMOSMea\_FunctionNet, [36](#)
- AssociateToThis
  - CMcsUsbNet, [244](#)
- AutomaticAnalogOut
  - CSCUFunctionNet, [423](#)
- Axes\_I
  - CRoboDeviceNet, [401](#)
- Axes\_X
  - CRoboDeviceNet, [401](#)
- Axes\_Y
  - CRoboDeviceNet, [401](#)
- Axes\_Z
  - CRoboDeviceNet, [401](#)
- Axis\_I
  - CRoboDeviceNet, [401](#)
- Axis\_X
  - CRoboDeviceNet, [401](#)
- Axis\_Y
  - CRoboDeviceNet, [401](#)
- Axis\_Z
  - CRoboDeviceNet, [401](#)
- B
  - CFilterCoefficientsNet, [68](#)
- BatteryState, [29](#)
  - Charge, [30](#)
  - ChargeRegionString, [30](#)
  - ChargeString, [30](#)
  - Voltage, [30](#)
  - VoltageString, [30](#)
- BcdDevice
  - DeviceldNet, [593](#)
- BeginImpedanceCheck
  - CIntanMea\_FunctionNet, [115](#)
- BesselFilterHighPassNet, [30](#)
  - BesselFilterHighPassNet, [30](#)
- BesselFilterLowPassNet, [31](#)
  - BesselFilterLowPassNet, [31](#)
- bmRequestType
  - usbSetupPacket\_t, [624](#)
- BOOST\_BIT
  - CW2100\_StimulatorFunctionNet, [539](#)
- Bootstrap
  - FirmwareDestinationNames, [601](#)
- bRequest
  - usbSetupPacket\_t, [624](#)
- BurnAdcOffset
  - COctoPotDeviceNet, [319](#)
- BurnDacOffset
  - CDacCalibrationFunctionNet, [51](#)
  - COctoPotDeviceNet, [319](#)
- BUS1\_MCSBUS1
  - FirmwareDestinationNames, [601](#)
- BUS1\_MCSBUS2
  - FirmwareDestinationNames, [602](#)
- BusType
  - DeviceldNet, [593](#)
- ButterworthFilterHighPassNet, [31](#)
  - ButterworthFilterHighPassNet, [32](#)
- ButterworthFilterLowPassNet, [32](#)
  - ButterworthFilterLowPassNet, [32](#)
- CalibrateThermocouple
  - CFluidControlDeviceNet, [75](#)
  - CTcxDeviceNet, [499](#)
- CancelInternalCalibration
  - CTEERFunctionNet, [512](#)
- CancelPoolLoop
  - CRoboDeviceNet, [394](#)
- CancelPoolLoopAndStopMovement
  - CRoboDeviceNet, [394](#)
- CancelTableLoop
  - CRoboDacqNet, [380](#)
- CancelTableLoopAndStopTable
  - CRoboDacqNet, [380](#)
- CapacityTest
  - CMultiBatteryChargerDeviceNet, [298](#)
- CatchAmpGetAdcMean
  - CMcsBus\_SensorNet, [152](#)
- CatchAmpGetAdcValue
  - CMcsBus\_SensorNet, [153](#)
- CatchAmpGetAdcValueH
  - CMcsBus\_SensorNet, [153](#)
- CatchAmpGetAdcValueL
  - CMcsBus\_SensorNet, [153](#)
- CatchAmpGetDacAmplitude
  - CMcsBus\_SensorNet, [153](#)
- CatchAmpGetDacEnable
  - CMcsBus\_SensorNet, [153](#)
- CatchAmpGetDacOffset
  - CMcsBus\_SensorNet, [153](#)
- CatchAmpGetPwmEnable
  - CMcsBus\_SensorNet, [153](#)
- CatchAmpSetDacAmplitude
  - CMcsBus\_SensorNet, [153](#)
- CatchAmpSetDacEnable
  - CMcsBus\_SensorNet, [154](#)
- CatchAmpSetDacOffset
  - CMcsBus\_SensorNet, [154](#)
- CatchAmpSetPwmEnable

- CMcsBus\_SensorNet, 154
- CChannelTestDeviceNet, 33
  - ~CChannelTestDeviceNet, 33
  - CChannelTestDeviceNet, 33
  - SetAmplitude, 33
  - SetAttenuation, 33
  - SetFrequency, 33
  - SetWaveform, 34
- CCMOSMea\_FunctionNet, 34
  - AreTransistorVoltagesSet, 36
  - CCMOSMea\_FunctionNet, 36
  - ClearSTGOutput, 36
  - DetectChipType, 36
  - EnableChannelsInGroup, 36, 37
  - GetADCInputOffset, 37
  - GetBath, 37
  - GetBathMode, 37
  - GetEnabledChannelsInGroup, 37
  - GetGate, 37
  - GetGNDI, 37
  - GetGroupADCBits, 37
  - GetGroupChannelBitmaskBySelect, 38
  - GetGroupChannelBitmaskHS1NCBathCurrent, 38
  - GetGroupChannelBitmaskHS1NCCol2Current, 38
  - GetGroupChannelBitmaskHS1NChipTemp, 38
  - GetGroupChannelBitmaskHS1Sidebands, 39
  - GetGroupChannelBitmaskHS1TriggerStatus, 39
  - GetGroupChannelBitmaskIFDigChannels, 39
  - GetGroupChannelBitmaskInterfaceADC, 39
  - GetGroupChannelBitmaskPacketFrameContext, 40
  - GetGroupChannelBitmaskSTG1DACSignal, 40
  - GetGroupDCOffset, 40
  - GetGroupID, 40
  - GetGroupNumberOfChannels, 41
  - GetGroupResolutionPerDigit, 41
  - GetGroupSampleSize, 41
  - GetGroupType, 41
  - GetGroupUnit, 41, 42
  - GetMaxNumOfColumns, 42
  - GetNeurochipMemoryData, 42
  - GetNeurochipMemorySize, 42
  - GetNumberOfSupportedGroups, 42
  - GetSourceBulk, 42
  - GetSourceDrain, 42
  - GetSourceGate, 42
  - GetStimulusSites, 43
  - GetVDD3I, 43
  - GetVDDI, 43
  - IsChipPowered, 43
  - IsGateFloating, 43
  - PowerChip, 43
  - SetADCInputOffset, 43
  - SetBath, 43
  - SetBathMode, 43
  - SetGate, 43
  - SetGateFloating, 44
  - SetGateToVOP, 44
  - SetNeurochipMemoryData, 44
  - SetSourceBulk, 44
  - SetSourceDrain, 44
  - SetSourceGate, 44
  - SetStimulusSites, 44
  - UpdateTransistorVoltages, 44
  - VOPSTimerSetResetTimes, 44, 45
- CCMOSMeaDeviceNet, 45
  - ~CCMOSMeaDeviceNet, 46
  - CCMOSMeaDeviceNet, 46
  - CMosMea, 47
  - GetAvailableBaseSamplerates, 46
  - GetBaseSamplerate, 46
  - GetChannelDataI16, 46
  - GetChannelDataI32, 46
  - GetChannelDataUI16, 46
  - GetChannelDataUI32, 47
  - GetCMOSDataDictionary, 47
  - SetBaseSamplerate, 47
  - SetRegionOfInterests, 47
  - Stimulus, 47
  - UpdateChannelBlock, 47
- CCMOSMeaDeviceNet::CRegionOfInterestRect, 366
  - CRegionOfInterestRect, 366
  - DeepCopy, 366
  - m\_Bottom, 367
  - m\_Left, 367
  - m\_Right, 367
  - m\_Top, 367
- CCreateFilterNet, 48
  - ~CCreateFilterNet, 48
  - CCreateFilterNet, 48, 49
  - CutoffFrequency, 49
  - FindFilter, 49
  - GetBiQuad, 49
  - GetBiQuads, 49
  - NumCoefSets, 49
  - Order, 49
  - SampleRate, 50
  - Scale, 50
- CDacCalibrationFunctionNet, 50
  - !CDacCalibrationFunctionNet, 51
  - ~CDacCalibrationFunctionNet, 51
  - BurnDacOffset, 51
  - CDacCalibrationFunctionNet, 50, 51
  - GetDacOffset, 51
  - SetDacOffset, 51
- CDacqGroupChannelGenericSelectionNet, 52
  - CDacqGroupChannelGenericSelectionNet, 52
- CDacqGroupChannelSelectionNet, 52
  - CDacqGroupChannelSelectionNet, 53
- CDacqGroupChannelSelectionTemplateNet
  - CDacqGroupChannelSelectionTemplateNet< Dac-qGroupChannelEnumTemplateNet, Dac-qGroupChannelEnumTemplate, CDevice-GroupChannelInfoTemplateNet >, 54
- CDacqGroupChannelSelectionTemplateNet< Dac-qGroupChannelEnumTemplateNet, Dac-qGroupChannelEnumTemplate, CDevice-

- GroupChannelInfoTemplateNet >, 53
- CDacqGroupChannelSelectionTemplateNet, 54
- EnableChannelsInGroup, 54
- GetDeviceGroupChannelInfos, 54
- GetEnabledChannelsInGroup, 54
- GetGroupID, 54, 55
- GetGroupNumberOfChannels, 55
- GetGroupSampleSize, 55
- GetGroupType, 55
- GetNumberOfSupportedGroups, 55
- CDeviceGroupChannelInfoGenericNet, 56
  - CDeviceGroupChannelInfoGenericNet, 56
- CDeviceGroupChannelInfoMEA2100\_256Net, 56
  - CDeviceGroupChannelInfoMEA2100\_256Net, 57
- CDeviceGroupChannelInfoNet, 57
  - CDeviceGroupChannelInfoNet, 57
- CDeviceGroupChannelInfoSCUNet, 57
  - CDeviceGroupChannelInfoSCUNet, 58
- CDeviceGroupChannelInfoTemplateNet
  - CDeviceGroupChannelInfoTemplateNet< Dacq-GroupChannelEnumTemplateNet >, 58
- CDeviceGroupChannelInfoTemplateNet< Dacq-GroupChannelEnumTemplateNet >, 58
  - CDeviceGroupChannelInfoTemplateNet, 58
  - GroupID, 58
  - GroupType, 58
  - NumberOfChannels, 59
- CDeviceGroupChannelInfoW2100Net, 59
  - CDeviceGroupChannelInfoW2100Net, 59
- CDigOutStimulatorFunctionNet, 59
  - !CDigOutStimulatorFunctionNet, 61
  - ~CDigOutStimulatorFunctionNet, 61
  - CDigOutStimulatorFunctionNet, 60
  - ClearChannel, 61
  - GetGlobalRepeat, 61
  - GetNumberOfChannels, 61
  - GetStartTriggerSlope, 61
  - GetStopTriggerSlope, 62
  - PrepareChannelData, 62
  - SendChannelData, 62
  - SetGlobalRepeat, 63
  - SetStartTriggerSlope, 63
  - SetStopTriggerSlope, 63
- CEncapsulatorDeviceNet, 64
  - CEncapsulatorDeviceNet, 64
  - GetRoboFluidDevice, 64
- CExternDTesterDeviceNet, 64
  - !CExternDTesterDeviceNet, 65
  - ~CExternDTesterDeviceNet, 65
  - CExternDTesterDeviceNet, 65
  - Read, 65
  - Read2, 66
  - Write, 66
  - Write2, 66
- CFilterCoefficientsNet, 66
  - ~CFilterCoefficientsNet, 67
  - A, 68
  - B, 68
- CFilterCoefficientsNet, 67
  - GetUIntA, 68
  - GetUIntB, 68
  - IsEqual, 68
- CFilterCoefficientsNet::s\_FilterAttributesNet, 619
  - CommaPositionA, 620
  - CommaPositionB, 620
  - PostCommaA, 620
  - PostCommaB, 620
  - PreCommaA, 620
  - PreCommaB, 621
  - s\_FilterAttributesNet, 620
  - ToCpp, 620
- CFilterConfigurationNet, 68
  - CFilterConfigurationNet, 69
  - EraseFilterParameterPermanent, 69
  - GetFilterAttributes, 69
  - GetHighpassFilterEnable, 69
  - ResetHighpassFilter, 70
  - SetFilterParameter, 70
  - SetFilterParameterPermanent, 70
  - SetHighpassFilterEnable, 70
- CFilterConfigurationRegisterNet, 70
  - CFilterConfigurationRegisterNet, 71
  - EraseFilterParameterPermanent, 71
  - SetFilterParameter, 71, 72
  - SetFilterParameterPermanent, 72
- CFilterPropertyNet, 72
  - ~CFilterPropertyNet, 73
  - CFilterPropertyNet, 73
  - CornerFrequency, 73
  - CornerFrequencycymHz, 73
  - FilterActive, 73
  - FilterBand, 73
  - FilterFamily, 73
  - FilterType, 73
  - Order, 74
  - ToString, 73
- CFluidControlDeviceNet, 74
  - ~CFluidControlDeviceNet, 75
  - CalibrateThermocouple, 75
  - CFluidControlDeviceNet, 75
  - 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
- CGrapheneFunctionNet, 100
  - !CGrapheneFunctionNet, 102
  - ~CGrapheneFunctionNet, 102
  - CGrapheneFunctionNet, 102
  - GetDACOffset, 102, 103
  - GetVdVs, 103
  - GetVdVsDAC, 103, 104
  - GetVoltageRange, 104
  - GetVoltageReached, 105
  - GetVoltageResolution, 105
  - SetDACOffset, 106
  - SetVds, 106
  - SetVdVs, 107
  - SetVdVsDAC, 107
  - SetVgs, 108
  - SetVoltageRange, 108
  - SetVoltageResolution, 109
- ChangeSerialNumber
  - CMcsUsbFactoryNet, 223
- channel
  - CMeaAudioFunctionNet::s\_setaudionet, 621
  - CW2100\_FunctionNet::AudioChannelsNet, 29
- ChannelBlock\_AvailFrames
  - CMcsUsbDacqNet, 180, 181
- ChannelBlock\_ReadAsFrameArrayI16
  - CMcsUsbDacqNet, 181
- ChannelBlock\_ReadAsFrameArrayI32
  - CMcsUsbDacqNet, 182, 183
- ChannelBlock\_ReadAsFrameArrayUI16
  - CMcsUsbDacqNet, 183, 184
- ChannelBlock\_ReadAsFrameArrayUI32
  - CMcsUsbDacqNet, 185, 186
- ChannelBlock\_ReadFramesDictI16
  - CMcsUsbDacqNet, 186
- ChannelBlock\_ReadFramesDictI32
  - CMcsUsbDacqNet, 187
- ChannelBlock\_ReadFramesDictUI16
  - CMcsUsbDacqNet, 187
- ChannelBlock\_ReadFramesDictUI32
  - CMcsUsbDacqNet, 188
- ChannelBlock\_ReadFramesI16
  - CMcsUsbDacqNet, 189
- ChannelBlock\_ReadFramesI32
  - CMcsUsbDacqNet, 190, 191
- ChannelBlock\_ReadFramesUI16
  - CMcsUsbDacqNet, 191, 192
- ChannelBlock\_ReadFramesUI32
  - CMcsUsbDacqNet, 193
- ChannelDataEvent
  - CMcsUsbDacqNet, 219
- ChannelReset
  - CMultiBatteryChargerDeviceNet, 299
- Charge
  - BatteryState, 30
- ChargeRegionString
  - BatteryState, 30
- ChargeString
  - BatteryState, 30
- CHiClampDeviceNet, 109
  - CHiClampDeviceNet, 110
  - RoboDacq, 110
- CHLADacqNet, 110
  - CHLADacqNet, 111
- CHLADeviceNet, 111
  - CHLADeviceNet, 111
  - HLADacq, 112
  - SerialPort, 112
- CHWInfo
  - CMcsUsbDacqNet::CHWInfo, 113
- CIntanMea\_FunctionNet, 114
  - AmplifierSettle, 115
  - BeginImpedanceCheck, 115
  - CIntanMea\_FunctionNet, 115
  - GetDSPHighPassByIndex, 115
  - GetImpedanceResult, 115
  - GetIntanRegister, 115
  - GetLowerFrequencyByIndex, 115
  - GetUpperFrequencyByIndex, 116
  - SetBandwidthByIndex, 116

- SetDiagnosticMode, 116
- SetDSPHighPassByIndex, 116
- SetIntanRegister, 116
- CInterfaceboard2FunctionNet, 116
  - !CInterfaceboard2FunctionNet, 117
  - ~CInterfaceboard2FunctionNet, 117
  - CInterfaceboard2FunctionNet, 117
  - GetIoVoltage, 117
  - SetIoVoltage, 118
- CInterfaceboardFunctionNet, 118
  - !CInterfaceboardFunctionNet, 119
  - ~CInterfaceboardFunctionNet, 119
  - CInterfaceboardFunctionNet, 119
  - GetCardinalDacqSamplerate, 119
  - GetCardinalStgOutputrate, 119
  - SetCardinalDacqSamplerate, 119
  - SetCardinalStgOutputrate, 120
- ClampAmpRestart
  - CRoboDacqNet, 380
- ClearBuffers
  - CMcsUsbDacqNet, 194
- ClearChannel
  - CDigOutStimulatorFunctionNet, 61
- ClearChannel\_PrepAndSendData
  - CStg200xDownloadNet, 478
  - CStimulusFunctionNet, 485
- ClearChannelData
  - CStg200xDownloadBasicNet, 469
  - CStimulusFunctionNet, 485
  - CW2100\_StimulatorFunctionNet, 534
- ClearMultiplexedData
  - CStimulusFunctionNet, 485
- ClearSTGOutput
  - CCMOSMea\_FunctionNet, 36
- ClearStimulusParametersCache
  - CW2100\_FunctionNet, 527
- ClearSyncData
  - CStg200xDownloadBasicNet, 469
  - CStimulusFunctionNet, 485
- ClearTable
  - CRetinaLedDeviceNet, 368
- ClearTableName
  - CWarnerValveControllerDeviceNet, 563
- ClearUserDefinedNameCache
  - CW2100\_FunctionNet, 527
- ClearValveTable
  - CWarnerValveControllerDeviceNet, 563
- CLIH3DeviceNet, 120
  - !CLIH3DeviceNet, 122
  - ~CLIH3DeviceNet, 122
  - CLIH3DeviceNet, 122
  - DummyCommand, 122
  - EnableUserTrigger, 122
  - ErasePermanentAdcOffset, 123
  - ErasePermanentDacOffset, 123
  - GetAdcOffset, 123
  - GetAudioOutDacParameter, 123
  - GetDacIdleValue, 124
  - GetDacOffset, 124
  - GetDacqRunStatus, 124
  - GetDacUseldleValue, 124
  - GetDigInState, 125
  - GetEEPROMPage, 125
  - GetSampleInterval, 125
  - IsUserTriggerEnabled, 125
  - ReadClipping, 126
  - SendCommand, 126
  - SetAdcOffset, 126
  - SetAdcOffsetPermanent, 127
  - SetAudioOutDacParameter, 127
  - SetDacIdleValue, 127
  - SetDacOffset, 127
  - SetDacOffsetPermanent, 128
  - SetDacUseldleValue, 128
  - SetDigOutState, 128
  - SetEEPROMPage, 128
  - SetSampleInterval, 129
  - StimulusFunction, 129
- CloseAllValves
  - CRoboFluidDeviceNet, 406
- ClosePipe
  - CGenericDevelopDeviceNet, 88
- ClosePlateClamp
  - CMultiwellDeviceNet, 308
- CMcsBus\_AxisParametersNet, 129
  - ~CMcsBus\_AxisParametersNet, 130
  - CMcsBus\_AxisParametersNet, 129
  - GetAxisParametersSignedEeprom, 130
  - GetAxisParametersUnsignedEeprom, 130
  - SetAxisParametersEeprom, 130
- CMcsBus\_ExtensionNet, 130
  - ~CMcsBus\_ExtensionNet, 131
  - CMcsBus\_ExtensionNet, 131
  - GetLEDSwitch, 131
  - SetLEDSwitch, 131
- CMcsBus\_FYIExtensionNet, 131
  - ~CMcsBus\_FYIExtensionNet, 132
  - CMcsBus\_FYIExtensionNet, 132
  - GetDIO, 132
  - GetSingleHeater, 132
  - GetValves, 132
  - SetDIO, 132
  - SetSingleHeater, 133
  - SetValves, 133
- CMcsBus\_MotorControlNet, 133
  - ~CMcsBus\_MotorControlNet, 136
  - CMcsBus\_MotorControlNet, 136
  - GetMCAcceleration, 136
  - GetMCAccelerationEeprom, 136
  - GetMCAccelerationShortCommand, 137
  - GetMCAXisRevisionEeprom, 137
  - GetMCBreakCurrent, 137
  - GetMCBreakCurrentEeprom, 137
  - GetMCConfig, 137
  - GetMCConfigEeprom, 137
  - GetMCCurrent, 137



GetMCCurrentEeprom, 138  
GetMCCurrentMode, 138  
GetMCCurrentModeEeprom, 138  
GetMCCurrentModeShortCommand, 138  
GetMCCurrentPosition, 138  
GetMCCurrentShortCommand, 138  
GetMCCurrentSpeed, 138  
GetMCMaxAcceleration, 139  
GetMCMaxAccelerationEeprom, 139  
GetMCMaxCurrent, 139  
GetMCMaxCurrentEeprom, 139  
GetMCMaxSpeed, 139  
GetMCMaxSpeedEeprom, 139  
GetMCMaxTravel, 139  
GetMCMaxTravelEeprom, 140  
GetMCMaxTravelShortCommand, 140  
GetMCMovement, 140  
GetMCNewPosition, 140  
GetMCOutputOnOff, 140  
GetMCPhase, 140  
GetMCPhaseOffset, 140  
GetMCReference, 141  
GetMCReferenceCurrent, 141  
GetMCReferenceCurrentEeprom, 141  
GetMCRegulatorGain, 141  
GetMCRegulatorGainEeprom, 141  
GetMCScalingFactor, 141  
GetMCScalingFactorEeprom, 141  
GetMCSpeed, 142  
GetMCSpeedEeprom, 142  
GetMCSpeedShortCommand, 142  
GetMCSpeedUnitEeprom, 142  
GetMCStandbyCurrent, 142  
GetMCStandbyCurrentEeprom, 142  
GetMCStandbyTime, 142  
GetMCStandbyTimeEeprom, 143  
GetSubChannel, 143  
SetMCAcceleration, 143  
SetMCAccelerationEeprom, 143  
SetMCAccelerationShortCommand, 143  
SetMCAXisRevisionEeprom, 143  
SetMCBreakCurrent, 143  
SetMCBreakCurrentEeprom, 144  
SetMCCConfig, 144  
SetMCCConfigEeprom, 144  
SetMCCCurrent, 144  
SetMCCCurrentEeprom, 144  
SetMCCCurrentMode, 144  
SetMCCCurrentModeEeprom, 145  
SetMCCurrentModeShortCommand, 145  
SetMCCurrentPosition, 145  
SetMCCurrentShortCommand, 145  
SetMCMaxAcceleration, 145  
SetMCMaxAccelerationEeprom, 145  
SetMCMaxCurrent, 146  
SetMCMaxCurrentEeprom, 146  
SetMCMaxSpeed, 146  
SetMCMaxSpeedEeprom, 146

SetMCMaxTravel, 146  
SetMCMaxTravelEeprom, 146  
SetMCMaxTravelShortCommand, 147  
SetMCNewPosition, 147  
SetMCOutputOnOff, 147  
SetMCReference, 147  
SetMCReferenceCurrent, 147  
SetMCReferenceCurrentEeprom, 147  
SetMCRegulatorGain, 148  
SetMCRegulatorGainEeprom, 148  
SetMCRotation, 148  
SetMCScalingFactor, 148  
SetMCScalingFactorEeprom, 148  
SetMCSpeed, 148  
SetMCSpeedEeprom, 149  
SetMCSpeedShortCommand, 149  
SetMCSpeedUnitEeprom, 149  
SetMCStandbyCurrent, 149  
SetMCStandbyCurrentEeprom, 149  
SetMCStandbyTime, 149  
SetMCStandbyTimeEeprom, 150  
SetSubChannel, 150  
StartMCMovement, 150  
StopMCMovement, 150  
CMcsBus\_SensorNet, 150  
    ~CMcsBus\_SensorNet, 152  
CatchAmpGetAdcMean, 152  
CatchAmpGetAdcValue, 153  
CatchAmpGetAdcValueH, 153  
CatchAmpGetAdcValueL, 153  
CatchAmpGetDacAmplitude, 153  
CatchAmpGetDacEnable, 153  
CatchAmpGetDacOffset, 153  
CatchAmpGetPwmEnable, 153  
CatchAmpSetDacAmplitude, 153  
CatchAmpSetDacEnable, 154  
CatchAmpSetDacOffset, 154  
CatchAmpSetPwmEnable, 154  
CMcsBus\_SensorNet, 152  
Get2AnalogInput, 154  
Get2DigitalInput, 154  
Get4ADC, 154  
Get4ADCAverage, 154  
Get4ADCCatchampAverageShift, 154  
Get4ADCMode, 155  
Get4DAC, 155  
GetADCs, 155  
GetADCsLoop, 155  
GetBubbleStatus, 155  
GetDACs, 155  
GetDetectionThreshold, 155  
GetDetectorValue, 155  
GetLatency, 156  
GetLatencyCounter, 156  
GetMinimalThreshold, 156  
GetMovePump, 156  
GetPiezoState, 156  
GetPressure, 156

- GetPressureOffset, [157](#)
- GetRegulationTimeouts, [157](#)
- GetRegulatorFactor, [157](#)
- GetRegulatorOnOff, [157](#)
- GetRegulatorStatus, [157](#)
- GetRotatePump, [157](#)
- GetSamplePeriode, [158](#)
- GetSollPressure, [158](#)
- GetSyncState, [158](#)
- Set4ADCCatchampAverageShift, [158](#)
- Set4ADCMode, [158](#)
- SetDACs, [158](#)
- SetDetectionThreshold, [159](#)
- SetLatency, [159](#)
- SetMinimalThreshold, [159](#)
- SetMovePump, [159](#)
- SetPiezoState, [159](#)
- SetPressureOffset, [159](#)
- SetRegulationTimeouts, [159](#)
- SetRegulatorFactor, [160](#)
- SetRegulatorOnOff, [160](#)
- SetRotatePump, [160](#)
- SetSamplePeriode, [160](#)
- SetSollPressure, [160](#)
- StartSync, [160](#)
- TactSwitchGetState, [160](#)
- TactSwitchSetDisplay, [161](#)
- CMcsBus\_TempSensorNet, [161](#)
  - ~CMcsBus\_TempSensorNet, [161](#)
- CMcsBus\_TempSensorNet, [161](#)
- GetNanoVoltsPerKelvin, [162](#)
- GetTemperatur, [162](#)
- GetThermoOffset, [162](#)
- GetThermoTemp, [162](#)
- GetThermoVoltage, [162](#)
- SetNanoVoltsPerKelvin, [162](#)
- SetThermoOffset, [162](#)
- CMcsBus\_VoltageModeNet, [163](#)
  - ~CMcsBus\_VoltageModeNet, [164](#)
- CMcsBus\_VoltageModeNet, [164](#)
- GetVMMaxNegativeCurrent, [164](#)
- GetVMMaxNegativeCurrentEeprom, [164](#)
- GetVMMaxNegativeVoltage, [164](#)
- GetVMMaxNegativeVoltageEeprom, [164](#)
- GetVMMaxPositiveCurrent, [165](#)
- GetVMMaxPositiveCurrentEeprom, [165](#)
- GetVMMaxPositiveVoltage, [165](#)
- GetVMMaxPositiveVoltageEeprom, [165](#)
- GetVMOutputOnOff, [165](#)
- GetVMVoltage, [165](#)
- SetVMMaxNegativeCurrent, [165](#)
- SetVMMaxNegativeCurrentEeprom, [166](#)
- SetVMMaxNegativeVoltage, [166](#)
- SetVMMaxNegativeVoltageEeprom, [166](#)
- SetVMMaxPositiveCurrent, [166](#)
- SetVMMaxPositiveCurrentEeprom, [166](#)
- SetVMMaxPositiveVoltage, [166](#)
- SetVMMaxPositiveVoltageEeprom, [167](#)
- SetVMOutputOnOff, [167](#)
- SetVMVoltage, [167](#)
- CMcsBusNet, [167](#)
  - ~CMcsBusNet, [168](#)
- CMcsBusNet, [168](#)
- CMcsBusNet::GetMode, [168](#)
- CMcsBusNet::GetModeEeprom, [168](#)
- CMcsBusNet::SetMode, [169](#)
- CMcsBusNet::SetModeEeprom, [169](#)
- GetBusAddress, [169](#)
- GetBusAddressEeprom, [169](#)
- GetCommand, [169](#), [170](#)
- GetHWRevisionEeprom, [170](#)
- SetBusAddress, [170](#)
- SetBusAddressEeprom, [170](#)
- SetCommand, [170](#), [171](#)
- SetHWRevisionEeprom, [171](#)
- CMcsBusNet::GetMode
  - CMcsBusNet, [168](#)
- CMcsBusNet::GetModeEeprom
  - CMcsBusNet, [168](#)
- CMcsBusNet::SetMode
  - CMcsBusNet, [169](#)
- CMcsBusNet::SetModeEeprom
  - CMcsBusNet, [169](#)
- CMcsUsbDacqNet, [171](#)
  - ~CMcsUsbDacqNet, [177](#)
- AddSelectedChannelsQueue, [177](#), [179](#), [180](#)
- ChannelBlock\_AvailFrames, [180](#), [181](#)
- ChannelBlock\_ReadAsFrameArrayI16, [181](#)
- ChannelBlock\_ReadAsFrameArrayI32, [182](#), [183](#)
- ChannelBlock\_ReadAsFrameArrayUI16, [183](#), [184](#)
- ChannelBlock\_ReadAsFrameArrayUI32, [185](#), [186](#)
- ChannelBlock\_ReadFramesDictI16, [186](#)
- ChannelBlock\_ReadFramesDictI32, [187](#)
- ChannelBlock\_ReadFramesDictUI16, [187](#)
- ChannelBlock\_ReadFramesDictUI32, [188](#)
- ChannelBlock\_ReadFramesI16, [189](#)
- ChannelBlock\_ReadFramesI32, [190](#), [191](#)
- ChannelBlock\_ReadFramesUI16, [191](#), [192](#)
- ChannelBlock\_ReadFramesUI32, [193](#)
- ChannelDataEvent, [219](#)
- ClearBuffers, [194](#)
- CMcsUsbDacqNet, [177](#)
- CMcsUsbDacqNet::GetFilterProperties, [194](#)
- Error\_Callback\_Aquisition\_Stopped, [218](#)
- Error\_Callback\_Data\_lost, [218](#)
- Error\_Callback\_Frames\_Lost, [218](#)
- Error\_Callback\_Packet\_Error, [218](#)
- Error\_Callback\_Queue\_Full, [219](#)
- Error\_Callback\_RingQueue\_Full, [219](#)
- ErrorEvent, [219](#)
- GetAdapterType, [194](#)
- GetAdcDataFormat, [194](#)
- GetAdcZero, [194](#)
- GetAnalogValueUnit, [195](#)
- GetChannelDataFillSize, [195](#)



- GetChannelLayout, 195
- GetChannelsInBlock, 195
- GetDataFormat, 195
- GetDataMode, 195
- GetDigitalSource, 196, 197
- GetFilterProperty, 198
- GetGroupChannelData16, 198
- GetGroupChannelData32, 198
- GetGroupChannelDataUI16, 199
- GetGroupChannelDataUI32, 199
- GetHardwareMaxRange, 200
- GetHardwareMinRange, 200
- GetMaxSamplingFrequency, 200
- GetMeaLayout, 200
- GetMinSamplingFrequencyStepsize, 201
- GetNumberOfDataBits, 201
- GetPoti, 201
- GetResolutionPerDigit, 201
- GetSamplerate, 201
- GetVoltageRangeIndex, 201
- GetVoltageRangeInMicroVolt, 202
- GetVoltageRangeInMilliVolt, 202
- HWInfo, 202
- Samplerate, 219
- SendStartDacq, 202
- SendStartStgAndDacq, 202
- SendStopDacq, 203
- SendStopStgAndDacq, 203
- SendStopStgAndDacqWithOptions, 203
- SetDataMode, 204
- SetDigitalSource, 204–206
- SetPoti, 206
- SetSamplerate, 206
- SetSelectedChannels, 207–209
- SetSelectedChannelsQueue, 209–211
- SetSelectedData, 211–213
- SetupGroupDacqQueue, 213
- SetVoltageRangeByIndex, 213
- SetVoltageRangeInMicroVolt, 214
- StartDacq, 214, 215
- StartLoop, 216, 217
- StopDacq, 218
- StopLoop, 218
- CMcsUsbDacqNet::CHWInfo, 112
  - CHWInfo, 113
  - GetAvailableSampleRates, 113
  - GetAvailableVoltageRangesInMicroVolt, 113
  - GetAvailableVoltageRangesInMicroVoltAnd-StringsInMilliVolt, 113
  - GetNumberOfHWADCCChannels, 113
  - GetNumberOfHWDigitalChannels, 113
  - IsDigitalChannelDedicated, 114
- CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet, 524
  - CVoltageRangeInfoNet, 524
  - VoltageRangeDisplayStringMilliVolt, 524
  - VoltageRangeInMicroVolt, 524
- CMcsUsbDacqNet::GetFilterProperties
  - CMcsUsbDacqNet, 194
- CMcsUsbDeviceStatePushFunctionNet, 219
  - CMcsUsbDeviceStatePushFunctionNet, 220
  - McsUsbDeviceStateEvent, 220
  - TriggerStatus, 220
- CMcsUsbDeviceStatePushNet, 220
  - CMcsUsbDeviceStatePushNet, 221
  - McsUsbDeviceStateEvent, 221
  - TriggerStatus, 221
- CMcsUsbFactoryNet, 221
  - ~CMcsUsbFactoryNet, 223
  - ChangeSerialNumber, 223
  - CMcsUsbFactoryNet, 223
  - Coldstart, 223
  - CompareFirmware, 224
  - DownloadFirmware, 224
  - FindFirmwareVersionMagicInBuffer, 224
  - FX3MCSDDataAddress, 229
  - FX3MCSDDataDeviceIdOffset, 229
  - FX3MCSDDataIFB1ImageOffset, 229
  - FX3MCSDDataIFB2ImageOffset, 229
  - FX3MCSDDataVersionOffset, 229
  - GetChecksumFromFX3Image, 224
  - GetDestination, 224
  - GetDestinationDisplayLabel, 224
  - GetDestinationName, 224, 225
  - GetDestinationSerialNumber, 225
  - GetDestinationTargetAddress, 225
  - GetFirmwareVersionFromFile, 225
  - GetFirmwareVersionFromHexFile, 225
  - GetNumDestinations, 225
  - GetUSBDeviceIDFromFX3Image, 226
  - GetUserCodeFromBitFile, 226
  - GetUserCodeFromFlash, 226
  - GetXilinxFlashOffset, 226
  - GetXilinxFlashReadCommand, 226
  - LoadUserFirmware, 226, 227
  - ReadBlockFromFlash, 227
  - ReadBlockFromIFBGlobalEEPROM, 227
  - ReadBlockFromNVMEM, 227
  - SetDestinationSerialNumber, 227
  - UpdateFirmware, 227–229
- CMcsUsbFunctionNet, 230
  - !CMcsUsbFunctionNet, 231
  - ~CMcsUsbFunctionNet, 231
  - CMcsUsbFunctionNet, 230, 231
  - m\_pMcsUsb, 231
  - m\_pMcsUsbFunction, 231
  - ThrowCUsbExceptionNetOnError, 231
- CMcsUsbFunctionPointerContainer, 231
- CMcsUsbListEntryNet, 231
  - ~CMcsUsbListEntryNet, 232
- DevicId, 236
- DeviceName, 236
- Equals, 233
- GetEntry, 234
- GetEntryCount, 235
- HwVersion, 236

- Manufacturer, [236](#)
- Product, [236](#)
- SerialNumber, [236](#)
- SetStringFormat, [235](#)
- ToString, [236](#)
- CMcsUsbListNet, [237](#)
  - !CMcsUsbListNet, [238](#)
  - ~CMcsUsbListNet, [238](#)
  - CMcsUsbListNet, [237](#)
  - Count, [239](#)
  - DeviceArrival, [239](#)
  - DeviceRemoval, [239](#)
  - GetNumberOfDevices, [238](#)
  - GetUsbListEntries, [238](#)
  - GetUsbListEntry, [238](#)
  - IsDeviceTypeOf, [239](#)
  - SetStringFormat, [239](#)
- CMcsUsbNet, [240](#)
  - !CMcsUsbNet, [244](#)
  - ~CMcsUsbNet, [244](#)
  - AddSoftwareKey, [244](#)
  - AssociateToThis, [244](#)
  - CMcsUsbNet, [244](#)
  - Connect, [244](#), [245](#)
  - CyclePort, [246](#)
  - Disconnect, [246](#)
  - EmptyKey, [246](#)
  - EnableExceptions, [246](#)
  - EraseEepromRegisterPreconfig, [246](#)
  - GetConfiguration, [247](#)
  - GetDeviceCannotStallOutRequests, [247](#)
  - GetDeviceCapableSpeed, [247](#)
  - GetDeviceEnum, [247](#)
  - GetDeviceId, [247](#)
  - GetDeviceRootHubVendorEnum, [247](#)
  - GetDeviceRootHubVendorID, [247](#)
  - GetDeviceRootHubVendorName, [247](#)
  - GetDeviceSpeed, [248](#)
  - GetErrorText, [248](#)
  - GetFirmwareVersion, [248](#)
  - GetHardwareRevision, [248](#)
  - GetHeadstageActive, [249](#)
  - GetHeadstageID, [249](#)
  - GetHeadstagePresent, [249](#)
  - GetIdent, [249](#)
  - GetLastUSBError, [250](#)
  - GetMea21UsbPort, [250](#)
  - GetNumConfigurations, [250](#)
  - GetSerialNumber, [250](#)
  - GetSoftwareKey, [250](#)
  - GetSoftwareKeyString, [250](#)
  - GetStatus, [250](#)
  - GetStatusOfLastCommand, [251](#)
  - GetUsbListEntry, [251](#)
  - GetVersion, [251](#)
  - HasSoftwareKey, [251](#)
  - IsConnected, [251](#)
  - IsDeviceHighSpeed, [251](#)
  - IsDeviceHighSpeedCapable, [252](#)
  - IsExceptionsEnabled, [252](#)
  - MultibootGetCypressImageId, [252](#)
  - MultibootGetImageId, [252](#)
  - MultibootGetSelectedImage, [252](#)
  - MultibootSelectImage, [252](#)
  - ReadEepromRegisterPreconfig, [253](#)
  - ReadRegister, [253](#)
  - ReadRegister32, [253](#)
  - ReadRegisterTimeSlot, [253](#)
  - RemoveSoftwareKey, [254](#)
  - RescanHeadstage, [254](#)
  - SerialNumber, [261](#)
  - SetConfiguration, [254](#)
  - SetSoftwareKey, [254](#)
  - Status\_AlreadyConfigured, [256](#)
  - Status\_BadStartFrame, [257](#)
  - Status\_Btstuff, [257](#)
  - Status\_BufferOverrun, [257](#)
  - Status\_BufferUnderrun, [257](#)
  - Status\_Canceled, [257](#)
  - Status\_Canceling, [257](#)
  - Status\_ConnectedPipes, [257](#)
  - Status\_ControlNotOwned, [257](#)
  - Status\_Crc, [257](#)
  - Status\_DataOverrun, [257](#)
  - Status\_DataToggleMismatch, [257](#)
  - Status\_DataUnderrun, [258](#)
  - Status\_DeviceLocked, [258](#)
  - Status\_DeviceNotFound, [258](#)
  - Status\_DeviceRemoved, [258](#)
  - Status\_DevNotResponding, [258](#)
  - Status\_EndpointHalted, [258](#)
  - Status\_ErrorBusy, [258](#)
  - Status\_ErrorShortTransfer, [258](#)
  - Status\_Fifo, [258](#)
  - Status\_FrameControlOwned, [258](#)
  - Status\_InternalHcError, [258](#)
  - Status\_InvalidDeviceHandle, [259](#)
  - Status\_InvalidHandle, [259](#)
  - Status\_InvalidParameter, [259](#)
  - Status\_InvalidPipeHandle, [259](#)
  - Status\_InvalidUrbFunction, [259](#)
  - Status\_IoPending, [259](#)
  - Status\_IoTimeout, [259](#)
  - Status\_IsochRequestFailed, [259](#)
  - Status\_LastUsbErrorMismatch, [259](#)
  - Status\_NoBandwidth, [259](#)
  - Status\_NoMemory, [260](#)
  - Status\_NoSuchDevice, [260](#)
  - Status\_NotAccessed, [260](#)
  - Status\_NotSupported, [260](#)
  - Status\_PidCheckFailure, [260](#)
  - Status\_PipeNotLinked, [260](#)
  - Status\_RequestFailed, [260](#)
  - Status\_RequestMutexFailed, [260](#)
  - Status\_RequestMutexTimeout, [260](#)
  - Status\_Stall, [260](#)

- Status\_Unconfigured, [260](#)
- Status\_UnexpectedPid, [261](#)
- ThrowCUsbExceptionNetOnError, [254](#)
- TxnGetSerialNumber, [254](#)
- TxnSetSerialNumber, [254](#)
- TxnTestMemoryReadAndCheck, [254](#)
- TxnTestMemoryWrite, [255](#)
- ValidKey, [255](#)
- WPAError\_ScanningIsPending, [261](#)
- WriteEepromRegisterPreconfig, [255](#)
- WriteRegister, [255](#), [256](#)
- WriteRegister32, [256](#)
- WriteRegisterArray, [256](#)
- WriteRegisterTimeSlot, [256](#)
- WriteRegisterValue, [256](#)
- CMcsUsbPointerContainer, [261](#)
- CMEA2100\_256DacqGroupChannelSelectionNet, [261](#)
  - CMEA2100\_256DacqGroupChannelSelectionNet, [261](#)
- CMEA2100x256FunctionNet, [262](#)
  - !CMEA2100x256FunctionNet, [263](#)
  - ~CMEA2100x256FunctionNet, [263](#)
  - CMEA2100x256FunctionNet, [262](#)
  - GetLayoutConfiguration, [263](#)
  - SetLayoutConfiguration, [263](#)
- CMeaAudioFunctionNet, [263](#)
  - CMeaAudioFunctionNet, [264](#)
  - GetAudioChannels, [264](#), [265](#)
  - GetNumberOfAudioChannels, [265](#)
  - SetAudioChannels, [265](#), [266](#)
- CMeaAudioFunctionNet::s\_setaudionet, [621](#)
  - amplification, [621](#)
  - channel, [621](#)
- CMeaCleanDeviceNet, [266](#)
  - !CMeaCleanDeviceNet, [268](#)
  - ~CMeaCleanDeviceNet, [267](#)
  - CMeaCleanDeviceNet, [267](#)
  - GetCycle, [268](#)
  - GetCycles, [268](#)
  - GetMaxVoltage, [268](#)
  - GetMinVoltage, [268](#)
  - GetOutputVoltage, [268](#)
  - GetSlope, [269](#)
  - IsRunning, [269](#)
  - SetCycles, [269](#)
  - SetMaxVoltage, [269](#)
  - SetMinVoltage, [270](#)
  - SetSlope, [270](#)
  - Start, [270](#)
  - Stop, [270](#)
- CMeaCoatDeviceNet, [270](#)
  - !CMeaCoatDeviceNet, [272](#)
  - ~CMeaCoatDeviceNet, [272](#)
  - CMeaCoatDeviceNet, [272](#)
  - GetCurrentCycle, [272](#)
  - GetCycles, [272](#)
  - GetDuration, [272](#)
  - GetMaxCurrent, [272](#)
  - GetOffsetCurrent, [273](#)
  - GetOutputCurrent, [273](#)
  - GetPauseDuration, [273](#)
  - GetSlope, [273](#)
  - GetTimeInPause, [273](#)
  - GetTimeInPlateau, [274](#)
  - IsRunning, [274](#)
  - SetCycles, [274](#)
  - SetDuration, [274](#)
  - SetMaxCurrent, [275](#)
  - SetOffsetCurrent, [275](#)
  - SetPauseDuration, [275](#)
  - SetSlope, [275](#)
  - Start, [275](#)
  - Stop, [276](#)
- CMeaDeviceNet, [276](#)
  - ~CMeaDeviceNet, [278](#)
  - AnalogGain, [283](#)
  - CMeaDeviceNet, [277](#)
  - EnableChecksum, [278](#)
  - EnableDigitalIn, [278](#), [279](#)
  - EnableTimestamp, [279](#)
  - Gain, [283](#)
  - GetAnalogGain, [280](#)
  - GetEnumerationSpeed, [280](#)
  - GetGain, [280](#)
  - MeaAudioFunctionNet, [283](#)
  - MeaDigitalDataFunctionNet, [283](#)
  - MeaFeedbackFunctionNet, [283](#)
  - MeFunctionNet, [284](#)
  - SetDigitalOut, [280](#)
  - SetNumberOfAnalogChannels, [280](#)
  - SetNumberOfChannels, [281](#), [282](#)
  - SetTriggerMaskValue, [282](#)
  - SetTriggerPeriod, [283](#)
  - W2100\_FunctionNet, [284](#)
  - WClassicFunctionNet, [284](#)
- CMeaDigitalDataFunctionNet, [284](#)
  - CMeaDigitalDataFunctionNet, [284](#)
  - GetDigitalData, [285](#)
  - SetDigitalData, [285](#)
- CMeaFeedbackFunctionNet, [286](#)
  - CMeaFeedbackFunctionNet, [287](#)
  - FeedbackGetSampleTimerCount, [287](#)
  - FeedbackSetAnalogSource, [287](#)
  - FeedbackSetChannelFilter, [287](#)
  - FeedbackSetDigitalMapping, [287](#)
  - FeedbackSetFeedback, [287](#)
  - FeedbackSetFilterOff, [288](#)
  - FeedbackSetFilterParameter, [288](#)
  - FeedbackSetFilterParameter32, [288](#)
  - FeedbackSetGlobalChannelFilter, [288](#)
  - FeedbackSetIIRFilterParameter, [288](#)
  - FeedbackSetLogic, [288](#)
  - FeedbackSetMkFilter, [288](#)
  - FeedbackSetNumberOfLogics, [289](#)
  - FeedbackSetNumberOfRateCounter, [289](#)
  - FeedbackSetNumberOfRateDetectors, [289](#)

- FeedbackSetNumberOfSpikeDetectors, 289
- FeedbackSetNumberOfTriggers, 289
- FeedbackSetRateCounter, 289
- FeedbackSetRateDetector, 289
- FeedbackSetSpikeDetectorThreshold, 289
- FeedbackSetTrigger, 290
- CMealImpedanceDeviceNet, 290
  - ~CMealImpedanceDeviceNet, 290
  - CMealImpedanceDeviceNet, 290
  - GetAdapterCode, 291
  - GetArraySize, 291
  - GetImpedanceTestFrequency, 291
  - GetReady, 291
  - GetResult, 291
  - SetImpedanceTestFrequency, 291
  - StartMeasurement, 291
- CMeasureTableDeviceNet, 291
  - CMeasureTableDeviceNet, 292
  - Sensor, 292
- CMeaSwitchDeviceNet, 292
  - ~CMeaSwitchDeviceNet, 293
  - CMeaSwitchDeviceNet, 293
  - GetNumber, 293
  - GetPattern, 293
  - GetPatternBool, 294
  - SetPattern, 294
  - SetPatternBool, 294
- CMeaUSBDeviceNet, 294
  - ~CMeaUSBDeviceNet, 295
  - CMeaUSBDeviceNet, 295
- CMeFunctionNet, 295
  - !CMeFunctionNet, 296
  - ~CMeFunctionNet, 296
  - CMeFunctionNet, 296
  - SetTransformer, 296
- CMosMea
  - CCMOSMeaDeviceNet, 47
- CMultiBatteryChargerDeviceNet, 297
  - !CMultiBatteryChargerDeviceNet, 298
  - ~CMultiBatteryChargerDeviceNet, 298
  - CapacityTest, 298
  - ChannelReset, 299
  - CMultiBatteryChargerDeviceNet, 298
  - GetBatteryVoltage, 299
  - GetChannels, 299
  - GetChannelState, 299
  - GetChargeCapacity, 300
  - GetChargeCurrent, 300
  - GetChargingMode, 300
  - GetChargingPCoefficient, 300
  - GetDischargeCapacity, 301
  - GetDischargeCurrent, 301
  - GetDischargeCurrentSetPoint, 301
  - GetFinalDischargeVoltage, 302
  - GetRatedCapacity, 302
  - SetChargingMode, 302
  - SetChargingPCoefficient, 302
  - SetDischargeCurrentSetPoint, 303
  - SetFinalDischargeVoltage, 303
  - SetRatedCapacity, 303
  - SetRatedCapacityVolatile, 303
- CMultiwellCallbackFunctionNet, 304
  - !CMultiwellCallbackFunctionNet, 305
  - ~CMultiwellCallbackFunctionNet, 305
  - CMultiwellCallbackFunctionNet, 305
  - GetPlateClampStateByHeadstage, 305
  - GetPlateClampStateByHeadstageEvent, 306
  - OnGetPlateClampStateByHeadstage, 305
- CMultiwellDeviceNet, 306
  - !CMultiwellDeviceNet, 307
  - ~CMultiwellDeviceNet, 307
  - ClosePlateClamp, 308
  - CMultiwellDeviceNet, 307
  - GetPlateClampLockState, 308
  - GetPlateClampState, 308
  - GetPlateMux, 308, 309
  - GetPlateType, 309
  - GetPowerMuxPlate, 309
  - IsPlateTypeValid, 310
  - LockPlateClamp, 310
  - OpenPlateClamp, 310
  - SetPlateMux, 310, 311
  - SetPlateType, 311
  - SetPowerMuxPlate, 311
  - StopPlateClamp, 312
  - UnlockPlateClamp, 312
- CMultiwellOptoStimFunctionNet, 312
  - !CMultiwellOptoStimFunctionNet, 313
  - ~CMultiwellOptoStimFunctionNet, 313
  - CMultiwellOptoStimFunctionNet, 313
  - GetAbsMaxCurrentInMicroAmp, 313
  - GetColorRgb, 313
  - GetColorStr, 314
  - GetMaxDurationHighCurrentInMicroSec, 314
  - GetMaxDutyCycleHighCurrent, 314
  - GetPermanentCurrentInMicroAmp, 315
  - GetWaveLengthInNanometer, 315
  - SetAbsMaxCurrentInMicroAmp, 315
  - SetColorRgb, 315
  - SetColorStr, 316
  - SetMaxDurationHighCurrentInMicroSec, 316
  - SetMaxDutyCycleHighCurrent, 316
  - SetPermanentCurrentInMicroAmp, 316
  - SetWaveLengthInNanometer, 317
- CNF\_GenDeviceNet, 317
  - ~CNF\_GenDeviceNet, 317
  - CNF\_GenDeviceNet, 317
  - Set\_Values, 317
- COctoPotDeviceNet, 318
  - BurnAdcOffset, 319
  - BurnDacOffset, 319
  - COctoPotDeviceNet, 318, 319
  - EnableChecksum, 319
  - EnableDigitalIn, 319
  - EnableTimestamp, 319
  - GetAdcOffset, 319

- GetDacOffset, [319](#)
- PatternListStart, [319](#)
- RampStart, [320](#)
- ResetAdcOffset, [320](#)
- ResetDacOffset, [320](#)
- SetAdcOffset, [320](#)
- SetAmplificationSwitch, [320](#)
- SetBathclamp, [320](#)
- SetChannelSwitch, [320](#)
- SetDacAutoControl, [320](#)
- SetDacOffset, [320](#)
- SetDacValue, [321](#)
- SetNumberOfChannels, [321](#)
- SetOutputRate, [321](#)
- SetPatternListEntry, [321](#)
- SetPidParameter, [321](#)
- SetRampParameter, [321](#)
- SetSineParameter, [321](#)
- SineStart, [322](#)
- CokuvisionStimulatorDeviceNet, [322](#)
  - ~CokuvisionStimulatorDeviceNet, [323](#)
- CokuvisionStimulatorDeviceNet, [323](#)
- GetCheckVoltage, [323](#)
- GetCurrentFactor, [323](#)
- GetDACOffset, [323](#)
- GetMaxPower, [323](#)
- GetMaxVoltage, [323](#)
- GetPulseform, [323](#)
- GetRTC, [324](#)
- GetStimulatorStatus, [324](#)
- GetVoltage, [324](#)
- SetCheckVoltage, [324](#)
- SetCurrentFactor, [324](#)
- SetDACOffset, [324](#)
- SetMaxPower, [325](#)
- SetMaxVoltage, [325](#)
- SetPulseform, [325](#)
- SetRTC, [325](#)
- Coldstart
  - CMcsUsbFactoryNet, [223](#)
- CommaPositionA
  - CFilterCoefficientsNet::s\_FilterAttributesNet, [620](#)
- CommaPositionB
  - CFilterCoefficientsNet::s\_FilterAttributesNet, [620](#)
- CompareFirmware
  - CMcsUsbFactoryNet, [224](#)
- CompareTo
  - HeadStageIDType, [605](#)
- CompensateElectrodeOffset
  - CWarnerUssingFunctionNet, [544](#)
- Connect
  - CMcsUsbNet, [244](#), [245](#)
  - CRFFunctionNet, [370](#)
- ConnectDevice
  - CRadioControlledDevicesNet, [365](#)
- ConnectedImp
  - CPositionImpDeviceNet, [342](#)
- ConnectImp
  - CPositionImpDeviceNet, [342](#)
- ConnectSlave
  - CGilsonDeviceNet, [99](#)
- ControlState
  - HeadStageIDTypeState, [609](#)
- CornerFrequency
  - CFilterPropertyNet, [73](#)
- CornerFrequencymHz
  - CFilterPropertyNet, [73](#)
- Count
  - CMcsUsbListNet, [239](#)
- CPatchServerDeviceNet, [325](#)
  - CPatchServerDeviceNet, [326](#)
  - Sensor, [326](#)
- CPathIdentDeviceNet, [326](#)
  - ~CPathIdentDeviceNet, [327](#)
  - CPathIdentDeviceNet, [327](#)
  - Measure, [327](#)
  - Set\_Values, [327](#)
- CPedoterDeviceNet, [327](#)
  - !CPedoterDeviceNet, [328](#)
  - ~CPedoterDeviceNet, [328](#)
  - CPedoterDeviceNet, [328](#)
  - GetCommand, [328](#)
  - SetCommand, [328](#)
- CPeristalticPumpDeviceNet, [329](#)
  - ~CPeristalticPumpDeviceNet, [329](#)
  - CPeristalticPumpDeviceNet, [329](#)
  - McsBus\_MotorControl, [330](#)
- CPgaDeviceNet, [330](#)
  - ~CPgaDeviceNet, [330](#)
  - ApplyGains, [331](#)
  - CPgaDeviceNet, [330](#)
  - DefineAmplification, [331](#)
  - DefineFrequencyRange, [331](#)
  - DefineNumAmplifications, [331](#)
  - DefineNumFrequencyRanges, [331](#)
  - GetAmplification, [331](#)
  - GetFrequencyRange, [331](#)
  - GetGain, [332](#)
  - GetNumAmplifications, [332](#)
  - GetNumFrequencyRanges, [332](#)
  - SetGain, [332](#)
- CPositionIIDeviceNet, [332](#)
  - !CPositionIIDeviceNet, [334](#)
  - ~CPositionIIDeviceNet, [334](#)
  - CPositionIIDeviceNet, [334](#)
  - GetCoilCommunication, [334](#)
  - GetDebugData, [334](#)
  - GetEventData, [335](#)
  - GetImplantCurrentSetpoint, [335](#)
  - GetImplantResult, [336](#)
  - GetImplantState, [336](#)
  - GetOnOff, [336](#)
  - GetPowerStrength, [337](#)
  - GetRTC, [337](#)
  - GetStateDebugData, [337](#)
  - GetStateEventData, [338](#)

- RFFunction, [341](#)
- SetImplantCurrentSetpoint, [338](#)
- SetPowerStrength, [338](#)
- SetRTC, [338](#)
- SetStateDebugData, [340](#)
- SetStateEventData, [340](#)
- SwitchOnOff, [340](#)
- CPositionImpDeviceNet, [341](#)
  - !CPositionImpDeviceNet, [342](#)
  - ~CPositionImpDeviceNet, [342](#)
  - ConnectedImp, [342](#)
  - ConnectImp, [342](#)
  - CPositionImpDeviceNet, [342](#)
  - GetDeviceList, [342](#)
  - GetImpld, [343](#)
  - GetRFFrequency, [343](#)
  - SetDeviceList, [343](#)
  - SetImpld, [343](#)
  - SetRFFrequency, [344](#)
- CPPCDeviceNet, [344](#)
  - CPPCDeviceNet, [344](#)
  - McsBus, [345](#)
  - McsBus\_MotorControl, [345](#)
  - McsBus\_Sensor, [345](#)
  - PPCFunction, [345](#)
- CPPCFunctionNet, [345](#)
  - !CPPCFunctionNet, [347](#)
  - ~CPPCFunctionNet, [347](#)
  - CPPCFunctionNet, [346](#)
  - FirePressurePulse, [347](#)
  - GetAnalogVoltage, [347](#)
  - GetAnalogVoltageRange, [347](#)
  - GetDigitalIn, [349](#)
  - GetPressureRange, [349](#)
  - GetPumpModeType, [349](#)
  - GetPumpSpeedUnit, [350](#)
  - GetSupplyVoltage, [350](#)
  - GetValveActive, [350](#)
  - IsBusy, [350](#)
  - LoadPressure, [352](#)
  - MeasureReservoir, [352](#)
  - SetAnalogVoltageRange, [352](#)
  - SetPressureOffset, [352](#)
  - SetPressureRange, [352](#)
  - SetPumpModeType, [353](#)
  - SetPumpSpeedUnit, [353](#)
  - SetValveActive, [353](#)
- CPPS\_DeviceNet, [354](#)
  - CPPS\_DeviceNet, [354](#)
  - McsBus, [354](#)
  - McsBus\_MotorControl, [354](#)
  - McsBus\_Sensor, [354](#)
  - PPS\_Function, [354](#)
- CPPS\_FunctionNet, [355](#)
  - CPPS\_FunctionNet, [355](#), [356](#)
  - GetAnalogVoltage, [356](#)
  - GetAnalogVoltages, [356](#)
  - GetBubbleState, [356](#)
  - GetDigitalIn, [356](#)
  - GetPumpCouple, [356](#)
  - GetPumpEnableSpeedRatio, [356](#)
  - GetPumpFastOnOff, [356](#)
  - GetPumpFastSpeed, [356](#)
  - GetPumpFunctionSpeeds, [357](#)
  - GetPumpManualOnOff, [357](#)
  - GetPumpMaxSpeed, [357](#)
  - GetPumpModeType, [357](#)
  - GetPumpSpeedRatio, [357](#)
  - GetPumpSpeedUnit, [357](#)
  - GetSupplyVoltage, [357](#)
  - GetUseBubble, [357](#)
  - SetAnalogVoltages, [357](#)
  - SetPumpCouple, [358](#)
  - SetPumpEnableSpeedRatio, [358](#)
  - SetPumpFastOnOff, [358](#)
  - SetPumpFastSpeed, [358](#)
  - SetPumpFunctionSpeeds, [358](#)
  - SetPumpManualOnOff, [358](#)
  - SetPumpMaxSpeed, [358](#)
  - SetPumpModeType, [358](#)
  - SetPumpSpeedRatio, [359](#)
  - SetPumpSpeedUnit, [359](#)
  - SetUseBubble, [359](#)
- CPPSDeviceNet, [359](#)
  - CPPSDeviceNet, [360](#)
- CProgramPressureCurveNet, [360](#)
  - !CProgramPressureCurveNet, [361](#)
  - ~CProgramPressureCurveNet, [360](#)
  - CProgramPressureCurveNet, [360](#)
  - GetRepeats, [361](#)
  - Program, [361](#)
  - SetRepeats, [361](#)
- CPulseGeneratorFunctionNet, [361](#)
  - !CPulseGeneratorFunctionNet, [362](#)
  - ~CPulseGeneratorFunctionNet, [362](#)
  - CPulseGeneratorFunctionNet, [362](#)
  - GetModeSelect, [363](#)
  - GetPeriod, [363](#)
  - GetPulseLength, [363](#)
  - SetModeSelect, [364](#)
  - SetPeriod, [364](#)
  - SetPulseLength, [364](#)
- CRadioControlledDevicesNet, [364](#)
  - ConnectDevice, [365](#)
  - CRadioControlledDevicesNet, [365](#)
  - DisconnectDevice, [365](#)
  - GetDeviceNames, [365](#)
  - GetFrequency, [366](#)
  - HasRadioControl, [366](#)
  - SetFrequency, [366](#)
  - StillConnected, [366](#)
- CreateSideband
  - CStimulusFunctionNet, [485](#)
- CreateWirelessHeadstageSerialNumberString
  - CWirelessBaseFunctionNet, [591](#)
- CRegionOfInterestRect



- CCMOSMeaDeviceNet::CRegionOfInterestRect, 366
- CRetinaLedDeviceNet, 367
  - ~CRetinaLedDeviceNet, 368
  - AddLoopEntry, 368
  - AddTableEntry, 368
  - ClearTable, 368
  - CRetinaLedDeviceNet, 368
  - GetTablepointer, 368
  - SetLED, 368
  - SetLumi, 368
  - SetPersistency, 368
  - SetRepeat, 369
  - SetTablepointer, 369
  - SetTrigger, 369
- CRFFunctionNet, 369
  - !CRFFunctionNet, 370
  - ~CRFFunctionNet, 370
  - Connect, 370
  - CRFFunctionNet, 370
  - GetAvailableDeviceList, 371
  - GetAvailableDeviceListEx, 371
  - GetAvailableStateList, 371
  - GetAvailableStateListEx, 371
  - GetBaseFrequency, 372
  - GetConnectedDevice, 372
  - GetState, 372
  - GetTestMode, 372
  - GetWorkingFrequency, 373
  - SetBaseFrequency, 373
  - SetTestMode, 373
  - SetWorkingFrequency, 373
- CRobo\_FYIPProgram\_FunctionNet, 374
  - CRobo\_FYIPProgram\_FunctionNet, 374
  - GetLength, 374
  - GetState, 375
  - GetValve1, 375
  - GetValve2, 375
  - SetLength, 375
  - SetValve1, 375
  - SetValve2, 375
  - Start, 375
- CRobo\_FYITemp\_FunctionNet, 375
  - CRobo\_FYITemp\_FunctionNet, 376
  - GetICoeff, 376
  - GetMaxPower, 376
  - GetPCoeff, 376
  - GetRegulatorOnOff, 376
  - GetSollTemp, 376
  - SetICoeff, 377
  - SetMaxPower, 377
  - SetPCoeff, 377
  - SetRegulatorOnOff, 377
  - SetSollTemp, 377
- CRoboDacqNet, 377
  - CancelTableLoop, 380
  - CancelTableLoopAndStopTable, 380
  - ClampAmpRestart, 380
  - CRoboDacqNet, 380
  - DoRamp, 380
  - Emu\_GetCellCapacity, 380
  - Emu\_GetCellPotential, 380
  - Emu\_GetCellResists, 380
  - Emu\_GetElectrodeResists, 381
  - Emu\_GetNoise, 381
  - Emu\_SetCellCapacity, 381
  - Emu\_SetCellPotential, 381
  - Emu\_SetCellResists, 381
  - Emu\_SetElectrodeResists, 381
  - Emu\_SetNoise, 381
  - GetAllDigout, 381
  - GetCapacityC, 381
  - GetCapacityV, 381
  - GetCapacityX, 382
  - GetClampAmpSerialNumber, 382
  - GetCommand, 382
  - GetConfigurationBit, 382
  - GetConfigurationBitAxc, 382
  - GetConfigurationBitBlu\_Led, 382
  - GetConfigurationBitBlu\_LedToggleFast, 382
  - GetConfigurationBitBlu\_LedToggleSlow, 382
  - GetConfigurationBitCC\_Gen, 382
  - GetConfigurationBitCV\_Gen, 382
  - GetConfigurationBitRC\_Gen, 382
  - GetConfigurationBitRed\_Led, 383
  - GetConfigurationBitRed\_LedSaturation, 383
  - GetConfigurationBitRed\_LedToggleFast, 383
  - GetConfigurationBitRed\_LedToggleSlow, 383
  - GetConfigurationBitRelais, 383
  - GetConfigurationBitRV\_Gen, 383
  - GetConfigurationBits, 383
  - GetConfigurationBitStream, 383
  - GetConfigurationBitSupply, 383
  - GetCrossTalkOffset, 383
  - GetCrossTalkOptimum, 383
  - GetDigout, 384
  - GetDisplayText, 384
  - GetDownsampleFactor, 384
  - GetFilter, 384
  - GetFilterCoeffs, 384
  - GetIC, 384
  - GetIClamp, 384
  - GetICOffset, 384
  - GetIGain, 384
  - GetNIC\_MS, 384
  - GetNUC\_MS, 384
  - GetNUV\_MS, 385
  - GetPGain, 385
  - GetRecordingNumber, 385
  - GetResistanceC, 385
  - GetResistanceV, 385
  - GetScreen, 385
  - GetSimulation, 385
  - GetUC, 385
  - GetUClamp, 385
  - GetUCOffset, 385

- GetUpdateDisplay, [385](#)
- GetUV, [386](#)
- GetUVOffset, [386](#)
- GetXGain, [386](#)
- RunTable, [386](#)
- SetAllDigout, [386](#)
- SetCommand, [386](#)
- SetConfigurationBit, [386](#)
- SetConfigurationBitAxc, [386](#)
- SetConfigurationBitBlu\_Led, [386](#)
- SetConfigurationBitBlu\_LedToggleFast, [387](#)
- SetConfigurationBitBlu\_LedToggleSlow, [387](#)
- SetConfigurationBitCC\_Gen, [387](#)
- SetConfigurationBitCV\_Gen, [387](#)
- SetConfigurationBitRC\_Gen, [387](#)
- SetConfigurationBitRed\_Led, [387](#)
- SetConfigurationBitRed\_LedSaturation, [387](#)
- SetConfigurationBitRed\_LedToggleFast, [387](#)
- SetConfigurationBitRed\_LedToggleSlow, [387](#)
- SetConfigurationBitRelais, [387](#)
- SetConfigurationBitRV\_Gen, [388](#)
- SetConfigurationBitStream, [388](#)
- SetConfigurationBitSupply, [388](#)
- SetCrossTalkOffset, [388](#)
- SetCrossTalkOptimum, [388](#)
- SetDigout, [388](#)
- SetDisplayText, [388](#)
- SetDownsampleFactor, [388](#)
- SetFilter, [388](#)
- SetFilterCoeffs, [389](#)
- SetI Clamp, [389](#)
- SetIOffset, [389](#)
- SetIGain, [389](#)
- SetNoFilterCoeffs, [389](#)
- SetPGain, [389](#)
- SetRecordingNumber, [389](#)
- SetScreen, [389](#)
- SetSimulation, [389](#)
- SetU Clamp, [390](#)
- SetUOffset, [390](#)
- SetUVOffset, [390](#)
- SetXGain, [390](#)
- StopTable, [390](#)
- Table\_Wait, [390](#)
- TableDefBegin, [390](#)
- TableDefEnd, [390](#)
- UpdateDisplay, [390](#)
- CRoboDeviceNet, [391](#)
  - ~CRoboDeviceNet, [394](#)
  - Axes\_I, [401](#)
  - Axes\_X, [401](#)
  - Axes\_Y, [401](#)
  - Axes\_Z, [401](#)
  - Axis\_I, [401](#)
  - Axis\_X, [401](#)
  - Axis\_Y, [401](#)
  - Axis\_Z, [401](#)
  - CancelPoolLoop, [394](#)
  - CancelPoolLoopAndStopMovement, [394](#)
  - CRoboDeviceNet, [394](#)
  - FindReference, [394](#)
  - GetAirpressure, [395](#)
  - GetAirpressureLimit, [395](#)
  - GetAirValve, [395](#)
  - GetCurrentAirvalve, [395](#)
  - GetCurrentAirvalveLimit, [395](#)
  - GetCurrentPosition, [395](#)
  - GetErrorAirpressure, [396](#)
  - GetErrorCurrentAirvalve, [396](#)
  - GetErrorVoltage12V, [396](#)
  - GetErrorVoltage5V, [396](#)
  - GetErrorVoltageAirvalve, [396](#)
  - GetErrorVoltageRs485A, [396](#)
  - GetErrorVoltageRs485B, [396](#)
  - GetErrorVoltageValves, [396](#)
  - GetInMovement, [396](#)
  - GetMinPressure, [396](#)
  - GetMovementError, [396](#)
  - GetVoltage12V, [397](#)
  - GetVoltage12VLimit, [397](#)
  - GetVoltage5V, [397](#)
  - GetVoltage5VLimit, [397](#)
  - GetVoltageAirvalve, [397](#)
  - GetVoltageAirvalveLimit, [397](#)
  - GetVoltageRs485A, [397](#)
  - GetVoltageRs485ALimit, [397](#)
  - GetVoltageRs485B, [397](#)
  - GetVoltageRs485BLimit, [398](#)
  - GetVoltageValves, [398](#)
  - GetVoltageValvesLimit, [398](#)
  - McsBus, [405](#)
  - McsBus\_MotorControl, [405](#)
  - McsBus\_XY, [402](#)
  - McsBus\_ZI, [402](#)
  - MoveAbs, [398](#)
  - RoboError\_AnotherMaster, [402](#)
  - RoboError\_Base, [402](#)
  - RoboError\_CannotEscapeEndSwitch, [402](#)
  - RoboError\_CommandAlreadyInProgress, [402](#)
  - RoboError\_CommandNotPossible, [402](#)
  - RoboError\_CommunicationTimeout, [402](#)
  - RoboError\_DacqNotReady, [402](#)
  - RoboError\_DLLMovementTimeout, [403](#)
  - RoboError\_FindReferenceMethod, [403](#)
  - RoboError\_GilsonCommandPending, [403](#)
  - RoboError\_GilsonTimeout, [403](#)
  - RoboError\_GilsonWrondID, [403](#)
  - RoboError\_McsBus\_UnknownCommand, [403](#)
  - RoboError\_NoEndSwitch, [403](#)
  - RoboError\_NoMoreData, [403](#)
  - RoboError\_NoReference, [403](#)
  - RoboError\_NoSpeedOrAcceleration, [403](#)
  - RoboError\_OverPressure, [404](#)
  - RoboError\_ParameterNotAllowed, [404](#)
  - RoboError\_PeristalticTimeout, [404](#)
  - RoboError\_Phase0OutOfRange, [404](#)



- RoboError\_PollLoopCanceled, [404](#)
- RoboError\_PollLoopCanceledAndStopMovement, [404](#)
- RoboError\_Pressure, [404](#)
- RoboError\_RangeExceeded, [404](#)
- RoboError\_StateChangeNotPossible, [404](#)
- RoboError\_Timeout, [404](#)
- RoboError\_UnknownCommand, [405](#)
- RoboMainLowLevelCommand, [405](#)
- RoboStatusEvent, [405](#)
- SetAirpressureLimit, [398](#)
- SetAirValve, [399](#)
- SetCurrentAirvalveLimit, [399](#)
- SetCurrentAndAir, [399](#)
- SetInMovement, [399](#)
- SetMinPressure, [399](#)
- SetVoltage12VLimit, [399](#)
- SetVoltage5VLimit, [399](#)
- SetVoltageAirvalveLimit, [400](#)
- SetVoltageRs485ALimit, [400](#)
- SetVoltageRs485BLimit, [400](#)
- SetVoltageValvesLimit, [400](#)
- StopMovement, [400](#)
- CRoboDeviceNet::RoboMainLowLevelCommands, [612](#)
  - FindReferencePhase0, [613](#)
  - GetAxisConfig, [613](#)
  - GetHWConfig, [613](#)
  - GetHWRevision, [614](#)
  - GetMaxNoPressure, [614](#)
  - GetMaxNoPressureWaitTime, [614](#)
  - GetMaxPressureWaitTime, [614](#)
  - GetMinNoPressureWaitTime, [614](#)
  - GetMinPressure, [614](#)
  - GetMinPressureWaitTime, [614](#)
  - GetParameter, [614](#)
  - GetPhases, [614](#)
  - GetSearchReferenceFastAccel, [615](#)
  - GetSearchReferenceFastSpeed, [615](#)
  - GetSearchReferenceFineAccel, [615](#)
  - GetSearchReferenceFineSpeed, [615](#)
  - GetSearchReferenceMethod, [615](#)
  - GetSearchReferenceMoveOut, [615](#)
  - GetSearchReferenceOffsetPos, [615](#)
  - GetUserParameter, [615](#), [616](#)
  - HasRef, [616](#)
  - SetAxisConfig, [616](#)
  - SetHWConfig, [616](#)
  - SetHWRevision, [616](#)
  - SetMaxNoPressure, [616](#)
  - SetMaxNoPressureWaitTime, [616](#)
  - SetMaxPressureWaitTime, [617](#)
  - SetMinNoPressureWaitTime, [617](#)
  - SetMinPressure, [617](#)
  - SetMinPressureWaitTime, [617](#)
  - SetParameter, [617](#)
  - SetSearchReferenceFastAccel, [617](#)
  - SetSearchReferenceFastSpeed, [617](#)
  - SetSearchReferenceFineAccel, [618](#)
  - SetSearchReferenceFineSpeed, [618](#)
  - SetSearchReferenceMethod, [618](#)
  - SetSearchReferenceMoveOut, [618](#)
  - SetSearchReferenceOffsetPos, [618](#)
  - SetUserParameter, [618](#), [619](#)
- CRoboFluidDeviceNet, [405](#)
  - ~CRoboFluidDeviceNet, [406](#)
  - CloseAllValves, [406](#)
  - CRoboFluidDeviceNet, [406](#)
  - GetPumpSpeed, [407](#)
  - GetSingleValve, [407](#)
  - GetValve, [407](#)
  - IsPumpMotorOn, [407](#)
  - m\_pMcsBus\_MotorControlNet, [408](#)
  - m\_pRoboFluidDevice, [408](#)
  - McsBus\_MotorControl, [408](#)
  - PumpOff, [407](#)
  - PumpOn, [407](#)
  - SetPumpSpeed, [407](#)
  - SetSingleValve, [408](#)
  - SetValve, [408](#)
- CRoboInjectDeviceNet, [409](#)
  - CRoboInjectDeviceNet, [409](#)
- CRoboocyte2DeviceNet, [409](#)
  - CRoboocyte2DeviceNet, [410](#)
  - GetAxisLED, [410](#)
  - GetGilsonDevice, [410](#)
  - GetMcsBus\_Extension, [410](#)
  - GetRoboDacq, [410](#)
  - GetRoboFluidDevice, [410](#)
  - SetAxisLED, [411](#)
- CRoboStatorDeviceNet, [411](#)
  - CRoboStatorDeviceNet, [412](#)
  - FindReferenceI, [412](#)
  - FindReferenceXY, [412](#), [413](#)
  - FindReferenceZ, [413](#)
  - GetCurrentPositionI, [413](#)
  - GetCurrentPositionXY, [413](#)
  - GetCurrentPositionZ, [413](#)
  - HasRefI, [413](#)
  - HasRefXY, [413](#)
  - HasRefZ, [413](#)
  - MoveAbsI, [413](#), [414](#)
  - MoveAbsXY, [414](#)
  - MoveAbsZ, [414](#)
  - RoboMainStatorLowLevelCommand, [417](#)
  - SetAccelerationI, [414](#)
  - SetAccelerationNativeI, [414](#)
  - SetAccelerationNativeXY, [414](#)
  - SetAccelerationNativeZ, [414](#)
  - SetAccelerationXY, [415](#)
  - SetAccelerationZ, [415](#)
  - SetCurrentAndAirXY, [415](#)
  - SetSpeedI, [415](#)
  - SetSpeedNativeI, [415](#)
  - SetSpeedNativeXY, [415](#)
  - SetSpeedNativeZ, [415](#)
  - SetSpeedXY, [415](#)

- SetSpeedZ, 416
- SetVelocityI, 416
- SetVelocityXY, 416
- SetVelocityZ, 416
- StopMovementI, 416
- StopMovementXY, 416
- StopMovementZ, 416
- CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands, 619
  - FindReferencePhase0XY, 619
- CSafeISDeviceNet, 417
  - ~CSafeISDeviceNet, 418
  - CSafeISDeviceNet, 418
  - DacqDevice, 419
  - FluidControlDevice, 420
  - RoboDevice, 420
  - SetAdcChannels, 418
  - SetAdcSamplePos, 418
  - SetDacMode, 418
  - SetDacPeriode, 419
  - SetDacPulseform, 419
  - SetSwitches, 419
- CSCUDacqGroupChannelSelectionNet, 420
  - CSCUDacqGroupChannelSelectionNet, 420
- CSCUFunctionNet, 421
  - !CSCUFunctionNet, 423
  - ~CSCUFunctionNet, 423
  - AutomaticAnalogOut, 423
  - CSCUFunctionNet, 423
  - EnableAnalogOut, 424
  - GetAnalogOutADCRange, 424
  - GetAnalogOutChannels, 424
  - GetAnalogOutDACRange, 424
  - GetAvailableHeadstages, 424
  - GetAvailableHeadstagesEvent, 435
  - GetFilterProperties, 425
  - GetFilterProperty, 425
  - GetHeadstageAdcBits, 425
  - GetHeadstageAdcRangeInMicroVolt, 426
  - GetHeadstageDacBits, 426
  - GetHeadstageDacCurrentRangeInMicroAmpere, 426
  - GetHeadstageDacCurrentResolutionInNanoAmpere, 427
  - GetHeadstageDacVoltageRangeInMilliVolt, 427
  - GetHeadstageDacVoltageResolutionInMicroVolt, 427
  - GetHeadstageGainInPer mille, 428
  - GetHeadstageID, 428
  - GetHeadstageNumberOfAnalogChannels, 428
  - GetHeadstageNumberOfStimulationChannels, 429
  - GetHeadstagePowerStateAtStart, 429
  - GetHeadstageSamplerate, 429
  - GetHeadstageSerialNumber, 430
  - GetMaxNumberOfHeadstages, 430
  - GetMaxStimulusChannelsPerHeadstage, 430
  - GetReferenceElectrodeMode, 430
  - GetReferenceElectrodeSwitchState, 431
  - HasAnalogOut, 431
  - HasGalvanicIsolation, 431
  - HasHSPowerSwitch, 431
  - IsAnalogOutEnabled, 432
  - IsAutomaticAnalogOut, 432
  - IsHeadstageAvailable, 432
  - IsHeadstageAvailableEvent, 435
  - IsHSPowered, 432
  - IsInDacqLegacyMode, 433
  - OnGetAvailableHeadstages, 433
  - OnIsHeadstageAvailable, 433
  - PowerHS, 433
  - SetAnalogOutADCRange, 433
  - SetAnalogOutChannels, 434
  - SetAnalogOutDACRange, 434
  - SetDacqLegacyMode, 434
  - SetHeadstagePowerStateAtStart, 434
  - SetReferenceElectrodeMode, 435
  - SetReferenceElectrodeSwitchState, 435
- CSerialPortNet, 436
  - CSerialPortNet, 436
  - GetBytesAvailable, 436
  - Receive, 436
  - ReceiveString, 436, 437
  - Send, 437
- CStg200xBasicNet, 437
  - ~CStg200xBasicNet, 441
  - GetAnalogRanges, 441
  - GetAnalogResolution, 442
  - GetAutocalibrationDisabled, 442
  - GetAvailableMemory, 442
  - GetBlankingEnable, 442, 443
  - GetCurrentRangeInNanoAmp, 443
  - GetCurrentResolutionInNanoAmp, 443
  - GetDacAmplificationFactor, 444
  - GetDACResolution, 444
  - GetDiginValue, 444
  - GetDigoutMode, 444
  - GetDigoutValue, 445
  - GetElectrodeDacMux, 445
  - GetElectrodeEnable, 446
  - GetElectrodeMode, 447
  - GetEnableAmplifierProtectionSwitch, 447, 448
  - GetExternalElectrodeEnable, 448
  - GetFAAmplification, 449
  - GetHeadstage, 449
  - GetListModelIndexRange, 449
  - GetListModelTriggerSource, 449
  - GetNumberOfAnalogChannels, 449
  - GetNumberOfHWDACPaths, 449
  - GetNumberOfStimulationElectrodes, 449
  - GetNumberOfStimulationSourcesPerElectrode, 450
  - GetNumberOfSyncoutChannels, 450
  - GetNumberOfTriggerInputs, 450
  - GetOutputRate, 450
  - GetStgProgramInfo, 450, 451
  - GetStgVersionInfo, 451

- GetSyncoutMap, [451](#)
- GetTotalMemory, [452](#)
- GetTriggerSource, [452](#)
- GetVoltageRangeInMicroVolt, [452](#)
- GetVoltageResolutionInMicroVolt, [452](#)
- ListModeSendStart, [453](#)
- ListModeSendStop, [453](#)
- SendStart, [453](#)
- SendStop, [453](#)
- SetAutocalibrationDisabled, [453](#)
- SetBlankingEnable, [454](#), [455](#)
- SetCurrentMode, [455](#)
- SetDacAmplificationFactor, [455](#)
- SetDigoutMode, [456](#)
- SetDigoutValue, [456](#)
- SetElectrodeDacMux, [456–458](#)
- SetElectrodeEnable, [458–460](#)
- SetElectrodeMode, [461](#), [462](#)
- SetEnableAmplifierProtectionSwitch, [462](#), [463](#)
- SetExternalElectrodeEnable, [464](#)
- SetFAAmplification, [465](#)
- SetHeadstage, [465](#)
- SetListmodeIndexRange, [465](#)
- SetListmodeTriggerSource, [465](#)
- SetMeasurementMode, [465](#)
- SetOutputRate, [466](#)
- SetStgProgramInfo, [466](#)
- SetSyncoutMap, [466](#)
- SetTriggerSource, [466](#), [467](#)
- SetVoltageMode, [467](#)
- CStg200xDownloadBasicNet, [467](#)
  - ClearChannelData, [469](#)
  - ClearSyncData, [469](#)
  - DisableAutoReset, [469](#)
  - EnableAutoReset, [470](#)
  - ForceStatusEvent, [470](#)
  - GetMemoryUsageDAC, [470](#)
  - GetMemoryUsageSyncout, [470](#)
  - GetSweepCount, [471](#)
  - GetTrigger, [471](#)
  - ResetStatus, [471](#)
  - SendChannelData, [473](#)
  - SendSyncData, [473](#)
  - SetupRetriggerMode, [474](#)
  - SetupTrigger, [474](#)
  - SetupTriggerSingle, [475](#)
  - Stimulus, [476](#)
- CStg200xDownloadNet, [476](#)
  - ~CStg200xDownloadNet, [477](#)
  - ClearChannel\_PrepareAndSendData, [478](#)
  - CStg200xDownloadNet, [477](#)
  - DisableMultiFileMode, [478](#)
  - EnableMultiFileMode, [478](#)
  - GetModuleCurrent, [479](#)
  - GetModuleTemp, [479](#)
  - MwPollStatusEvent, [482](#)
  - PrepareAndAppendData, [479](#)
  - PrepareAndSendData, [480](#)
  - QueryTriggerstatus, [481](#)
  - SendSegmentDefine, [481](#)
  - SendSegmentSelect, [481](#)
  - SendSegmentStart, [482](#)
  - SetOutputMap, [482](#)
  - Stg200xPollStatusEvent, [482](#)
- CStimulusFunctionNet, [483](#)
  - ClearChannel\_PrepareAndSendData, [485](#)
  - ClearChannelData, [485](#)
  - ClearMultiplexedData, [485](#)
  - ClearSyncData, [485](#)
  - CreateSideband, [485](#)
  - CStimulusFunctionNet, [484](#)
  - ForceStatusEvent, [486](#)
  - GetAvailableMemory, [486](#)
  - GetCurrentRangeInNanoAmp, [487](#)
  - GetCurrentResolutionInNanoAmp, [487](#)
  - GetDACResolution, [487](#)
  - GetMultiplexedDataChannelsInBlock, [487](#)
  - GetNumberOfAnalogChannels, [488](#)
  - GetTotalMemory, [488](#)
  - GetVoltageRangeInMicroVolt, [488](#)
  - GetVoltageResolutionInMicroVolt, [488](#)
  - PollStatusEvent, [494](#)
  - PrepareAndAppendData, [489](#)
  - PrepareAndSendData, [490](#)
  - PrepareData, [490](#)
  - SendMultiplexedData, [491](#)
  - SendPreparedData, [491](#)
  - SendStart, [491](#)
  - SendStop, [491](#)
  - SetupTrigger, [492](#)
  - SetupTriggerSingle, [492](#)
  - StartPoll, [494](#)
  - StopPoll, [494](#)
- CStimulusFunctionNet::SidebandData, [621](#)
  - !SidebandData, [622](#)
  - ~SidebandData, [622](#)
  - Duration, [622](#)
  - Sideband, [622](#)
  - SidebandData, [621](#)
- CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, [623](#)
  - !StimulusDeviceDataAndUnrolledData, [623](#)
  - ~StimulusDeviceDataAndUnrolledData, [623](#)
  - DeviceData, [624](#)
  - DeviceDataLength, [624](#)
  - StimulusDeviceDataAndUnrolledData, [623](#)
  - UnrolledAmplitude, [624](#)
  - UnrolledDuration, [624](#)
  - UnrolledSync, [624](#)
- CSw2to64DeviceNet, [494](#)
  - ~CSw2to64DeviceNet, [495](#)
  - CSw2to64DeviceNet, [495](#)
  - GetChannel, [495](#)
  - GetChannels, [496](#)
  - GetNumber, [496](#)
  - SetChannel, [496](#)

- SetChannels, [496](#)
- CTcxDeviceNet, [497](#)
  - ~CTcxDeviceNet, [499](#)
  - CalibrateThermocouple, [499](#)
  - CTcxDeviceNet, [499](#)
  - FactoryReset, [499](#)
  - GetBoardTemp, [500](#)
  - GetCalibration, [500](#)
  - GetCalibrationDecp, [500](#)
  - GetCalibrationMax, [500](#)
  - GetCalibrationMin, [500](#)
  - GetCurrent, [500](#)
  - GetD, [500](#)
  - GetDDecp, [500](#)
  - GetDevice, [500](#)
  - GetDeviceType, [501](#)
  - GetDevname, [501](#)
  - GetDMax, [501](#)
  - GetDMin, [501](#)
  - GetDuty, [501](#)
  - GetEnableHeaterLimit, [501](#)
  - GetEnableThermocouple, [501](#)
  - GetHasThermocouple, [501](#)
  - GetHeaterLimit, [502](#)
  - GetHeaterTemp, [502](#)
  - GetI, [502](#)
  - GetIDecp, [502](#)
  - GetIMax, [502](#)
  - GetIMin, [502](#)
  - GetIOut, [502](#)
  - GetMaxHeaterPowerMultiwell, [503](#)
  - GetMaxP, [503](#)
  - GetMaxpDecp, [503](#)
  - GetMaxpMax, [503](#)
  - GetMaxpMin, [503](#)
  - GetNumControlChannels, [503](#)
  - GetNumDevices, [503](#)
  - GetNumMeasureChannels, [503](#)
  - GetOnOff, [504](#)
  - GetP, [504](#)
  - GetPDecp, [504](#)
  - GetPMax, [504](#)
  - GetPMin, [504](#)
  - GetPOut, [504](#)
  - GetPwrOut, [504](#)
  - GetPwrSet, [505](#)
  - GetRes1, [505](#)
  - GetRes2, [505](#)
  - GetResS, [505](#)
  - GetResX, [505](#)
  - GetROut, [505](#)
  - GetSensorType, [505](#)
  - GetSetpoint, [505](#)
  - GetSetpointDecp, [506](#)
  - GetSetpointMax, [506](#)
  - GetSetpointMin, [506](#)
  - GetThermocoupleCalibration, [506](#)
  - GetThermocoupleNanovoltPerKelvin, [506](#)
  - GetThermocoupleReferenceTemp, [506](#)
  - GetThermocoupleTemp, [506](#)
  - GetThermocoupleTempAbs, [507](#)
  - GetUnit, [507](#)
  - GetUOut, [507](#)
  - GetValue, [507](#)
  - GetValueHires, [507](#)
  - GetVolti, [507](#)
  - SetCalibration, [507](#)
  - SetD, [508](#)
  - SetDevice, [508](#)
  - SetDeviceType, [508](#)
  - SetDevname, [508](#)
  - SetEnableHeaterLimit, [508](#)
  - SetEnableThermocouple, [508](#)
  - SetHeaterLimit, [508](#)
  - SetI, [508](#)
  - SetMaxHeaterPowerMultiwell, [509](#)
  - SetMaxP, [509](#)
  - SetOnOff, [509](#)
  - SetP, [509](#)
  - SetSensorType, [509](#)
  - SetSetpoint, [509](#)
  - SetThermocoupleNanovoltPerKelvin, [510](#)
- CTEERFunctionNet, [510](#)
  - !CTEERFunctionNet, [512](#)
  - ~CTEERFunctionNet, [512](#)
  - CancelInternalCalibration, [512](#)
  - CTEERFunctionNet, [512](#)
  - GetAdapterCode, [513](#)
  - GetAdcOffsetU1, [513](#)
  - GetAdcOffsetU2, [513](#)
  - GetAmplitude\_nA, [513](#)
  - GetBytesPerSample, [513](#)
  - GetClampMode, [514](#)
  - GetControllerParams, [514](#)
  - GetCurrentEnable, [514](#)
  - GetDacZero, [514](#)
  - GetFrameErrorCounter, [514](#)
  - GetLiquidResistance, [515](#)
  - GetMaxChunkSize\_Byte, [515](#)
  - GetNumberOfAvailableSamples, [515](#)
  - GetPeriod\_us, [515](#)
  - GetRotaryPositionCode, [515](#)
  - GetSampleBufferChunk, [515](#)
  - GetSampleRate, [516](#)
  - GetSampleVoltageBuffer\_uV, [516](#)
  - GetScaleFactorU1, [516](#)
  - GetScaleFactorU2, [516](#)
  - GetUptimeSeconds, [517](#)
  - GetWaveform, [517](#)
  - IsInternalCalibrationFinished, [517](#)
  - IsSamplingFinished, [517](#)
  - SetAmplitude\_nA, [517](#)
  - SetBufferIndex, [518](#)
  - SetClampMode, [518](#)
  - SetControllerParams, [518](#)
  - SetCurrentEnable, [518](#)

- SetExternalLED, [519](#)
- SetLiquidResistance, [519](#)
- SetPeriod\_us, [519](#)
- SetWaveform, [519](#)
- StartInternalCalibration, [520](#)
- StartSampling, [520](#)
- StopSampling, [520](#)
- CTEERMachineDeviceNet, [520](#)
  - ~CTEERMachineDeviceNet, [521](#)
  - CTEERMachineDeviceNet, [521](#)
  - TEERFunctionNet, [521](#)
- CurrentRangeInNanoAmp
  - W2100\_StimulusParametersNet, [625](#)
- CurrentResolutionInNanoAmp
  - W2100\_StimulusParametersNet, [625](#)
- CUsbDeviceConfigurationFunctionNet, [521](#)
  - !CUsbDeviceConfigurationFunctionNet, [522](#)
  - ~CUsbDeviceConfigurationFunctionNet, [522](#)
  - CUsbDeviceConfigurationFunctionNet, [522](#)
  - SetDeviceId, [522](#)
  - SetDeviceName, [522](#)
- CUsbExceptionNet, [523](#)
  - CUsbExceptionNet, [523](#), [524](#)
  - Status, [524](#)
- CutoffFrequency
  - CCreateFilterNet, [49](#)
- CVoltageRangeInfoNet
  - CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet, [524](#)
- CW2100\_FunctionNet, [525](#)
  - ClearStimulusParametersCache, [527](#)
  - ClearUserDefinedNameCache, [527](#)
  - CW2100\_FunctionNet, [526](#)
  - DeselectAllHeadstages, [527](#)
  - DeselectHeadstage, [527](#)
  - GetAccelGyroCurrentRate, [527](#)
  - GetAccelGyroDesiredRate, [527](#)
  - GetAccelGyroEnabled, [527](#)
  - GetAccelRange, [527](#)
  - GetAnalogOutChannel, [528](#)
  - GetAnalogOutFilter, [528](#)
  - GetAudioChannels, [528](#)
  - GetAvailableHeadstages, [528](#)
  - GetBatteryState, [528](#)
  - GetDacRange, [528](#)
  - GetFilterProperties, [528](#)
  - GetFilterProperty, [528](#)
  - GetFPGA FirmwareType, [528](#)
  - GetGyroRange, [528](#)
  - GetHeadstageOnOff, [529](#)
  - GetHeadstageSamplingActive, [529](#)
  - GetMultiHeadstageMode, [529](#)
  - GetPicFirmwareType, [529](#)
  - GetSelectedChannels, [529](#)
  - GetSelectedHeadstageState, [529](#)
  - GetStimulusParametersCache, [529](#)
  - GetStimulusParametersFromSelectedHS, [529](#)
  - GetStimulusParameters, [530](#)
  - GetUserDefinedName, [530](#)
  - GetUserDefinedNameCache, [530](#)
  - GetUserDefinedNameFromSelectedHS, [530](#)
  - PulseGenerator, [532](#)
  - SelectHeadstage, [530](#)
  - SetAccelGyroDesiredRate, [530](#)
  - SetAccelGyroEnabled, [530](#)
  - SetAccelRange, [530](#)
  - SetAnalogOutChannel, [531](#)
  - SetAnalogOutFilter, [531](#)
  - SetAudioChannels, [531](#)
  - SetDacRange, [531](#)
  - SetGyroRange, [531](#)
  - SetHeadstageOnOff, [531](#)
  - SetHeadstageSamplingActive, [531](#)
  - SetHeadstageToSleep, [531](#)
  - SetMultiHeadstageMode, [532](#)
  - SetSelectedChannels, [532](#)
  - Stimulator, [532](#)
- CW2100\_FunctionNet::AudioChannelsNet, [29](#)
  - amplification, [29](#)
  - channel, [29](#)
  - dacqgroup, [29](#)
- CW2100\_StimulatorFunctionNet, [532](#)
  - BOOST\_BIT, [539](#)
  - ClearChannelData, [534](#)
  - CW2100\_StimulatorFunctionNet, [534](#)
  - GetBoostAlwaysOnMode, [534](#)
  - GetBoostPreTime, [534](#)
  - GetCurrentRangeInNanoAmp, [534](#)
  - GetCurrentResolutionInNanoAmp, [535](#)
  - GetDACResolution, [535](#)
  - GetDigitalStimulatorTrigger, [535](#)
  - GetDigitalStimulatorTriggerSlope, [535](#)
  - GetNumberOfAnalogChannels, [535](#)
  - GetNumberOfSyncoutChannels, [536](#)
  - GetNumberOfTriggerInputs, [536](#)
  - GetStimulationPatternMemory, [536](#)
  - GetTimeResolutionInNanoSeconds, [536](#)
  - GetTimeSlot, [536](#)
  - GetVoltageRangeInMicroVolt, [536](#)
  - GetVoltageResolutionInMicroVolt, [536](#)
  - GND\_SWITCH\_BIT, [539](#)
  - PollStatusEvent, [540](#)
  - PrepareData, [538](#)
  - PrepareDataSync, [538](#)
  - SelectTimeSlot, [538](#)
  - SendPreparedData, [538](#)
  - SendStart, [538](#)
  - SendStop, [539](#)
  - SetDigitalStimulatorTrigger, [539](#)
  - SetDigitalStimulatorTriggerSlope, [539](#)
  - StartPoll, [539](#)
  - StopPoll, [539](#)
  - SYNC\_BIT0, [540](#)
  - SYNC\_BIT1, [540](#)
- CW2100DacqGroupChannelSelectionNet, [540](#)
  - CW2100DacqGroupChannelSelectionNet, [540](#)

- CWarnerUssingDeviceNet, 541
  - ICWarnerUssingDeviceNet, 541
  - ~CWarnerUssingDeviceNet, 541
  - CWarnerUssingDeviceNet, 541
  - WarnerUssingFunction, 542
- CWarnerUssingFunctionNet, 542
  - ICWarnerUssingFunctionNet, 544
  - ~CWarnerUssingFunctionNet, 544
  - CompensateElectrodeOffset, 544
  - CWarnerUssingFunctionNet, 544
  - GetAvailableChambers, 545
  - GetChannelsCountOfChamber, 545
  - GetClampMode, 545
  - GetComplianceVoltageThreshold, 545
  - GetDacPampsPerDigitHighCurrentRange, 546
  - GetDacPampsPerDigitLowCurrentRange, 546
  - GetDacZero, 546
  - GetHighCurrentRange, 547
  - GetIdleModeOffset, 547
  - GetLiquidResistance, 547
  - GetLowCurrentRange, 548
  - GetNumberOfAvailableChambers, 548
  - GetNumberOfHardwareSlotsForChambers, 548
  - GetU1Offset, 548
  - GetU1Reference, 550
  - GetU2Offset, 550
  - GetU2Reference, 550
  - GetUnitDescription, 551
  - GetUnitExponent, 551
  - GetUnitName, 551
  - GetUnitsPerDigit, 552
  - GetUptimeSeconds, 552
  - GetVoltageClampControllerParam\_D, 552
  - GetVoltageClampControllerParam\_I, 553
  - GetVoltageClampControllerParam\_P, 553
  - IsChamberAvailable, 553
  - IsHighCurrentMode, 554
  - IsInternalCalibrationFinished, 554
  - IsPulseEnabled, 554
  - SetClampMode, 555
  - SetEnablePulse, 555
  - SetHighCurrentMode, 555
  - SetIdleModeOffset, 556
  - SetLiquidResistance, 556
  - SetLowCurrentMode, 556
  - SetPulse, 556
  - SetVoltageClampControllerParam\_D, 557
  - SetVoltageClampControllerParam\_I, 557
  - SetVoltageClampControllerParam\_P, 557
  - WaitForAllChambers, 558
  - WaitForChamber, 558
- CWarnerValveControllerDeviceNet, 558
  - ICWarnerValveControllerDeviceNet, 563
  - ~CWarnerValveControllerDeviceNet, 563
  - ClearTableName, 563
  - ClearValveTable, 563
  - CWarnerValveControllerDeviceNet, 562
  - GetActiveRunningTableNumber, 563
  - GetActiveRunningTableNumberEvent, 580
  - GetAnalogThresholdHigh, 563
  - GetAnalogThresholdHighEvent, 580
  - GetAnalogThresholdLow, 564
  - GetAnalogThresholdLowEvent, 580
  - GetAnalogVoltage, 564
  - GetAnalogVoltageEvent, 580
  - GetCurrentEditTableNumber, 564
  - GetCurrentNumberOfValves, 564
  - GetCurrentNumberOfValvesEvent, 580
  - GetDigitalOutPortValve, 565
  - GetDigitalOutPortValveEvent, 581
  - GetDigitalPortDirection, 566
  - GetDigitalPortDirectionEvent, 581
  - GetDisplayMode, 566
  - GetDisplayModeEvent, 581
  - GetTableName, 566
  - GetTableNamebyIndex, 566
  - GetTableNamebyIndexEvent, 581
  - GetTotalNumberOfDigitalPorts, 567
  - GetTotalNumberOfTables, 567
  - GetTotalNumberOfValves, 567
  - GetTotalTableSize, 567
  - GetValveActive, 567
  - GetValveActiveEvent, 581
  - GetValveBoardRevision, 568
  - GetValveBoardRevisionEvent, 581
  - GetValveBoardRevisionString, 568
  - GetValveDigitalInPort, 568
  - GetValveDigitalInPortEvent, 581
  - GetValveLedOn, 568
  - GetValveLedOnEvent, 582
  - GetValveManualGroup, 569
  - GetValveManualGroupEvent, 582
  - GetValveManualState, 569
  - GetValveManualStateEvent, 582
  - GetValveMode, 569
  - GetValveModeEvent, 582
  - GetValvesActiveMap, 569
  - GetValvesManualStateMap, 570
  - GetValveTableEntry, 570
  - IsDigitalOutPortInverted, 570
  - IsDigitalOutPortInvertedEvent, 582
  - IsValveDigitalInInverted, 570
  - IsValveDigitalInInvertedEvent, 582
  - IsValveOpen, 571
  - IsValveOpenEvent, 582
  - IsValveOpenInAnalogMode, 571
  - IsValveOpenInAnalogModeEvent, 582
  - IsValveOpenInDigitalMode, 571
  - IsValveOpenInDigitalModeEvent, 583
  - LoadValveTable, 572
  - OnGetActiveRunningTableNumber, 572
  - OnGetAnalogThresholdHigh, 572
  - OnGetAnalogThresholdLow, 572
  - OnGetAnalogVoltage, 572
  - OnGetCurrentNumberOfValves, 572
  - OnGetDigitalOutPortValve, 572



- OnGetDigitalPortDirection, [572](#)
- OnGetDisplayMode, [573](#)
- OnGetTableNamebyIndex, [573](#)
- OnGetValveActive, [573](#)
- OnGetValveBoardRevision, [573](#)
- OnGetValveDigitalInPort, [573](#)
- OnGetValveLedOn, [573](#)
- OnGetValveManualGroup, [573](#)
- OnGetValveManualState, [573](#)
- OnGetValveMode, [573](#)
- OnIsDigitalOutPortInverted, [574](#)
- OnIsValveDigitalInInverted, [574](#)
- OnIsValveOpen, [574](#)
- OnIsValveOpenInAnalogMode, [574](#)
- OnIsValveOpenInDigitalMode, [574](#)
- OnTableEntryChanged, [574](#)
- SetActiveRunningTableNumber, [574](#)
- SetAnalogThresholdHigh, [575](#)
- SetAnalogThresholdLow, [575](#)
- SetCurrentEditTableNumber, [575](#)
- SetDefault, [575](#)
- SetDigitalOutPortInvert, [575](#)
- SetDigitalOutPortValve, [576](#)
- SetDigitalPortDirection, [576](#)
- SetDisplayMode, [576](#)
- SetTableName, [576](#)
- SetTableStep, [577](#)
- SetTableStepAll, [577](#)
- SetValveActive, [577](#)
- SetValveDigitalInInvert, [577](#)
- SetValveDigitalInPort, [578](#)
- SetValveLedOn, [578](#)
- SetValveManualGroup, [578](#)
- SetValveManualState, [578](#)
- SetValveMode, [579](#)
- SetValvesActiveMap, [579](#)
- SetValvesManualStateMap, [579](#)
- SetValveTableEntry, [579](#)
- StoreValveTable, [580](#)
- TableEntryChangedEvent, [583](#)
- CWarnerValveControllerDeviceTesterFunctionNet, [583](#)
  - !CWarnerValveControllerDeviceTesterFunctionNet, [584](#)
  - ~CWarnerValveControllerDeviceTesterFunctionNet, [584](#)
  - CWarnerValveControllerDeviceTesterFunctionNet, [584](#)
  - GetIO, [584](#)
  - GetSync, [584](#)
  - SetADC, [585](#)
  - SetIO, [585](#)
  - SetIODirection, [585](#)
  - SetTrigger, [585](#)
  - SetTriggerSyncDirection, [586](#)
- CWClassicFunctionNet, [586](#)
  - CWClassicFunctionNet, [587](#)
  - GetFilterParametersHeadstage, [587](#)
  - GetHasChecksum, [587](#)
  - GetHasRedLedHeadstage, [587](#)
  - GetHeadstageOnOff, [587](#)
  - GetResetFilter, [588](#)
  - GetRFConnectionStatus, [588](#)
  - GetRFFrequencyHeadstage, [588](#)
  - GetRFFrequencyReceiver, [588](#)
  - GetRFPower, [588](#)
  - GetScanHeadstagesResult, [588](#)
  - GetSelectedHeadstage, [588](#)
  - GetSerialNumberHeadstage, [588](#)
  - GetWPADebugMode, [588](#)
  - GetWPAType, [588](#)
  - ResetChannelmap, [589](#)
  - ScanForHeadstages, [589](#)
  - SetChannelmap, [589](#)
  - SetFilterParametersHeadstage, [589](#)
  - SetHasChecksum, [589](#)
  - SetHeadstageOnOff, [589](#)
  - SetHWSelectedChannels, [589](#)
  - SetResetFilter, [589](#)
  - SetRFFrequencyHeadstage, [589](#)
  - SetRFFrequencyReceiver, [590](#)
  - SetRFFrequencyReceiverEeprom, [590](#)
  - SetRFLostBehaviour, [590](#)
  - SetRFPower, [590](#)
  - SetSelectedHeadstage, [590](#)
  - SetSerialNumberHeadstage, [590](#)
  - SetWPADebugMode, [590](#)
  - SetWPAType, [590](#)
- CWirelessBaseFunctionNet, [591](#)
  - CreateWirelessHeadstageSerialNumberString, [591](#)
  - CWirelessBaseFunctionNet, [591](#)
- CyclePort
  - CMcsUsbNet, [246](#)
- DacqDevice
  - CSafeISDeviceNet, [419](#)
- dacqgroup
  - CW2100\_FunctionNet::AudioChannelsNet, [29](#)
- DACResolution
  - W2100\_StimulusParametersNet, [625](#)
- DataState
  - HeadStageIDTypeState, [609](#)
- DeepCopy
  - CCMOSMeaDeviceNet::CRegionOfInterestRect, [366](#)
- DefineAmplification
  - CPgaDeviceNet, [331](#)
- DefineFrequencyRange
  - CPgaDeviceNet, [331](#)
- DefineNumAmplifications
  - CPgaDeviceNet, [331](#)
- DefineNumFrequencyRanges
  - CPgaDeviceNet, [331](#)
- DeselectAllHeadstages
  - CW2100\_FunctionNet, [527](#)
- DeselectHeadstage
  - CW2100\_FunctionNet, [527](#)

- DetectChipType
  - CCMOSMea\_FunctionNet, [36](#)
- DEVICE\_NOT\_FOUND
  - Mcs::Usb, [27](#)
- DeviceArrival
  - CMcsUsbListNet, [239](#)
- DeviceData
  - CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, [624](#)
- DeviceDataLength
  - CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, [624](#)
- DeviceId
  - CMcsUsbListEntryNet, [236](#)
- DeviceIdNet, [591](#)
  - BcdDevice, [593](#)
  - BusType, [593](#)
  - DeviceIdNet, [592](#)
  - IdProduct, [593](#)
  - IdVendor, [593](#)
  - operator=, [592](#)
- DeviceName
  - CMcsUsbListEntryNet, [236](#)
- DeviceRemoval
  - CMcsUsbListNet, [239](#)
- DigitalSource
  - DigitalSource< digitalsourceenum >, [593](#)
- DigitalSource< digitalsourceenum >, [593](#)
  - DigitalSource, [593](#)
  - MaxBitNumber, [594](#)
  - MaxBitNumberStatic, [594](#)
  - size, [594](#)
  - Source, [594](#)
- DigitalSourceGeneral, [594](#)
  - DigitalSourceGeneral, [595](#)
  - MaxBitNumber, [595](#)
  - size, [595](#)
  - Source, [595](#)
- DisableAutoReset
  - CStg200xDownloadBasicNet, [469](#)
- DisableMultiFileMode
  - CStg200xDownloadNet, [478](#)
- Disconnect
  - CMcsUsbNet, [246](#)
- DisConnectDevice
  - CRadioControlledDevicesNet, [365](#)
- DoRamp
  - CRoboDacqNet, [380](#)
- DownloadFirmware
  - CMcsUsbFactoryNet, [224](#)
- DriverVersionNet, [596](#)
  - ~DriverVersionNet, [597](#)
  - DriverVersionNet, [597](#)
  - DriverVersionNet::FormatVersion, [597](#)
  - GetDestinationCode, [597](#)
  - GetDestinationName, [597](#), [598](#)
  - GetMajor, [598](#)
  - GetMinor, [598](#)
  - GetNumEntries, [599](#)
  - GetSerialNumber, [599](#)
  - GetStatus, [599](#)
  - GetVersionInt, [600](#)
  - GetVersionString, [600](#)
- DriverVersionNet::FormatVersion
  - DriverVersionNet, [597](#)
- ISBData, [622](#)
  - FirmwareDestinationNames, [602](#)
- DummyCommand
  - CMcsUsbNet, [246](#)
- EMU\_H3DeviceNet, [122](#)
  - Duration
    - CStimulusFunctionNet::SidebandData, [622](#)
- ElectricalStimulation
  - HeadStageIDType, [605](#)
- EmptyKey
  - CMcsUsbNet, [246](#)
- Emu\_GetCellCapacity
  - CRoboDacqNet, [380](#)
- Emu\_GetCellPotential
  - CRoboDacqNet, [380](#)
- Emu\_GetCellResists
  - CRoboDacqNet, [380](#)
- Emu\_GetElectrodeResists
  - CRoboDacqNet, [381](#)
- Emu\_GetNoise
  - CRoboDacqNet, [381](#)
- Emu\_SetCellCapacity
  - CRoboDacqNet, [381](#)
- Emu\_SetCellPotential
  - CRoboDacqNet, [381](#)
- Emu\_SetCellResists
  - CRoboDacqNet, [381](#)
- Emu\_SetElectrodeResists
  - CRoboDacqNet, [381](#)
- Emu\_SetNoise
  - CRoboDacqNet, [381](#)
- EnableAnalogOut
  - CSCUFunctionNet, [424](#)
- EnableAutoReset
  - CStg200xDownloadBasicNet, [470](#)
- EnableChannelsInGroup
  - CCMOSMea\_FunctionNet, [36](#), [37](#)
  - CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet >, [54](#)
- EnableChecksum
  - CMeaDeviceNet, [278](#)
  - COctoPotDeviceNet, [319](#)
- EnableDigitalIn
  - CMeaDeviceNet, [278](#), [279](#)
  - COctoPotDeviceNet, [319](#)
- EnableExceptions
  - CMcsUsbNet, [246](#)
- EnableMultiFileMode
  - CStg200xDownloadNet, [478](#)
- EnableTimestamp



- CMeaDeviceNet, 279
- COctoPotDeviceNet, 319
- EnableUserTrigger
  - CLIH3DeviceNet, 122
- enCMosMeaChipType
  - Mcs::Usb, 27
- EnSTG200x\_STATUS
  - Mcs::Usb, 27
- Entry
  - HeadStageIDType, 606
- Equals
  - CMcsUsbListEntryNet, 233
  - HeadStageIDType, 605
  - HeadstageIDTypeObject, 608
- EraseEepromRegisterPreconfig
  - CMcsUsbNet, 246
- EraseFilterParameterPermanent
  - CFilterConfigurationNet, 69
  - CFilterConfigurationRegisterNet, 71
- ErasePermanentAdcOffset
  - CLIH3DeviceNet, 123
- ErasePermanentDacOffset
  - CLIH3DeviceNet, 123
- Error\_Callback\_Aquisition\_Stopped
  - CMcsUsbDacqNet, 218
- Error\_Callback\_Data\_lost
  - CMcsUsbDacqNet, 218
- Error\_Callback\_Frames\_Lost
  - CMcsUsbDacqNet, 218
- Error\_Callback\_Packet\_Error
  - CMcsUsbDacqNet, 218
- Error\_Callback\_Queue\_Full
  - CMcsUsbDacqNet, 219
- Error\_Callback\_RingQueue\_Full
  - CMcsUsbDacqNet, 219
- ErrorEvent
  - CMcsUsbDacqNet, 219
- FactoryReset
  - CTcxDeviceNet, 499
- FeedbackGetSampleTimerCount
  - CMeaFeedbackFunctionNet, 287
- FeedbackSetAnalogSource
  - CMeaFeedbackFunctionNet, 287
- FeedbackSetChannelFilter
  - CMeaFeedbackFunctionNet, 287
- FeedbackSetDigitalMapping
  - CMeaFeedbackFunctionNet, 287
- FeedbackSetFeedback
  - CMeaFeedbackFunctionNet, 287
- FeedbackSetFilterOff
  - CMeaFeedbackFunctionNet, 288
- FeedbackSetFilterParameter
  - CMeaFeedbackFunctionNet, 288
- FeedbackSetFilterParameter32
  - CMeaFeedbackFunctionNet, 288
- FeedbackSetGlobalChannelFilter
  - CMeaFeedbackFunctionNet, 288
- FeedbackSetIIRFilterParameter
  - CMeaFeedbackFunctionNet, 288
- FeedbackSetLogic
  - CMeaFeedbackFunctionNet, 288
- FeedbackSetMkFilter
  - CMeaFeedbackFunctionNet, 288
- FeedbackSetNumberOfLogics
  - CMeaFeedbackFunctionNet, 289
- FeedbackSetNumberOfRateCounter
  - CMeaFeedbackFunctionNet, 289
- FeedbackSetNumberOfRateDetectors
  - CMeaFeedbackFunctionNet, 289
- FeedbackSetNumberOfSpikeDetectors
  - CMeaFeedbackFunctionNet, 289
- FeedbackSetNumberOfTriggers
  - CMeaFeedbackFunctionNet, 289
- FeedbackSetRateCounter
  - CMeaFeedbackFunctionNet, 289
- FeedbackSetRateDetector
  - CMeaFeedbackFunctionNet, 289
- FeedbackSetSpikeDetectorThreshold
  - CMeaFeedbackFunctionNet, 289
- FeedbackSetTrigger
  - CMeaFeedbackFunctionNet, 290
- FilterActive
  - CFilterPropertyNet, 73
- FilterBand
  - CFilterPropertyNet, 73
- FilterFamily
  - CFilterPropertyNet, 73
- FilterType
  - CFilterPropertyNet, 73
- FindFilter
  - CCreateFilterNet, 49
- FindFirmwareVersionMagicInBuffer
  - CMcsUsbFactoryNet, 224
- FindReference
  - CRoboDeviceNet, 394
- FindReferenceI
  - CRoboStatorDeviceNet, 412
- FindReferencePhase0
  - CRoboDeviceNet::RoboMainLowLevelCommands, 613
- FindReferencePhase0XY
  - CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands, 619
- FindReferenceXY
  - CRoboStatorDeviceNet, 412, 413
- FindReferenceZ
  - CRoboStatorDeviceNet, 413
- FirePressurePulse
  - CPPCFunctionNet, 347
- FirmwareDestinationNames, 601
  - Altera, 601
  - Bootstrap, 601
  - BUS1\_MCSBUS1, 601
  - BUS1\_MCSBUS2, 602
  - DSP, 602
  - FPGA2, 602

- FPGA3, [602](#)
- FPGA4, [602](#)
- FPGA5, [602](#)
- FPGA6, [602](#)
- MCSBUS1, [602](#)
- MCSBUS10, [602](#)
- MCSBUS11, [602](#)
- MCSBUS12, [602](#)
- MCSBUS13, [603](#)
- MCSBUS2, [603](#)
- MCSBUS3, [603](#)
- MCSBUS4, [603](#)
- MCSBUS5, [603](#)
- MCSBUS6, [603](#)
- MCSBUS7, [603](#)
- MCSBUS8, [603](#)
- MCSBUS9, [603](#)
- MCU1, [603](#)
- PIC, [603](#)
- PIC2, [604](#)
- PIC3, [604](#)
- PIC4, [604](#)
- USB, [604](#)
- FluidControlDevice
  - CSafeISDeviceNet, [420](#)
- ForceStatusEvent
  - CStg200xDownloadBasicNet, [470](#)
  - CStimulusFunctionNet, [486](#)
- FPGA2
  - FirmwareDestinationNames, [602](#)
- FPGA3
  - FirmwareDestinationNames, [602](#)
- FPGA4
  - FirmwareDestinationNames, [602](#)
- FPGA5
  - FirmwareDestinationNames, [602](#)
- FPGA6
  - FirmwareDestinationNames, [602](#)
- FromIntPtr
  - StgStatusNet, [622](#)
- FromPtr
  - StgStatusNet, [622](#)
- FX3MCSDDataAddress
  - CMcsUsbFactoryNet, [229](#)
- FX3MCSDDataDeviceIdOffset
  - CMcsUsbFactoryNet, [229](#)
- FX3MCSDDataIFB1ImageOffset
  - CMcsUsbFactoryNet, [229](#)
- FX3MCSDDataIFB2ImageOffset
  - CMcsUsbFactoryNet, [229](#)
- FX3MCSDDataVersionOffset
  - CMcsUsbFactoryNet, [229](#)
- FYIProgram
  - CFYIDeviceNet, [81](#)
- FYITemp
  - CFYIDeviceNet, [81](#)
- Gain
  - CMeaDeviceNet, [283](#)
- Get2AnalogInput
  - CMcsBus\_SensorNet, [154](#)
- Get2DigitalInput
  - CMcsBus\_SensorNet, [154](#)
- Get4ADC
  - CMcsBus\_SensorNet, [154](#)
- Get4ADCAverage
  - CMcsBus\_SensorNet, [154](#)
- Get4ADCCatchampAverageShift
  - CMcsBus\_SensorNet, [154](#)
- Get4ADCMode
  - CMcsBus\_SensorNet, [155](#)
- Get4DAC
  - CMcsBus\_SensorNet, [155](#)
- GetAbsMaxCurrentInMicroAmp
  - CMultiwellOptoStimFunctionNet, [313](#)
- GetAccelGyroCurrentRate
  - CW2100\_FunctionNet, [527](#)
- GetAccelGyroDesiredRate
  - CW2100\_FunctionNet, [527](#)
- GetAccelGyroEnabled
  - CW2100\_FunctionNet, [527](#)
- GetAccelRange
  - CW2100\_FunctionNet, [527](#)
- GetActiveRunningTableNumber
  - CWarnerValveControllerDeviceNet, [563](#)
- GetActiveRunningTableNumberEvent
  - CWarnerValveControllerDeviceNet, [580](#)
- GetAdapterCode
  - CMealImpedanceDeviceNet, [291](#)
  - CTEERFunctionNet, [513](#)
- GetAdapterType
  - CMcsUsbDacqNet, [194](#)
- GetAdc
  - CFluidControlDeviceNet, [76](#)
- GetAdcDataFormat
  - CMcsUsbDacqNet, [194](#)
- GetADCInputOffset
  - CCMOSMea\_FunctionNet, [37](#)
- GetAdcOffset
  - CLIH3DeviceNet, [123](#)
  - COctoPotDeviceNet, [319](#)
- GetAdcOffsetU1
  - CTEERFunctionNet, [513](#)
- GetAdcOffsetU2
  - CTEERFunctionNet, [513](#)
- GetADCs
  - CMcsBus\_SensorNet, [155](#)
- GetADCsLoop
  - CMcsBus\_SensorNet, [155](#)
- GetAdcZero
  - CMcsUsbDacqNet, [194](#)
- GetAirpressure
  - CRoboDeviceNet, [395](#)
- GetAirpressureLimit
  - CRoboDeviceNet, [395](#)
- GetAirValve
  - CRoboDeviceNet, [395](#)

- GetAllDigout
  - CRoboDacqNet, 381
- GetAmplification
  - CPgaDeviceNet, 331
- GetAmplitude\_nA
  - CTEERFunctionNet, 513
- GetAnalogGain
  - CMeaDeviceNet, 280
- GetAnalogOutADCRRange
  - CSCUFunctionNet, 424
- GetAnalogOutChannel
  - CW2100\_FunctionNet, 528
- GetAnalogOutChannels
  - CSCUFunctionNet, 424
- GetAnalogOutDACRange
  - CSCUFunctionNet, 424
- GetAnalogOutFilter
  - CW2100\_FunctionNet, 528
- GetAnalogRanges
  - CStg200xBasicNet, 441
- GetAnalogResolution
  - CStg200xBasicNet, 442
- GetAnalogThresholdHigh
  - CWarnerValveControllerDeviceNet, 563
- GetAnalogThresholdHighEvent
  - CWarnerValveControllerDeviceNet, 580
- GetAnalogThresholdLow
  - CWarnerValveControllerDeviceNet, 564
- GetAnalogThresholdLowEvent
  - CWarnerValveControllerDeviceNet, 580
- GetAnalogValueUnit
  - CMcsUsbDacqNet, 195
- GetAnalogVoltage
  - CPPCFunctionNet, 347
  - CPPS\_FunctionNet, 356
  - CWarnerValveControllerDeviceNet, 564
- GetAnalogVoltageEvent
  - CWarnerValveControllerDeviceNet, 580
- GetAnalogVoltageRange
  - CPPCFunctionNet, 347
- GetAnalogVoltages
  - CPPS\_FunctionNet, 356
- GetArraySize
  - CMealImpedanceDeviceNet, 291
- GetAudioChannels
  - CMeaAudioFunctionNet, 264, 265
  - CW2100\_FunctionNet, 528
- GetAudioOutDacParameter
  - CLIH3DeviceNet, 123
- GetAutocalibrationDisabled
  - CStg200xBasicNet, 442
- GetAvailableBaseSamplerates
  - CCMOSMeaDeviceNet, 46
- GetAvailableChambers
  - CWarnerUssingFunctionNet, 545
- GetAvailableDeviceList
  - CRFFFunctionNet, 371
- GetAvailableDeviceListEx
  - CRFFFunctionNet, 371
- GetAvailableHeadstages
  - CSCUFunctionNet, 424
  - CW2100\_FunctionNet, 528
- GetAvailableHeadstagesEvent
  - CSCUFunctionNet, 435
- GetAvailableMemory
  - CStg200xBasicNet, 442
  - CStimulusFunctionNet, 486
- GetAvailableSampleRates
  - CMcsUsbDacqNet::CHWInfo, 113
- GetAvailableStateList
  - CRFFFunctionNet, 371
- GetAvailableStateListEx
  - CRFFFunctionNet, 371
- GetAvailableVoltageRangesInMicroVolt
  - CMcsUsbDacqNet::CHWInfo, 113
- GetAvailableVoltageRangesInMicroVoltAndStringsInMilliVolt
  - CMcsUsbDacqNet::CHWInfo, 113
- GetAxisConfig
  - CRoboDeviceNet::RoboMainLowLevelCommands, 613
- GetAxisLED
  - CRoboocyte2DeviceNet, 410
- GetAxisParametersSignedEeprom
  - CMcsBus\_AxisParametersNet, 130
- GetAxisParametersUnsignedEeprom
  - CMcsBus\_AxisParametersNet, 130
- GetBaseFrequency
  - CRFFFunctionNet, 372
- GetBaseSamplerate
  - CCMOSMeaDeviceNet, 46
- GetBath
  - CCMOSMea\_FunctionNet, 37
- GetBathMode
  - CCMOSMea\_FunctionNet, 37
- GetBatteryState
  - CW2100\_FunctionNet, 528
- GetBatteryVoltage
  - CMultiBatteryChargerDeviceNet, 299
- GetBiQuad
  - CCreateFilterNet, 49
- GetBiQuads
  - CCreateFilterNet, 49
- GetBlankingEnable
  - CStg200xBasicNet, 442, 443
- GetBoardTemp
  - CTcxDeviceNet, 500
- GetBoostAlwaysOnMode
  - CW2100\_StimulatorFunctionNet, 534
- GetBoostPreTime
  - CW2100\_StimulatorFunctionNet, 534
- GetBubbleState
  - CPPS\_FunctionNet, 356
- GetBubbleStatus
  - CMcsBus\_SensorNet, 155
- GetBuffer
  - CGenericDevelopDeviceNet, 88

- GetBusAddress
  - CMcsBusNet, [169](#)
- GetBusAddressEeprom
  - CMcsBusNet, [169](#)
- GetByteBuffer
  - CGenericDevelopDeviceNet, [89](#)
- GetBytesAvailable
  - CSerialPortNet, [436](#)
- GetBytesPerSample
  - CTEERFunctionNet, [513](#)
- GetCalibration
  - CTcxDeviceNet, [500](#)
- GetCalibrationDecp
  - CTcxDeviceNet, [500](#)
- GetCalibrationMax
  - CTcxDeviceNet, [500](#)
- GetCalibrationMin
  - CTcxDeviceNet, [500](#)
- GetCapacityC
  - CRoboDacqNet, [381](#)
- GetCapacityV
  - CRoboDacqNet, [381](#)
- GetCapacityX
  - CRoboDacqNet, [382](#)
- GetCardinalDacqSamplerate
  - CInterfaceboardFunctionNet, [119](#)
- GetCardinalStgOutputrate
  - CInterfaceboardFunctionNet, [119](#)
- GetChannel
  - CSw2to64DeviceNet, [495](#)
- GetChannelDataFillSize
  - CMcsUsbDacqNet, [195](#)
- GetChannelDataI16
  - CCMOSMeaDeviceNet, [46](#)
- GetChannelDataI32
  - CCMOSMeaDeviceNet, [46](#)
- GetChannelDataUI16
  - CCMOSMeaDeviceNet, [46](#)
- GetChannelDataUI32
  - CCMOSMeaDeviceNet, [47](#)
- GetChannelLayout
  - CMcsUsbDacqNet, [195](#)
- GetChannels
  - CMultiBatteryChargerDeviceNet, [299](#)
  - CSw2to64DeviceNet, [496](#)
- GetChannelsCountOfChamber
  - CWarnerUssingFunctionNet, [545](#)
- GetChannelsInBlock
  - CMcsUsbDacqNet, [195](#)
- GetChannelState
  - CMultiBatteryChargerDeviceNet, [299](#)
- GetChargeCapacity
  - CMultiBatteryChargerDeviceNet, [300](#)
- GetChargeCurrent
  - CMultiBatteryChargerDeviceNet, [300](#)
- GetChargingMode
  - CMultiBatteryChargerDeviceNet, [300](#)
- GetChargingPCoefficient
  - CMultiBatteryChargerDeviceNet, [300](#)
- GetChecksumFromFX3Image
  - CMcsUsbFactoryNet, [224](#)
- GetCheckVoltage
  - CokuvisionStimulatorDeviceNet, [323](#)
- GetClampAmpSerialNumber
  - CRoboDacqNet, [382](#)
- GetClampMode
  - CTEERFunctionNet, [514](#)
  - CWarnerUssingFunctionNet, [545](#)
- GetCMOSDataDictionary
  - CCMOSMeaDeviceNet, [47](#)
- GetCoilCommunication
  - CPositionIIDeviceNet, [334](#)
- GetColorRgb
  - CMultiwellOptoStimFunctionNet, [313](#)
- GetColorStr
  - CMultiwellOptoStimFunctionNet, [314](#)
- GetCommand
  - CMcsBusNet, [169](#), [170](#)
  - CPedoterDeviceNet, [328](#)
  - CRoboDacqNet, [382](#)
- GetComplianceVoltageThreshold
  - CWarnerUssingFunctionNet, [545](#)
- GetConfiguration
  - CMcsUsbNet, [247](#)
- GetConfigurationBit
  - CRoboDacqNet, [382](#)
- GetConfigurationBitAxc
  - CRoboDacqNet, [382](#)
- GetConfigurationBitBlu\_Led
  - CRoboDacqNet, [382](#)
- GetConfigurationBitBlu\_LedToggleFast
  - CRoboDacqNet, [382](#)
- GetConfigurationBitBlu\_LedToggleSlow
  - CRoboDacqNet, [382](#)
- GetConfigurationBitCC\_Gen
  - CRoboDacqNet, [382](#)
- GetConfigurationBitCV\_Gen
  - CRoboDacqNet, [382](#)
- GetConfigurationBitRC\_Gen
  - CRoboDacqNet, [382](#)
- GetConfigurationBitRed\_Led
  - CRoboDacqNet, [383](#)
- GetConfigurationBitRed\_LedSaturation
  - CRoboDacqNet, [383](#)
- GetConfigurationBitRed\_LedToggleFast
  - CRoboDacqNet, [383](#)
- GetConfigurationBitRed\_LedToggleSlow
  - CRoboDacqNet, [383](#)
- GetConfigurationBitRelais
  - CRoboDacqNet, [383](#)
- GetConfigurationBitRV\_Gen
  - CRoboDacqNet, [383](#)
- GetConfigurationBits
  - CRoboDacqNet, [383](#)
- GetConfigurationBitStream
  - CRoboDacqNet, [383](#)

- GetConfigurationBitSupply
  - CRoboDacqNet, [383](#)
- GetConnectedDevice
  - CRFFunctionNet, [372](#)
- GetControllerParams
  - CTEERFunctionNet, [514](#)
- GetCrossTalkOffset
  - CRoboDacqNet, [383](#)
- GetCrossTalkOptimum
  - CRoboDacqNet, [383](#)
- GetCurrent
  - CTcxDeviceNet, [500](#)
- GetCurrentAirvalve
  - CRoboDeviceNet, [395](#)
- GetCurrentAirvalveLimit
  - CRoboDeviceNet, [395](#)
- GetCurrentCycle
  - CMeaCoatDeviceNet, [272](#)
- GetCurrentEditTableNumber
  - CWarnerValveControllerDeviceNet, [564](#)
- GetCurrentEnable
  - CTEERFunctionNet, [514](#)
- GetCurrentFactor
  - CokuvisionStimulatorDeviceNet, [323](#)
- GetCurrentNumberOfValves
  - CWarnerValveControllerDeviceNet, [564](#)
- GetCurrentNumberOfValvesEvent
  - CWarnerValveControllerDeviceNet, [580](#)
- GetCurrentPosition
  - CRoboDeviceNet, [395](#)
- GetCurrentPositionI
  - CRoboStatorDeviceNet, [413](#)
- GetCurrentPositionXY
  - CRoboStatorDeviceNet, [413](#)
- GetCurrentPositionZ
  - CRoboStatorDeviceNet, [413](#)
- GetCurrentRangeInNanoAmp
  - CStg200xBasicNet, [443](#)
  - CStimulusFunctionNet, [487](#)
  - CW2100\_StimulatorFunctionNet, [534](#)
- GetCurrentResolutionInNanoAmp
  - CStg200xBasicNet, [443](#)
  - CStimulusFunctionNet, [487](#)
  - CW2100\_StimulatorFunctionNet, [535](#)
- GetCycle
  - CMeaCleanDeviceNet, [268](#)
- GetCycles
  - CMeaCleanDeviceNet, [268](#)
  - CMeaCoatDeviceNet, [272](#)
- GetD
  - CTcxDeviceNet, [500](#)
- GetDacAmplificationFactor
  - CStg200xBasicNet, [444](#)
- GetDacIdleValue
  - CLIH3DeviceNet, [124](#)
- GetDACOffset
  - CGrapheneFunctionNet, [102](#), [103](#)
  - CokuvisionStimulatorDeviceNet, [323](#)
- GetDacOffset
  - CDacCalibrationFunctionNet, [51](#)
  - CLIH3DeviceNet, [124](#)
  - COctoPotDeviceNet, [319](#)
- GetDacPampsPerDigitHighCurrentRange
  - CWarnerUssingFunctionNet, [546](#)
- GetDacPampsPerDigitLowCurrentRange
  - CWarnerUssingFunctionNet, [546](#)
- GetDacqRunStatus
  - CLIH3DeviceNet, [124](#)
- GetDacRange
  - CW2100\_FunctionNet, [528](#)
- GetDACResolution
  - CStg200xBasicNet, [444](#)
  - CStimulusFunctionNet, [487](#)
  - CW2100\_StimulatorFunctionNet, [535](#)
- GetDACs
  - CMcsBus\_SensorNet, [155](#)
- GetDacUseldleValue
  - CLIH3DeviceNet, [124](#)
- GetDacZero
  - CTEERFunctionNet, [514](#)
  - CWarnerUssingFunctionNet, [546](#)
- GetDataFormat
  - CMcsUsbDacqNet, [195](#)
- GetDataMode
  - CMcsUsbDacqNet, [195](#)
- GetDDecp
  - CTcxDeviceNet, [500](#)
- GetDebugData
  - CPositionIIDeviceNet, [334](#)
- GetDestination
  - CMcsUsbFactoryNet, [224](#)
- GetDestinationCode
  - DriverVersionNet, [597](#)
- GetDestinationDisplayLabel
  - CMcsUsbFactoryNet, [224](#)
- GetDestinationName
  - CMcsUsbFactoryNet, [224](#), [225](#)
  - DriverVersionNet, [597](#), [598](#)
- GetDestinationSerialNumber
  - CMcsUsbFactoryNet, [225](#)
- GetDestinationTargetAddress
  - CMcsUsbFactoryNet, [225](#)
- GetDetectionThreshold
  - CMcsBus\_SensorNet, [155](#)
- GetDetectorValue
  - CMcsBus\_SensorNet, [155](#)
- GetDevice
  - CTcxDeviceNet, [500](#)
- GetDeviceCannotStallOutRequests
  - CMcsUsbNet, [247](#)
- GetDeviceCapableSpeed
  - CMcsUsbNet, [247](#)
- GetDeviceEnum
  - CMcsUsbNet, [247](#)
- GetDeviceGroupChannelInfos

- CDacqGroupChannelSelectionTemplateNet< Dac-  
qGroupChannelEnumTemplateNet, Dac-  
qGroupChannelEnumTemplate, CDevice-  
GroupChannelInfoTemplateNet >, 54
- GetDeviceId
  - CMcsUsbNet, 247
- GetDeviceList
  - CPositionImpDeviceNet, 342
- GetDeviceNames
  - CRadioControlledDevicesNet, 365
- GetDeviceRootHubVendorEnum
  - CMcsUsbNet, 247
- GetDeviceRootHubVendorID
  - CMcsUsbNet, 247
- GetDeviceRootHubVendorName
  - CMcsUsbNet, 247
- GetDeviceSpeed
  - CMcsUsbNet, 248
- GetDeviceType
  - CTcxDeviceNet, 501
- GetDevname
  - CTcxDeviceNet, 501
- GetDigin
  - CFluidControlDeviceNet, 76
- GetDiginState
  - CLIH3DeviceNet, 125
- GetDiginValue
  - CStg200xBasicNet, 444
- GetDigitalData
  - CMeaDigitalDataFunctionNet, 285
- GetDigitalIn
  - CPPCFunctionNet, 349
  - CPPS\_FunctionNet, 356
- GetDigitalOutPortValve
  - CWarnerValveControllerDeviceNet, 565
- GetDigitalOutPortValveEvent
  - CWarnerValveControllerDeviceNet, 581
- GetDigitalPortDirection
  - CWarnerValveControllerDeviceNet, 566
- GetDigitalPortDirectionEvent
  - CWarnerValveControllerDeviceNet, 581
- GetDigitalSource
  - CMcsUsbDacqNet, 196, 197
- GetDigitalStimulatorTrigger
  - CW2100\_StimulatorFunctionNet, 535
- GetDigitalStimulatorTriggerSlope
  - CW2100\_StimulatorFunctionNet, 535
- GetDigout
  - CFluidControlDeviceNet, 76
  - CRoboDacqNet, 384
- GetDigoutMode
  - CStg200xBasicNet, 444
- GetDigoutValue
  - CStg200xBasicNet, 445
- GetDIO
  - CMcsBus\_FYIExtensionNet, 132
- GetDischargeCapacity
  - CMultiBatteryChargerDeviceNet, 301
- GetDischargeCurrent
  - CMultiBatteryChargerDeviceNet, 301
- GetDischargeCurrentSetPoint
  - CMultiBatteryChargerDeviceNet, 301
- GetDisplayMode
  - CWarnerValveControllerDeviceNet, 566
- GetDisplayModeEvent
  - CWarnerValveControllerDeviceNet, 581
- GetDisplayText
  - CRoboDacqNet, 384
- GetDMax
  - CTcxDeviceNet, 501
- GetDMin
  - CTcxDeviceNet, 501
- GetDownsampleFactor
  - CRoboDacqNet, 384
- GetDSPHighPassByIndex
  - CIntanMea\_FunctionNet, 115
- GetDuration
  - CMeaCoatDeviceNet, 272
- GetDuty
  - CTcxDeviceNet, 501
- GetEEPromPage
  - CLIH3DeviceNet, 125
- GetElectrodeDacMux
  - CStg200xBasicNet, 445
- GetElectrodeEnable
  - CStg200xBasicNet, 446
- GetElectrodeMode
  - CStg200xBasicNet, 447
- GetEnableAmplifierProtectionSwitch
  - CStg200xBasicNet, 447, 448
- GetEnabledChannelsInGroup
  - CCMOSMea\_FunctionNet, 37
- CDacqGroupChannelSelectionTemplateNet< Dac-  
qGroupChannelEnumTemplateNet, Dac-  
qGroupChannelEnumTemplate, CDevice-  
GroupChannelInfoTemplateNet >, 54
- GetEnableHeaterLimit
  - CTcxDeviceNet, 501
- GetEnableThermocouple
  - CTcxDeviceNet, 501
- GetEntry
  - CMcsUsbListEntryNet, 234
- GetEntryCount
  - CMcsUsbListEntryNet, 235
- GetEnumerationSpeed
  - CMeaDeviceNet, 280
- GetErrorAirpressure
  - CRoboDeviceNet, 396
- GetErrorCurrentAirvalve
  - CRoboDeviceNet, 396
- GetErrorText
  - CMcsUsbNet, 248
- GetErrorVoltage12V
  - CRoboDeviceNet, 396
- GetErrorVoltage5V
  - CRoboDeviceNet, 396



- GetErrorVoltageAirvalve
  - CRoboDeviceNet, [396](#)
- GetErrorVoltageRs485A
  - CRoboDeviceNet, [396](#)
- GetErrorVoltageRs485B
  - CRoboDeviceNet, [396](#)
- GetErrorVoltageValves
  - CRoboDeviceNet, [396](#)
- GetEventData
  - CPositionIIDeviceNet, [335](#)
- GetExternalElectrodeEnable
  - CStg200xBasicNet, [448](#)
- GetFAAmplification
  - CStg200xBasicNet, [449](#)
- GetFilter
  - CRoboDacqNet, [384](#)
- GetFilterAttributes
  - CFilterConfigurationNet, [69](#)
- GetFilterCoeffs
  - CRoboDacqNet, [384](#)
- GetFilterParametersHeadstage
  - CWClassicFunctionNet, [587](#)
- GetFilterProperties
  - CSCUFunctionNet, [425](#)
  - CW2100\_FunctionNet, [528](#)
- GetFilterProperty
  - CMcsUsbDacqNet, [198](#)
  - CSCUFunctionNet, [425](#)
  - CW2100\_FunctionNet, [528](#)
- GetFinalDischargeVoltage
  - CMultiBatteryChargerDeviceNet, [302](#)
- GetFirmwareVersion
  - CMcsUsbNet, [248](#)
- GetFirmwareVersionFromFile
  - CMcsUsbFactoryNet, [225](#)
- GetFirmwareVersionFromHexFile
  - CMcsUsbFactoryNet, [225](#)
- GetFPGA FirmwareType
  - CW2100\_FunctionNet, [528](#)
- GetFrameErrorCounter
  - CTEERFunctionNet, [514](#)
- GetFrequency
  - CRadioControlledDevicesNet, [366](#)
- GetFrequencyRange
  - CPgaDeviceNet, [331](#)
- GetGain
  - CMeaDeviceNet, [280](#)
  - CPgaDeviceNet, [332](#)
- GetGate
  - CCMOSMea\_FunctionNet, [37](#)
- GetGilsonDevice
  - CRoboocyte2DeviceNet, [410](#)
- GetGlobalRepeat
  - CDigOutStimulatorFunctionNet, [61](#)
- GetGNDI
  - CCMOSMea\_FunctionNet, [37](#)
- GetGroupADCBits
  - CCMOSMea\_FunctionNet, [37](#)
- GetGroupChannelBitmaskBySelect
  - CCMOSMea\_FunctionNet, [38](#)
- GetGroupChannelBitmaskHS1NCBathCurrent
  - CCMOSMea\_FunctionNet, [38](#)
- GetGroupChannelBitmaskHS1NCCol2Current
  - CCMOSMea\_FunctionNet, [38](#)
- GetGroupChannelBitmaskHS1NChipTemp
  - CCMOSMea\_FunctionNet, [38](#)
- GetGroupChannelBitmaskHS1Sidebands
  - CCMOSMea\_FunctionNet, [39](#)
- GetGroupChannelBitmaskHS1TriggerStatus
  - CCMOSMea\_FunctionNet, [39](#)
- GetGroupChannelBitmaskIFDigChannels
  - CCMOSMea\_FunctionNet, [39](#)
- GetGroupChannelBitmaskInterfaceADC
  - CCMOSMea\_FunctionNet, [39](#)
- GetGroupChannelBitmaskPacketFrameContext
  - CCMOSMea\_FunctionNet, [40](#)
- GetGroupChannelBitmaskSTG1DACSignal
  - CCMOSMea\_FunctionNet, [40](#)
- GetGroupChannelDataI16
  - CMcsUsbDacqNet, [198](#)
- GetGroupChannelDataI32
  - CMcsUsbDacqNet, [198](#)
- GetGroupChannelDataUI16
  - CMcsUsbDacqNet, [199](#)
- GetGroupChannelDataUI32
  - CMcsUsbDacqNet, [199](#)
- GetGroupDCOffset
  - CCMOSMea\_FunctionNet, [40](#)
- GetGroupID
  - CCMOSMea\_FunctionNet, [40](#)
- CDacqGroupChannelSelectionTemplateNet< Dac-qGroupChannelEnumTemplateNet, Dac-qGroupChannelEnumTemplate, CDevice-GroupChannelInfoTemplateNet >, [54](#), [55](#)
- GetGroupNumberOfChannels
  - CCMOSMea\_FunctionNet, [41](#)
- CDacqGroupChannelSelectionTemplateNet< Dac-qGroupChannelEnumTemplateNet, Dac-qGroupChannelEnumTemplate, CDevice-GroupChannelInfoTemplateNet >, [55](#)
- GetGroupResolutionPerDigit
  - CCMOSMea\_FunctionNet, [41](#)
- GetGroupSampleSize
  - CCMOSMea\_FunctionNet, [41](#)
- CDacqGroupChannelSelectionTemplateNet< Dac-qGroupChannelEnumTemplateNet, Dac-qGroupChannelEnumTemplate, CDevice-GroupChannelInfoTemplateNet >, [55](#)
- GetGroupType
  - CCMOSMea\_FunctionNet, [41](#)
- CDacqGroupChannelSelectionTemplateNet< Dac-qGroupChannelEnumTemplateNet, Dac-qGroupChannelEnumTemplate, CDevice-GroupChannelInfoTemplateNet >, [55](#)
- GetGroupUnit
  - CCMOSMea\_FunctionNet, [41](#), [42](#)

- GetGyroRange
  - CW2100\_FunctionNet, [528](#)
- GetHardwareMaxRange
  - CMcsUsbDacqNet, [200](#)
- GetHardwareMinRange
  - CMcsUsbDacqNet, [200](#)
- GetHardwareRevision
  - CMcsUsbNet, [248](#)
- GetHasChecksum
  - CWClassicFunctionNet, [587](#)
- GetHashCode
  - HeadstageIDTypeObject, [608](#)
- GetHasRedLedHeadstage
  - CWClassicFunctionNet, [587](#)
- GetHasThermocouple
  - CTcxDeviceNet, [501](#)
- GetHeadstage
  - CStg200xBasicNet, [449](#)
- GetHeadstageActive
  - CMcsUsbNet, [249](#)
- GetHeadstageAdcBits
  - CSCUFunctionNet, [425](#)
- GetHeadstageAdcRangeInMicroVolt
  - CSCUFunctionNet, [426](#)
- GetHeadstageDacBits
  - CSCUFunctionNet, [426](#)
- GetHeadstageDacCurrentRangeInMicroAmpere
  - CSCUFunctionNet, [426](#)
- GetHeadstageDacCurrentResolutionInNanoAmpere
  - CSCUFunctionNet, [427](#)
- GetHeadstageDacVoltageRangeInMilliVolt
  - CSCUFunctionNet, [427](#)
- GetHeadstageDacVoltageResolutionInMicroVolt
  - CSCUFunctionNet, [427](#)
- GetHeadstageGainInPer mille
  - CSCUFunctionNet, [428](#)
- GetHeadstageID
  - CMcsUsbNet, [249](#)
  - CSCUFunctionNet, [428](#)
- GetHeadstageNumberOfAnalogChannels
  - CSCUFunctionNet, [428](#)
- GetHeadstageNumberOfStimulationChannels
  - CSCUFunctionNet, [429](#)
- GetHeadstageOnOff
  - CW2100\_FunctionNet, [529](#)
  - CWClassicFunctionNet, [587](#)
- GetHeadstagePowerStateAtStart
  - CSCUFunctionNet, [429](#)
- GetHeadstagePresent
  - CMcsUsbNet, [249](#)
- GetHeadstageSamplerate
  - CSCUFunctionNet, [429](#)
- GetHeadstageSamplingActive
  - CW2100\_FunctionNet, [529](#)
- GetHeadstageSerialNumber
  - CSCUFunctionNet, [430](#)
- GetHeaterLimit
  - CTcxDeviceNet, [502](#)
- GetHeaterTemp
  - CTcxDeviceNet, [502](#)
- GetHighCurrentRange
  - CWarnerUssingFunctionNet, [547](#)
- GetHighpassFilterEnable
  - CFilterConfigurationNet, [69](#)
- GetHWConfig
  - CRoboDeviceNet::RoboMainLowLevelCommands, [613](#)
- GetHWRevision
  - CRoboDeviceNet::RoboMainLowLevelCommands, [614](#)
- GetHWRevisionEeprom
  - CMcsBusNet, [170](#)
- GetI
  - CTcxDeviceNet, [502](#)
- GetIC
  - CRoboDacqNet, [384](#)
- GetIClamp
  - CRoboDacqNet, [384](#)
- GetICoeff
  - CRobo\_FYITemp\_FunctionNet, [376](#)
- GetICOffset
  - CRoboDacqNet, [384](#)
- GetIDecp
  - CTcxDeviceNet, [502](#)
- GetIDent
  - CMcsUsbNet, [249](#)
- GetIdleModeOffset
  - CWarnerUssingFunctionNet, [547](#)
- GetIGain
  - CRoboDacqNet, [384](#)
- GetIMax
  - CTcxDeviceNet, [502](#)
- GetIMin
  - CTcxDeviceNet, [502](#)
- GetImpedanceResult
  - CIntanMea\_FunctionNet, [115](#)
- GetImpedanceTestFrequency
  - CMealImpedanceDeviceNet, [291](#)
- GetImpld
  - CPositionImpDeviceNet, [343](#)
- GetImplantCurrentSetpoint
  - CPositionIIDeviceNet, [335](#)
- GetImplantResult
  - CPositionIIDeviceNet, [336](#)
- GetImplantState
  - CPositionIIDeviceNet, [336](#)
- GetInMovement
  - CRoboDeviceNet, [396](#)
- GetIntanRegister
  - CIntanMea\_FunctionNet, [115](#)
- GetIntBuffer
  - CGenericDevelopDeviceNet, [89](#)
- GetIO
  - CWarnerValveControllerDeviceTesterFunctionNet, [584](#)
- GetIOut



CTcxDeviceNet, 502  
GetIoVoltage  
  CInterfaceboard2FunctionNet, 117  
GetLastAnswer  
  CGilsonDeviceNet, 100  
GetLastUSBError  
  CMcsUsbNet, 250  
GetLatency  
  CMcsBus\_SensorNet, 156  
GetLatencyCounter  
  CMcsBus\_SensorNet, 156  
GetLayoutConfiguration  
  CMEA2100x256FunctionNet, 263  
GetLEDSwitch  
  CMcsBus\_ExtensionNet, 131  
GetLength  
  CRobo\_FYIProgram\_FunctionNet, 374  
GetLiquidResistance  
  CTEERFunctionNet, 515  
  CWarnerUssingFunctionNet, 547  
GetListModelIndexRange  
  CStg200xBasicNet, 449  
GetListModelTriggerSource  
  CStg200xBasicNet, 449  
GetLowCurrentRange  
  CWarnerUssingFunctionNet, 548  
GetLowerFrequencyByIndex  
  CIntanMea\_FunctionNet, 115  
GetMajor  
  DriverVersionNet, 598  
GetMaxChunkSize\_Byte  
  CTEERFunctionNet, 515  
GetMaxCurrent  
  CMeaCoatDeviceNet, 272  
GetMaxDurationHighCurrentInMicroSec  
  CMultiwellOptoStimFunctionNet, 314  
GetMaxDutyCycleHighCurrent  
  CMultiwellOptoStimFunctionNet, 314  
GetMaxHeaterPowerMultiwell  
  CTcxDeviceNet, 503  
GetMaxNoPressure  
  CRoboDeviceNet::RoboMainLowLevelCommands, 614  
GetMaxNoPressureWaitTime  
  CRoboDeviceNet::RoboMainLowLevelCommands, 614  
GetMaxNumberOfHeadstages  
  CSCUFunctionNet, 430  
GetMaxNumOfColumns  
  CCMOSMea\_FunctionNet, 42  
GetMaxP  
  CTcxDeviceNet, 503  
GetMaxpDecp  
  CTcxDeviceNet, 503  
GetMaxpMax  
  CTcxDeviceNet, 503  
GetMaxpMin  
  CTcxDeviceNet, 503  
GetMaxPower  
  COKuvisionStimulatorDeviceNet, 323  
  CRobo\_FYITemp\_FunctionNet, 376  
GetMaxPressureWaitTime  
  CRoboDeviceNet::RoboMainLowLevelCommands, 614  
GetMaxSamplingFrequency  
  CMcsUsbDacqNet, 200  
GetMaxStimulusChannelsPerHeadstage  
  CSCUFunctionNet, 430  
GetMaxVoltage  
  CMeaCleanDeviceNet, 268  
  COKuvisionStimulatorDeviceNet, 323  
GetMCAcceleration  
  CMcsBus\_MotorControlNet, 136  
GetMCAccelerationEeprom  
  CMcsBus\_MotorControlNet, 136  
GetMCAccelerationShortCommand  
  CMcsBus\_MotorControlNet, 137  
GetMCAxisRevisionEeprom  
  CMcsBus\_MotorControlNet, 137  
GetMCBreakCurrent  
  CMcsBus\_MotorControlNet, 137  
GetMCBreakCurrentEeprom  
  CMcsBus\_MotorControlNet, 137  
GetMCCConfig  
  CMcsBus\_MotorControlNet, 137  
GetMCCConfigEeprom  
  CMcsBus\_MotorControlNet, 137  
GetMCCCurrent  
  CMcsBus\_MotorControlNet, 137  
GetMCCCurrentEeprom  
  CMcsBus\_MotorControlNet, 138  
GetMCCCurrentMode  
  CMcsBus\_MotorControlNet, 138  
GetMCCCurrentModeEeprom  
  CMcsBus\_MotorControlNet, 138  
GetMCCCurrentModeShortCommand  
  CMcsBus\_MotorControlNet, 138  
GetMCCCurrentPosition  
  CMcsBus\_MotorControlNet, 138  
GetMCCCurrentShortCommand  
  CMcsBus\_MotorControlNet, 138  
GetMCCurrentSpeed  
  CMcsBus\_MotorControlNet, 138  
GetMCMaxAcceleration  
  CMcsBus\_MotorControlNet, 139  
GetMCMaxAccelerationEeprom  
  CMcsBus\_MotorControlNet, 139  
GetMCMaxCurrent  
  CMcsBus\_MotorControlNet, 139  
GetMCMaxCurrentEeprom  
  CMcsBus\_MotorControlNet, 139  
GetMCMaxSpeed  
  CMcsBus\_MotorControlNet, 139  
GetMCMaxSpeedEeprom  
  CMcsBus\_MotorControlNet, 139  
GetMCMaxTravel

- CMcsBus\_MotorControlNet, 139
- GetMCMaxTravelEeprom
  - CMcsBus\_MotorControlNet, 140
- GetMCMaxTravelShortCommand
  - CMcsBus\_MotorControlNet, 140
- GetMCMovement
  - CMcsBus\_MotorControlNet, 140
- GetMCNewPosition
  - CMcsBus\_MotorControlNet, 140
- GetMCOutputOnOff
  - CMcsBus\_MotorControlNet, 140
- GetMCPhase
  - CMcsBus\_MotorControlNet, 140
- GetMCPhaseOffset
  - CMcsBus\_MotorControlNet, 140
- GetMCReference
  - CMcsBus\_MotorControlNet, 141
- GetMCReferenceCurrent
  - CMcsBus\_MotorControlNet, 141
- GetMCReferenceCurrentEeprom
  - CMcsBus\_MotorControlNet, 141
- GetMCRegulatorGain
  - CMcsBus\_MotorControlNet, 141
- GetMCRegulatorGainEeprom
  - CMcsBus\_MotorControlNet, 141
- GetMcsBus\_Extension
  - CRobocyte2DeviceNet, 410
- GetMCScalingFactor
  - CMcsBus\_MotorControlNet, 141
- GetMCScalingFactorEeprom
  - CMcsBus\_MotorControlNet, 141
- GetMCSpeed
  - CMcsBus\_MotorControlNet, 142
- GetMCSpeedEeprom
  - CMcsBus\_MotorControlNet, 142
- GetMCSpeedShortCommand
  - CMcsBus\_MotorControlNet, 142
- GetMCSpeedUnitEeprom
  - CMcsBus\_MotorControlNet, 142
- GetMCStandbyCurrent
  - CMcsBus\_MotorControlNet, 142
- GetMCStandbyCurrentEeprom
  - CMcsBus\_MotorControlNet, 142
- GetMCStandbyTime
  - CMcsBus\_MotorControlNet, 142
- GetMCStandbyTimeEeprom
  - CMcsBus\_MotorControlNet, 143
- GetMea21UsbPort
  - CMcsUsbNet, 250
- GetMeaLayout
  - CMcsUsbDacqNet, 200
- GetMemoryUsageDAC
  - CStg200xDownloadBasicNet, 470
- GetMemoryUsageSyncout
  - CStg200xDownloadBasicNet, 470
- GetMinimalThreshold
  - CMcsBus\_SensorNet, 156
- GetMinNoPressureWaitTime
  - CRoboDeviceNet::RoboMainLowLevelCommands, 614
- GetMinor
  - DriverVersionNet, 598
- GetMinPressure
  - CRoboDeviceNet, 396
  - CRoboDeviceNet::RoboMainLowLevelCommands, 614
- GetMinPressureWaitTime
  - CRoboDeviceNet::RoboMainLowLevelCommands, 614
- GetMinSamplingFrequencyStepsize
  - CMcsUsbDacqNet, 201
- GetMinVoltage
  - CMeaCleanDeviceNet, 268
- GetModeSelect
  - CPulseGeneratorFunctionNet, 363
- GetModuleCurrent
  - CStg200xDownloadNet, 479
- GetModuleTemp
  - CStg200xDownloadNet, 479
- GetMovementError
  - CRoboDeviceNet, 396
- GetMovePump
  - CMcsBus\_SensorNet, 156
- GetMultiHeadstageMode
  - CW2100\_FunctionNet, 529
- GetMultiplexedDataChannelsInBlock
  - CStimulusFunctionNet, 487
- GetNanoVoltsPerKelvin
  - CMcsBus\_TempSensorNet, 162
- GetNeurochipMemoryData
  - CCMOSMea\_FunctionNet, 42
- GetNeurochipMemorySize
  - CCMOSMea\_FunctionNet, 42
- GetNIC\_MS
  - CRoboDacqNet, 384
- GetNUC\_MS
  - CRoboDacqNet, 384
- GetNumAmplifications
  - CPgaDeviceNet, 332
- GetNumber
  - CMeaSwitchDeviceNet, 293
  - CSw2to64DeviceNet, 496
- GetNumberOfAnalogChannels
  - CStg200xBasicNet, 449
  - CStimulusFunctionNet, 488
  - CW2100\_StimulatorFunctionNet, 535
- GetNumberOfAudioChannels
  - CMeaAudioFunctionNet, 265
- GetNumberOfAvailableChambers
  - CWarnerUssingFunctionNet, 548
- GetNumberOfAvailableSamples
  - CTEERFunctionNet, 515
- GetNumberOfChannels
  - CDigOutStimulatorFunctionNet, 61
- GetNumberOfDataBits
  - CMcsUsbDacqNet, 201

- GetNumberOfDevices
  - CMcsUsbListNet, [238](#)
- GetNumberOfHardwareSlotsForChambers
  - CWarnerUssingFunctionNet, [548](#)
- GetNumberOfHWADCCChannels
  - CMcsUsbDacqNet::CHWInfo, [113](#)
- GetNumberOfHWDACPaths
  - CStg200xBasicNet, [449](#)
- GetNumberOfHWDigitalChannels
  - CMcsUsbDacqNet::CHWInfo, [113](#)
- GetNumberOfStimulationElectrodes
  - CStg200xBasicNet, [449](#)
- GetNumberOfStimulationSourcesPerElectrode
  - CStg200xBasicNet, [450](#)
- GetNumberOfSupportedGroups
  - CCMOSMea\_FunctionNet, [42](#)
  - CDacqGroupChannelSelectionTemplateNet< Dac-  
qGroupChannelEnumTemplateNet, Dac-  
qGroupChannelEnumTemplate, CDevice-  
GroupChannelInfoTemplateNet >, [55](#)
- GetNumberOfSyncoutChannels
  - CStg200xBasicNet, [450](#)
  - CW2100\_StimulatorFunctionNet, [536](#)
- GetNumberOfTriggerInputs
  - CStg200xBasicNet, [450](#)
  - CW2100\_StimulatorFunctionNet, [536](#)
- GetNumConfigurations
  - CMcsUsbNet, [250](#)
- GetNumControlChannels
  - CTcxDeviceNet, [503](#)
- GetNumDestinations
  - CMcsUsbFactoryNet, [225](#)
- GetNumDevices
  - CTcxDeviceNet, [503](#)
- GetNumEntries
  - DriverVersionNet, [599](#)
- GetNumFrequencyRanges
  - CPgaDeviceNet, [332](#)
- GetNumMeasureChannels
  - CTcxDeviceNet, [503](#)
- GetNUV\_MS
  - CRoboDacqNet, [385](#)
- GetOffsetCurrent
  - CMeaCoatDeviceNet, [273](#)
- GetOnOff
  - CPositionIIDeviceNet, [336](#)
  - CTcxDeviceNet, [504](#)
- GetOutputCurrent
  - CMeaCoatDeviceNet, [273](#)
- GetOutputRate
  - CStg200xBasicNet, [450](#)
- GetOutputVoltage
  - CMeaCleanDeviceNet, [268](#)
- GetP
  - CTcxDeviceNet, [504](#)
- GetParameter
  - CRoboDeviceNet::RoboMainLowLevelCommands,  
[614](#)
- GetPattern
  - CMeaSwitchDeviceNet, [293](#)
- GetPatternBool
  - CMeaSwitchDeviceNet, [294](#)
- GetPauseDuration
  - CMeaCoatDeviceNet, [273](#)
- GetPCoeff
  - CRobo\_FYITemp\_FunctionNet, [376](#)
- GetPDecp
  - CTcxDeviceNet, [504](#)
- GetPeriod
  - CPulseGeneratorFunctionNet, [363](#)
- GetPeriod\_us
  - CTEERFunctionNet, [515](#)
- GetPermanentCurrentInMicroAmp
  - CMultiwellOptoStimFunctionNet, [315](#)
- GetPGain
  - CRoboDacqNet, [385](#)
- GetPhases
  - CRoboDeviceNet::RoboMainLowLevelCommands,  
[614](#)
- GetPicFirmwareType
  - CW2100\_FunctionNet, [529](#)
- GetPiezoState
  - CMcsBus\_SensorNet, [156](#)
- GetPlateClampLockState
  - CMultiwellDeviceNet, [308](#)
- GetPlateClampState
  - CMultiwellDeviceNet, [308](#)
- GetPlateClampStateByHeadstage
  - CMultiwellCallbackFunctionNet, [305](#)
- GetPlateClampStateByHeadstageEvent
  - CMultiwellCallbackFunctionNet, [306](#)
- GetPlateMux
  - CMultiwellDeviceNet, [308](#), [309](#)
- GetPlateType
  - CMultiwellDeviceNet, [309](#)
- GetPMax
  - CTcxDeviceNet, [504](#)
- GetPMin
  - CTcxDeviceNet, [504](#)
- GetPoti
  - CMcsUsbDacqNet, [201](#)
- GetPOut
  - CTcxDeviceNet, [504](#)
- GetPowerMuxPlate
  - CMultiwellDeviceNet, [309](#)
- GetPowerStrength
  - CPositionIIDeviceNet, [337](#)
- GetPressure
  - CMcsBus\_SensorNet, [156](#)
- GetPressureOffset
  - CMcsBus\_SensorNet, [157](#)
- GetPressureRange
  - CPPCFunctionNet, [349](#)
- GetPulseform
  - COKuvisionStimulatorDeviceNet, [323](#)
- GetPulseLength

- CPulseGeneratorFunctionNet, 363
- GetPumpCouple
  - CPPS\_FunctionNet, 356
- GetPumpEnableSpeedRatio
  - CPPS\_FunctionNet, 356
- GetPumpFastOnOff
  - CPPS\_FunctionNet, 356
- GetPumpFastSpeed
  - CPPS\_FunctionNet, 356
- GetPumpFunctionSpeeds
  - CPPS\_FunctionNet, 357
- GetPumpManualOnOff
  - CPPS\_FunctionNet, 357
- GetPumpMaxSpeed
  - CPPS\_FunctionNet, 357
- GetPumpModeType
  - CPPCFunctionNet, 349
  - CPPS\_FunctionNet, 357
- GetPumpSpeed
  - CRoboFluidDeviceNet, 407
- GetPumpSpeedRatio
  - CPPS\_FunctionNet, 357
- GetPumpSpeedUnit
  - CPPCFunctionNet, 350
  - CPPS\_FunctionNet, 357
- GetPWM
  - CFluidControlDeviceNet, 76
- GetPwrOut
  - CTcxDeviceNet, 504
- GetPwrSet
  - CTcxDeviceNet, 505
- GetRatedCapacity
  - CMultiBatteryChargerDeviceNet, 302
- GetReady
  - CMealImpedanceDeviceNet, 291
- GetRecordingNumber
  - CRoboDacqNet, 385
- GetReferenceElectrodeMode
  - CSCUFunctionNet, 430
- GetReferenceElectrodeSwitchState
  - CSCUFunctionNet, 431
- GetReferenceTemperature
  - CFluidControlDeviceNet, 76
- GetRegulationTimeouts
  - CMcsBus\_SensorNet, 157
- GetRegulatorFactor
  - CMcsBus\_SensorNet, 157
- GetRegulatorOnOff
  - CMcsBus\_SensorNet, 157
  - CRobo\_FYITemp\_FunctionNet, 376
- GetRegulatorStatus
  - CMcsBus\_SensorNet, 157
- GetRepeats
  - CProgramPressureCurveNet, 361
- GetRes1
  - CTcxDeviceNet, 505
- GetRes2
  - CTcxDeviceNet, 505
- GetResetFilter
  - CWClassicFunctionNet, 588
- GetResistanceC
  - CRoboDacqNet, 385
- GetResistanceV
  - CRoboDacqNet, 385
- GetResolutionPerDigit
  - CMcsUsbDacqNet, 201
- GetResS
  - CTcxDeviceNet, 505
- GetResult
  - CMealImpedanceDeviceNet, 291
- GetResX
  - CTcxDeviceNet, 505
- GetRFConnectionStatus
  - CWClassicFunctionNet, 588
- GetRFFrequency
  - CPositionImpDeviceNet, 343
- GetRFFrequencyHeadstage
  - CWClassicFunctionNet, 588
- GetRFFrequencyReceiver
  - CWClassicFunctionNet, 588
- GetRFPower
  - CWClassicFunctionNet, 588
- GetRoboDacq
  - CRoboocyte2DeviceNet, 410
- GetRoboFluidDevice
  - CEncapsulatorDeviceNet, 64
  - CRoboocyte2DeviceNet, 410
- GetRotaryPositionCode
  - CTEERFunctionNet, 515
- GetRotatePump
  - CMcsBus\_SensorNet, 157
- GetROut
  - CTcxDeviceNet, 505
- GetRTC
  - CokuvisionStimulatorDeviceNet, 324
  - CPositionIIDeviceNet, 337
- GetSampleBufferChunk
  - CTEERFunctionNet, 515
- GetSampleInterval
  - CLIH3DeviceNet, 125
- GetSamplePeriode
  - CMcsBus\_SensorNet, 158
- GetSampleRate
  - CTEERFunctionNet, 516
- GetSamplerate
  - CMcsUsbDacqNet, 201
- GetSampleVoltageBuffer\_uV
  - CTEERFunctionNet, 516
- GetScaleFactorU1
  - CTEERFunctionNet, 516
- GetScaleFactorU2
  - CTEERFunctionNet, 516
- GetScanHeadstagesResult
  - CWClassicFunctionNet, 588
- GetScreen
  - CRoboDacqNet, 385

- GetSearchReferenceFastAccel
  - CRoboDeviceNet::RoboMainLowLevelCommands, [615](#)
- GetSearchReferenceFastSpeed
  - CRoboDeviceNet::RoboMainLowLevelCommands, [615](#)
- GetSearchReferenceFineAccel
  - CRoboDeviceNet::RoboMainLowLevelCommands, [615](#)
- GetSearchReferenceFineSpeed
  - CRoboDeviceNet::RoboMainLowLevelCommands, [615](#)
- GetSearchReferenceMethod
  - CRoboDeviceNet::RoboMainLowLevelCommands, [615](#)
- GetSearchReferenceMoveOut
  - CRoboDeviceNet::RoboMainLowLevelCommands, [615](#)
- GetSearchReferenceOffsetPos
  - CRoboDeviceNet::RoboMainLowLevelCommands, [615](#)
- GetSelectedChannels
  - CW2100\_FunctionNet, [529](#)
- GetSelectedHeadstage
  - CWClassicFunctionNet, [588](#)
- GetSelectedHeadstageState
  - CW2100\_FunctionNet, [529](#)
- GetSensorType
  - CTcxDeviceNet, [505](#)
- GetSerialNumber
  - CMcsUsbNet, [250](#)
  - DriverVersionNet, [599](#)
- GetSerialNumberHeadstage
  - CWClassicFunctionNet, [588](#)
- GetSetpoint
  - CTcxDeviceNet, [505](#)
- GetSetpointDecp
  - CTcxDeviceNet, [506](#)
- GetSetpointMax
  - CTcxDeviceNet, [506](#)
- GetSetpointMin
  - CTcxDeviceNet, [506](#)
- GetShortBuffer
  - CGenericDevelopDeviceNet, [90](#)
- GetSimulation
  - CRoboDacqNet, [385](#)
- GetSingleHeater
  - CMcsBus\_FYIExtensionNet, [132](#)
- GetSingleValve
  - CFluidControlDeviceNet, [77](#)
  - CRoboFluidDeviceNet, [407](#)
- GetSlope
  - CMeaCleanDeviceNet, [269](#)
  - CMeaCoatDeviceNet, [273](#)
- GetSoftwareKey
  - CMcsUsbNet, [250](#)
- GetSoftwareKeyString
  - CMcsUsbNet, [250](#)
- GetSollPressure
  - CMcsBus\_SensorNet, [158](#)
- GetSollTemp
  - CRobo\_FYITemp\_FunctionNet, [376](#)
- GetSourceBulk
  - CCMOSMea\_FunctionNet, [42](#)
- GetSourceDrain
  - CCMOSMea\_FunctionNet, [42](#)
- GetSourceGate
  - CCMOSMea\_FunctionNet, [42](#)
- GetStartTriggerSlope
  - CDigOutStimulatorFunctionNet, [61](#)
- GetState
  - CRFFFunctionNet, [372](#)
  - CRobo\_FYIProgram\_FunctionNet, [375](#)
- GetStateDebugData
  - CPositionIIDeviceNet, [337](#)
- GetStateEventData
  - CPositionIIDeviceNet, [338](#)
- GetStatus
  - CMcsUsbNet, [250](#)
  - DriverVersionNet, [599](#)
- GetStatusOfLastCommand
  - CMcsUsbNet, [251](#)
- GetStgProgramInfo
  - CStg200xBasicNet, [450](#), [451](#)
- GetStgVersionInfo
  - CStg200xBasicNet, [451](#)
- GetStimulationPatternMemory
  - CW2100\_StimulatorFunctionNet, [536](#)
- GetStimulatorStatus
  - COKuvisionStimulatorDeviceNet, [324](#)
- GetStimulusParametersCache
  - CW2100\_FunctionNet, [529](#)
- GetStimulusParametersFromSelectedHS
  - CW2100\_FunctionNet, [529](#)
- GetStimulusSites
  - CCMOSMea\_FunctionNet, [43](#)
- GetStiumlusParameters
  - CW2100\_FunctionNet, [530](#)
- GetStopTriggerSlope
  - CDigOutStimulatorFunctionNet, [62](#)
- GetSubChannel
  - CMcsBus\_MotorControlNet, [143](#)
- GetSupplyVoltage
  - CPPCFunctionNet, [350](#)
  - CPPS\_FunctionNet, [357](#)
- GetSweepCount
  - CStg200xDownloadBasicNet, [471](#)
- GetSync
  - CWarnerValveControllerDeviceTesterFunctionNet, [584](#)
- GetSyncoutMap
  - CStg200xBasicNet, [451](#)
- GetSyncState
  - CMcsBus\_SensorNet, [158](#)
- GetTableName
  - CWarnerValveControllerDeviceNet, [566](#)

- GetTableNamebyIndex
  - CWarnerValveControllerDeviceNet, 566
- GetTableNamebyIndexEvent
  - CWarnerValveControllerDeviceNet, 581
- GetTablepointer
  - CRetinaLedDeviceNet, 368
- GetTemperatur
  - CMcsBus\_TempSensorNet, 162
- GetTestMode
  - CRFFFunctionNet, 372
- GetThermocoupleCalibration
  - CFluidControlDeviceNet, 77
  - CTcxDeviceNet, 506
- GetThermocoupleNanovoltPerKelvin
  - CFluidControlDeviceNet, 77
  - CTcxDeviceNet, 506
- GetThermocoupleReferenceTemp
  - CTcxDeviceNet, 506
- GetThermocoupleTemp
  - CTcxDeviceNet, 506
- GetThermocoupleTempAbs
  - CTcxDeviceNet, 507
- GetThermocoupleTemperature
  - CFluidControlDeviceNet, 78
- GetThermoOffset
  - CMcsBus\_TempSensorNet, 162
- GetThermoTemp
  - CMcsBus\_TempSensorNet, 162
- GetThermoVoltage
  - CMcsBus\_TempSensorNet, 162
- GetTimeInPause
  - CMeaCoatDeviceNet, 273
- GetTimeInPlateau
  - CMeaCoatDeviceNet, 274
- GetTimeResolutionInNanoSeconds
  - CW2100\_StimulatorFunctionNet, 536
- GetTimeSlot
  - CW2100\_StimulatorFunctionNet, 536
- GetTotalMemory
  - CStg200xBasicNet, 452
  - CStimulusFunctionNet, 488
- GetTotalNumberOfDigitalPorts
  - CWarnerValveControllerDeviceNet, 567
- GetTotalNumberOfTables
  - CWarnerValveControllerDeviceNet, 567
- GetTotalNumberOfValves
  - CWarnerValveControllerDeviceNet, 567
- GetTotalTableSize
  - CWarnerValveControllerDeviceNet, 567
- GetTrigger
  - CStg200xDownloadBasicNet, 471
- GetTriggerSource
  - CStg200xBasicNet, 452
- GetU1Offset
  - CWarnerUssingFunctionNet, 548
- GetU1Reference
  - CWarnerUssingFunctionNet, 550
- GetU2Offset
  - CWarnerUssingFunctionNet, 550
- GetU2Reference
  - CWarnerUssingFunctionNet, 550
- GetUByteBuffer
  - CGenericDevelopDeviceNet, 91
- GetUC
  - CRoboDacqNet, 385
- GetUClamp
  - CRoboDacqNet, 385
- GetUCOffset
  - CRoboDacqNet, 385
- GetUIntA
  - CFilterCoefficientsNet, 68
- GetUIntB
  - CFilterCoefficientsNet, 68
- GetUIntBuffer
  - CGenericDevelopDeviceNet, 91
- GetUnit
  - CTcxDeviceNet, 507
- GetUnitDescription
  - CWarnerUssingFunctionNet, 551
- GetUnitExponent
  - CWarnerUssingFunctionNet, 551
- GetUnitName
  - CWarnerUssingFunctionNet, 551
- GetUnitsPerDigit
  - CWarnerUssingFunctionNet, 552
- GetUOut
  - CTcxDeviceNet, 507
- GetUpdateDisplay
  - CRoboDacqNet, 385
- GetUpperFrequencyByIndex
  - CIntanMea\_FunctionNet, 116
- GetUptimeSeconds
  - CTEERFunctionNet, 517
  - CWarnerUssingFunctionNet, 552
- GetUSBDeviceIDFromFX3Image
  - CMcsUsbFactoryNet, 226
- GetUsbListEntries
  - CMcsUsbListNet, 238
- GetUsbListEntry
  - CMcsUsbListNet, 238
  - CMcsUsbNet, 251
- GetUseBubble
  - CPPS\_FunctionNet, 357
- GetUserCodeFromBitFile
  - CMcsUsbFactoryNet, 226
- GetUserCodeFromFlash
  - CMcsUsbFactoryNet, 226
- GetUserDefinedName
  - CW2100\_FunctionNet, 530
- GetUserDefinedNameCache
  - CW2100\_FunctionNet, 530
- GetUserDefinedNameFromSelectedHS
  - CW2100\_FunctionNet, 530
- GetUserParameter
  - CRoboDeviceNet::RoboMainLowLevelCommands, 615, 616



- GetUShortBuffer
  - CGenericDevelopDeviceNet, [92](#)
- GetUV
  - CRoboDacqNet, [386](#)
- GetUOffset
  - CRoboDacqNet, [386](#)
- GetValue
  - CTcxDeviceNet, [507](#)
- GetValueHires
  - CTcxDeviceNet, [507](#)
- GetValve
  - CFluidControlDeviceNet, [78](#)
  - CRoboFluidDeviceNet, [407](#)
- GetValve1
  - CRobo\_FYIProgram\_FunctionNet, [375](#)
- GetValve2
  - CRobo\_FYIProgram\_FunctionNet, [375](#)
- GetValveActive
  - CPPCFunctionNet, [350](#)
  - CWarnerValveControllerDeviceNet, [567](#)
- GetValveActiveEvent
  - CWarnerValveControllerDeviceNet, [581](#)
- GetValveBoardRevision
  - CWarnerValveControllerDeviceNet, [568](#)
- GetValveBoardRevisionEvent
  - CWarnerValveControllerDeviceNet, [581](#)
- GetValveBoardRevisionString
  - CWarnerValveControllerDeviceNet, [568](#)
- GetValveDigitalInPort
  - CWarnerValveControllerDeviceNet, [568](#)
- GetValveDigitalInPortEvent
  - CWarnerValveControllerDeviceNet, [581](#)
- GetValveLedOn
  - CWarnerValveControllerDeviceNet, [568](#)
- GetValveLedOnEvent
  - CWarnerValveControllerDeviceNet, [582](#)
- GetValveManualGroup
  - CWarnerValveControllerDeviceNet, [569](#)
- GetValveManualGroupEvent
  - CWarnerValveControllerDeviceNet, [582](#)
- GetValveManualState
  - CWarnerValveControllerDeviceNet, [569](#)
- GetValveManualStateEvent
  - CWarnerValveControllerDeviceNet, [582](#)
- GetValveMode
  - CWarnerValveControllerDeviceNet, [569](#)
- GetValveModeEvent
  - CWarnerValveControllerDeviceNet, [582](#)
- GetValves
  - CMcsBus\_FYIExtensionNet, [132](#)
- GetValvesActiveMap
  - CWarnerValveControllerDeviceNet, [569](#)
- GetValvesManualStateMap
  - CWarnerValveControllerDeviceNet, [570](#)
- GetValveTableEntry
  - CWarnerValveControllerDeviceNet, [570](#)
- GetVDD3I
  - CCMOSMea\_FunctionNet, [43](#)
- GetVDDI
  - CCMOSMea\_FunctionNet, [43](#)
- GetVdVs
  - CGrapheneFunctionNet, [103](#)
- GetVdVsDAC
  - CGrapheneFunctionNet, [103, 104](#)
- GetVersion
  - CMcsUsbNet, [251](#)
- GetVersionInt
  - DriverVersionNet, [600](#)
- GetVersionString
  - DriverVersionNet, [600](#)
- GetVMMaxNegativeCurrent
  - CMcsBus\_VoltageModeNet, [164](#)
- GetVMMaxNegativeCurrentEeprom
  - CMcsBus\_VoltageModeNet, [164](#)
- GetVMMaxNegativeVoltage
  - CMcsBus\_VoltageModeNet, [164](#)
- GetVMMaxNegativeVoltageEeprom
  - CMcsBus\_VoltageModeNet, [164](#)
- GetVMMaxPositiveCurrent
  - CMcsBus\_VoltageModeNet, [165](#)
- GetVMMaxPositiveCurrentEeprom
  - CMcsBus\_VoltageModeNet, [165](#)
- GetVMMaxPositiveVoltage
  - CMcsBus\_VoltageModeNet, [165](#)
- GetVMMaxPositiveVoltageEeprom
  - CMcsBus\_VoltageModeNet, [165](#)
- GetVMOutputOnOff
  - CMcsBus\_VoltageModeNet, [165](#)
- GetVMVoltage
  - CMcsBus\_VoltageModeNet, [165](#)
- GetVoltage
  - CokuvisionStimulatorDeviceNet, [324](#)
- GetVoltage12V
  - CRoboDeviceNet, [397](#)
- GetVoltage12VLimit
  - CRoboDeviceNet, [397](#)
- GetVoltage5V
  - CRoboDeviceNet, [397](#)
- GetVoltage5VLimit
  - CRoboDeviceNet, [397](#)
- GetVoltageAirvalve
  - CRoboDeviceNet, [397](#)
- GetVoltageAirvalveLimit
  - CRoboDeviceNet, [397](#)
- GetVoltageClampControllerParam\_D
  - CWarnerUssingFunctionNet, [552](#)
- GetVoltageClampControllerParam\_I
  - CWarnerUssingFunctionNet, [553](#)
- GetVoltageClampControllerParam\_P
  - CWarnerUssingFunctionNet, [553](#)
- GetVoltageRange
  - CGrapheneFunctionNet, [104](#)
- GetVoltageRangeIndex
  - CMcsUsbDacqNet, [201](#)
- GetVoltageRangeInMicroVolt
  - CMcsUsbDacqNet, [202](#)

- CStg200xBasicNet, [452](#)
- CStimulusFunctionNet, [488](#)
- CW2100\_StimulatorFunctionNet, [536](#)
- GetVoltageRangeInMilliVolt
  - CMcsUsbDacqNet, [202](#)
- GetVoltageReached
  - CGrapheneFunctionNet, [105](#)
- GetVoltageResolution
  - CGrapheneFunctionNet, [105](#)
- GetVoltageResolutionInMicroVolt
  - CStg200xBasicNet, [452](#)
  - CStimulusFunctionNet, [488](#)
  - CW2100\_StimulatorFunctionNet, [536](#)
- GetVoltageRs485A
  - CRoboDeviceNet, [397](#)
- GetVoltageRs485ALimit
  - CRoboDeviceNet, [397](#)
- GetVoltageRs485B
  - CRoboDeviceNet, [397](#)
- GetVoltageRs485BLimit
  - CRoboDeviceNet, [398](#)
- GetVoltageValves
  - CRoboDeviceNet, [398](#)
- GetVoltageValvesLimit
  - CRoboDeviceNet, [398](#)
- GetVolti
  - CTcxDeviceNet, [507](#)
- GetWaveform
  - CTEERFunctionNet, [517](#)
- GetWaveLengthInNanometer
  - CMultiwellOptoStimFunctionNet, [315](#)
- GetWorkingFrequency
  - CRFFFunctionNet, [373](#)
- GetWPADebugMode
  - CWClassicFunctionNet, [588](#)
- GetWPAType
  - CWClassicFunctionNet, [588](#)
- GetXGain
  - CRoboDacqNet, [386](#)
- GetXilinxFlashOffset
  - CMcsUsbFactoryNet, [226](#)
- GetXilinxFlashReadCommand
  - CMcsUsbFactoryNet, [226](#)
- GND\_SWITCH\_BIT
  - CW2100\_StimulatorFunctionNet, [539](#)
- GroupID
  - CDeviceGroupChannelInfoTemplateNet< Dacq-GroupChannelEnumTemplateNet >, [58](#)
- GroupType
  - CDeviceGroupChannelInfoTemplateNet< Dacq-GroupChannelEnumTemplateNet >, [58](#)
- HasAnalogOut
  - CSCUFunctionNet, [431](#)
- HasGalvanicIsolation
  - CSCUFunctionNet, [431](#)
- HasHSPowerSwitch
  - CSCUFunctionNet, [431](#)
- HasIMU
  - HeadStageIDType, [606](#)
- HasOptoCurrentMeasurement
  - HeadStageIDType, [606](#)
- HasRadioControl
  - CRadioControlledDevicesNet, [366](#)
- HasRef
  - CRoboDeviceNet::RoboMainLowLevelCommands, [616](#)
- HasRefI
  - CRoboStatorDeviceNet, [413](#)
- HasRefXY
  - CRoboStatorDeviceNet, [413](#)
- HasRefZ
  - CRoboStatorDeviceNet, [413](#)
- HasSoftwareKey
  - CMcsUsbNet, [251](#)
- HeadStageIDType, [604](#)
  - CompareTo, [605](#)
  - ElectricalStimulation, [605](#)
  - Entry, [606](#)
  - Equals, [605](#)
  - HasIMU, [606](#)
  - HasOptoCurrentMeasurement, [606](#)
  - HeadStageIDType, [605](#)
  - HeadstageType, [606](#)
  - HeadstageTypeEnum, [605](#)
  - ID, [606](#)
  - MeasuringOnly, [605](#)
  - NumberOfAnalogChannels, [606](#)
  - NumberOfStimulationChannels, [606](#)
  - OpticalStimulation, [605](#)
  - SN, [606](#)
  - StimulusParameters, [606](#)
  - ToString, [606](#)
  - Type, [607](#)
  - TypeValue, [607](#)
  - Unknown, [605](#)
  - UserDefinedName, [607](#)
  - Valid, [607](#)
  - W16IsW14, [607](#)
- HeadstageIDTypeObject, [607](#)
  - \_AdditionalText, [608](#)
  - \_IdType, [608](#)
  - AdditionalText, [608](#)
  - Equals, [608](#)
  - GetHashCode, [608](#)
  - HeadstageIDTypeObject, [607](#)
  - IdType, [608](#)
  - ToString, [608](#)
- HeadStageIDTypeState, [608](#)
  - ControlState, [609](#)
  - DataState, [609](#)
  - IdType, [609](#)
  - State, [609](#)
- HeadstageType
  - HeadStageIDType, [606](#)
- HeadstageTypeEnum
  - HeadStageIDType, [605](#)



- HLADacq
  - CHLADeviceNet, 112
- HWInfo
  - CMcsUsbDacqNet, 202
- HwVersion
  - CMcsUsbListEntryNet, 236
- ID
  - HeadStageIDType, 606
- IdProduct
  - DeviceIdNet, 593
- IdType
  - HeadstageIDTypeObject, 608
  - HeadstageIDTypeState, 609
- IdVendor
  - DeviceIdNet, 593
- IsAnalogOutEnabled
  - CSCUFunctionNet, 432
- IsAutomaticAnalogOut
  - CSCUFunctionNet, 432
- IsBusy
  - CPPCFunctionNet, 350
- IsChamberAvailable
  - CWarnerUssingFunctionNet, 553
- IsChipPowered
  - CCMOSMea\_FunctionNet, 43
- IsConnected
  - CMcsUsbNet, 251
- IsDeviceHighSpeed
  - CMcsUsbNet, 251
- IsDeviceHighSpeedCapable
  - CMcsUsbNet, 252
- IsDeviceTypeOf
  - CMcsUsbListNet, 239
- IsDigitalChannelDedicated
  - CMcsUsbDacqNet::CHWInfo, 114
- IsDigitalOutPortInverted
  - CWarnerValveControllerDeviceNet, 570
- IsDigitalOutPortInvertedEvent
  - CWarnerValveControllerDeviceNet, 582
- IsEqual
  - CFilterCoefficientsNet, 68
- IsExceptionsEnabled
  - CMcsUsbNet, 252
- IsGateFloating
  - CCMOSMea\_FunctionNet, 43
- IsHeadstageAvailable
  - CSCUFunctionNet, 432
- IsHeadstageAvailableEvent
  - CSCUFunctionNet, 435
- IsHighCurrentMode
  - CWarnerUssingFunctionNet, 554
- IsHSPowered
  - CSCUFunctionNet, 432
- IsInDacqLegacyMode
  - CSCUFunctionNet, 433
- IsInternalCalibrationFinished
  - CTEERFunctionNet, 517
  - CWarnerUssingFunctionNet, 554
- IsPlateTypeValid
  - CMultiwellDeviceNet, 310
- IsPulseEnabled
  - CWarnerUssingFunctionNet, 554
- IsPumpMotorOn
  - CRoboFluidDeviceNet, 407
- IsRunning
  - CMeaCleanDeviceNet, 269
  - CMeaCoatDeviceNet, 274
- IsSamplingFinished
  - CTEERFunctionNet, 517
- IsUserTriggerEnabled
  - CLIH3DeviceNet, 125
- IsValveDigitalInInverted
  - CWarnerValveControllerDeviceNet, 570
- IsValveDigitalInInvertedEvent
  - CWarnerValveControllerDeviceNet, 582
- IsValveOpen
  - CWarnerValveControllerDeviceNet, 571
- IsValveOpenEvent
  - CWarnerValveControllerDeviceNet, 582
- IsValveOpenInAnalogMode
  - CWarnerValveControllerDeviceNet, 571
- IsValveOpenInAnalogModeEvent
  - CWarnerValveControllerDeviceNet, 582
- IsValveOpenInDigitalMode
  - CWarnerValveControllerDeviceNet, 571
- IsValveOpenInDigitalModeEvent
  - CWarnerValveControllerDeviceNet, 583
- ListModeSendStart
  - CStg200xBasicNet, 453
- ListModeSendStop
  - CStg200xBasicNet, 453
- ListOfChangedTriggers
  - StgStatusNet, 623
- LoadPressure
  - CPPCFunctionNet, 352
- LoadUserFirmware
  - CMcsUsbFactoryNet, 226, 227
- LoadValveTable
  - CWarnerValveControllerDeviceNet, 572
- LockPlateClamp
  - CMultiwellDeviceNet, 310
- m\_Bottom
  - CCMOSMeaDeviceNet::CRegionOfInterestRect, 367
- m\_Left
  - CCMOSMeaDeviceNet::CRegionOfInterestRect, 367
- m\_pGilsonDevice
  - CGilsonDeviceNet, 100
- m\_pMcsBus\_MotorControlNet
  - CRoboFluidDeviceNet, 408
- m\_pMcsUsb
  - CMcsUsbFunctionNet, 231
- m\_pMcsUsbFunction
  - CMcsUsbFunctionNet, 231

- m\_pRoboFluidDevice
  - CRoboFluidDeviceNet, [408](#)
- m\_Right
  - CCMOSMeaDeviceNet::CRegionOfInterestRect, [367](#)
- m\_Top
  - CCMOSMeaDeviceNet::CRegionOfInterestRect, [367](#)
- Manufacturer
  - CMcsUsbListEntryNet, [236](#)
- MaxBitNumber
  - DigitalSource< digitalsourceenum >, [594](#)
  - DigitalSourceGeneral, [595](#)
- MaxBitNumberStatic
  - DigitalSource< digitalsourceenum >, [594](#)
- Mcs, [22](#)
- Mcs::Usb, [22](#)
  - DEVICE\_NOT\_FOUND, [27](#)
  - enCMosMeaChipType, [27](#)
  - EnSTG200x\_STATUS, [27](#)
  - nMos16LV, [27](#)
  - nMos32LV, [27](#)
  - nMos36LN, [27](#)
  - nMos64LN, [27](#)
  - NOT\_CONNECTED, [27](#)
  - OK, [27](#)
  - OnChannelData, [27](#)
  - OnDeviceArrivalRemoval, [27](#)
  - OnError, [28](#)
  - OnMcsUsbDeviceState, [28](#)
  - OnMcsUsbDeviceStateCallback, [28](#)
  - OnMwPollStatus, [28](#)
  - OnStg200xDataHandler, [28](#)
  - OnStg200xErrorHandler, [28](#)
  - OnStgPollStatus, [28](#)
  - OnUpdateFirmwareProgress, [28](#)
  - OnUpdateFirmwareStatusChange, [29](#)
  - RoboStatusEventDelegate, [29](#)
  - unknown, [27](#)
- McsBus
  - CPPCDeviceNet, [345](#)
  - CPPS\_DeviceNet, [354](#)
  - CRoboDeviceNet, [405](#)
- MCSBUS1
  - FirmwareDestinationNames, [602](#)
- MCSBUS10
  - FirmwareDestinationNames, [602](#)
- MCSBUS11
  - FirmwareDestinationNames, [602](#)
- MCSBUS12
  - FirmwareDestinationNames, [602](#)
- MCSBUS13
  - FirmwareDestinationNames, [603](#)
- MCSBUS2
  - FirmwareDestinationNames, [603](#)
- MCSBUS3
  - FirmwareDestinationNames, [603](#)
- MCSBUS4
  - FirmwareDestinationNames, [603](#)
- MCSBUS5
  - FirmwareDestinationNames, [603](#)
- MCSBUS6
  - FirmwareDestinationNames, [603](#)
- MCSBUS7
  - FirmwareDestinationNames, [603](#)
- MCSBUS8
  - FirmwareDestinationNames, [603](#)
- MCSBUS9
  - FirmwareDestinationNames, [603](#)
- McsBus\_MotorControl
  - CPeristalticPumpDeviceNet, [330](#)
  - CPPCDeviceNet, [345](#)
  - CPPS\_DeviceNet, [354](#)
  - CRoboDeviceNet, [405](#)
  - CRoboFluidDeviceNet, [408](#)
- McsBus\_Sensor
  - CPPCDeviceNet, [345](#)
  - CPPS\_DeviceNet, [354](#)
- McsBus\_VoltageMode
  - CFluidControlDeviceNet, [80](#)
- McsBus\_XY
  - CRoboDeviceNet, [402](#)
- McsBus\_ZI
  - CRoboDeviceNet, [402](#)
- McsUsbDeviceStateEvent
  - CMcsUsbDeviceStatePushFunctionNet, [220](#)
  - CMcsUsbDeviceStatePushNet, [221](#)
- MCU1
  - FirmwareDestinationNames, [603](#)
- MeaAudioFunctionNet
  - CMeaDeviceNet, [283](#)
- MeaDigitalDataFunctionNet
  - CMeaDeviceNet, [283](#)
- MeaFeedbackFunctionNet
  - CMeaDeviceNet, [283](#)
- Measure
  - CPathIdentDeviceNet, [327](#)
- MeasureReservoir
  - CPPCFunctionNet, [352](#)
- MeasuringOnly
  - HeadStageIDType, [605](#)
- MeFunctionNet
  - CMeaDeviceNet, [284](#)
- mkfilter
  - mkfilterNet, [610](#)
  - mkfilter\_coef\_in\_one\_set
    - mkfilterNet, [610](#)
  - mkfilter\_highpass\_coeff
    - mkfilterNet, [610](#)
  - mkfilter\_highpass\_frequency\_from\_coeff
    - mkfilterNet, [610](#)
  - mkfilter\_highpass\_frequency\_from\_k
    - mkfilterNet, [610](#)
  - mkfilter\_highpass\_k
    - mkfilterNet, [610](#)
  - mkfilter\_MCS

- mkfilterNet, [610](#), [611](#)
- mkfilter\_MCS\_k
  - mkfilterNet, [611](#)
- mkfilter\_normalize\_coeffs\_int
  - mkfilterNet, [611](#)
- mkfilter\_normalize\_coeffs\_short
  - mkfilterNet, [611](#)
- mkfilter\_normalize\_scale\_coeffs\_int
  - mkfilterNet, [612](#)
- mkfilter\_scale\_coef\_in\_one\_set
  - mkfilterNet, [612](#)
- mkfilterNet, [609](#)
  - mkfilter, [610](#)
  - mkfilter\_coef\_in\_one\_set, [610](#)
  - mkfilter\_highpass\_coeff, [610](#)
  - mkfilter\_highpass\_frequency\_from\_coeff, [610](#)
  - mkfilter\_highpass\_frequency\_from\_k, [610](#)
  - mkfilter\_highpass\_k, [610](#)
  - mkfilter\_MCS, [610](#), [611](#)
  - mkfilter\_MCS\_k, [611](#)
  - mkfilter\_normalize\_coeffs\_int, [611](#)
  - mkfilter\_normalize\_coeffs\_short, [611](#)
  - mkfilter\_normalize\_scale\_coeffs\_int, [612](#)
  - mkfilter\_scale\_coef\_in\_one\_set, [612](#)
- MoveAbs
  - CRoboDeviceNet, [398](#)
- MoveAbsI
  - CRoboStatorDeviceNet, [413](#), [414](#)
- MoveAbsXY
  - CRoboStatorDeviceNet, [414](#)
- MoveAbsZ
  - CRoboStatorDeviceNet, [414](#)
- MultibootGetCypressImgId
  - CMcsUsbNet, [252](#)
- MultibootGetImgId
  - CMcsUsbNet, [252](#)
- MultibootGetSelectedImage
  - CMcsUsbNet, [252](#)
- MultibootSelectImage
  - CMcsUsbNet, [252](#)
- MwPollStatusEvent
  - CStg200xDownloadNet, [482](#)
- nMos16LV
  - Mcs::Usb, [27](#)
- nMos32LV
  - Mcs::Usb, [27](#)
- nMos36LN
  - Mcs::Usb, [27](#)
- nMos64LN
  - Mcs::Usb, [27](#)
- NOT\_CONNECTED
  - Mcs::Usb, [27](#)
- NumberOfAnalogChannels
  - HeadStageIDType, [606](#)
- NumberOfChannels
  - CDeviceGroupChannelInfoTemplateNet < Dacq-GroupChannelEnumTemplateNet >, [59](#)
- NumberOfStimulationChannels
  - HeadStageIDType, [606](#)
- NumCoefSets
  - CCreateFilterNet, [49](#)
- OK
  - Mcs::Usb, [27](#)
- OnChannelData
  - Mcs::Usb, [27](#)
- OnDeviceArrivalRemoval
  - Mcs::Usb, [27](#)
- OnError
  - Mcs::Usb, [28](#)
- OnGetActiveRunningTableNumber
  - CWarnerValveControllerDeviceNet, [572](#)
- OnGetAnalogThresholdHigh
  - CWarnerValveControllerDeviceNet, [572](#)
- OnGetAnalogThresholdLow
  - CWarnerValveControllerDeviceNet, [572](#)
- OnGetAnalogVoltage
  - CWarnerValveControllerDeviceNet, [572](#)
- OnGetAvailableHeadstages
  - CSCUFunctionNet, [433](#)
- OnGetCurrentNumberOfValves
  - CWarnerValveControllerDeviceNet, [572](#)
- OnGetDigitalOutPortValve
  - CWarnerValveControllerDeviceNet, [572](#)
- OnGetDigitalPortDirection
  - CWarnerValveControllerDeviceNet, [572](#)
- OnGetDisplayMode
  - CWarnerValveControllerDeviceNet, [573](#)
- OnGetPlateClampStateByHeadstage
  - CMultiwellCallbackFunctionNet, [305](#)
- OnGetTableNamebyIndex
  - CWarnerValveControllerDeviceNet, [573](#)
- OnGetValveActive
  - CWarnerValveControllerDeviceNet, [573](#)
- OnGetValveBoardRevision
  - CWarnerValveControllerDeviceNet, [573](#)
- OnGetValveDigitalInPort
  - CWarnerValveControllerDeviceNet, [573](#)
- OnGetValveLedOn
  - CWarnerValveControllerDeviceNet, [573](#)
- OnGetValveManualGroup
  - CWarnerValveControllerDeviceNet, [573](#)
- OnGetValveManualState
  - CWarnerValveControllerDeviceNet, [573](#)
- OnGetValveMode
  - CWarnerValveControllerDeviceNet, [573](#)
- OnIsDigitalOutPortInverted
  - CWarnerValveControllerDeviceNet, [574](#)
- OnIsHeadstageAvailable
  - CSCUFunctionNet, [433](#)
- OnIsValveDigitalInInverted
  - CWarnerValveControllerDeviceNet, [574](#)
- OnIsValveOpen
  - CWarnerValveControllerDeviceNet, [574](#)
- OnIsValveOpenInAnalogMode
  - CWarnerValveControllerDeviceNet, [574](#)
- OnIsValveOpenInDigitalMode

- CWarnerValveControllerDeviceNet, 574
- OnMcsUsbDeviceState
  - Mcs::Usb, 28
- OnMcsUsbDeviceStateCallback
  - Mcs::Usb, 28
- OnMwPollStatus
  - Mcs::Usb, 28
- OnStg200xDataHandler
  - Mcs::Usb, 28
- OnStg200xErrorHandler
  - Mcs::Usb, 28
- OnStgPollStatus
  - Mcs::Usb, 28
- OnTableEntryChanged
  - CWarnerValveControllerDeviceNet, 574
- OnUpdateFirmwareProgress
  - Mcs::Usb, 28
- OnUpdateFirmwareStatusChange
  - Mcs::Usb, 29
- OpenPipe
  - CGenericDevelopDeviceNet, 93
- OpenPlateClamp
  - CMultiwellDeviceNet, 310
- operator=
  - DeviceldNet, 592
- OpticalStimulation
  - HeadStageIDType, 605
- Order
  - CCreateFilterNet, 49
  - CFilterPropertyNet, 74
- PatternListStart
  - COctoPotDeviceNet, 319
- PIC
  - FirmwareDestinationNames, 603
- PIC2
  - FirmwareDestinationNames, 604
- PIC3
  - FirmwareDestinationNames, 604
- PIC4
  - FirmwareDestinationNames, 604
- PollStatusEvent
  - CStimulusFunctionNet, 494
  - CW2100\_StimulatorFunctionNet, 540
- PostCommaA
  - CFilterCoefficientsNet::s\_FilterAttributesNet, 620
- PostCommaB
  - CFilterCoefficientsNet::s\_FilterAttributesNet, 620
- PowerChip
  - CCMOSMea\_FunctionNet, 43
- PowerHS
  - CSCUFunctionNet, 433
- PPCFunction
  - CPPCDeviceNet, 345
- PPS\_Function
  - CPPS\_DeviceNet, 354
- PreCommaA
  - CFilterCoefficientsNet::s\_FilterAttributesNet, 620
- PreCommaB
  - CFilterCoefficientsNet::s\_FilterAttributesNet, 621
- PrepareAndAppendData
  - CStg200xDownloadNet, 479
  - CStimulusFunctionNet, 489
- PrepareAndSendData
  - CStg200xDownloadNet, 480
  - CStimulusFunctionNet, 490
- PrepareChannelData
  - CDigOutStimulatorFunctionNet, 62
- PrepareData
  - CStimulusFunctionNet, 490
  - CW2100\_StimulatorFunctionNet, 538
- PrepareDataSync
  - CW2100\_StimulatorFunctionNet, 538
- Product
  - CMcsUsbListEntryNet, 236
- Program
  - CProgramPressureCurveNet, 361
- PulseGenerator
  - CW2100\_FunctionNet, 532
- PumpOff
  - CRoboFluidDeviceNet, 407
- PumpOn
  - CRoboFluidDeviceNet, 407
- QueryTriggerstatus
  - CStg200xDownloadNet, 481
- RampStart
  - COctoPotDeviceNet, 320
- Read
  - CExternDTesterDeviceNet, 65
- Read2
  - CExternDTesterDeviceNet, 66
- ReadBlockFromFlash
  - CMcsUsbFactoryNet, 227
- ReadBlockFromIFBGlobalEEPROM
  - CMcsUsbFactoryNet, 227
- ReadBlockFromNVMEM
  - CMcsUsbFactoryNet, 227
- ReadClipping
  - CLIH3DeviceNet, 126
- ReadEepromRegisterPreconfig
  - CMcsUsbNet, 253
- ReadPipe
  - CGenericDevelopDeviceNet, 93
- ReadRegister
  - CMcsUsbNet, 253
- ReadRegister32
  - CMcsUsbNet, 253
- ReadRegisterTimeSlot
  - CMcsUsbNet, 253
- Receive
  - CSerialPortNet, 436
- ReceiveString
  - CSerialPortNet, 436, 437
- RemoveSoftwareKey
  - CMcsUsbNet, 254
- RescanHeadstage

- CMcsUsbNet, [254](#)
- ResetAdcOffset
  - COctoPotDeviceNet, [320](#)
- ResetChannelmap
  - CWClassicFunctionNet, [589](#)
- ResetDacOffset
  - COctoPotDeviceNet, [320](#)
- ResetHighpassFilter
  - CFilterConfigurationNet, [70](#)
- ResetPipe
  - CGenericDevelopDeviceNet, [93](#)
- ResetStatus
  - CStg200xDownloadBasicNet, [471](#)
- RFFunction
  - CPositionIIDeviceNet, [341](#)
- RoboDacq
  - CHiClampDeviceNet, [110](#)
- RoboDevice
  - CSafeISDeviceNet, [420](#)
- RoboError\_AnotherMaster
  - CRoboDeviceNet, [402](#)
- RoboError\_Base
  - CRoboDeviceNet, [402](#)
- RoboError\_CannotEscapeEndSwitch
  - CRoboDeviceNet, [402](#)
- RoboError\_CommandAlreadyInProgress
  - CRoboDeviceNet, [402](#)
- RoboError\_CommandNotPossible
  - CRoboDeviceNet, [402](#)
- RoboError\_CommunicationTimeout
  - CRoboDeviceNet, [402](#)
- RoboError\_DacqNotReady
  - CRoboDeviceNet, [402](#)
- RoboError\_DLLMovementTimeout
  - CRoboDeviceNet, [403](#)
- RoboError\_FindReferenceMethod
  - CRoboDeviceNet, [403](#)
- RoboError\_GilsonCommandPending
  - CRoboDeviceNet, [403](#)
- RoboError\_GilsonTimeout
  - CRoboDeviceNet, [403](#)
- RoboError\_GilsonWrondID
  - CRoboDeviceNet, [403](#)
- RoboError\_McsBus\_UnknownCommand
  - CRoboDeviceNet, [403](#)
- RoboError\_NoEndSwitch
  - CRoboDeviceNet, [403](#)
- RoboError\_NoMoreData
  - CRoboDeviceNet, [403](#)
- RoboError\_NoReference
  - CRoboDeviceNet, [403](#)
- RoboError\_NoSpeedOrAcceleration
  - CRoboDeviceNet, [403](#)
- RoboError\_OverPressure
  - CRoboDeviceNet, [404](#)
- RoboError\_ParameterNotAllowed
  - CRoboDeviceNet, [404](#)
- RoboError\_PeristalticTimeout
  - CRoboDeviceNet, [404](#)
- RoboError\_Phase0OutOfRange
  - CRoboDeviceNet, [404](#)
- RoboError\_PollLoopCanceled
  - CRoboDeviceNet, [404](#)
- RoboError\_PollLoopCanceledAndStopMovement
  - CRoboDeviceNet, [404](#)
- RoboError\_Pressure
  - CRoboDeviceNet, [404](#)
- RoboError\_RangeExceeded
  - CRoboDeviceNet, [404](#)
- RoboError\_StateChangeNotPossible
  - CRoboDeviceNet, [404](#)
- RoboError\_Timeout
  - CRoboDeviceNet, [404](#)
- RoboError\_UnknownCommand
  - CRoboDeviceNet, [405](#)
- RoboMainLowLevelCommand
  - CRoboDeviceNet, [405](#)
- RoboMainStatorLowLevelCommand
  - CRoboStatorDeviceNet, [417](#)
- RoboStatusEvent
  - CRoboDeviceNet, [405](#)
- RoboStatusEventDelegate
  - Mcs::Usb, [29](#)
- RunTable
  - CRoboDacqNet, [386](#)
- s\_FilterAttributesNet
  - CFilterCoefficientsNet::s\_FilterAttributesNet, [620](#)
- SampleRate
  - CCreateFilterNet, [50](#)
- Samplerate
  - CMcsUsbDacqNet, [219](#)
- Scale
  - CCreateFilterNet, [50](#)
- ScanForHeadstages
  - CWClassicFunctionNet, [589](#)
- SelectHeadstage
  - CW2100\_FunctionNet, [530](#)
- SelectTimeSlot
  - CW2100\_StimulatorFunctionNet, [538](#)
- Send
  - CSerialPortNet, [437](#)
- SendBuffered
  - CGilsonDeviceNet, [100](#)
- SendChannelData
  - CDigOutStimulatorFunctionNet, [62](#)
  - CStg200xDownloadBasicNet, [473](#)
- SendCommand
  - CLIH3DeviceNet, [126](#)
- SendImmediate
  - CGilsonDeviceNet, [100](#)
- SendImmediateGetResponse
  - CGilsonDeviceNet, [100](#)
- SendMultiplexedData
  - CStimulusFunctionNet, [491](#)
- SendPreparedData
  - CStimulusFunctionNet, [491](#)

- CW2100\_StimulatorFunctionNet, 538
- SendSegmentDefine
  - CStg200xDownloadNet, 481
- SendSegmentSelect
  - CStg200xDownloadNet, 481
- SendSegmentStart
  - CStg200xDownloadNet, 482
- SendStart
  - CStg200xBasicNet, 453
  - CStimulusFunctionNet, 491
  - CW2100\_StimulatorFunctionNet, 538
- SendStartDacq
  - CMcsUsbDacqNet, 202
- SendStartStgAndDacq
  - CMcsUsbDacqNet, 202
- SendStop
  - CStg200xBasicNet, 453
  - CStimulusFunctionNet, 491
  - CW2100\_StimulatorFunctionNet, 539
- SendStopDacq
  - CMcsUsbDacqNet, 203
- SendStopStgAndDacq
  - CMcsUsbDacqNet, 203
- SendStopStgAndDacqWithOptions
  - CMcsUsbDacqNet, 203
- SendSyncData
  - CStg200xDownloadBasicNet, 473
- Sensor
  - CFYIDeviceNet, 81
  - CMeasureTableDeviceNet, 292
  - CPatchServerDeviceNet, 326
- SerialNumber
  - CMcsUsbListEntryNet, 236
  - CMcsUsbNet, 261
- SerialPort
  - CHLADeviceNet, 112
- Set4ADCCatchampAverageShift
  - CMcsBus\_SensorNet, 158
- Set4ADCMode
  - CMcsBus\_SensorNet, 158
- Set4DAC
  - CMcsBus\_SensorNet, 158
- Set\_Values
  - CNF\_GenDeviceNet, 317
  - CPathIdentDeviceNet, 327
- SetAbsMaxCurrentInMicroAmp
  - CMultiwellOptoStimFunctionNet, 315
- SetAccelerationI
  - CRoboStatorDeviceNet, 414
- SetAccelerationNativeI
  - CRoboStatorDeviceNet, 414
- SetAccelerationNativeXY
  - CRoboStatorDeviceNet, 414
- SetAccelerationNativeZ
  - CRoboStatorDeviceNet, 414
- SetAccelerationXY
  - CRoboStatorDeviceNet, 415
- SetAccelerationZ
  - CRoboStatorDeviceNet, 415
- SetAccelGyroDesiredRate
  - CW2100\_FunctionNet, 530
- SetAccelGyroEnabled
  - CW2100\_FunctionNet, 530
- SetAccelRange
  - CW2100\_FunctionNet, 530
- SetActiveRunningTableNumber
  - CWarnerValveControllerDeviceNet, 574
- SetADC
  - CWarnerValveControllerDeviceTesterFunctionNet, 585
- SetAdcChannels
  - CSafeISDeviceNet, 418
- SetADCInputOffset
  - CCMOSMea\_FunctionNet, 43
- SetAdcOffset
  - CLIH3DeviceNet, 126
  - COctoPotDeviceNet, 320
- SetAdcOffsetPermanent
  - CLIH3DeviceNet, 127
- SetAdcSamplePos
  - CSafeISDeviceNet, 418
- SetAirpressureLimit
  - CRoboDeviceNet, 398
- SetAirValve
  - CRoboDeviceNet, 399
- SetAllDigout
  - CRoboDacqNet, 386
- SetAmplificationSwitch
  - COctoPotDeviceNet, 320
- SetAmplitude
  - CChannelTestDeviceNet, 33
- SetAmplitude\_nA
  - CTEERFunctionNet, 517
- SetAnalogOutADCRange
  - CSCUFunctionNet, 433
- SetAnalogOutChannel
  - CW2100\_FunctionNet, 531
- SetAnalogOutChannels
  - CSCUFunctionNet, 434
- SetAnalogOutDACRange
  - CSCUFunctionNet, 434
- SetAnalogOutFilter
  - CW2100\_FunctionNet, 531
- SetAnalogThresholdHigh
  - CWarnerValveControllerDeviceNet, 575
- SetAnalogThresholdLow
  - CWarnerValveControllerDeviceNet, 575
- SetAnalogVoltageRange
  - CPPCFFunctionNet, 352
- SetAnalogVoltages
  - CPPS\_FunctionNet, 357
- SetAttenuation
  - CChannelTestDeviceNet, 33
- SetAudioChannels
  - CMeaAudioFunctionNet, 265, 266
  - CW2100\_FunctionNet, 531



- SetAudioOutDacParameter
  - CLIH3DeviceNet, [127](#)
- SetAutocalibrationDisabled
  - CStg200xBasicNet, [453](#)
- SetAxisConfig
  - CRoboDeviceNet::RoboMainLowLevelCommands, [616](#)
- SetAxisLED
  - CRoboocyte2DeviceNet, [411](#)
- SetAxisParametersEeprom
  - CMcsBus\_AxisParametersNet, [130](#)
- SetBandwidthByIndex
  - CIntanMea\_FunctionNet, [116](#)
- SetBaseFrequency
  - CRFFunctionNet, [373](#)
- SetBaseSamplerate
  - CCMOSMeaDeviceNet, [47](#)
- SetBath
  - CCMOSMea\_FunctionNet, [43](#)
- SetBathclamp
  - COctoPotDeviceNet, [320](#)
- SetBathMode
  - CCMOSMea\_FunctionNet, [43](#)
- SetBlankingEnable
  - CStg200xBasicNet, [454](#), [455](#)
- SetBuffer
  - CGenericDevelopDeviceNet, [94](#)
- SetBufferIndex
  - CTEERFunctionNet, [518](#)
- SetBusAddress
  - CMcsBusNet, [170](#)
- SetBusAddressEeprom
  - CMcsBusNet, [170](#)
- SetByteBuffer
  - CGenericDevelopDeviceNet, [94](#)
- SetCalibration
  - CTcxDeviceNet, [507](#)
- SetCardinalDacqSamplerate
  - CInterfaceboardFunctionNet, [119](#)
- SetCardinalStgOutputrate
  - CInterfaceboardFunctionNet, [120](#)
- SetChannel
  - CSw2to64DeviceNet, [496](#)
- SetChannelmap
  - CWClassicFunctionNet, [589](#)
- SetChannels
  - CSw2to64DeviceNet, [496](#)
- SetChannelSwitch
  - COctoPotDeviceNet, [320](#)
- SetChargingMode
  - CMultiBatteryChargerDeviceNet, [302](#)
- SetChargingPCoefficient
  - CMultiBatteryChargerDeviceNet, [302](#)
- SetCheckVoltage
  - COkuvisionStimulatorDeviceNet, [324](#)
- SetClampMode
  - CTEERFunctionNet, [518](#)
  - CWarnerUssingFunctionNet, [555](#)
- SetColorRgb
  - CMultiwellOptoStimFunctionNet, [315](#)
- SetColorStr
  - CMultiwellOptoStimFunctionNet, [316](#)
- SetCommand
  - CMcsBusNet, [170](#), [171](#)
  - CPedoterDeviceNet, [328](#)
  - CRoboDacqNet, [386](#)
- SetConfiguration
  - CMcsUsbNet, [254](#)
- SetConfigurationBit
  - CRoboDacqNet, [386](#)
- SetConfigurationBitAxc
  - CRoboDacqNet, [386](#)
- SetConfigurationBitBlu\_Led
  - CRoboDacqNet, [386](#)
- SetConfigurationBitBlu\_LedToggleFast
  - CRoboDacqNet, [387](#)
- SetConfigurationBitBlu\_LedToggleSlow
  - CRoboDacqNet, [387](#)
- SetConfigurationBitCC\_Gen
  - CRoboDacqNet, [387](#)
- SetConfigurationBitCV\_Gen
  - CRoboDacqNet, [387](#)
- SetConfigurationBitRC\_Gen
  - CRoboDacqNet, [387](#)
- SetConfigurationBitRed\_Led
  - CRoboDacqNet, [387](#)
- SetConfigurationBitRed\_LedSaturation
  - CRoboDacqNet, [387](#)
- SetConfigurationBitRed\_LedToggleFast
  - CRoboDacqNet, [387](#)
- SetConfigurationBitRed\_LedToggleSlow
  - CRoboDacqNet, [387](#)
- SetConfigurationBitRelais
  - CRoboDacqNet, [387](#)
- SetConfigurationBitRV\_Gen
  - CRoboDacqNet, [388](#)
- SetConfigurationBitStream
  - CRoboDacqNet, [388](#)
- SetConfigurationBitSupply
  - CRoboDacqNet, [388](#)
- SetControllerParams
  - CTEERFunctionNet, [518](#)
- SetCrossTalkOffset
  - CRoboDacqNet, [388](#)
- SetCrossTalkOptimum
  - CRoboDacqNet, [388](#)
- SetCurrentAirvalveLimit
  - CRoboDeviceNet, [399](#)
- SetCurrentAndAir
  - CRoboDeviceNet, [399](#)
- SetCurrentAndAirXY
  - CRoboStatorDeviceNet, [415](#)
- SetCurrentEditTableNumber
  - CWarnerValveControllerDeviceNet, [575](#)
- SetCurrentEnable
  - CTEERFunctionNet, [518](#)

- SetCurrentFactor
  - COkuvisionStimulatorDeviceNet, 324
- SetCurrentMode
  - CStg200xBasicNet, 455
- SetCycles
  - CMeaCleanDeviceNet, 269
  - CMeaCoatDeviceNet, 274
- SetD
  - CTcxDeviceNet, 508
- SetDacAmplificationFactor
  - CStg200xBasicNet, 455
- SetDacAutoControl
  - COctoPotDeviceNet, 320
- SetDacIdleValue
  - CLIH3DeviceNet, 127
- SetDacMode
  - CSafeISDeviceNet, 418
- SetDACOffset
  - CGrapheneFunctionNet, 106
  - COkuvisionStimulatorDeviceNet, 324
- SetDacOffset
  - CDacCalibrationFunctionNet, 51
  - CLIH3DeviceNet, 127
  - COctoPotDeviceNet, 320
- SetDacOffsetPermanent
  - CLIH3DeviceNet, 128
- SetDacPeriode
  - CSafeISDeviceNet, 419
- SetDacPulseform
  - CSafeISDeviceNet, 419
- SetDacqLegacyMode
  - CSCUFunctionNet, 434
- SetDacRange
  - CW2100\_FunctionNet, 531
- SetDACs
  - CMcsBus\_SensorNet, 158
- SetDacUseIdleValue
  - CLIH3DeviceNet, 128
- SetDacValue
  - COctoPotDeviceNet, 321
- SetDataMode
  - CMcsUsbDacqNet, 204
- SetDefault
  - CWarnerValveControllerDeviceNet, 575
- SetDestinationSerialNumber
  - CMcsUsbFactoryNet, 227
- SetDetectionThreshold
  - CMcsBus\_SensorNet, 159
- SetDevice
  - CTcxDeviceNet, 508
- SetDeviceld
  - CUsbDeviceConfigurationFunctionNet, 522
- SetDeviceList
  - CPositionImpDeviceNet, 343
- SetDeviceName
  - CUsbDeviceConfigurationFunctionNet, 522
- SetDeviceType
  - CTcxDeviceNet, 508
- SetDevname
  - CTcxDeviceNet, 508
- SetDiagnosticMode
  - CIntanMea\_FunctionNet, 116
- SetDigitalData
  - CMeaDigitalDataFunctionNet, 285
- SetDigitalOut
  - CMeaDeviceNet, 280
- SetDigitalOutPortInvert
  - CWarnerValveControllerDeviceNet, 575
- SetDigitalOutPortValve
  - CWarnerValveControllerDeviceNet, 576
- SetDigitalPortDirection
  - CWarnerValveControllerDeviceNet, 576
- SetDigitalSource
  - CMcsUsbDacqNet, 204–206
- SetDigitalStimulatorTrigger
  - CW2100\_StimulatorFunctionNet, 539
- SetDigitalStimulatorTriggerSlope
  - CW2100\_StimulatorFunctionNet, 539
- SetDigout
  - CFluidControlDeviceNet, 78
  - CRoboDacqNet, 388
- SetDigoutMode
  - CStg200xBasicNet, 456
- SetDigOutState
  - CLIH3DeviceNet, 128
- SetDigoutValue
  - CStg200xBasicNet, 456
- SetDIO
  - CMcsBus\_FYIExtensionNet, 132
- SetDischargeCurrentSetPoint
  - CMultiBatteryChargerDeviceNet, 303
- SetDisplayMode
  - CWarnerValveControllerDeviceNet, 576
- SetDisplayText
  - CRoboDacqNet, 388
- SetDownsampleFactor
  - CRoboDacqNet, 388
- SetDSPHighPassByIndex
  - CIntanMea\_FunctionNet, 116
- SetDuration
  - CMeaCoatDeviceNet, 274
- SetEEPromPage
  - CLIH3DeviceNet, 128
- SetElectrodeDacMux
  - CStg200xBasicNet, 456–458
- SetElectrodeEnable
  - CStg200xBasicNet, 458–460
- SetElectrodeMode
  - CStg200xBasicNet, 461, 462
- SetEnableAmplifierProtectionSwitch
  - CStg200xBasicNet, 462, 463
- SetEnableHeaterLimit
  - CTcxDeviceNet, 508
- SetEnablePulse
  - CWarnerUssingFunctionNet, 555
- SetEnableThermocouple



- CTcxDeviceNet, [508](#)
- SetExternalElectrodeEnable
  - CStg200xBasicNet, [464](#)
- SetExternalLED
  - CTEERFunctionNet, [519](#)
- SetFAAmplification
  - CStg200xBasicNet, [465](#)
- SetFilter
  - CRoboDacqNet, [388](#)
- SetFilterCoeffs
  - CRoboDacqNet, [389](#)
- SetFilterParameter
  - CFilterConfigurationNet, [70](#)
  - CFilterConfigurationRegisterNet, [71](#), [72](#)
- SetFilterParameterPermanent
  - CFilterConfigurationNet, [70](#)
  - CFilterConfigurationRegisterNet, [72](#)
- SetFilterParametersHeadstage
  - CWClassicFunctionNet, [589](#)
- SetFinalDischargeVoltage
  - CMultiBatteryChargerDeviceNet, [303](#)
- SetFrequency
  - CChannelTestDeviceNet, [33](#)
  - CRadioControlledDevicesNet, [366](#)
- SetGain
  - CPgaDeviceNet, [332](#)
- SetGate
  - CCMOSMea\_FunctionNet, [43](#)
- SetGateFloating
  - CCMOSMea\_FunctionNet, [44](#)
- SetGateToVOP
  - CCMOSMea\_FunctionNet, [44](#)
- SetGlobalRepeat
  - CDigOutStimulatorFunctionNet, [63](#)
- SetGyroRange
  - CW2100\_FunctionNet, [531](#)
- SetHasChecksum
  - CWClassicFunctionNet, [589](#)
- SetHeadstage
  - CStg200xBasicNet, [465](#)
- SetHeadstageOnOff
  - CW2100\_FunctionNet, [531](#)
  - CWClassicFunctionNet, [589](#)
- SetHeadstagePowerStateAtStart
  - CSCUFunctionNet, [434](#)
- SetHeadstageSamplingActive
  - CW2100\_FunctionNet, [531](#)
- SetHeadstageToSleep
  - CW2100\_FunctionNet, [531](#)
- SetHeaterLimit
  - CTcxDeviceNet, [508](#)
- SetHighCurrentMode
  - CWarnerUssingFunctionNet, [555](#)
- SetHighpassFilterEnable
  - CFilterConfigurationNet, [70](#)
- SetHWConfig
  - CRoboDeviceNet::RoboMainLowLevelCommands, [616](#)
- SetHWRevision
  - CRoboDeviceNet::RoboMainLowLevelCommands, [616](#)
- SetHWRevisionEeprom
  - CMcsBusNet, [171](#)
- SetHWSelectedChannels
  - CWClassicFunctionNet, [589](#)
- SetI
  - CTcxDeviceNet, [508](#)
- SetI Clamp
  - CRoboDacqNet, [389](#)
- SetICoeff
  - CRobo\_FYITemp\_FunctionNet, [377](#)
- SetIOffset
  - CRoboDacqNet, [389](#)
- SetIdleModeOffset
  - CWarnerUssingFunctionNet, [556](#)
- SetIGain
  - CRoboDacqNet, [389](#)
- SetImpedanceTestFrequency
  - CMealImpedanceDeviceNet, [291](#)
- SetImpld
  - CPositionImpDeviceNet, [343](#)
- SetImplantCurrentSetpoint
  - CPositionIIDeviceNet, [338](#)
- SetInMovement
  - CRoboDeviceNet, [399](#)
- SetIntanRegister
  - CIntanMea\_FunctionNet, [116](#)
- SetIntBuffer
  - CGenericDevelopDeviceNet, [94](#)
- SetIO
  - CWarnerValveControllerDeviceTesterFunctionNet, [585](#)
- SetIODirection
  - CWarnerValveControllerDeviceTesterFunctionNet, [585](#)
- SetIoVoltage
  - CInterfaceboard2FunctionNet, [118](#)
- SetLatency
  - CMcsBus\_SensorNet, [159](#)
- SetLayoutConfiguration
  - CMEA2100x256FunctionNet, [263](#)
- SetLED
  - CRetinaLedDeviceNet, [368](#)
- SetLEDSwitch
  - CMcsBus\_ExtensionNet, [131](#)
- SetLength
  - CRobo\_FYIProgram\_FunctionNet, [375](#)
- SetLiquidResistance
  - CTEERFunctionNet, [519](#)
  - CWarnerUssingFunctionNet, [556](#)
- SetListmodeIndexRange
  - CStg200xBasicNet, [465](#)
- SetListmodeTriggerSource
  - CStg200xBasicNet, [465](#)
- SetLowCurrentMode
  - CWarnerUssingFunctionNet, [556](#)

- SetLumi
  - CRetinaLedDeviceNet, [368](#)
- SetMaxCurrent
  - CMeaCoatDeviceNet, [275](#)
- SetMaxDurationHighCurrentInMicroSec
  - CMultiwellOptoStimFunctionNet, [316](#)
- SetMaxDutyCycleHighCurrent
  - CMultiwellOptoStimFunctionNet, [316](#)
- SetMaxHeaterPowerMultiwell
  - CTcxDeviceNet, [509](#)
- SetMaxNoPressure
  - CRoboDeviceNet::RoboMainLowLevelCommands, [616](#)
- SetMaxNoPressureWaitTime
  - CRoboDeviceNet::RoboMainLowLevelCommands, [616](#)
- SetMaxP
  - CTcxDeviceNet, [509](#)
- SetMaxPower
  - CokuvisionStimulatorDeviceNet, [325](#)
  - CRobo\_FYITemp\_FunctionNet, [377](#)
- SetMaxPressureWaitTime
  - CRoboDeviceNet::RoboMainLowLevelCommands, [617](#)
- SetMaxVoltage
  - CMeaCleanDeviceNet, [269](#)
  - CokuvisionStimulatorDeviceNet, [325](#)
- SetMCAcceleration
  - CMcsBus\_MotorControlNet, [143](#)
- SetMCAccelerationEeprom
  - CMcsBus\_MotorControlNet, [143](#)
- SetMCAccelerationShortCommand
  - CMcsBus\_MotorControlNet, [143](#)
- SetMCAxisRevisionEeprom
  - CMcsBus\_MotorControlNet, [143](#)
- SetMCBreakCurrent
  - CMcsBus\_MotorControlNet, [143](#)
- SetMCBreakCurrentEeprom
  - CMcsBus\_MotorControlNet, [144](#)
- SetMCConfig
  - CMcsBus\_MotorControlNet, [144](#)
- SetMCConfigEeprom
  - CMcsBus\_MotorControlNet, [144](#)
- SetMCCurrent
  - CMcsBus\_MotorControlNet, [144](#)
- SetMCCurrentEeprom
  - CMcsBus\_MotorControlNet, [144](#)
- SetMCCurrentMode
  - CMcsBus\_MotorControlNet, [144](#)
- SetMCCurrentModeEeprom
  - CMcsBus\_MotorControlNet, [145](#)
- SetMCCurrentModeShortCommand
  - CMcsBus\_MotorControlNet, [145](#)
- SetMCCurrentPosition
  - CMcsBus\_MotorControlNet, [145](#)
- SetMCCurrentShortCommand
  - CMcsBus\_MotorControlNet, [145](#)
- SetMCMaxAcceleration
  - CMcsBus\_MotorControlNet, [145](#)
- SetMCMaxAccelerationEeprom
  - CMcsBus\_MotorControlNet, [145](#)
- SetMCMaxCurrent
  - CMcsBus\_MotorControlNet, [146](#)
- SetMCMaxCurrentEeprom
  - CMcsBus\_MotorControlNet, [146](#)
- SetMCMaxSpeed
  - CMcsBus\_MotorControlNet, [146](#)
- SetMCMaxSpeedEeprom
  - CMcsBus\_MotorControlNet, [146](#)
- SetMCMaxTravel
  - CMcsBus\_MotorControlNet, [146](#)
- SetMCMaxTravelEeprom
  - CMcsBus\_MotorControlNet, [146](#)
- SetMCMaxTravelShortCommand
  - CMcsBus\_MotorControlNet, [147](#)
- SetMCNewPosition
  - CMcsBus\_MotorControlNet, [147](#)
- SetMCOutputOnOff
  - CMcsBus\_MotorControlNet, [147](#)
- SetMCReference
  - CMcsBus\_MotorControlNet, [147](#)
- SetMCReferenceCurrent
  - CMcsBus\_MotorControlNet, [147](#)
- SetMCReferenceCurrentEeprom
  - CMcsBus\_MotorControlNet, [147](#)
- SetMCRegulatorGain
  - CMcsBus\_MotorControlNet, [148](#)
- SetMCRegulatorGainEeprom
  - CMcsBus\_MotorControlNet, [148](#)
- SetMCRotation
  - CMcsBus\_MotorControlNet, [148](#)
- SetMCScalingFactor
  - CMcsBus\_MotorControlNet, [148](#)
- SetMCScalingFactorEeprom
  - CMcsBus\_MotorControlNet, [148](#)
- SetMCSpeed
  - CMcsBus\_MotorControlNet, [148](#)
- SetMCSpeedEeprom
  - CMcsBus\_MotorControlNet, [149](#)
- SetMCSpeedShortCommand
  - CMcsBus\_MotorControlNet, [149](#)
- SetMCSpeedUnitEeprom
  - CMcsBus\_MotorControlNet, [149](#)
- SetMCStandbyCurrent
  - CMcsBus\_MotorControlNet, [149](#)
- SetMCStandbyCurrentEeprom
  - CMcsBus\_MotorControlNet, [149](#)
- SetMCStandbyTime
  - CMcsBus\_MotorControlNet, [149](#)
- SetMCStandbyTimeEeprom
  - CMcsBus\_MotorControlNet, [150](#)
- SetMeasurementMode
  - CStg200xBasicNet, [465](#)
- SetMinimalThreshold
  - CMcsBus\_SensorNet, [159](#)
- SetMinNoPressureWaitTime

- CRoboDeviceNet::RoboMainLowLevelCommands, [617](#)
- SetMinPressure
  - CRoboDeviceNet, [399](#)
  - CRoboDeviceNet::RoboMainLowLevelCommands, [617](#)
- SetMinPressureWaitTime
  - CRoboDeviceNet::RoboMainLowLevelCommands, [617](#)
- SetMinVoltage
  - CMeaCleanDeviceNet, [270](#)
- SetModeSelect
  - CPulseGeneratorFunctionNet, [364](#)
- SetMovePump
  - CMcsBus\_SensorNet, [159](#)
- SetMultiHeadstageMode
  - CW2100\_FunctionNet, [532](#)
- SetNanoVoltsPerKelvin
  - CMcsBus\_TempSensorNet, [162](#)
- SetNeurochipMemoryData
  - CCMOSMea\_FunctionNet, [44](#)
- SetNoFilterCoeffs
  - CRoboDacqNet, [389](#)
- SetNumberOfAnalogChannels
  - CMeaDeviceNet, [280](#)
- SetNumberOfChannels
  - CMeaDeviceNet, [281](#), [282](#)
  - COctoPotDeviceNet, [321](#)
- SetOffsetCurrent
  - CMeaCoatDeviceNet, [275](#)
- SetOnOff
  - CTcxDeviceNet, [509](#)
- SetOutputMap
  - CStg200xDownloadNet, [482](#)
- SetOutputRate
  - COctoPotDeviceNet, [321](#)
  - CStg200xBasicNet, [466](#)
- SetP
  - CTcxDeviceNet, [509](#)
- SetParameter
  - CRoboDeviceNet::RoboMainLowLevelCommands, [617](#)
- SetPattern
  - CMeaSwitchDeviceNet, [294](#)
- SetPatternBool
  - CMeaSwitchDeviceNet, [294](#)
- SetPatternListEntry
  - COctoPotDeviceNet, [321](#)
- SetPauseDuration
  - CMeaCoatDeviceNet, [275](#)
- SetPCoeff
  - CRobo\_FYITemp\_FunctionNet, [377](#)
- SetPeriod
  - CPulseGeneratorFunctionNet, [364](#)
- SetPeriod\_us
  - CTEERFunctionNet, [519](#)
- SetPermanentCurrentInMicroAmp
  - CMultiwellOptoStimFunctionNet, [316](#)
- SetPersistency
  - CRetinaLedDeviceNet, [368](#)
- SetPGain
  - CRoboDacqNet, [389](#)
- SetPidParameter
  - COctoPotDeviceNet, [321](#)
- SetPiezoState
  - CMcsBus\_SensorNet, [159](#)
- SetPlateMux
  - CMultiwellDeviceNet, [310](#), [311](#)
- SetPlateType
  - CMultiwellDeviceNet, [311](#)
- SetPoti
  - CMcsUsbDacqNet, [206](#)
- SetPowerMuxPlate
  - CMultiwellDeviceNet, [311](#)
- SetPowerStrength
  - CPositionIIDeviceNet, [338](#)
- SetPressureOffset
  - CMcsBus\_SensorNet, [159](#)
  - CPPCFunctionNet, [352](#)
- SetPressureRange
  - CPPCFunctionNet, [352](#)
- SetPulse
  - CWarnerUssingFunctionNet, [556](#)
- SetPulseform
  - COkuvisionStimulatorDeviceNet, [325](#)
- SetPulseLength
  - CPulseGeneratorFunctionNet, [364](#)
- SetPumpCouple
  - CPPS\_FunctionNet, [358](#)
- SetPumpEnableSpeedRatio
  - CPPS\_FunctionNet, [358](#)
- SetPumpFastOnOff
  - CPPS\_FunctionNet, [358](#)
- SetPumpFastSpeed
  - CPPS\_FunctionNet, [358](#)
- SetPumpFunctionSpeeds
  - CPPS\_FunctionNet, [358](#)
- SetPumpManualOnOff
  - CPPS\_FunctionNet, [358](#)
- SetPumpMaxSpeed
  - CPPS\_FunctionNet, [358](#)
- SetPumpModeType
  - CPPCFunctionNet, [353](#)
  - CPPS\_FunctionNet, [358](#)
- SetPumpSpeed
  - CRoboFluidDeviceNet, [407](#)
- SetPumpSpeedRatio
  - CPPS\_FunctionNet, [359](#)
- SetPumpSpeedUnit
  - CPPCFunctionNet, [353](#)
  - CPPS\_FunctionNet, [359](#)
- SetPWM
  - CFluidControlDeviceNet, [78](#)
- SetRampParameter
  - COctoPotDeviceNet, [321](#)
- SetRatedCapacity

- CMultiBatteryChargerDeviceNet, [303](#)
- SetRatedCapacityVolatile
  - CMultiBatteryChargerDeviceNet, [303](#)
- SetRecordingNumber
  - CRoboDacqNet, [389](#)
- SetReferenceElectrodeMode
  - CSCUFunctionNet, [435](#)
- SetReferenceElectrodeSwitchState
  - CSCUFunctionNet, [435](#)
- SetRegionOfInterests
  - CCMOSMeaDeviceNet, [47](#)
- SetRegulationTimeouts
  - CMcsBus\_SensorNet, [159](#)
- SetRegulatorFactor
  - CMcsBus\_SensorNet, [160](#)
- SetRegulatorOnOff
  - CMcsBus\_SensorNet, [160](#)
  - CRobo\_FYITemp\_FunctionNet, [377](#)
- SetRepeat
  - CRetinalLedDeviceNet, [369](#)
- SetRepeats
  - CProgramPressureCurveNet, [361](#)
- SetResetFilter
  - CWClassicFunctionNet, [589](#)
- SetRFFrequency
  - CPositionImpDeviceNet, [344](#)
- SetRFFrequencyHeadstage
  - CWClassicFunctionNet, [589](#)
- SetRFFrequencyReceiver
  - CWClassicFunctionNet, [590](#)
- SetRFFrequencyReceiverEeprom
  - CWClassicFunctionNet, [590](#)
- SetRFLostBehaviour
  - CWClassicFunctionNet, [590](#)
- SetRFPower
  - CWClassicFunctionNet, [590](#)
- SetRotatePump
  - CMcsBus\_SensorNet, [160](#)
- SetRTC
  - COkuvisionStimulatorDeviceNet, [325](#)
  - CPositionIIDeviceNet, [338](#)
- SetSampleInterval
  - CLIH3DeviceNet, [129](#)
- SetSamplePeriode
  - CMcsBus\_SensorNet, [160](#)
- SetSamplerate
  - CMcsUsbDacqNet, [206](#)
- SetScreen
  - CRoboDacqNet, [389](#)
- SetSearchReferenceFastAccel
  - CRoboDeviceNet::RoboMainLowLevelCommands, [617](#)
- SetSearchReferenceFastSpeed
  - CRoboDeviceNet::RoboMainLowLevelCommands, [617](#)
- SetSearchReferenceFineAccel
  - CRoboDeviceNet::RoboMainLowLevelCommands, [618](#)
- SetSearchReferenceFineSpeed
  - CRoboDeviceNet::RoboMainLowLevelCommands, [618](#)
- SetSearchReferenceMethod
  - CRoboDeviceNet::RoboMainLowLevelCommands, [618](#)
- SetSearchReferenceMoveOut
  - CRoboDeviceNet::RoboMainLowLevelCommands, [618](#)
- SetSearchReferenceOffsetPos
  - CRoboDeviceNet::RoboMainLowLevelCommands, [618](#)
- SetSelectedChannels
  - CMcsUsbDacqNet, [207–209](#)
  - CW2100\_FunctionNet, [532](#)
- SetSelectedChannelsQueue
  - CMcsUsbDacqNet, [209–211](#)
- SetSelectedData
  - CMcsUsbDacqNet, [211–213](#)
- SetSelectedHeadstage
  - CWClassicFunctionNet, [590](#)
- SetSensorType
  - CTcxDeviceNet, [509](#)
- SetSerialNumberHeadstage
  - CWClassicFunctionNet, [590](#)
- SetSetpoint
  - CTcxDeviceNet, [509](#)
- SetShortBuffer
  - CGenericDevelopDeviceNet, [95](#)
- SetSimulation
  - CRoboDacqNet, [389](#)
- SetSineParameter
  - COctoPotDeviceNet, [321](#)
- SetSingleHeater
  - CMcsBus\_FYIExtensionNet, [133](#)
- SetSingleValve
  - CFluidControlDeviceNet, [79](#)
  - CRoboFluidDeviceNet, [408](#)
- SetSlope
  - CMeaCleanDeviceNet, [270](#)
  - CMeaCoatDeviceNet, [275](#)
- SetSoftwareKey
  - CMcsUsbNet, [254](#)
- SetSoilPressure
  - CMcsBus\_SensorNet, [160](#)
- SetSoilTemp
  - CRobo\_FYITemp\_FunctionNet, [377](#)
- SetSourceBulk
  - CCMOSMea\_FunctionNet, [44](#)
- SetSourceDrain
  - CCMOSMea\_FunctionNet, [44](#)
- SetSourceGate
  - CCMOSMea\_FunctionNet, [44](#)
- SetSpeedI
  - CRoboStatorDeviceNet, [415](#)
- SetSpeedNative
  - CRoboStatorDeviceNet, [415](#)
- SetSpeedNativeXY

- CRoboStatorDeviceNet, 415
- SetSpeedNativeZ
  - CRoboStatorDeviceNet, 415
- SetSpeedXY
  - CRoboStatorDeviceNet, 415
- SetSpeedZ
  - CRoboStatorDeviceNet, 416
- SetStartTriggerSlope
  - CDigOutStimulatorFunctionNet, 63
- SetStateDebugData
  - CPositionIIDeviceNet, 340
- SetStateEventData
  - CPositionIIDeviceNet, 340
- SetStgProgramInfo
  - CStg200xBasicNet, 466
- SetStimulusSites
  - CCMOSMea\_FunctionNet, 44
- SetStopTriggerSlope
  - CDigOutStimulatorFunctionNet, 63
- SetStringFormat
  - CMcsUsbListEntryNet, 235
  - CMcsUsbListNet, 239
- SetSubChannel
  - CMcsBus\_MotorControlNet, 150
- SetSwitches
  - CSafeISDeviceNet, 419
- SetSyncoutMap
  - CStg200xBasicNet, 466
- SetTableName
  - CWarnerValveControllerDeviceNet, 576
- SetTablepointer
  - CRetinalLedDeviceNet, 369
- SetTableStep
  - CWarnerValveControllerDeviceNet, 577
- SetTableStepAll
  - CWarnerValveControllerDeviceNet, 577
- SetTestMode
  - CRFFFunctionNet, 373
- SetThermocoupleNanovoltPerKelvin
  - CFluidControlDeviceNet, 79
  - CTcxDeviceNet, 510
- SetThermoOffset
  - CMcsBus\_TempSensorNet, 162
- SetTransformer
  - CMeFunctionNet, 296
- SetTrigger
  - CRetinalLedDeviceNet, 369
  - CWarnerValveControllerDeviceTesterFunctionNet, 585
- SetTriggerMaskValue
  - CMeaDeviceNet, 282
- SetTriggerPeriod
  - CMeaDeviceNet, 283
- SetTriggerSource
  - CStg200xBasicNet, 466, 467
- SetTriggerSyncDirection
  - CWarnerValveControllerDeviceTesterFunctionNet, 586
- SetUByteBuffer
  - CGenericDevelopDeviceNet, 96
- SetUClamp
  - CRoboDacqNet, 390
- SetUOffset
  - CRoboDacqNet, 390
- SetUIntBuffer
  - CGenericDevelopDeviceNet, 96
- SetupGroupDacqQueue
  - CMcsUsbDacqNet, 213
- SetupRetriggerMode
  - CStg200xDownloadBasicNet, 474
- SetupTrigger
  - CStg200xDownloadBasicNet, 474
  - CStimulusFunctionNet, 492
- SetupTriggerSingle
  - CStg200xDownloadBasicNet, 475
  - CStimulusFunctionNet, 492
- SetUseBubble
  - CPPS\_FunctionNet, 359
- SetUserParameter
  - CRoboDeviceNet::RoboMainLowLevelCommands, 618, 619
- SetUShortBuffer
  - CGenericDevelopDeviceNet, 97
- SetUOffset
  - CRoboDacqNet, 390
- SetValue
  - CGenericDevelopDeviceNet, 97
- SetValve
  - CFluidControlDeviceNet, 79
  - CRoboFluidDeviceNet, 408
- SetValve1
  - CRobo\_FYIProgram\_FunctionNet, 375
- SetValve2
  - CRobo\_FYIProgram\_FunctionNet, 375
- SetValveActive
  - CPPCFunctionNet, 353
  - CWarnerValveControllerDeviceNet, 577
- SetValveDigitalInInvert
  - CWarnerValveControllerDeviceNet, 577
- SetValveDigitalInPort
  - CWarnerValveControllerDeviceNet, 578
- SetValveLedOn
  - CWarnerValveControllerDeviceNet, 578
- SetValveManualGroup
  - CWarnerValveControllerDeviceNet, 578
- SetValveManualState
  - CWarnerValveControllerDeviceNet, 578
- SetValveMode
  - CWarnerValveControllerDeviceNet, 579
- SetValves
  - CMcsBus\_FYIExtensionNet, 133
- SetValvesActiveMap
  - CWarnerValveControllerDeviceNet, 579
- SetValvesManualStateMap
  - CWarnerValveControllerDeviceNet, 579
- SetValveTableEntry

- CWarnerValveControllerDeviceNet, 579
- SetVds
  - CGrapheneFunctionNet, 106
- SetVdVs
  - CGrapheneFunctionNet, 107
- SetVdVsDAC
  - CGrapheneFunctionNet, 107
- SetVelocityI
  - CRoboStatorDeviceNet, 416
- SetVelocityXY
  - CRoboStatorDeviceNet, 416
- SetVelocityZ
  - CRoboStatorDeviceNet, 416
- SetVgs
  - CGrapheneFunctionNet, 108
- SetVMMaxNegativeCurrent
  - CMcsBus\_VoltageModeNet, 165
- SetVMMaxNegativeCurrentEeprom
  - CMcsBus\_VoltageModeNet, 166
- SetVMMaxNegativeVoltage
  - CMcsBus\_VoltageModeNet, 166
- SetVMMaxNegativeVoltageEeprom
  - CMcsBus\_VoltageModeNet, 166
- SetVMMaxPositiveCurrent
  - CMcsBus\_VoltageModeNet, 166
- SetVMMaxPositiveCurrentEeprom
  - CMcsBus\_VoltageModeNet, 166
- SetVMMaxPositiveVoltage
  - CMcsBus\_VoltageModeNet, 166
- SetVMMaxPositiveVoltageEeprom
  - CMcsBus\_VoltageModeNet, 167
- SetVMOOutputOnOff
  - CMcsBus\_VoltageModeNet, 167
- SetVMVoltage
  - CMcsBus\_VoltageModeNet, 167
- SetVoltage12VLimit
  - CRoboDeviceNet, 399
- SetVoltage5VLimit
  - CRoboDeviceNet, 399
- SetVoltageAirvalveLimit
  - CRoboDeviceNet, 400
- SetVoltageClampControllerParam\_D
  - CWarnerUssingFunctionNet, 557
- SetVoltageClampControllerParam\_I
  - CWarnerUssingFunctionNet, 557
- SetVoltageClampControllerParam\_P
  - CWarnerUssingFunctionNet, 557
- SetVoltageMode
  - CStg200xBasicNet, 467
- SetVoltageRange
  - CGrapheneFunctionNet, 108
- SetVoltageRangeByIndex
  - CMcsUsbDacqNet, 213
- SetVoltageRangeInMicroVolt
  - CMcsUsbDacqNet, 214
- SetVoltageResolution
  - CGrapheneFunctionNet, 109
- SetVoltageRs485ALimit
  - CRoboDeviceNet, 400
- SetVoltageRs485BLimit
  - CRoboDeviceNet, 400
- SetVoltageValvesLimit
  - CRoboDeviceNet, 400
- SetWaveform
  - CChannelTestDeviceNet, 34
  - CTEERFunctionNet, 519
- SetWaveLengthInNanometer
  - CMultiwellOptoStimFunctionNet, 317
- SetWorkingFrequency
  - CRFFFunctionNet, 373
- SetWPADebugMode
  - CWClassicFunctionNet, 590
- SetWPAType
  - CWClassicFunctionNet, 590
- SetXGain
  - CRoboDacqNet, 390
- Sideband
  - CStimulusFunctionNet::SidebandData, 622
- SidebandData
  - CStimulusFunctionNet::SidebandData, 621
- SineStart
  - COctoPotDeviceNet, 322
- size
  - DigitalSource< digitalsourceenum >, 594
  - DigitalSourceGeneral, 595
- SN
  - HeadStageIDType, 606
- Source
  - DigitalSource< digitalsourceenum >, 594
  - DigitalSourceGeneral, 595
- Start
  - CMeaCleanDeviceNet, 270
  - CMeaCoatDeviceNet, 275
  - CRobo\_FYIProgram\_FunctionNet, 375
- StartDacq
  - CMcsUsbDacqNet, 214, 215
- StartInternalCalibration
  - CTEERFunctionNet, 520
- StartLoop
  - CMcsUsbDacqNet, 216, 217
- StartMCMovement
  - CMcsBus\_MotorControlNet, 150
- StartMeasurement
  - CMealImpedanceDeviceNet, 291
- StartPoll
  - CStimulusFunctionNet, 494
  - CW2100\_StimulatorFunctionNet, 539
- StartSampling
  - CTEERFunctionNet, 520
- StartSync
  - CMcsBus\_SensorNet, 160
- State
  - HeadStageIDTypeState, 609
- Status
  - CUsbExceptionNet, 524
- Status\_AlreadyConfigured



- CMcsUsbNet, [256](#)
- Status\_BadStartFrame
  - CMcsUsbNet, [257](#)
- Status\_Btstuff
  - CMcsUsbNet, [257](#)
- Status\_BufferOverrun
  - CMcsUsbNet, [257](#)
- Status\_BufferUnderrun
  - CMcsUsbNet, [257](#)
- Status\_Canceled
  - CMcsUsbNet, [257](#)
- Status\_Canceling
  - CMcsUsbNet, [257](#)
- Status\_ConnectedPipes
  - CMcsUsbNet, [257](#)
- Status\_ControlNotOwned
  - CMcsUsbNet, [257](#)
- Status\_Crc
  - CMcsUsbNet, [257](#)
- Status\_DataOverrun
  - CMcsUsbNet, [257](#)
- Status\_DataToggleMismatch
  - CMcsUsbNet, [257](#)
- Status\_DataUnderrun
  - CMcsUsbNet, [258](#)
- Status\_DeviceLocked
  - CMcsUsbNet, [258](#)
- Status\_DeviceNotFound
  - CMcsUsbNet, [258](#)
- Status\_DeviceRemoved
  - CMcsUsbNet, [258](#)
- Status\_DevNotResponding
  - CMcsUsbNet, [258](#)
- Status\_EndpointHalted
  - CMcsUsbNet, [258](#)
- Status\_ErrorBusy
  - CMcsUsbNet, [258](#)
- Status\_ErrorShortTransfer
  - CMcsUsbNet, [258](#)
- Status\_Fifo
  - CMcsUsbNet, [258](#)
- Status\_FrameControlOwned
  - CMcsUsbNet, [258](#)
- Status\_InternalHcError
  - CMcsUsbNet, [258](#)
- Status\_InvalidDeviceHandle
  - CMcsUsbNet, [259](#)
- Status\_InvalidHandle
  - CMcsUsbNet, [259](#)
- Status\_InvalidParameter
  - CMcsUsbNet, [259](#)
- Status\_InvalidPipeHandle
  - CMcsUsbNet, [259](#)
- Status\_InvalidUrbFunction
  - CMcsUsbNet, [259](#)
- Status\_IoPending
  - CMcsUsbNet, [259](#)
- Status\_IoTimeout
  - CMcsUsbNet, [259](#)
- Status\_IsochRequestFailed
  - CMcsUsbNet, [259](#)
- Status\_LastUsbErrorMismatch
  - CMcsUsbNet, [259](#)
- Status\_NoBandwidth
  - CMcsUsbNet, [259](#)
- Status\_NoMemory
  - CMcsUsbNet, [260](#)
- Status\_NoSuchDevice
  - CMcsUsbNet, [260](#)
- Status\_NotAccessed
  - CMcsUsbNet, [260](#)
- Status\_NotSupported
  - CMcsUsbNet, [260](#)
- Status\_PidCheckFailure
  - CMcsUsbNet, [260](#)
- Status\_PipeNotLinked
  - CMcsUsbNet, [260](#)
- Status\_RequestFailed
  - CMcsUsbNet, [260](#)
- Status\_RequestMutexFailed
  - CMcsUsbNet, [260](#)
- Status\_RequestMutexTimeout
  - CMcsUsbNet, [260](#)
- Status\_Stall
  - CMcsUsbNet, [260](#)
- Status\_Unconfigured
  - CMcsUsbNet, [260](#)
- Status\_UnexpectedPid
  - CMcsUsbNet, [261](#)
- Stg200xPollStatusEvent
  - CStg200xDownloadNet, [482](#)
- StgStatusNet, [622](#)
  - FromIntPtr, [622](#)
  - FromPtr, [622](#)
  - ListOfChangedTriggers, [623](#)
  - TiggerStatus, [623](#)
- StillConnected
  - CRadioControlledDevicesNet, [366](#)
- Stimulator
  - CW2100\_FunctionNet, [532](#)
- Stimulus
  - CCMOSMeaDeviceNet, [47](#)
  - CStg200xDownloadBasicNet, [476](#)
- StimulusDeviceDataAndUnrolledData
  - CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, [623](#)
- StimulusFunction
  - CLIH3DeviceNet, [129](#)
- StimulusParameters
  - HeadStageIDType, [606](#)
- Stop
  - CMeaCleanDeviceNet, [270](#)
  - CMeaCoatDeviceNet, [276](#)
- StopDacq
  - CMcsUsbDacqNet, [218](#)
- StopLoop

- CMcsUsbDacqNet, 218
- StopMCMovement
  - CMcsBus\_MotorControlNet, 150
- StopMovement
  - CRoboDeviceNet, 400
- StopMovementI
  - CRoboStatorDeviceNet, 416
- StopMovementXY
  - CRoboStatorDeviceNet, 416
- StopMovementZ
  - CRoboStatorDeviceNet, 416
- StopPlateClamp
  - CMultiwellDeviceNet, 312
- StopPoll
  - CStimulusFunctionNet, 494
  - CW2100\_StimulatorFunctionNet, 539
- StopSampling
  - CTEERFunctionNet, 520
- StopTable
  - CRoboDacqNet, 390
- StoreValveTable
  - CWarnerValveControllerDeviceNet, 580
- SwitchOnOff
  - CPositionIIDeviceNet, 340
- SYNC\_BIT0
  - CW2100\_StimulatorFunctionNet, 540
- SYNC\_BIT1
  - CW2100\_StimulatorFunctionNet, 540
- Table\_Wait
  - CRoboDacqNet, 390
- TableDefBegin
  - CRoboDacqNet, 390
- TableDefEnd
  - CRoboDacqNet, 390
- TableEntryChangedEvent
  - CWarnerValveControllerDeviceNet, 583
- TactSwitchGetState
  - CMcsBus\_SensorNet, 160
- TactSwitchSetDisplay
  - CMcsBus\_SensorNet, 161
- TEERFunctionNet
  - CTEERMachinDeviceNet, 521
- ThrowCUsbExceptionNetOnError
  - CMcsUsbFunctionNet, 231
  - CMcsUsbNet, 254
- TiggerStatus
  - StgStatusNet, 623
- TimeResolutionInNanoSeconds
  - W2100\_StimulusParametersNet, 625
- ToCpp
  - CFilterCoefficientsNet::s\_FilterAttributesNet, 620
- ToString
  - CFilterPropertyNet, 73
  - CMcsUsbListEntryNet, 236
  - HeadStageIDType, 606
  - HeadstageIDTypeObject, 608
- TriggerStatus
  - CMcsUsbDeviceStatePushFunctionNet, 220
- CMcsUsbDeviceStatePushNet, 221
- TxnGetSerialNumber
  - CMcsUsbNet, 254
- TxnSetSerialNumber
  - CMcsUsbNet, 254
- TxnTestMemoryReadAndCheck
  - CMcsUsbNet, 254
- TxnTestMemoryWrite
  - CMcsUsbNet, 255
- Type
  - HeadStageIDType, 607
- TypeValue
  - HeadStageIDType, 607
- Unknown
  - HeadStageIDType, 605
- unknown
  - Mcs::Usb, 27
- UnlockPlateClamp
  - CMultiwellDeviceNet, 312
- UnrolledAmplitude
  - CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, 624
- UnrolledDuration
  - CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, 624
- UnrolledSync
  - CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, 624
- UpdateChannelBlock
  - CCMOSMeaDeviceNet, 47
- UpdateDisplay
  - CRoboDacqNet, 390
- UpdateFirmware
  - CMcsUsbFactoryNet, 227–229
- UpdateTransistorVoltages
  - CCMOSMea\_FunctionNet, 44
- USB
  - FirmwareDestinationNames, 604
- usbSetupPacket\_t, 624
  - bmRequestType, 624
  - bRequest, 624
  - wIndex, 625
  - wLength, 625
  - wValue, 625
- UserDefinedName
  - HeadStageIDType, 607
- Valid
  - HeadStageIDType, 607
- ValidKey
  - CMcsUsbNet, 255
- Voltage
  - BatteryState, 30
- VoltageRangeDisplayStringMilliVolt
  - CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet, 524
- VoltageRangeInMicroVolt



- CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet,  
524
- W2100\_StimulusParametersNet, 625
- VoltageResolutionInMicroVolt
  - W2100\_StimulusParametersNet, 625
- VoltageString
  - BatteryState, 30
- VOPSTimerSetResetTimes
  - CCMOSMea\_FunctionNet, 44, 45
- W16IsW14
  - HeadStageIDType, 607
- W2100\_FunctionNet
  - CMeaDeviceNet, 284
- W2100\_StimulusParametersNet, 625
  - CurrentRangeInNanoAmp, 625
  - CurrentResolutionInNanoAmp, 625
  - DACResolution, 625
  - TimeResolutionInNanoSeconds, 625
  - VoltageRangeInMicroVolt, 625
  - VoltageResolutionInMicroVolt, 625
- WaitForAllChambers
  - CWarnerUssingFunctionNet, 558
- WaitForChamber
  - CWarnerUssingFunctionNet, 558
- WarnerUssingFunction
  - CWarnerUssingDeviceNet, 542
- WClassicFunctionNet
  - CMeaDeviceNet, 284
- wIndex
  - usbSetupPacket\_t, 625
- wLength
  - usbSetupPacket\_t, 625
- WPAError\_ScanningIsPending
  - CMcsUsbNet, 261
- Write
  - CExternDTesterDeviceNet, 66
- Write2
  - CExternDTesterDeviceNet, 66
- WriteEepromRegisterPreconfig
  - CMcsUsbNet, 255
- WritePipe
  - CGenericDevelopDeviceNet, 98
- WriteRegister
  - CMcsUsbNet, 255, 256
- WriteRegister32
  - CMcsUsbNet, 256
- WriteRegisterArray
  - CMcsUsbNet, 256
- WriteRegisterTimeSlot
  - CMcsUsbNet, 256
- WriteRegisterValue
  - CMcsUsbNet, 256
- wValue
  - usbSetupPacket\_t, 625