

multichannel* systems

McsUsbNet.dll

Version 5.1.36

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
www.multichannelsystems.com

Generated by Doxygen 1.9.1

1 McsUsbNet.dll for MCS USB devices	1
1.1 Introduction	1
1.2 System requirements	2
1.3 Connecting to an MCS device	2
2 Device Classes	2
2.1 The MCS FluidControl Device	2
2.1.1 Introduction	2
2.1.2 Access to the FluidControl device	3
2.2 MCS-USB-Sw2to64 device	3
3 Function Classes	4
4 Data ACQuisition (DACQ) Devices	5
5 The MCS Robo Device	5
5.1 Introduction	5
6 STG200x & STG400x STimulus Generator	6
6.1 Introduction	6
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
7 Namespace Index	11
7.1 Namespace List	11
8 Hierarchical Index	11
8.1 Class Hierarchy	11
9 Class Index	17
9.1 Class List	17
10 Namespace Documentation	23
10.1 Mcs Namespace Reference	23
10.2 Mcs::Usb Namespace Reference	23
10.2.1 Enumeration Type Documentation	52
10.2.2 Function Documentation	93
11 Class Documentation	95
11.1 CW2100_FunctionNet::AudioChannelsNet Struct Reference	95
11.1.1 Member Data Documentation	95
11.2 BatteryState Class Reference	95
11.2.1 Property Documentation	95
11.3 Bessel3dBFilterBandPassNet Class Reference	96

11.3.1 Constructor & Destructor Documentation	96
11.4 Bessel3dBFilterHighPassNet Class Reference	96
11.4.1 Constructor & Destructor Documentation	97
11.5 Bessel3dBFilterLowPassNet Class Reference	97
11.5.1 Constructor & Destructor Documentation	97
11.6 BesselConstDelayFilterBandPassNet Class Reference	98
11.6.1 Constructor & Destructor Documentation	98
11.7 BesselConstDelayFilterHighPassNet Class Reference	98
11.7.1 Constructor & Destructor Documentation	98
11.8 BesselConstDelayFilterLowPassNet Class Reference	99
11.8.1 Constructor & Destructor Documentation	99
11.9 BesselLegacyFilterHighPassNet Class Reference	99
11.9.1 Constructor & Destructor Documentation	100
11.10 BesselLegacyFilterLowPassNet Class Reference	100
11.10.1 Constructor & Destructor Documentation	100
11.11 ButterworthFilterBandPassNet Class Reference	100
11.11.1 Constructor & Destructor Documentation	101
11.12 ButterworthFilterHighPassNet Class Reference	101
11.12.1 Constructor & Destructor Documentation	101
11.13 ButterworthFilterLowPassNet Class Reference	102
11.13.1 Constructor & Destructor Documentation	102
11.14 CChannelTestDeviceNet Class Reference	102
11.14.1 Constructor & Destructor Documentation	103
11.14.2 Member Function Documentation	103
11.15 CCMOSMea_FunctionNet Class Reference	103
11.15.1 Constructor & Destructor Documentation	105
11.15.2 Member Function Documentation	106
11.16 CCMOSMeaDeviceNet Class Reference	114
11.16.1 Constructor & Destructor Documentation	115
11.16.2 Member Function Documentation	115
11.16.3 Property Documentation	117
11.17 CCreateFilterNet Class Reference	117
11.17.1 Constructor & Destructor Documentation	119
11.17.2 Member Function Documentation	119
11.17.3 Property Documentation	119
11.18 CDacCalibrationFunctionNet Class Reference	120
11.18.1 Detailed Description	121
11.18.2 Constructor & Destructor Documentation	121
11.18.3 Member Function Documentation	121
11.19 CDacqGroupChannelGenericSelectionNet Class Reference	122
11.19.1 Constructor & Destructor Documentation	122
11.20 CDacqGroupChannelSelectionNet Class Reference	123

11.20.1 Constructor & Destructor Documentation	123
11.21 CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet > Class Template Reference	123
11.21.1 Constructor & Destructor Documentation	124
11.21.2 Member Function Documentation	124
11.22 CDeviceGroupChannelInfoGenericNet Class Reference	126
11.22.1 Constructor & Destructor Documentation	126
11.23 CDeviceGroupChannelInfoMEA2100_256Net Class Reference	126
11.23.1 Constructor & Destructor Documentation	127
11.24 CDeviceGroupChannelInfoNet Class Reference	127
11.24.1 Constructor & Destructor Documentation	127
11.25 CDeviceGroupChannelInfoSCUNet Class Reference	128
11.25.1 Constructor & Destructor Documentation	128
11.26 CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumTemplateNet > Class Template Reference	128
11.26.1 Constructor & Destructor Documentation	128
11.26.2 Member Data Documentation	129
11.27 CDeviceGroupChannelInfoW2100Net Class Reference	129
11.27.1 Constructor & Destructor Documentation	129
11.28 CDigOutStimulatorFunctionNet Class Reference	130
11.28.1 Detailed Description	130
11.28.2 Constructor & Destructor Documentation	131
11.28.3 Member Function Documentation	131
11.29 CEncapsulatorDeviceNet Class Reference	134
11.29.1 Detailed Description	134
11.29.2 Constructor & Destructor Documentation	134
11.29.3 Member Function Documentation	135
11.30 CExternDTesterDeviceNet Class Reference	135
11.30.1 Detailed Description	135
11.30.2 Constructor & Destructor Documentation	135
11.30.3 Member Function Documentation	136
11.31 CFilterCoefficientsNet Class Reference	137
11.31.1 Constructor & Destructor Documentation	137
11.31.2 Member Function Documentation	138
11.31.3 Property Documentation	139
11.32 CFilterConfigurationNet Class Reference	139
11.32.1 Constructor & Destructor Documentation	139
11.32.2 Member Function Documentation	140
11.33 CFilterConfigurationRegisterNet Class Reference	141
11.33.1 Constructor & Destructor Documentation	141
11.33.2 Member Function Documentation	142
11.34 CFilterPropertyNet Class Reference	143
11.34.1 Constructor & Destructor Documentation	143

11.34.2 Member Function Documentation	143
11.34.3 Property Documentation	144
11.35 CFluidControlDeviceNet Class Reference	144
11.35.1 Detailed Description	145
11.35.2 Constructor & Destructor Documentation	146
11.35.3 Member Function Documentation	146
11.35.4 Property Documentation	151
11.36 CFYIDeviceNet Class Reference	151
11.36.1 Detailed Description	152
11.36.2 Constructor & Destructor Documentation	152
11.36.3 Property Documentation	152
11.37 CGenericDevelopDeviceNet Class Reference	153
11.37.1 Detailed Description	160
11.37.2 Constructor & Destructor Documentation	160
11.37.3 Member Function Documentation	160
11.38 CGilsonDeviceNet Class Reference	171
11.38.1 Detailed Description	171
11.38.2 Constructor & Destructor Documentation	171
11.38.3 Member Function Documentation	172
11.38.4 Member Data Documentation	172
11.39 CGrapheneASICDeviceNet Class Reference	173
11.39.1 Constructor & Destructor Documentation	173
11.39.2 Member Function Documentation	173
11.40 CGrapheneFunctionNet Class Reference	174
11.40.1 Detailed Description	176
11.40.2 Constructor & Destructor Documentation	176
11.40.3 Member Function Documentation	177
11.41 CMcsUsbDacqNet::ChannelBlockNet Class Reference	187
11.41.1 Constructor & Destructor Documentation	189
11.41.2 Member Function Documentation	189
11.42 CHiClampDeviceNet Class Reference	211
11.42.1 Detailed Description	212
11.42.2 Constructor & Destructor Documentation	212
11.42.3 Property Documentation	212
11.43 CHLADacqNet Class Reference	212
11.43.1 Constructor & Destructor Documentation	213
11.44 CHLADeviceNet Class Reference	213
11.44.1 Detailed Description	213
11.44.2 Constructor & Destructor Documentation	213
11.44.3 Property Documentation	214
11.45 CMcsUsbDacqNet::CHWInfo Class Reference	214
11.45.1 Detailed Description	214

11.45.2 Constructor & Destructor Documentation	214
11.45.3 Member Function Documentation	215
11.46 CIntanMea_FunctionNet Class Reference	217
11.46.1 Constructor & Destructor Documentation	217
11.46.2 Member Function Documentation	218
11.47 CInterfaceboard2FunctionNet Class Reference	219
11.47.1 Detailed Description	219
11.47.2 Constructor & Destructor Documentation	220
11.47.3 Member Function Documentation	220
11.48 CInterfaceboardFunctionNet Class Reference	221
11.48.1 Detailed Description	221
11.48.2 Constructor & Destructor Documentation	221
11.48.3 Member Function Documentation	222
11.49 CLIH3DeviceNet Class Reference	223
11.49.1 Detailed Description	225
11.49.2 Constructor & Destructor Documentation	225
11.49.3 Member Function Documentation	225
11.49.4 Property Documentation	233
11.50 CMcsBus_AxisParametersNet Class Reference	233
11.50.1 Constructor & Destructor Documentation	233
11.50.2 Member Function Documentation	234
11.51 CMcsBus_ExtensionNet Class Reference	234
11.51.1 Constructor & Destructor Documentation	235
11.51.2 Member Function Documentation	235
11.52 CMcsBus_FYIExtensionNet Class Reference	235
11.52.1 Constructor & Destructor Documentation	236
11.52.2 Member Function Documentation	236
11.53 CMcsBus_MotorControlNet Class Reference	237
11.53.1 Constructor & Destructor Documentation	240
11.53.2 Member Function Documentation	240
11.54 CMcsBus_SensorNet Class Reference	254
11.54.1 Constructor & Destructor Documentation	256
11.54.2 Member Function Documentation	256
11.55 CMcsBus_TempSensorNet Class Reference	265
11.55.1 Constructor & Destructor Documentation	265
11.55.2 Member Function Documentation	266
11.56 CMcsBus_VoltageModeNet Class Reference	267
11.56.1 Constructor & Destructor Documentation	268
11.56.2 Member Function Documentation	268
11.57 CMcsBusNet Class Reference	271
11.57.1 Constructor & Destructor Documentation	272
11.57.2 Member Function Documentation	272

11.58 CMcsUsbDacqNet Class Reference	275
11.58.1 Detailed Description	279
11.58.2 Constructor & Destructor Documentation	279
11.58.3 Member Function Documentation	279
11.58.4 Member Data Documentation	297
11.58.5 Property Documentation	297
11.58.6 Event Documentation	297
11.59 CMcsUsbDeviceStatePushFunctionNet Class Reference	298
11.59.1 Constructor & Destructor Documentation	298
11.59.2 Member Function Documentation	298
11.59.3 Event Documentation	299
11.60 CMcsUsbDeviceStatePushNet Class Reference	299
11.60.1 Constructor & Destructor Documentation	299
11.60.2 Member Function Documentation	300
11.60.3 Event Documentation	300
11.61 CMcsUsbFactoryNet Class Reference	300
11.61.1 Constructor & Destructor Documentation	301
11.61.2 Member Function Documentation	302
11.61.3 Member Data Documentation	307
11.62 CMcsUsbFunctionNet Class Reference	308
11.62.1 Constructor & Destructor Documentation	309
11.62.2 Member Function Documentation	309
11.62.3 Member Data Documentation	310
11.63 CMcsUsbFunctionPointerContainer Class Reference	310
11.64 CMcsUsbListEntryNet Class Reference	310
11.64.1 Detailed Description	311
11.64.2 Constructor & Destructor Documentation	311
11.64.3 Member Function Documentation	311
11.64.4 Property Documentation	313
11.65 CMcsUsbListNet Class Reference	314
11.65.1 Detailed Description	315
11.65.2 Constructor & Destructor Documentation	315
11.65.3 Member Function Documentation	315
11.65.4 Property Documentation	316
11.65.5 Event Documentation	316
11.66 CMcsUsbNet Class Reference	317
11.66.1 Detailed Description	321
11.66.2 Constructor & Destructor Documentation	321
11.66.3 Member Function Documentation	321
11.66.4 Member Data Documentation	333
11.66.5 Property Documentation	338
11.67 CMcsUsbPointerContainer Class Reference	338

11.68 CMEA2100_256DacqGroupChannelSelectionNet Class Reference	338
11.68.1 Constructor & Destructor Documentation	338
11.69 CMEA2100x256FunctionNet Class Reference	339
11.69.1 Detailed Description	339
11.69.2 Constructor & Destructor Documentation	339
11.69.3 Member Function Documentation	340
11.70 CMeaAudioFunctionNet Class Reference	340
11.70.1 Constructor & Destructor Documentation	341
11.70.2 Member Function Documentation	341
11.71 CMeaCleanDeviceNet Class Reference	343
11.71.1 Detailed Description	344
11.71.2 Constructor & Destructor Documentation	344
11.71.3 Member Function Documentation	345
11.72 CMeaCoatDeviceNet Class Reference	347
11.72.1 Detailed Description	348
11.72.2 Constructor & Destructor Documentation	349
11.72.3 Member Function Documentation	349
11.73 CMeaDeviceNet Class Reference	353
11.73.1 Detailed Description	354
11.73.2 Constructor & Destructor Documentation	354
11.73.3 Member Function Documentation	355
11.73.4 Property Documentation	360
11.74 CMeaDigitalDataFunctionNet Class Reference	361
11.74.1 Constructor & Destructor Documentation	362
11.74.2 Member Function Documentation	362
11.75 CMeaFeedbackFunctionNet Class Reference	363
11.75.1 Constructor & Destructor Documentation	364
11.75.2 Member Function Documentation	364
11.76 CMeaImpedanceDeviceNet Class Reference	368
11.76.1 Constructor & Destructor Documentation	368
11.76.2 Member Function Documentation	368
11.77 CMeasureTableDeviceNet Class Reference	369
11.77.1 Detailed Description	370
11.77.2 Constructor & Destructor Documentation	370
11.77.3 Property Documentation	370
11.78 CMeaSwitchDeviceNet Class Reference	370
11.78.1 Detailed Description	371
11.78.2 Constructor & Destructor Documentation	371
11.78.3 Member Function Documentation	371
11.79 CMeaUSBDeviceNet Class Reference	372
11.79.1 Detailed Description	372
11.79.2 Constructor & Destructor Documentation	372

11.79.3 Member Function Documentation	373
11.80 CMeFunctionNet Class Reference	373
11.80.1 Detailed Description	374
11.80.2 Constructor & Destructor Documentation	374
11.80.3 Member Function Documentation	374
11.81 CMultiBatteryChargerDeviceNet Class Reference	374
11.81.1 Detailed Description	376
11.81.2 Constructor & Destructor Documentation	376
11.81.3 Member Function Documentation	376
11.82 CMultiwellCallbackFunctionNet Class Reference	382
11.82.1 Detailed Description	382
11.82.2 Constructor & Destructor Documentation	382
11.82.3 Member Function Documentation	383
11.82.4 Event Documentation	383
11.83 CMultiwellDeviceNet Class Reference	384
11.83.1 Detailed Description	385
11.83.2 Constructor & Destructor Documentation	385
11.83.3 Member Function Documentation	386
11.84 CMultiwellOptoStimFunctionNet Class Reference	391
11.84.1 Detailed Description	392
11.84.2 Constructor & Destructor Documentation	392
11.84.3 Member Function Documentation	392
11.85 CNF_GenDeviceNet Class Reference	396
11.85.1 Constructor & Destructor Documentation	396
11.85.2 Member Function Documentation	397
11.86 COctoPotDeviceNet Class Reference	397
11.86.1 Constructor & Destructor Documentation	398
11.86.2 Member Function Documentation	398
11.87 COkuvisionStimulatorDeviceNet Class Reference	401
11.87.1 Constructor & Destructor Documentation	402
11.87.2 Member Function Documentation	402
11.88 CPatchServerDeviceNet Class Reference	404
11.88.1 Detailed Description	405
11.88.2 Constructor & Destructor Documentation	405
11.88.3 Property Documentation	405
11.89 CPathIdentDeviceNet Class Reference	405
11.89.1 Constructor & Destructor Documentation	406
11.89.2 Member Function Documentation	406
11.90 CPedoterDeviceNet Class Reference	406
11.90.1 Detailed Description	407
11.90.2 Constructor & Destructor Documentation	407
11.90.3 Member Function Documentation	407

11.91 CPeristalticPumpDeviceNet Class Reference	408
11.91.1 Detailed Description	408
11.91.2 Constructor & Destructor Documentation	408
11.91.3 Property Documentation	409
11.92 CPgaDeviceNet Class Reference	409
11.92.1 Constructor & Destructor Documentation	409
11.92.2 Member Function Documentation	410
11.93 CPositionIIDeviceNet Class Reference	411
11.93.1 Detailed Description	413
11.93.2 Constructor & Destructor Documentation	413
11.93.3 Member Function Documentation	413
11.93.4 Property Documentation	420
11.94 CPositionImpDeviceNet Class Reference	420
11.94.1 Detailed Description	420
11.94.2 Constructor & Destructor Documentation	421
11.94.3 Member Function Documentation	421
11.95 CPPCDeviceNet Class Reference	423
11.95.1 Constructor & Destructor Documentation	423
11.95.2 Property Documentation	424
11.96 CPPCFunctionNet Class Reference	424
11.96.1 Detailed Description	425
11.96.2 Constructor & Destructor Documentation	425
11.96.3 Member Function Documentation	426
11.97 CPPS_DeviceNet Class Reference	433
11.97.1 Constructor & Destructor Documentation	433
11.97.2 Property Documentation	433
11.98 CPPS_FunctionNet Class Reference	434
11.98.1 Constructor & Destructor Documentation	434
11.98.2 Member Function Documentation	435
11.99 CPPSDeviceNet Class Reference	438
11.99.1 Detailed Description	438
11.99.2 Constructor & Destructor Documentation	439
11.100 CProgramPressureCurveNet Class Reference	439
11.100.1 Detailed Description	439
11.100.2 Constructor & Destructor Documentation	439
11.100.3 Member Function Documentation	440
11.101 CPulseGeneratorFunctionNet Class Reference	440
11.101.1 Detailed Description	441
11.101.2 Constructor & Destructor Documentation	441
11.101.3 Member Function Documentation	442
11.102 CRadioControlledDevicesNet Class Reference	443
11.102.1 Constructor & Destructor Documentation	444

11.102.2 Member Function Documentation	444
11.103 CCMOSMeaDeviceNet::CRegionOfInterestRect Class Reference	445
11.103.1 Constructor & Destructor Documentation	445
11.103.2 Member Function Documentation	445
11.103.3 Member Data Documentation	446
11.104 CRetinaLedDeviceNet Class Reference	446
11.104.1 Constructor & Destructor Documentation	447
11.104.2 Member Function Documentation	447
11.105 CRFFunctionNet Class Reference	448
11.105.1 Detailed Description	449
11.105.2 Constructor & Destructor Documentation	449
11.105.3 Member Function Documentation	449
11.106 CRobo_FYIProgram_FunctionNet Class Reference	453
11.106.1 Constructor & Destructor Documentation	453
11.106.2 Member Function Documentation	453
11.107 CRobo_FYITemp_FunctionNet Class Reference	454
11.107.1 Constructor & Destructor Documentation	455
11.107.2 Member Function Documentation	455
11.108 CRoboDacqNet Class Reference	456
11.108.1 Constructor & Destructor Documentation	459
11.108.2 Member Function Documentation	459
11.108.3 Member Data Documentation	470
11.109 CRoboDeviceNet Class Reference	471
11.109.1 Detailed Description	474
11.109.2 Constructor & Destructor Documentation	474
11.109.3 Member Function Documentation	474
11.109.4 Member Data Documentation	482
11.109.5 Property Documentation	486
11.109.6 Event Documentation	487
11.110 CRoboFluidDeviceNet Class Reference	487
11.110.1 Constructor & Destructor Documentation	488
11.110.2 Member Function Documentation	488
11.110.3 Member Data Documentation	490
11.110.4 Property Documentation	490
11.111 CRoboInjectDeviceNet Class Reference	490
11.111.1 Detailed Description	491
11.111.2 Constructor & Destructor Documentation	491
11.112 CRoboocyte2DeviceNet Class Reference	491
11.112.1 Detailed Description	491
11.112.2 Constructor & Destructor Documentation	492
11.112.3 Member Function Documentation	492
11.113 CRoboStatorDeviceNet Class Reference	492

11.113.1 Constructor & Destructor Documentation	494
11.113.2 Member Function Documentation	494
11.113.3 Property Documentation	498
11.114 CSafeISDeviceNet Class Reference	498
11.114.1 Detailed Description	499
11.114.2 Constructor & Destructor Documentation	499
11.114.3 Member Function Documentation	500
11.114.4 Property Documentation	501
11.115 CSCUDacqGroupChannelSelectionNet Class Reference	501
11.115.1 Constructor & Destructor Documentation	502
11.116 CSCUFunctionNet Class Reference	502
11.116.1 Detailed Description	505
11.116.2 Constructor & Destructor Documentation	505
11.116.3 Member Function Documentation	505
11.116.4 Event Documentation	520
11.117 CSerialPortNet Class Reference	521
11.117.1 Constructor & Destructor Documentation	521
11.117.2 Member Function Documentation	521
11.118 CStg200xBasicNet Class Reference	522
11.118.1 Detailed Description	527
11.118.2 Constructor & Destructor Documentation	527
11.118.3 Member Function Documentation	527
11.119 CStg200xDownloadBasicNet Class Reference	561
11.119.1 Detailed Description	563
11.119.2 Member Function Documentation	563
11.119.3 Property Documentation	570
11.120 CStg200xDownloadNet Class Reference	570
11.120.1 Detailed Description	571
11.120.2 Constructor & Destructor Documentation	571
11.120.3 Member Function Documentation	571
11.120.4 Event Documentation	576
11.121 CStimulusFunctionNet Class Reference	577
11.121.1 Constructor & Destructor Documentation	578
11.121.2 Member Function Documentation	579
11.121.3 Event Documentation	589
11.122 CSw2to64DeviceNet Class Reference	589
11.122.1 Detailed Description	589
11.122.2 Constructor & Destructor Documentation	590
11.122.3 Member Function Documentation	590
11.123 CTcxDeviceNet Class Reference	591
11.123.1 Detailed Description	593
11.123.2 Constructor & Destructor Documentation	593

11.123.3 Member Function Documentation	594
11.124 CTEERFunctionNet Class Reference	604
11.124.1 Detailed Description	606
11.124.2 Constructor & Destructor Documentation	606
11.124.3 Member Function Documentation	607
11.125 CTEERMachineDeviceNet Class Reference	615
11.125.1 Constructor & Destructor Documentation	615
11.125.2 Property Documentation	616
11.126 CUsbDeviceConfigurationFunctionNet Class Reference	616
11.126.1 Detailed Description	616
11.126.2 Constructor & Destructor Documentation	616
11.126.3 Member Function Documentation	617
11.127 CUsbExceptionNet Class Reference	617
11.127.1 Detailed Description	618
11.127.2 Constructor & Destructor Documentation	618
11.127.3 Property Documentation	618
11.128 CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet Class Reference	618
11.128.1 Constructor & Destructor Documentation	619
11.128.2 Member Data Documentation	619
11.129 CW2100_FunctionNet Class Reference	619
11.129.1 Constructor & Destructor Documentation	621
11.129.2 Member Function Documentation	621
11.129.3 Property Documentation	626
11.130 CW2100_StimulatorFunctionNet Class Reference	627
11.130.1 Constructor & Destructor Documentation	628
11.130.2 Member Function Documentation	628
11.130.3 Member Data Documentation	633
11.130.4 Event Documentation	634
11.131 CW2100DacqGroupChannelSelectionNet Class Reference	634
11.131.1 Constructor & Destructor Documentation	634
11.132 CWarnerUssingDeviceNet Class Reference	634
11.132.1 Detailed Description	635
11.132.2 Constructor & Destructor Documentation	635
11.132.3 Property Documentation	635
11.133 CWarnerUssingFunctionNet Class Reference	636
11.133.1 Detailed Description	638
11.133.2 Constructor & Destructor Documentation	638
11.133.3 Member Function Documentation	638
11.134 CWarnerValveControllerDeviceNet Class Reference	652
11.134.1 Detailed Description	656
11.134.2 Constructor & Destructor Documentation	656
11.134.3 Member Function Documentation	656

11.134.4 Event Documentation	673
11.135 CWarnerValveControllerDeviceTesterFunctionNet Class Reference	676
11.135.1 Detailed Description	677
11.135.2 Constructor & Destructor Documentation	677
11.135.3 Member Function Documentation	678
11.136 CWClassicFunctionNet Class Reference	679
11.136.1 Constructor & Destructor Documentation	680
11.136.2 Member Function Documentation	681
11.137 CWirelessBaseFunctionNet Class Reference	684
11.137.1 Constructor & Destructor Documentation	685
11.137.2 Member Function Documentation	685
11.138 DeviceIdNet Struct Reference	685
11.138.1 Detailed Description	685
11.138.2 Constructor & Destructor Documentation	686
11.138.3 Member Function Documentation	686
11.138.4 Member Data Documentation	686
11.139 DigitalSource< digitalsourceenum > Class Template Reference	687
11.139.1 Constructor & Destructor Documentation	687
11.139.2 Member Function Documentation	687
11.139.3 Property Documentation	688
11.140 DigitalSourceGeneral Class Reference	688
11.140.1 Constructor & Destructor Documentation	688
11.140.2 Member Function Documentation	688
11.140.3 Property Documentation	689
11.141 DriverVersionNet Class Reference	689
11.141.1 Detailed Description	690
11.141.2 Constructor & Destructor Documentation	690
11.141.3 Member Function Documentation	690
11.142 FirmwareDestinationNames Class Reference	695
11.142.1 Member Data Documentation	696
11.143 HeadStageIDType Class Reference	698
11.143.1 Member Enumeration Documentation	699
11.143.2 Constructor & Destructor Documentation	699
11.143.3 Member Function Documentation	700
11.143.4 Property Documentation	700
11.144 HeadstageIDTypeObject Class Reference	701
11.144.1 Constructor & Destructor Documentation	702
11.144.2 Member Function Documentation	702
11.144.3 Member Data Documentation	702
11.144.4 Property Documentation	703
11.145 HeadStageIDTypeState Class Reference	703
11.145.1 Property Documentation	703

11.146 mkfilterNet Class Reference	704
11.146.1 Member Function Documentation	704
11.147 CRoboDeviceNet::RoboMainLowLevelCommands Class Reference	707
11.147.1 Member Function Documentation	708
11.148 CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands Class Reference	714
11.148.1 Member Function Documentation	714
11.149 CFilterCoefficientsNet::s_FilterAttributesNet Struct Reference	714
11.149.1 Constructor & Destructor Documentation	714
11.149.2 Member Function Documentation	714
11.149.3 Member Data Documentation	715
11.150 CMeaAudioFunctionNet::s_setaudionet Struct Reference	715
11.150.1 Member Data Documentation	715
11.151 CStimulusFunctionNet::SidebandData Class Reference	716
11.151.1 Constructor & Destructor Documentation	716
11.151.2 Property Documentation	716
11.152 StgStatusNet Class Reference	717
11.152.1 Member Function Documentation	717
11.152.2 Member Data Documentation	717
11.153 CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData Class Reference	717
11.153.1 Constructor & Destructor Documentation	718
11.153.2 Property Documentation	718
11.154 usbSetupPacket_t Class Reference	719
11.154.1 Member Data Documentation	719
11.155 W2100_StimulusParametersNet Struct Reference	719
11.155.1 Member Data Documentation	720
Index	721

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

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

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

2 Device Classes

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

2.1 The MCS FluidControl Device

2.1.1 Introduction

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

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

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

2.1.2 Access to the FluidControl device

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

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

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

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

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

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

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

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

2.2 MCS-USB-Sw2to64 device

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

First construct an object of the class:

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

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

To get the number of channels the device handles:

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

Set all channel switches at once:

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

Get all channel switches at once:

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

Set one channel switch:

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

Get one channel switch:

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

3 Function Classes

- [Mcs.Usb.CCMOSMea_FunctionNet](#)
- [Mcs.Usb.CDacCalibrationFunctionNet](#)
- [Mcs.Usb.CDigOutStimulatorFunctionNet](#)
- [Mcs.Usb.CGrapheneFunctionNet](#)
- [Mcs.Usb.CIntanMea_FunctionNet](#)
- [Mcs.Usb.CInterfaceboard2FunctionNet](#)
- [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.CPulseGeneratorFunctionNet](#)
- [Mcs.Usb.CRFFFunctionNet](#)
- [Mcs.Usb.CRobo_FYITemp_FunctionNet](#)
- [Mcs.Usb.CRobo_FYIProgram_FunctionNet](#)
- [Mcs.Usb.CSCUFunctionNet](#)
- [Mcs.Usb.CStimulusFunctionNet](#)
- [Mcs.Usb.CTEERFunctionNet](#)
- [Mcs.Usb.CW2100_FunctionNet](#)
- [Mcs.Usb.CW2100_StimulatorFunctionNet](#)
- [Mcs.Usb.CWarnerUssingFunctionNet](#)
- [Mcs.Usb.CWarnerValveControllerDeviceTesterFunctionNet](#)
- [Mcs.Usb.CWClassicFunctionNet](#)
- [Mcs.Usb.CWirelessBaseFunctionNet](#)

4 Data ACQuisition (DACQ) Devices

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

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

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

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

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

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

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

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

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

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

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

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

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

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

see MEA_Recording in the Examples directory.

5 The MCS Robo Device

5.1 Introduction

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

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

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

6 STG200x & STG400x STimulus Generator

6.1 Introduction

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

6.2 Download mode

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

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

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

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

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

6.2.1 Memory Layout and Trigger Setup

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

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

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

Mcs	23
Mcs::Usb	23

8 Hierarchical Index

8.1 Class Hierarchy

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

CW2100_FunctionNet::AudioChannelsNet	95
BatteryState	95
CCreateFilterNet	117
Bessel3dBFilterBandPassNet	96
Bessel3dBFilterHighPassNet	96
Bessel3dBFilterLowPassNet	97
BesselConstDelayFilterBandPassNet	98
BesselConstDelayFilterHighPassNet	98
BesselConstDelayFilterLowPassNet	99
BesselLegacyFilterHighPassNet	99
BesselLegacyFilterLowPassNet	100
ButterworthFilterBandPassNet	100

ButterworthFilterHighPassNet	101
ButterworthFilterLowPassNet	102
CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumTemplateNet >	128
CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumNet >	128
CDeviceGroupChannelInfoNet	127
CDeviceGroupChannelInfoTemplateNet< int >	128
CDeviceGroupChannelInfoGenericNet	126
CDeviceGroupChannelInfoTemplateNet< MEA2100_256DacqGroupChannelEnumNet >	128
CDeviceGroupChannelInfoMEA2100_256Net	126
CDeviceGroupChannelInfoTemplateNet< SCUDacqGroupChannelEnumNet >	128
CDeviceGroupChannelInfoSCUNet	128
CDeviceGroupChannelInfoTemplateNet< W2100DacqGroupChannelEnumNet >	128
CDeviceGroupChannelInfoW2100Net	129
CFilterCoefficientsNet	137
CFilterPropertyNet	143
CMcsUsbDacqNet::ChannelBlockNet	187
CMcsUsbDacqNet::CHWInfo	214
CMcsUsbFunctionNet	308
CDacqGroupChannelSelectionTemplateNet< W2100DacqGroupChannelEnumNet, W2100DacqGroupChannelEnum, CDeviceGroupChannelInfoW2100Net >	123
CW2100DacqGroupChannelSelectionNet	634
CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumNet, DacqGroupChannelEnum, CDeviceGroupChannelInfoNet >	123
CDacqGroupChannelSelectionNet	123
CDacqGroupChannelSelectionTemplateNet< int, int, CDeviceGroupChannelInfoGenericNet >	123
CDacqGroupChannelGenericSelectionNet	122
CDacqGroupChannelSelectionTemplateNet< SCUDacqGroupChannelEnumNet, SCUDacqGroupChannelEnum, CDeviceGroupChannelInfoSCUNet >	123
CSCUDacqGroupChannelSelectionNet	501
CDacqGroupChannelSelectionTemplateNet< MEA2100_256DacqGroupChannelEnumNet, MEA2100_256DacqGroupChannelEnum, CDeviceGroupChannelInfoMEA2100_256Net >	123
CMEA2100_256DacqGroupChannelSelectionNet	338
CCMOSMea_FunctionNet	103
CDacCalibrationFunctionNet	120

CDacqGroupChannelSelectionTemplateNet < DacqGroupChannelEnumTemplateNet, Dacq↔ GroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet >	123
CDigOutStimulatorFunctionNet	130
CFilterConfigurationNet	139
CFilterConfigurationRegisterNet	141
CGrapheneFunctionNet	174
CIntanMea_FunctionNet	217
CInterfaceboardFunctionNet	221
CInterfaceboard2FunctionNet	219
CMEA2100x256FunctionNet	339
CMcsBusNet	271
CMcsBus_AxisParametersNet	233
CMcsBus_ExtensionNet	234
CMcsBus_FYIExtensionNet	235
CMcsBus_MotorControlNet	237
CMcsBus_SensorNet	254
CMcsBus_TempSensorNet	265
CMcsBus_VoltageModeNet	267
CMcsUsbDeviceStatePushFunctionNet	298
CMultiwellCallbackFunctionNet	382
CSCUFunctionNet	502
CMeFunctionNet	373
CMeaAudioFunctionNet	340
CMeaDigitalDataFunctionNet	361
CMeaFeedbackFunctionNet	363
CMultiwellOptoStimFunctionNet	391
CPPCFunctionNet	424
CPPS_FunctionNet	434
CProgramPressureCurveNet	439
CPulseGeneratorFunctionNet	440
CRFFunctionNet	448
CRobo_FYIProgram_FunctionNet	453

CRobo_FYITemp_FunctionNet	454
CStimulusFunctionNet	577
CTEERFunctionNet	604
CUsbDeviceConfigurationFunctionNet	616
CW2100_StimulatorFunctionNet	627
CWarnerUssingFunctionNet	636
CWarnerValveControllerDeviceTesterFunctionNet	676
CWirelessBaseFunctionNet	684
CW2100_FunctionNet	619
CWClassicFunctionNet	679
CMcsUsbFunctionPointerContainer	310
CMcsUsbListEntryNet	310
CMcsUsbListNet	314
CMcsUsbNet	317
CExternDTesterDeviceNet	135
CFluidControlDeviceNet	144
CGenericDevelopDeviceNet	153
CGilsonDeviceNet	171
CMcsUsbDacqNet	275
CMeaDeviceNet	353
CMeaUSBDeviceNet	372
CCMOSMeaDeviceNet	114
CGrapheneASICDeviceNet	173
CHLADacqNet	212
CLIH3DeviceNet	223
CMultiwellDeviceNet	384
CWarnerUssingDeviceNet	634
COctoPotDeviceNet	397
CRoboDacqNet	456
CMcsUsbDeviceStatePushNet	299
CWarnerValveControllerDeviceNet	652
CMcsUsbFactoryNet	300

CMeaCleanDeviceNet	343
CMeaCoatDeviceNet	347
CMeaImpedanceDeviceNet	368
CMeaSwitchDeviceNet	370
CChannelTestDeviceNet	102
CMultiBatteryChargerDeviceNet	374
CNF_GenDeviceNet	396
COkuvisionStimulatorDeviceNet	401
CPPCDeviceNet	423
CPPS_DeviceNet	433
CPathIdentDeviceNet	405
CPedoterDeviceNet	406
CPeristalticPumpDeviceNet	408
CPgaDeviceNet	409
CPositionIIDeviceNet	411
CPositionImpDeviceNet	420
CRadioControlledDevicesNet	443
CRetinaLedDeviceNet	446
CRoboDeviceNet	471
CEncapsulatorDeviceNet	134
CFYIDeviceNet	151
CHLADeviceNet	213
CHiClampDeviceNet	211
CMeasureTableDeviceNet	369
CPPSDeviceNet	438
CPatchServerDeviceNet	404
CRoboStatorDeviceNet	492
CRoboInjectDeviceNet	490
CRoboocyte2DeviceNet	491
CTEERMachineDeviceNet	615
CRoboFluidDeviceNet	487
CSafeISDeviceNet	498

CSerialPortNet	521
CStg200xBasicNet	522
CStg200xDownloadBasicNet	561
CStg200xDownloadNet	570
CSw2to64DeviceNet	589
CTcxDeviceNet	591
CMcsUsbPointerContainer	338
CCMOSMeaDeviceNet::CRegionOfInterestRect	445
CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet	618
DeviceIdNet	685
DigitalSource< digitalsourceenum >	687
DigitalSourceGeneral	688
DriverVersionNet	689
Exception	
CUsbExceptionNet	617
FirmwareDestinationNames	695
HeadstageIDTypeObject	701
HeadStageIDTypeState	703
IComparable	
HeadStageIDType	698
mkfilterNet	704
CRoboDeviceNet::RoboMainLowLevelCommands	707
CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands	714
CFilterCoefficientsNet::s_FilterAttributesNet	714
CMeaAudioFunctionNet::s_setaudionet	715
CStimulusFunctionNet::SidebandData	716
StgStatusNet	717
stgstreaming	
CStg200xBasicNet	522
CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData	717
usbSetupPacket_t	719
W2100_StimulusParametersNet	719

9 Class Index

9.1 Class List

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

CW2100_FunctionNet::AudioChannelsNet	95
BatteryState	95
Bessel3dBFilterBandPassNet	96
Bessel3dBFilterHighPassNet	96
Bessel3dBFilterLowPassNet	97
BesselConstDelayFilterBandPassNet	98
BesselConstDelayFilterHighPassNet	98
BesselConstDelayFilterLowPassNet	99
BesselLegacyFilterHighPassNet	99
BesselLegacyFilterLowPassNet	100
ButterworthFilterBandPassNet	100
ButterworthFilterHighPassNet	101
ButterworthFilterLowPassNet	102
CChannelTestDeviceNet	102
CCMOSMea_FunctionNet	103
CCMOSMeaDeviceNet	114
CCreateFilterNet	117
CDacCalibrationFunctionNet 120	
CDacqGroupChannelGenericSelectionNet	122
CDacqGroupChannelSelectionNet	123
CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplateNet> 123	
CDeviceGroupChannelInfoGenericNet	126
CDeviceGroupChannelInfoMEA2100_256Net	126
CDeviceGroupChannelInfoNet	127
CDeviceGroupChannelInfoSCUNet	128
CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumTemplateNet >	128
CDeviceGroupChannelInfoW2100Net	129

CDigOutStimulatorFunctionNet	
CDigOutStimulatorFunctionNet is the class of the DigOut stimulator function class	130
CEncapsulatorDeviceNet	
CEncapsulatorDeviceNet is the to control the MCS HiClamp device	134
CExternDTesterDeviceNet	
CExternDTesterDeviceNet is the class to access the ExternD Tester (Handheld Device Tester D)	135
CFilterCoefficientsNet	137
CFilterConfigurationNet	139
CFilterConfigurationRegisterNet	141
CFilterPropertyNet	143
CFluidControlDeviceNet	
CFluidControlDeviceNet is the class to control MCS FluidControl (FCB and FCX) device	144
CFYIDeviceNet	
CFYIDeviceNet is the class to control the MCS FYI device	151
CGenericDevelopDeviceNet	
CGenericDevelopDeviceNet is the class to use during development of a new device	153
CGilsonDeviceNet	
CGilsonDeviceNet is the class to control a Gilson device	171
CGrapheneASICDeviceNet	173
CGrapheneFunctionNet	
CGrapheneFunctionNet is the class to control Graphene device functions	174
CMcsUsbDacqNet::ChannelBlockNet	187
CHiClampDeviceNet	
CHiClampDeviceNet is the to control the MCS HiClamp device	211
CHLADacqNet	212
CHLADeviceNet	
CHLADeviceNet is the to control the MCS HLA device	213
CMcsUsbDacqNet::CHWInfo	
Class to provide hardware information about the device	214
CIntanMea_FunctionNet	217
CInterfaceboard2FunctionNet	
CInterfaceboard2FunctionNet is the class to control the Interfaceboard	219
CInterfaceboardFunctionNet	
CInterfaceboardFunctionNet is the class to control the Interfaceboard	221
CLIH3DeviceNet	
CLIH3DeviceNet is the class to access the HEKA LIH3 device	223
CMcsBus_AxisParametersNet	233
CMcsBus_ExtensionNet	234

CMcsBus_FYIExtensionNet	235
CMcsBus_MotorControlNet	237
CMcsBus_SensorNet	254
CMcsBus_TempSensorNet	265
CMcsBus_VoltageModeNet	267
CMcsBusNet	271
CMcsUsbDacqNet	
Base class for data acquisition devices	275
CMcsUsbDeviceStatePushFunctionNet	298
CMcsUsbDeviceStatePushNet	299
CMcsUsbFactoryNet	300
CMcsUsbFunctionNet	308
CMcsUsbFunctionPointerContainer	310
CMcsUsbListEntryNet	
McsUsbListEntryNet identifies a connected device	310
CMcsUsbListNet	
Class to handle a list of connected MCS USB devices	314
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	317
CMcsUsbPointerContainer	338
CMEA2100_256DacqGroupChannelSelectionNet	338
CMEA2100x256FunctionNet	
CMEA2100x256FunctionNet is the class to control the MEA2100-256 device needs #include "<Stg200xNet.h" to resolve documentation reference	339
CMeaAudioFunctionNet	340
CMeaCleanDeviceNet	
CMeaCleanDeviceNet is the class to access the MEA Clean device	343
CMeaCoatDeviceNet	
CMeaCoatDeviceNet is the class to access the MEA Coat device	347
CMeaDeviceNet	
Base class for MEA data acquisition devices	353
CMeaDigitalDataFunctionNet	361
CMeaFeedbackFunctionNet	363
CMeaImpedanceDeviceNet	368
CMeasureTableDeviceNet	
CMeasureTableDeviceNet is the to control the MCS HLA device	369

CMeaSwitchDeviceNet	
The class to control the USB-MEA-Switch	370
CMeaUSBDeviceNet	
Class for data acquisition via ME and MEA USB amplifiers	372
CMeFunctionNet	
373	
CMultiBatteryChargerDeviceNet	
CMultiBatteryChargerDeviceNet is the class to access the MBC-08 device	374
CMultiwellCallbackFunctionNet	
CMultiwellCallbackFunctionNet is the class to access the Multiwell-Mini-Stimulator	382
CMultiwellDeviceNet	
CMultiwellDeviceNet is the class to access the Multiwell device	384
CMultiwellOptoStimFunctionNet	
CMultiwellOptoStimFunctionNet is the class to access the optical properties of the Multiwell Optostim device	391
CNF_GenDeviceNet	396
COctoPotDeviceNet	397
CokuvisionStimulatorDeviceNet	401
CPatchServerDeviceNet	
CPatchServerDeviceNet is the class to control the MCS PatchServer device	404
CPathIdentDeviceNet	405
CPedoterDeviceNet	
406	
CPeristalticPumpDeviceNet	
CPeristalticPumpDeviceNet is the class to control a Persistaltic Pump	408
CPgaDeviceNet	409
CPositionIIDeviceNet	
CPositionIIDeviceNet is the class to control PositionII devices	411
CPositionImpDeviceNet	
CPositionImpDeviceNet is the class to access the Position/Imp devices	420
CPPCDeviceNet	423
CPPCFunctionNet	
CPPCFunctionNet is the class to access the PPC (high precision Patch Peristaltic patch Pump	424
CPPS_DeviceNet	433
CPPS_FunctionNet	434
CPPSDeviceNet	
CPPS4plus1DeviceNet is the to control the MCS HLA device	438
CProgramPressureCurveNet	
CProgramPressureCurveNet is the class to program pressure curves	439

CPulseGeneratorFunctionNet	
CPulseGeneratorFunctionNet is the class to control the pulse generator for video tracking	440
CRadioControlledDevicesNet	443
CCMOSMeaDeviceNet::CRegionOfInterestRect	445
CRetinaLedDeviceNet	446
CRFFFunctionNet	
CRFFFunctionNet is the class to control RF devices	448
CRobo_FYIProgram_FunctionNet	453
CRobo_FYITemp_FunctionNet	454
CRoboDacqNet	456
CRoboDeviceNet	
CRoboDeviceNet is the base class for all Robo platform based devices	471
CRoboFluidDeviceNet	487
CRoboInjectDeviceNet	
CRoboInjectDeviceNet is the to control the MCS RoboInject device	490
CRoboocyte2DeviceNet	
CRoboocyte2DeviceNet is the class to control the MCS Roboocyte2 device	491
CRoboStatorDeviceNet	492
CSafeISDeviceNet	
498	
CSCUDacqGroupChannelSelectionNet	501
CSCUFunctionNet	
CSCUFunctionNet is the class to control the SCU device	502
CSerialPortNet	521
CStg200xBasicNet	
Base class for the Stg200x	522
CStg200xDownloadBasicNet	
CStg200xDownloadBasicNet is the base class to control the download mode of the MCS STG device	561
CStg200xDownloadNet	
Main class for the STG download mode This class implements the STG download mode interface.	570
CStimulusFunctionNet	577
CSw2to64DeviceNet	
The class to control the MCS-USB-Sw2to64 device	589
CTcxDeviceNet	
Class to control a Temperature Controller (TCX)	591
CTEERFunctionNet	
CTEERFunctionNet is the class to control the TEER device	604

CTEERMachineDeviceNet	615
CUsbDeviceConfigurationFunctionNet CUsbDeviceConfigurationFunctionNet is the class to configure the USB firmware	616
CUsbExceptionNet Exception class that is thrown in case of an USB error	617
CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet	618
CW2100_FunctionNet	619
CW2100_StimulatorFunctionNet	627
CW2100DacqGroupChannelSelectionNet	634
CWarnerUssingDeviceNet CWarnerUssingDeviceNet is the class to control the Ussing device	634
CWarnerUssingFunctionNet CWarnerUssingFunctionNet is the class to control the Ussing device	636
CWarnerValveControllerDeviceNet CWarnerValveControllerDeviceNet is the class to access the Warner Valve Controller	652
CWarnerValveControllerDeviceTesterFunctionNet CWarnerValveControllerDeviceTesterFunctionNet is the class to access the functions for the Warner Valve Controller Device Tester	676
CWClassicFunctionNet	679
CWirelessBaseFunctionNet	684
DeviceIdNet Device Id	685
DigitalSource< digitalsourceenum >	687
DigitalSourceGeneral	688
DriverVersionNet Class gives firmware versions of the device's firmware destinations	689
FirmwareDestinationNames	695
HeadStageIDType	698
HeadstageIDTypeObject	701
HeadStageIDTypeState	703
mkfilterNet	704
CRoboDeviceNet::RoboMainLowLevelCommands	707
CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands	714
CFilterCoefficientsNet::s_FilterAttributesNet	714
CMeaAudioFunctionNet::s_setaudionet	715
CStimulusFunctionNet::SidebandData	716

StgStatusNet	717
CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData	717
usbSetupPacket_t	719
W2100_StimulusParametersNet	719

10 Namespace Documentation

10.1 Mcs Namespace Reference

Namespaces

- [Usb](#)

10.2 Mcs::Usb Namespace Reference

Classes

- class [CChannelTestDeviceNet](#)
- class [CCMOSMea_FunctionNet](#)
- class [CCMOSMeaDeviceNet](#)
- class [CCreateFilterNet](#)
- class [ButterworthFilterLowPassNet](#)
- class [ButterworthFilterHighPassNet](#)
- class [ButterworthFilterBandPassNet](#)
- class [BesselLegacyFilterLowPassNet](#)
- class [BesselLegacyFilterHighPassNet](#)
- class [BesselConstDelayFilterLowPassNet](#)
- class [BesselConstDelayFilterHighPassNet](#)
- class [BesselConstDelayFilterBandPassNet](#)
- class [Bessel3dBFilterLowPassNet](#)
- class [Bessel3dBFilterHighPassNet](#)
- class [Bessel3dBFilterBandPassNet](#)
- class [CDeviceGroupChannelInfoTemplateNet](#)
- class [CDeviceGroupChannelInfoGenericNet](#)
- class [CDeviceGroupChannelInfoNet](#)
- class [CDeviceGroupChannelInfoW2100Net](#)
- class [CDeviceGroupChannelInfoSCUNet](#)
- class [CDeviceGroupChannelInfoMEA2100_256Net](#)
- class [CDacqGroupChannelSelectionTemplateNet](#)
- class [CDacqGroupChannelGenericSelectionNet](#)
- class [CDacqGroupChannelSelectionNet](#)
- class [CW2100DacqGroupChannelSelectionNet](#)
- class [CSCUDacqGroupChannelSelectionNet](#)
- class [CMEA2100_256DacqGroupChannelSelectionNet](#)
- class [CDacCalibrationFunctionNet](#)
- class [CDigOutStimulatorFunctionNet](#)
- *[CDigOutStimulatorFunctionNet](#) is the class of the DigOut stimulator function class.*
- class [CExternDTesterDeviceNet](#)

- CExternDTesterDeviceNet* is the class to access the ExternD Tester (Handheld Device Tester D)
- class **CGrapheneFunctionNet**

CGrapheneFunctionNet is the class to control Graphene device functions
- 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 **CMEA2100x256FunctionNet**

CMEA2100x256FunctionNet is the class to control the MEA2100-256 device needs #include "Stg200xNet.h" to resolve documentation reference
- 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 **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 **CPedoterDeviceNet**
- class **CPositionIIDeviceNet**

CPositionIIDeviceNet is the class to control PositionII devices
- class **CPositionImpDeviceNet**

CPositionImpDeviceNet is the class to access the Position/Imp devices
- class **CPPCFunctionNet**

CPPCFunctionNet is the class to access the PPC (high precision Patch Peristaltic patch Pump
- class **CPulseGeneratorFunctionNet**

CPulseGeneratorFunctionNet is the class to control the pulse generator for video tracking
- class **CRFFunctionNet**

CRFFunctionNet is the class to control RF devices
- class **CSCUFunctionNet**

CSCUFunctionNet is the class to control the SCU device
- class **CTEERFunctionNet**

CTEERFunctionNet is the class to control the TEER device
- class **CUsbDeviceConfigurationFunctionNet**

CUsbDeviceConfigurationFunctionNet is the class to configure the USB firmware
- 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
- struct **DeviceldNet**

Device Id.

- class [CFilterCoefficientsNet](#)
- class [CFilterConfigurationNet](#)
- class [CFilterConfigurationRegisterNet](#)
- class [CFilterPropertyNet](#)
- class [CFluidControlDeviceNet](#)

CFluidControlDeviceNet is the class to control MCS FluidControl (FCB and FCX) 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 [CGrapheneASICDeviceNet](#)
- class [CIntanMea_FunctionNet](#)
- class [CMcsBusNet](#)
- class [CMcsBus_MotorControlNet](#)
- class [CMcsBus_VoltageModeNet](#)
- class [CMcsBus_AxisParametersNet](#)
- class [CMcsBus_SensorNet](#)
- class [CMcsBus_TempSensorNet](#)
- class [CMcsBus_ExtensionNet](#)
- class [CMcsBus_FYIExtensionNet](#)
- class [CSerialPortNet](#)
- class [usbSetupPacket_t](#)
- class [CMcsUsbDeviceStatePushFunctionNet](#)
- class [CMcsUsbDeviceStatePushNet](#)
- class [CMcsUsbFactoryNet](#)
- class [CMcsUsbFunctionPointerContainer](#)
- class [CMcsUsbFunctionNet](#)
- class [CMcsUsbListEntryNet](#)

McsUsbListEntryNet identifies a connected device.

- class [CMcsUsbListNet](#)

Class to handle a list of connected MCS USB devices.

- class [CUsbExceptionNet](#)

Exception class that is thrown in case of an USB error.

- class [FirmwareDestinationNames](#)
- class [DriverVersionNet](#)

Class gives firmware versions of the device's firmware destinations.

- class [CMcsUsbPointerContainer](#)
- 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 [DigitalSourceGeneral](#)
- class [DigitalSource](#)
- class [StgStatusNet](#)
- class [CMeaAudioFunctionNet](#)
- class [CMeaDeviceNet](#)

Base class for MEA data acquisition devices.

- class [CMeaUSBDeviceNet](#)

Class for data acquisition via ME and MEA USB amplifiers

- class [CMeaDigitalDataFunctionNet](#)
- class [CMeaFeedbackFunctionNet](#)
- class [CMeaImpedanceDeviceNet](#)
- class [CMeaSwitchDeviceNet](#)

The class to control the USB-MEA-Switch.

- class [CMeFunctionNet](#)
- class [mkfilterNet](#)
- class [CNF_GenDeviceNet](#)
- class [COctoPotDeviceNet](#)
- class [CokuvisionStimulatorDeviceNet](#)
- class [CPathIdentDeviceNet](#)
- class [CPeristalticPumpDeviceNet](#)

CPeristalticPumpDeviceNet is the class to control a Persistaltic Pump.

- class [CPgaDeviceNet](#)
- class [CPPCDeviceNet](#)
- class [CPPS_DeviceNet](#)
- class [CPPS_FunctionNet](#)
- class [CProgramPressureCurveNet](#)

CProgramPressureCurveNet is the class to program pressure curves

- class [CRadioControlledDevicesNet](#)
- class [CRetinaLedDeviceNet](#)
- class [CRobo_FYITemp_FunctionNet](#)
- class [CRobo_FYIProgram_FunctionNet](#)
- class [CRoboDacqNet](#)
- class [CHLADacqNet](#)
- class [CRoboDeviceNet](#)

CRoboDeviceNet is the base class for all Robo platform based devices

- class [CRoboStatorDeviceNet](#)
- class [CRoboocyte2DeviceNet](#)

CRoboocyte2DeviceNet is the class to control the MCS Roboocyte2 device

- class [CRoboInjectDeviceNet](#)

CRoboInjectDeviceNet is the to control the MCS RoboInject device

- class [CHiClampDeviceNet](#)

CHiClampDeviceNet is the to control the MCS HiClamp device

- class [CEncapsulatorDeviceNet](#)

CEncapsulatorDeviceNet is the to control the MCS HiClamp device

- class [CHLADeviceNet](#)

CHLADeviceNet is the to control the MCS HLA device

- class [CPPSDeviceNet](#)

CPPS4plus1DeviceNet is the to control the MCS HLA device

- class [CMeasureTableDeviceNet](#)

CMeasureTableDeviceNet is the to control the MCS HLA device

- class [CFYIDeviceNet](#)

CFYIDeviceNet is the class to control the MCS FYI device

- class [CPatchServerDeviceNet](#)

CPatchServerDeviceNet is the class to control the MCS PatchServer device

- class [CTEERMachinDeviceNet](#)
- class [CRoboFluidDeviceNet](#)
- class [CSafeISDeviceNet](#)
- class [CStg200xDownloadNet](#)

Main class for the STG download mode This class implements the STG download mode interface.

- 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 [CStimulusFunctionNet](#)

- class [CSw2to64DeviceNet](#)
The class to control the MCS-USB-Sw2to64 device.
- class [CTcxDeviceNet](#)
Class to control a Temperature Controller (TCX)
- class [CMcsUsbDacqNet](#)
Base class for data acquisition devices.
- struct [W2100_StimulusParametersNet](#)
- class [HeadStageIDType](#)
- class [HeadStageIDTypeState](#)
- class [HeadstageIDTypeObject](#)
- class [BatteryState](#)
- class [CW2100_StimulatorFunctionNet](#)
- class [CW2100_FunctionNet](#)
- class [CWClassicFunctionNet](#)
- class [CWirelessBaseFunctionNet](#)

Enumerations

- enum class [enCMosMeaChipType](#) {
[unknown](#) = 0 ,
[nMos16LV](#) = 1 ,
[nMos32LV](#) = 3 ,
[nMos36LN](#) = 6 ,
[nMos64LN](#) = 7 }
- enum class [DeviceEnumNet](#) {
[MCS_DEVICE_ANY](#) = EN_DEVICETYPE_MCS_ANY ,
[MCS_GENERIC_DEVELOPMENT_DEVICE](#) = EN_DEVICETYPE_MCS_GENERIC_DEVELOPMENT ,
[MCS_DEVICE_USB](#) = EN_DEVICETYPE_MCS_USB ,
[MCS_MCCARD_DEVICE](#) = EN_DEVICETYPE_MCS_MCCARD ,
[MCS_STG_DEVICE](#) = EN_DEVICETYPE_MCS_STG ,
[MCS_MC_STIMULUS_DEVICE](#) = EN_DEVICETYPE_MCS_MC_STIMULUS ,
[MCS_MEASUSB_DEVICE](#) = EN_DEVICETYPE_MCS_MEASUSB ,
[MCS_MEA_DEVICE](#) = EN_DEVICETYPE_MCS_MEA ,
[MCS_OCTOPOT_DEVICE](#) = EN_DEVICETYPE_MCS_OCTOPOT ,
[MCS_TERSENS_DEVICE](#) = EN_DEVICETYPE_MCS_TERSENS ,
[MCS_PGA_DEVICE](#) = EN_DEVICETYPE_MCS_PGA ,
[MCS_PCX_DEVICE](#) = EN_DEVICETYPE_MCS_PCX ,
[MCS_TCX_DEVICE](#) = EN_DEVICETYPE_MCS_TCX ,
[MCS_FCX_DEVICE](#) = EN_DEVICETYPE_MCS_FCX ,
[MCS_RETINA_LED_DEVICE](#) = EN_DEVICETYPE_MCS_RETINA_LED ,
[MCS_MEA_SWITCH_DEVICE](#) = EN_DEVICETYPE_MCS_MEA_SWITCH ,
[MCS_MEA_IMPEDANCE_DEVICE](#) = EN_DEVICETYPE_MCS_MEA_IMPEDANCE ,
[MCS_CHANNELTEST_DEVICE](#) = EN_DEVICETYPE_MCS_CHANNELTEST ,
[MCS_SW2TO64_DEVICE](#) = EN_DEVICETYPE_MCS_SW2TO64 ,
[MCS_RETINA_AMS_DONGLE](#) = EN_DEVICETYPE_MCS_RETINA_AMS_DONGLE ,
[MCS_PATHIDENT_DEVICE](#) = EN_DEVICETYPE_MCS_PATHIDENT ,
[MCS_ROBO_DEVICE](#) = EN_DEVICETYPE_MCS_ROBO ,
[MCS_ROBOOCYTE2_DEVICE](#) = EN_DEVICETYPE_MCS_ROBOOCYTE2 ,
[MCS_ROBOINJECT_DEVICE](#) = EN_DEVICETYPE_MCS_ROBOINJECT ,
[MCS_HICLAMP_DEVICE](#) = EN_DEVICETYPE_MCS_HICLAMP ,
[MCS_PATCHSERVER_DEVICE](#) = EN_DEVICETYPE_MCS_PATCHSERVER ,
[MCS_ENCAPSULATOR_DEVICE](#) = EN_DEVICETYPE_MCS_ENCAPSULATOR ,
[MCS_MEASURETABLE_DEVICE](#) = EN_DEVICETYPE_MCS_MEASURETABLE ,
[MCS_FYI_DEVICE](#) = EN_DEVICETYPE_MCS_FYI ,
[MCS_HLA_DEVICE](#) = EN_DEVICETYPE_MCS_HLA ,
[MCS_PPS_DEVICE](#) = EN_DEVICETYPE_MCS_PPS ,

```

MCS_PPS5_DEVICE = EN_DEVICETYPE_MCS_PPS5 ,
MCS_OKUVISION_STIMULATOR_DEVICE = EN_DEVICETYPE_MCS_OKUVISION_STIMULATOR ,
MCS_NF_GEN_DEVICE = EN_DEVICETYPE_MCS_NF_GEN ,
MCS_SAFEIS_DEVICE = EN_DEVICETYPE_MCS_SAFEIS ,
MCS_PERISTALTIC_PUMP_DEVICE = EN_DEVICETYPE_MCS_PERISTALTIC_PUMP ,
MCS_EXTERN_BC_TESTER_DEVICE = EN_DEVICETYPE_MCS_EXTERN_BC_TESTER ,
MCS_EXTERN_D_TESTER_DEVICE = EN_DEVICETYPE_MCS_EXTERN_D_TESTER ,
MCS_SOFTWARE_DONGLE_DEVICE = EN_DEVICETYPE_MCS_SOFTWARE_DONGLE ,
MCS_MEA_CLEAN_DEVICE = EN_DEVICETYPE_MCS_MEA_CLEAN ,
MCS_MEA_COAT_DEVICE = EN_DEVICETYPE_MCS_MEA_COAT ,
MCS_SMARTIMPLANT_DEVICE = EN_DEVICETYPE_MCS_SMARTIMPLANT ,
MCS_MBC08_DEVICE = EN_DEVICETYPE_MCS_MBC08 ,
MCS_PEDOTER_DEVICE = EN_DEVICETYPE_MCS_PEDOTER ,
MCS_PPC_DEVICE = EN_DEVICETYPE_MCS_PPC ,
WARNER_VALVE_CONTROL_DEVICE = EN_DEVICETYPE_WARNER_VALVE_CONTROL ,
WARNER_USSING_DEVICE = EN_DEVICETYPE_WARNER_USSING ,
HEKA_LIH3_DEVICE = EN_DEVICETYPE_HEKA_LIH3 ,
ALA_VC3_DEVICE = EN_DEVICETYPE_ALA_VC3 ,
MCS_DEVICE_USB_CYPRESS = EN_DEVICETYPE_MCS_USB_CYPRESS ,
MCS_DEVICE_NONE = EN_DEVICETYPE_NONE }

```

Enumerates the group of MCS devices to connect to.

- enum class [VendorIdEnumNet](#) {
 [Any](#) = -1 ,
 [None](#) = 0 ,
 [MCS_old](#) = 0xABCD ,
 [MCS](#) = MCS_VENDOR_ID ,
 [PCI](#) = 0x10E8 ,
 [Cypress](#) = CYPRESS_VENDOR_ID ,
 [ALA_VC3](#) = ALA_VC3_VENDOR_ID }

Enumerates the group of MCS devices to connect to.

- enum class [ProductIdEnumNet](#) {
 [Any](#) = -1 ,
 [None](#) = 0 ,
 [LegacyMeaUsb](#) = MCS_PRODUCT_ID_MEASUSB ,
 [ALA_VC3](#) = ALA_VC3_VENDOR_ID ,
 [Cypress_FX1](#) = CY_FX1_PRODUCT_ID ,
 [Cypress_FX2](#) = CY_FX2_PRODUCT_ID ,
 [Cypress_FX3](#) = CY_FX3_PRODUCT_ID ,
 [MC_Card](#) = MCS_PRODUCT_ID_MC_CARD ,
 [Campden_Ci4600EphysVideoDataIntegrator](#) = MCS_PRODUCT_ID_CAMPDEN_CI4600EPHYS_VIDEO<-_DATA_INTEGRATOR ,
 [HekaLIH30](#) = MCS_PRODUCT_ID_HEKA_LIH30 ,
 [HekaEPC10Single](#) = MCS_PRODUCT_ID_HEKA_EPC10_SINGLE ,
 [HekaEPC10Double](#) = MCS_PRODUCT_ID_HEKA_EPC10_DOUBLE ,
 [HekaEPC10Triple](#) = MCS_PRODUCT_ID_HEKA_EPC10_TRIPLE ,
 [HekaEPC10Quadro](#) = MCS_PRODUCT_ID_HEKA_EPC10_QUADRO ,
 [HekaLIH408](#) = MCS_PRODUCT_ID_HEKA_LIH_408 ,
 [HekaLIH816](#) = MCS_PRODUCT_ID_HEKA_LIH_816 ,
 [HekaITEV100](#) = MCS_PRODUCT_ID_HEKA_ITEV_100 ,
 [HekaPG610](#) = MCS_PRODUCT_ID_HEKA_PG_610 ,
 [HekaPG611](#) = MCS_PRODUCT_ID_HEKA_PG_611 ,
 [HekaPG612](#) = MCS_PRODUCT_ID_HEKA_PG_612 ,
 [HekaPG618](#) = MCS_PRODUCT_ID_HEKA_PG_618 ,
 [HekaPG690](#) = MCS_PRODUCT_ID_HEKA_PG_690 ,
 [HekaEPCLite](#) = MCS_PRODUCT_ID_HEKA_EPC_Lite ,
 [STG](#) = MCS_PRODUCT_ID_STG ,
 [Octopot](#) = MCS_PRODUCT_ID_OCTOPOT ,
 [Tersens](#) = MCS_PRODUCT_ID_TERSENS ,

`Dotriapot` = MCS_PRODUCT_ID_DOTRIAPOT ,
`HLA` = MCS_PRODUCT_ID_HLA ,
`STG400x` = MCS_PRODUCT_ID_STG400x ,
`STG4002` = MCS_PRODUCT_ID_STG4002 ,
`STG4004` = MCS_PRODUCT_ID_STG4004 ,
`STG4008` = MCS_PRODUCT_ID_STG4008 ,
`STG400x_opto` = MCS_PRODUCT_ID_STG400x_OPTO ,
`STG4002_opto` = MCS_PRODUCT_ID_STG4002_OPTO ,
`STG4004_opto` = MCS_PRODUCT_ID_STG4004_OPTO ,
`STG4008_opto` = MCS_PRODUCT_ID_STG4008_OPTO ,
`STG5` = MCS_PRODUCT_ID_STG5 ,
`STG3008_FA` = MCS_PRODUCT_ID_STG3008_FA ,
`MultiwellOptoStim` = MCS_PRODUCT_ID_MULTIWELLOPTOSTIM ,
`Generic` = MCS_PRODUCT_ID_GENERIC ,
`PGA` = MCS_PRODUCT_ID_PGA ,
`PCX` = MCS_PRODUCT_ID_PCX ,
`TCX` = MCS_PRODUCT_ID_TCX ,
`FCX` = MCS_PRODUCT_ID_FCX ,
`FCB` = MCS_PRODUCT_ID_FCB ,
`TC01` = MCS_PRODUCT_ID_TC01 ,
`TC02` = MCS_PRODUCT_ID_TC02 ,
`Retina_LED` = MCS_PRODUCT_ID_RETINA_LED ,
`AMS_Dongle` = MCS_PRODUCT_ID_RETINA_AMS_DONGLE ,
`Okuvision_Stimulator` = MCS_PRODUCT_ID_OKUVISION_STIMULATOR ,
`ExternBCTester` = MCS_PRODUCT_ID_RETINAIMPLANT_EXTERNBCTESTER ,
`Triggerbox_IMS` = MCS_PRODUCT_ID_RIAG_TRIGGERBOX_IMS ,
`Triggerbox_AMS` = MCS_PRODUCT_ID_RIAG_TRIGGERBOX_AMS ,
`Triggerbox_AMS3` = MCS_PRODUCT_ID_RIAG_TRIGGERBOX_AMS3 ,
`ExternDTester` = MCS_PRODUCT_ID_RETINAIMPLANT_EXTERNNDTESTER ,
`FunkDongleS` = MCS_PRODUCT_ID_RIAG_FUNKDONGLES ,
`ExternSTester` = MCS_PRODUCT_ID_RIAG_EXTERNSTESTER ,
`DongleS` = MCS_PRODUCT_ID_RIAG_DONGLES ,
`Triggerbox_R5` = MCS_PRODUCT_ID_RIAG_TRIGGERBOX_R5 ,
`MEA_Switch` = MCS_PRODUCT_ID_MEA_SWITCH ,
`MEA_Impedance` = MCS_PRODUCT_ID_MEA_IMPEDANCE ,
`ChannelTest` = MCS_PRODUCT_ID_CHANNELTEST ,
`Sw2to64` = MCS_PRODUCT_ID_SW2TO64 ,
`PeristalticPump` = MCS_PRODUCT_ID_PERISTALTIC_PUMP ,
`MEA_Switch_2_1` = MCS_PRODUCT_ID_MEA_SWITCH_2_1 ,
`MEA_Switch_4_2` = MCS_PRODUCT_ID_MEA_SWITCH_4_2 ,
`PPS4plus1` = MCS_PRODUCT_ID_PPS4plus1 ,
`PPS5` = MCS_PRODUCT_ID_PPS5 ,
`PPS2` = MCS_PRODUCT_ID_PPS2 ,
`PPS5_DIG` = MCS_PRODUCT_ID_PPS5_DIG ,
`MEA_Clean` = MCS_PRODUCT_ID_MEA_CLEAN ,
`MEA_Coat` = MCS_PRODUCT_ID_MEA_COAT ,
`Multiwell_ICC` = MCS_PRODUCT_ID_MULTIWELL_ICC ,
`MBC08` = MCS_PRODUCT_ID_MBC08 ,
`PPC` = MCS_PRODUCT_ID_PPC ,
`MEA1060` = MCS_PRODUCT_ID_MEA1060 ,
`MEA_Sanofi` = MCS_PRODUCT_ID_MEA_SANOFI ,
`ME256` = MCS_PRODUCT_ID_ME256 ,
`ME128` = MCS_PRODUCT_ID_ME128 ,
`ME64` = MCS_PRODUCT_ID_ME64 ,
`ME32` = MCS_PRODUCT_ID_ME32 ,
`ME16` = MCS_PRODUCT_ID_ME16 ,
`MEA2100_Mini_Usb_develop` = MCS_PRODUCT_ID_MEA2100_MINI_USB_DEVELOP ,
`MEA256` = MCS_PRODUCT_ID_MEA256 ,

```

MEA2100 = MCS_PRODUCT_ID_MEA2100 ,
MEA2100_32 = MCS_PRODUCT_ID_MEA2100_32 ,
MEA2100_Lite = MCS_PRODUCT_ID_MEA21_LITE ,
Multiwell = MCS_PRODUCT_ID_MULTIWELL ,
MEA2100_256 = MCS_PRODUCT_ID_MEA2100_256 ,
ME2100 = MCS_PRODUCT_ID_ME2100 ,
MEA2100BetaScreen = MCS_PRODUCT_ID_MEA2100_BETA_SCREEN ,
MEA2100_Mini = MCS_PRODUCT_ID_MEA2100_MINI ,
TBSI_Dacq = MCS_PRODUCT_ID_TBSI_DACQ ,
Multiwell_MEA_Mini = MCS_PRODUCT_ID_MULTIWELL_MEA_MINI ,
Whole_Cell_Patch = MCS_PRODUCT_ID_WHOLE_CELL_PATCH ,
eCube = MCS_PRODUCT_ID_ECUBE ,
Graphene_ASIC = MCS_PRODUCT_ID_GRAPHENE_ASIC ,
GE2100 = MCS_PRODUCT_ID_GE2100 ,
M4M_Dock = MCS_PRODUCT_ID_M4MDOCK ,
Multiboot = MCS_PRODUCT_ID_MULTIBOOT ,
WPA8 = MCS_PRODUCT_ID_WPA8 ,
WPA4 = MCS_PRODUCT_ID_WPA4 ,
WPA16 = MCS_PRODUCT_ID_WPA16 ,
WPA32 = MCS_PRODUCT_ID_WPA32 ,
W2100 = MCS_PRODUCT_ID_W2100 ,
NeuroChip = MCS_PRODUCT_ID_NEUROCHIP ,
UsbTest = MCS_PRODUCT_ID_USB_TEST ,
SoftwareDongle = MCS_PRODUCT_ID_SOFTWAREDONGLE ,
PathIdent = MCS_PRODUCT_ID_PATHIDENT ,
NF_Gen = MCS_PRODUCT_ID_NF_GEN ,
SafeIS = MCS_PRODUCT_ID_SAFEIS ,
Encapsulator = MCS_PRODUCT_ID_ENCAPSULATOR ,
NeurochipConfig = MCS_PRODUCT_ID_NEUROCHIP_CONFIG ,
MeasureTable = MCS_PRODUCT_ID_MEASURETABLE ,
Roboocyte2 = MCS_PRODUCT_ID_ROBOOCYTE2 ,
RoboInject = MCS_PRODUCT_ID_ROBOINJECT ,
HiClamp = MCS_PRODUCT_ID_HICLAMP ,
PatchServer = MCS_PRODUCT_ID_PATCHSERVER ,
Dilutor = MCS_PRODUCT_ID_DILUTOR ,
HiClamp4Uart = MCS_PRODUCT_ID_HICLAMP4UART ,
IM16S16KRA = MCS_PRODUCT_ID_IM16S16KRA ,
IM64KRB = MCS_PRODUCT_ID_IM64KRB ,
IS32KRA = MCS_PRODUCT_ID_IS32KRA ,
IM64KRC = MCS_PRODUCT_ID_IM64KRC ,
IM16S8KRA = MCS_PRODUCT_ID_IM16S8KRA ,
IM16KRC = MCS_PRODUCT_ID_IM16KRC ,
SmartImplant = MCS_PRODUCT_ID_SMARTIMPLANT ,
PositionImp = MCS_PRODUCT_ID_POSITION_IMP ,
PositionBase = MCS_PRODUCT_ID_POSITION_BASE ,
PositionIICentralUnit = MCS_PRODUCT_ID_POSITIONII_CENTRAL_UNIT ,
PositionIIBase = MCS_PRODUCT_ID_POSITIONII_BASE ,
GrapheneProjectTestDevice = MCS_PRODUCT_ID_GRAPHENE_PROJECT_TEST_DEVICE ,
Pos900 = MCS_PRODUCT_ID_POS900 ,
Neptun = MCS_PRODUCT_ID_NEPTUN ,
Warner_Valve_Control = MCS_PRODUCT_ID_WARNER_VALVE_CONTROL ,
Warner_TEER_Machine = MCS_PRODUCT_ID_WARNER_TEER_MACHINE ,
Warner_Ussing = MCS_PRODUCT_ID_WARNER_USSING }

```

Enumerates the group of MCS devices to connect to.

- enum class `McsBusTypeEnumNet` {
`MCS_ANY_BUS` = -1 ,
`MCS_UNDEFINED_BUS` = 0 ,
`MCS_USB_BUS` ,

`MCS_PCI_BUS` }

Enumerates the bus to use, either USB, PCI or any

- enum class `McsUsbSpeedEnumNet` {

`LowSpeed` = 0 ,
`FullSpeed` = 1 ,
`HighSpeed` = 2 ,
`SuperSpeed` = 3 ,
`UnknownSpeed` = 0xff }

Enumerates the current connection speed of the device

- enum class `CFirmwareDestinationNet` {

`FPGA_NORMAL` = 0 ,
`DSP` = MCSUSB_DEST_DSP ,
`USB` = MCSUSB_DEST_USB ,
`MCU1` = MCSUSB_DEST_MCU1 ,
`MCSBUS1` = MCSUSB_DEST_MCSBUS1 ,
`MCSBUS2` = MCSUSB_DEST_MCSBUS2 ,
`MCSBUS3` = MCSUSB_DEST_MCSBUS3 ,
`MCSBUS4` = MCSUSB_DEST_MCSBUS4 ,
`MCSBUS5` = MCSUSB_DEST_MCSBUS5 ,
`MCSBUS6` = MCSUSB_DEST_MCSBUS6 ,
`MCSBUS7` = MCSUSB_DEST_MCSBUS7 ,
`MCSBUS8` = MCSUSB_DEST_MCSBUS8 ,
`MCSBUS9` = MCSUSB_DEST_MCSBUS9 ,
`MCSBUS10` = MCSUSB_DEST_MCSBUS10 ,
`MCSBUS11` = MCSUSB_DEST_MCSBUS11 ,
`MCSBUS12` = MCSUSB_DEST_MCSBUS12 ,
`MCSBUS13` = MCSUSB_DEST_MCSBUS13 ,
`MCSBUS14` = MCSUSB_DEST_MCSBUS14 ,
`MCSBUS15` = MCSUSB_DEST_MCSBUS15 ,
`MCSBUS0` = MCSUSB_DEST_MCSBUS0 ,
`BUSNUMBER0` = MCSUSB_DEST_BUSNUMBER0 ,
`BUS0MCSBUS1` = MCSUSB_DEST_BUS0_MCSBUS1 ,
`BUS0MCSBUS2` = MCSUSB_DEST_BUS0_MCSBUS2 ,
`BUS0MCSBUS3` = MCSUSB_DEST_BUS0_MCSBUS3 ,
`BUS0MCSBUS4` = MCSUSB_DEST_BUS0_MCSBUS4 ,
`BUS0MCSBUS5` = MCSUSB_DEST_BUS0_MCSBUS5 ,
`BUS0MCSBUS6` = MCSUSB_DEST_BUS0_MCSBUS6 ,
`BUS0MCSBUS7` = MCSUSB_DEST_BUS0_MCSBUS7 ,
`BUS0MCSBUS8` = MCSUSB_DEST_BUS0_MCSBUS8 ,
`BUS0MCSBUS9` = MCSUSB_DEST_BUS0_MCSBUS9 ,
`BUS0MCSBUS10` = MCSUSB_DEST_BUS0_MCSBUS10 ,
`BUS0MCSBUS11` = MCSUSB_DEST_BUS0_MCSBUS11 ,
`BUS0MCSBUS12` = MCSUSB_DEST_BUS0_MCSBUS12 ,
`BUS0MCSBUS13` = MCSUSB_DEST_BUS0_MCSBUS13 ,
`BUS0MCSBUS14` = MCSUSB_DEST_BUS0_MCSBUS14 ,
`BUS0MCSBUS15` = MCSUSB_DEST_BUS0_MCSBUS15 ,
`BUS0MCSBUS0` = MCSUSB_DEST_BUS0_MCSBUS0 ,
`BUSNUMBER1` = MCSUSB_DEST_BUSNUMBER1 ,
`BUS1MCSBUS1` = MCSUSB_DEST_BUS1_MCSBUS1 ,
`BUS1MCSBUS2` = MCSUSB_DEST_BUS1_MCSBUS2 ,
`BUS1MCSBUS3` = MCSUSB_DEST_BUS1_MCSBUS3 ,
`BUS1MCSBUS4` = MCSUSB_DEST_BUS1_MCSBUS4 ,
`BUS1MCSBUS5` = MCSUSB_DEST_BUS1_MCSBUS5 ,
`BUS1MCSBUS6` = MCSUSB_DEST_BUS1_MCSBUS6 ,
`BUS1MCSBUS7` = MCSUSB_DEST_BUS1_MCSBUS7 ,
`BUS1MCSBUS8` = MCSUSB_DEST_BUS1_MCSBUS8 ,
`BUS1MCSBUS9` = MCSUSB_DEST_BUS1_MCSBUS9 ,
`BUS1MCSBUS10` = MCSUSB_DEST_BUS1_MCSBUS10 ,

```

BUS1MCSBUS11 = MCSUSB_DEST_BUS1_MCSBUS11 ,
BUS1MCSBUS12 = MCSUSB_DEST_BUS1_MCSBUS12 ,
BUS1MCSBUS13 = MCSUSB_DEST_BUS1_MCSBUS13 ,
BUS1MCSBUS14 = MCSUSB_DEST_BUS1_MCSBUS14 ,
BUS1MCSBUS15 = MCSUSB_DEST_BUS1_MCSBUS15 ,
BUS1MCSBUS0 = MCSUSB_DEST_BUS1_MCSBUS0 ,
BUSNUMBER2 = MCSUSB_DEST_BUSNUMBER2 ,
BUS2MCSBUS1 = MCSUSB_DEST_BUS2_MCSBUS1 ,
BUS2MCSBUS2 = MCSUSB_DEST_BUS2_MCSBUS2 ,
BUS2MCSBUS3 = MCSUSB_DEST_BUS2_MCSBUS3 ,
BUS2MCSBUS4 = MCSUSB_DEST_BUS2_MCSBUS4 ,
BUS2MCSBUS5 = MCSUSB_DEST_BUS2_MCSBUS5 ,
BUS2MCSBUS6 = MCSUSB_DEST_BUS2_MCSBUS6 ,
BUS2MCSBUS7 = MCSUSB_DEST_BUS2_MCSBUS7 ,
BUS2MCSBUS8 = MCSUSB_DEST_BUS2_MCSBUS8 ,
BUS2MCSBUS9 = MCSUSB_DEST_BUS2_MCSBUS9 ,
BUS2MCSBUS10 = MCSUSB_DEST_BUS2_MCSBUS10 ,
BUS2MCSBUS11 = MCSUSB_DEST_BUS2_MCSBUS11 ,
BUS2MCSBUS12 = MCSUSB_DEST_BUS2_MCSBUS12 ,
BUS2MCSBUS13 = MCSUSB_DEST_BUS2_MCSBUS13 ,
BUS2MCSBUS14 = MCSUSB_DEST_BUS2_MCSBUS14 ,
BUS2MCSBUS15 = MCSUSB_DEST_BUS2_MCSBUS15 ,
BUS2MCSBUS0 = MCSUSB_DEST_BUS2_MCSBUS0 ,
PIC = MCSUSB_DEST_PIC ,
PIC2 = MCSUSB_DEST_PIC2 ,
PIC3 = MCSUSB_DEST_PIC3 ,
PIC4 = MCSUSB_DEST_PIC4 ,
PIC5 = MCSUSB_DEST_PIC5 ,
PIC6 = MCSUSB_DEST_PIC6 ,
PIC7 = MCSUSB_DEST_PIC7 ,
PIC8 = MCSUSB_DEST_PIC8 ,
PIC9 = MCSUSB_DEST_PIC9 ,
PIC10 = MCSUSB_DEST_PIC10 ,
PIC11 = MCSUSB_DEST_PIC11 ,
PIC12 = MCSUSB_DEST_PIC12 ,
ChannelPIC = MCSUSB_DEST_CHANNELPIC ,
Bootstrap = MCSUSB_DEST_BOOTSTRAP ,
BootstrapOtherCypress = MCSUSB_DEST_BOOTSTAP_OTHER_CYPRESS ,
ALTERA = MCSUSB_DEST_ALTERA ,
FPGA2 = MCSUSB_DEST_FPGA2 ,
FPGA3 = MCSUSB_DEST_FPGA3 ,
FPGA4 = MCSUSB_DEST_FPGA4 ,
FPGA5 = MCSUSB_DEST_FPGA5 ,
FPGA6 = MCSUSB_DEST_FPGA6 ,
FPGA7 = MCSUSB_DEST_FPGA7 ,
FPGA8 = MCSUSB_DEST_FPGA8 ,
FPGA9 = MCSUSB_DEST_FPGA9 ,
FPGA10 = MCSUSB_DEST_FPGA10 ,
FPGA11 = MCSUSB_DEST_FPGA11 ,
FPGA12 = MCSUSB_DEST_FPGA12 ,
FPGA13 = MCSUSB_DEST_FPGA13 ,
FPGA14 = MCSUSB_DEST_FPGA14 ,
FPGA15 = MCSUSB_DEST_FPGA15 ,
FPGA16 = MCSUSB_DEST_FPGA16 ,
FPGA_GOLD = XILINX_DEST_GOLDEN ,
ALTERA_GOLD = (MCSUSB_DEST_ALTERA | XILINX_DEST_GOLDEN) ,
FPGA2_GOLD = (MCSUSB_DEST_FPGA2 | XILINX_DEST_GOLDEN) ,
FPGA3_GOLD = (MCSUSB_DEST_FPGA3 | XILINX_DEST_GOLDEN) ,

```



```

FPGA4_GOLD = (MCSUSB_DEST_FPGA4 | XILINX_DEST_GOLDEN) ,
FPGA5_GOLD = (MCSUSB_DEST_FPGA5 | XILINX_DEST_GOLDEN) ,
FPGA6_GOLD = (MCSUSB_DEST_FPGA6 | XILINX_DEST_GOLDEN) ,
FPGA7_GOLD = (MCSUSB_DEST_FPGA7 | XILINX_DEST_GOLDEN) ,
FPGA8_GOLD = (MCSUSB_DEST_FPGA8 | XILINX_DEST_GOLDEN) ,
FPGA9_GOLD = (MCSUSB_DEST_FPGA9 | XILINX_DEST_GOLDEN) ,
FPGA10_GOLD = (MCSUSB_DEST_FPGA10 | XILINX_DEST_GOLDEN) ,
FPGA11_GOLD = (MCSUSB_DEST_FPGA11 | XILINX_DEST_GOLDEN) ,
FPGA12_GOLD = (MCSUSB_DEST_FPGA12 | XILINX_DEST_GOLDEN) ,
FPGA13_GOLD = (MCSUSB_DEST_FPGA13 | XILINX_DEST_GOLDEN) ,
FPGA14_GOLD = (MCSUSB_DEST_FPGA14 | XILINX_DEST_GOLDEN) ,
FPGA15_GOLD = (MCSUSB_DEST_FPGA15 | XILINX_DEST_GOLDEN) ,
FPGA16_GOLD = (MCSUSB_DEST_FPGA16 | XILINX_DEST_GOLDEN) ,
FPGA_BASE = XILINX_DEST_BASEIMAGE ,
ALTERA_BASE = (MCSUSB_DEST_ALTERA | XILINX_DEST_BASEIMAGE) ,
FPGA2_BASE = (MCSUSB_DEST_FPGA2 | XILINX_DEST_BASEIMAGE) ,
FPGA3_BASE = (MCSUSB_DEST_FPGA3 | XILINX_DEST_BASEIMAGE) ,
FPGA4_BASE = (MCSUSB_DEST_FPGA4 | XILINX_DEST_BASEIMAGE) ,
FPGA5_BASE = (MCSUSB_DEST_FPGA5 | XILINX_DEST_BASEIMAGE) ,
FPGA6_BASE = (MCSUSB_DEST_FPGA6 | XILINX_DEST_BASEIMAGE) ,
FPGA7_BASE = (MCSUSB_DEST_FPGA7 | XILINX_DEST_BASEIMAGE) ,
FPGA8_BASE = (MCSUSB_DEST_FPGA8 | XILINX_DEST_BASEIMAGE) ,
FPGA9_BASE = (MCSUSB_DEST_FPGA9 | XILINX_DEST_BASEIMAGE) ,
FPGA10_BASE = (MCSUSB_DEST_FPGA10 | XILINX_DEST_BASEIMAGE) ,
FPGA11_BASE = (MCSUSB_DEST_FPGA11 | XILINX_DEST_BASEIMAGE) ,
FPGA12_BASE = (MCSUSB_DEST_FPGA12 | XILINX_DEST_BASEIMAGE) ,
FPGA13_BASE = (MCSUSB_DEST_FPGA13 | XILINX_DEST_BASEIMAGE) ,
FPGA14_BASE = (MCSUSB_DEST_FPGA14 | XILINX_DEST_BASEIMAGE) ,
FPGA15_BASE = (MCSUSB_DEST_FPGA15 | XILINX_DEST_BASEIMAGE) ,
FPGA16_BASE = (MCSUSB_DEST_FPGA16 | XILINX_DEST_BASEIMAGE) ,
FPGA_BOOTSTRAP = XILINX_DEST_BOOTSTRAP ,
ALTERA_BOOTSTRAP = (MCSUSB_DEST_ALTERA | XILINX_DEST_BOOTSTRAP) ,
DEST_TARGET1 = FLASH_DEST_TARGET1 ,
DEST_TARGET2 = FLASH_DEST_TARGET2 ,
DEST_TARGET3 = FLASH_DEST_TARGET3 ,
DEST_TARGET4 = FLASH_DEST_TARGET4 ,
DEST_TARGET5 = FLASH_DEST_TARGET5 ,
DEST_TARGET6 = FLASH_DEST_TARGET6 ,
DEST_TARGET7 = FLASH_DEST_TARGET7 ,
DEST_TARGET8 = FLASH_DEST_TARGET8 ,
DEST_TARGET9 = FLASH_DEST_TARGET9 ,
DEST_TARGET10 = FLASH_DEST_TARGET10 ,
DEST_TARGET11 = FLASH_DEST_TARGET11 ,
DEST_TARGET12 = FLASH_DEST_TARGET12 ,
DEST_TARGET13 = FLASH_DEST_TARGET13 ,
DEST_TARGET14 = FLASH_DEST_TARGET14 ,
DEST_TARGET15 = FLASH_DEST_TARGET15 ,
DEST_TARGET_MASK = FPGA_DEST_TARGET_MASK ,
DEST_FX3_TARGET_MASK = FX3_DEST_TARGET_MASK ,
ALTERA_TARGET1 = (MCSUSB_DEST_ALTERA | FLASH_DEST_TARGET1) ,
ALTERA_TARGET2 = (MCSUSB_DEST_ALTERA | FLASH_DEST_TARGET2) ,
ALTERA_TARGET3 = (MCSUSB_DEST_ALTERA | FLASH_DEST_TARGET3) ,
USB_TARGET1 = (MCSUSB_DEST_USB | FLASH_DEST_TARGET1) ,
USB_TARGET2 = (MCSUSB_DEST_USB | FLASH_DEST_TARGET2) ,
USB_TARGET3 = (MCSUSB_DEST_USB | FLASH_DEST_TARGET3) ,
UnknownDest = MCSUSB_DEST_UNKNOWN }

```

Enumerates the destination processor for the firmware.

- enum class [DigitalTargetEnumNet](#) {

```

Digout = (MEA_COMMAND << 16) + MEA_MEA21_DIGOUT_SOURCE ,
Digstream = (MEA_COMMAND << 16) + MEA_MEA21_DIGSTREAM_SOURCE ,
DacqTrigger = (MEA_COMMAND << 16) + MEA_MEA21_DACQTRIGGER_SOURCE ,
StgTrigger = (STG200x_COMMAND << 16) + STG200x_TRIGGER_SOURCE ,
StgListModeTrigger = (STG200x_COMMAND << 16) + STG200x_MEA21_LISTMODE_TRIGGERSOURCE
,
DigOutStimulatorStartTrigger = (MEA_COMMAND << 16) + MEA_DIGOUT_STG_START_TRIGGER_↵
SOURCE ,
DigOutStimulatorStopTrigger = (MEA_COMMAND << 16) + MEA_DIGOUT_STG_STOP_TRIGGER_↵
SOURCE ,
DigStreamToReceiver = (MEA_COMMAND << 16) + MEA_DIGSTREAMTORECEIVER_SOURCE }

```

Enumerates the Digital Targets for Digital Sources

- enum class `MEA2100DigitalSourceEnumNet` {


```

DigitalInOfOutPort = 0 ,
DigitalIn = 16 ,
DigitalPulse = 32 ,
Feedback = 64 ,
AuxIn = 96 ,
Zero = 98 ,
One = 99 ,
HS1Trigger1Status = 100 ,
HS1Trigger2Status = 102 ,
HS1Trigger3Status = 104 ,
HS1Trigger4Status = 106 ,
HS1Trigger5Status = 108 ,
HS1Trigger6Status = 110 ,
HS1Sideband1 = 112 ,
HS1Sideband2 = 128 ,
HS1Sideband3 = 144 ,
HS1Sideband4 = 160 ,
HS1Sideband5 = 176 ,
HS1Sideband6 = 192 ,
HS2Trigger1Status = 208 ,
HS2Trigger2Status = 210 ,
HS2Trigger3Status = 212 ,
HS2Trigger4Status = 214 ,
HS2Trigger5Status = 216 ,
HS2Trigger6Status = 218 ,
HS2Sideband1 = 220 ,
HS2Sideband2 = 236 ,
HS2Sideband3 = 252 ,
HS2Sideband4 = 268 ,
HS2Sideband5 = 284 ,
HS2Sideband6 = 300 ,
PulseGenerator = 316 ,
DigitalOutStimulator = 320 ,
DigitalData = 336 ,
DeviceRunStatus = 368 ,
LastPosition = 372 }

```

Enumerates the digital source of the MEA2100 device.

- enum class `MultiwellDigitalSourceEnumNet` {


```

DigitalInOfOutPort = 0 ,
DigitalIn = 16 ,
DigitalPulse = 32 ,
Feedback = 64 ,
AuxIn = 96 ,
Zero = 98 ,
One = 99 ,

```



```

HS1Trigger1Status = 100 ,
HS1Trigger2Status = 102 ,
HS1Sideband1 = 112 ,
HS1Sideband2 = 128 ,
HS2Trigger1Status = 208 ,
HS2Trigger2Status = 210 ,
HS2Sideband1 = 220 ,
HS2Sideband2 = 236 ,
PulseGenerator = 316 ,
DigitalOutStimulator = 320 ,
DigitalData = 336 ,
DeviceRunStatus = 368 ,
LastPosition = 372 }

```

Enumerates the digital source of the Multiwell device.

- enum class [CMOSMEA5000DigitalSourceEnumNet](#) {


```

DigitalInOfOutPort = 0 ,
DigitalIn = 16 ,
DigitalPulse = 32 ,
Feedback = 64 ,
AuxIn = 96 ,
Zero = 98 ,
One = 99 ,
HS1Trigger1Status = 100 ,
HS1Trigger2Status = 102 ,
HS1Trigger3Status = 104 ,
HS1Sideband1 = 112 ,
HS1Sideband2 = 128 ,
HS1Sideband3 = 144 ,
PulseGenerator = 316 ,
DigitalOutStimulator = 320 ,
DigitalData = 336 ,
DeviceRunStatus = 368 ,
LastPosition = 372 }

```

Enumerates the digital source of the CMOSMEA5000 device.

- enum class [W2100DigitalSourceEnumNet](#) {


```

DigitalInOfOutPort = 0 ,
DigitalIn = 16 ,
DigitalPulse = 32 ,
Feedback = 64 ,
AuxIn = 96 ,
Zero = 98 ,
One = 99 ,
PulseGenerator = 100 ,
DigDataFromReceiver = 128 ,
DigitalOutStimulator = 192 ,
DigitalData = 208 ,
DeviceRunStatus = 240 ,
DigStreamFromReceiver = 256 ,
LastPosition = 320 }

```

Enumerates the digital source of the W2100 device.

- enum class [SCUDigitalSourceEnumNet](#) {


```

DigitalInOfOutPort = (0x00 << 8) + 0 ,
DigitalIn = (0x00 << 8) + 16 ,
DigitalPulse = (0x01 << 8) ,
Feedback = (0x02 << 8) ,
AuxIn = (0x03 << 8) + 0 ,
Zero = (0x03 << 8) + 2 ,
One = (0x03 << 8) + 3 ,

```

```

PulseGenerator = (0x03 << 8) + 8 ,
DigitalOutStimulator = (0x03 << 8) + 16 ,
DigitalData = (0x04 << 8) ,
DeviceRunStatus = (0x05 << 8) + 0 ,
SCU1HS1Sideband1TriggerStatus = (0x40 << 8) + 0 ,
SCU1HS1Sideband2TriggerStatus = (0x40 << 8) + 2 ,
SCU1HS2Sideband1TriggerStatus = (0x40 << 8) + 4 ,
SCU1HS2Sideband2TriggerStatus = (0x40 << 8) + 6 ,
SCU1HS3Sideband1TriggerStatus = (0x40 << 8) + 8 ,
SCU1HS3Sideband2TriggerStatus = (0x40 << 8) + 10 ,
SCU1HS4Sideband1TriggerStatus = (0x40 << 8) + 12 ,
SCU1HS4Sideband2TriggerStatus = (0x40 << 8) + 14 ,
SCU1LEDStim1TriggerStatus = (0x40 << 8) + 16 ,
SCU1LEDStim2TriggerStatus = (0x40 << 8) + 18 ,
SCU1LEDStim3TriggerStatus = (0x40 << 8) + 20 ,
SCU1LEDStim4TriggerStatus = (0x40 << 8) + 22 ,
SCU1HS1Sideband1 = (0x42 << 8) ,
SCU1HS1Sideband2 = (0x43 << 8) ,
SCU1HS2Sideband1 = (0x44 << 8) ,
SCU1HS2Sideband2 = (0x45 << 8) ,
SCU1HS3Sideband1 = (0x46 << 8) ,
SCU1HS3Sideband2 = (0x47 << 8) ,
SCU1HS4Sideband1 = (0x48 << 8) ,
SCU1HS4Sideband2 = (0x49 << 8) ,
SCU1LEDStim1 = (0x4A << 8) ,
SCU1LEDStim2 = (0x4B << 8) ,
SCU1LEDStim3 = (0x4C << 8) ,
SCU1LEDStim4 = (0x4D << 8) ,
SCU2HS1Sideband1TriggerStatus = (0x80 << 8) + 0 ,
SCU2HS1Sideband2TriggerStatus = (0x80 << 8) + 2 ,
SCU2HS2Sideband1TriggerStatus = (0x80 << 8) + 4 ,
SCU2HS2Sideband2TriggerStatus = (0x80 << 8) + 6 ,
SCU2HS3Sideband1TriggerStatus = (0x80 << 8) + 8 ,
SCU2HS3Sideband2TriggerStatus = (0x80 << 8) + 10 ,
SCU2HS4Sideband1TriggerStatus = (0x80 << 8) + 12 ,
SCU2HS4Sideband2TriggerStatus = (0x80 << 8) + 14 ,
SCU2LEDStim1TriggerStatus = (0x80 << 8) + 16 ,
SCU2LEDStim2TriggerStatus = (0x80 << 8) + 18 ,
SCU2LEDStim3TriggerStatus = (0x80 << 8) + 20 ,
SCU2LEDStim4TriggerStatus = (0x80 << 8) + 22 ,
SCU2HS1Sideband1 = (0x82 << 8) ,
SCU2HS1Sideband2 = (0x83 << 8) ,
SCU2HS2Sideband1 = (0x84 << 8) ,
SCU2HS2Sideband2 = (0x85 << 8) ,
SCU2HS3Sideband1 = (0x86 << 8) ,
SCU2HS3Sideband2 = (0x87 << 8) ,
SCU2HS4Sideband1 = (0x88 << 8) ,
SCU2HS4Sideband2 = (0x89 << 8) ,
SCU2LEDStim1 = (0x8A << 8) ,
SCU2LEDStim2 = (0x8B << 8) ,
SCU2LEDStim3 = (0x8C << 8) ,
SCU2LEDStim4 = (0x8D << 8) ,
LastPosition = (0xFF << 8) }

```

Enumerates the digital source of the SCU device.

- enum class MEA2100_256DigitalSourceEnumNet {
 DigitalInOfOutPort = (0x00 << 8) + 0 ,
 DigitalIn = (0x00 << 8) + 16 ,
 DigitalPulse = (0x01 << 8) ,

```
Feedback = (0x02 << 8) ,
AuxIn = (0x03 << 8) + 0 ,
Zero = (0x03 << 8) + 2 ,
One = (0x03 << 8) + 3 ,
DeviceRunStatus = (0x03 << 8) + 4 ,
PulseGenerator = (0x03 << 8) + 8 ,
DigitalOutStimulator = (0x03 << 8) + 16 ,
DigitalData = (0x04 << 8) ,
HS1Trigger1Status = (0x40 << 8) + 0 ,
HS1Trigger2Status = (0x40 << 8) + 2 ,
HS1Trigger3Status = (0x40 << 8) + 4 ,
HS1Trigger4Status = (0x40 << 8) + 6 ,
HS1Trigger5Status = (0x40 << 8) + 8 ,
HS1Trigger6Status = (0x40 << 8) + 10 ,
HS1Trigger7Status = (0x40 << 8) + 12 ,
HS1Trigger8Status = (0x40 << 8) + 14 ,
HS1Trigger9Status = (0x40 << 8) + 16 ,
HS1Trigger10Status = (0x40 << 8) + 18 ,
HS1Trigger11Status = (0x40 << 8) + 20 ,
HS1Trigger12Status = (0x40 << 8) + 22 ,
HS1Trigger13Status = (0x40 << 8) + 24 ,
HS1Trigger14Status = (0x40 << 8) + 26 ,
HS1Trigger15Status = (0x40 << 8) + 28 ,
HS1Trigger16Status = (0x40 << 8) + 30 ,
HS1Trigger17Status = (0x41 << 8) + 0 ,
HS1Trigger18Status = (0x41 << 8) + 2 ,
HS1Sideband1 = (0x42 << 8) ,
HS1Sideband2 = (0x43 << 8) ,
HS1Sideband3 = (0x44 << 8) ,
HS1Sideband4 = (0x45 << 8) ,
HS1Sideband5 = (0x46 << 8) ,
HS1Sideband6 = (0x47 << 8) ,
HS1Sideband7 = (0x48 << 8) ,
HS1Sideband8 = (0x49 << 8) ,
HS1Sideband9 = (0x4A << 8) ,
HS1Sideband10 = (0x4B << 8) ,
HS1Sideband11 = (0x4C << 8) ,
HS1Sideband12 = (0x4D << 8) ,
HS1Sideband13 = (0x4E << 8) ,
HS1Sideband14 = (0x4F << 8) ,
HS1Sideband15 = (0x50 << 8) ,
HS1Sideband16 = (0x51 << 8) ,
HS1Sideband17 = (0x52 << 8) ,
HS1Sideband18 = (0x53 << 8) ,
HS2Trigger1Status = (0x80 << 8) + 0 ,
HS2Trigger2Status = (0x80 << 8) + 2 ,
HS2Trigger3Status = (0x80 << 8) + 4 ,
HS2Trigger4Status = (0x80 << 8) + 6 ,
HS2Trigger5Status = (0x80 << 8) + 8 ,
HS2Trigger6Status = (0x80 << 8) + 10 ,
HS2Trigger7Status = (0x80 << 8) + 12 ,
HS2Trigger8Status = (0x80 << 8) + 14 ,
HS2Trigger9Status = (0x80 << 8) + 16 ,
HS2Trigger10Status = (0x80 << 8) + 18 ,
HS2Trigger11Status = (0x80 << 8) + 20 ,
HS2Trigger12Status = (0x80 << 8) + 22 ,
HS2Trigger13Status = (0x80 << 8) + 24 ,
HS2Trigger14Status = (0x80 << 8) + 26 ,
```

```

HS2Trigger15Status = (0x80 << 8) + 28 ,
HS2Trigger16Status = (0x80 << 8) + 30 ,
HS2Trigger17Status = (0x81 << 8) + 0 ,
HS2Trigger18Status = (0x81 << 8) + 2 ,
HS2Sideband1 = (0x82 << 8) ,
HS2Sideband2 = (0x83 << 8) ,
HS2Sideband3 = (0x84 << 8) ,
HS2Sideband4 = (0x85 << 8) ,
HS2Sideband5 = (0x86 << 8) ,
HS2Sideband6 = (0x87 << 8) ,
HS2Sideband7 = (0x88 << 8) ,
HS2Sideband8 = (0x89 << 8) ,
HS2Sideband9 = (0x8A << 8) ,
HS2Sideband10 = (0x8B << 8) ,
HS2Sideband11 = (0x8C << 8) ,
HS2Sideband12 = (0x8D << 8) ,
HS2Sideband13 = (0x8E << 8) ,
HS2Sideband14 = (0x8F << 8) ,
HS2Sideband15 = (0x90 << 8) ,
HS2Sideband16 = (0x91 << 8) ,
HS2Sideband17 = (0x92 << 8) ,
HS2Sideband18 = (0x93 << 8) ,
LastPosition = (0xFF << 8) }

```

Enumerates the digital source of the MEA2100-256 device.

- enum class **TBSI_DACQDigitalSourceEnumNet** {


```

DigitalInOfOutPort = (0x00 << 8) + 0 ,
DigitalIn = (0x00 << 8) + 16 ,
DigitalPulse = (0x01 << 8) ,
Feedback = (0x02 << 8) ,
AuxIn = (0x03 << 8) + 0 ,
Zero = (0x03 << 8) + 2 ,
One = (0x03 << 8) + 3 ,
DeviceRunStatus = (0x03 << 8) + 4 ,
PulseGenerator = (0x03 << 8) + 8 ,
DigitalOutStimulator = (0x03 << 8) + 16 ,
DigitalData = (0x04 << 8) ,
HS1DigitalData1 = (0x30 << 8) ,
HS2DigitalData1 = (0x70 << 8) ,
LastPosition = (0xFF << 8) }

```

Enumerates the digital source of the TBSI-DACQ device.

- enum class **TriggerSourceEnumNet** {


```

tsNone = 0 ,
tsDigitalIn1 = 1 ,
tsDigitalIn2 = 2 ,
tsDigitalIn3 = 3 ,
tsDigitalIn4 = 4 ,
tsDigitalIn5 = 5 ,
tsDigitalIn6 = 6 ,
tsDigitalIn7 = 7 ,
tsDigitalIn8 = 8 ,
tsDigitalIn9 = 9 ,
tsDigitalIn10 = 10 ,
tsDigitalIn11 = 11 ,
tsDigitalIn12 = 12 ,
tsDigitalIn13 = 13 ,
tsDigitalIn14 = 14 ,
tsDigitalIn15 = 15 ,
tsDigitalIn16 = 16 ,

```

tsDigitalIn17 = 17 ,
tsDigitalIn18 = 18 ,
tsDigitalIn19 = 19 ,
tsDigitalIn20 = 20 ,
tsDigitalIn21 = 21 ,
tsDigitalIn22 = 22 ,
tsDigitalIn23 = 23 ,
tsDigitalIn24 = 24 ,
tsDigitalIn25 = 25 ,
tsDigitalIn26 = 26 ,
tsDigitalIn27 = 27 ,
tsDigitalIn28 = 28 ,
tsDigitalIn29 = 29 ,
tsDigitalIn30 = 30 ,
tsDigitalIn31 = 31 ,
tsDigitalIn32 = 32 ,
tsFeedback1 = 33 ,
tsFeedback2 = 34 ,
tsFeedback3 = 35 ,
tsFeedback4 = 36 ,
tsFeedback5 = 37 ,
tsFeedback6 = 38 ,
tsFeedback7 = 39 ,
tsFeedback8 = 40 ,
tsFeedback9 = 41 ,
tsFeedback10 = 42 ,
tsFeedback11 = 43 ,
tsFeedback12 = 44 ,
tsFeedback13 = 45 ,
tsFeedback14 = 46 ,
tsFeedback15 = 47 ,
tsFeedback16 = 48 ,
tsFeedback17 = 49 ,
tsFeedback18 = 50 ,
tsFeedback19 = 51 ,
tsFeedback20 = 52 ,
tsFeedback21 = 53 ,
tsFeedback22 = 54 ,
tsFeedback23 = 55 ,
tsFeedback24 = 56 ,
tsFeedback25 = 57 ,
tsFeedback26 = 58 ,
tsFeedback27 = 59 ,
tsFeedback28 = 60 ,
tsFeedback29 = 61 ,
tsFeedback30 = 62 ,
tsFeedback31 = 63 ,
tsFeedback32 = 64 ,
tsAuxIn1 = 65 ,
tsAuxIn2 = 66 ,
tsDigitalPuse0 = 67 ,
tsDigitalPuse1 = 68 ,
tsDigitalPuse2 = 69 ,
tsDigitalPuse3 = 70 ,
tsDigitalPuse4 = 71 ,
tsDigitalPuse5 = 72 ,
tsDigitalPuse6 = 73 ,
tsDigitalPuse7 = 74 ,

```

tsDigitalPuse8 = 75 ,
tsDigitalPuse9 = 76 ,
tsDigitalPuse10 = 77 ,
tsDigitalPuse11 = 78 ,
tsDigitalPuse12 = 79 ,
tsDigitalPuse13 = 80 ,
tsDigitalPuse14 = 81 ,
tsDigitalPuse15 = 82 ,
tsDigitalPuse16 = 83 ,
tsDigitalPuse17 = 84 ,
tsDigitalPuse18 = 85 ,
tsDigitalPuse19 = 86 ,
tsDigitalPuse20 = 87 ,
tsDigitalPuse21 = 88 ,
tsDigitalPuse22 = 89 ,
tsDigitalPuse23 = 90 ,
tsDigitalPuse24 = 91 ,
tsDigitalPuse25 = 92 ,
tsDigitalPuse26 = 93 ,
tsDigitalPuse27 = 94 ,
tsDigitalPuse28 = 95 ,
tsDigitalPuse29 = 96 ,
tsDigitalPuse30 = 97 ,
tsDigitalPuse31 = 98 ,
tsTriggered = 99 ,
tsSidebandBit8 = 100 ,
tsDACQCy1Dev1Runs = 101 ,
tsDACQCy1Dev2Runs = 102 ,
tsDACQCy2Dev1Runs = 103 ,
tsDACQCy2Dev2Runs = 104 }

```

Enumerates the trigger source of the MEA2100 device.

- enum class [AnalogSourceEnumNet](#) {
[AnalogSource_HS1](#) ,
[AnalogSource_HS2](#) ,
[AnalogSource_IF](#) }

Enumerates the analog source of the MEA2100 device.

- enum class [Stg200xTriggerStatusEnumNet](#) {
[Idle](#) = 0 ,
[Running](#) = 1 ,
[Finished](#) = 2 ,
[Armed](#) = 3 }

Enumerates the STG download mode trigger status

- enum class [Stg3008FilterAmpAmplificationEnumNet](#) {
[Gain1000](#) = 0 ,
[Gain500](#) = 1 ,
[Gain200](#) = 2 ,
[Gain100](#) = 3 }

Enumerates the STG3008FA filter amplifier gains

- enum class [RetriggerActionEnumNet](#) {
[Stop](#) = STG200x_RETRIGGER_STOP ,
[Restart](#) = STG200x_RETRIGGER_RESTART ,
[Ignore](#) = STG200x_RETRIGGER_IGNORE ,
[Gate](#) = STG200x_RETRIGGER_GATEMODE ,
[Single](#) = STG200x_RETRIGGER_SINGLE }

Enumerates possible retrigger actions for STG200x devices.

- enum class [Stg200xSegmentFlagsEnumNet](#) {
[None](#) = 0 ,

```
UpdateTrigger = SEGMENTFLAGS_UPDATETRIGGER ,
DownloadOnly = SEGMENTFLAGS_DOWNLOADONLY ,
TriggerOnly = SEGMENTFLAGS_TRIGGERONLY ,
SyncStart = SEGMENTFLAGS_SYNCSTART }
```

Enumerates Segmentflag options for STG400x devices.

- enum class [Stg200xMultiFileSubmodeEnumNet](#) {
[MultiFileMode](#) = MULTIFILESUBMODE_MULTIFILEMODE ,
[ExtendedMultiFileMode](#) = MULTIFILESUBMODE_EXTENDEDMULTIFILEMODE }

Enumerates EnableMultifileMode submodes for STG devices.

- enum class [Stg200xDigoutModeEnumNet](#) {
[Monitor](#) = STG200x_DIGOUTMODE_MONITOR ,
[Manual](#) = STG200x_DIGOUTMODE_MANUAL ,
[SYNCOUT1](#) = STG200x_DIGOUTMODE_SYNCOUT1 ,
[SYNCOUT2](#) = STG200x_DIGOUTMODE_SYNCOUT2 ,
[SYNCOUT3](#) = STG200x_DIGOUTMODE_SYNCOUT3 ,
[SYNCOUT4](#) = STG200x_DIGOUTMODE_SYNCOUT4 ,
[SYNCOUT5](#) = STG200x_DIGOUTMODE_SYNCOUT5 ,
[SYNCOUT6](#) = STG200x_DIGOUTMODE_SYNCOUT6 ,
[SYNCOUT7](#) = STG200x_DIGOUTMODE_SYNCOUT7 ,
[SYNCOUT8](#) = STG200x_DIGOUTMODE_SYNCOUT8 }

Enumerates the DigoutMode on STG400x devices.

- enum class [DigitalStimulatorTriggerSlopeEnumNet](#) {
[Falling](#) = 0 ,
[Rising](#) = 1 }

Enumerates start/stop conditions for DigOut/DigStim trigger. /summary>

- enum class [DigitalStimulatorTriggerEventEnumNet](#) {
[Start](#) = 0 ,
[Stop](#) = 1 }

Enumerates start/stop event for DigOut/DigStim trigger. /summary>

- enum class [AdapterTypeEnumNet](#) {
[None](#) = 0 ,
[MEA60](#) = 1 ,
[MEA2x60](#) = 2 ,
[MEA120](#) = 3 ,
[MEA32](#) = 4 ,
[MEA2x32](#) = 5 ,
[Multiwell96](#) = 6 ,
[WirelessTestAdapter](#) = 7 ,
[MEA252](#) = 8 ,
[MEA_2_252_2](#) = 9 ,
[MEA_2_252_2_6Well](#) = 10 ,
[MEA_2_252_2_9Well](#) = 11 ,
[MEA_2_252_2_Test](#) = 12 ,
[TBSI_5](#) = 13 ,
[TBSI_15](#) = 14 ,
[TBSI_31](#) = 15 ,
[TBSI_63](#) = 16 ,
[TBSI_127](#) = 17 ,
[TBSI_Reserved](#) = 18 ,
[Ci4600Intan](#) = 20 ,
[Unknown](#) = ADAPTER_TYPE_UNKOWN ,
[NotApplicable](#) = ADAPTER_TYPE_ENUM_NOT_APPLICABLE }

Enumerates the adapter type of the MEA2100 device.

- enum class [MeaLayoutEnumNet](#) {
[mlUnknown](#) = 0 ,
[mlMEA60](#) = 1 }

Enumerates the MEA layout of the MEA2100 device.

- enum class `DataModeEnumNet` {
`Unsigned_16bit` = 0 ,
`Unsigned_24bit` = 2 ,
`Unsigned_32bit` = 3 ,
`Signed_16bit` = 8 ,
`Signed_24bit` = 10 ,
`Signed_32bit` = 11 }

Enumerates the data mode of the device, either 16, 24 or 32 bit, can be signed or unsigned.

- enum class `SampleSizeNet` {
`SampleSize16Unsigned` = 2 ,
`SampleSize16Signed` = 2 + 0x100 ,
`SampleSize24Unsigned` = 3 ,
`SampleSize24Signed` = 3 + 0x100 ,
`SampleSize32Unsigned` = 4 ,
`SampleSize32Signed` = 4 + 0x100 ,
`SampleSize64Unsigned` = 8 ,
`SampleSize64Signed` = 8 + 0x100 }

Enumerates the data format for ChannelBlock functions.

- enum class `ChannelBlockTypeNet` {
`OneHandleOneQueue` = 0 ,
`OneHandleMultipleQueues` = 1 ,
`MultipleHandlesOneQueues` = 2 }

Enumerates the type of ChannelBlock functions.

- enum class `SampleDstSizeNet` {
`SampleDstSize16` = 2 ,
`SampleDstSize32` = 4 }

Enumerates the destination data format for ChannelBlock functions.

- enum class `TcxDeviceTypeEnumNet` {
`Unknown` = 0 ,
`Regular` = 1 ,
`BMI` = 2 ,
`Nanion` = 3 ,
`Warner` = 4 }

Enumerates the type of TCX devices.

- enum class `TcxSensorTypeEnumNet` {
`Reserved5` = 0 ,
`Reserved4` = 1 ,
`Reserved3` = 2 ,
`Reserved2` = 3 ,
`Reserved1` = 4 ,
`NTC10K` = 5 ,
`PT1000` = 6 ,
`PT100` = 7 }

Enumerates the sensor types for TCX devices

- enum class `STG_DestinationEnumNet` {
`channeldata_voltage` ,
`channeldata_current` ,
`syncoutdata` ,
`channeldata_positive_voltage` ,
`channeldata_positive_current` ,
`rawdata` ,
`channeldata_current_own_sync` ,
`channeldata_positive_current_own_sync` ,
`channeldata_current_own_boost_gnd_sync` ,
`channeldata_positive_current_own_boost_gnd_sync` ,


```
channeldata_current_always_boost ,
channeldata_current_always_boost_own_sync }
```

Enumerates the destination for STG downloads.

- enum class [ElectrodeModeEnumNet](#) {
[emAutomatic](#) = 0 ,
[emManual](#) = 3 }

Enumerates the mode of each electrode, can be automatic or manual. In automatic mode, the blanking of the electrode is controlled by the sideband signal, in manual mode, the stimulation configuration is independant of the sideband signal.

- enum class [ElectrodeDacMuxEnumNet](#) {
[Ground](#) = 0 ,
[Stg1](#) = 1 ,
[Stg2](#) = 2 ,
[Stg3](#) = 3 }

Enumerates the setting of the Stimulation DAC Multiplexer.

- enum class [DacqGroupChannelEnumNet](#) {
[HeadstageElectrodeGroup](#) = 0x00 ,
[InterfaceADCGroup](#) = INTERFACEANALOGCHANNELSGROUP ,
[DSPDataGroup](#) = DSPDATACHANNELSGROUP ,
[Headstage1NCBathCurrentGroup](#) = 0x30 ,
[Headstage1NCCol2CurrentGroup](#) = 0x31 ,
[Headstage1NChipTempGroup](#) = 0x32 ,
[STG1DACSignalGroup](#) = 0x38 ,
[LIH30ADCModulesGroup](#) = 0x50 ,
[IFDigChannelsGroup](#) = INTERFACEDIGITALCHANNELSGROUP ,
[STG1SidebandsGroup](#) = 0x90 ,
[STG1TriggerStatusGroup](#) = 0x91 ,
[DACQ1DigitalGroup](#) = 0xA0 ,
[AudioTestChannelGroup](#) = AUDIOTESTCHANNELGROUP ,
[PacketFrameContextGroup](#) = PACKETFRAMECONTEXTGROUP }

Enumerates the Channel Groups of Datastream

- enum class [W2100DacqGroupChannelEnumNet](#) {
[InterfaceADCGroup](#) = INTERFACEANALOGCHANNELSGROUP ,
[DSPDataGroup](#) = DSPDATACHANNELSGROUP ,
[WirelessHeadStageAnalogRE1HS1](#) = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 0 + 0 ,
[WirelessHeadStageStatusRE1HS1](#) = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 0 + 1 ,
[WirelessHeadStageAnalogRE1HS2](#) = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 2 + 0 ,
[WirelessHeadStageStatusRE1HS2](#) = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 2 + 1 ,
[WirelessHeadStageAnalogRE1HS3](#) = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 4 + 0 ,
[WirelessHeadStageStatusRE1HS3](#) = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 4 + 1 ,
[WirelessHeadStageAnalogRE1HS4](#) = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 6 + 0 ,
[WirelessHeadStageStatusRE1HS4](#) = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 6 + 1 ,
[WirelessHeadStageAnalogRE2HS1](#) = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 0 + 0 ,
[WirelessHeadStageStatusRE2HS1](#) = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 0 + 1 ,
[WirelessHeadStageAnalogRE2HS2](#) = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 2 + 0 ,
[WirelessHeadStageStatusRE2HS2](#) = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 2 + 1 ,
[WirelessHeadStageAnalogRE2HS3](#) = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 4 + 0 ,
[WirelessHeadStageStatusRE2HS3](#) = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 4 + 1 ,
[WirelessHeadStageAnalogRE2HS4](#) = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 6 + 0 ,
[WirelessHeadStageStatusRE2HS4](#) = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 6 + 1 ,
[WirelessHeadStageGyroDataRE1HS1](#) = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 0 + 0 + 16 ,
[WirelessHeadStageAccDataRE1HS1](#) = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 0 + 1 + 16 ,
[WirelessHeadStageGyroDataRE1HS2](#) = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 2 + 0 + 16 ,
[WirelessHeadStageAccDataRE1HS2](#) = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 2 + 1 + 16 ,
[WirelessHeadStageGyroDataRE1HS3](#) = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 4 + 0 + 16 ,
[WirelessHeadStageAccDataRE1HS3](#) = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 4 + 1 + 16 ,
[WirelessHeadStageGyroDataRE1HS4](#) = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 6 + 0 + 16 ,

WirelessHeadStageAccDataRE1HS4 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 6 + 1 + 16 ,
 WirelessHeadStageGyroDataRE2HS1 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 0 + 0 + 16 ,
 WirelessHeadStageAccDataRE2HS1 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 0 + 1 + 16 ,
 WirelessHeadStageGyroDataRE2HS2 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 2 + 0 + 16 ,
 WirelessHeadStageAccDataRE2HS2 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 2 + 1 + 16 ,
 WirelessHeadStageGyroDataRE2HS3 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 4 + 0 + 16 ,
 WirelessHeadStageAccDataRE2HS3 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 4 + 1 + 16 ,
 WirelessHeadStageGyroDataRE2HS4 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 6 + 0 + 16 ,
 WirelessHeadStageAccDataRE2HS4 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 6 + 1 + 16 ,
 WirelessHeadStageOptoStimCurrentRE1HS1 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 0 +
 0 + 32 ,
 WirelessHeadStageReservedARE1HS1 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 0 + 1 + 32
 ,
 WirelessHeadStageOptoStimCurrentRE1HS2 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 2 +
 0 + 32 ,
 WirelessHeadStageReservedARE1HS2 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 2 + 1 + 32
 ,
 WirelessHeadStageOptoStimCurrentRE1HS3 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 4 +
 0 + 32 ,
 WirelessHeadStageReservedARE1HS3 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 4 + 1 + 32
 ,
 WirelessHeadStageOptoStimCurrentRE1HS4 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 6 +
 0 + 32 ,
 WirelessHeadStageReservedARE1HS4 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 6 + 1 + 32
 ,
 WirelessHeadStageOptoStimCurrentRE2HS1 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 0 +
 0 + 32 ,
 WirelessHeadStageReservedARE2HS1 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 0 + 1 + 32
 ,
 WirelessHeadStageOptoStimCurrentRE2HS2 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 2 +
 0 + 32 ,
 WirelessHeadStageReservedARE2HS2 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 2 + 1 + 32
 ,
 WirelessHeadStageOptoStimCurrentRE2HS3 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 4 +
 0 + 32 ,
 WirelessHeadStageReservedARE2HS3 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 4 + 1 + 32
 ,
 WirelessHeadStageOptoStimCurrentRE2HS4 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 6 +
 0 + 32 ,
 WirelessHeadStageReservedARE2HS4 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 6 + 1 + 32
 ,
 WirelessHeadStageReservedBRE1HS1 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 0 + 0 + 48
 ,
 WirelessHeadStageReservedCRE1HS1 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 0 + 1 + 48
 ,
 WirelessHeadStageReservedBRE1HS2 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 2 + 0 + 48
 ,
 WirelessHeadStageReservedCRE1HS2 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 2 + 1 + 48
 ,
 WirelessHeadStageReservedBRE1HS3 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 4 + 0 + 48
 ,
 WirelessHeadStageReservedCRE1HS3 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 4 + 1 + 48
 ,
 WirelessHeadStageReservedBRE1HS4 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 6 + 0 + 48
 ,
 WirelessHeadStageReservedCRE1HS4 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 6 + 1 + 48
 ,
 WirelessHeadStageReservedBRE2HS1 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 0 + 0 + 48

```
,
WirelessHeadStageReservedCRE2HS1 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 0 + 1 + 48
,
WirelessHeadStageReservedBRE2HS2 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 2 + 0 + 48
,
WirelessHeadStageReservedCRE2HS2 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 2 + 1 + 48
,
WirelessHeadStageReservedBRE2HS3 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 4 + 0 + 48
,
WirelessHeadStageReservedCRE2HS3 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 4 + 1 + 48
,
WirelessHeadStageReservedBRE2HS4 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 6 + 0 + 48
,
WirelessHeadStageReservedCRE2HS4 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 6 + 1 + 48
,
IFDigChannelsGroup = INTERFACEDIGITALCHANNELSGROUP ,
AudioTestChannelGroup = AUDIOTESTCHANNELGROUP ,
PacketFrameContextGroup = PACKETFRAMECONTEXTGROUP }
```

Enumerates the W2100 Channel Groups of Datastream

- enum class SCUDacqGroupChannelEnumNet {
 SCU1ElectrodeGroupHS1 = 0x00 ,
 SCU1ElectrodeGroupHS2 = 0x01 ,
 SCU1ElectrodeGroupHS3 = 0x02 ,
 SCU1ElectrodeGroupHS4 = 0x03 ,
 SCU2ElectrodeGroupHS1 = 0x08 ,
 SCU2ElectrodeGroupHS2 = 0x09 ,
 SCU2ElectrodeGroupHS3 = 0x0A ,
 SCU2ElectrodeGroupHS4 = 0x0B ,
 InterfaceADCGroup = INTERFACEANALOGCHANNELSGROUP ,
 STG1DACSignalGroup = 0x40 ,
 STG2DACSignalGroup = 0x41 ,
 DSPAnalogGroup = DSPDATACHANNELSGROUP ,
 DSPDigitalGroup = 0xA0 ,
 IFDigChannelsGroup = INTERFACEDIGITALCHANNELSGROUP ,
 STG1TriggerStatusGroup = 0x90 ,
 STG1SidebandsGroup = 0x91 ,
 STG2TriggerStatusGroup = 0x98 ,
 STG2SidebandsGroup = 0x99 ,
 AudioTestChannelGroup = AUDIOTESTCHANNELGROUP ,
 PacketFrameContextGroup = PACKETFRAMECONTEXTGROUP }

Enumerates the SCU Channel Groups of Datastream

- enum class MEA2100_256DacqGroupChannelEnumNet {
 HS1ElectrodeGroup = 0x00 ,
 HS2ElectrodeGroup = 0x08 ,
 InterfaceADCGroup = INTERFACEANALOGCHANNELSGROUP ,
 STG1DACSignalGroup = 0x40 ,
 STG2DACSignalGroup = 0x41 ,
 DSPAnalogGroup = DSPDATACHANNELSGROUP ,
 DSPDigitalGroup = 0xA0 ,
 IFDigChannelsGroup = INTERFACEDIGITALCHANNELSGROUP ,
 STG1TriggerStatusGroup = 0x90 ,
 STG1SidebandsGroup = 0x91 ,
 STG2TriggerStatusGroup = 0x98 ,
 STG2SidebandsGroup = 0x99 ,
 AudioTestChannelGroup = AUDIOTESTCHANNELGROUP ,
 PacketFrameContextGroup = PACKETFRAMECONTEXTGROUP }

Enumerates the MEA2100-256 Channel Groups of Datastream

- enum class [DacqMeaGroupTypeEnumNet](#) {
[AnalogGroup](#) = ANALOG_GROUP ,
[DigitalGroup](#) = DIGITAL_GROUP ,
[FrameContextGroup](#) = FRAME_CONTEXT_GROUP }

Enumerations of CMOS MEA Groups to detect whether it is an Analog, Digital or Frame Context Group

- enum class [CMOSMeaValueUnitEnumNet](#) {
[NoUnit](#) = 0x00 ,
[NanoVolt](#) = 0x11 ,
[PicoAmpere](#) = 0x21 ,
[NanoAmpere](#) = 0x22 ,
[MicroAmpere](#) = 0x23 ,
[MilliDegreeCelsius](#) = 0x31 }

Enumerations of CMOS MEA Units of Values in Data stream

- enum class [CMOSMeaInterfaceADCEnumNet](#) {
[IFChannel1](#) = 0x01 ,
[IFChannel2](#) = 0x02 ,
[IFChannel3](#) = 0x04 ,
[IFChannel4](#) = 0x08 ,
[IFChannel5](#) = 0x10 ,
[IFChannel6](#) = 0x20 ,
[IFChannel7](#) = 0x40 ,
[IFChannel8](#) = 0x80 }

Enumerations of CMOS MEA IF Analog Channels Group Bitmask

- enum class [CMOSMeaHeadstage1NCBathCurrentEnumNet](#) { [NCBathCurrent](#) = 0x01 }

Enumerations of CMOS MEA HS Current Monitoring Channels Group Bitmask

- enum class [CMOSMeaHeadstage1NCCol2CurrentEnumNet](#) { [NCCol2Current](#) = 0x01 }

Enumerations of CMOS MEA HS Current Monitoring Channels Group Bitmask

- enum class [CMOSMeaHeadstage1NChipTempEnumNet](#) { [NChipTemperature](#) = 0x01 }

Enumerations of CMOS MEA HS Temperature Monitoring Channels Group Bitmask

- enum class [CMOSMeaSTG1DACSignalEnumNet](#) {
[DAC1Channel](#) = 0x01 ,
[DAC2Channel](#) = 0x02 ,
[DAC3Channel](#) = 0x04 ,
[DAC4Channel](#) = 0x08 }

Enumerations of CMOS MEA DAC Stimulation Channels Group Bitmask

- enum class [CMOSMeaIFDigChannelEnumNet](#) {
[DigitalMux](#) = 0x01 ,
[DigitalInPort](#) = 0x02 ,
[DigitalOutReg](#) = 0x04 ,
[FeedbackReg](#) = 0x08 ,
[DigitalReg](#) = 0x10 ,
[AuxPort](#) = 0x20 }

Enumerations of CMOS MEA IF Digital Channels Group Bitmask

- enum class [CMOSMeaHS1SidebandEnumNet](#) {
[SBSVector1](#) = 0x01 ,
[SBSVector2](#) = 0x02 ,
[SBSVector3](#) = 0x04 ,
[SBSVector4](#) = 0x08 }

Enumerations of CMOS MEA HS STG Sideband Channels Group Bitmask

- enum class [CMOSMeaHS1TriggerStatusEnumNet](#) {
[TriggerStatus1](#) = 0x01 ,
[TriggerStatus2](#) = 0x02 ,
[TriggerStatus3](#) = 0x04 ,
[TriggerStatus4](#) = 0x08 }

Enumerations of CMOS MEA HS STG Trigger Status Channels Group Bitmask

- enum class [AnalogUnitEnumNet](#) {
[Unknown](#) ,
[Volt](#) ,
[Ampere](#) ,
[Kelvin](#) }
 - enum class [CMOSMeaPacketFrameContextGroupEnumNet](#) {
[SOAndCTRLword](#) = 0x01 ,
[ChecksumAndPacketCounter](#) = 0x02 ,
[Timestamp](#) = 0x04 ,
[EOAndCRC](#) = 0x08 }
- Enumerations of CMOS MEA HS STG Trigger Status Channels Group Bitmask*
- enum class [CMOSMeaBathModeEnumNet](#) {
[Ground](#) = 0x02 ,
[Stimulation](#) = 0x01 ,
[CurrentMeasure](#) = 0x00 }
- Enumerations of CMOS MEA Bath Mode*
- enum class [PatchServAdcModeEnumNet](#) {
[Normal](#) = 0 ,
[CatchAmp](#) = 1 }
 - enum class [RoboCurrentModeEnumNet](#) {
[Off](#) = ROBO_CURRENT_OFFMODE ,
[Break](#) = ROBO_CURRENT_BREAKMODE ,
[Standby](#) = ROBO_CURRENT_STANDBYMODE ,
[Reference](#) = ROBO_CURRENT_REFERENCEMODE ,
[Movement](#) = ROBO_CURRENT_MOVEMENTMODE }
 - enum class [TeerClampModeEnumNet](#) {
[ClampModeVoltage](#) = 0 ,
[ClampModeCurrent](#) = 1 ,
[ClampModeOpen](#) = 2 ,
[ClampModeInternalCalibration](#) = 3 }
 - enum class [TeerWaveformEnumNet](#) {
[Rectangle](#) = 0 ,
[Sine](#) = 1 }
 - enum class [UssingClampModeEnumNet](#) {
[VoltageClamp](#) = 1 ,
[CurrentClamp](#) = 2 ,
[OpenClamp](#) = 3 ,
[Standby](#) = 4 ,
[ElectrodeOffset](#) = 5 }
 - enum class [UssingUnitEnumNet](#) {
[Volt](#) = 0 ,
[Ampere](#) = 1 ,
[State](#) = 2 }
 - enum class [PlateClampEnumNet](#) {
[Close](#) = 0 ,
[Open](#) = 1 ,
[Stop](#) = 2 }
 - enum class [PlateClampLockEnumNet](#) {
[Lock](#) = 0 ,
[Unlock](#) = 1 }
 - enum class [MultiwellPlateTypeEnumNet](#) {
[Plate_Dummy](#) = HS_PLATETYPE_0 ,
[Plate_24W700_100FMA](#) = 1 ,
[Plate_24W030MGA](#) = 2 ,
[Plate_72W500_100PMA](#) = 3 ,
[Plate_72W500_100FMA](#) = 5 ,
[Plate_24W700_100FMB](#) = HS_PLATETYPE_6 ,

```

Plate_96W700_100FMA = HS_PLATETYPE_7 ,
Plate_96W300_80_1152FMA = HS_PLATETYPE_33 ,
Plate_96W400_80_1152FMB = HS_PLATETYPE_36 ,
Plate_24W300_30_1152GBA = HS_PLATETYPE_40 ,
Plate_24W700_100FMC = HS_PLATETYPE_44 ,
Plate_96W700_100FMB = HS_PLATETYPE_48 ,
Plate_96W700_100GBC = HS_PLATETYPE_49 ,
Plate_96W700_100GBD = HS_PLATETYPE_51 ,
Plate_24W700_100PBB = HS_PLATETYPE_60 ,
Plate_96W700_100PBB = HS_PLATETYPE_61 ,
Plate_ICB8_24W700_100P8A = HS_PLATETYPE_80 ,
Plate_ICB8_96W700_100P8A = HS_PLATETYPE_81 ,
Plate_Dummy_126 = HS_PLATETYPE_126 ,
Plate_24W300_30GMA = HS_PLATETYPE_193 ,
Plate_96W700_100GMA = 194 ,
Plate_24W300_30GBA = HS_PLATETYPE_195 ,
Plate_96W700_100GBA = HS_PLATETYPE_224 ,
Plate_24W300_30GBB = HS_PLATETYPE_232 ,
Plate_96W700_100GBB = HS_PLATETYPE_244 ,
No_Plate = 255 }
• enum class FpgaldEnumNet {
DeviceNotConnected = FPGA_ID_NOT_CONNECTED ,
Mea2100Interfaceboard = FPGA_ID_MEA2100_IF ,
Mea2100Headstage = FPGA_ID_MEA2100_HS ,
Mea2100STG = FPGA_ID_MEA2100_STG ,
MultiwellHeadstage = FPGA_ID_HS_MULTIWELL ,
MultiwellInterfaceboard = FPGA_ID_IF_MULTIWELL ,
TbsiDacqInterfaceboard = FPGA_ID_TBSI_DACQ_IF ,
TbsiDacqHeadstage = FPGA_ID_TBSI_DACQ_HS ,
CmosMealInterfaceboard = FPGA_ID_CMOS_MEA_IF ,
CmosMeaHeadstage = FPGA_ID_CMOS_MEA_HS ,
Mea2100MultiwellIFB2 = FPGA_ID_MEA2100_MW_IFB2 ,
Me2100Interfaceboard = FPGA_ID_ME2100_IFB ,
Me2100InvivoSignalCollectorUnit = FPGA_ID_ME2100_InvivoSCU ,
Me2100InvitroSignalCollectorUnit = FPGA_ID_ME2100_InvitroSCU ,
Me2100_32XilinxHeadstage = FPGA_ID_ME2100_32_XILINX_HS ,
Me2100_32PICiCE40Headstage = FPGA_ID_ME2100_32_PIC_ICE40_HS ,
Mea2100_256Interfaceboard = FPGA_ID_MEA2100_256_IF ,
Mea2100_256Headstage = FPGA_ID_MEA2100_256_HS ,
W2100Interfaceboard = FPGA_ID_W2100_IF ,
W2100WirelessReceiver = FPGA_ID_W2100_REC ,
W2100WirelessReceiverAnalog = FPGA_ID_W2100_REC_ANALOG ,
Mea2100Mini60PICiCE40Headstage = FPGA_ID_MEA2100MINI60_PIC_ICE40_HS ,
Mea2100BetaScreenHeadstage = FPGA_ID_MEA2100BETASCREEN_HS ,
Me2100UPA32Headstage = FPGA_ID_ME2100UPA32_HS ,
MultiwellMiniHeadstage = FPGA_ID_MULTIWELL_MINI_HS ,
Mea2100Mini120Headstage = FPGA_ID_MEA2100MINI120_HS ,
Mea2100Mini60ECP5Headstage = FPGA_ID_MEA2100MINI60_ECP5_HS ,
eCubeHeadstage = FPGA_ID_ECUBE_HS ,
Me2100Graphene16_32Headstage = FPGA_ID_ME2100_GRAPHENE_16_32_HS ,
GrapheneASICHeadstage = FPGA_ID_GRAPHENE_ASIC_HS ,
WholeCellPatchHeadstage = FPGA_ID_WHOLE_CELL_PATCH_HS ,
InterfaceBoard2 = FPGA_ID_INTERFACEBOARD2 ,
W2100IFB2 = FPGA_ID_W2100_IFB2 ,
CmosmealIFB2 = FPGA_ID_CMOS_MEA_IFB2 ,
M4M_Dock = FPGA_ID_M4M_DOCK ,
M4M_ReUse = FPGA_ID_M4M_REUSE ,
Mea2100LiteHeadstage = FPGA_ID_MEA2100_LITE_HS ,

```

```

LIH30Interfaceboard = FPGA_ID_LIH30_USB_IF ,
LIH30ADCCtrl = FPGA_ID_LIH30_ADC_CTRL ,
UssingRail = FPGA_ID_USSING_RAIL ,
UssingChamber = FPGA_ID_USSING_CHAMBER ,
IFB2GoldenInterfaceboard = FPGA_ID_IFB2_GOLDEN ,
IFB30GoldenInterfaceboard = FPGA_ID_IFB30_GOLDEN ,
DeviceHasNoHeadstage = FPGA_ID_HAS_NO_HS }
• enum class HeadstageIdEnumNet {
DeviceNotConnected = FPGA_ID_NOT_CONNECTED ,
Mea2100 = FPGA_ID_MEA2100_HS ,
Multiwell = FPGA_ID_HS_MULTIWELL ,
TbsiDacq = FPGA_ID_TBSI_DACQ_HS ,
CmosMea = FPGA_ID_CMOS_MEA_HS ,
InvivoSignalCollectorUnit = FPGA_ID_ME2100_InvivoSCU ,
InvitroSignalCollectorUnit = FPGA_ID_ME2100_InvitroSCU ,
Mea2100_256 = FPGA_ID_MEA2100_256_HS ,
W2100WirelessReceiver = FPGA_ID_W2100_REC ,
W2100WirelessReceiverAnalog = FPGA_ID_W2100_REC_ANALOG ,
Mea2100_Lite = FPGA_ID_MEA2100_LITE_HS ,
LIH30ADCCtrl = FPGA_ID_LIH30_ADC_CTRL ,
DeviceHasNoHeadstage = FPGA_ID_HAS_NO_HS }
• enum class SCU_HeadstageIdEnumNet {
DeviceNotConnected = FPGA_ID_NOT_CONNECTED ,
Me2100_32Xilinx = FPGA_ID_ME2100_32_XILINX_HS ,
Me2100_32PICiCE40 = FPGA_ID_ME2100_32_PIC_ICE40_HS ,
Mea2100Mini60PICiCE40 = FPGA_ID_MEA2100MINI60_PIC_ICE40_HS ,
Mea2100BetaScreen = FPGA_ID_MEA2100BETASCREEN_HS ,
Me2100UPA32 = FPGA_ID_ME2100UPA32_HS ,
MultiwellMini = FPGA_ID_MULTIWELL_MINI_HS ,
Mea2100Mini120 = FPGA_ID_MEA2100MINI120_HS ,
Mea2100Mini60ECP5 = FPGA_ID_MEA2100MINI60_ECP5_HS ,
eCube = FPGA_ID_ECUBE_HS ,
Me2100Graphene16_32 = FPGA_ID_ME2100_GRAPHENE_16_32_HS ,
GrapheneASIC = FPGA_ID_GRAPHENE_ASIC_HS ,
WholeCellPatch = FPGA_ID_WHOLE_CELL_PATCH_HS ,
DeviceHasNoHeadstage = FPGA_ID_HAS_NO_HS }
• enum class UsbVendorIdEnumNet {
Unknown = -1 ,
None = 0 ,
Renesas = 0x1912 ,
ASMedia = 0x1b21 ,
Intel = 0x8086 }
• enum class FilterCalculationDirectionEnumNet {
DoubleToInt = 0 ,
IntToDouble = 1 }
• enum class FilterBandEnumNet {
Unknown = 0 ,
Lowpass = 1 ,
Highpass = 2 }
• enum class FilterFamilyEnumNet {
Unknown = 0 ,
Bessel = 1 ,
Butterworth = 2 ,
RC = 3 }
• enum class FilterTypeEnumNet {
Hardware = 0 ,
Software = 1 }

```


- enum class `FilterAttributeEnumNet` {
`PreCommaB` = 0 ,
`PostCommaB` = 1 ,
`CommaPositionB` = 2 ,
`PreCommaA` = 3 ,
`PostCommaA` = 4 ,
`CommaPositionA` = 5 }
- enum class `AnalogOut_DAC_Range_EnumNet` {
`PlusMinus2Comma5Volts` = 0 ,
`PlusMinus5Volts` = 1 ,
`PlusMinus10Volts` = 2 }
- enum class `PP_Pump_Mode_Type_EnumNet` {
`Manual` = 1 ,
`Digital` = 2 ,
`Analog` = 3 }
- enum class `MbcChargingModeEnumNet` {
`StorageCharge` = 0 ,
`FullCharge` = 1 }
- enum class `MbcRatedCapacityEnumNet` {
`rc30mAh` = 0 ,
`rc100mAh` = 1 ,
`rc200mAh` = 2 ,
`rc300mAh` = 3 ,
`rcGreater300mAh` = 4 }
- enum class `MbcChannelStateEnumNet` {
`csIdleNoBattery` = 0 ,
`csIdleChargeFinished` = 1 ,
`csCapacityTestPrecharge` = 2 ,
`csCapacityTestDischarge` = 3 ,
`csRefreshBattery` = 4 ,
`csCharge` = 5 ,
`csDischarge` = 6 ,
`csError` = 7 }
- enum class `PulseGenerator_Mode_EnumNet` {
`Off` = 0 ,
`AlwaysOn` = 1 ,
`Gated_Low_Active` = 2 ,
`Gated_High_Active` = 3 }
- enum class `LIH30_EPC10_Bus_EnumNet` {
`A` = 0 ,
`B` = 1 }
- enum class `W2100_Accel_Gyro_Select_EnumNet` {
`Off` = 0 ,
`GyroOnly` = 1 ,
`AccelOnly` = 2 ,
`Both` = 3 }
enumerates the accelerometer configuration on the W2100 device
- enum class `WvcValveModeEnumNet` {
`Manual` = `WVC_VALVE_MODE_MANUAL` ,
`Digital` = `WVC_VALVE_MODE_DIGITAL` ,
`Analog` = `WVC_VALVE_MODE_ANALOG` ,
`Table` = `WVC_VALVE_MODE_TABLE` }
enumerates Wvc valve mode
- enum class `WvcDisplayModeEnumNet` {
`Work` = `WVC_DISPLAY_MODE_WORK` ,
`PC` = `WVC_DISPLAY_MODE_PC` ,
`Settings` = `WVC_DISPLAY_MODE_SETTINGS` ,
`TouchTest` = `WVC_DISPLAY_MODE_TOUCH_TEST` }

- enumerates Wvc display mode*
- enum class [PortDirectionEnumNet](#) {
 - [Output](#) = 0 ,
 - [Input](#) = 1 }
 - enumerates a port direction*
- enum class [StimulationLayoutConfigurationEnumNet](#) {
 - [SingleWell](#) = 1 ,
 - [SixWell](#) = 2 ,
 - [NineWell](#) = 3 }
 - enumerates the layout configuration for the MEA2100-256 device*
- enum class [ReferenceElectrodeSwitchPositionEnumNet](#) {
 - [off](#) = 0 ,
 - [Ref8](#) = 1 ,
 - [Ref16](#) = 2 ,
 - [Ref24](#) = 3 ,
 - [Ref32](#) = 4 }
 - enumerates the possible positions of the reference electrode switch of the ME2100 device*
- enum class [ReferenceElectrodeModeEnumNet](#) {
 - [SubtractionOff](#) = 0 ,
 - [SubtractFromAllOther](#) = 1 ,
 - [SubtractFromReferenceElectrodeOnly](#) = 2 ,
 - [SubtractFromAll](#) = 3 }
 - enumerates the electrode subtraction modes*
- enum class [DigitalDatastreamEnableEnumNet](#) {
 - [None](#) = 0x0000 ,
 - [Mux](#) = 0x0001 ,
 - [MuxOtherDevice](#) = 0x0002 ,
 - [DigitalInReserverd](#) = 0x0004 ,
 - [DigitalIn](#) = 0x0008 ,
 - [DigitalOut](#) = 0x0010 ,
 - [DigitalOutReserved](#) = 0x0020 ,
 - [RegisterLow](#) = 0x0040 ,
 - [RegisterHigh](#) = 0x0080 ,
 - [FeedbackLow](#) = 0x0100 ,
 - [FeedbackHigh](#) = 0x0200 ,
 - [Aux](#) = 0x0400 ,
 - [PeriodicPulse](#) = 0x0800 ,
 - [DigOutStim](#) = 0x1000 ,
 - [Hs1Digital](#) = 0x00008000 ,
 - [Hs1Trigger](#) = 0x00010000 ,
 - [Hs1SidebandLow](#) = 0x00020000 ,
 - [Hs1SidebandHigh](#) = 0x00040000 ,
 - [Hs2Digital](#) = 0x00800000 ,
 - [Hs2Trigger](#) = 0x01000000 ,
 - [Hs2SidebandLow](#) = 0x02000000 ,
 - [Hs2SidebandHigh](#) = 0x04000000 }
 - enumerates the streams available as digital datastream*
- enum class [IoVoltageEnumNet](#) {
 - [Voltage_3V3](#) = IFB2_IO_VOLTAGE_3V3 ,
 - [Voltage_5V0](#) = IFB2_IO_VOLTAGE_5V0 }
 - enumerates the I/O Voltages available on the IFB2*
- enum class [EnSTG200x_STATUS](#) {
 - [OK](#) ,
 - [NOT_CONNECTED](#) ,
 - [DEVICE_NOT_FOUND](#) }

Functions

- public delegate void [OnMcsUsbDeviceState](#) ([usbSetupPacket_t](#)^ request)
- private delegate void [OnMcsUsbDeviceStateCallback](#) (IntPtr pThis, uint32_t size, IntPtr buffer)
- public delegate void [OnUpdateFirmwareStatusChange](#) (String^)
- public delegate void [OnUpdateFirmwareProgress](#) (int)
- public delegate void [OnDeviceArrivalRemoval](#) ([CMcsUsbListEntryNet](#)^ entry)
Delegate to show a device arrival or removal.
- public delegate void [OnStgPollStatus](#) (unsigned int status, [StgStatusNet](#)^ stgStatusNet, array< int >^ index_list)
- public delegate void [OnMwPollStatus](#) (unsigned int CurrentTemp, unsigned int PlateState, unsigned int SwitchState)
- public delegate void [RoboStatusEventDelegate](#) (array< unsigned char >^ buffer)
- public delegate void [OnStg200xDataHandler](#) (uint32_t trigger)
- public delegate void [OnStg200xErrorHandler](#) ()
- public delegate void [OnChannelData](#) ([CMcsUsbDacqNet](#)^ dacq, int CbHandle, int numFrames)
- public delegate void [OnError](#) (String^ msg, int action)

10.2.1 Enumeration Type Documentation

10.2.1.1 AdapterTypeEnumNet `enum AdapterTypeEnumNet [strong]`

Enumerates the adapter type of the MEA2100 device.

Enumerator

None	
MEA60	
MEA2x60	
MEA120	
MEA32	
MEA2x32	
Multiwell96	
WirelessTestAdapter	
MEA252	
MEA_2_252_2	
MEA_2_252_2_6Well	
MEA_2_252_2_9Well	
MEA_2_252_2_Test	
TBSI_5	
TBSI_15	
TBSI_31	
TBSI_63	
TBSI_127	
TBSI_Reserved	
Ci4600Intan	
Unknown	
NotApplicable	

10.2.1.2 AnalogOut_DAC_Range_EnumNet enum [AnalogOut_DAC_Range_EnumNet](#) [strong]

Enumerator

PlusMinus2Comma5Volts	
PlusMinus5Volts	
PlusMinus10Volts	

10.2.1.3 AnalogSourceEnumNet enum [AnalogSourceEnumNet](#) [strong]

Enumerates the analog source of the MEA2100 device.

Enumerator

AnalogSource_HS1	
AnalogSource_HS2	
AnalogSource_IF	

10.2.1.4 AnalogUnitEnumNet enum [AnalogUnitEnumNet](#) [strong]

Enumerator

Unknown	
Volt	
Ampere	
Kelvin	

10.2.1.5 CFirmwareDestinationNet enum [CFirmwareDestinationNet](#) [strong]

Enumerates the destination processor for the firmware.

Enumerator

FPGA_NORMAL	
DSP	The DSP.
USB	The USB controller.
MCU1	The DSP on the MEA2100 system.
MCSBUS1	
MCSBUS2	

Enumerator

MCSBUS3	
MCSBUS4	
MCSBUS5	
MCSBUS6	
MCSBUS7	
MCSBUS8	
MCSBUS9	
MCSBUS10	
MCSBUS11	
MCSBUS12	
MCSBUS13	
MCSBUS14	
MCSBUS15	
MCSBUS0	
BUSNUMBER0	
BUS0MCSBUS1	
BUS0MCSBUS2	
BUS0MCSBUS3	
BUS0MCSBUS4	
BUS0MCSBUS5	
BUS0MCSBUS6	
BUS0MCSBUS7	
BUS0MCSBUS8	
BUS0MCSBUS9	
BUS0MCSBUS10	
BUS0MCSBUS11	
BUS0MCSBUS12	
BUS0MCSBUS13	
BUS0MCSBUS14	
BUS0MCSBUS15	
BUS0MCSBUS0	
BUSNUMBER1	
BUS1MCSBUS1	
BUS1MCSBUS2	
BUS1MCSBUS3	
BUS1MCSBUS4	
BUS1MCSBUS5	
BUS1MCSBUS6	
BUS1MCSBUS7	
BUS1MCSBUS8	
BUS1MCSBUS9	
BUS1MCSBUS10	
BUS1MCSBUS11	
BUS1MCSBUS12	
BUS1MCSBUS13	
BUS1MCSBUS14	
BUS1MCSBUS15	
BUS1MCSBUS0	
BUSNUMBER2	
BUS2MCSBUS1	

Enumerator

BUS2MCSBUS2	
BUS2MCSBUS3	
BUS2MCSBUS4	
BUS2MCSBUS5	
BUS2MCSBUS6	
BUS2MCSBUS7	
BUS2MCSBUS8	
BUS2MCSBUS9	
BUS2MCSBUS10	
BUS2MCSBUS11	
BUS2MCSBUS12	
BUS2MCSBUS13	
BUS2MCSBUS14	
BUS2MCSBUS15	
BUS2MCSBUS0	
PIC	
PIC2	
PIC3	
PIC4	
PIC5	
PIC6	
PIC7	
PIC8	
PIC9	
PIC10	
PIC11	
PIC12	
ChannelPIC	
Bootstrap	
BootstrapOtherCypress	
ALTERA	
FPGA2	
FPGA3	
FPGA4	
FPGA5	
FPGA6	
FPGA7	
FPGA8	
FPGA9	
FPGA10	
FPGA11	
FPGA12	
FPGA13	
FPGA14	
FPGA15	
FPGA16	
FPGA_GOLD	
ALTERA_GOLD	
FPGA2_GOLD	
FPGA3_GOLD	

Enumerator

FPGA4_GOLD	
FPGA5_GOLD	
FPGA6_GOLD	
FPGA7_GOLD	
FPGA8_GOLD	
FPGA9_GOLD	
FPGA10_GOLD	
FPGA11_GOLD	
FPGA12_GOLD	
FPGA13_GOLD	
FPGA14_GOLD	
FPGA15_GOLD	
FPGA16_GOLD	
FPGA_BASE	
ALTERA_BASE	
FPGA2_BASE	
FPGA3_BASE	
FPGA4_BASE	
FPGA5_BASE	
FPGA6_BASE	
FPGA7_BASE	
FPGA8_BASE	
FPGA9_BASE	
FPGA10_BASE	
FPGA11_BASE	
FPGA12_BASE	
FPGA13_BASE	
FPGA14_BASE	
FPGA15_BASE	
FPGA16_BASE	
FPGA_BOOTSTRAP	
ALTERA_BOOTSTRAP	
DEST_TARGET1	
DEST_TARGET2	
DEST_TARGET3	
DEST_TARGET4	
DEST_TARGET5	
DEST_TARGET6	
DEST_TARGET7	
DEST_TARGET8	
DEST_TARGET9	
DEST_TARGET10	
DEST_TARGET11	
DEST_TARGET12	
DEST_TARGET13	
DEST_TARGET14	
DEST_TARGET15	
DEST_TARGET_MASK	
DEST_FX3_TARGET_MASK	

Enumerator

ALTERA_TARGET1	
ALTERA_TARGET2	
ALTERA_TARGET3	
USB_TARGET1	
USB_TARGET2	
USB_TARGET3	
UnknownDest	

10.2.1.6 ChannelBlockTypeNet enum [ChannelBlockTypeNet](#) [strong]

Enumerates the type of ChannelBlock functions.

Enumerator

OneHandleOneQueue	
OneHandleMultipleQueues	
MultipleHandlesOneQueues	

10.2.1.7 CMOSMEA5000DigitalSourceEnumNet enum [CMOSMEA5000DigitalSourceEnumNet](#) [strong]

Enumerates the digital source of the CMOSMEA5000 device.

Enumerator

DigitalInOfOutPort	
DigitalIn	
DigitalPulse	
Feedback	
AuxIn	
Zero	
One	
HS1Trigger1Status	
HS1Trigger2Status	
HS1Trigger3Status	
HS1Sideband1	
HS1Sideband2	
HS1Sideband3	
PulseGenerator	
DigitalOutStimulator	
DigitalData	
DeviceRunStatus	
LastPosition	

10.2.1.8 CMOSMeaBathModeEnumNet enum [CMOSMeaBathModeEnumNet](#) [strong]

Enumerations of CMOS MEA Bath Mode

Enumerator

Ground	
Stimulation	
CurrentMeasure	

10.2.1.9 CMOSMeaHeadstage1NCBathCurrentEnumNet enum [CMOSMeaHeadstage1NCBathCurrentEnumNet](#) [strong]

Enumerations of CMOS MEA HS Current Monitoring Channels Group Bitmask

Enumerator

NCBathCurrent	
---------------	--

10.2.1.10 CMOSMeaHeadstage1NCCol2CurrentEnumNet enum [CMOSMeaHeadstage1NCCol2CurrentEnumNet](#) [strong]

Enumerations of CMOS MEA HS Current Monitoring Channels Group Bitmask

Enumerator

NCCol2Current	
---------------	--

10.2.1.11 CMOSMeaHeadstage1NChipTempEnumNet enum [CMOSMeaHeadstage1NChipTempEnumNet](#) [strong]

Enumerations of CMOS MEA HS Temperature Monitoring Channels Group Bitmask

Enumerator

NChipTemperature	
------------------	--

10.2.1.12 CMOSMeaHS1SidebandEnumNet enum [CMOSMeaHS1SidebandEnumNet](#) [strong]

Enumerations of CMOS MEA HS STG Sideband Channels Group Bitmask

Enumerator

SBSVector1	
SBSVector2	
SBSVector3	
SBSVector4	

10.2.1.13 CMOSMeaHS1TriggerStatusEnumNet enum CMOSMeaHS1TriggerStatusEnumNet [strong]

Enumerations of CMOS MEA HS STG Trigger Status Channels Group Bitmask

Enumerator

TriggerStatus1	
TriggerStatus2	
TriggerStatus3	
TriggerStatus4	

10.2.1.14 CMOSMeaIFDigChannelEnumNet enum CMOSMeaIFDigChannelEnumNet [strong]

Enumerations of CMOS MEA IF Digital Channels Group Bitmask

Enumerator

DigitalMux	
DigitalInPort	
DigitalOutReg	
FeedbackReg	
DigitalReg	
AuxPort	

10.2.1.15 CMOSMeaInterfaceADCEnumNet enum CMOSMeaInterfaceADCEnumNet [strong]

Enumerations of CMOS MEA IF Analog Channels Group Bitmask

Enumerator

IFChannel1	
IFChannel2	
IFChannel3	

Enumerator

IFChannel4	
IFChannel5	
IFChannel6	
IFChannel7	
IFChannel8	

10.2.1.16 CMOSMeaPacketFrameContextGroupEnumNet enum [CMOSMeaPacketFrameContextGroupEnumNet](#) [strong]

Enumerations of CMOS MEA HS STG Trigger Status Channels Group Bitmask

Enumerator

SOFAndCTRLword	
ChecksumAndPacketCounter	
Timestamp	
EOFAndCRC	

10.2.1.17 CMOSMeaSTG1DACSignalEnumNet enum [CMOSMeaSTG1DACSignalEnumNet](#) [strong]

Enumerations of CMOS MEA DAC Stimulation Channels Group Bitmask

Enumerator

DAC1Channel	
DAC2Channel	
DAC3Channel	
DAC4Channel	

10.2.1.18 CMOSMeaValueUnitEnumNet enum [CMOSMeaValueUnitEnumNet](#) [strong]

Enumerations of CMOS MEA Units of Values in Data stream

Enumerator

NoUnit	
NanoVolt	
PicoAmpere	
NanoAmpere	
MicroAmpere	
MilliDegreeCelsius	

10.2.1.19 DacqGroupChannelEnumNet enum [DacqGroupChannelEnumNet](#) [strong]

Enumerates the Channel Groups of Datastream

Enumerator

HeadstageElectrodeGroup	
InterfaceADCGroup	
DSPDataGroup	
Headstage1NCBathCurrentGroup	
Headstage1NCCol2CurrentGroup	
Headstage1NChipTempGroup	
STG1DACSignalGroup	
LIH30ADCModulesGroup	
IFDigChannelsGroup	
STG1SidebandsGroup	
STG1TriggerStatusGroup	
DACQ1DigitalGroup	
AudioTestChannelGroup	
PacketFrameContextGroup	

10.2.1.20 DacqMeaGroupTypeEnumNet enum [DacqMeaGroupTypeEnumNet](#) [strong]

Enumerations of CMOS MEA Groups to detect wether it is an Analog, Digital or Frame Context Group

Enumerator

AnalogGroup	
DigitalGroup	
FrameContextGroup	

10.2.1.21 DataModeEnumNet enum [DataModeEnumNet](#) [strong]

Enumerates the data mode of the device, either 16, 24 or 32 bit, can be signed or unsigned.

Enumerator

Unsigned_16bit	
Unsigned_24bit	
Unsigned_32bit	
Signed_16bit	
Signed_24bit	
Signed_32bit	

10.2.1.22 DeviceEnumNet enum DeviceEnumNet [strong]

Enumerates the group of MCS devices to connect to.

Enumerator

MCS_DEVICE_ANY	To connect to any MCS device.
MCS_GENERIC_DEVELOPMENT_DEVICE	Please use this only for MCS internal development.
MCS_DEVICE_USB	To connect to any MCS USB device.
MCS_MCCARD_DEVICE	Connect to an MC_Card.
MCS_STG_DEVICE	Connect to an MCS device with STG capability.
MCS_MC_STIMULUS_DEVICE	Devices which should be accessible from MC_Stimulus.
MCS_MEASUSB_DEVICE	Connect to an MCS MeaUsb device.
MCS_MEA_DEVICE	Connect to an MCS MeaUsb device.
MCS_OCTOPOT_DEVICE	Connect to an MCS Octopot device.
MCS_TERSENS_DEVICE	Connect to an MCS Tersens device.
MCS_PGA_DEVICE	Connect to an MCS PGA device.
MCS_PCX_DEVICE	Connect to an MCS PCX device.
MCS_TCX_DEVICE	Connect to an MCS TCX device.
MCS_FCX_DEVICE	Connect to an MCS FCX device.
MCS_RETINA_LED_DEVICE	Connect to an MCS RetineLed device.
MCS_MEA_SWITCH_DEVICE	Connect to an MCS Mea Switch device.
MCS_MEA_IMPEDANCE_DEVICE	Connect to an MCS Mea Impedance device.
MCS_CHANNELTEST_DEVICE	Connect to an MCS ChannelTest device.
MCS_SW2TO64_DEVICE	Connect to an MCS SW2TO64 device.
MCS_RETINA_AMS_DONGLE	Connect to an MCS Retina AMS Dongle (Radio device)
MCS_PATHIDENT_DEVICE	Connect to an MCS Pathident device.
MCS_ROBO_DEVICE	Connect to an MCS Robo Platform device.
MCS_ROBOOCYTE2_DEVICE	Connect to an MCS Roboocyte2 device.
MCS_ROBOINJECT_DEVICE	Connect to an MCS RoboInject device.
MCS_HICLAMP_DEVICE	Connect to an MCS HiClamp device.
MCS_PATCHSERVER_DEVICE	Connect to an MCS PatchServer device.
MCS_ENCAPSULATOR_DEVICE	Connect to an MCS Encapsulator device.
MCS_MEASURETABLE_DEVICE	Connect to an MCS Encapsulator device.
MCS_FYI_DEVICE	Connect to an MCS FYI device.
MCS_HLA_DEVICE	Connect to an MCS HLA device.
MCS_PPS_DEVICE	Connect to an MCS PPS device.
MCS_PPS5_DEVICE	Connect to an MCS PPS5 device.
MCS_OKUVISION_STIMULATOR_DEVICE	Connect to an Okuvision Stimulator device.
MCS_NF_GEN_DEVICE	Connect to an MCS NF-Gen device.
MCS_SAFEIS_DEVICE	Connect to an MCS SafeIS device.
MCS_PERISTALTIC_PUMP_DEVICE	Connect to an MCS PeristalticPump device.
MCS_EXTERN_BC_TESTER_DEVICE	Connect to an ExternBCTester device.
MCS_EXTERN_D_TESTER_DEVICE	Connect to an ExternDTester device.
MCS_SOFTWARE_DONGLE_DEVICE	Connect to an Software Dongle device.
MCS_MEA_CLEAN_DEVICE	Connect to a MEA Clean device.
MCS_MEA_COAT_DEVICE	Connect to a MEA Clean device.
MCS_SMARTIMPLANT_DEVICE	Connect to a SmartImplant device.

Enumerator

MCS_MBC08_DEVICE	Connect to a MultiBatteryCharger device.
MCS_PEDOTER_DEVICE	Connect to a Pedoter device.
MCS_PPC_DEVICE	Connect to a PPC device.
WARNER_VALVE_CONTROL_DEVICE	Connect to a Warner Valve Control device.
WARNER_USSING_DEVICE	Connect to a Warner Valve Control device.
HEKA_LIH3_DEVICE	Connect to a HEKA LIH3 device.
ALA_VC3_DEVICE	Connect to an ALA VC3 Valve Commander.
MCS_DEVICE_USB_CYPRESS	Connect to a Cypress USB device.
MCS_DEVICE_NONE	

10.2.1.23 DigitalDataStreamEnableEnumNet `enum DigitalDataStreamEnableEnumNet` [strong]

enumerates the streams available as digital datastream

Enumerator

None	No digital datastream.
Mux	16 bits from the standard MUX datastream.
MuxOtherDevice	The 16 bits of the standard MUX datastream used by the other virtual device.
DigitalInReserverd	The lower 16 bits of the Digital IN port, these ports are on the device by default used as Digital OUT, thus not available as Digital IN.
DigitalIn	The upper 16 bits of the Digital IN port, use when Digital IN datastream is needed.
DigitalOut	The lower 16 bits of the Digital OUT port, use when Digital OUT datastream is needed.
DigitalOutReserved	The upper 16 bits of the Digital OUT port, these ports are on the device by default used as Digital IN, thus not available as Digital OUT.
RegisterLow	The lower 16 bits of the digital register.
RegisterHigh	The upper 16 bits of the digital register.
FeedbackLow	The lower 16 bits of the realtime feedback register.
FeedbackHigh	The upper 16 bits of the realtime feedback register.
Aux	The 2 bits of the AUX port.
PeriodicPulse	The 8 bits of the Periodic Pulse Generator (Video-Sync).
DigOutStim	The 16 bits of the Digital Out Stimulator.
Hs1Digital	Headstage 1 digital signals.
Hs1Trigger	Headstage 1 trigger signals.
Hs1SidebandLow	Headstage 1 lower 16 bits of sideband data.
Hs1SidebandHigh	Headstage 1 upper 16 bits of sideband data.
Hs2Digital	Headstage 2 digital signals.
Hs2Trigger	Headstage 2 trigger signals.
Hs2SidebandLow	Headstage 2 lower 16 bits of sideband data.
Hs2SidebandHigh	Headstage 2 upper 16 bits of sideband data.

10.2.1.24 DigitalStimulatorTriggerEventEnumNet `enum DigitalStimulatorTriggerEventEnumNet` [strong]

Enumerates start/stop event for DigOut/DigStim trigger. /summary>

Enumerator

Start	
Stop	

10.2.1.25 DigitalStimulatorTriggerSlopeEnumNet enum [DigitalStimulatorTriggerSlopeEnumNet](#) [strong]

Enumerates start/stop conditions for DigOut/DigStim trigger. /summary>

Enumerator

Falling	
Rising	

10.2.1.26 DigitalTargetEnumNet enum [DigitalTargetEnumNet](#) [strong]

Enumerates the Digital Targets for Digital Sources

Enumerator

Digout	
Digstream	
DacqTrigger	
StgTrigger	
StgListModeTrigger	
DigOutStimulatorStartTrigger	
DigOutStimulatorStopTrigger	
DigStreamToReceiver	

10.2.1.27 ElectrodeDacMuxEnumNet enum [ElectrodeDacMuxEnumNet](#) [strong]

Enumerates the setting of the Stimulation DAC Multiplexer.

Enumerator

Ground	Connect the electrode to Ground while stimulation is active.
Stg1	Connect the electrode tp STG 1 while stimulation is active.
Stg2	Connect the electrode tp STG 2 while stimulation is active.
Stg3	Connect the electrode tp STG 3 while stimulation is active.

10.2.1.28 ElectrodeModeEnumNet enum [ElectrodeModeEnumNet](#) [strong]

Enumerates the mode of each electrode, can be automatic or manual. In automatic mode, the blanking of the electrode is controlled by the sideband signal, in manual mode, the stimulation configuration is independant of the sideband signal.

Enumerator

emAutomatic	
emManual	

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

Enumerator

unknown	
nMos16LV	
nMos32LV	
nMos36LN	
nMos64LN	

10.2.1.30 EnSTG200x_STATUS enum [EnSTG200x_STATUS](#) [strong]

Enumerator

OK	
NOT_CONNECTED	
DEVICE_NOT_FOUND	

10.2.1.31 FilterAttributeEnumNet enum [FilterAttributeEnumNet](#) [strong]

Enumerator

PreCommaB	
PostCommaB	
CommaPositionB	
PreCommaA	
PostCommaA	
CommaPositionA	

10.2.1.32 FilterBandEnumNet enum `FilterBandEnumNet` [strong]

Enumerator

Unknown	
Lowpass	
Highpass	

10.2.1.33 FilterCalculationDirectionEnumNet enum `FilterCalculationDirectionEnumNet` [strong]

Enumerator

DoubleToInt	
IntToDouble	

10.2.1.34 FilterFamilyEnumNet enum `FilterFamilyEnumNet` [strong]

Enumerator

Unknown	
Bessel	
Butterworth	
RC	

10.2.1.35 FilterTypeEnumNet enum `FilterTypeEnumNet` [strong]

Enumerator

Hardware	
Software	

10.2.1.36 FpgaldEnumNet enum `FpgaIdEnumNet` [strong]

Enumerator

DeviceNotConnected	
--------------------	--

Enumerator

Mea2100Interfaceboard	
Mea2100Headstage	
Mea2100STG	
MultiwellHeadstage	
MultiwellInterfaceboard	
TbsiDacqInterfaceboard	
TbsiDacqHeadstage	
CmosMealInterfaceboard	
CmosMealHeadstage	
Mea2100MultiwellIFB2	
Me2100Interfaceboard	
Me2100InvivoSignalCollectorUnit	
Me2100InvitroSignalCollectorUnit	
Me2100_32XilinxHeadstage	
Me2100_32PICiCE40Headstage	
Mea2100_256Interfaceboard	
Mea2100_256Headstage	
W2100Interfaceboard	
W2100WirelessReceiver	
W2100WirelessReceiverAnalog	
Mea2100Mini60PICiCE40Headstage	
Mea2100BetaScreenHeadstage	
Me2100UPA32Headstage	
MultiwellMiniHeadstage	
Mea2100Mini120Headstage	
Mea2100Mini60ECP5Headstage	
eCubeHeadstage	
Me2100Graphene16_32Headstage	
GrapheneASICHeadstage	
WholeCellPatchHeadstage	
InterfaceBoard2	
W2100IFB2	
CmosmealFB2	
M4M_Dock	
M4M_ReUse	
Mea2100LiteHeadstage	
LIH30Interfaceboard	
LIH30ADCCtrl	
UssingRail	
UssingChamber	
IFB2GoldenInterfaceboard	
IFB30GoldenInterfaceboard	
DeviceHasNoHeadstage	

10.2.1.37 HeadstageIdEnumNet

enum [HeadstageIdEnumNet](#) [strong]

Enumerator

DeviceNotConnected	
Mea2100	
Multiwell	
TbsiDacq	
CmosMea	
InvivoSignalCollectorUnit	
InvitroSignalCollectorUnit	
Mea2100_256	
W2100WirelessReceiver	
W2100WirelessReceiverAnalog	
Mea2100_Lite	
LIH30ADCCtrl	
DeviceHasNoHeadstage	

10.2.1.38 IoVoltageEnumNet enum [IoVoltageEnumNet](#) [strong]

enumerates the I/O Voltages available on the IFB2

Enumerator

Voltage_3V3	
Voltage_5V0	

10.2.1.39 LIH30_EPC10_Bus_EnumNet enum [LIH30_EPC10_Bus_EnumNet](#) [strong]

Enumerator

A	
B	

10.2.1.40 MbcChannelStateEnumNet enum [MbcChannelStateEnumNet](#) [strong]

Enumerator

csIdleNoBattery	
csIdleChargeFinished	
csCapacityTestPrecharge	
csCapacityTestDischarge	
csRefreshBattery	
csCharge	

Enumerator

csDischarge	
csError	

10.2.1.41 MbcChargingModeEnumNet enum [MbcChargingModeEnumNet](#) [strong]

Enumerator

StorageCharge	
FullCharge	

10.2.1.42 MbcRatedCapacityEnumNet enum [MbcRatedCapacityEnumNet](#) [strong]

Enumerator

rc30mAh	
rc100mAh	
rc200mAh	
rc300mAh	
rcGreater300mAh	

10.2.1.43 McsBusTypeEnumNet enum [McsBusTypeEnumNet](#) [strong]

Enumerates the bus to use, either USB, PCI or any

Enumerator

MCS_ANY_BUS	
MCS_UNDEFINED_BUS	
MCS_USB_BUS	
MCS_PCI_BUS	

10.2.1.44 McsUsbSpeedEnumNet enum [McsUsbSpeedEnumNet](#) [strong]

Enumerates the current connection speed of the device

Enumerator

LowSpeed	
FullSpeed	
HighSpeed	
SuperSpeed	
UnknownSpeed	

10.2.1.45 MEA2100_256DacqGroupChannelEnumNet enum [MEA2100_256DacqGroupChannelEnumNet](#) [strong]

Enumerates the MEA2100-256 Channel Groups of Datastream

Enumerator

HS1ElectrodeGroup	
HS2ElectrodeGroup	
InterfaceADCGroup	
STG1DACSignalGroup	
STG2DACSignalGroup	
DSPAnalogGroup	
DSPDigitalGroup	
IFDigChannelsGroup	
STG1TriggerStatusGroup	
STG1SidebandsGroup	
STG2TriggerStatusGroup	
STG2SidebandsGroup	
AudioTestChannelGroup	
PacketFrameContextGroup	

10.2.1.46 MEA2100_256DigitalSourceEnumNet enum [MEA2100_256DigitalSourceEnumNet](#) [strong]

Enumerates the digital source of the MEA2100-256 device.

Enumerator

DigitalInOfOutPort	
DigitalIn	
DigitalPulse	
Feedback	
AuxIn	
Zero	
One	
DeviceRunStatus	
PulseGenerator	

Enumerator

DigitalOutStimulator	
DigitalData	
HS1Trigger1Status	
HS1Trigger2Status	
HS1Trigger3Status	
HS1Trigger4Status	
HS1Trigger5Status	
HS1Trigger6Status	
HS1Trigger7Status	
HS1Trigger8Status	
HS1Trigger9Status	
HS1Trigger10Status	
HS1Trigger11Status	
HS1Trigger12Status	
HS1Trigger13Status	
HS1Trigger14Status	
HS1Trigger15Status	
HS1Trigger16Status	
HS1Trigger17Status	
HS1Trigger18Status	
HS1Sideband1	
HS1Sideband2	
HS1Sideband3	
HS1Sideband4	
HS1Sideband5	
HS1Sideband6	
HS1Sideband7	
HS1Sideband8	
HS1Sideband9	
HS1Sideband10	
HS1Sideband11	
HS1Sideband12	
HS1Sideband13	
HS1Sideband14	
HS1Sideband15	
HS1Sideband16	
HS1Sideband17	
HS1Sideband18	
HS2Trigger1Status	
HS2Trigger2Status	
HS2Trigger3Status	
HS2Trigger4Status	
HS2Trigger5Status	
HS2Trigger6Status	
HS2Trigger7Status	
HS2Trigger8Status	
HS2Trigger9Status	
HS2Trigger10Status	

Enumerator

HS2Trigger11Status	
HS2Trigger12Status	
HS2Trigger13Status	
HS2Trigger14Status	
HS2Trigger15Status	
HS2Trigger16Status	
HS2Trigger17Status	
HS2Trigger18Status	
HS2Sideband1	
HS2Sideband2	
HS2Sideband3	
HS2Sideband4	
HS2Sideband5	
HS2Sideband6	
HS2Sideband7	
HS2Sideband8	
HS2Sideband9	
HS2Sideband10	
HS2Sideband11	
HS2Sideband12	
HS2Sideband13	
HS2Sideband14	
HS2Sideband15	
HS2Sideband16	
HS2Sideband17	
HS2Sideband18	
LastPosition	

10.2.1.47 MEA2100DigitalSourceEnumNet enum [MEA2100DigitalSourceEnumNet](#) [strong]

Enumerates the digital source of the MEA2100 device.

Enumerator

DigitalInOfOutPort	
DigitalIn	
DigitalPulse	
Feedback	
AuxIn	
Zero	
One	
HS1Trigger1Status	
HS1Trigger2Status	
HS1Trigger3Status	
HS1Trigger4Status	
HS1Trigger5Status	

Enumerator

HS1Trigger6Status	
HS1Sideband1	
HS1Sideband2	
HS1Sideband3	
HS1Sideband4	
HS1Sideband5	
HS1Sideband6	
HS2Trigger1Status	
HS2Trigger2Status	
HS2Trigger3Status	
HS2Trigger4Status	
HS2Trigger5Status	
HS2Trigger6Status	
HS2Sideband1	
HS2Sideband2	
HS2Sideband3	
HS2Sideband4	
HS2Sideband5	
HS2Sideband6	
PulseGenerator	
DigitalOutStimulator	
DigitalData	
DeviceRunStatus	
LastPosition	

10.2.1.48 MeaLayoutEnumNet `enum MeaLayoutEnumNet [strong]`

Enumerates the MEA layout of the MEA2100 device.

Enumerator

mlUnknown	
mlMEA60	

10.2.1.49 MultiwellDigitalSourceEnumNet `enum MultiwellDigitalSourceEnumNet [strong]`

Enumerates the digital source of the Multiwell device.

Enumerator

DigitalInOfOutPort	
DigitalIn	
DigitalPulse	

Enumerator

Feedback	
AuxIn	
Zero	
One	
HS1Trigger1Status	
HS1Trigger2Status	
HS1Sideband1	
HS1Sideband2	
HS2Trigger1Status	
HS2Trigger2Status	
HS2Sideband1	
HS2Sideband2	
PulseGenerator	
DigitalOutStimulator	
DigitalData	
DeviceRunStatus	
LastPosition	

10.2.1.50 MultiwellPlateTypeEnumNet `enum MultiwellPlateTypeEnumNet [strong]`

Enumerator

Plate_Dummy	
Plate_24W700_100FMA	
Plate_24W030MGA	
Plate_72W500_100PMA	
Plate_72W500_100FMA	
Plate_24W700_100FMB	
Plate_96W700_100FMA	
Plate_96W300_80_1152FMA	
Plate_96W400_80_1152FMB	
Plate_24W300_30_1152GBA	
Plate_24W700_100FMC	
Plate_96W700_100FMB	
Plate_96W700_100GBC	
Plate_96W700_100GBD	
Plate_24W700_100PBB	
Plate_96W700_100PBB	
Plate_ICB8_24W700_100P8A	
Plate_ICB8_96W700_100P8A	
Plate_Dummy_126	
Plate_24W300_30GMA	
Plate_96W700_100GMA	
Plate_24W300_30GBA	
Plate_96W700_100GBA	
Plate_24W300_30GBB	
Plate_96W700_100GBB	
No_Plate	

10.2.1.51 PatchServAdcModeEnumNet enum [PatchServAdcModeEnumNet](#) [strong]

Enumerator

Normal	
CatchAmp	

10.2.1.52 PlateClampEnumNet enum [PlateClampEnumNet](#) [strong]

Enumerator

Close	
Open	
Stop	

10.2.1.53 PlateClampLockEnumNet enum [PlateClampLockEnumNet](#) [strong]

Enumerator

Lock	
Unlock	

10.2.1.54 PortDirectionEnumNet enum [PortDirectionEnumNet](#) [strong]

enumerates a port direction

Enumerator

Output	
Input	

10.2.1.55 PP_Pump_Mode_Type_EnumNet enum [PP_Pump_Mode_Type_EnumNet](#) [strong]

Enumerator

Manual	
--------	--

Enumerator

Digital	
Analog	

10.2.1.56 ProductIdEnumNet enum `ProductIdEnumNet` [strong]

Enumerates the group of MCS devices to connect to.

Enumerator

Any	
None	
LegacyMeaUsb	
ALA_VC3	
Cypress_FX1	
Cypress_FX2	
Cypress_FX3	
MC_Card	
Campden_Ci4600EphysVideoDataIntegrator	
HekaLIH30	
HekaEPC10Single	
HekaEPC10Double	
HekaEPC10Triple	
HekaEPC10Quadro	
HekaLIH408	
HekaLIH816	
HekaITEV100	
HekaPG610	
HekaPG611	
HekaPG612	
HekaPG618	
HekaPG690	
HekaEPCLite	
STG	
Octopot	
Tersens	
Dotriapot	
HLA	
STG400x	
STG4002	
STG4004	
STG4008	
STG400x_opto	
STG4002_opto	
STG4004_opto	
STG4008_opto	
STG5	

Enumerator

STG3008_FA	
MultiwellOptoStim	
Generic	
PGA	
PCX	
TCX	
FCX	
FCB	
TC01	
TC02	
Retina_LED	
AMS_Dongle	
Okuvision_Stimulator	
ExternBCTester	
Triggerbox_IMS	
Triggerbox_AMS	
Triggerbox_AMS3	
ExternDTester	
FunkDongleS	
ExternSTester	
DongleS	
Triggerbox_R5	
MEA_Switch	
MEA_Impedance	
ChannelTest	
Sw2to64	
PeristalticPump	
MEA_Switch_2_1	
MEA_Switch_4_2	
PPS4plus1	
PPS5	
PPS2	
PPS5_DIG	
MEA_Clean	
MEA_Coat	
Multiwell_ICC	
MBC08	
PPC	
MEA1060	
MEA_Sanofi	
ME256	
ME128	
ME64	
ME32	
ME16	
MEA2100_Mini_Usb_develop	
MEA256	
MEA2100	
MEA2100_32	

Enumerator

MEA2100_Lite	
Multiwell	
MEA2100_256	
ME2100	
MEA2100BetaScreen	
MEA2100_Mini	
TBSI_Dacq	
Multiwell_MEA_Mini	
Whole_Cell_Patch	
eCube	
Graphene_ASIC	
GE2100	
M4M_Dock	
Multiboot	
WPA8	
WPA4	
WPA16	
WPA32	
W2100	
NeuroChip	
UsbTest	
SoftwareDongle	
PathIdent	
NF_Gen	
SafeIS	
Encapsulator	
NeurochipConfig	
MeasureTable	
Roboocyte2	
RoboInject	
HiClamp	
PatchServer	
Dilutor	
HiClamp4Uart	
IM16S16KRA	
IM64KRB	
IS32KRA	
IM64KRC	
IM16S8KRA	
IM16KRC	
SmartImplant	
PositionImp	
PositionBase	
PositionIICentralUnit	
PositionIIBase	
GrapheneProjectTestDevice	
Pos900	
Neptun	
Warner_Valve_Control	

Enumerator

Warner_TEER_Machine	
Warner_Ussing	

10.2.1.57 PulseGenerator_Mode_EnumNet enum [PulseGenerator_Mode_EnumNet](#) [strong]

Enumerator

Off	
AlwaysOn	
Gated_Low_Active	
Gated_High_Active	

10.2.1.58 ReferenceElectrodeModeEnumNet enum [ReferenceElectrodeModeEnumNet](#) [strong]

enumerates the electrode subtraction modes

Enumerator

SubtractionOff	
SubtractFromAllOther	
SubtractFromReferenceElectrodeOnly	
SubtractFromAll	

10.2.1.59 ReferenceElectrodeSwitchPositionEnumNet enum [ReferenceElectrodeSwitchPositionEnumNet](#) [strong]

enumerates the possible positions of the referemce electrode switch of the ME2100 device

Enumerator

off	
Ref8	
Ref16	
Ref24	
Ref32	

10.2.1.60 RetriggerActionEnumNet enum [RetriggerActionEnumNet](#) [strong]

Enumerates possible retrigger actions for STG200x devices.

Enumerator

Stop	
Restart	
Ignore	
Gate	
Single	

10.2.1.61 RoboCurrentModeEnumNet enum [RoboCurrentModeEnumNet](#) [strong]

Enumerator

Off	
Break	
Standby	
Reference	
Movement	

10.2.1.62 SampleDstSizeNet enum [SampleDstSizeNet](#) [strong]

Enumerates the destination data format for ChannelBlock functions.

Enumerator

SampleDstSize16	
SampleDstSize32	

10.2.1.63 SampleSizeNet enum [SampleSizeNet](#) [strong]

Enumerates the data format for ChannelBlock functions.

Enumerator

SampleSize16Unsigned	
SampleSize16Signed	
SampleSize24Unsigned	
SampleSize24Signed	
SampleSize32Unsigned	
SampleSize32Signed	
SampleSize64Unsigned	
SampleSize64Signed	

10.2.1.64 SCU_HeadstageIdEnumNet enum [SCU_HeadstageIdEnumNet](#) [strong]

Enumerator

DeviceNotConnected	
Me2100_32Xilinx	
Me2100_32PICiCE40	
Mea2100Mini60PICiCE40	
Mea2100BetaScreen	
Me2100UPA32	
MultiwellMini	
Mea2100Mini120	
Mea2100Mini60ECP5	
eCube	
Me2100Graphene16_32	
GrapheneASIC	
WholeCellPatch	
DeviceHasNoHeadstage	

10.2.1.65 SCUDacqGroupChannelEnumNet enum [SCUDacqGroupChannelEnumNet](#) [strong]

Enumerates the SCU Channel Groups of Datastream

Enumerator

SCU1ElectrodeGroupHS1	
SCU1ElectrodeGroupHS2	
SCU1ElectrodeGroupHS3	
SCU1ElectrodeGroupHS4	
SCU2ElectrodeGroupHS1	
SCU2ElectrodeGroupHS2	
SCU2ElectrodeGroupHS3	
SCU2ElectrodeGroupHS4	
InterfaceADCGroup	
STG1DACSignalGroup	
STG2DACSignalGroup	
DSPAnalogGroup	
DSPDigitalGroup	
IFDigChannelsGroup	
STG1TriggerStatusGroup	
STG1SidebandsGroup	
STG2TriggerStatusGroup	
STG2SidebandsGroup	
AudioTestChannelGroup	
PacketFrameContextGroup	

10.2.1.66 SCUDigitalSourceEnumNet enum SCUDigitalSourceEnumNet [strong]

Enumerates the digital source of the SCU device.

Enumerator

DigitalInOfOutPort	
DigitalIn	
DigitalPulse	
Feedback	
AuxIn	
Zero	
One	
PulseGenerator	
DigitalOutStimulator	
DigitalData	
DeviceRunStatus	
SCU1HS1Sideband1TriggerStatus	
SCU1HS1Sideband2TriggerStatus	
SCU1HS2Sideband1TriggerStatus	
SCU1HS2Sideband2TriggerStatus	
SCU1HS3Sideband1TriggerStatus	
SCU1HS3Sideband2TriggerStatus	
SCU1HS4Sideband1TriggerStatus	
SCU1HS4Sideband2TriggerStatus	
SCU1LEDStim1TriggerStatus	
SCU1LEDStim2TriggerStatus	
SCU1LEDStim3TriggerStatus	
SCU1LEDStim4TriggerStatus	
SCU1HS1Sideband1	
SCU1HS1Sideband2	
SCU1HS2Sideband1	
SCU1HS2Sideband2	
SCU1HS3Sideband1	
SCU1HS3Sideband2	
SCU1HS4Sideband1	
SCU1HS4Sideband2	
SCU1LEDStim1	
SCU1LEDStim2	
SCU1LEDStim3	
SCU1LEDStim4	
SCU2HS1Sideband1TriggerStatus	
SCU2HS1Sideband2TriggerStatus	
SCU2HS2Sideband1TriggerStatus	
SCU2HS2Sideband2TriggerStatus	
SCU2HS3Sideband1TriggerStatus	
SCU2HS3Sideband2TriggerStatus	
SCU2HS4Sideband1TriggerStatus	

Enumerator

SCU2HS4Sideband2TriggerStatus	
SCU2LEDStim1TriggerStatus	
SCU2LEDStim2TriggerStatus	
SCU2LEDStim3TriggerStatus	
SCU2LEDStim4TriggerStatus	
SCU2HS1Sideband1	
SCU2HS1Sideband2	
SCU2HS2Sideband1	
SCU2HS2Sideband2	
SCU2HS3Sideband1	
SCU2HS3Sideband2	
SCU2HS4Sideband1	
SCU2HS4Sideband2	
SCU2LEDStim1	
SCU2LEDStim2	
SCU2LEDStim3	
SCU2LEDStim4	
LastPosition	

10.2.1.67 Stg200xDigoutModeEnumNet enum [Stg200xDigoutModeEnumNet](#) [strong]

Enumerates the DigoutMode on STG400x devices.

Enumerator

Monitor	Monitor digital input pins. Digital out is a mirror of the digital input pins.
Manual	Manually set the value on the digital out pins with SetDigoutValue.
SYNCOUT1	show bit 7 to 15 of syncout channel 1 on the digital outputs
SYNCOUT2	show bit 7 to 15 of syncout channel 2 on the digital outputs
SYNCOUT3	show bit 7 to 15 of syncout channel 3 on the digital outputs
SYNCOUT4	show bit 7 to 15 of syncout channel 4 on the digital outputs
SYNCOUT5	show bit 7 to 15 of syncout channel 5 on the digital outputs
SYNCOUT6	show bit 7 to 15 of syncout channel 6 on the digital outputs
SYNCOUT7	show bit 7 to 15 of syncout channel 7 on the digital outputs
SYNCOUT8	show bit 7 to 15 of syncout channel 8 on the digital outputs

10.2.1.68 Stg200xMultiFileSubmodeEnumNet enum [Stg200xMultiFileSubmodeEnumNet](#) [strong]

Enumerates EnableMultifileMode submodes for STG devices.

Enumerator

MultiFileMode	No Flags.
ExtendedMultiFileMode	Assign all channels to the trigger which number is the given segment number.

10.2.1.69 Stg200xSegmentFlagsEnumNet enum [Stg200xSegmentFlagsEnumNet](#) [strong]

Enumerates Segmentflag options for STG400x devices.

Enumerator

None	No Flags.
UpdateTrigger	Assign all channels to the trigger which number is the given segment number.
DownloadOnly	Only switch the segment for the next download, keep current segment running.
TriggerOnly	Only switch the segment for the next sweep, keep current download segment.
SyncStart	Delay the start the new segment with SegmentStart() until the next sweep has finished.

10.2.1.70 Stg200xTriggerStatusEnumNet enum [Stg200xTriggerStatusEnumNet](#) [strong]

Enumerates the STG download mode trigger status

The STG maintains the status for each of the STG200x_NUM_TRIGGER triggers

Enumerator

Idle	
Running	
Finished	
Armed	

10.2.1.71 Stg3008FilterAmpAmplificationEnumNet enum [Stg3008FilterAmpAmplificationEnumNet](#) [strong]

Enumerates the STG3008FA filter amplifier gains

Enumerator

Gain1000	
Gain500	
Gain200	
Gain100	

10.2.1.72 STG_DestinationEnumNet enum [STG_DestinationEnumNet](#) [strong]

Enumerates the destination for STG downloads.

Enumerator

channeldata_voltage	
channeldata_current	
syncoutdata	
channeldata_positive_voltage	
channeldata_positive_current	
rawdata	
channeldata_current_own_sync	
channeldata_positive_current_own_sync	
channeldata_current_own_boost_gnd_sync	
channeldata_positive_current_own_boost_gnd_sync	
channeldata_current_always_boost	
channeldata_current_always_boost_own_sync	

10.2.1.73 StimulationLayoutConfigurationEnumNet enum [StimulationLayoutConfigurationEnumNet](#) [strong]

enumerates the layout configuration for the MEA2100-256 device

Enumerator

SingleWell	
SixWell	
NineWell	

10.2.1.74 TBSI_DACQDigitalSourceEnumNet enum [TBSI_DACQDigitalSourceEnumNet](#) [strong]

Enumerates the digital source of the TBSI-DACQ device.

Enumerator

DigitalInOfOutPort	
DigitalIn	
DigitalPulse	
Feedback	
AuxIn	
Zero	
One	
DeviceRunStatus	
PulseGenerator	
DigitalOutStimulator	
DigitalData	
HS1DigitalData1	
HS2DigitalData1	
LastPosition	

10.2.1.75 TcxDeviceTypeEnumNet enum [TcxDeviceTypeEnumNet](#) [strong]

Enumerates the type of TCX devices.

Enumerator

Unknown	
Regular	
BMI	
Nanion	
Warner	

10.2.1.76 TcxSensorTypeEnumNet enum [TcxSensorTypeEnumNet](#) [strong]

Enumerates the sensor types for TCX devices

Enumerator

Reserved5	
Reserved4	
Reserved3	
Reserved2	
Reserved1	
NTC10K	
PT1000	
PT100	

10.2.1.77 TeerClampModeEnumNet enum [TeerClampModeEnumNet](#) [strong]

Enumerator

ClampModeVoltage	
ClampModeCurrent	
ClampModeOpen	
ClampModeInternalCalibration	

10.2.1.78 TeerWaveformEnumNet enum [TeerWaveformEnumNet](#) [strong]

Enumerator

Rectangle	
Sine	

10.2.1.79 TriggerSourceEnumNet enum [TriggerSourceEnumNet](#) [strong]

Enumerates the trigger source of the MEA2100 device.

Enumerator

tsNone	
tsDigitalIn1	
tsDigitalIn2	
tsDigitalIn3	
tsDigitalIn4	
tsDigitalIn5	
tsDigitalIn6	
tsDigitalIn7	
tsDigitalIn8	
tsDigitalIn9	
tsDigitalIn10	
tsDigitalIn11	
tsDigitalIn12	
tsDigitalIn13	
tsDigitalIn14	
tsDigitalIn15	
tsDigitalIn16	
tsDigitalIn17	
tsDigitalIn18	
tsDigitalIn19	
tsDigitalIn20	
tsDigitalIn21	
tsDigitalIn22	
tsDigitalIn23	
tsDigitalIn24	
tsDigitalIn25	
tsDigitalIn26	
tsDigitalIn27	
tsDigitalIn28	
tsDigitalIn29	
tsDigitalIn30	
tsDigitalIn31	
tsDigitalIn32	
tsFeedback1	
tsFeedback2	
tsFeedback3	

Enumerator

tsFeedback4	
tsFeedback5	
tsFeedback6	
tsFeedback7	
tsFeedback8	
tsFeedback9	
tsFeedback10	
tsFeedback11	
tsFeedback12	
tsFeedback13	
tsFeedback14	
tsFeedback15	
tsFeedback16	
tsFeedback17	
tsFeedback18	
tsFeedback19	
tsFeedback20	
tsFeedback21	
tsFeedback22	
tsFeedback23	
tsFeedback24	
tsFeedback25	
tsFeedback26	
tsFeedback27	
tsFeedback28	
tsFeedback29	
tsFeedback30	
tsFeedback31	
tsFeedback32	
tsAuxIn1	
tsAuxIn2	
tsDigitalPuse0	
tsDigitalPuse1	
tsDigitalPuse2	
tsDigitalPuse3	
tsDigitalPuse4	
tsDigitalPuse5	
tsDigitalPuse6	
tsDigitalPuse7	
tsDigitalPuse8	
tsDigitalPuse9	
tsDigitalPuse10	
tsDigitalPuse11	
tsDigitalPuse12	
tsDigitalPuse13	
tsDigitalPuse14	
tsDigitalPuse15	
tsDigitalPuse16	
tsDigitalPuse17	

Enumerator

tsDigitalPuse18	
tsDigitalPuse19	
tsDigitalPuse20	
tsDigitalPuse21	
tsDigitalPuse22	
tsDigitalPuse23	
tsDigitalPuse24	
tsDigitalPuse25	
tsDigitalPuse26	
tsDigitalPuse27	
tsDigitalPuse28	
tsDigitalPuse29	
tsDigitalPuse30	
tsDigitalPuse31	
tsTriggered	
tsSidebandBit8	
tsDACQCy1Dev1Runs	
tsDACQCy1Dev2Runs	
tsDACQCy2Dev1Runs	
tsDACQCy2Dev2Runs	

10.2.1.80 UsbVendorIdEnumNet enum [UsbVendorIdEnumNet](#) [strong]

Enumerator

Unknown	
None	
Renesas	
ASMedia	
Intel	

10.2.1.81 UssingClampModeEnumNet enum [UssingClampModeEnumNet](#) [strong]

Enumerator

VoltageClamp	
CurrentClamp	
OpenClamp	
Standby	
ElectrodeOffset	

10.2.1.82 UssingUnitEnumNet enum [UssingUnitEnumNet](#) [strong]

Enumerator

Volt	
Ampere	
State	

10.2.1.83 VendorIdEnumNet enum [VendorIdEnumNet](#) [strong]

Enumerates the group of MCS devices to connect to.

Enumerator

Any	
None	
MCS_old	
MCS	
PCI	
Cypress	
ALA_VC3	

10.2.1.84 W2100_Accel_Gyro_Select_EnumNet enum [W2100_Accel_Gyro_Select_EnumNet](#) [strong]

enumerates the accelerometer configuration on the W2100 device

Enumerator

Off	
GyroOnly	
AccelOnly	
Both	

10.2.1.85 W2100DacqGroupChannelEnumNet enum [W2100DacqGroupChannelEnumNet](#) [strong]

Enumerates the W2100 Channel Groups of Datastream

Enumerator

InterfaceADCGroup	
-------------------	--

Enumerator

DSPDataGroup	
WirelessHeadStageAnalogRE1HS1	
WirelessHeadStageStatusRE1HS1	
WirelessHeadStageAnalogRE1HS2	
WirelessHeadStageStatusRE1HS2	
WirelessHeadStageAnalogRE1HS3	
WirelessHeadStageStatusRE1HS3	
WirelessHeadStageAnalogRE1HS4	
WirelessHeadStageStatusRE1HS4	
WirelessHeadStageAnalogRE2HS1	
WirelessHeadStageStatusRE2HS1	
WirelessHeadStageAnalogRE2HS2	
WirelessHeadStageStatusRE2HS2	
WirelessHeadStageAnalogRE2HS3	
WirelessHeadStageStatusRE2HS3	
WirelessHeadStageAnalogRE2HS4	
WirelessHeadStageStatusRE2HS4	
WirelessHeadStageGyroDataRE1HS1	
WirelessHeadStageAccDataRE1HS1	
WirelessHeadStageGyroDataRE1HS2	
WirelessHeadStageAccDataRE1HS2	
WirelessHeadStageGyroDataRE1HS3	
WirelessHeadStageAccDataRE1HS3	
WirelessHeadStageGyroDataRE1HS4	
WirelessHeadStageAccDataRE1HS4	
WirelessHeadStageGyroDataRE2HS1	
WirelessHeadStageAccDataRE2HS1	
WirelessHeadStageGyroDataRE2HS2	
WirelessHeadStageAccDataRE2HS2	
WirelessHeadStageGyroDataRE2HS3	
WirelessHeadStageAccDataRE2HS3	
WirelessHeadStageGyroDataRE2HS4	
WirelessHeadStageAccDataRE2HS4	
WirelessHeadStageOptoStimCurrentRE1HS1	
WirelessHeadStageReservedARE1HS1	
WirelessHeadStageOptoStimCurrentRE1HS2	
WirelessHeadStageReservedARE1HS2	
WirelessHeadStageOptoStimCurrentRE1HS3	
WirelessHeadStageReservedARE1HS3	
WirelessHeadStageOptoStimCurrentRE1HS4	
WirelessHeadStageReservedARE1HS4	
WirelessHeadStageOptoStimCurrentRE2HS1	
WirelessHeadStageReservedARE2HS1	
WirelessHeadStageOptoStimCurrentRE2HS2	
WirelessHeadStageReservedARE2HS2	
WirelessHeadStageOptoStimCurrentRE2HS3	

Enumerator

WirelessHeadStageReservedARE2HS3	
WirelessHeadStageOptoStimCurrentRE2HS4	
WirelessHeadStageReservedARE2HS4	
WirelessHeadStageReservedBRE1HS1	
WirelessHeadStageReservedCRE1HS1	
WirelessHeadStageReservedBRE1HS2	
WirelessHeadStageReservedCRE1HS2	
WirelessHeadStageReservedBRE1HS3	
WirelessHeadStageReservedCRE1HS3	
WirelessHeadStageReservedBRE1HS4	
WirelessHeadStageReservedCRE1HS4	
WirelessHeadStageReservedBRE2HS1	
WirelessHeadStageReservedCRE2HS1	
WirelessHeadStageReservedBRE2HS2	
WirelessHeadStageReservedCRE2HS2	
WirelessHeadStageReservedBRE2HS3	
WirelessHeadStageReservedCRE2HS3	
WirelessHeadStageReservedBRE2HS4	
WirelessHeadStageReservedCRE2HS4	
IFDigChannelsGroup	
AudioTestChannelGroup	
PacketFrameContextGroup	

10.2.1.86 W2100DigitalSourceEnumNet enum `W2100DigitalSourceEnumNet` [strong]

Enumerates the digital source of the W2100 device.

Enumerator

DigitalInOfOutPort	
DigitalIn	
DigitalPulse	
Feedback	
AuxIn	
Zero	
One	
PulseGenerator	
DigDataFromReceiver	
DigitalOutStimulator	
DigitalData	
DeviceRunStatus	
DigStreamFromReceiver	
LastPosition	

10.2.1.87 WvcDisplayModeEnumNet enum [WvcDisplayModeEnumNet](#) [strong]

enumerates Wvc display mode

Enumerator

Work	
PC	
Settings	
TouchTest	

10.2.1.88 WvcValveModeEnumNet enum [WvcValveModeEnumNet](#) [strong]

enumerates Wvc valve mode

Enumerator

Manual	
Digital	
Analog	
Table	

10.2.2 Function Documentation

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

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

Delegate to show a device arrival or removal.

10.2.2.3 OnError() public delegate void Mcs::Usb::OnError (
 String[^] *msg*,
 int *action*)

10.2.2.4 OnMcsUsbDeviceState() public delegate void OnMcsUsbDeviceState (
usbSetupPacket_t^ request)

10.2.2.5 OnMcsUsbDeviceStateCallback() private delegate void OnMcsUsbDeviceStateCallback (
IntPtr pThis,
uint32_t size,
IntPtr buffer)

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

10.2.2.7 OnStg200xDataHandler() public delegate void Mcs::Usb::OnStg200xDataHandler (
uint32_t trigger)

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

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

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

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

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

11 Class Documentation

11.1 CW2100_FunctionNet::AudioChannelsNet Struct Reference

Public Attributes

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

11.1.1 Member Data Documentation

11.1.1.1 `amplification` `int` `amplification`

11.1.1.2 `channel` `int` `channel`

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

11.2 BatteryState Class Reference

Properties

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

11.2.1 Property Documentation

11.2.1.1 `Charge` `unsigned int` `Charge` `[get]`

11.2.1.2 `ChargeRegionString` `System::String^` `ChargeRegionString` `[get]`

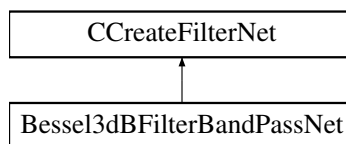
11.2.1.3 ChargeString `System:: String^ ChargeString [get]`

11.2.1.4 Voltage `unsigned int Voltage [get]`

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

11.3 Bessel3dBFilterBandPassNet Class Reference

Inheritance diagram for Bessel3dBFilterBandPassNet:



Public Member Functions

- [Bessel3dBFilterBandPassNet](#) (int numCoefSets, int order, double sampleRate, double centerFrequency, double widthFrequency, double scale)

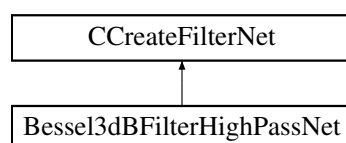
Additional Inherited Members

11.3.1 Constructor & Destructor Documentation

11.3.1.1 Bessel3dBFilterBandPassNet() `Bessel3dBFilterBandPassNet (`
 int numCoefSets,
 int order,
 double sampleRate,
 double centerFrequency,
 double widthFrequency,
 double scale)

11.4 Bessel3dBFilterHighPassNet Class Reference

Inheritance diagram for Bessel3dBFilterHighPassNet:



Public Member Functions

- [Bessel3dBFilterHighPassNet](#) (int numCoefSets, int order, double sampleRate, double cutoffFrequency, double scale)

Additional Inherited Members

11.4.1 Constructor & Destructor Documentation

11.4.1.1 Bessel3dBFilterHighPassNet() `Bessel3dBFilterHighPassNet` (

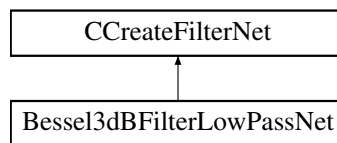
```

    int numCoefSets,
    int order,
    double sampleRate,
    double cutoffFrequency,
    double scale )

```

11.5 Bessel3dBFilterLowPassNet Class Reference

Inheritance diagram for Bessel3dBFilterLowPassNet:



Public Member Functions

- [Bessel3dBFilterLowPassNet](#) (int numCoefSets, int order, double sampleRate, double cutoffFrequency, double scale)

Additional Inherited Members

11.5.1 Constructor & Destructor Documentation

11.5.1.1 Bessel3dBFilterLowPassNet() `Bessel3dBFilterLowPassNet` (

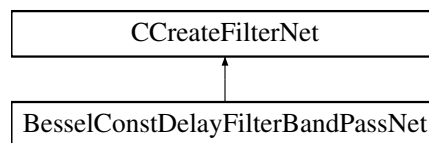
```

    int numCoefSets,
    int order,
    double sampleRate,
    double cutoffFrequency,
    double scale )

```

11.6 BesselConstDelayFilterBandPassNet Class Reference

Inheritance diagram for BesselConstDelayFilterBandPassNet:



Public Member Functions

- [BesselConstDelayFilterBandPassNet](#) (int numCoefSets, int order, double sampleRate, double centerFrequency, double widthFrequency, double scale)

Additional Inherited Members

11.6.1 Constructor & Destructor Documentation

11.6.1.1 BesselConstDelayFilterBandPassNet() [BesselConstDelayFilterBandPassNet](#) (

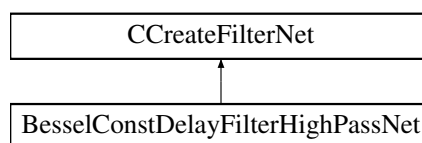
```

    int numCoefSets,
    int order,
    double sampleRate,
    double centerFrequency,
    double widthFrequency,
    double scale )

```

11.7 BesselConstDelayFilterHighPassNet Class Reference

Inheritance diagram for BesselConstDelayFilterHighPassNet:



Public Member Functions

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

Additional Inherited Members

11.7.1 Constructor & Destructor Documentation

11.7.1.1 BesselConstDelayFilterHighPassNet() [BesselConstDelayFilterHighPassNet](#) (

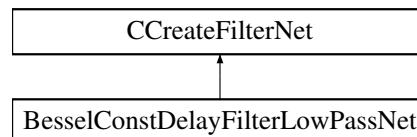
```

    int numCoefSets,
    int order,
    double sampleRate,
    double cutoffFrequency,
    double scale )

```

11.8 BesselConstDelayFilterLowPassNet Class Reference

Inheritance diagram for BesselConstDelayFilterLowPassNet:



Public Member Functions

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

Additional Inherited Members

11.8.1 Constructor & Destructor Documentation

11.8.1.1 BesselConstDelayFilterLowPassNet() [BesselConstDelayFilterLowPassNet](#) (

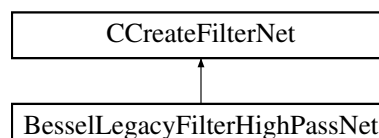
```

    int numCoefSets,
    int order,
    double sampleRate,
    double cutoffFrequency,
    double scale )

```

11.9 BesselLegacyFilterHighPassNet Class Reference

Inheritance diagram for BesselLegacyFilterHighPassNet:



Public Member Functions

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

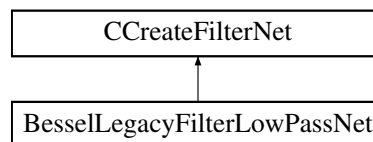
Additional Inherited Members

11.9.1 Constructor & Destructor Documentation

11.9.1.1 BesselLegacyFilterHighPassNet() `BesselLegacyFilterHighPassNet (`
 `int numCoefSets,`
 `int order,`
 `double sampleRate,`
 `double cutoffFrequency,`
 `double scale)`

11.10 BesselLegacyFilterLowPassNet Class Reference

Inheritance diagram for BesselLegacyFilterLowPassNet:



Public Member Functions

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

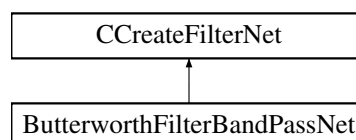
Additional Inherited Members

11.10.1 Constructor & Destructor Documentation

11.10.1.1 BesselLegacyFilterLowPassNet() `BesselLegacyFilterLowPassNet (`
 `int numCoefSets,`
 `int order,`
 `double sampleRate,`
 `double cutoffFrequency,`
 `double scale)`

11.11 ButterworthFilterBandPassNet Class Reference

Inheritance diagram for ButterworthFilterBandPassNet:



Public Member Functions

- [ButterworthFilterBandPassNet](#) (int numCoefSets, int order, double sampleRate, double centerFrequency, double widthFrequency, double scale)

Additional Inherited Members

11.11.1 Constructor & Destructor Documentation

11.11.1.1 ButterworthFilterBandPassNet() [ButterworthFilterBandPassNet](#) (

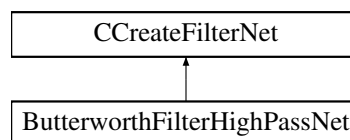
```

    int numCoefSets,
    int order,
    double sampleRate,
    double centerFrequency,
    double widthFrequency,
    double scale )

```

11.12 ButterworthFilterHighPassNet Class Reference

Inheritance diagram for ButterworthFilterHighPassNet:



Public Member Functions

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

Additional Inherited Members

11.12.1 Constructor & Destructor Documentation

11.12.1.1 ButterworthFilterHighPassNet() [ButterworthFilterHighPassNet](#) (

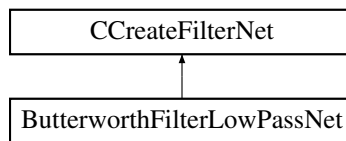
```

    int numCoefSets,
    int order,
    double sampleRate,
    double cutoffFrequency,
    double scale )

```

11.13 ButterworthFilterLowPassNet Class Reference

Inheritance diagram for ButterworthFilterLowPassNet:



Public Member Functions

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

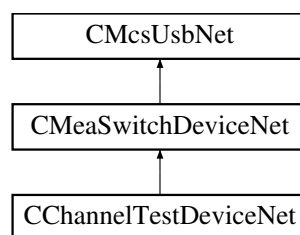
Additional Inherited Members

11.13.1 Constructor & Destructor Documentation

11.13.1.1 ButterworthFilterLowPassNet() `ButterworthFilterLowPassNet (`
 int numCoefSets,
 int order,
 double sampleRate,
 double cutoffFrequency,
 double scale)

11.14 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**11.14.1 Constructor & Destructor Documentation**

11.14.1.1 CChannelTestDeviceNet() `CChannelTestDeviceNet ()`

11.14.1.2 ~CChannelTestDeviceNet() `~CChannelTestDeviceNet ()`

11.14.2 Member Function Documentation

11.14.2.1 SetAmplitude() `void SetAmplitude (unsigned int Amplitude)`

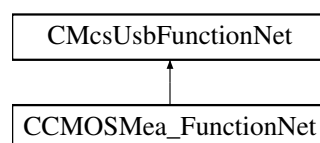
11.14.2.2 SetAttenuation() `void SetAttenuation (unsigned int Attenuation)`

11.14.2.3 SetFrequency() `void SetFrequency (unsigned int Frequency)`

11.14.2.4 SetWaveform() `void SetWaveform (unsigned int Waveform)`

11.15 CCMOSMea_FunctionNet Class Reference

Inheritance diagram for CCMOSMea_FunctionNet:



Public Member Functions

- [CCMOSMea_FunctionNet](#) ([CMcsUsbNet](#)[^] mcsusb, [CMcsUsbFunctionPointerContainer](#)[^] cMOSMea_[↵]FunctionPointerContainer)
- [CCMOSMea_FunctionNet](#) ([CMcsUsbNet](#)[^] 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 virtual[↵]Device)
- void [EnableChannelsInGroup](#) ([DacqGroupChannelEnumNet](#) GroupID, List< bool >[^] EnabledChannelsBit[↵]Map)
- void [EnableChannelsInGroup](#) ([DacqGroupChannelEnumNet](#) GroupID, List< bool >[^] EnabledChannelsBit[↵]Map, 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 Channel↵
Number)`
- `uint32_t GetGroupChannelBitmaskBySelect (DacqGroupChannelEnumNet GroupID, uint32_t Channel↵
Number, uint32_t virtualDevice)`
- `CMOSMeaInterfaceADCEnumNet GetGroupChannelBitmaskInterfaceADC (uint32_t ChannelNumber)`
- `CMOSMeaInterfaceADCEnumNet GetGroupChannelBitmaskInterfaceADC (uint32_t ChannelNumber,
uint32_t virtualDevice)`
- `CMOSMeaIFDigChannelEnumNet GetGroupChannelBitmaskIFDigChannels (uint32_t ChannelNumber)`
- `CMOSMeaIFDigChannelEnumNet GetGroupChannelBitmaskIFDigChannels (uint32_t ChannelNumber,
uint32_t virtualDevice)`
- `CMOSMeaHeadstage1NCBathCurrentEnumNet GetGroupChannelBitmaskHS1NCBathCurrent (uint32_t
ChannelNumber)`
- `CMOSMeaHeadstage1NCBathCurrentEnumNet GetGroupChannelBitmaskHS1NCBathCurrent (uint32_t
ChannelNumber, uint32_t virtualDevice)`
- `CMOSMeaHeadstage1NCCol2CurrentEnumNet GetGroupChannelBitmaskHS1NCCol2Current (uint32_t
ChannelNumber)`
- `CMOSMeaHeadstage1NCCol2CurrentEnumNet GetGroupChannelBitmaskHS1NCCol2Current (uint32_t
ChannelNumber, uint32_t virtualDevice)`
- `CMOSMeaHeadstage1NChipTempEnumNet GetGroupChannelBitmaskHS1NChipTemp (uint32_t Channel↵
Number)`
- `CMOSMeaHeadstage1NChipTempEnumNet GetGroupChannelBitmaskHS1NChipTemp (uint32_t Channel↵
Number, uint32_t virtualDevice)`
- `CMOSMeaSTG1DACSignalEnumNet GetGroupChannelBitmaskSTG1DACSignal (uint32_t ChannelNumber)`
- `CMOSMeaSTG1DACSignalEnumNet GetGroupChannelBitmaskSTG1DACSignal (uint32_t ChannelNumber,
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

11.15.1 Constructor & Destructor Documentation

11.15.1.1 CCMOSMea_FunctionNet() [1/2] `CCMOSMea_FunctionNet (`
`CMcsUsbNet^ mcsusb,`
`CMcsUsbFunctionPointerContainer^ cMOSMea_FunctionPointerContainer)`

11.15.1.2 CCMOSMea_FunctionNet() [2/2] `CCMOSMea_FunctionNet (`
`CMcsUsbNet^ mcsusb)`

11.15.2 Member Function Documentation

11.15.2.1 AreTransistorVoltagesSet() `bool AreTransistorVoltagesSet ()`

11.15.2.2 ClearSTGOutput() `void ClearSTGOutput (`
`uint32_t Channel)`

11.15.2.3 DetectChipType() `enCMosMeaChipType DetectChipType ()`

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

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

11.15.2.6 GetADCInputOffset() `int32_t GetADCInputOffset ()`

11.15.2.7 GetBath() `int32_t GetBath ()`

11.15.2.8 GetBathMode() [CMOSMeaBathModeEnumNet](#) GetBathMode ()

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

11.15.2.10 GetEnabledChannelsInGroup() [2/2] [List<bool>](#) ^ GetEnabledChannelsInGroup ([DacqGroupChannelEnumNet](#) *GroupID*, [uint32_t](#) *virtualDevice*)

11.15.2.11 GetGate() [int32_t](#) GetGate ()

11.15.2.12 GetGNDI() [int32_t](#) GetGNDI ()

11.15.2.13 GetGroupADCBits() [1/2] [int32_t](#) GetGroupADCBits ([DacqGroupChannelEnumNet](#) *GroupID*)

11.15.2.14 GetGroupADCBits() [2/2] [int32_t](#) GetGroupADCBits ([DacqGroupChannelEnumNet](#) *GroupID*, [uint32_t](#) *virtualDevice*)

11.15.2.15 GetGroupChannelBitmaskBySelect() [1/2] [uint32_t](#) GetGroupChannelBitmaskBySelect ([DacqGroupChannelEnumNet](#) *GroupID*, [uint32_t](#) *ChannelNumber*)

11.15.2.16 GetGroupChannelBitmaskBySelect() [2/2] [uint32_t](#) GetGroupChannelBitmaskBySelect ([DacqGroupChannelEnumNet](#) *GroupID*, [uint32_t](#) *ChannelNumber*, [uint32_t](#) *virtualDevice*)

11.15.2.17 GetGroupChannelBitmaskHS1NCBathCurrent() [1/2] [CMOSMeaHeadstage1NCBathCurrentEnumNet](#)

```
GetGroupChannelBitmaskHS1NCBathCurrent (
    uint32_t ChannelNumber )
```

11.15.2.18 GetGroupChannelBitmaskHS1NCBathCurrent() [2/2] [CMOSMeaHeadstage1NCBathCurrentEnumNet](#)

```
GetGroupChannelBitmaskHS1NCBathCurrent (
    uint32_t ChannelNumber,
    uint32_t virtualDevice )
```

11.15.2.19 GetGroupChannelBitmaskHS1NCCol2Current() [1/2] [CMOSMeaHeadstage1NCCol2CurrentEnumNet](#)

```
GetGroupChannelBitmaskHS1NCCol2Current (
    uint32_t ChannelNumber )
```

11.15.2.20 GetGroupChannelBitmaskHS1NCCol2Current() [2/2] [CMOSMeaHeadstage1NCCol2CurrentEnumNet](#)

```
GetGroupChannelBitmaskHS1NCCol2Current (
    uint32_t ChannelNumber,
    uint32_t virtualDevice )
```

11.15.2.21 GetGroupChannelBitmaskHS1NChipTemp() [1/2] [CMOSMeaHeadstage1NChipTempEnumNet](#)

```
GetGroupChannelBitmaskHS1NChipTemp (
    uint32_t ChannelNumber )
```

11.15.2.22 GetGroupChannelBitmaskHS1NChipTemp() [2/2] [CMOSMeaHeadstage1NChipTempEnumNet](#)

```
GetGroupChannelBitmaskHS1NChipTemp (
    uint32_t ChannelNumber,
    uint32_t virtualDevice )
```

11.15.2.23 GetGroupChannelBitmaskHS1Sidebands() [1/2] [CMOSMeaHS1SidebandEnumNet](#) [GetGroup↔](#)

```
ChannelBitmaskHS1Sidebands (
    uint32_t ChannelNumber )
```

11.15.2.24 GetGroupChannelBitmaskHS1Sidebands() [2/2] [CMOSMeaHS1SidebandEnumNet](#) [GetGroup↔](#)

```
ChannelBitmaskHS1Sidebands (
    uint32_t ChannelNumber,
    uint32_t virtualDevice )
```

11.15.2.25 GetGroupChannelBitmaskHS1TriggerStatus() [1/2] [CMOSMeaHS1TriggerStatusEnumNet](#) Get↔
 GroupChannelBitmaskHS1TriggerStatus (
 uint32_t *ChannelNumber*)

11.15.2.26 GetGroupChannelBitmaskHS1TriggerStatus() [2/2] [CMOSMeaHS1TriggerStatusEnumNet](#) Get↔
 GroupChannelBitmaskHS1TriggerStatus (
 uint32_t *ChannelNumber*,
 uint32_t *virtualDevice*)

11.15.2.27 GetGroupChannelBitmaskIFDigChannels() [1/2] [CMOSMeaIFDigChannelEnumNet](#) GetGroup↔
 ChannelBitmaskIFDigChannels (
 uint32_t *ChannelNumber*)

11.15.2.28 GetGroupChannelBitmaskIFDigChannels() [2/2] [CMOSMeaIFDigChannelEnumNet](#) GetGroup↔
 ChannelBitmaskIFDigChannels (
 uint32_t *ChannelNumber*,
 uint32_t *virtualDevice*)

11.15.2.29 GetGroupChannelBitmaskInterfaceADC() [1/2] [CMOSMeaInterfaceADCEnumNet](#) GetGroup↔
 ChannelBitmaskInterfaceADC (
 uint32_t *ChannelNumber*)

11.15.2.30 GetGroupChannelBitmaskInterfaceADC() [2/2] [CMOSMeaInterfaceADCEnumNet](#) GetGroup↔
 ChannelBitmaskInterfaceADC (
 uint32_t *ChannelNumber*,
 uint32_t *virtualDevice*)

11.15.2.31 GetGroupChannelBitmaskPacketFrameContext() [1/2] [CMOSMeaPacketFrameContextGroupEnumNet](#)
 GetGroupChannelBitmaskPacketFrameContext (
 uint32_t *ChannelNumber*)

11.15.2.32 GetGroupChannelBitmaskPacketFrameContext() [2/2] [CMOSMeaPacketFrameContextGroupEnumNet](#)
 GetGroupChannelBitmaskPacketFrameContext (
 uint32_t *ChannelNumber*,
 uint32_t *virtualDevice*)

11.15.2.33 GetGroupChannelBitmaskSTG1DACSignal() [1/2] [CMOSMeaSTG1DACSignalEnumNet](#) Get↔
GroupChannelBitmaskSTG1DACSignal (
 uint32_t ChannelNumber)

11.15.2.34 GetGroupChannelBitmaskSTG1DACSignal() [2/2] [CMOSMeaSTG1DACSignalEnumNet](#) Get↔
GroupChannelBitmaskSTG1DACSignal (
 uint32_t ChannelNumber,
 uint32_t virtualDevice)

11.15.2.35 GetGroupDCOffset() [1/2] int32_t GetGroupDCOffset (
 [DacqGroupChannelEnumNet](#) GroupID)

11.15.2.36 GetGroupDCOffset() [2/2] int32_t GetGroupDCOffset (
 [DacqGroupChannelEnumNet](#) GroupID,
 uint32_t virtualDevice)

11.15.2.37 GetGroupID() [1/2] [DacqGroupChannelEnumNet](#) GetGroupID (
 uint32_t Index)

11.15.2.38 GetGroupID() [2/2] [DacqGroupChannelEnumNet](#) GetGroupID (
 uint32_t Index,
 uint32_t virtualDevice)

11.15.2.39 GetGroupNumberOfChannels() [1/2] uint32_t GetGroupNumberOfChannels (
 [DacqGroupChannelEnumNet](#) GroupID)

11.15.2.40 GetGroupNumberOfChannels() [2/2] uint32_t GetGroupNumberOfChannels (
 [DacqGroupChannelEnumNet](#) GroupID,
 uint32_t virtualDevice)

11.15.2.41 GetGroupResolutionPerDigit() [1/2] uint32_t GetGroupResolutionPerDigit (
 [DacqGroupChannelEnumNet](#) GroupID)

11.15.2.42 GetGroupResolutionPerDigit() [2/2] `uint32_t GetGroupResolutionPerDigit (`
 `DacqGroupChannelEnumNet GroupID,`
 `uint32_t virtualDevice)`

11.15.2.43 GetGroupSampleSize() [1/2] `SampleSizeNet GetGroupSampleSize (`
 `DacqGroupChannelEnumNet GroupID)`

11.15.2.44 GetGroupSampleSize() [2/2] `SampleSizeNet GetGroupSampleSize (`
 `DacqGroupChannelEnumNet GroupID,`
 `uint32_t virtualDevice)`

11.15.2.45 GetGroupType() [1/2] `DacqMeaGroupTypeEnumNet GetGroupType (`
 `DacqGroupChannelEnumNet GroupID)`

11.15.2.46 GetGroupType() [2/2] `DacqMeaGroupTypeEnumNet GetGroupType (`
 `DacqGroupChannelEnumNet GroupID,`
 `uint32_t virtualDevice)`

11.15.2.47 GetGroupUnit() [1/2] `CMOSMeaValueUnitEnumNet GetGroupUnit (`
 `DacqGroupChannelEnumNet GroupID)`

11.15.2.48 GetGroupUnit() [2/2] `CMOSMeaValueUnitEnumNet GetGroupUnit (`
 `DacqGroupChannelEnumNet GroupID,`
 `uint32_t virtualDevice)`

11.15.2.49 GetMaxNumOfColumns() `uint32_t GetMaxNumOfColumns (`
 `uint32_t Samplerate)`

11.15.2.50 GetNeurochipMemoryData() [1/2] `uint32_t GetNeurochipMemoryData (`
 `uint16_t MemAddress)`

11.15.2.51 GetNeurochipMemoryData() [2/2] `array<uint32_t> ^ GetNeurochipMemoryData (`
 `uint16_t MemAddress,`
 `uint32_t RequestLength)`

11.15.2.52 GetNeurochipMemorySize() `uint32_t GetNeurochipMemorySize ()`

11.15.2.53 GetNumberOfSupportedGroups() [1/2] `uint32_t GetNumberOfSupportedGroups ()`

11.15.2.54 GetNumberOfSupportedGroups() [2/2] `uint32_t GetNumberOfSupportedGroups (`
 `uint32_t virtualDevice)`

11.15.2.55 GetSourceBulk() `int32_t GetSourceBulk ()`

11.15.2.56 GetSourceDrain() `int32_t GetSourceDrain ()`

11.15.2.57 GetSourceGate() `int32_t GetSourceGate ()`

11.15.2.58 GetStimulusSites() `List<int16_t> ^ GetStimulusSites ()`

11.15.2.59 GetVDD3I() `int32_t GetVDD3I ()`

11.15.2.60 GetVDDI() `int32_t GetVDDI ()`

11.15.2.61 IsChipPowered() `bool IsChipPowered ()`

11.15.2.62 IsGateFloating() `bool IsGateFloating ()`

11.15.2.63 PowerChip() `void PowerChip (`
`bool on)`

11.15.2.64 SetADCInputOffset() `void SetADCInputOffset (`
`int32_t offset)`

11.15.2.65 SetBath() `void SetBath (`
`int32_t voltage)`

11.15.2.66 SetBathMode() `void SetBathMode (`
`CMOSMeaBathModeEnumNet Mode)`

11.15.2.67 SetGate() `void SetGate (`
`int32_t voltage)`

11.15.2.68 SetGateFloating() `void SetGateFloating ()`

11.15.2.69 SetGateToVOP() `void SetGateToVOP ()`

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

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

11.15.2.72 SetSourceBulk() `void SetSourceBulk (`
 `int32_t voltage)`

11.15.2.73 SetSourceDrain() `void SetSourceDrain (`
 `int32_t voltage)`

11.15.2.74 SetSourceGate() `void SetSourceGate (`
 `int32_t voltage)`

11.15.2.75 SetStimulusSites() `void SetStimulusSites (`
 `List< int16_t >^ SwitchPosition)`

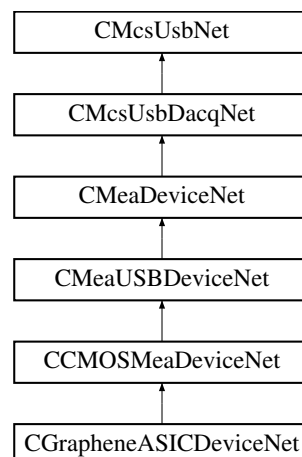
11.15.2.76 UpdateTransistorVoltages() `void UpdateTransistorVoltages ()`

11.15.2.77 VOPSTimerSetResetTimes() [1/2] `void VOPSTimerSetResetTimes (`
 `uint32_t ResetTime,`
 `uint32_t IntervalTime)`

11.15.2.78 VOPSTimerSetResetTimes() [2/2] `void VOPSTimerSetResetTimes (`
 `uint32_t ResetTime,`
 `uint32_t IntervalTime,`
 `uint32_t HPFilterResetTime)`

11.16 CCMOSMeaDeviceNet Class Reference

Inheritance diagram for CCMOSMeaDeviceNet:



Classes

- class [CRegionOfInterestRect](#)

Public Member Functions

- [CCMOSMeaDeviceNet](#) (void)
- [~CCMOSMeaDeviceNet](#) ()
- virtual void [SetBaseSamplerate](#) (int BaseSamplerate)
- virtual int [GetBaseSamplerate](#) ()
- virtual array< int > ^ [GetAvailableBaseSamplerates](#) ()
- int [GetMaxReadableColumns](#) ()
- virtual 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](#) (DacqGroupChannelEnumNet group, int frames, [System::Runtime::InteropServices::Out]int % frames_ret)
- System::Collections::Generic::Dictionary< int, array< int16_t > ^> ^ [GetChannelDataI16](#) (DacqGroupChannelEnumNet group, int frames, [System::Runtime::InteropServices::Out]int % frames_ret)
- System::Collections::Generic::Dictionary< int, array< uint32_t > ^> ^ [GetChannelDataUI32](#) (DacqGroupChannelEnumNet group, int frames, [System::Runtime::InteropServices::Out]int % frames_ret)
- System::Collections::Generic::Dictionary< int, array< int32_t > ^> ^ [GetChannelDataI32](#) (DacqGroupChannelEnumNet group, int frames, [System::Runtime::InteropServices::Out]int % frames_ret)

Properties

- [CCMOSMea_FunctionNet](#) ^ [CMosMea](#) [get]
- [CStimulusFunctionNet](#) ^ [Stimulus](#) [get]

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

11.16.1.1 CCMOSMeaDeviceNet() [CCMOSMeaDeviceNet](#) (void)

11.16.1.2 ~CCMOSMeaDeviceNet() [~CCMOSMeaDeviceNet](#) ()

11.16.2 Member Function Documentation

11.16.2.1 GetAvailableBaseSamplerates() `virtual array<int> ^ GetAvailableBaseSamplerates ()`
[virtual]

Reimplemented in [CGrapheneASICDeviceNet](#).

11.16.2.2 GetBaseSamplerate() `virtual int GetBaseSamplerate ()` [virtual]

Reimplemented in [CGrapheneASICDeviceNet](#).

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

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

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

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

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

11.16.2.8 GetMaxReadableColumns() `int GetMaxReadableColumns ()`

11.16.2.9 SetBaseSamplerate() `virtual void SetBaseSamplerate (
int BaseSamplerate) [virtual]`

Reimplemented in [CGrapheneASICDeviceNet](#).

11.16.2.10 SetRegionOfInterests() `virtual void SetRegionOfInterests (
System::Collections::Generic::Dictionary< int, CRegionOfInterestRect^>^ rois)
[virtual]`

11.16.2.11 UpdateChannelBlock() `void UpdateChannelBlock (
int queuesize,
int threshold,
int channels_in_block)`

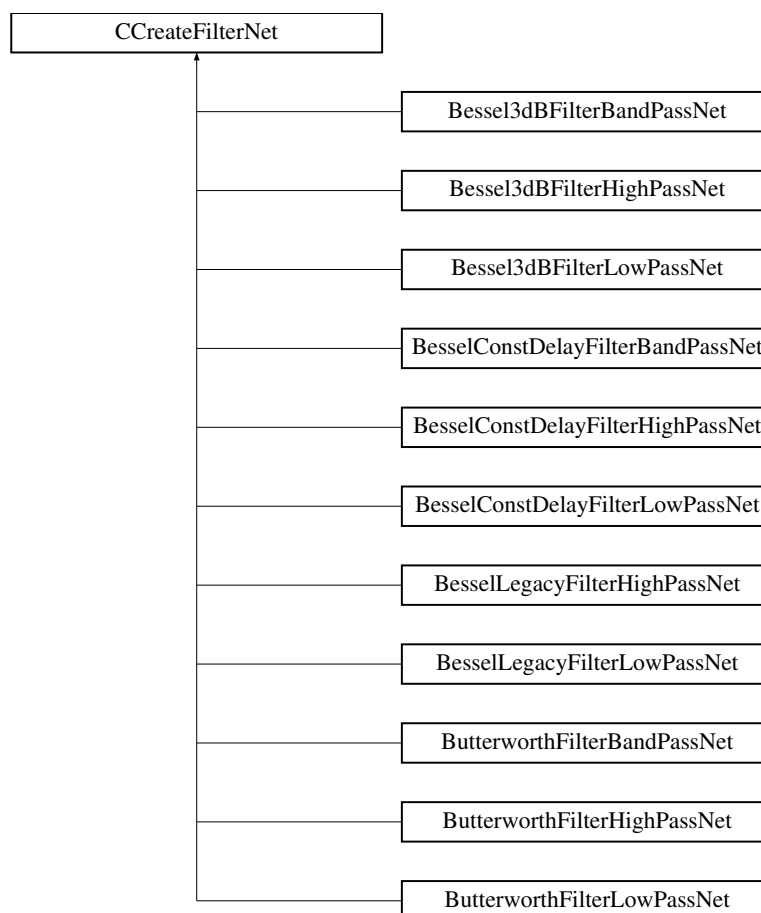
11.16.3 Property Documentation

11.16.3.1 CMosMea `CCMOSMea_FunctionNet^ CMosMea [get]`

11.16.3.2 Stimulus `CStimulusFunctionNet^ Stimulus [get]`

11.17 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](#) ()
- CCreateFilter * [GetCpp](#) ()

Static Public Member Functions

- 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 [CutoffOrCenterFrequency](#) [get]
- double [WidthFrequency](#) [get]
- double [Scale](#) [get]

11.17.1 Constructor & Destructor Documentation

11.17.1.1 CCreateFilterNet() [1/2] `CCreateFilterNet` (

```

    int numCoefSets,
    int order,
    double sampleRate,
    double cutoffFrequency,
    double scale )

```

11.17.1.2 ~CCreateFilterNet() `~CCreateFilterNet` ()

11.17.1.3 CCreateFilterNet() [2/2] `CCreateFilterNet` (

```

    int numCoefSets,
    CCreateFilter * pCreateFilter ) [protected]

```

11.17.2 Member Function Documentation

11.17.2.1 FindFilter() `static int FindFilter` (

```

    array< array< uint64_t >^>^ coef,
    array< CCreateFilterNet^>^ param,
    CFilterCoefficientsNet::s_FilterAttributesNet^ FiltAttr,
    bool DoMCSLegacyCompare ) [static]

```

11.17.2.2 GetBiQuad() `CFilterCoefficientsNet ^ GetBiQuad` (

```

    int index )

```

11.17.2.3 GetBiQuads() `array<CFilterCoefficientsNet^> ^ GetBiQuads` ()

11.17.2.4 GetCpp() `CCreateFilter* GetCpp` ()

11.17.3 Property Documentation

11.17.3.1 CutoffOrCenterFrequency `double CutoffOrCenterFrequency [get]`

11.17.3.2 NumCoefSets `int NumCoefSets [get]`

11.17.3.3 Order `int Order [get]`

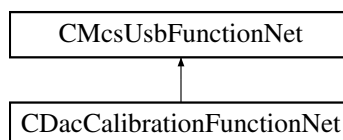
11.17.3.4 SampleRate `double SampleRate [get]`

11.17.3.5 Scale `double Scale [get]`

11.17.3.6 WidthFrequency `double WidthFrequency [get]`

11.18 CDacCalibrationFunctionNet Class Reference

Inheritance diagram for CDacCalibrationFunctionNet:



Public Member Functions

- [CDacCalibrationFunctionNet](#) ([CMcsUsbNet](#)[^] mcsusb, [CMcsUsbFunctionPointerContainer](#)[^] pDac↔ CalibrationFunctionPointerContainer)
Initializes a new instance of the [CDacCalibrationFunctionNet](#) class.
- [CDacCalibrationFunctionNet](#) ([CMcsUsbNet](#)[^] mcsusb)
- virtual [~CDacCalibrationFunctionNet](#) ()
- [!CDacCalibrationFunctionNet](#) ()
- void [SetDacOffset](#) (uint16_t dacChannel, int32_t offset)
Sets the offset of a DAC channel.
- int32_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**11.18.1 Detailed Description****11.18.2 Constructor & Destructor Documentation**

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

Initializes a new instance of the `CDacCalibrationFunctionNet` class.

11.18.2.2 CDacCalibrationFunctionNet() [2/2] `CDacCalibrationFunctionNet (CMcsUsbNet^ mcsusb)`

11.18.2.3 ~CDacCalibrationFunctionNet() `virtual ~CDacCalibrationFunctionNet () [virtual]`

11.18.2.4 "!CDacCalibrationFunctionNet() `!CDacCalibrationFunctionNet ()`

11.18.3 Member Function Documentation

11.18.3.1 BurnDacOffset() `void BurnDacOffset (uint16_t dacChannel)`

Writes the offset of a DAC channel to permanent memory.

Parameters

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

11.18.3.2 GetDacOffset() `int32_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.

11.18.3.3 SetDacOffset() `void SetDacOffset (`
 `uint16_t dacChannel,`
 `int32_t offset)`

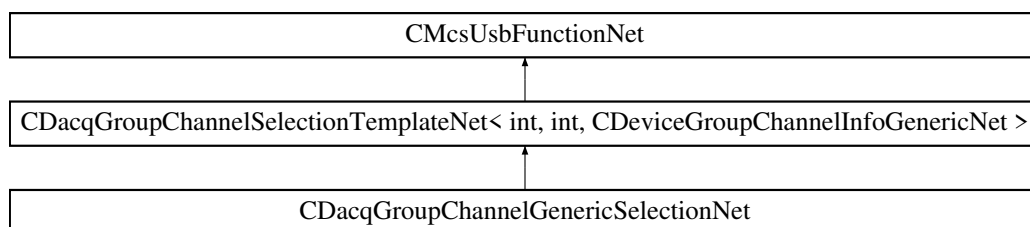
Sets the offset of a DAC channel.

Parameters

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

11.19 CDacqGroupChannelGenericSelectionNet Class Reference

Inheritance diagram for CDacqGroupChannelGenericSelectionNet:



Public Member Functions

- [CDacqGroupChannelGenericSelectionNet](#) ([CMcsUsbNet](#)[^] mcsusb)

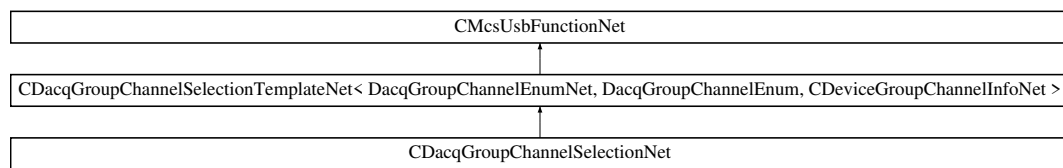
Additional Inherited Members

11.19.1 Constructor & Destructor Documentation

11.19.1.1 CDacqGroupChannelGenericSelectionNet() [CDacqGroupChannelGenericSelectionNet](#) ([CMcsUsbNet](#)[^] mcsusb)

11.20 CDacqGroupChannelSelectionNet Class Reference

Inheritance diagram for CDacqGroupChannelSelectionNet:



Public Member Functions

- [CDacqGroupChannelSelectionNet](#) ([CMcsUsbNet](#)[^] mcsusb)

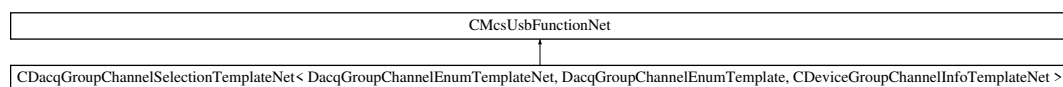
Additional Inherited Members

11.20.1 Constructor & Destructor Documentation

11.20.1.1 CDacqGroupChannelSelectionNet() [CDacqGroupChannelSelectionNet](#) (
[CMcsUsbNet](#)[^] mcsusb)

11.21 CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet > Class Template Reference

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



Public Member Functions

- [CDacqGroupChannelSelectionTemplateNet](#) ([CMcsUsbNet](#)[^] 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 >^ Enabled↔ ChannelsBitMap)
- void [EnableChannelsInGroup](#) (DacqGroupChannelEnumTemplateNet GroupID, List< bool >^ Enabled↔ ChannelsBitMap, uint32_t virtualDevice)
- List< bool >^ [GetEnabledChannelsInGroup](#) (DacqGroupChannelEnumTemplateNet GroupID)
- List< bool >^ [GetEnabledChannelsInGroup](#) (DacqGroupChannelEnumTemplateNet GroupID, uint32_t virtualDevice)
- [SampleSizeNet GetGroupSampleSize](#) (DacqGroupChannelEnumTemplateNet GroupID)
- [SampleSizeNet GetGroupSampleSize](#) (DacqGroupChannelEnumTemplateNet GroupID, uint32_t virtualDevice)
- List< CDeviceGroupChannelInfoTemplateNet^>^ [GetDeviceGroupChannelInfos](#) ()
- List< CDeviceGroupChannelInfoTemplateNet^>^ [GetDeviceGroupChannelInfos](#) (uint32_t virtualDevice)

Additional Inherited Members

11.21.1 Constructor & Destructor Documentation

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

11.21.2 Member Function Documentation

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

11.21.2.2 EnableChannelsInGroup() [2/2] void [EnableChannelsInGroup](#) (DacqGroupChannelEnumTemplateNet *GroupID*, List< bool >^ *EnabledChannelsBitMap*, uint32_t *virtualDevice*)

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

11.21.2.4 GetDeviceGroupChannelInfos() [2/2] List<[CDeviceGroupChannelInfoTemplateNet](#)^>^ [GetDeviceGroupChannelInfos](#) (uint32_t *virtualDevice*)

11.21.2.5 GetEnabledChannelsInGroup() [1/2] `List<bool> ^ GetEnabledChannelsInGroup (`
`DacqGroupChannelEnumTemplateNet GroupID)`

11.21.2.6 GetEnabledChannelsInGroup() [2/2] `List<bool> ^ GetEnabledChannelsInGroup (`
`DacqGroupChannelEnumTemplateNet GroupID,`
`uint32_t virtualDevice)`

11.21.2.7 GetGroupID() [1/2] `DacqGroupChannelEnumTemplateNet GetGroupID (`
`uint32_t Index)`

11.21.2.8 GetGroupID() [2/2] `DacqGroupChannelEnumTemplateNet GetGroupID (`
`uint32_t Index,`
`uint32_t virtualDevice)`

11.21.2.9 GetGroupNumberOfChannels() [1/2] `uint32_t GetGroupNumberOfChannels (`
`DacqGroupChannelEnumTemplateNet GroupID)`

11.21.2.10 GetGroupNumberOfChannels() [2/2] `uint32_t GetGroupNumberOfChannels (`
`DacqGroupChannelEnumTemplateNet GroupID,`
`uint32_t virtualDevice)`

11.21.2.11 GetGroupSampleSize() [1/2] `SampleSizeNet GetGroupSampleSize (`
`DacqGroupChannelEnumTemplateNet GroupID)`

11.21.2.12 GetGroupSampleSize() [2/2] `SampleSizeNet GetGroupSampleSize (`
`DacqGroupChannelEnumTemplateNet GroupID,`
`uint32_t virtualDevice)`

11.21.2.13 GetGroupType() [1/2] `DacqMeaGroupTypeEnumNet GetGroupType (`
`DacqGroupChannelEnumTemplateNet GroupID)`

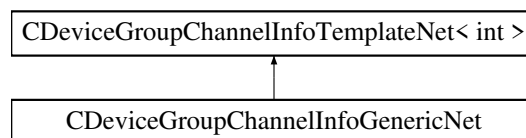
11.21.2.14 GetGroupType() [2/2] [DacqMeaGroupTypeEnumNet](#) GetGroupType (
 DacqGroupChannelEnumTemplateNet GroupID,
 uint32_t virtualDevice)

11.21.2.15 GetNumberOfSupportedGroups() [1/2] uint32_t GetNumberOfSupportedGroups ()

11.21.2.16 GetNumberOfSupportedGroups() [2/2] uint32_t GetNumberOfSupportedGroups (
 uint32_t virtualDevice)

11.22 CDeviceGroupChannelInfoGenericNet Class Reference

Inheritance diagram for CDeviceGroupChannelInfoGenericNet:



Public Member Functions

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

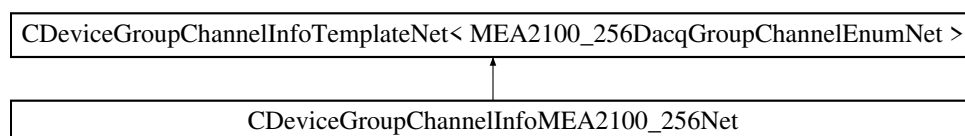
Additional Inherited Members

11.22.1 Constructor & Destructor Documentation

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

11.23 CDeviceGroupChannelInfoMEA2100_256Net Class Reference

Inheritance diagram for CDeviceGroupChannelInfoMEA2100_256Net:



Public Member Functions

- [CDeviceGroupChannelInfoMEA2100_256Net](#) ([MEA2100_256DacqGroupChannelEnumNet](#) id, int channels, [DacqMeaGroupTypeEnumNet](#) type)

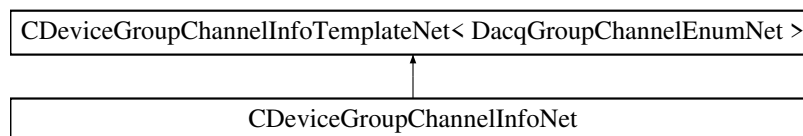
Additional Inherited Members

11.23.1 Constructor & Destructor Documentation

11.23.1.1 CDeviceGroupChannelInfoMEA2100_256Net() [CDeviceGroupChannelInfoMEA2100_256Net](#) ([MEA2100_256DacqGroupChannelEnumNet](#) id, int channels, [DacqMeaGroupTypeEnumNet](#) type)

11.24 CDeviceGroupChannelInfoNet Class Reference

Inheritance diagram for CDeviceGroupChannelInfoNet:



Public Member Functions

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

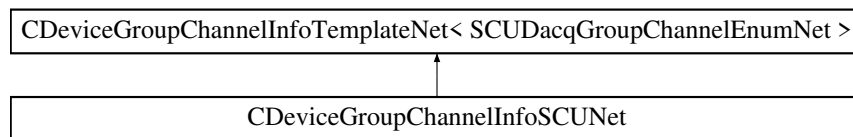
Additional Inherited Members

11.24.1 Constructor & Destructor Documentation

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

11.25 CDeviceGroupChannelInfoSCUNet Class Reference

Inheritance diagram for CDeviceGroupChannelInfoSCUNet:



Public Member Functions

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

Additional Inherited Members

11.25.1 Constructor & Destructor Documentation

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

11.26 CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumTemplateNet > Class Template Reference

Public Member Functions

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

Public Attributes

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

11.26.1 Constructor & Destructor Documentation

11.26.1.1 CDeviceGroupChannelInfoTemplateNet() `CDeviceGroupChannelInfoTemplateNet` (
`DacqGroupChannelEnumTemplateNet` *id*,
`int` *channels*,
`DacqMeaGroupTypeEnumNet` *type*)

11.26.2 Member Data Documentation

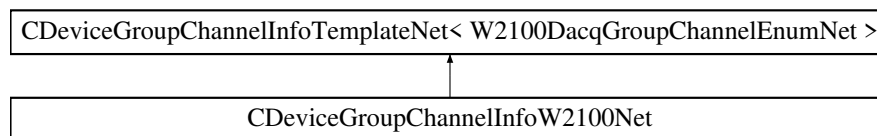
11.26.2.1 GroupID `DacqGroupChannelEnumTemplateNet` GroupID

11.26.2.2 GroupType `DacqMeaGroupTypeEnumNet` GroupType

11.26.2.3 NumberOfChannels `int` NumberOfChannels

11.27 CDeviceGroupChannelInfoW2100Net Class Reference

Inheritance diagram for CDeviceGroupChannelInfoW2100Net:



Public Member Functions

- `CDeviceGroupChannelInfoW2100Net` (`W2100DacqGroupChannelEnumNet` *id*, `int` *channels*, `DacqMeaGroupTypeEnumNet` *type*)

Additional Inherited Members

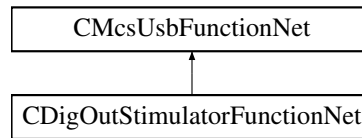
11.27.1 Constructor & Destructor Documentation

11.27.1.1 CDeviceGroupChannelInfoW2100Net() `CDeviceGroupChannelInfoW2100Net` (
`W2100DacqGroupChannelEnumNet` *id*,
`int` *channels*,
`DacqMeaGroupTypeEnumNet` *type*)

11.28 CDigOutStimulatorFunctionNet Class Reference

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

Inheritance diagram for [CDigOutStimulatorFunctionNet](#):



Public Member Functions

- [CDigOutStimulatorFunctionNet](#) ([CMcsUsbNet](#)[^] mcsusb, [CMcsUsbFunctionPointerContainer](#)[^] pDigOutStimulatorFunctionPointerContainer)
Initializes a new instance of the [CDigOutStimulatorFunctionNet](#) class.
- [CDigOutStimulatorFunctionNet](#) ([CMcsUsbNet](#)[^] mcsusb)
- virtual [~CDigOutStimulatorFunctionNet](#) ()
- [!CDigOutStimulatorFunctionNet](#) ()
- 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](#)[^] 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

11.28.1 Detailed Description

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

11.28.2 Constructor & Destructor Documentation

11.28.2.1 CDigOutStimulatorFunctionNet() [1/2] `CDigOutStimulatorFunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ pDigOutStimulatorFunctionPointerContainer)`

Initializes a new instance of the `CDigOutStimulatorFunctionNet` class.

11.28.2.2 CDigOutStimulatorFunctionNet() [2/2] `CDigOutStimulatorFunctionNet (CMcsUsbNet^ mcsusb)`

11.28.2.3 ~CDigOutStimulatorFunctionNet() `virtual ~CDigOutStimulatorFunctionNet ()` [virtual]

11.28.2.4 !CDigOutStimulatorFunctionNet() `!CDigOutStimulatorFunctionNet ()`

11.28.3 Member Function Documentation

11.28.3.1 ClearChannel() `void ClearChannel (int32_t NrChannel)`

clear stimulation pattern

Parameters

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

11.28.3.2 GetGlobalRepeat() `bool GetGlobalRepeat (int32_t NrChannel)`

get repeat whole stimulation pattern

Parameters

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

Returns

current value

11.28.3.3 GetNumberOfChannels() `int32_t GetNumberOfChannels ()`

get the number of channels available on the device

Returns

the number of channels

11.28.3.4 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)

11.28.3.5 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)

11.28.3.6 PrepareChannelData() `CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData ^ Prepare↔ ChannelData (`

```
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

11.28.3.7 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

11.28.3.8 SetGlobalRepeat() void SetGlobalRepeat (
int32_t *NrChannel*,
bool *value*)

set repeat whole stimulation pattern

Parameters

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

11.28.3.9 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)

11.28.3.10 SetStopTriggerSlope() void SetStopTriggerSlope (
 int32_t *NrChannel*,
 DigitalStimulatorTriggerSlopeEnumNet *Condition*)

sets stop condition of digital out stimulator

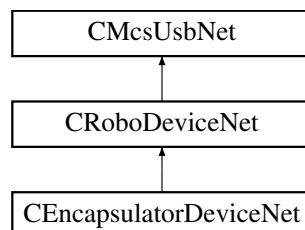
Parameters

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

11.29 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

11.29.1 Detailed Description

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

11.29.2 Constructor & Destructor Documentation

11.29.2.1 CEncapsulatorDeviceNet() `CEncapsulatorDeviceNet (void)`

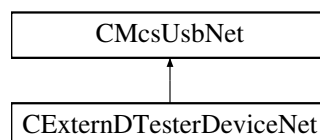
11.29.3 Member Function Documentation

11.29.3.1 GetRoboFluidDevice() `CRoboFluidDeviceNet ^ GetRoboFluidDevice ()`

11.30 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](#) ()
- [ICExternDTesterDeviceNet](#) ()
- `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

11.30.1 Detailed Description

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

11.30.2 Constructor & Destructor Documentation

11.30.2.1 CExternDTesterDeviceNet() `CExternDTesterDeviceNet ()`

Initializes a new instance of the `CExternDTesterDeviceNet` class.

11.30.2.2 ~CExternDTesterDeviceNet() `virtual ~CExternDTesterDeviceNet () [virtual]`**11.30.2.3 !CExternDTesterDeviceNet()** `!CExternDTesterDeviceNet ()`**11.30.3 Member Function Documentation****11.30.3.1 Read()** `array<uint8_t> ^ Read (`
`int configString_Length)`

Reads the config string from the device.

Parameters

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

Returns

The config string.

11.30.3.2 Read2() `String ^ Read2 ()`

Reads the config string from the device.

Returns

The config string.

11.30.3.3 Write() `void Write (`
`array< uint8_t >^ configString)`

Reads the config string from the device.

Parameters

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

11.30.3.4 Write2() `void Write2 (String^ configString)`

Reads the config string from the device.

Parameters

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

11.31 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 DoMCSLegacyCompare)
- uint64_t [GetUIntB](#) (int index, [s_FilterAttributesNet](#)^ FiltAttr)
- uint64_t [GetUIntA](#) (int index, [s_FilterAttributesNet](#)^ FiltAttr)
- array< uint64_t >^ [GetUints](#) ([s_FilterAttributesNet](#)^ FiltAttr)

Properties

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

11.31.1 Constructor & Destructor Documentation

11.31.1.1 CFilterCoefficientsNet() [1/4] [CFilterCoefficientsNet](#) ()

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

11.31.1.3 CFilterCoefficientsNet() [3/4] `CFilterCoefficientsNet` (
double *b0*,
double *b1*,
double *a1*)

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

11.31.1.5 ~CFilterCoefficientsNet() `~CFilterCoefficientsNet` ()

11.31.2 Member Function Documentation

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

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

11.31.2.3 GetUints() `array<uint64_t> ^ GetUints` (
`s_FilterAttributesNet`^ *FiltAttr*)

11.31.2.4 IsEqual() `bool IsEqual` (
array< uint64_t >^ *coefficients*,
`s_FilterAttributesNet`^ *FiltAttr*,
bool *DoMCSLegacyCompare*)

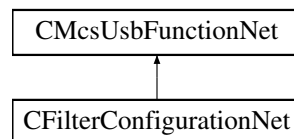
11.31.3 Property Documentation

11.31.3.1 A `array< double>^ A [get]`

11.31.3.2 B `array< double>^ B [get]`

11.32 CFilterConfigurationNet Class Reference

Inheritance diagram for CFilterConfigurationNet:



Public Member Functions

- [CFilterConfigurationNet](#) ([CMcsUsbNet](#)^ mcsusb)
- void [SetFilterParameter](#) ([DacqGroupChannelEnumNet](#) GroupID, uint32_t FilterNumber, [CFilterCoefficientsNet](#)^ Coefficients, [CFilterPropertyNet](#)^ FilterProp)
- void [SetFilterParameter](#) ([DacqGroupChannelEnumNet](#) GroupID, uint32_t FilterNumber, [CFilterCoefficientsNet](#)^ CoefficientsSet1, [CFilterCoefficientsNet](#)^ CoefficientsSet2, [CFilterPropertyNet](#)^ 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](#) ^ [GetFilterAttributes](#) ([DacqGroupChannelEnumNet](#) GroupID, uint32_t FilterNumber)

Static Public Member Functions

- static [CFilterCoefficientsNet::s_FilterAttributesNet](#) ^ [GetDefaultFilterAttributes](#) ()

Additional Inherited Members

11.32.1 Constructor & Destructor Documentation

11.32.1.1 CFilterConfigurationNet() `CFilterConfigurationNet (`
`CMcsUsbNet^ mcsusb)`

11.32.2 Member Function Documentation

11.32.2.1 EraseFilterParameterPermanent() `void EraseFilterParameterPermanent (`
`DacqGroupChannelEnumNet GroupID,`
`uint32_t FilterNumber)`

11.32.2.2 GetDefaultFilterAttributes() `static CFilterCoefficientsNet::s_FilterAttributesNet ^`
`GetDefaultFilterAttributes () [static]`

11.32.2.3 GetFilterAttributes() [1/2] `CFilterCoefficientsNet::s_FilterAttributesNet ^ GetFilter←`
`Attributes (`
`DacqGroupChannelEnumNet GroupID,`
`uint32_t FilterNumber)`

11.32.2.4 GetFilterAttributes() [2/2] `uint32_t GetFilterAttributes (`
`DacqGroupChannelEnumNet GroupID,`
`uint32_t FilterNumber,`
`FilterAttributeEnumNet index)`

11.32.2.5 GetHighpassFilterEnable() `bool GetHighpassFilterEnable ()`

11.32.2.6 ResetHighpassFilter() `void ResetHighpassFilter ()`

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

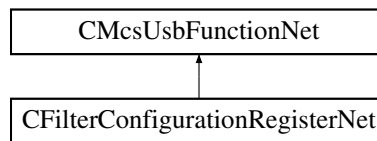
11.32.2.8 SetFilterParameter() [2/2] void SetFilterParameter (
 DacqGroupChannelEnumNet GroupID,
 uint32_t FilterNumber,
 CFilterCoefficientsNet^ CoefficientsSet1,
 CFilterCoefficientsNet^ CoefficientsSet2,
 CFilterPropertyNet^ FilterProp)

11.32.2.9 SetFilterParameterPermanent() void SetFilterParameterPermanent (
 DacqGroupChannelEnumNet GroupID,
 uint32_t FilterNumber)

11.32.2.10 SetHighpassFilterEnable() void SetHighpassFilterEnable (
 bool enable)

11.33 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

11.33.1 Constructor & Destructor Documentation

11.33.1.1 CFilterConfigurationRegisterNet() `CFilterConfigurationRegisterNet (CMcsUsbNet^ mcsusb)`

11.33.2 Member Function Documentation

11.33.2.1 EraseFilterParameterPermanent() [1/2] `void EraseFilterParameterPermanent (uint32_t FilterCoefDmaReg, uint32_t FilterInfoDmaReg, uint32_t EEPROMBase, uint32_t EEPROMSize)`

11.33.2.2 EraseFilterParameterPermanent() [2/2] `void EraseFilterParameterPermanent (uint32_t FilterCoefSet1DmaReg, uint32_t FilterCoefSet2DmaReg, uint32_t FilterInfoDmaReg, uint32_t EEPROMBase, uint32_t EEPROMSize)`

11.33.2.3 SetFilterParameter() [1/2] `void SetFilterParameter (uint32_t FilterCoefRegBase, CFilterCoefficientsNet^ Coefficients, uint32_t FilterInfoRegBase, CFilterPropertyNet^ FilterProp)`

11.33.2.4 SetFilterParameter() [2/2] `void SetFilterParameter (uint32_t FilterCoefSet1RegBase, CFilterCoefficientsNet^ CoefficientsSet1, uint32_t FilterCoefSet2RegBase, CFilterCoefficientsNet^ CoefficientsSet2, uint32_t FilterInfoRegBase, CFilterPropertyNet^ FilterProp)`

11.33.2.5 SetFilterParameterPermanent() [1/2] `void SetFilterParameterPermanent (uint32_t FilterCoefRegBase, uint32_t FilterCoefDmaReg, uint32_t FilterInfoRegBase, uint32_t FilterInfoDmaReg, uint32_t EEPROMBase, uint32_t EEPROMSize)`

11.33.2.6 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)`

11.34 CFilterPropertyNet Class Reference

Public Member Functions

- [CFilterPropertyNet](#) (uint32_t CornerFrequenzymHz, uint32_t [Order](#), [FilterBandEnumNet](#) [FilterBand](#), [FilterFamilyEnumNet](#) [FilterFamily](#), [FilterTypeEnumNet](#) [FilterType](#), bool Active)
- [~CFilterPropertyNet](#) ()
- virtual System::String ^ [ToString](#) () override

Properties

- uint32_t [CornerFrequencymHz](#) [get]
- double [CornerFrequency](#) [get]
- uint32_t [Order](#) [get]
- [FilterBandEnumNet](#) [FilterBand](#) [get]
- [FilterFamilyEnumNet](#) [FilterFamily](#) [get]
- [FilterTypeEnumNet](#) [FilterType](#) [get]
- bool [FilterActive](#) [get]

11.34.1 Constructor & Destructor Documentation

11.34.1.1 CFilterPropertyNet() `CFilterPropertyNet (`
`uint32_t CornerFrequenzymHz,`
`uint32_t Order,`
`FilterBandEnumNet FilterBand,`
`FilterFamilyEnumNet FilterFamily,`
`FilterTypeEnumNet FilterType,`
`bool Active)`

11.34.1.2 ~CFilterPropertyNet() `~CFilterPropertyNet ()`

11.34.2 Member Function Documentation

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

11.34.3 Property Documentation

11.34.3.1 CornerFrequency `double CornerFrequency [get]`

11.34.3.2 CornerFrequencymHz `uint32_t CornerFrequencymHz [get]`

11.34.3.3 FilterActive `bool FilterActive [get]`

11.34.3.4 FilterBand `FilterBandEnumNet FilterBand [get]`

11.34.3.5 FilterFamily `FilterFamilyEnumNet FilterFamily [get]`

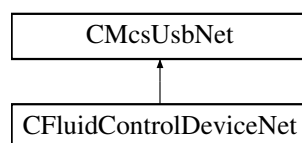
11.34.3.6 FilterType `FilterTypeEnumNet FilterType [get]`

11.34.3.7 Order `uint32_t Order [get]`

11.35 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](#)[^] [McsBus_VoltageMode](#) [get]

Additional Inherited Members

11.35.1 Detailed Description

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

11.35.2 Constructor & Destructor Documentation

11.35.2.1 CFluidControlDeviceNet() `CFluidControlDeviceNet ()`

Initialize a new instance of the `CFluidControlDeviceNet` class.

11.35.2.2 ~CFluidControlDeviceNet() `~CFluidControlDeviceNet ()`

Default destructor.

11.35.3 Member Function Documentation

11.35.3.1 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.
----------------	------------------------------

11.35.3.2 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.

11.35.3.3 GetDigin() `unsigned int GetDigin ()`

Reads the digital input.

Returns

The bit pattern of the state of the digital inputs.

11.35.3.4 GetDigout() `unsigned int GetDigout ()`

Gets the state of the digital output.

Returns

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

11.35.3.5 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.

11.35.3.6 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.

11.35.3.7 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

11.35.3.8 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.

11.35.3.9 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.

11.35.3.10 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.

11.35.3.11 GetValve() `unsigned int GetValve ()`

Gets the state of the valves.

Returns

The current state of the valves as a bit pattern.

11.35.3.12 SetDigout() `void SetDigout (`
`unsigned int value)`

Define the pattern on the Digital Output.

Parameters

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

11.35.3.13 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.

11.35.3.14 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.
--------------	--------------------------

11.35.3.15 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>	proportional constant for the Thermocouple in Nanovolt per Kelvin.

11.35.3.16 SetValve() `void SetValve (unsigned int value)`

Open or Close valves.

Parameters

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

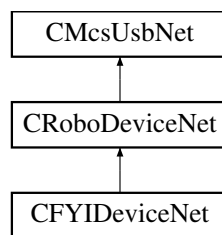
11.35.4 Property Documentation

11.35.4.1 McsBus_VoltageMode `CMcsBus_VoltageModeNet^ McsBus_VoltageMode [get]`

11.36 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](#)^ [FYITemp](#) [get]
- [CRobo_FYIProgram_FunctionNet](#)^ [FYIProgram](#) [get]
- [CMcsBus_SensorNet](#)^ [Sensor](#) [get]

Additional Inherited Members

11.36.1 Detailed Description

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

11.36.2 Constructor & Destructor Documentation

11.36.2.1 [CFYIDeviceNet](#)() [CFYIDeviceNet](#) (
void)

11.36.3 Property Documentation

11.36.3.1 [FYIProgram](#) [CRobo_FYIProgram_FunctionNet](#)^ [FYIProgram](#) [get]

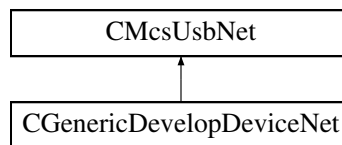
11.36.3.2 [FYITemp](#) [CRobo_FYITemp_FunctionNet](#)^ [FYITemp](#) [get]

11.36.3.3 Sensor `CMcsBus_SensorNet`^ Sensor [get]

11.37 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.

- `template<typename C >`
`void VendorOutRequest (uint8_t request, uint16_t value, uint16_t index, array< C >^ buffer)`
- `template<typename C >`
`array< C > ^ VendorInRequest (uint8_t request, uint16_t value, uint16_t index, int size)`
- `array< uint8_t > ^ FindEndpoints (uint8_t type, uint8_t direction)`
- `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

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

Additional Inherited Members

11.37.1 Detailed Description

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

11.37.2 Constructor & Destructor Documentation

11.37.2.1 CGenericDevelopDeviceNet() [CGenericDevelopDeviceNet](#) (
 void)

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

11.37.2.2 ~CGenericDevelopDeviceNet() [~CGenericDevelopDeviceNet](#) (
 void)

11.37.3 Member Function Documentation

11.37.3.1 ClosePipe() void ClosePipe (
 IntPtr *pipeHandle*)

Close a Pipe to an USB Endpoint.

Parameters

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

11.37.3.2 FindEndpoints() array<uint8_t> ^ FindEndpoints (
 uint8_t *type*,
 uint8_t *direction*)

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

11.37.3.4 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.

11.37.3.5 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.

11.37.3.6 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.

11.37.3.7 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.

11.37.3.8 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.

```
11.37.3.9 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.

11.37.3.10 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.

11.37.3.11 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.

11.37.3.12 ResetPipe() `void ResetPipe (`
 `IntPtr pipeHandle)`

Reset a Pipe to an USB Endpoint.

Parameters

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

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

11.37.3.14 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.
---------------	---------------------

11.37.3.15 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.
---------------	---------------------

11.37.3.16 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.
---------------	---------------------

11.37.3.17 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.
---------------	---------------------

11.37.3.18 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.
---------------	---------------------

11.37.3.19 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.
---------------	---------------------

11.37.3.20 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.
--------------	---------------------------

11.37.3.21 VendorInRequest() `array<C> ^ VendorInRequest (`
 `uint8_t request,`
 `uint16_t value,`
 `uint16_t index,`
 `int size)`

11.37.3.22 VendorOutRequest() `void VendorOutRequest (`
 `uint8_t request,`
 `uint16_t value,`
 `uint16_t index,`
 `array< C >^ buffer)`

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

Write data to an USB Endpoint.

Parameters

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

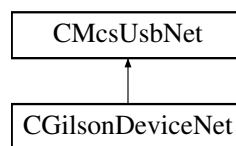
Parameters

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

11.38 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

11.38.1 Detailed Description

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

11.38.2 Constructor & Destructor Documentation

11.38.2.1 CGilsonDeviceNet() `CGilsonDeviceNet (`
`void)`

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

11.38.2.2 ~CGilsonDeviceNet() `~CGilsonDeviceNet (`
`void)`

11.38.3 Member Function Documentation

11.38.3.1 ConnectSlave() `void ConnectSlave (`
`byte ID)`

11.38.3.2 GetLastAnswer() `String ^ GetLastAnswer ()`

11.38.3.3 SendBuffered() `void SendBuffered (`
`String^ command)`

11.38.3.4 SendImmediate() `void SendImmediate (`
`wchar_t command)`

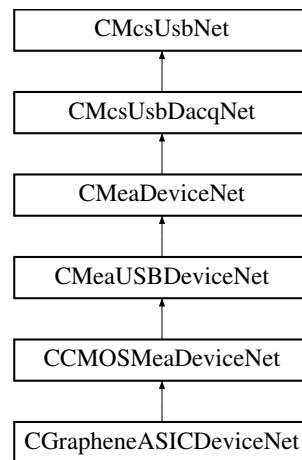
11.38.3.5 SendImmediateGetResponse() `String ^ SendImmediateGetResponse (`
`wchar_t command)`

11.38.4 Member Data Documentation

11.38.4.1 m_pGilsonDevice `CGilsonDevice* m_pGilsonDevice [protected]`

11.39 CGrapheneASICDeviceNet Class Reference

Inheritance diagram for CGrapheneASICDeviceNet:



Public Member Functions

- [CGrapheneASICDeviceNet](#) (void)
- [~CGrapheneASICDeviceNet](#) ()
- void [SetBaseSamplerate](#) (int BaseSamplerate) override
- int [GetBaseSamplerate](#) () override
- array< int > ^ [GetAvailableBaseSamplerates](#) () override
- void [SetRegionOfInterests](#) (System::Collections::Generic::Dictionary< int, [CCMOSMeaDeviceNet::CRegionOfInterestRect](#)>^ rois) override

Additional Inherited Members

11.39.1 Constructor & Destructor Documentation

11.39.1.1 CGrapheneASICDeviceNet() [CGrapheneASICDeviceNet](#) (void)

11.39.1.2 ~CGrapheneASICDeviceNet() [~CGrapheneASICDeviceNet](#) ()

11.39.2 Member Function Documentation

11.39.2.1 GetAvailableBaseSamplerates() `array<int> ^ GetAvailableBaseSamplerates () [override], [virtual]`

Reimplemented from [CCMOSMeaDeviceNet](#).

11.39.2.2 GetBaseSamplerate() `int GetBaseSamplerate () [override], [virtual]`

Reimplemented from [CCMOSMeaDeviceNet](#).

11.39.2.3 SetBaseSamplerate() `void SetBaseSamplerate (int BaseSamplerate) [override], [virtual]`

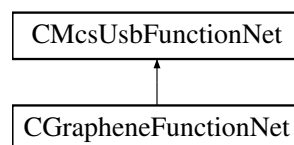
Reimplemented from [CCMOSMeaDeviceNet](#).

11.39.2.4 SetRegionOfInterests() `void SetRegionOfInterests (System::Collections::Generic::Dictionary< int, CCMOSMeaDeviceNet::CRegionOfInterestRect ^>^ rois) [override]`

11.40 CGrapheneFunctionNet Class Reference

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

Inheritance diagram for CGrapheneFunctionNet:



Public Member Functions

- [CGrapheneFunctionNet](#) ([CMcsUsbNet](#)^ mcsusb, [CMcsUsbFunctionPointerContainer](#)^ pGrapheneFunctionPointerContainer)
Initializes a new instance of the [CGrapheneFunctionNet](#) class.
- [CGrapheneFunctionNet](#) ([CMcsUsbNet](#)^ 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 [SetVdsVgs](#) (int32_t Vds, int32_t Vgs)
Sets Vds and Vgs
- void [SetVdsVgs](#) (uint32_t Headstage, int32_t Vds, int32_t Vgs)
Sets Vds and Vgs
- int32_t [GetCur2VolResistance](#) ()
Gets the resistance of the current to voltage converter
- int32_t [GetCur2VolResistance](#) (uint32_t Headstage)

- Gets the resistance of the current to voltage converter*
- `int32_t GetADCOffset (uint32_t Channel)`
- Gets the ADC offset*
- `int32_t GetADCOffset (uint32_t Headstage, uint32_t Channel)`
- Gets the ADC offset*
- `void SetADCOffset (uint32_t Channel, int32_t offset)`
- Sets the ADC offset*
- `void SetADCOffset (uint32_t Headstage, uint32_t Channel, int32_t offset)`
- Sets the ADC offset*
- `int32_t GetADCGain (uint32_t Channel)`
- Gets the ADC offset*
- `int32_t GetADCGain (uint32_t Headstage, uint32_t Channel)`
- Gets the ADC offset*
- `void SetADCGain (uint32_t Channel, int32_t gain)`
- Sets the ADC offset*
- `void SetADCGain (uint32_t Headstage, uint32_t Channel, int32_t gain)`
- Sets the ADC offset*
- `void GetVdsVgs ([System::Runtime::InteropServices::Out]int32_t% Vds, [System::Runtime::InteropServices::Out]int32_t% Vgs)`
- Gets Vds and Vgs*
- `void GetVdsVgs (uint32_t Headstage, [System::Runtime::InteropServices::Out]int32_t% Vds, [System::Runtime::InteropServices::Out]int32_t% Vgs)`
- Gets Vds and Vgs*

Additional Inherited Members

11.40.1 Detailed Description

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

11.40.2 Constructor & Destructor Documentation

11.40.2.1 CGrapheneFunctionNet() [1/2] `CGrapheneFunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ pGrapheneFunctionPointerContainer)`

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

11.40.2.2 CGrapheneFunctionNet() [2/2] `CGrapheneFunctionNet (CMcsUsbNet^ mcsusb)`

11.40.2.3 `~CGrapheneFunctionNet()` `virtual ~CGrapheneFunctionNet () [virtual]`

11.40.2.4 `!CGrapheneFunctionNet()` `!CGrapheneFunctionNet ()`

11.40.3 Member Function Documentation

11.40.3.1 `GetADCGain()` [1/2] `int32_t GetADCGain (`
`uint32_t Channel)`

Gets the ADC offset

Parameters

<i>Channel</i>	The channel to query.
----------------	-----------------------

Returns

The gain

11.40.3.2 `GetADCGain()` [2/2] `int32_t GetADCGain (`
`uint32_t Headstage,`
`uint32_t Channel)`

Gets the ADC offset

Parameters

<i>Headstage</i>	The headstage to query.
<i>Channel</i>	The channel to query.

Returns

The gain

11.40.3.3 `GetADCOffset()` [1/2] `int32_t GetADCOffset (`
`uint32_t Channel)`

Gets the ADC offset

Parameters

<i>Channel</i>	The channel to query.
----------------	-----------------------

Returns

The offset

11.40.3.4 GetADCOffset() [2/2] `int32_t GetADCOffset (`
 `uint32_t Headstage,`
 `uint32_t Channel)`

Gets the ADC offset

Parameters

<i>Headstage</i>	The headstage to query.
<i>Channel</i>	The channel to query.

Returns

The offset

11.40.3.5 GetCur2VolResistance() [1/2] `int32_t GetCur2VolResistance ()`

Gets the resistance of the current to voltage converter

Returns

The resistance in Ohm

11.40.3.6 GetCur2VolResistance() [2/2] `int32_t GetCur2VolResistance (`
 `uint32_t Headstage)`

Gets the resistance of the current to voltage converter

Parameters

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

Returns

The resistance in Ohm

11.40.3.7 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 in DAC Units
<i>offset_vs</i>	Vs offset in DAC Units

11.40.3.8 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 in DAC Units
<i>offset_vs</i>	Vs offset in DAC Units

11.40.3.9 GetVdsVgs() [1/2] void GetVdsVgs (
 [System::Runtime::InteropServices::Out] int32_t% *Vds*,
 [System::Runtime::InteropServices::Out] int32_t% *Vgs*)

Gets Vds and Vgs

Parameters

<i>Vds</i>	Vds in μV
<i>Vgs</i>	Vgs in μV

11.40.3.10 GetVdsVgs() [2/2] void GetVdsVgs (
 uint32_t *Headstage*,

```
[System::Runtime::InteropServices::Out] int32_t% Vds,
[System::Runtime::InteropServices::Out] int32_t% Vgs )
```

Gets Vds and Vgs

Parameters

<i>Headstage</i>	The headstage to query.
<i>Vds</i>	Vds in μV
<i>Vgs</i>	Vgs in μV

11.40.3.11 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 in μV
<i>Vs</i>	Vs in μV

11.40.3.12 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 in μV
<i>Vs</i>	Vs in μV

11.40.3.13 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 in DAC Units
<i>Vs</i>	Vs in DAC Units

11.40.3.14 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 in DAC Units
<i>Vs</i>	Vs in DAC Units

11.40.3.15 GetVoltageRange() [1/2] int32_t GetVoltageRange ()

Gets the voltage range

Returns

The voltage range in mV

11.40.3.16 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 in mV

11.40.3.17 GetVoltageReached() [1/2] bool GetVoltageReached ()

Gets the reached voltage

Returns

the reached voltage

11.40.3.18 GetVoltageReached() [2/2] `bool GetVoltageReached (`
`uint32_t Headstage)`

Gets the reached voltage

Parameters

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

Returns

The reached voltage

11.40.3.19 GetVoltageResolution() [1/2] `int32_t GetVoltageResolution ()`

Gets the voltage resolution

Returns

The voltage resolution in $\mu\text{V}/\text{digit}$

11.40.3.20 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 in $\mu\text{V}/\text{digit}$

11.40.3.21 SetADCGain() [1/2] `void SetADCGain (`
`uint32_t Channel,`
`int32_t gain)`

Sets the ADC offset

Parameters

<i>Channel</i>	The channel to query.
<i>gain</i>	The gain

11.40.3.22 SetADCGain() [2/2] `void SetADCGain (`
 `uint32_t Headstage,`
 `uint32_t Channel,`
 `int32_t gain)`

Sets the ADC offset

Parameters

<i>Headstage</i>	The headstage to query.
<i>Channel</i>	The channel to query.
<i>gain</i>	The gain

11.40.3.23 SetADCOffset() [1/2] `void SetADCOffset (`
 `uint32_t Channel,`
 `int32_t offset)`

Sets the ADC offset

Parameters

<i>Channel</i>	The channel to query.
<i>offset</i>	The offset

11.40.3.24 SetADCOffset() [2/2] `void SetADCOffset (`
 `uint32_t Headstage,`
 `uint32_t Channel,`
 `int32_t offset)`

Sets the ADC offset

Parameters

<i>Headstage</i>	The headstage to query.
<i>Channel</i>	The channel to query.
<i>offset</i>	The offset

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

Sets the DAC offset

Parameters

<i>offset_vd</i>	Vd offset in DAC Units
<i>offset_vs</i>	Vs offset in DAC Units

11.40.3.26 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 offset in DAC Units
<i>offset_vs</i>	Vs offset in DAC Units

11.40.3.27 SetVdsVgs() [1/2] `void SetVdsVgs (`
 `int32_t Vds,`
 `int32_t Vgs)`

Sets Vds and Vgs

Parameters

<i>Vds</i>	Vds in μV
<i>Vgs</i>	Vgs in μV

11.40.3.28 SetVdsVgs() [2/2] `void SetVdsVgs (`
 `uint32_t Headstage,`
 `int32_t Vds,`
 `int32_t Vgs)`

Sets Vds and Vgs

Parameters

<i>Headstage</i>	The headstage to query.
<i>Vds</i>	Vds in μV
<i>Vgs</i>	Vgs in μV

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

Sets Vd and Vs

Parameters

<i>Vd</i>	Vd in μV
<i>Vs</i>	Vs in μV

11.40.3.30 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 in μV
<i>Vs</i>	Vs in μV

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

Sets Vd and Vs

Parameters

<i>Vd</i>	Vd in DAC Units
<i>Vs</i>	Vs in DAC Units

11.40.3.32 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 in DAC Units
<i>Vs</i>	Vs in DAC Units

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

Sets the voltage range

Parameters

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

11.40.3.34 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 in mV

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

Sets the voltage resolution

Parameters

<i>resolution</i>	The voltage resolution in $\mu\text{V}/\text{digit}$
-------------------	--

11.40.3.36 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 in $\mu\text{V}/\text{digit}$

11.41 CMcsUsbDacqNet::ChannelBlockNet Class Reference

Public Member Functions

- [ChannelBlockNet](#) ([CMcsUsbDacqNet](#)[^] parent)
- int [AddBlocksAndChannels](#) ([ChannelBlockTypeNet](#) type, int nChannels, int queuesize, int threshold, [SampleSize](#) samplesize, [SampleDstSize](#) sampleDstSize, int nByteOffset, int nChannelOffset)
Create FIFO queues. ChannelBlockTypeNet determines if multiple handles, multiples queues or one queue and and channel will be used.
- int [AddBlocksAndChannels](#) ([ChannelBlockTypeNet](#) type, array< bool >[^] selectedChannels, int queuesize, int threshold, [SampleSize](#) samplesize, [SampleDstSize](#) sampleDstSize, int nByteOffset, int nChannelOffset)
Create FIFO queues. ChannelBlockTypeNet determines if multiple handles, multiples queues or one queue and and channel will be used.
- virtual void [SetSelectedChannels](#) (int nChannels, int queuesize, int threshold, [SampleSizeNet](#) samplesize, [SampleDstSizeNet](#) sampleDstSize, 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)
Create a FIFO queue per channel. Each channel will have its own FIFO and Callback function.
- virtual void [SetSelectedData](#) (int nChannels, int queuesize, int threshold, [SampleSizeNet](#) samplesize, [SampleDstSizeNet](#) sampleDstSize, 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)
Create a common FIFO queue for all channels. Use handle = 0 in the ChannelBlock_ReadFrames... functions.
- virtual int [AddSelectedChannelsQueue](#) (int nByteOffset, int nChannelOffset, int nChannels, int queuesize, int threshold, [SampleSizeNet](#) samplesize, [SampleDstSizeNet](#) sampleDstSize)
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.
- virtual int [AddSelectedChannelsQueue](#) (int nByteOffset, int nChannelOffset, array< bool >[^] selected↵Channels, int queuesize, int threshold, [SampleSizeNet](#) samplesize, [SampleDstSizeNet](#) sampleDstSize)
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.
- virtual void [SetSelectedChannelsQueue](#) (int nChannels, int queuesize, int threshold, [SampleSizeNet](#) sample-size, [SampleDstSizeNet](#) sampleDstSize, 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)
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 uint32_t [AvailFrames](#) (int handle, int queue)
Get the number of sample frames already available in the FIFO.
- virtual array< uint16_t >[^] [ReadFramesUI16](#) (int handle, int queue, int frames, [System::Runtime::Interop↵Services::Out]int% frames_ret)
Read data from a FIFO queue in uint16_t data format

- virtual array< int16_t > ^ [ReadFramesI16](#) (int handle, int queue, int frames, [System::Runtime::InteropServices::Out]int% frames_ret)
Read data from a FIFO queue in int16_t data format
- virtual array< uint32_t > ^ [ReadFramesUI32](#) (int handle, int queue, int frames, [System::Runtime::InteropServices::Out]int% frames_ret)
Read data from a FIFO queue in uint32_t data format
- virtual array< int32_t > ^ [ReadFramesI32](#) (int handle, int queue, 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 >^> ^ [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 >^> ^ [ReadAsFrameArrayI16](#) (int handle, int queue, int frames, [System::Runtime::InteropServices::Out]int% frames_ret)
Read data from a FIFO queue as array of int16_t data frame arrays
- virtual array< array< uint32_t >^> ^ [ReadAsFrameArrayUI32](#) (int handle, int queue, int frames, [System::Runtime::InteropServices::Out]int% frames_ret)
Read data from a FIFO queue as array of uint32_t data frame arrays
- virtual array< array< int32_t >^> ^ [ReadAsFrameArrayI32](#) (int handle, int queue, int frames, [System::Runtime::InteropServices::Out]int% frames_ret)
Read data from a FIFO queue as array of int32_t data frame arrays
- virtual System::Collections::Generic::Dictionary< int, array< uint16_t >^> ^ [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 >^> ^ [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 >^> ^ [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 >^> ^ [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)
- virtual uint32_t [UpdateQueueSize](#) (int handle)
- virtual void [Init](#) (int words_per_sample)
- virtual void [AddBlock](#) ([System::Runtime::InteropServices::Out]int% handle, int queuesize, int threshold, [SampleSizeNet](#) samplesize, [SampleDstSizeNet](#) sampleDstSize)
- virtual void [SetQueueSize](#) (int handle, int queuesize, int callbackThreshold)
- virtual void [SetQueueSizeAll](#) (int queuesize, int callbackThreshold)
- virtual void [AddChannel](#) (int handle, int queue, int byte_offset, int channel_offset, int channels)
- virtual void [GetChannel](#) (int handle, int queue, int channelentry, [System::Runtime::InteropServices::Out]int% totalchannels, [System::Runtime::InteropServices::Out]int% byte_offset, [System::Runtime::InteropServices::Out]int% channel_offset, [System::Runtime::InteropServices::Out]int% channels)
- virtual int [GetNumberOfQueues](#) (int handle)
- virtual void [SetCommonThreshold](#) (int common_threshold)
- virtual void [SetCheckChecksum](#) (uint32_t checksumchannels, uint32_t checksumoffset_from_end)
- virtual uint32_t [GetCheckChecksum](#) ()

Static Public Member Functions

- static int [GetNListTrue](#) (List< bool >^ enabled_channels)

11.41.1 Constructor & Destructor Documentation

11.41.1.1 ChannelBlockNet() [ChannelBlockNet](#) (
[CMcsUsbDacqNet](#)^ parent)

11.41.2 Member Function Documentation

11.41.2.1 AddBlock() virtual void AddBlock (
[System::Runtime::InteropServices::Out] int% handle,
int queuesize,
int threshold,
[SampleSizeNet](#) samplesize,
[SampleDstSizeNet](#) sampleDstSize) [virtual]

11.41.2.2 AddBlocksAndChannels() [1/2] int AddBlocksAndChannels (
[ChannelBlockTypeNet](#) type,
array< bool >^ selectedChannels,
int queuesize,
int threshold,
SampleSize samplesize,
SampleDstSize sampleDstSize,
int nByteOffset,
int nChannelOffset)

Create FIFO queues. ChannelBlockTypeNet determines if multiple handles, multiples queues or one queue and and channel will be used.

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>type</i>	Determines if multiple handles, multiples queues or one queue and and channel will be used.
-------------	---

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 samples frames the FIFO must acquire before the callback function is called.
------------------	--

Parameters

<i>samplesize</i>	size of the datawords, either 16, 24 or 32bit.
<i>sampleDstSize</i>	size of the destination datawords, either 16 or 32bit.
<i>nByteOffset</i>	Number of bytes to start with.

Parameters

<i>nChannelOffset</i>	Number of channel to start with (counted in samplesize bytes).
-----------------------	--

11.41.2.3 AddBlocksAndChannels() [2/2] `int AddBlocksAndChannels (`
 `ChannelBlockTypeNet type,`
 `int nChannels,`
 `int queuesize,`
 `int threshold,`

```

SampleSize samplesize,
SampleDstSize sampleDstSize,
int nByteOffset,
int nChannelOffset )

```

Create FIFO queues. ChannelBlockTypeNet determines if multiple handles, multiples queues or one queue and and channel will be used.

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>type</i>	Determines if multiple handles, multiples queues or one queue and and channel will be used.
-------------	---

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, 24 or 32bit.
<i>sampleDstSize</i>	size of the destination datawords, either 16 or 32bit.
<i>nByteOffset</i>	Number of bytes to start with.

Parameters

<i>nChannelOffset</i>	Number of channel to start with (counted in samplesize bytes).
-----------------------	--

11.41.2.4 AddChannel() `virtual void AddChannel (`
 `int handle,`
 `int queue,`
 `int byte_offset,`
 `int channel_offset,`
 `int channels) [virtual]`

11.41.2.5 AddSelectedChannelsQueue() `[1/2] virtual int AddSelectedChannelsQueue (`
 `int nByteOffset,`
 `int nChannelOffset,`
 `array< bool >^ selectedChannels,`
 `int queuesize,`
 `int threshold,`
 `SampleSizeNet samplesize,`
 `SampleDstSizeNet sampleDstSize) [virtual]`

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.

When using 32 bit data format, `ChannelsInBlock` is still the number of 16 bit channels per frame, as obtained from `GetChannelsInBlock`, while `nChannels` is the number of 32 bit channels to be read from the device. So when all channels from a device are read in 32 bit data format `nChannels = ChannelsInBlock/2`

Parameters

<code>nByteOffset</code>	Number of bytes to start with.
--------------------------	--------------------------------

Parameters

<code>nChannelOffset</code>	Number of channel to start with (counted in samplesize bytes).
-----------------------------	--

Parameters

<code>selectedChannels</code>	List of channels to be collected in the FIFO.
-------------------------------	---

Parameters

<code>queuesize</code>	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.

11.41.2.6 AddSelectedChannelsQueue() [2/2] `virtual int AddSelectedChannelsQueue (`
`int nByteOffset,`
`int nChannelOffset,`
`int nChannels,`
`int queuesize,`
`int threshold,`
`SampleSizeNet samplesize,`
`SampleDstSizeNet sampleDstSize) [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.

11.41.2.7 AvailFrames() `virtual uint32_t AvailFrames (`
 `int handle,`
 `int queue) [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.

11.41.2.8 GetChannel() virtual void GetChannel (
 int *handle*,
 int *queue*,
 int *channelentry*,
 [System::Runtime::InteropServices::Out] int% *totalchannels*,
 [System::Runtime::InteropServices::Out] int% *byte_offset*,
 [System::Runtime::InteropServices::Out] int% *channel_offset*,
 [System::Runtime::InteropServices::Out] int% *channels*) [virtual]

11.41.2.9 GetCheckChecksum() virtual uint32_t GetCheckChecksum () [virtual]

11.41.2.10 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.

11.41.2.11 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.

```
11.41.2.12 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.

11.41.2.13 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.

11.41.2.14 GetNListTrue() static int GetNListTrue (
 List< bool > ^ enabled_channels) [static]

11.41.2.15 GetNumberOfQueues() virtual int GetNumberOfQueues (
 int handle) [virtual]

11.41.2.16 Init() virtual void Init (
 int words_per_sample) [virtual]

11.41.2.17 ReadAsFrameArrayI16() virtual array<array<int16_t>> ^ 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.

```
11.41.2.18 ReadAsFrameArrayI32() virtual array<array<int32_t>^> ^ 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.

11.41.2.19 ReadAsFrameArrayUI16() virtual array<array<uint16_t>^> ^ 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.

11.41.2.20 ReadAsFrameArrayUI32() virtual array<array<uint32_t>^> ^ 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.

```
11.41.2.21 ReadFramesDictI16() virtual System::Collections::Generic::Dictionary<int, array<int16←
_t>^> ^ 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.

11.41.2.22 ReadFramesDictI32() `virtual System::Collections::Generic::Dictionary<int, array<int32_t>^> ^ 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.

11.41.2.23 ReadFramesDictUI16() `virtual System::Collections::Generic::Dictionary<int, array<uint16_t>^> ^ 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.

```
11.41.2.24 ReadFramesDictUI32() virtual System::Collections::Generic::Dictionary<int, array<uint32_t>^> ^ 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.

```
11.41.2.25 ReadFramesI16() virtual array<int16_t> ^ ReadFramesI16 (
    int handle,
    int queue,
    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>queue</i>	Handle of the queue.
--------------	----------------------

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.
-------------------	--

11.41.2.26 ReadFramesI32() `virtual array<int32_t> ^ ReadFramesI32 (`
 `int handle,`
 `int queue,`
 `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.
-------------------	--

11.41.2.27 ReadFramesUI16() `virtual array<uint16_t> ^ ReadFramesUI16 (`
 `int handle,`
 `int queue,`
 `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.

11.41.2.28 ReadFramesUI32() `virtual array<uint32_t> ^ ReadFramesUI32 (`
 `int handle,`
 `int queue,`
 `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.
-------------------	--

11.41.2.29 SetCheckChecksum() virtual void SetCheckChecksum (
 uint32_t *checksumchannels*,
 uint32_t *checksumoffset_from_end*) [virtual]

11.41.2.30 SetCommonThreshold() virtual void SetCommonThreshold (
 int *common_threshold*) [virtual]

11.41.2.31 SetQueueSize() virtual void SetQueueSize (
 int *handle*,
 int *queuesize*,
 int *callbackThreshold*) [virtual]

11.41.2.32 SetQueueSizeAll() virtual void SetQueueSizeAll (
 int *queuesize*,
 int *callbackThreshold*) [virtual]

11.41.2.33 SetSelectedChannels() [1/2] `virtual void SetSelectedChannels (`
 `array< bool >^ selectedChannels,`
 `int queuesize,`
 `int threshold,`
 `SampleSizeNet samplesize,`
 `SampleDstSizeNet sampleDstSize,`
 `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.
------------------------	---

11.41.2.34 SetSelectedChannels() [2/2] `virtual void SetSelectedChannels (`
 `int nChannels,`
 `int queuesize,`

```

    int threshold,
    SampleSizeNet samplesize,
    SampleDstSizeNet sampleDstSize,
    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.

11.41.2.35 SetSelectedChannelsQueue() [1/2] `virtual void SetSelectedChannelsQueue (`

```

    array< bool >^ selectedChannels,
    int queuesize,
    int threshold,
    SampleSizeNet samplesize,
    SampleDstSizeNet sampleDstSize,
    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.
------------------------	---

11.41.2.36 SetSelectedChannelsQueue() [2/2] `virtual void SetSelectedChannelsQueue (`
 `int nChannels,`
 `int queuesize,`
 `int threshold,`
 `SampleSizeNet samplesize,`
 `SampleDstSizeNet sampleDstSize,`
 `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.
------------------------	---

11.41.2.37 SetSelectedData() [1/2] virtual void SetSelectedData (
 array< bool >^ *selectedChannels*,
 int *queuesize*,
 int *threshold*,
 SampleSizeNet *samplesize*,
 SampleDstSizeNet *sampleDstSize*,
 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.

11.41.2.38 SetSelectedData() [2/2] `virtual void SetSelectedData (`
 `int nChannels,`
 `int queuesize,`
 `int threshold,`
 `SampleSizeNet samplesize,`
 `SampleDstSizeNet sampleDstSize,`
 `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.

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

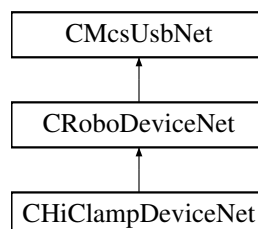
11.41.2.40 SetupGroupDacqQueue() [2/2] void SetupGroupDacqQueue (
 int *queuesize*,
 int *threshold*,
 unsigned int *virtualDevice*)

11.41.2.41 UpdateQueueSize() virtual uint32_t UpdateQueueSize (
 int *handle*) [virtual]

11.42 CHiClampDeviceNet Class Reference

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

Inheritance diagram for CHiClampDeviceNet:



Public Member Functions

- [CHiClampDeviceNet](#) (void)

- ## Additional Inherited Members

CHiClampDeviceNet is the to control the MCS HiClamp device

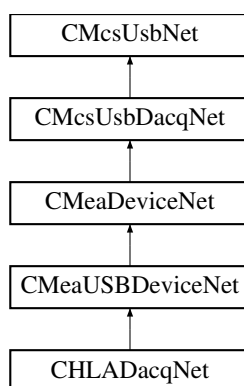
11.42.2 Constructor & Destructor Documentation

11.42.3 Property Documentation

11.42.3.1 RoboDacq CRoboDacqNet[^] RoboDacq [get]

11.43 CHLADacqNet Class Reference

Inheritance diagram for CHLADacqNet:



- CHLADacqNet (void)

Additional Inherited Members

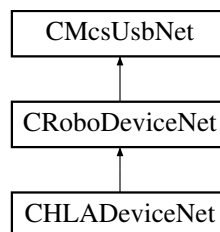
11.43.1 Constructor & Destructor Documentation

11.43.1.1 CHLADacqNet() `CHLADacqNet (`
`void)`

11.44 CHLADeviceNet Class Reference

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

Inheritance diagram for CHLADeviceNet:



Public Member Functions

- [CHLADeviceNet](#) (void)

Properties

- [CHLADacqNet](#)[^] [HLADacq](#) [get]
- [CSerialPortNet](#)[^] [SerialPort](#) [get]

Additional Inherited Members

11.44.1 Detailed Description

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

11.44.2 Constructor & Destructor Documentation

11.44.2.1 CHLADeviceNet() `CHLADeviceNet (`
`void)`

11.44.3 Property Documentation

11.44.3.1 HLADacq [CHLADacqNet](#)[^] HLADacq [get]

11.44.3.2 SerialPort [CSerialPortNet](#)[^] SerialPort [get]

11.45 CMcsUsbDacqNet::CHWInfo Class Reference

Class to provide hardware information about the device.

Classes

- class [CVoltageRangeInfoNet](#)

Public Member Functions

- [CHWInfo](#) ([CMcsUsbDacqNet](#)[^] 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 System::Collections::Generic::List< int32_t > ^ [GetAvailableVoltageRangesInMicroVolt](#) (int milliGain)
Gets a List of voltage ranges the device supports.
- virtual System::Collections::Generic::List< [CVoltageRangeInfoNet](#)[^] > ^ [GetAvailableVoltageRangesInMicroVoltAndStringsInMicroVolt](#) (int milliGain)
Gets a List of voltage ranges the device supports.

11.45.1 Detailed Description

Class to provide hardware information about the device.

11.45.2 Constructor & Destructor Documentation

11.45.2.1 CHWInfo() `CHWInfo (CMcsUsbDacqNet^ device)`

11.45.3 Member Function Documentation

11.45.3.1 GetAvailableSampleRates() `virtual uint32_t GetAvailableSampleRates ([System::Runtime::InteropServices::Out] System::Collections::Generic::List< int32_t> ^ sampleRates) [virtual]`

11.45.3.2 GetAvailableVoltageRangesInMicroVolt() `virtual System::Collections::Generic::List<int32_t> ^ GetAvailableVoltageRangesInMicroVolt (int milliGain) [virtual]`

Gets a List of voltage ranges the device supports.

The List is scaled by the gain factor given as argument to this function. Use "1000" as scale factor for backwards compatibility. To get a list of voltage ranges for the headstage, obtain the gain of the headstage with the `GetGain()` call and use the result in the `milliGain` parameter. To get a list of voltage ranges for the analog inputs of the interfaceboard, obtain the gain of the analog inputs with the `GetAnalogGain()` call and use the result in the `milliGain` parameter.

Parameters

<i>milliGain</i>	The gain factor (in milliGain) used to scale the list of voltage ranges.
------------------	--

Returns

List of voltage ranges in μ V.

11.45.3.3 GetAvailableVoltageRangesInMicroVoltAndStringsInMilliVolt() `virtual System::Collections::Generic::List<CVoltageRangeInfoNet^> ^ GetAvailableVoltageRangesInMicroVoltAndStringsInMilliVolt (int milliGain) [virtual]`

Gets a List of voltage ranges the device supports.

The List is scaled by the gain factor given as argument to this function. Use "1000" as scale factor for backwards compatibility. Each list entry contains the voltage range as an integer and as a string. To get a list of voltage ranges for the headstage, obtain the gain of the headstage with the `GetGain()` call and use the result in the `milliGain` parameter. To get a list of voltage ranges for the analog inputs of the interfaceboard, obtain the gain of the analog inputs with the `GetAnalogGain()` call and use the result in the `milliGain` parameter.

Parameters

<i>milliGain</i>	The gain factor (in milliGain) used to scale the list of voltage ranges.
------------------	--

Returns

List of voltage ranges in μV .

11.45.3.4 GetNumberOfHWADCCannels() `virtual uint32_t GetNumberOfHWADCCannels ([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.

11.45.3.5 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.

11.45.3.6 IsDigitalChannelDedicated() `virtual bool IsDigitalChannelDedicated () [virtual]`

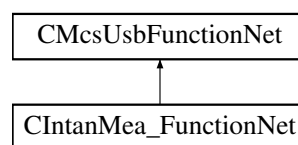
Query if the digital channel replaces an analog channel when enabled (e.g. on MC_Card) or adds a channel link on USB devices.

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.

11.46 CIntanMea_FunctionNet Class Reference

Inheritance diagram for CIntanMea_FunctionNet:



Public Member Functions

- [CIntanMea_FunctionNet](#) ([CMcsUsbNet](#)[^] mcsusb, [CMcsUsbFunctionPointerContainer](#)[^] intalMea_FunctionPointerContainer)
- [CIntanMea_FunctionNet](#) ([CMcsUsbNet](#)[^] mcsusb)
- [int](#) [GetUpperFrequencyByIndex](#) (unsigned short index)
- [int](#) [GetLowerFrequencyByIndex](#) (unsigned short index)
- [int64_t](#) [GetDSPHighPassByIndex](#) (unsigned short index)
- [int](#) [GetIntanRegister](#) (unsigned short chip, unsigned short registernumber)
- [int](#) [GetImpedanceResult](#) (unsigned short channel)
- [void](#) [SetBandwidthByIndex](#) (int upper_index, int lower_index)
- [void](#) [SetDSPHighPassByIndex](#) (int index)
- [void](#) [AmplifierSettle](#) ()
- [void](#) [SetIntanRegister](#) (unsigned short register_number, int value)
- [void](#) [SetDiagnosticMode](#) (unsigned char onoff)
- [void](#) [BeginImpedanceCheck](#) (array< int >[^] config_values)

Additional Inherited Members

11.46.1 Constructor & Destructor Documentation

11.46.1.1 CIntanMea_FunctionNet() [1/2] `CIntanMea_FunctionNet (` `CMcsUsbNet^ mcsusb,` `CMcsUsbFunctionPointerContainer^ intalMea_FunctionPointerContainer)`

11.46.1.2 CIntanMea_FunctionNet() [2/2] `CIntanMea_FunctionNet (
CMcsUsbNet^ mcsusb)`

11.46.2 Member Function Documentation

11.46.2.1 AmplifierSettle() `void AmplifierSettle ()`

11.46.2.2 BeginImpedanceCheck() `void BeginImpedanceCheck (
array< int >^ config_values)`

11.46.2.3 GetDSPHighPassByIndex() `int64_t GetDSPHighPassByIndex (
unsigned short index)`

11.46.2.4 GetImpedanceResult() `int GetImpedanceResult (
unsigned short channel)`

11.46.2.5 GetIntanRegister() `int GetIntanRegister (
unsigned short chip,
unsigned short registernumber)`

11.46.2.6 GetLowerFrequencyByIndex() `int GetLowerFrequencyByIndex (
unsigned short index)`

11.46.2.7 GetUpperFrequencyByIndex() `int GetUpperFrequencyByIndex (
unsigned short index)`

11.46.2.8 SetBandwidthByIndex() `void SetBandwidthByIndex (
int upper_index,
int lower_index)`

11.46.2.9 SetDiagnosticMode() void SetDiagnosticMode (
 unsigned char *onoff*)

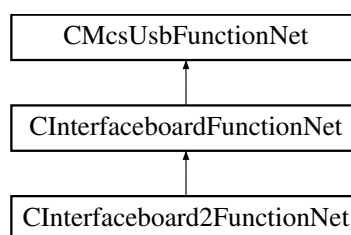
11.46.2.10 SetDSPHighPassByIndex() void SetDSPHighPassByIndex (
 int *index*)

11.46.2.11 SetIntanRegister() void SetIntanRegister (
 unsigned short *register_number*,
 int *value*)

11.47 CInterfaceboard2FunctionNet Class Reference

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

Inheritance diagram for CInterfaceboard2FunctionNet:



Public Member Functions

- [CInterfaceboard2FunctionNet](#) ([CMcsUsbNet](#)[^] mcsusb, [CMcsUsbFunctionPointerContainer](#)[^] pInterfaceboard2↔
FunctionPointerContainer)
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

11.47.1 Detailed Description

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

11.47.2 Constructor & Destructor Documentation

11.47.2.1 CInterfaceboard2FunctionNet() [1/2] `CInterfaceboard2FunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ pInterfaceboard2FunctionPointerContainer)`

Initializes a new instance of the `CInterfaceboard2FunctionNet` class.

11.47.2.2 CInterfaceboard2FunctionNet() [2/2] `CInterfaceboard2FunctionNet (CMcsUsbNet^ mcsusb)`

11.47.2.3 ~CInterfaceboard2FunctionNet() `virtual ~CInterfaceboard2FunctionNet () [virtual]`

11.47.2.4 !CInterfaceboard2FunctionNet() `!CInterfaceboard2FunctionNet ()`

11.47.3 Member Function Documentation

11.47.3.1 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).

11.47.3.2 SetIoVoltage() `void SetIoVoltage (IoVoltageEnumNet ioVoltage)`

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

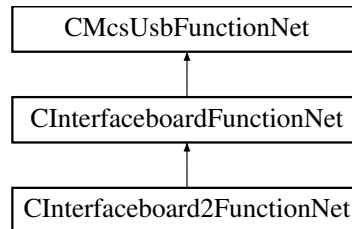
Parameters

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

11.48 CInterfaceboardFunctionNet Class Reference

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

Inheritance diagram for CInterfaceboardFunctionNet:



Public Member Functions

- [CInterfaceboardFunctionNet](#) ([CMcsUsbNet](#)[^] mcsusb, [CMcsUsbFunctionPointerContainer](#)[^] pInterfaceboard↔
FunctionPointerContainer)
Initializes a new instance of the [CInterfaceboardFunctionNet](#) class.
- [CInterfaceboardFunctionNet](#) ([CMcsUsbNet](#)[^] mcsusb)
- virtual [~CInterfaceboardFunctionNet](#) ()
- [!CInterfaceboardFunctionNet](#) ()
- void [SetCardinalDacqSamplerate](#) (uint32_t samplerate)
Sets the fundamental/cardinal data aquisition samplerate of the Interfaceboard, default is 50 kHz
- uint32_t [GetCardinalDacqSamplerate](#) ()
Gets the fundamental/cardinal data aquisition 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

11.48.1 Detailed Description

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

11.48.2 Constructor & Destructor Documentation

- ##### 11.48.2.1 CInterfaceboardFunctionNet() [1/2] [CInterfaceboardFunctionNet](#) (
- ```

CMcsUsbNet^ mcsusb,
CMcsUsbFunctionPointerContainer^ pInterfaceboardFunctionPointerContainer)

```

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

**11.48.2.2 CInterfaceboardFunctionNet()** [2/2] `CInterfaceboardFunctionNet (   
CMcsUsbNet^ mcsusb )`

**11.48.2.3 ~CInterfaceboardFunctionNet()** `virtual ~CInterfaceboardFunctionNet ( ) [virtual]`

**11.48.2.4 "!CInterfaceboardFunctionNet()** `!CInterfaceboardFunctionNet ( )`

### 11.48.3 Member Function Documentation

**11.48.3.1 GetCardinalDacqSamplerate()** `uint32_t GetCardinalDacqSamplerate ( )`

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

#### Returns

The samplerate in Hz.

**11.48.3.2 GetCardinalStgOutputrate()** `uint32_t GetCardinalStgOutputrate ( )`

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

#### Returns

The output rate in Hz.

**11.48.3.3 SetCardinalDacqSamplerate()** `void SetCardinalDacqSamplerate (   
uint32_t samplerate )`

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

#### Parameters

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

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

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

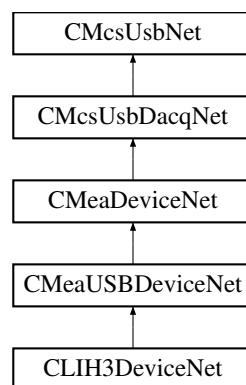
#### Parameters

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

## 11.49 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](#) (uint32\_t AdcChannel, int32\_t Offset)  
*Sets the ADC offset of the DACQ for a single channel*
- int32\_t [GetAdcOffset](#) (uint32\_t AdcChannel)  
*Gets the ADC offset of the DACQ for a single channel*

- void [SetAdcOffsetPermanent](#) (uint32\_t AdcChannel)  
*Writes the ADC offset of the DACQ for a single channel to permanent EEPROM memory*
- void [ErasePermanentAdcOffset](#) (uint32\_t 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](#) (uint32\_t DacChannel, int32\_t Offset)  
*Sets the offset of a DAC channel.*
- int32\_t [GetDacOffset](#) (uint32\_t DacChannel)  
*Gets the offset of a DAC channel.*
- void [SetDacOffsetPermanent](#) (uint32\_t DacChannel)  
*Writes the DAC offset of the STG for a single channel to permanent EEPROM memory*
- void [ErasePermanentDacOffset](#) (uint32\_t 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.*
- String ^ [ReadUARTData](#) ()  
*Reads the config string from the device.*
- void [WriteUARTData](#) (String^ commandString)  
*Write the command string to the device.*
- uint32\_t [GetCardinalDacqSamplerate](#) ()  
*Gets the fundamental/cardinal data aquisition samplerate of the LIH 3.X, default is 2 MHz*
- uint32\_t [GetCardinalStgOutputrate](#) ()  
*Gets the fundamental/cardinal STG output rate of the LIH3.X, default is 2 MHz*

## Properties

- [CStimulusFunctionNet](#) ^ [StimulusFunction](#) [get]

## Additional Inherited Members

### 11.49.1 Detailed Description

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

### 11.49.2 Constructor & Destructor Documentation

#### 11.49.2.1 CLIH3DeviceNet() [CLIH3DeviceNet](#) ( )

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

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

#### 11.49.2.3 ~!CLIH3DeviceNet() [!CLIH3DeviceNet](#) ( )

### 11.49.3 Member Function Documentation

#### 11.49.3.1 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 |
|-----------------------|---------------------------------|

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

Enables the User Trigger

##### Parameters

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

**11.49.3.3 ErasePermanentAdcOffset()** `void ErasePermanentAdcOffset (`  
`uint32_t AdcChannel )`

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

**Parameters**

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

**11.49.3.4 ErasePermanentDacOffset()** `void ErasePermanentDacOffset (`  
`uint32_t DacChannel )`

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

**Parameters**

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

**11.49.3.5 GetAdcOffset()** `int32_t GetAdcOffset (`  
`uint32_t 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

**11.49.3.6 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) |



**11.49.3.7 GetCardinalDacqSamplerate()** `uint32_t GetCardinalDacqSamplerate ( )`

Gets the fundamental/cardinal data acquisition samplerate of the LIH 3.X, default is 2 MHz

**Returns**

The samplerate in Hz.

**11.49.3.8 GetCardinalStgOutputrate()** `uint32_t GetCardinalStgOutputrate ( )`

Gets the fundamental/cardinal STG output rate of the LIH3.X, default is 2 MHz

**Returns**

The output rate in Hz.

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

Gets the DAC Idle value

**Parameters**

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

**Returns**

The idle value

**11.49.3.10 GetDacOffset()** `int32_t GetDacOffset (   
uint32_t DacChannel )`

Gets the offset of a DAC channel.

**Parameters**

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

**Returns**

The offset for the given channel number

**11.49.3.11 GetDacqRunStatus()** `uint16_t GetDacqRunStatus ( )`

Gets the data acquisition running status

**Returns**

The status (1: running / 0: stopped)

**11.49.3.12 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

**11.49.3.13 GetDigInState()** `uint16_t GetDigInState ( )`

Reads from the LIH30 digital input

**Returns**

The bit mask defining the digital input state

**11.49.3.14 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

**11.49.3.15 GetSampleInterval()** `uint32_t GetSampleInterval ( )`

Gets the Sample Interval for the DACQ and Stimulation

## Returns

Sample Interval configured on the device

**11.49.3.16 IsUserTriggerEnabled()** `bool IsUserTriggerEnabled ( )`

Is the User Trigger enabled

## Returns

Enabled

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

Gets the clipping information

## Parameters

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

## Returns

The clipping value

**11.49.3.18 ReadUARTData()** `String ^ ReadUARTData ( )`

Reads the config string from the device.

**Returns**

The config string.

**11.49.3.19 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   |

**11.49.3.20 SetAdcOffset()** `void SetAdcOffset (`  
`uint32_t 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 |

**11.49.3.21 SetAdcOffsetPermanent()** `void SetAdcOffsetPermanent (`  
`uint32_t AdcChannel )`

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

**Parameters**

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

**11.49.3.22 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) |

**11.49.3.23 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  |

**11.49.3.24 SetDacOffset()** void SetDacOffset (  
uint32\_t 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 |

**11.49.3.25 SetDacOffsetPermanent()** void SetDacOffsetPermanent (  
uint32\_t DacChannel )

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

#### Parameters

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

**11.49.3.26 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  |

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

Writes to the LIH30 digital output

**Parameters**

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

**11.49.3.28 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                                               |

**11.49.3.29 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 |
|-----------------------|---------------------------------------------------------------------|

**11.49.3.30 WriteUARTData()** void WriteUARTData (   
 String^ *commandString* )

Write the command string to the device.

Parameters

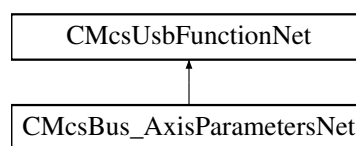
|                      |                    |
|----------------------|--------------------|
| <i>commandString</i> | The config string. |
|----------------------|--------------------|

## 11.49.4 Property Documentation

**11.49.4.1 StimulusFunction** CStimulusFunctionNet^ StimulusFunction [get]

## 11.50 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

## 11.50.1 Constructor & Destructor Documentation

**11.50.1.1 CMcsBus\_AxisParametersNet()** [CMcsBus\\_AxisParametersNet](#) (   
 [CMcsUsbNet](#)^ *device* )

**11.50.1.2** `~CMcsBus_AxisParametersNet()` `~CMcsBus_AxisParametersNet (`  
`void )`

## 11.50.2 Member Function Documentation

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

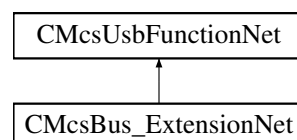
**11.50.2.2** `GetAxisParametersUnsignedEeprom()` `unsigned int GetAxisParametersUnsignedEeprom (`  
`unsigned char busnumber,`  
`unsigned char busaddress,`  
`unsigned char axis,`  
`unsigned short index )`

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

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

## 11.51 CMcsBus\_ExtensionNet Class Reference

Inheritance diagram for CMcsBus\_ExtensionNet:





**Public Member Functions**

- [CMcsBus\\_ExtensionNet](#) ([CMcsUsbNet](#)^ 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****11.51.1 Constructor & Destructor Documentation**

**11.51.1.1 CMcsBus\_ExtensionNet()** [CMcsBus\\_ExtensionNet](#) (  
[CMcsUsbNet](#)^ device )

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

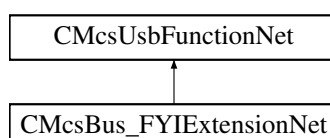
**11.51.2 Member Function Documentation**

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

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

**11.52 CMcsBus\_FYIExtensionNet Class Reference**

Inheritance diagram for CMcsBus\_FYIExtensionNet:



## Public Member Functions

- [CMcsBus\\_FYIExtensionNet](#) ([CMcsUsbNet](#)^ 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

### 11.52.1 Constructor & Destructor Documentation

**11.52.1.1** [CMcsBus\\_FYIExtensionNet\(\)](#) [CMcsBus\\_FYIExtensionNet](#) (  
    [CMcsUsbNet](#)^ device )

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

### 11.52.2 Member Function Documentation

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

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

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

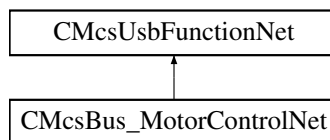
**11.52.2.4 SetDIO()** void SetDIO (  
     unsigned char *busnumber*,  
     unsigned char *busaddress*,  
     unsigned short *io* )

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

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

## 11.53 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 [SetMCRevisionEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, unsigned short revision)
- unsigned short [GetMCRevisionEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCSpeedUnitEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, int32\_t speedunit)
- int32\_t [GetMCSpeedUnitEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCOutputOnOff](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, bool OnOff\_status)
- bool [GetMCOutputOnOff](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCSpeedShortCommand](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, short speed)
- short [GetMCSpeedShortCommand](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCAccelerationShortCommand](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, unsigned short acceleration)
- unsigned short [GetMCAccelerationShortCommand](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCCurrentShortCommand](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, short current)
- short [GetMCCurrentShortCommand](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCMaxTravelShortCommand](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, int travel)
- int [GetMCMaxTravelShortCommand](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCCurrentPosition](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, int position)
- int [GetMCCurrentPosition](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)

- void [SetMCNewPosition](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, int position)
- int [GetMCNewPosition](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- short [GetMCCurrentSpeed](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [StartMCMovement](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCRotation](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, unsigned char onoff)
- unsigned short [GetMCMovement](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCReference](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, unsigned char switch\_enable, unsigned char switch\_polarity)
- unsigned char [GetMCReference](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, [System::Runtime::InteropServices::Out]unsigned char% switch\_port)
- void [StopMCMovement](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetMCCurrentModeShortCommand](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, [RoboCurrentModeEnumNet](#) mode)
- [RoboCurrentModeEnumNet](#) [GetMCCurrentModeShortCommand](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- unsigned short [GetMCPhase](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- unsigned short [GetMCPhaseOffset](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)
- void [SetSubChannel](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis, unsigned short subchannel)
- unsigned short [GetSubChannel](#) (unsigned char busnumber, unsigned char busaddress, unsigned char axis)

## Additional Inherited Members

### 11.53.1 Constructor & Destructor Documentation

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

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

### 11.53.2 Member Function Documentation

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

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

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

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

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

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

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

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

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

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

**11.53.2.11 GetMCCurrentMode()** [RoboCurrentModeEnumNet](#) GetMCCurrentMode (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

**11.53.2.12 GetMCCurrentModeEeprom()** [RoboCurrentModeEnumNet](#) GetMCCurrentModeEeprom (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

**11.53.2.13 GetMCCurrentModeShortCommand()** [RoboCurrentModeEnumNet](#) GetMCCurrentModeShort↵  
Command (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

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

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



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

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

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

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

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

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

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

**11.53.2.23 GetMCMaxTravel()** int GetMCMaxTravel (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis* )

**11.53.2.24 GetMCMaxTravelEeprom()** int GetMCMaxTravelEeprom (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis* )

**11.53.2.25 GetMCMaxTravelShortCommand()** int GetMCMaxTravelShortCommand (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis* )

**11.53.2.26 GetMCMovement()** unsigned short GetMCMovement (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis* )

**11.53.2.27 GetMCNewPosition()** int GetMCNewPosition (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis* )

**11.53.2.28 GetMCOutputOnOff()** bool GetMCOutputOnOff (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis* )

**11.53.2.29 GetMCPhase()** unsigned short GetMCPhase (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis* )

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

**11.53.2.50 SetMCAXisRevisionEeprom()** void SetMCAXisRevisionEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 unsigned short *revision* )

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

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

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

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

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

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

**11.53.2.57 SetMCCurrentMode()** void SetMCCurrentMode (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 [RoboCurrentModeEnumNet](#) *mode* )

**11.53.2.58 SetMCCurrentModeEeprom()** void SetMCCurrentModeEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 [RoboCurrentModeEnumNet](#) *mode* )

**11.53.2.59 SetMCCurrentModeShortCommand()** void SetMCCurrentModeShortCommand (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 [RoboCurrentModeEnumNet](#) *mode* )

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

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

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

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

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

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

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

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

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



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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

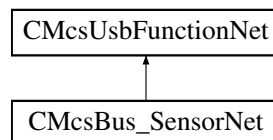
**11.53.2.89 SetSubChannel()** void SetSubChannel (   
    unsigned char *busnumber*,   
    unsigned char *busaddress*,   
    unsigned char *axis*,   
    unsigned short *subchannel* )

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

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

## 11.54 CMcsBus\_SensorNet Class Reference

Inheritance diagram for CMcsBus\_SensorNet:



## Public Member Functions

- [CMcsBus\\_SensorNet](#) ([CMcsUsbNet](#)^ 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 > ^ [GetPressure](#) (unsigned char busnumber, unsigned char busaddress, int n)
- int [GetPressure](#) (unsigned char busnumber, unsigned char busaddress, unsigned short index)
- void [SetRegulatorOnOff](#) (unsigned char busnumber, unsigned char busaddress, unsigned short index, unsigned char onoff)
- unsigned char [GetRegulatorOnOff](#) (unsigned char busnumber, unsigned char busaddress, unsigned short index)
- void [SetSollPressure](#) (unsigned char busnumber, unsigned char busaddress, unsigned short index, int pressure)
- int [GetSollPressure](#) (unsigned char busnumber, unsigned char busaddress, unsigned short index)
- void [SetRegulatorFactor](#) (unsigned char busnumber, unsigned char busaddress, unsigned short index, int factor)
- int [GetRegulatorFactor](#) (unsigned char busnumber, unsigned char busaddress, unsigned short index)
- void [SetPressureOffset](#) (unsigned char busnumber, unsigned char busaddress, unsigned short index)
- array< unsigned short > ^ [GetPressureOffset](#) (unsigned char busnumber, unsigned char busaddress)
- int [GetPressureOffset](#) (unsigned char busnumber, unsigned char busaddress, unsigned short index)
- unsigned int [GetRegulatorStatus](#) (unsigned char busnumber, unsigned char busaddress, unsigned short index)
- void [SetRotatePump](#) (unsigned char busnumber, unsigned char busaddress, unsigned short index, short speed)
- short [GetRotatePump](#) (unsigned char busnumber, unsigned char busaddress, unsigned short index)
- void [SetMovePump](#) (unsigned char busnumber, unsigned char busaddress, unsigned short index, unsigned short speed, int position)
- void [GetMovePump](#) (unsigned char busnumber, unsigned char busaddress, unsigned short index, [System::Runtime::InteropServices::Out]unsigned short% speed, [System::Runtime::InteropServices::Out]int% position)
- void [SetRegulationTimeouts](#) (unsigned char busnumber, unsigned char busaddress, unsigned short MaxSpeedWait, unsigned short MaxSignChange)
- void [GetRegulationTimeouts](#) (unsigned char busnumber, unsigned char busaddress, [System::Runtime::InteropServices::Out]unsigned short% MaxSpeedWait, [System::Runtime::InteropServices::Out]unsigned short% MaxSignChange)
- array< int > ^ [Get4ADC](#) (unsigned char busnumber, unsigned char busaddress)
- array< int > ^ [Get4ADCAverage](#) (unsigned char busnumber, unsigned char busaddress)
- void [Set4DAC](#) (unsigned char busnumber, unsigned char busaddress, array< unsigned short >^ dac)
- array< unsigned short > ^ [Get4DAC](#) (unsigned char busnumber, unsigned char busaddress)
- void [Set4ADCMode](#) (unsigned char busnumber, unsigned char busaddress, [PatchServAdcModeEnumNet](#) mode)
- [PatchServAdcModeEnumNet](#) [Get4ADCMode](#) (unsigned char busnumber, unsigned char busaddress)
- void [Set4ADCCatchampAverageShift](#) (unsigned char busnumber, unsigned char busaddress, unsigned int shift)
- unsigned int [Get4ADCCatchampAverageShift](#) (unsigned char busnumber, unsigned char busaddress)
- array< unsigned short > ^ [Get2AnalogInput](#) (unsigned char busnumber, unsigned char busaddress)
- unsigned short [Get2DigitalInput](#) (unsigned char busnumber, unsigned char busaddress)

- array< unsigned short > ^ [GetADCs](#) (unsigned char busnumber, unsigned char busaddress, int n)
- array< unsigned short > ^ [GetADCsLoop](#) (unsigned char busnumber, unsigned char busaddress, int n)
- void [SetPiezoState](#) (unsigned char busnumber, unsigned char busaddress, int state)
- void [GetPiezoState](#) (unsigned char busnumber, unsigned char busaddress, [System::Runtime::InteropServices::Out]int% state, [System::Runtime::InteropServices::Out]int% reason)
- void [SetDACs](#) (unsigned char busnumber, unsigned char busaddress, unsigned short index, array< unsigned short > ^ dac\_times\_voltages)
- array< unsigned short > ^ [GetDACs](#) (unsigned char busnumber, unsigned char busaddress, unsigned short index)
- void [SetSamplePeriode](#) (unsigned char busnumber, unsigned char busaddress, unsigned short periode)
- unsigned short [GetSamplePeriode](#) (unsigned char busnumber, unsigned char busaddress)
- void [StartSync](#) (unsigned char busnumber, unsigned char busaddress)
- unsigned short [GetSyncState](#) (unsigned char busnumber, unsigned char busaddress)
- void [CatchAmpSetDacAmplitude](#) (unsigned char busnumber, unsigned char busaddress, unsigned short dacAmplitude)
- unsigned short [CatchAmpGetDacAmplitude](#) (unsigned char busnumber, unsigned char busaddress)
- void [CatchAmpSetDacOffset](#) (unsigned char busnumber, unsigned char busaddress, short dacOffset)
- short [CatchAmpGetDacOffset](#) (unsigned char busnumber, unsigned char busaddress)
- int [CatchAmpGetAdcMean](#) (unsigned char busnumber, unsigned char busaddress)
- int [CatchAmpGetAdcValue](#) (unsigned char busnumber, unsigned char busaddress)
- int [CatchAmpGetAdcValueH](#) (unsigned char busnumber, unsigned char busaddress)
- int [CatchAmpGetAdcValueL](#) (unsigned char busnumber, unsigned char busaddress)
- void [CatchAmpSetPwmEnable](#) (unsigned char busnumber, unsigned char busaddress, bool pwmEnable)
- bool [CatchAmpGetPwmEnable](#) (unsigned char busnumber, unsigned char busaddress)
- void [CatchAmpSetDacEnable](#) (unsigned char busnumber, unsigned char busaddress, bool dacEnable)
- bool [CatchAmpGetDacEnable](#) (unsigned char busnumber, unsigned char busaddress)
- int [TactSwitchGetState](#) (unsigned char busnumber, unsigned char busaddress)
- void [TactSwitchSetDisplay](#) (unsigned char busnumber, unsigned char busaddress, int Melody)

## Additional Inherited Members

### 11.54.1 Constructor & Destructor Documentation

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

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

### 11.54.2 Member Function Documentation

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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



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

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

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

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

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

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

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

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

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

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

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

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

**11.54.2.29 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* )

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

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

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

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

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

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

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

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

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

**11.54.2.39 GetRotatePump()** short GetRotatePump (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned short *index* )

**11.54.2.40 GetSamplePeriode()** unsigned short GetSamplePeriode (   
 unsigned char *busnumber*,  
 unsigned char *busaddress* )

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

**11.54.2.42 GetSyncState()** unsigned short GetSyncState (   
 unsigned char *busnumber*,  
 unsigned char *busaddress* )

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

**11.54.2.58 SetSollPressure()** void SetSollPressure (   
    unsigned char *busnumber*,   
    unsigned char *busaddress*,   
    unsigned short *index*,   
    int *pressure* )

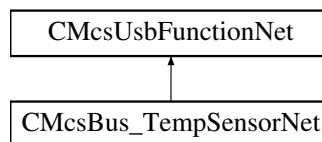
**11.54.2.59 StartSync()** void StartSync (   
    unsigned char *busnumber*,   
    unsigned char *busaddress* )

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

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

## 11.55 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

#### 11.55.1 Constructor & Destructor Documentation

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

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

## 11.55.2 Member Function Documentation

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

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

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

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

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

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

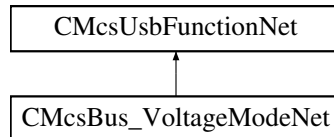
**11.55.2.7 SetNanoVoltsPerKelvin()** `void SetNanoVoltsPerKelvin (`  
    `unsigned char busnumber,`  
    `unsigned char busaddress,`  
    `int nanovoltsperkelvin )`



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

## 11.56 CMcsBus\_VoltageModeNet Class Reference

Inheritance diagram for CMcsBus\_VoltageModeNet:



### Public Member Functions

- `CMcsBus_VoltageModeNet` (`CMcsUsbNet`<sup>^</sup> device)
- `~CMcsBus_VoltageModeNet` (void)
- void `SetVMMMaxPositiveCurrentEeprom` (unsigned char busnumber, unsigned char busaddress, unsigned char channel, short current)
- short `GetVMMMaxPositiveCurrentEeprom` (unsigned char busnumber, unsigned char busaddress, unsigned char channel)
- void `SetVMMMaxPositiveCurrent` (unsigned char busnumber, unsigned char busaddress, unsigned char channel, short current)
- short `GetVMMMaxPositiveCurrent` (unsigned char busnumber, unsigned char busaddress, unsigned char channel)
- void `SetVMMMaxNegativeCurrentEeprom` (unsigned char busnumber, unsigned char busaddress, unsigned char channel, short current)
- short `GetVMMMaxNegativeCurrentEeprom` (unsigned char busnumber, unsigned char busaddress, unsigned char channel)
- void `SetVMMMaxNegativeCurrent` (unsigned char busnumber, unsigned char busaddress, unsigned char channel, short current)
- short `GetVMMMaxNegativeCurrent` (unsigned char busnumber, unsigned char busaddress, unsigned char channel)
- void `SetVMMMaxPositiveVoltageEeprom` (unsigned char busnumber, unsigned char busaddress, unsigned char channel, short voltage)
- short `GetVMMMaxPositiveVoltageEeprom` (unsigned char busnumber, unsigned char busaddress, unsigned char channel)
- void `SetVMMMaxPositiveVoltage` (unsigned char busnumber, unsigned char busaddress, unsigned char channel, short voltage)
- short `GetVMMMaxPositiveVoltage` (unsigned char busnumber, unsigned char busaddress, unsigned char channel)
- void `SetVMMMaxNegativeVoltageEeprom` (unsigned char busnumber, unsigned char busaddress, unsigned char channel, short voltage)
- short `GetVMMMaxNegativeVoltageEeprom` (unsigned char busnumber, unsigned char busaddress, unsigned char channel)
- void `SetVMMMaxNegativeVoltage` (unsigned char busnumber, unsigned char busaddress, unsigned char channel, short voltage)
- short `GetVMMMaxNegativeVoltage` (unsigned char busnumber, unsigned char busaddress, unsigned char channel)

- void [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

### 11.56.1 Constructor & Destructor Documentation

**11.56.1.1 [CMcsBus\\_VoltageModeNet\(\)](#)** [CMcsBus\\_VoltageModeNet](#) (  
    [CMcsUsbNet](#)<sup>^</sup> device )

**11.56.1.2 [~CMcsBus\\_VoltageModeNet\(\)](#)** [~CMcsBus\\_VoltageModeNet](#) (  
    void )

### 11.56.2 Member Function Documentation

**11.56.2.1 [GetVMMaxNegativeCurrent\(\)](#)** short [GetVMMaxNegativeCurrent](#) (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *channel* )

**11.56.2.2 [GetVMMaxNegativeCurrentEeprom\(\)](#)** short [GetVMMaxNegativeCurrentEeprom](#) (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *channel* )

**11.56.2.3 [GetVMMaxNegativeVoltage\(\)](#)** short [GetVMMaxNegativeVoltage](#) (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *channel* )

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

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

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

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

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

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

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

- 11.56.2.11 SetVMMaxNegativeCurrent()** void SetVMMaxNegativeCurrent (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *channel*,  
    short *current* )
- 11.56.2.12 SetVMMaxNegativeCurrentEeprom()** void SetVMMaxNegativeCurrentEeprom (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *channel*,  
    short *current* )
- 11.56.2.13 SetVMMaxNegativeVoltage()** void SetVMMaxNegativeVoltage (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *channel*,  
    short *voltage* )
- 11.56.2.14 SetVMMaxNegativeVoltageEeprom()** void SetVMMaxNegativeVoltageEeprom (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *channel*,  
    short *voltage* )
- 11.56.2.15 SetVMMaxPositiveCurrent()** void SetVMMaxPositiveCurrent (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *channel*,  
    short *current* )
- 11.56.2.16 SetVMMaxPositiveCurrentEeprom()** void SetVMMaxPositiveCurrentEeprom (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *channel*,  
    short *current* )

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

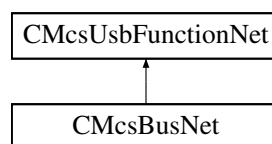
**11.56.2.18 SetVMMaxPositiveVoltageEeprom()** `void SetVMMaxPositiveVoltageEeprom (`  
     `unsigned char busnumber,`  
     `unsigned char busaddress,`  
     `unsigned char channel,`  
     `short voltage )`

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

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

## 11.57 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

### 11.57.1 Constructor & Destructor Documentation

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

**11.57.1.2 [~CMcsBusNet\(\)](#)** virtual [~CMcsBusNet](#) (   
 void ) [virtual]

### 11.57.2 Member Function Documentation

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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



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

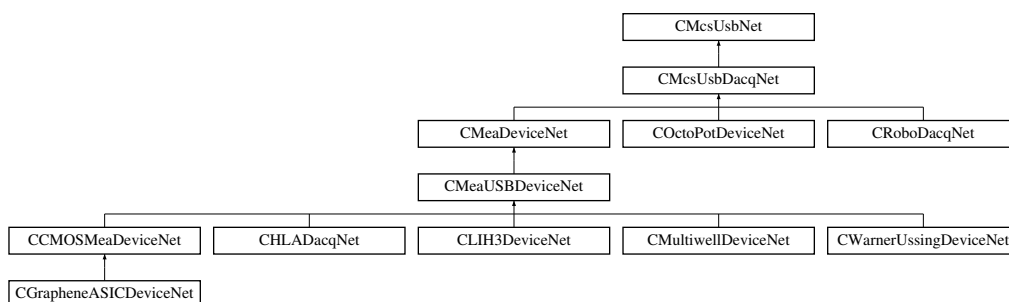
**11.57.2.17 SetCommand() [4/4]** void SetCommand (   
     unsigned char *command*,   
     unsigned char *busnumber*,   
     unsigned char *busaddress*,   
     unsigned char *axis*,   
     unsigned short *value* )

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

## 11.58 CMcsUsbDacqNet Class Reference

Base class for data acquisition devices.

Inheritance diagram for CMcsUsbDacqNet:



### Classes

- class [ChannelBlockNet](#)
- class [CHWInfo](#)

*Class to provide hardware information about the device.*

## Public Member Functions

- [CMcsUsbDacqNet](#) ()
- [~CMcsUsbDacqNet](#) ()
- [uint32\\_t GetErrorMessage](#) ([System::Runtime::InteropServices::Out]String^% errorString, [System::Runtime::InteropServices::Out]int% info)
- [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 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, [MEA2100DigitalSourceEnumNet](#) source, int bitnumber\_offset)  
*Sets the function/source of an digital output bit.*
- [void SetDigitalSource](#) ([DigitalTargetEnumNet](#) digitaltarget, int32\_t NrChannel, [MultiwellDigitalSourceEnumNet](#) source, int bitnumber\_offset)  
*Sets the function/source of an digital output bit.*
- [void SetDigitalSource](#) ([DigitalTargetEnumNet](#) digitaltarget, int32\_t NrChannel, [CMOSMEA5000DigitalSourceEnumNet](#) 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][MEA2100DigitalSourceEnumNet](#)% 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][MultiwellDigitalSourceEnumNet](#)% 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][CMOSMEA5000DigitalSourceEnumNet](#)% 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 int32\_t [GetSampleratePerFramerate](#) (unsigned int virtualDevice)

*Queries the ratio between samplerate and framerate.*

- 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 void [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](#) ()
- [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.*
- [ChannelBlockNet](#)^ [ChannelBlock](#) [get, set]

### Events

- [OnChannelData](#)^ [ChannelDataEvent](#) [add, remove, raise]
- [OnError](#)^ [ErrorEvent](#) [add, remove, raise]

### Additional Inherited Members

#### 11.58.1 Detailed Description

Base class for data acquisition devices.

#### 11.58.2 Constructor & Destructor Documentation

**11.58.2.1 CMcsUsbDacqNet()** [CMcsUsbDacqNet](#) ( )

**11.58.2.2 ~CMcsUsbDacqNet()** [~CMcsUsbDacqNet](#) ( )

#### 11.58.3 Member Function Documentation

**11.58.3.1 ClearBuffers()** [virtual void ClearBuffers](#) ( ) [virtual]

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

**11.58.3.3 GetAdapterType()** virtual AdapterTypeEnumNet GetAdapterType ( ) [virtual]

Gets the adapter which is connected to the MEA2100 device.

**Returns**

AdapterTypeEnumNet which enumerates the possible adapters.

**11.58.3.4 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.

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

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

**11.58.3.7 GetChannelDataFillSize()** int GetChannelDataFillSize ( )

**11.58.3.8 GetChannelLayout()** virtual void 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]

**11.58.3.9 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.

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

**11.58.3.11 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.

**11.58.3.12 GetDigitalSource()** [1/7] void GetDigitalSource (   
DigitalTargetEnumNet digitaltarget,   
int32\_t NrChannel,   
[System::Runtime::InteropServices::Out] CMOSMEA5000DigitalSourceEnumNet% source,   
[System::Runtime::InteropServices::Out] int% bitnumber\_offset )

Gets the function/source of an digital output bit.

This overload is for the CMOSMEA5000 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 assigned to the digital target. |
| <i>bitnumber_offset</i> | An offset / bit number with the source/function.    |

**11.58.3.13 GetDigitalSource() [2/7]** `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 assigned to the digital target. |
| <i>bitnumber_offset</i> | An offset / bit number with the source/function.    |

**11.58.3.14 GetDigitalSource() [3/7]** `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 assigned to the digital target. |
| <i>bitnumber_offset</i> | An offset / bit number with the source/function.    |

**11.58.3.15 GetDigitalSource() [4/7]** `void GetDigitalSource (`  
`DigitalTargetEnumNet digitaltarget,`



```
int32_t NrChannel,
[System::Runtime::InteropServices::Out] MEA2100DigitalSourceEnumNet% 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.   |

**11.58.3.16 GetDigitalSource() [5/7]** void GetDigitalSource (   
 DigitalTargetEnumNet digitaltarget,   
 int32\_t NrChannel,   
 [System::Runtime::InteropServices::Out] MultiwellDigitalSourceEnumNet% source,   
 [System::Runtime::InteropServices::Out] int% bitnumber\_offset )

Gets the function/source of an digital output bit.

This overload is for the Multiwell 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.   |

**11.58.3.17 GetDigitalSource() [6/7]** 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.   |

**11.58.3.18 GetDigitalSource()** [7/7] 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 assigned to the digital target. |
| <i>bitnumber_offset</i> | An offset / bit number with the source/function.    |

**11.58.3.19 GetErrorMessage()** uint32\_t GetErrorMessage (   
     [System::Runtime::InteropServices::Out] String^% errorString,   
     [System::Runtime::InteropServices::Out] int% info )

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

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

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

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

Gets the maximal sampling frequency of the device.

#### Returns

Sampling frequency in Hz.

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

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

#### Returns

MeaLayoutEnumNet which enumerates the MEA types.

**11.58.3.25 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.

**11.58.3.26 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

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

**11.58.3.28 GetResolutionPerDigit()** `virtual uint32_t GetResolutionPerDigit (
 uint32_t virtualDevice,
 DacqGroupChannelEnumNet group,
 [System::Runtime::InteropServices::Out] int% res,
 [System::Runtime::InteropServices::Out] int% resUnit ) [virtual]`

**11.58.3.29 GetSamplerate()** `virtual int32_t GetSamplerate (
 unsigned int virtualDevice ) [virtual]`

Gets the sampling frequency of the device.

#### Returns

Sampling frequency in Hz.

**11.58.3.30 GetSampleratePerFramerate()** `virtual int32_t GetSampleratePerFramerate (
 unsigned int virtualDevice ) [virtual]`

Queries the ratio between samplerate and framerate.

Devices with a samplerate higher than framerate send multiple samples of the the channels in one frame. Use this call to determine the number of samples for each channel in one frame. The multiple samples for the same channel are sent in one continous block followed by a block for the next channel.

#### Parameters

|                      |                        |
|----------------------|------------------------|
| <i>virtualDevice</i> | virtual device to use. |
|----------------------|------------------------|

#### Returns

the ratio.

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

**11.58.3.32 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.

**11.58.3.33 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.

**11.58.3.34 HWInfo()** `CHWInfo ^ HWInfo ( )`

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

Start sampling.

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

Start sampling.

#### Parameters

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

**11.58.3.37 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> |  |

**11.58.3.38 SendStopDacq()** [1/2] `virtual void SendStopDacq ( ) [virtual]`

Stop sampling.

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

Stop sampling.

Parameters

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

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

Stop sampling together with the STG.

Parameters

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

**11.58.3.41 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> |  |
|-----------------------|--|

**11.58.3.42 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.                              |

**11.58.3.43 SetDigitalSource()** `[1/7] void SetDigitalSource (   
     DigitalTargetEnumNet digitaltarget,   
     int32_t NrChannel,   
     CMOSMEA5000DigitalSourceEnumNet source,   
     int bitnumber_offset )`

Sets the function/source of an digital output bit.

This overload is for the CMOSMEA5000 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.     |

**11.58.3.44 SetDigitalSource()** `[2/7] 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.     |

**11.58.3.45 SetDigitalSource() [3/7]** `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.     |

**11.58.3.46 SetDigitalSource() [4/7]** `void SetDigitalSource (`  
`DigitalTargetEnumNet digitaltarget,`  
`int32_t NrChannel,`  
`MEA2100DigitalSourceEnumNet 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.     |

**11.58.3.47 SetDigitalSource() [5/7]** `void SetDigitalSource (`  
`DigitalTargetEnumNet digitaltarget,`



```
int32_t NrChannel,
MultiwellDigitalSourceEnumNet source,
int bitnumber_offset)
```

Sets the function/source of an digital output bit.

This overload is for the Multiwell 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.     |

**11.58.3.48 SetDigitalSource() [6/7]** 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.     |

**11.58.3.49 SetDigitalSource() [7/7]** 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.     |

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

**11.58.3.51 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. |
|-------------|---------------------------|

**11.58.3.52 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. |
|--------------------------|--------------------------------------------------------------------------|

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

Start the data acquisition thread and sampling.

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

Start the data acquisition thread and sampling.

**Parameters**

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

**11.58.3.55 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. |
|---------------------|----------------------|

**11.58.3.56 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. |
|----------------------|--------------------------|

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

Start the data acquisition thread.

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

Start the data acquisition thread.

**Parameters**

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

**11.58.3.59 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. |
|---------------------|----------------------|

**11.58.3.60 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. |
|----------------------|--------------------------|

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

Stop the data acquisition thread and sampling.

**11.58.3.62 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. |
|----------------------|--------------------------|

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

#### 11.58.4 Member Data Documentation

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

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

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

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

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

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

#### 11.58.5 Property Documentation

**11.58.5.1 ChannelBlock** `ChannelBlockNet^ ChannelBlock` [get], [set]

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

The sampling frequency of the device in Hz.

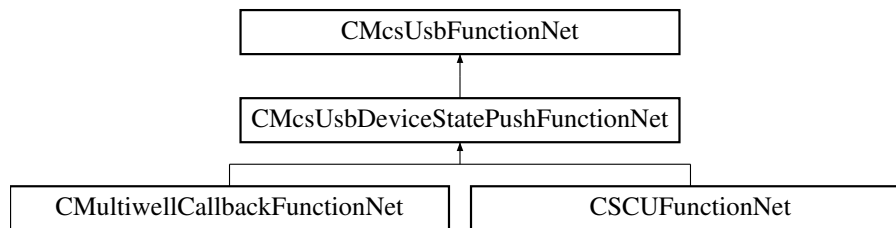
#### 11.58.6 Event Documentation

**11.58.6.1 ChannelDataEvent** `OnChannelData^ ChannelDataEvent [add], [remove], [raise]`

**11.58.6.2 ErrorEvent** `OnError^ ErrorEvent [add], [remove], [raise]`

## 11.59 CMcsUsbDeviceStatePushFunctionNet Class Reference

Inheritance diagram for CMcsUsbDeviceStatePushFunctionNet:



### Public Member Functions

- void `TriggerStatus` ()

### Protected Member Functions

- `CMcsUsbDeviceStatePushFunctionNet` (`CMcsUsbNet^ mcsusb`, `CMcsUsbFunctionPointerContainer^ pDevice`)

### Events

- `OnMcsUsbDeviceState^ McsUsbDeviceStateEvent` [add, remove, raise]

### Additional Inherited Members

#### 11.59.1 Constructor & Destructor Documentation

**11.59.1.1 CMcsUsbDeviceStatePushFunctionNet()** `CMcsUsbDeviceStatePushFunctionNet` (`CMcsUsbNet^ mcsusb`, `CMcsUsbFunctionPointerContainer^ pDevice`) [protected]

#### 11.59.2 Member Function Documentation



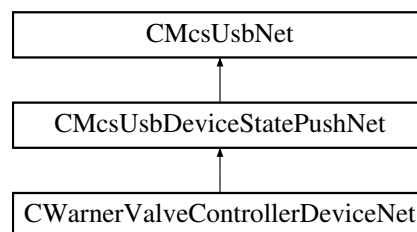
### 11.59.2.1 TriggerStatus() void TriggerStatus ( )

### 11.59.3 Event Documentation

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

## 11.60 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

### 11.60.1 Constructor & Destructor Documentation

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

## 11.60.2 Member Function Documentation

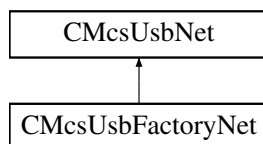
### 11.60.2.1 TriggerStatus() `void TriggerStatus ( )`

## 11.60.3 Event Documentation

### 11.60.3.1 McsUsbDeviceStateEvent `OnMcsUsbDeviceState^ McsUsbDeviceStateEvent [add], [remove], [raise]`

## 11.61 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

#### 11.61.1 Constructor & Destructor Documentation

**11.61.1.1 CMcsUsbFactoryNet()** `CMcsUsbFactoryNet ( )`

**11.61.1.2 ~CMcsUsbFactoryNet()** `~CMcsUsbFactoryNet ( )`

## 11.61.2 Member Function Documentation

**11.61.2.1 ChangeSerialNumber()** `uint32_t ChangeSerialNumber ( String^ serial )`

**11.61.2.2 Coldstart()** `uint32_t Coldstart ( CFirmwareDestinationNet dest )`

**11.61.2.3 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 )`

**11.61.2.4 DownloadFirmware()** `array<uint8_t> ^ DownloadFirmware ( CFirmwareDestinationNet Dest, uint32_t Address, uint32_t length )`

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

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

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

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

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

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

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

**11.61.2.12 GetDestinationSerialNumber()** String ^ GetDestinationSerialNumber (   
 CFirmwareDestinationNet dest )

**11.61.2.13 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.

**11.61.2.14 GetFirmwareVersionFromFile() [1/2]** static bool GetFirmwareVersionFromFile (   
String^ *FirmwareFile*,  
[System::Runtime::InteropServices::Out] uint32\_t% *Version* ) [static]

Retrives version info from a Firmware update file.

**11.61.2.15 GetFirmwareVersionFromFile() [2/2]** static bool GetFirmwareVersionFromFile (   
String^ *FirmwareFile*,  
[System::Runtime::InteropServices::Out] uint32\_t% *Version*,  
[System::Runtime::InteropServices::Out] uint32\_t% *Position* ) [static]

**11.61.2.16 GetFirmwareVersionFromHexFile()** static bool GetFirmwareVersionFromHexFile (   
String^ *FirmwareFile*,  
[System::Runtime::InteropServices::Out] uint32\_t% *Version* ) [static]

**11.61.2.17 GetNumDestinations()** unsigned int GetNumDestinations ( )

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

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

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

**11.61.2.21 GetXilinxFlashOffset()** `int32_t GetXilinxFlashOffset (   
 CFirmwareDestinationNet dest )`

**11.61.2.22 GetXilinxFlashReadCommand()** `uint32_t GetXilinxFlashReadCommand (   
 CFirmwareDestinationNet dest )`

**11.61.2.23 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> . |
|------------------|------------------------------------------------------------------------|

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

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

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

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

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

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

**11.61.2.30 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. |
|------------------|-----------------------------------|

**11.61.2.31 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. |
|------------------|-----------------------------------|

**11.61.2.32 UpdateFirmware() [3/5]** `bool UpdateFirmware (`  
     String^ *FirmwareFile*,  
     CMcsUsbListEntryNet^ *listEntry*,  
     CFirmwareDestinationNet *dest*,  
     bool *SkipWait*,  
     uint32\_t *LockMask* )

**11.61.2.33 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. |
|---------------------|--------------------------------|

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

## 11.61.3 Member Data Documentation

### 11.61.3.2 FX3MCSDataDeviceIdOffset

### 11.61.3.3 FX3MCSDatIFB1ImageOffset

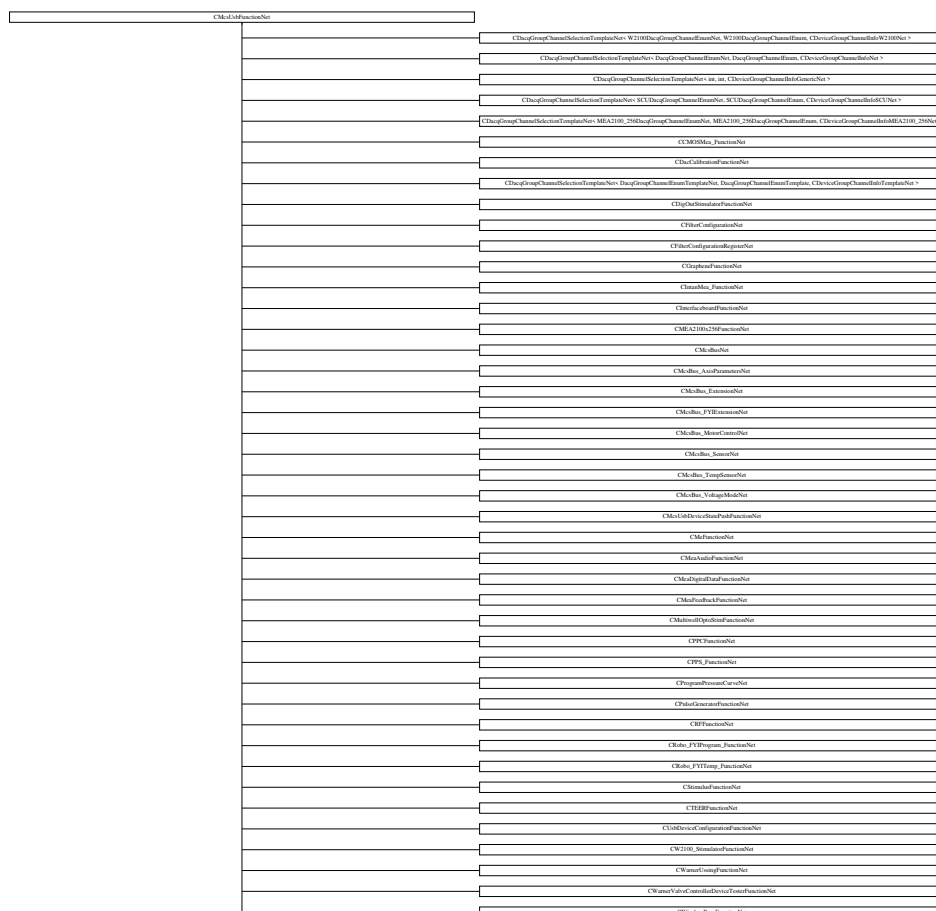
#### 11.61.3.4 FX3MCSDatIFB2ImageOffset

### 11.61.3.5 FX3MCSDataVersionOffset

```
const uint32_t FX3MCSDataVersionOffset = 0x8 [static]
```

## 11.62 CMcsUsbFunctionNet Class Reference

Inheritance diagram for CMcsUsbFunctionNet:



**Public Member Functions**

- [CMcsUsbFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- virtual [~CMcsUsbFunctionNet](#) (void)
- [!CMcsUsbFunctionNet](#) ()
- void [ThrowCUsbExceptionNetOnError](#) (uint32\_t status)

**Protected Member Functions**

- [CMcsUsbFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> mcsusbfunction)

**Protected Attributes**

- [CMcsUsbNet](#)<sup>^</sup> [m\\_pMcsUsb](#)
- [CMcsUsbFunction](#) \* [m\\_pMcsUsbFunction](#)

**11.62.1 Constructor & Destructor Documentation**

**11.62.1.1 [CMcsUsbFunctionNet](#)()** [1/2] [CMcsUsbFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> *mcsusb* )

**11.62.1.2 [~CMcsUsbFunctionNet](#)()** virtual [~CMcsUsbFunctionNet](#) (  
void ) [virtual]

**11.62.1.3 ["!CMcsUsbFunctionNet](#)()** [!CMcsUsbFunctionNet](#) ( )

**11.62.1.4 [CMcsUsbFunctionNet](#)()** [2/2] [CMcsUsbFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> *mcsusb*,  
[CMcsUsbFunctionPointerContainer](#)<sup>^</sup> *mcsusbfunction* ) [protected]

**11.62.2 Member Function Documentation**

**11.62.2.1 [ThrowCUsbExceptionNetOnError](#)()** void [ThrowCUsbExceptionNetOnError](#) (  
uint32\_t *status* )

### 11.62.3 Member Data Documentation

**11.62.3.1** `m_pMcsUsb` [CMcsUsbNet](#) ^ `m_pMcsUsb` [protected]

**11.62.3.2** `m_pMcsUsbFunction` `CMcsUsbFunction*` `m_pMcsUsbFunction` [protected]

## 11.63 CMcsUsbFunctionPointerContainer Class Reference

## 11.64 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]

### 11.64.1 Detailed Description

McsUsbListEntryNet identifies a connected device.

### 11.64.2 Constructor & Destructor Documentation

**11.64.2.1** `~CMcsUsbListEntryNet()` `~CMcsUsbListEntryNet ( )`

### 11.64.3 Member Function Documentation

**11.64.3.1** `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. |
|------------|----------------------------------------------------------|

**11.64.3.2** `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.

**11.64.3.3** `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.

**11.64.3.4 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.

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

Returns the number of devices connected to the computer.

**Returns**

The number of devices.

**11.64.3.6 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.

**11.64.3.7 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)". |
|---------------|----------------------------------------------------------------|

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

**11.64.4 Property Documentation**

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

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

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

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

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

#### 11.64.4.4 **Manufacturer** `String^ Manufacturer [get]`

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

#### 11.64.4.5 **Product** `String^ Product [get]`

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

#### 11.64.4.6 **SerialNumber** `String^ SerialNumber [get]`

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

### 11.65 **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< [DeviceldNet](#)^>^ [DeviceldList](#))  
*Initializes a new instance of [CMcsUsbListNet](#) class.*
- [~CMcsUsbListNet](#) ()  
*Destructor: called by [Dispose\(\)](#)*
- [!CMcsUsbListNet](#) ()  
*Finalizer: called by GC before collecting*
- void [SetStringFormat](#) (String ^ format)  
*Specify the text the [CMcsUsbListEntryNet.ToString\(\)](#) function should return. The special code N expands to the device name and S expands to the serial number of the device.*
- uint32\_t [GetNumberOfDevices](#) ()  
*Gets the number of devices currently in the list.*
- [CMcsUsbListEntryNet](#) ^ [GetUsbListEntry](#) (unsigned int index)  
*Returns one [CMcsUsbListEntryNet](#) from the list of USB Devices connected to the computer.*
- array< [CMcsUsbListEntryNet](#)^> ^ [GetUsbListEntries](#) ()  
*Returns all entries from the list of USB Devices connected to the computer.*
- bool [IsDeviceTypeOf](#) ([CMcsUsbListEntryNet](#)^ entry, [DeviceEnumNet](#) McsUsbDevice)

#### Properties

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



**Events**

- [OnDeviceArrivalRemoval](#)^ [DeviceArrival](#)
- [OnDeviceArrivalRemoval](#)^ [DeviceRemoval](#)

**11.65.1 Detailed Description**

Class to handle a list of connected MCS USB devices.

**11.65.2 Constructor & Destructor Documentation**

**11.65.2.1 CMcsUsbListNet()** [1/2] [CMcsUsbListNet](#) ( [DeviceEnumNet](#) *McsUsbDevice* )

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

**11.65.2.2 CMcsUsbListNet()** [2/2] [CMcsUsbListNet](#) ( [array](#)< [DeviceIdNet](#)^>^ *DeviceIdList* )

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

**11.65.2.3 ~CMcsUsbListNet()** [~CMcsUsbListNet](#) ( )

Destructor: called by Dispose()

**11.65.2.4 ~!CMcsUsbListNet()** [!CMcsUsbListNet](#) ( )

Finalizer: called by GC before collecting

**11.65.3 Member Function Documentation**

**11.65.3.1 GetNumberOfDevices()** [uint32\\_t](#) [GetNumberOfDevices](#) ( )

Gets the number of devices currently in the list.

**Returns**

The number of devices currently in the list.

**11.65.3.2 GetUsbListEntries()** [array](#)<[CMcsUsbListEntryNet](#)^> ^ [GetUsbListEntries](#) ( )

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

**11.65.3.3 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. |
|--------------|-----------------------------|

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

**11.65.3.5 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)". |
|---------------|----------------------------------------------------------------|

**11.65.4 Property Documentation**

**11.65.4.1 Count** `uint32_t Count [get]`

Gets the number of devices currently in the list.

**11.65.5 Event Documentation**

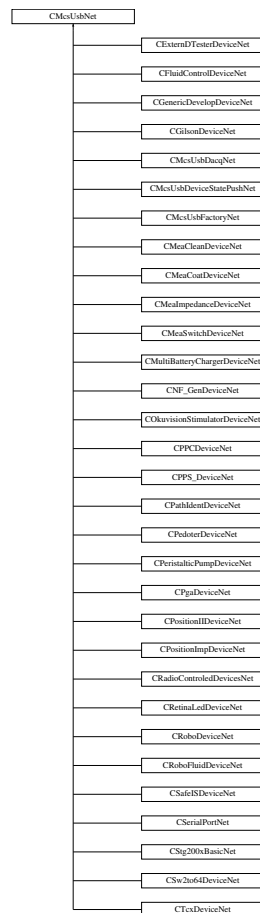
**11.65.5.1 DeviceArrival** `OnDeviceArrivalRemoval^ DeviceArrival`

**11.65.5.2 DeviceRemoval** `OnDeviceArrivalRemoval^ DeviceRemoval`

## 11.66 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](#)<sup>^</sup> entry)  
*Opens a connection to the device.*
- virtual [uint32\\_t](#) [Connect](#) ([CMcsUsbListEntryNet](#)<sup>^</sup> 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](#) ()
- 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)
- void [ModifyRegister](#) (unsigned int reg, int startpos, unsigned int length, unsigned int value)
- 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 queried 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\\_ScanningsPending](#) = ( (0xA0220000L) | 0x0036 )

## Properties

- virtual String^ [SerialNumber](#) [get]

### 11.66.1 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.

### 11.66.2 Constructor & Destructor Documentation

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

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

#### 11.66.2.2 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. |
|----------------|-------------------------------------------|

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

#### 11.66.2.4 "!CMcsUsbNet() `!CMcsUsbNet ( )`

### 11.66.3 Member Function Documentation

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

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

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

Opens a connection to the device.

**Returns**

Error Status. 0 on success.

**11.66.3.4 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.

**11.66.3.5 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.

**11.66.3.6 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.

**11.66.3.7 CyclePort()** `uint32_t CyclePort ( )`

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

Disconnect from a device.

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

**11.66.3.10 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. |
|---------------|----------------------------------------------|

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

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

**11.66.3.13 EraseEepromRegisterPreconfig()** [3/3] `void EraseEepromRegisterPreconfig (`  
    `uint32_t EEPROMBase,`  
    `uint32_t DMA_reg,`  
    `uint32_t EEPROMSize,`  
    `uint32_t EepromStartAddress )`

**11.66.3.14 GetConfiguration()** `uint8_t GetConfiguration ( )`

**11.66.3.15 GetDeviceCannotStallOutRequests()** `bool GetDeviceCannotStallOutRequests ( )`

**11.66.3.16 GetDeviceCapableSpeed()** `McsUsbSpeedEnumNet GetDeviceCapableSpeed ( )`

**11.66.3.17 GetDeviceEnum()** `DeviceEnumNet GetDeviceEnum ( )`

**11.66.3.18 GetDeviceId()** `DeviceIdNet ^ GetDeviceId ( )`

**11.66.3.19 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.

**11.66.3.20 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.

**11.66.3.21 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".

**11.66.3.22 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.

**11.66.3.23 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.

**11.66.3.24 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.

**11.66.3.25 GetHardwareRevision()** `String ^ GetHardwareRevision ( )`

**11.66.3.26 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

**11.66.3.27 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

**11.66.3.28 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

**11.66.3.29 GetLastUSBError()** unsigned int GetLastUSBError ( )

**11.66.3.30 GetMea21UsbPort()** uint32\_t GetMea21UsbPort ( )

Gets the USB port if an IFB that is used by this connection

**Returns**

number of used port; range: 0..1

**11.66.3.31 GetNumConfigurations()** uint8\_t GetNumConfigurations ( )

**11.66.3.32 GetSerialNumber()** virtual String ^ GetSerialNumber ( ) [virtual]

Query the Serial Number of the device.

**Returns**

The Serial Number.

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

**11.66.3.34 GetSoftwareKeyString()** [1/2] String ^ GetSoftwareKeyString (   
 SoftwareKeyProgrammIdsNet::ProgrammIdsNet *ProgrammID*,   
 uint8\_t *majorversion* )

**11.66.3.35 GetSoftwareKeyString()** [2/2] String ^ GetSoftwareKeyString (   
 uint8\_t *ProgrammID*,   
 uint8\_t *majorversion* )

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

**11.66.3.37 GetStatusOfLastCommand()** `uint32_t GetStatusOfLastCommand ( )`

Gets the status of the last call to the McsUsb Library.

**Returns**

The Error Status of the last McsUsb command. 0 on success.

**11.66.3.38 GetUsbListEntry()** `CMcsUsbListEntryNet ^ GetUsbListEntry ( )`

**11.66.3.39 GetVersion()** [1/2] `DriverVersionNet ^ GetVersion ( )`

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

**11.66.3.41 HasSoftwareKey()** [1/2] `bool HasSoftwareKey ( SoftwareKeyProgrammIdsNet::ProgrammIdsNet ProgrammID, uint8_t majorversion )`

**11.66.3.42 HasSoftwareKey()** [2/2] `bool HasSoftwareKey ( uint8_t ProgrammID, uint8_t majorversion )`

**11.66.3.43 IsConnected()** `virtual bool IsConnected ( ) [virtual]`

Check if a device is Connected.

**Returns**

true if the device is connected.

**11.66.3.44 IsDeviceHighSpeed()** `bool IsDeviceHighSpeed ( )`

**11.66.3.45 IsDeviceHighSpeedCapable()** `bool IsDeviceHighSpeedCapable ( )`

**11.66.3.46 IsExceptionsEnabled()** `bool IsExceptionsEnabled ( )`

**11.66.3.47 ModifyRegister()** `void ModifyRegister (`  
    `unsigned int reg,`  
    `int startpos,`  
    `unsigned int length,`  
    `unsigned int value )`

**11.66.3.48 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.

**11.66.3.49 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.

**11.66.3.50 MultibootGetSelectedImage()** `uint32_t MultibootGetSelectedImage ( )`

Gets sector index of selected FPGA boot image on IFB

#### Returns

Sector index of image; range: 0..2

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

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

#### Returns

Throws exception on error.

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

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

**11.66.3.54 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 )`

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

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

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



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

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

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

rescans and activates a headstage

#### Parameters

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

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

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

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

**11.66.3.64 TxnGetSerialNumber()** unsigned int TxnGetSerialNumber ( )

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

## 11.66.4 Member Data Documentation

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

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

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

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

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

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

**11.66.4.7 Status\_Canceling** `const uint32_t Status_Canceling = (0xE0120000L)` [static]

**11.66.4.8 Status\_ConnectedPipes** `const uint32_t Status_ConnectedPipes = (0xE01F000AL)` [static]

**11.66.4.9 Status\_ControlNotOwned** `const uint32_t Status_ControlNotOwned = (0xE0100D00L)` [static]

**11.66.4.10 Status\_Crc** `const uint32_t Status_Crc = (0xE0100001L)` [static]

**11.66.4.11 Status\_DataOverrun** `const uint32_t Status_DataOverrun = (0xE0100008L)` [static]

**11.66.4.12 Status\_DataToggleMismatch** `const uint32_t Status_DataToggleMismatch = (0xE0100003L)`  
[static]

**11.66.4.13 Status\_DataUnderrun** `const uint32_t Status_DataUnderrun = (0xE0100009L)` [static]

**11.66.4.14 Status\_DeviceLocked** `const uint32_t Status_DeviceLocked = (0xE01F0010L)` [static]

**11.66.4.15 Status\_DeviceNotFound** `const uint32_t Status_DeviceNotFound = (0xE01F0003L)` [static]

**11.66.4.16 Status\_DeviceRemoved** `const uint32_t Status_DeviceRemoved = (0xE01F0008L)` [static]

**11.66.4.17 Status\_DevNotResponding** `const uint32_t Status_DevNotResponding = (0xE0100005L)`  
[static]

**11.66.4.18 Status\_EndpointHalted** `const uint32_t Status_EndpointHalted = (0xE0100030L)` [static]

**11.66.4.19 Status\_ErrorBusy** `const uint32_t Status_ErrorBusy = (0xE0100400L)` [static]

**11.66.4.20 Status\_ErrorShortTransfer** `const uint32_t Status_ErrorShortTransfer = (0xE0100900L)`  
[static]

**11.66.4.21 Status\_Fifo** `const uint32_t Status_Fifo = (0xE0100010L)` [static]

**11.66.4.22 Status\_FrameControlOwned** `const uint32_t Status_FrameControlOwned = (0xE0100C00L)`  
[static]

**11.66.4.23 Status\_InternalHcError** `const uint32_t Status_InternalHcError = (0xE0100800L) [static]`

**11.66.4.24 Status\_InvalidDeviceHandle** `const uint32_t Status_InvalidDeviceHandle = (0xE0100013L) [static]`

**11.66.4.25 Status\_InvalidHandle** `const uint32_t Status_InvalidHandle = (0xE0100012L) [static]`

**11.66.4.26 Status\_InvalidParameter** `const uint32_t Status_InvalidParameter = (0xE0100300L) [static]`

**11.66.4.27 Status\_InvalidPipeHandle** `const uint32_t Status_InvalidPipeHandle = (0xE0100600L) [static]`

**11.66.4.28 Status\_InvalidUrbFunction** `const uint32_t Status_InvalidUrbFunction = (0xE0100200L) [static]`

**11.66.4.29 Status\_IoPending** `const uint32_t Status_IoPending = (0xE01F0006L) [static]`

**11.66.4.30 Status\_IoTimeout** `const uint32_t Status_IoTimeout = (0xE01F0007L) [static]`

**11.66.4.31 Status\_IsochRequestFailed** `const uint32_t Status_IsochRequestFailed = (0xE0100B00L) [static]`

**11.66.4.32 Status\_LastUsbErrorMismatch** `const uint32_t Status_LastUsbErrorMismatch = (0xE01F0022L) [static]`

**11.66.4.33 Status\_NoBandwidth** `const uint32_t Status_NoBandwidth = (0xE0100700L) [static]`

**11.66.4.34 Status\_NoMemory** `const uint32_t Status_NoMemory = (0xE0100100L) [static]`

**11.66.4.35 Status\_NoSuchDevice** `const uint32_t Status_NoSuchDevice = (0xE01F0002L) [static]`

**11.66.4.36 Status\_NotAccessed** `const uint32_t Status_NotAccessed = (0xE010000FL) [static]`

**11.66.4.37 Status\_NotSupported** `const uint32_t Status_NotSupported = (0xE01F0005L) [static]`

**11.66.4.38 Status\_PidCheckFailure** `const uint32_t Status_PidCheckFailure = (0xE0100006L) [static]`

**11.66.4.39 Status\_PipeNotLinked** `const uint32_t Status_PipeNotLinked = (0xE01F0009L) [static]`

**11.66.4.40 Status\_RequestFailed** `const uint32_t Status_RequestFailed = (0xE0100500L) [static]`

**11.66.4.41 Status\_RequestMutexFailed** `const uint32_t Status_RequestMutexFailed = (0xE01F0021L) [static]`

**11.66.4.42 Status\_RequestMutexTimeout** `const uint32_t Status_RequestMutexTimeout = (0xE01F0020L) [static]`

**11.66.4.43 Status\_Stall** `const uint32_t Status_Stall = (0xE0100004L) [static]`

**11.66.4.44 Status\_Unconfigured** `const uint32_t Status_Unconfigured = (0xE0110002L) [static]`

**11.66.4.45 Status\_UnexpectedPid** `const uint32_t Status_UnexpectedPid = (0xE0100007L) [static]`

**11.66.4.46 WPAError\_ScanningIsPending** `const uint32_t WPAError_ScanningIsPending = ( (0x←A0220000L) | 0x0036 ) [static]`

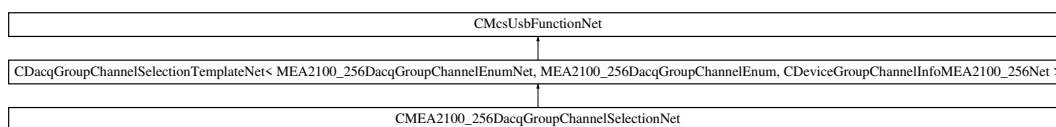
## 11.66.5 Property Documentation

**11.66.5.1 SerialNumber** `virtual String^ SerialNumber [get]`

## 11.67 CMcsUsbPointerContainer Class Reference

## 11.68 CMEA2100\_256DacqGroupChannelSelectionNet Class Reference

Inheritance diagram for CMEA2100\_256DacqGroupChannelSelectionNet:



### Public Member Functions

- [CMEA2100\\_256DacqGroupChannelSelectionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)

### Additional Inherited Members

#### 11.68.1 Constructor & Destructor Documentation

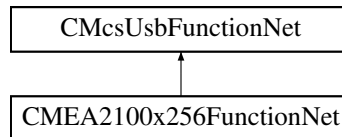
**11.68.1.1 CMEA2100\_256DacqGroupChannelSelectionNet()** [CMEA2100\\_256DacqGroupChannelSelectionNet](#)  
(  
    [CMcsUsbNet](#)<sup>^</sup> mcsusb )



## 11.69 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 DAC channels available per well is [Mcs::Usb::CStg200xBasicNet::GetNumberOfAnalogChannels](#) divided by [Mcs::Usb::CStg200xBasicNet::GetNumberOfStimulationWells](#).*
- void [SetLayoutConfiguration](#) ([StimulationLayoutConfigurationEnumNet](#) LayoutConfiguration)  
*Sets the stimulation layout configuration. Can be single well, 6-well or 9-well. The number of DAC channels available per well is [Mcs::Usb::CStg200xBasicNet::GetNumberOfAnalogChannels](#) divided by [Mcs::Usb::CStg200xBasicNet::GetNumberOfStimulationWells](#).*

### Additional Inherited Members

#### 11.69.1 Detailed Description

[CMEA2100x256FunctionNet](#) is the class to control the MEA2100-256 device needs #include "Stg200xNet.h" to resolve documentation reference

#### 11.69.2 Constructor & Destructor Documentation

**11.69.2.1 CMEA2100x256FunctionNet() [1/2]** [CMEA2100x256FunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb,  
[CMcsUsbFunctionPointerContainer](#)<sup>^</sup> pMEA2100x256FunctionPointerContainer )

Initializes a new instance of the [CMEA2100x256FunctionNet](#) class.

**11.69.2.2 CMEA2100x256FunctionNet()** [2/2] `CMEA2100x256FunctionNet ( CMcsUsbNet^ mcsusb )`

**11.69.2.3 ~CMEA2100x256FunctionNet()** `virtual ~CMEA2100x256FunctionNet ( ) [virtual]`

**11.69.2.4 ~!CMEA2100x256FunctionNet()** `!CMEA2100x256FunctionNet ( )`

### 11.69.3 Member Function Documentation

**11.69.3.1 GetLayoutConfiguration()** `StimulationLayoutConfigurationEnumNet GetLayoutConfiguration ( )`

Gets the stimulation layout configuration. Can be single well, 6-well or 9-well. The number of DAC channels available per well is `Mcs::Usb::CStg200xBasicNet::GetNumberOfAnalogChannels` divided by `Mcs::Usb::CStg200xBasicNet::GetNumberOfStimulationSourcesPerElectrode`.

#### Returns

The currently active stimulation layout configuration.

**11.69.3.2 SetLayoutConfiguration()** `void SetLayoutConfiguration ( StimulationLayoutConfigurationEnumNet LayoutConfiguration )`

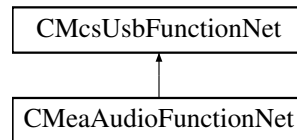
Sets the stimulation layout configuration. Can be single well, 6-well or 9-well. The number of DAC channels available per well is `Mcs::Usb::CStg200xBasicNet::GetNumberOfAnalogChannels` divided by `Mcs::Usb::CStg200xBasicNet::GetNumberOfStimulationSourcesPerElectrode`.

#### Parameters

|                            |                                           |
|----------------------------|-------------------------------------------|
| <i>LayoutConfiguration</i> | The new stimulation layout configuration. |
|----------------------------|-------------------------------------------|

## 11.70 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>><sup>^</sup> channels)  
*Sets the electrode to monitor and amplification for the audio channels.*
- virtual uint32\_t [SetAudioChannels](#) (array< [s\\_setaudionet](#)<sup>^</sup>><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>><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>><sup>^</sup>% channels, unsigned int virtualDevice)  
*Gets the electrode to monitor and amplification for the audio channels.*

## Additional Inherited Members

### 11.70.1 Constructor & Destructor Documentation

**11.70.1.1 CMeaAudioFunctionNet()** [1/2] [CMeaAudioFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb,  
[CMcsUsbFunctionPointerContainer](#)<sup>^</sup> meaAudioFunctionPointerContainer )

**11.70.1.2 CMeaAudioFunctionNet()** [2/2] [CMeaAudioFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb )

### 11.70.2 Member Function Documentation

**11.70.2.1 GetAudioChannels()** [1/2] virtual uint32\_t [GetAudioChannels](#) (  
[System::Runtime::InteropServices::Out] array< [s\\_setaudionet](#)<sup>^</sup>><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.

**11.70.2.2 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.

**11.70.2.3 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.

**11.70.2.4 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.

**11.70.2.5 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. |
|----------------------|------------------------|

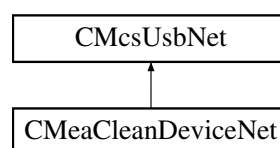
## Returns

Error Status. 0 on success.

## 11.71 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

### 11.71.1 Detailed Description

[CMeaCleanDeviceNet](#) is the class to access the MEA Clean device.

### 11.71.2 Constructor & Destructor Documentation

#### 11.71.2.1 [CMeaCleanDeviceNet](#)() [CMeaCleanDeviceNet](#) ( )

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

**11.71.2.2** `~CMeaCleanDeviceNet()` `virtual ~CMeaCleanDeviceNet ( ) [virtual]`

**11.71.2.3** `!CMeaCleanDeviceNet()` `!CMeaCleanDeviceNet ( )`

### 11.71.3 Member Function Documentation

**11.71.3.1** `GetCycle()` `int32_t GetCycle ( )`

Gets the current cycle.

#### Returns

The cycle number.

**11.71.3.2** `GetCycles()` `uint32_t GetCycles ( )`

Gets the number of cycles.

#### Returns

The number of cycles to run for.

**11.71.3.3** `GetMaxVoltage()` `int32_t GetMaxVoltage ( )`

Gets the upper voltage level

#### Returns

The upper voltage level in mV.

**11.71.3.4** `GetMinVoltage()` `int32_t GetMinVoltage ( )`

Gets the lower voltage level.

#### Returns

The lower voltage level in mV.

**11.71.3.5 GetOutputVoltage()** `int32_t GetOutputVoltage ( )`

Gets the output voltage.

**Returns**

The output voltage in mV.

**11.71.3.6 GetSlope()** `uint32_t GetSlope ( )`

Gets the voltage slope.

**Returns**

The voltage slope in mV/s.

**11.71.3.7 IsRunning()** `bool IsRunning ( )`

Gets if the MEA Clean device is running.

**Returns**

"true" when a run is in progress, otherwise "false".

**11.71.3.8 SetCycles()** `void SetCycles (   
uint32_t cycles )`

Sets the number of cycles.

**Parameters**

|               |                                            |
|---------------|--------------------------------------------|
| <i>cycles</i> | The number of cycles to run for (0 .. 99). |
|---------------|--------------------------------------------|

**11.71.3.9 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). |
|-------------------|------------------------------------------------|



**11.71.3.10 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). |
|-------------------|------------------------------------------------|

**11.71.3.11 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). |
|---------------------|------------------------------------------------|

**11.71.3.12 Start()** `void Start ( )`

Starts a MEA Clean run.

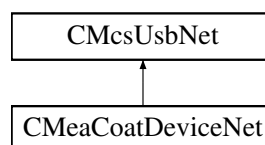
**11.71.3.13 Stop()** `void Stop ( )`

Stops a MEA Clean run.

## 11.72 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

### 11.72.1 Detailed Description

[CMeaCoatDeviceNet](#) is the class to access the MEA Coat device.

## 11.72.2 Constructor & Destructor Documentation

### 11.72.2.1 CMeaCoatDeviceNet() CMeaCoatDeviceNet ( )

Initializes a new instance of the CMeaCoatDeviceNet class.

### 11.72.2.2 ~CMeaCoatDeviceNet() virtual ~CMeaCoatDeviceNet ( ) [virtual]

### 11.72.2.3 !CMeaCoatDeviceNet() !CMeaCoatDeviceNet ( )

## 11.72.3 Member Function Documentation

### 11.72.3.1 GetCurrentCycle() int32\_t GetCurrentCycle ( )

Gets the current cycle.

#### Returns

The cycle number.

### 11.72.3.2 GetCycles() uint32\_t GetCycles ( )

Gets the number of cycles.

#### Returns

The number of cycles to run for.

### 11.72.3.3 GetDuration() uint32\_t GetDuration ( )

Gets the duration of a MEA Coat run.

#### Returns

The duration in ms.

**11.72.3.4 GetMaxCurrent()** `uint32_t GetMaxCurrent ( )`

Gets the limit of the current ramp (absolute value).

**Returns**

The limit of the current ramp in pA (absolute value).

**11.72.3.5 GetOffsetCurrent()** `int32_t GetOffsetCurrent ( )`

Gets the offset of the current.

**Returns**

The offset of the current in pA.

**11.72.3.6 GetOutputCurrent()** `int32_t GetOutputCurrent ( )`

Gets the output current.

**Returns**

The output current in pA.

**11.72.3.7 GetPauseDuration()** `uint32_t GetPauseDuration ( )`

Gets the duration of the pause between MEA Coat pulses.

**Returns**

The duration in ms.

**11.72.3.8 GetSlope()** `int32_t GetSlope ( )`

Gets the current slope.

**Returns**

The current slope in pA/s.

**11.72.3.9 GetTimeInPause()** `int32_t GetTimeInPause ( )`

Gets the time in the pause.

**Returns**

The time in the pause in ms.

**11.72.3.10 GetTimeInPlateau()** `int32_t GetTimeInPlateau ( )`

Gets the time in the plateau.

**Returns**

The time in the plateau in ms.

**11.72.3.11 IsRunning()** `bool IsRunning ( )`

Gets if the MEA Clean device is running.

**Returns**

"true" when a run is in progress, otherwise "false".

**11.72.3.12 SetCycles()** `void SetCycles (   
uint32_t cycles )`

Sets the number of cycles.

**Parameters**

|               |                                            |
|---------------|--------------------------------------------|
| <i>cycles</i> | The number of cycles to run for (0 .. 99). |
|---------------|--------------------------------------------|

**11.72.3.13 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). |
|-----------------|---------------------------------------|

**11.72.3.14 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). |
|-------------------|-------------------------------------------------------------------|

**11.72.3.15 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). |
|----------------------|-------------------------------------------------|

**11.72.3.16 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). |
|----------------------|---------------------------------------|

**11.72.3.17 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). |
|---------------------|---------------------------------------------------|

**11.72.3.18 Start()** `void Start ( )`

Starts a MEA Coat run.

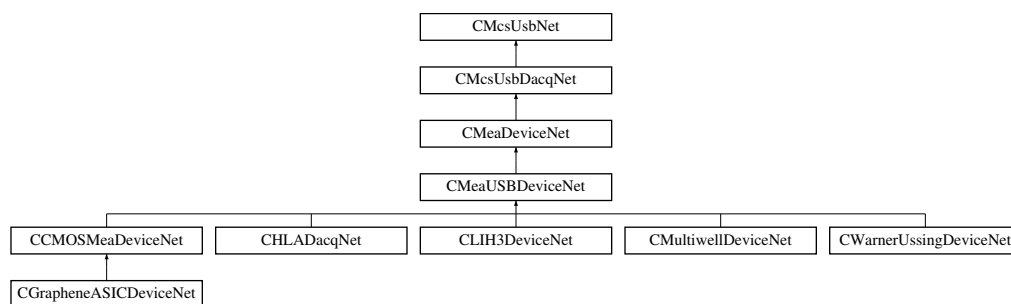
#### 11.72.3.19 Stop() void Stop ( )

Stops a MEA Coat run.

## 11.73 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

### 11.73.1 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

### 11.73.2 Constructor & Destructor Documentation

#### 11.73.2.1 [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. |
|----------------|-------------------------------------------|



**11.73.2.2 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. |
|--------------|------------------------------------------|

**11.73.2.3 ~CMeaDeviceNet()** `~CMeaDeviceNet ( )`

### 11.73.3 Member Function Documentation

**11.73.3.1 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.

**11.73.3.2 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.

**11.73.3.3 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.

**11.73.3.4 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.

**11.73.3.5 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.

**11.73.3.6 GetEnumerationSpeed()** `int32_t GetEnumerationSpeed ( )`**11.73.3.7 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.

**11.73.3.8 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. |
|--------------------|--------------------|

**11.73.3.9 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↔<br/>_IF</i> | Number of analog channels from the Interfaceboard. |
|----------------------------------|----------------------------------------------------|

**Parameters**

|                      |                       |
|----------------------|-----------------------|
| <i>virtualDevice</i> | virtualDevice to use. |
|----------------------|-----------------------|

**Returns**

Error Status. 0 on success.

**11.73.3.10 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.

**11.73.3.11 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.

**11.73.3.12 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.

**11.73.3.13 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.

**11.73.4 Property Documentation**

**11.73.4.1 AnalogGain** `virtual int AnalogGain [get]`

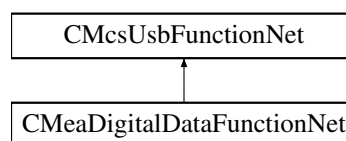
The gain of the analog inputs of the device. Value is gain times 1000, a value of 1000 corresponds to a gain of 1.0.

**11.73.4.2 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.

**11.73.4.3 MeaAudioFunctionNet** `CMeaAudioFunctionNet^ MeaAudioFunctionNet [get]`**11.73.4.4 MeaDigitalDataFunctionNet** `CMeaDigitalDataFunctionNet^ MeaDigitalDataFunctionNet [get]`**11.73.4.5 MeaFeedbackFunctionNet** `CMeaFeedbackFunctionNet^ MeaFeedbackFunctionNet [get]`**11.73.4.6 MeFunctionNet** `CMeFunctionNet^ MeFunctionNet [get]`**11.73.4.7 W2100\_FunctionNet** `CW2100_FunctionNet^ W2100_FunctionNet [get]`**11.73.4.8 WClassicFunctionNet** `CWClassicFunctionNet^ WClassicFunctionNet [get]`**11.74 CMeaDigitalDataFunctionNet Class Reference**

Inheritance diagram for CMeaDigitalDataFunctionNet:

**Public Member Functions**

- `CMeaDigitalDataFunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ meaDigitalFunctionPointerContainer)`
- `CMeaDigitalDataFunctionNet (CMcsUsbNet^ 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

### 11.74.1 Constructor & Destructor Documentation

**11.74.1.1 CMeaDigitalDataFunctionNet() [1/2]** `CMeaDigitalDataFunctionNet (   
CMcsUsbNet^ mcsusb,   
CMcsUsbFunctionPointerContainer^ meaDigitalFunctionPointerContainer )`

**11.74.1.2 CMeaDigitalDataFunctionNet() [2/2]** `CMeaDigitalDataFunctionNet (   
CMcsUsbNet^ mcsusb )`

### 11.74.2 Member Function Documentation

**11.74.2.1 GetDigitalData()** `unsigned int GetDigitalData ( )`

Get the value of the digital output.

#### Returns

Value on the digital data register.

**11.74.2.2 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. |
|--------------|------------|



**11.74.2.3 SetDigitalData()** [2/2] void SetDigitalData (   
     unsigned int *digital\_value*,   
     unsigned int *digital\_value\_mask* )

Generate a value on the digital output.

Parameters

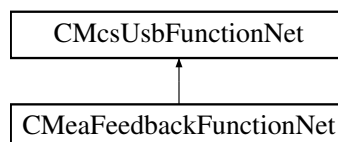
|                      |               |
|----------------------|---------------|
| <i>digital_value</i> | Value to set. |
|----------------------|---------------|

Parameters

|                           |                  |
|---------------------------|------------------|
| <i>digital_value_mask</i> | Mask for change. |
|---------------------------|------------------|

## 11.75 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 [FeedbackSetCreateFilter](#) (unsigned char filter, [CCreateFilterNet](#)<sup>^</sup> createFilter)
- 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 resultchannel)
- void [FeedbackSetNumberOfRateDetectors](#) (unsigned short number)
- void [FeedbackSetRateDetector](#) (unsigned short position, unsigned short resultchannel, unsigned short trigger, 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 >^ sourcechannel, unsigned short resultchannel, unsigned int lookup)
- void [FeedbackSetNumberOfTriggers](#) (unsigned short number)
- void [FeedbackSetTrigger](#) (unsigned short position, unsigned short sourcechannel, unsigned short resultchannel, unsigned short trigger, unsigned short totzeit)
- void [FeedbackSetAnalogSource](#) ([AnalogSourceEnumNet](#) AnalogSource, unsigned int Channels, unsigned int Offset)
- void [FeedbackSetTriggerTotzeitFactor](#) (unsigned short trigger\_factor, unsigned short totzeit\_factor)
- void [FeedbackGetTriggerTotzeitFactor](#) ([System::Runtime::InteropServices::Out]unsigned short% trigger\_factor, [System::Runtime::InteropServices::Out]unsigned short% totzeit\_factor)

## Additional Inherited Members

### 11.75.1 Constructor & Destructor Documentation

**11.75.1.1 CMeaFeedbackFunctionNet()** [1/2] [CMeaFeedbackFunctionNet](#) (   
[CMcsUsbNet](#)^ mcsusb,   
[CMcsUsbFunctionPointerContainer](#)^ meaFeedbackFunctionNet )

**11.75.1.2 CMeaFeedbackFunctionNet()** [2/2] [CMeaFeedbackFunctionNet](#) (   
[CMcsUsbNet](#)^ mcsusb )

### 11.75.2 Member Function Documentation

**11.75.2.1 FeedbackGetSampleTimerCount()** unsigned int FeedbackGetSampleTimerCount (   
[System::Runtime::InteropServices::Out] unsigned int% CurrentCount,   
[System::Runtime::InteropServices::Out] unsigned int% LastKnownCount,   
[System::Runtime::InteropServices::Out] bool% On )

**11.75.2.2 FeedbackGetTriggerTotzeitFactor()** void FeedbackGetTriggerTotzeitFactor (   
 [System::Runtime::InteropServices::Out] unsigned short% *trigger\_factor*,   
 [System::Runtime::InteropServices::Out] unsigned short% *totzeit\_factor* )

**11.75.2.3 FeedbackSetAnalogSource()** void FeedbackSetAnalogSource (   
 *AnalogSourceEnumNet* *AnalogSource*,   
 unsigned int *Channels*,   
 unsigned int *Offset* )

**11.75.2.4 FeedbackSetChannelFilter()** void FeedbackSetChannelFilter (   
 short *channel*,   
 char *filter* )

**11.75.2.5 FeedbackSetCreateFilter()** void FeedbackSetCreateFilter (   
 unsigned char *filter*,   
 *CCreateFilterNet*^ *createFilter* )

**11.75.2.6 FeedbackSetDigitalMapping()** void FeedbackSetDigitalMapping (   
 unsigned short *channel*,   
 unsigned short *outmapping*,   
 unsigned short *inmapping* )

**11.75.2.7 FeedbackSetFeedback()** void FeedbackSetFeedback (   
 unsigned char *on*,   
 unsigned short *digoutmask*,   
 unsigned short *diginmask* )

**11.75.2.8 FeedbackSetFilterOff()** void FeedbackSetFilterOff ( )

**11.75.2.9 FeedbackSetFilterParameter()** void FeedbackSetFilterParameter (   
 unsigned char *filter*,   
 array< short >^ *parameters* )

**11.75.2.10 FeedbackSetFilterParameter32()** void FeedbackSetFilterParameter32 (   
 unsigned char *filter*,  
 array< int >^ *parameters* )

**11.75.2.11 FeedbackSetGlobalChannelFilter()** void FeedbackSetGlobalChannelFilter (   
 char *filter*,  
 unsigned short *firstchannel*,  
 unsigned short *lastchannel* )

**11.75.2.12 FeedbackSetIIRFilterParameter()** void FeedbackSetIIRFilterParameter (   
 unsigned char *filter*,  
 int *length*,  
 array< double >^ *parameters* )

**11.75.2.13 FeedbackSetLogic()** void FeedbackSetLogic (   
 unsigned short *position*,  
 array< unsigned short >^ *sourcechannel*,  
 unsigned short *resultchannel*,  
 unsigned int *lookup* )

**11.75.2.14 FeedbackSetMkFilter()** void FeedbackSetMkFilter (   
 unsigned char *filter*,  
 String^ *filtertype*,  
 double *cheb\_ripple*,  
 String^ *passtype*,  
 int *order*,  
 double *alpha1*,  
 double *alpha2* )

**11.75.2.15 FeedbackSetNumberOfLogics()** void FeedbackSetNumberOfLogics (   
 unsigned short *number* )

**11.75.2.16 FeedbackSetNumberOfRateCounter()** void FeedbackSetNumberOfRateCounter (   
 unsigned short *number* )

**11.75.2.17 FeedbackSetNumberOfRateDetectors()** void FeedbackSetNumberOfRateDetectors ( unsigned short *number* )

**11.75.2.18 FeedbackSetNumberOfSpikeDetectors()** void FeedbackSetNumberOfSpikeDetectors ( unsigned short *number* )

**11.75.2.19 FeedbackSetNumberOfTriggers()** void FeedbackSetNumberOfTriggers ( unsigned short *number* )

**11.75.2.20 FeedbackSetRateCounter()** void FeedbackSetRateCounter ( unsigned short *position*,  
unsigned short *sourcechannel*,  
unsigned short *resultchannel* )

**11.75.2.21 FeedbackSetRateDetector()** void FeedbackSetRateDetector ( unsigned short *position*,  
unsigned short *resultchannel*,  
unsigned short *trigger*,  
unsigned short *totzeit*,  
unsigned short *pulses*,  
unsigned int *duration1*,  
unsigned int *duration2* )

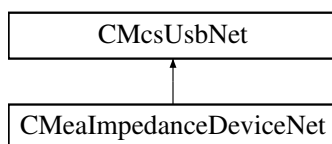
**11.75.2.22 FeedbackSetSpikeDetectorThreshold()** void FeedbackSetSpikeDetectorThreshold ( unsigned short *position*,  
unsigned short *sourcechannel*,  
unsigned short *resultchannel*,  
unsigned short *trigger*,  
unsigned short *totzeit*,  
int *threshold1*,  
int *threshold2*,  
short *slope* )

**11.75.2.23 FeedbackSetTrigger()** void FeedbackSetTrigger ( unsigned short *position*,  
unsigned short *sourcechannel*,  
unsigned short *resultchannel*,  
unsigned short *trigger*,  
unsigned short *totzeit* )

**11.75.2.24 FeedbackSetTriggerTotzeitFactor()** `void FeedbackSetTriggerTotzeitFactor (`  
     `unsigned short trigger_factor,`  
     `unsigned short totzeit_factor )`

## 11.76 CMeaImpedanceDeviceNet Class Reference

Inheritance diagram for CMeaImpedanceDeviceNet:



### Public Member Functions

- [CMeaImpedanceDeviceNet](#) ()
- [~CMeaImpedanceDeviceNet](#) ()
- 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

#### 11.76.1 Constructor & Destructor Documentation

**11.76.1.1 CMeaImpedanceDeviceNet()** [CMeaImpedanceDeviceNet](#) ( )

**11.76.1.2 ~CMeaImpedanceDeviceNet()** [~CMeaImpedanceDeviceNet](#) ( )

#### 11.76.2 Member Function Documentation

**11.76.2.1 GetAdapterCode()** `unsigned short GetAdapterCode ( )`

**11.76.2.2 GetArraySize()** `virtual unsigned short GetArraySize ( ) [virtual]`

**11.76.2.3 GetImpedanceTestFrequency()** `virtual unsigned int GetImpedanceTestFrequency ( ) [virtual]`

**11.76.2.4 GetReady()** `virtual unsigned short GetReady ( ) [virtual]`

**11.76.2.5 GetResult()** `virtual array<unsigned short> ^ GetResult ( ) [virtual]`

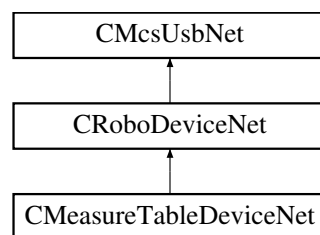
**11.76.2.6 SetImpedanceTestFrequency()** `virtual void SetImpedanceTestFrequency ( unsigned int TestFrequency_Hertz ) [virtual]`

**11.76.2.7 StartMeasurement()** `virtual void StartMeasurement ( unsigned short channel ) [virtual]`

## 11.77 CMeasureTableDeviceNet Class Reference

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

Inheritance diagram for CMeasureTableDeviceNet:



### Public Member Functions

- [CMeasureTableDeviceNet](#) (void)

### Properties

- [CMcsBus\\_SensorNet](#)<sup>^</sup> [Sensor](#) [get]

## Additional Inherited Members

### 11.77.1 Detailed Description

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

### 11.77.2 Constructor & Destructor Documentation

**11.77.2.1 CMeasureTableDeviceNet()** [CMeasureTableDeviceNet](#) (  
void )

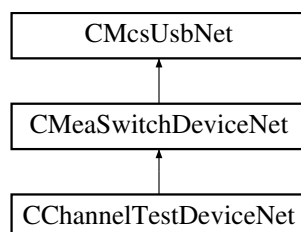
### 11.77.3 Property Documentation

**11.77.3.1 Sensor** [CMcsBus\\_SensorNet](#)^ Sensor [get]

## 11.78 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

### 11.78.1 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.

### 11.78.2 Constructor & Destructor Documentation

#### 11.78.2.1 CMeaSwitchDeviceNet() CMeaSwitchDeviceNet ( )

Constructor.

#### 11.78.2.2 ~CMeaSwitchDeviceNet() ~CMeaSwitchDeviceNet ( )

Destructor.

### 11.78.3 Member Function Documentation

#### 11.78.3.1 GetNumber() unsigned short GetNumber ( )

Gets the number of boards in the device.

The MEA-Switch are delivered with 64 or 128 channels

#### 11.78.3.2 GetPattern() array<unsigned char> ^ GetPattern ( )

Gets the pattern of the switches that are currently set in the device as char array.

#### 11.78.3.3 GetPatternBool() array<bool> ^ GetPatternBool ( )

Gets the pattern of the switches that are currently set in the device as bools.

**11.78.3.4 SetPattern()** `void SetPattern (`  
`array< unsigned char >^ pattern )`

Sets the pattern of switches from a char array.

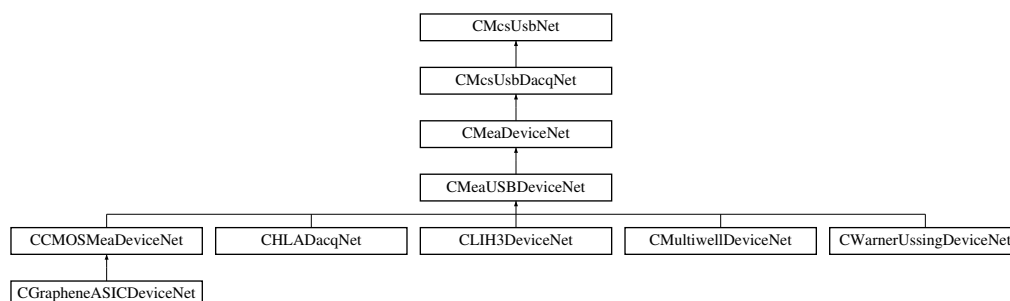
**11.78.3.5 SetPatternBool()** `void SetPatternBool (`  
`array< bool >^ pattern )`

Sets the pattern of switches from a.

## 11.79 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](#) ()
- void [SetVoltageRangeForIndex](#) (uint16\_t index, uint32\_t range)

### Additional Inherited Members

#### 11.79.1 Detailed Description

Class for data acquisition via ME and MEA USB amplifiers

#### 11.79.2 Constructor & Destructor Documentation

**11.79.2.1 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. |
|--------------|---------------------------------------|

### 11.79.2.2 CMeaUSBDeviceNet() [2/2] [CMeaUSBDeviceNet](#) ( )

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

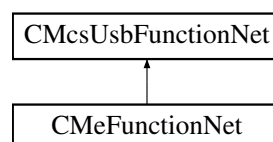
### 11.79.2.3 ~CMeaUSBDeviceNet() [~CMeaUSBDeviceNet](#) ( )

## 11.79.3 Member Function Documentation

### 11.79.3.1 SetVoltageRangeForIndex() `void SetVoltageRangeForIndex ( uint16_t index, uint32_t range )`

## 11.80 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

### 11.80.1 Detailed Description

### 11.80.2 Constructor & Destructor Documentation

**11.80.2.1 CMeFunctionNet()** [1/2] `CMeFunctionNet (`  
`CMcsUsbNet^ mcsusb,`  
`CMcsUsbFunctionPointerContainer^ meFunctionPointerContainer )`

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

**11.80.2.2 CMeFunctionNet()** [2/2] `CMeFunctionNet (`  
`CMcsUsbNet^ mcsusb )`

**11.80.2.3 ~CMeFunctionNet()** `virtual ~CMeFunctionNet (`  
`void ) [virtual]`

**11.80.2.4 "!CMeFunctionNet()** `!CMeFunctionNet (`  
`void )`

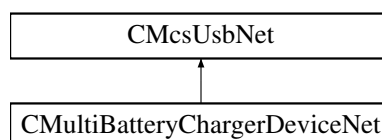
### 11.80.3 Member Function Documentation

**11.80.3.1 SetTransformer()** `void SetTransformer (`  
`unsigned int index,`  
`bool onoff )`

## 11.81 CMultiBatteryChargerDeviceNet Class Reference

[CMultiBatteryChargerDeviceNet](#) is the class to access the MBC-08 device.

Inheritance diagram for [CMultiBatteryChargerDeviceNet](#):



## Public Member Functions

- [CMultiBatteryChargerDeviceNet](#) ()  
*Initializes a new instance of the [CMultiBatteryChargerDeviceNet](#) class.*
- virtual [~CMultiBatteryChargerDeviceNet](#) ()
- [!CMultiBatteryChargerDeviceNet](#) ()
- uint32\_t [GetChargeCurrent](#) (uint32\_t NrChannel)  
*gets the charge current; unit: mA*
- uint32\_t [GetDischargeCurrent](#) (uint32\_t NrChannel)  
*gets the discharge current; unit: mA*
- void [SetDischargeCurrentSetPoint](#) (uint32\_t NrChannel, uint32\_t DischargeCurrent\_mA)  
*sets the setpoint for the discharge current; unit: mA*
- uint32\_t [GetDischargeCurrentSetPoint](#) (uint32\_t NrChannel)  
*gets the setpoint for the discharge current; unit: mA*
- void [SetFinalDischargeVoltage](#) (uint32\_t NrChannel, uint32\_t FinalDischargeVoltage\_mV)  
*sets the final discharge voltage; unit: mV*
- uint32\_t [GetFinalDischargeVoltage](#) (uint32\_t NrChannel)  
*gets the final discharge voltage; unit: mV*
- uint32\_t [GetDischargeCapacity](#) (uint32\_t NrChannel)  
*gets the discharge capacity; unit:  $\mu$ Ah*
- uint32\_t [GetChargeCapacity](#) (uint32\_t NrChannel)  
*gets the charge capacity; unit:  $\mu$ 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

### 11.81.1 Detailed Description

[CMultiBatteryChargerDeviceNet](#) is the class to access the MBC-08 device.

### 11.81.2 Constructor & Destructor Documentation

#### 11.81.2.1 **CMultiBatteryChargerDeviceNet()** [CMultiBatteryChargerDeviceNet](#) ( )

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

#### 11.81.2.2 **~CMultiBatteryChargerDeviceNet()** [virtual ~CMultiBatteryChargerDeviceNet](#) ( ) [virtual]

#### 11.81.2.3 **!CMultiBatteryChargerDeviceNet()** [!CMultiBatteryChargerDeviceNet](#) ( )

### 11.81.3 Member Function Documentation

#### 11.81.3.1 **CapacityTest()** [void CapacityTest](#) ( [uint32\\_t NrChannel](#) )

start capacity test on channel

##### Parameters

|                           |                    |
|---------------------------|--------------------|
| <a href="#">NrChannel</a> | the channel number |
|---------------------------|--------------------|

#### 11.81.3.2 **ChannelReset()** [void ChannelReset](#) ( [uint32\\_t NrChannel](#) )

cancel charging and capacity test functions; check if battery is connected

##### Parameters

|                           |                    |
|---------------------------|--------------------|
| <a href="#">NrChannel</a> | the channel number |
|---------------------------|--------------------|

**11.81.3.3 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

**11.81.3.4 GetChannels()** `uint32_t GetChannels ( )`

gets number of channels

**Returns**

number of channels

**11.81.3.5 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

**11.81.3.6 GetChargeCapacity()** `uint32_t GetChargeCapacity (`  
`uint32_t NrChannel )`

gets the charge capacity; unit:  $\mu$ Ah

**Parameters**

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

**Returns**

the capacity in uAh

**11.81.3.7 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

**11.81.3.8 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

**11.81.3.9 GetChargingPCoefficient()** `uint32_t GetChargingPCoefficient ( )`

gets the p-coefficient for charging in mA/V / nominal charging current

**Returns**

the p-coefficient



**11.81.3.10 GetDischargeCapacity()** `uint32_t GetDischargeCapacity (uint32_t NrChannel )`

gets the discharge capacity; unit:  $\mu$ Ah

**Parameters**

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

**Returns**

the capacity in uAh

**11.81.3.11 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

**11.81.3.12 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

**11.81.3.13 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

**11.81.3.14 GetRatedCapacity()** `MbcRatedCapacityEnumNet GetRatedCapacity (`  
`uint32_t NrChannel )`

gets the rated capacity

**Parameters**

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

**Returns**

the capacity

**11.81.3.15 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  |

**11.81.3.16 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 |
|---------------------|-------------------|

**11.81.3.17 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 |

**11.81.3.18 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 |

**11.81.3.19 SetRatedCapacity()** void SetRatedCapacity (   
    uint32\_t NrChannel,   
    MbcRatedCapacityEnumNet NewRatedCapacity )

sets the rated capacity

Parameters

|                         |                    |
|-------------------------|--------------------|
| <i>NrChannel</i>        | the channel number |
| <i>NewRatedCapacity</i> | the capacity       |

**11.81.3.20 SetRatedCapacityVolatile()** void SetRatedCapacityVolatile (   
    uint32\_t NrChannel,   
    MbcRatedCapacityEnumNet NewRatedCapacity )

sets the rated capacity (i.e. charge current) without storing it persistently

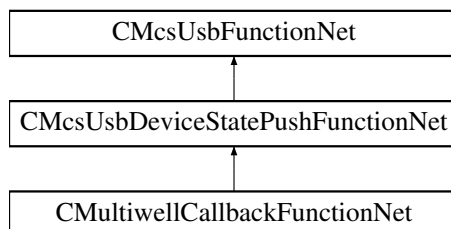
Parameters

|                         |                    |
|-------------------------|--------------------|
| <i>NrChannel</i>        | the channel number |
| <i>NewRatedCapacity</i> | the capacity       |

## 11.82 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

#### 11.82.1 Detailed Description

[CMultiwellCallbackFunctionNet](#) is the class to access the Multiwell-Mini-Stimulator

#### 11.82.2 Constructor & Destructor Documentation

**11.82.2.1 [CMultiwellCallbackFunctionNet](#)() [1/2]** [CMultiwellCallbackFunctionNet](#) (  
[CMcsUsbNet](#)^ mcsusb,  
[CMcsUsbFunctionPointerContainer](#)^ pMultiwellCallbackFunctionPointerContainer )

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

**11.82.2.2 CMultiwellCallbackFunctionNet()** [2/2] `CMultiwellCallbackFunctionNet ( CMcsUsbNet^ mcsusb )`

**11.82.2.3 ~CMultiwellCallbackFunctionNet()** `virtual ~CMultiwellCallbackFunctionNet ( )` [virtual]

**11.82.2.4 "!CMultiwellCallbackFunctionNet()** `!CMultiwellCallbackFunctionNet ( )`

### 11.82.3 Member Function Documentation

**11.82.3.1 GetPlateClampStateByHeadstage()** `PlateClampEnumNet GetPlateClampStateByHeadstage ( uint32_t Headstage )`

Gets the state of the plate

#### Parameters

|                  |                      |
|------------------|----------------------|
| <i>Headstage</i> | The headstage number |
|------------------|----------------------|

#### Returns

The plate state

**11.82.3.2 OnGetPlateClampStateByHeadstage()** `delegate void OnGetPlateClampStateByHeadstage ( uint32_t Headstage, PlateClampEnumNet plateState )`

### 11.82.4 Event Documentation

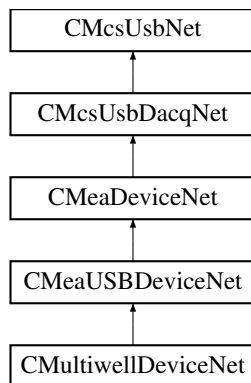
**11.82.4.1 GetPlateClampStateByHeadstageEvent** `OnGetPlateClampStateByHeadstage^ GetPlateClampStateByHeadstageEvent [add], [remove], [raise]`

Event fires when the plate state for the headstage number has changed

## 11.83 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.*
- void [SetTouchPadEnable](#) (uint32\_t Headstage, bool Enable)  
*Enables or disables manual opening/closing of plate clamp via touch pad.*
- bool [GetTouchPadEnable](#) (uint32\_t Headstage)  
*Manual opening/closing of plate clamp via touch pad can be disabled.*
- void [SetVolatileClampOffset](#) (uint32\_t Headstage, int32\_t CoverLipThickness\_um)  
*The distance travelled to clamp the plate can be reduced e.g. to compensate for a cover.*
- int32\_t [GetVolatileClampOffset](#) (uint32\_t Headstage)  
*The distance travelled to clamp the plate can be reduced e.g. to compensate for a cover.*

## Additional Inherited Members

### 11.83.1 Detailed Description

[CMultiwellDeviceNet](#) is the class to access the Multiwell device.

### 11.83.2 Constructor & Destructor Documentation

#### 11.83.2.1 CMultiwellDeviceNet() [CMultiwellDeviceNet](#) ( )

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

#### 11.83.2.2 ~CMultiwellDeviceNet() [virtual](#) [~CMultiwellDeviceNet](#) ( ) [\[virtual\]](#)

#### 11.83.2.3 "!CMultiwellDeviceNet() [!CMultiwellDeviceNet](#) ( )

### 11.83.3 Member Function Documentation

#### 11.83.3.1 ClosePlateClamp() `void ClosePlateClamp ( )`

Closes the plate clamp.

#### 11.83.3.2 GetPlateClampLockState() `uint32_t GetPlateClampLockState ( )`

Gets the state of the plate clamp lock.

##### Returns

the state of the plate lock (unlocked/locked)

#### 11.83.3.3 GetPlateClampState() [1/2] `PlateClampEnumNet GetPlateClampState ( )`

Gets the state of the Multiwell plate clamp.

##### Returns

the state of the plate clamp (open/closed)

#### 11.83.3.4 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

#### 11.83.3.5 GetPlateMux() [1/2] `uint32_t GetPlateMux ( )`

Gets the selected quarter of the electrodes on a high density Multiwell plate.



**Returns**

the selected quarter

**11.83.3.6 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

**11.83.3.7 GetPlateType()** [1/2] `MultiwellPlateTypeEnumNet GetPlateType ( )`

Gets the plate type.

**Returns**

the plate type

**11.83.3.8 GetPlateType()** [2/2] `MultiwellPlateTypeEnumNet GetPlateType (`  
`uint32_t Headstage )`

Gets the plate type.

**Parameters**

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

**Returns**

the plate type

**11.83.3.9 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

**11.83.3.10 GetTouchPadEnable()** `bool GetTouchPadEnable (`  
`uint32_t Headstage )`

Manual opening/closing of plate clamp via touch pad can be disabled.

**Parameters**

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

**Returns**

"true" when plate clamp can be driven manually, otherwise "false".

**11.83.3.11 GetVolatileClampOffset()** `int32_t GetVolatileClampOffset (`  
`uint32_t Headstage )`

The distance travelled to clamp the plate can be reduced e.g. to compensate for a cover.

**Parameters**

|                  |                               |
|------------------|-------------------------------|
| <i>Headstage</i> | The headstage to be affected. |
|------------------|-------------------------------|

**Returns**

Clamp distance reduction in um.

**11.83.3.12 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".

**11.83.3.13 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".

**11.83.3.14 LockPlateClamp()** `void LockPlateClamp ( )`

Locks the plate clamp.

**11.83.3.15 OpenPlateClamp()** `void OpenPlateClamp ( )`

Opens the plate clamp.

**11.83.3.16 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    |

**11.83.3.17 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 |
|---------------------|----------------------|

**11.83.3.18 SetPlateType()** [1/2] `void SetPlateType (`  
`MultiwellPlateTypeEnumNet plateType )`

Sets the plate type.

Parameters

|                  |                |
|------------------|----------------|
| <i>plateType</i> | the plate type |
|------------------|----------------|

**11.83.3.19 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          |

**11.83.3.20 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 |

**11.83.3.21 SetTouchPadEnable()** `void SetTouchPadEnable (`  
`uint32_t Headstage,`  
`bool Enable )`

Enables or disables manual opening/closing of plate clamp via touch pad.

Parameters

|                  |                                                                      |
|------------------|----------------------------------------------------------------------|
| <i>Headstage</i> | The headstage to be affected.                                        |
| <i>Enable</i>    | "true" when plate clamp shall be driven manually, otherwise "false". |

**11.83.3.22 SetVolatileClampOffset()** `void SetVolatileClampOffset (`  
`uint32_t Headstage,`  
`int32_t CoverLipThickness_um )`

The distance travelled to clamp the plate can be reduced e.g. to compensate for a cover.

#### Parameters

|                             |                                                                           |
|-----------------------------|---------------------------------------------------------------------------|
| <i>Headstage</i>            | The headstage to be affected.                                             |
| <i>CoverLipThickness_um</i> | Clamp distance reduction in um. Range: 200um..400um; Typical value: 300um |

**11.83.3.23 StopPlateClamp()** `void StopPlateClamp ( )`

Stops the plate clamp movement.

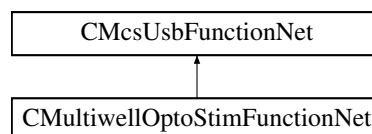
**11.83.3.24 UnlockPlateClamp()** `void UnlockPlateClamp ( )`

Unlocks the plate clamp.

## 11.84 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 ^ [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^ ColorString)

## Additional Inherited Members

### 11.84.1 Detailed Description

[CMultiwellOptoStimFunctionNet](#) is the class to access the optical properties of the Multiwell Optostim device

### 11.84.2 Constructor & Destructor Documentation

**11.84.2.1 [CMultiwellOptoStimFunctionNet\(\)](#) [1/2]** [CMultiwellOptoStimFunctionNet](#) (   
[CMcsUsbNet](#)^ *mcsusb*,   
[CMcsUsbFunctionPointerContainer](#)^ *pMultiwellOptoStimFunctionPointerContainer* )

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

**11.84.2.2 [CMultiwellOptoStimFunctionNet\(\)](#) [2/2]** [CMultiwellOptoStimFunctionNet](#) (   
[CMcsUsbNet](#)^ *mcsusb* )

**11.84.2.3 [~CMultiwellOptoStimFunctionNet\(\)](#)** virtual [~CMultiwellOptoStimFunctionNet](#) ( ) [virtual]

**11.84.2.4 ["!CMultiwellOptoStimFunctionNet\(\)](#)** [!CMultiwellOptoStimFunctionNet](#) ( )

### 11.84.3 Member Function Documentation

**11.84.3.1 [GetAbsMaxCurrentInMicroAmp\(\)](#)** uint32\_t [GetAbsMaxCurrentInMicroAmp](#) (   
uint16\_t *channel* )

## Parameters

|                |                             |
|----------------|-----------------------------|
| <i>channel</i> | the (analog) channel number |
|----------------|-----------------------------|

## Returns

absolute max. current; unit: uA

**11.84.3.2 GetColorRgb()** `uint32_t GetColorRgb (`  
`uint16_t channel )`

## Parameters

|                |                             |
|----------------|-----------------------------|
| <i>channel</i> | the (analog) channel number |
|----------------|-----------------------------|

## Returns

RGB-value of LED color

**11.84.3.3 GetColorStr()** `String ^ GetColorStr (`  
`uint16_t channel )`

## Parameters

|                |                             |
|----------------|-----------------------------|
| <i>channel</i> | the (analog) channel number |
|----------------|-----------------------------|

## Returns

LED color as string

**11.84.3.4 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

**11.84.3.5 GetMaxDutyCycleHighCurrent()** `uint32_t GetMaxDutyCycleHighCurrent (`  
`uint16_t channel )`

**Parameters**

|                |                             |
|----------------|-----------------------------|
| <i>channel</i> | the (analog) channel number |
|----------------|-----------------------------|

**Returns**

max. duty cycle at max. current; unit: 100\*%

**11.84.3.6 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

**11.84.3.7 GetWaveLengthInNanometer()** `uint32_t GetWaveLengthInNanometer (`  
`uint16_t channel )`

**Parameters**

|                |                             |
|----------------|-----------------------------|
| <i>channel</i> | the (analog) channel number |
|----------------|-----------------------------|

**Returns**

wavelength of this channel's LEDs; unit: nm

**11.84.3.8 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 |



**11.84.3.9 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      |

**11.84.3.10 SetColorStr()** `void SetColorStr (`  
     `uint16_t channel,`  
     `String^ ColorString )`

#### Parameters

|                    |                             |
|--------------------|-----------------------------|
| <i>channel</i>     | the (analog) channel number |
| <i>ColorString</i> | LED color as string         |

**11.84.3.11 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 |

**11.84.3.12 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*% |

**11.84.3.13 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 |

**11.84.3.14 SetWaveLengthInNanometer()** `void SetWaveLengthInNanometer (`  

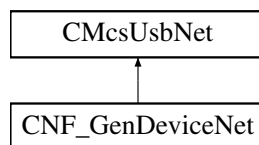
```
uint16_t channel,
uint32_t WaveLength_nm)
```

**Parameters**

|                      |                                             |
|----------------------|---------------------------------------------|
| <i>channel</i>       | the (analog) channel number                 |
| <i>WaveLength_nm</i> | wavelength of this channel's LEDs; unit: nm |

**11.85 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****11.85.1 Constructor & Destructor Documentation**

**11.85.1.1 CNF\_GenDeviceNet()** `CNF_GenDeviceNet (`  

```
void)
```

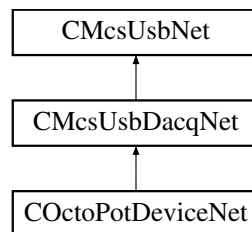
**11.85.1.2** `~CNF_GenDeviceNet()` `~CNF_GenDeviceNet` (  
`void` )

## 11.85.2 Member Function Documentation

**11.85.2.1** `Set_Values()` `void Set_Values` (  
`unsigned int frequency,`  
`unsigned int amplitude` )

## 11.86 COctoPotDeviceNet Class Reference

Inheritance diagram for COctoPotDeviceNet:



### Public Member Functions

- [COctoPotDeviceNet](#) (`void`)
- [COctoPotDeviceNet](#) ([OnChannelData](#)<sup>^</sup> `channelData`, [OnError](#)<sup>^</sup> `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

### 11.86.1 Constructor & Destructor Documentation

**11.86.1.1 COctoPotDeviceNet()** [1/2] `COctoPotDeviceNet ( void )`

**11.86.1.2 COctoPotDeviceNet()** [2/2] `COctoPotDeviceNet ( OnChannelData^ channelData, OnError^ error )`

### 11.86.2 Member Function Documentation

**11.86.2.1 BurnAdcOffset()** `uint32_t BurnAdcOffset ( )`

**11.86.2.2 BurnDacOffset()** `uint32_t BurnDacOffset ( )`

**11.86.2.3 EnableChecksum()** `uint32_t EnableChecksum ( bool enable )`

**11.86.2.4 EnableDigitalIn()** `uint32_t EnableDigitalIn ( bool enable )`

**11.86.2.5 EnableTimestamp()** `uint32_t EnableTimestamp ( bool enable )`

**11.86.2.6 GetAdcOffset()** `uint32_t GetAdcOffset ( unsigned int channel, [System::Runtime::InteropServices::Out] int ^ offset )`

**11.86.2.7 GetDacOffset()** uint32\_t GetDacOffset (   
 unsigned int *channel*,   
 [System::Runtime::InteropServices::Out] int ^ *offset* )

**11.86.2.8 PatternListStart()** uint32\_t PatternListStart (   
 int *channelmap* )

**11.86.2.9 RampStart()** uint32\_t RampStart (   
 int *channelmap* )

**11.86.2.10 ResetAdcOffset()** uint32\_t ResetAdcOffset (   
 unsigned int *channel* )

**11.86.2.11 ResetDacOffset()** uint32\_t ResetDacOffset (   
 unsigned int *channel* )

**11.86.2.12 SetAdcOffset()** uint32\_t SetAdcOffset (   
 unsigned int *channel*,   
 int *offset* )

**11.86.2.13 SetAmplificationSwitch()** uint32\_t SetAmplificationSwitch (   
 unsigned int *channel*,   
 unsigned int *state* )

**11.86.2.14 SetBathclamp()** uint32\_t SetBathclamp (   
 unsigned int *block*,   
 bool *enable* )

**11.86.2.15 SetChannelSwitch()** uint32\_t SetChannelSwitch (   
 unsigned int *channel*,   
 unsigned int *state* )

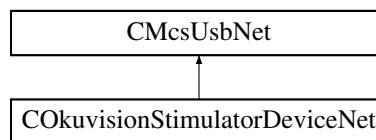
- 11.86.2.16 SetDacAutoControl()** uint32\_t SetDacAutoControl (   
    unsigned int *channel* )
- 11.86.2.17 SetDacOffset()** uint32\_t SetDacOffset (   
    unsigned int *channel*,   
    int *offset* )
- 11.86.2.18 SetDacValue()** uint32\_t SetDacValue (   
    int *channel*,   
    int *value* )
- 11.86.2.19 SetNumberOfChannels()** uint32\_t SetNumberOfChannels (   
    unsigned int *NumberOfChannels* )
- 11.86.2.20 SetOutputRate()** uint32\_t SetOutputRate (   
    uint32\_t *rate* )
- 11.86.2.21 SetPatternListEntry()** uint32\_t SetPatternListEntry (   
    unsigned int *channel*,   
    unsigned int *position*,   
    unsigned int *duration*,   
    int *value* )
- 11.86.2.22 SetPidParameter()** uint32\_t SetPidParameter (   
    unsigned int *channel*,   
    int *const\_p*,   
    int *const\_i*,   
    int *shift\_p*,   
    int *shift\_i* )
- 11.86.2.23 SetRampParameter()** uint32\_t SetRampParameter (   
    unsigned int *channel*,   
    int *start*,   
    int *min*,   
    int *max*,   
    int *slope*,   
    int *slope2*,   
    int *pause*,   
    unsigned int *samples* )

**11.86.2.24 SetSineParameter()** uint32\_t SetSineParameter (   
     unsigned int *channel*,   
     int *amplitude* )

**11.86.2.25 SineStart()** uint32\_t SineStart (   
     int *channelmap* )

## 11.87 COkuvisionStimulatorDeviceNet Class Reference

Inheritance diagram for COkuvisionStimulatorDeviceNet:



### Public Member Functions

- [COkuvisionStimulatorDeviceNet](#) (void)
- [~COkuvisionStimulatorDeviceNet](#) (void)
- void [SetPulseform](#) (int channel, int current, int pulsedwidth, int periode, int duration)
- void [GetPulseform](#) (int channel, [System::Runtime::InteropServices::Out] int% current, [System::Runtime::InteropServices::Out] int% pulsedwidth, [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

### 11.87.1 Constructor & Destructor Documentation

**11.87.1.1 COkuvisionStimulatorDeviceNet()** `COkuvisionStimulatorDeviceNet ( void )`

**11.87.1.2 ~COkuvisionStimulatorDeviceNet()** `~COkuvisionStimulatorDeviceNet ( void )`

### 11.87.2 Member Function Documentation

**11.87.2.1 GetCheckVoltage()** `int GetCheckVoltage ( int channel )`

**11.87.2.2 GetCurrentFactor()** `int GetCurrentFactor ( int channel )`

**11.87.2.3 GetDACOffset()** `int GetDACOffset ( int channel, int part )`

**11.87.2.4 GetMaxPower()** `int GetMaxPower ( int channel )`

**11.87.2.5 GetMaxVoltage()** `int GetMaxVoltage ( int channel )`



**11.87.2.6 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* )

**11.87.2.7 GetRTC()** [1/2] DateTime GetRTC ( )

**11.87.2.8 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* )

**11.87.2.9 GetStimulatorStatus()** void GetStimulatorStatus (   
[System::Runtime::InteropServices::Out] int% *startstop*,  
[System::Runtime::InteropServices::Out] int% *last\_error*,  
[System::Runtime::InteropServices::Out] int% *battery\_status* )

**11.87.2.10 GetVoltage()** int GetVoltage (   
int *channel* )

**11.87.2.11 SetCheckVoltage()** void SetCheckVoltage (   
int *channel*,  
int *voltage* )

**11.87.2.12 SetCurrentFactor()** void SetCurrentFactor (   
int *channel*,  
int *factor* )

**11.87.2.13 SetDACOffset()** `void SetDACOffset (`  
    `int channel,`  
    `int part,`  
    `int offset )`

**11.87.2.14 SetMaxPower()** `void SetMaxPower (`  
    `int channel,`  
    `int power )`

**11.87.2.15 SetMaxVoltage()** `void SetMaxVoltage (`  
    `int channel,`  
    `int voltage )`

**11.87.2.16 SetPulseform()** `void SetPulseform (`  
    `int channel,`  
    `int current,`  
    `int pulsewidth,`  
    `int periode,`  
    `int duration )`

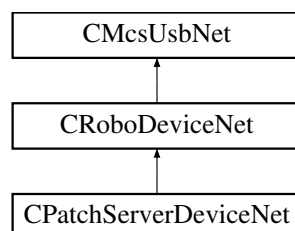
**11.87.2.17 SetRTC() [1/2]** `void SetRTC (`  
    `DateTime timestamp )`

**11.87.2.18 SetRTC() [2/2]** `void SetRTC (`  
    `uint8_t year,`  
    `uint8_t month,`  
    `uint8_t day,`  
    `uint8_t hour,`  
    `uint8_t minute,`  
    `uint8_t second )`

## 11.88 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^ Sensor](#) [get]

**Additional Inherited Members****11.88.1 Detailed Description**

[CPatchServerDeviceNet](#) is the class to control the MCS PatchServer device

**11.88.2 Constructor & Destructor Documentation**

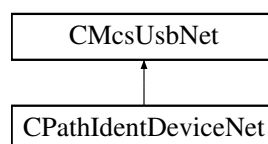
**11.88.2.1** [CPatchServerDeviceNet\(\)](#) [CPatchServerDeviceNet](#) (   
 void )

**11.88.3 Property Documentation**

**11.88.3.1** **Sensor** [CMcsBus\\_SensorNet^ Sensor](#) [get]

**11.89 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

### 11.89.1 Constructor & Destructor Documentation

**11.89.1.1 CPathIdentDeviceNet()** `CPathIdentDeviceNet (`  
`void )`

**11.89.1.2 ~CPathIdentDeviceNet()** `~CPathIdentDeviceNet (`  
`void )`

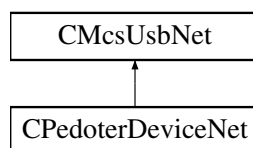
### 11.89.2 Member Function Documentation

**11.89.2.1 Measure()** `void Measure (`  
`[System::Runtime::InteropServices::Out] unsigned int% phase,`  
`[System::Runtime::InteropServices::Out] unsigned int% amplitude )`

**11.89.2.2 Set\_Values()** `void Set_Values (`  
`unsigned int frequency,`  
`unsigned int amplitude )`

## 11.90 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

### 11.90.1 Detailed Description

### 11.90.2 Constructor & Destructor Documentation

#### 11.90.2.1 CPedoterDeviceNet() `CPedoterDeviceNet ( )`

Initializes a new instance of the `CPedoterDeviceNet` class.

#### 11.90.2.2 `~CPedoterDeviceNet()` `virtual ~CPedoterDeviceNet ( ) [virtual]`

#### 11.90.2.3 `!CPedoterDeviceNet()` `!CPedoterDeviceNet ( )`

### 11.90.3 Member Function Documentation

#### 11.90.3.1 `GetCommand()` `uint32_t GetCommand ( uint16_t Argument )`

Get value from the pedoter device

##### Parameters

|                 |          |
|-----------------|----------|
| <i>Argument</i> | argument |
|-----------------|----------|

##### Returns

value

#### 11.90.3.2 `SetCommand()` `void SetCommand ( uint16_t Argument, uint32_t pData )`

Set value on the pedoter device

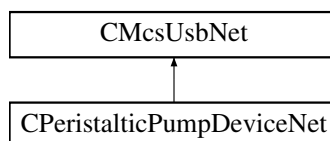
## Parameters

|                 |          |
|-----------------|----------|
| <i>Argument</i> | argument |
| <i>pData</i>    | value    |

## 11.91 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

#### 11.91.1 Detailed Description

[CPeristalticPumpDeviceNet](#) is the class to control a Persistaltic Pump.

#### 11.91.2 Constructor & Destructor Documentation

##### 11.91.2.1 CPeristalticPumpDeviceNet() [CPeristalticPumpDeviceNet](#) (void )

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

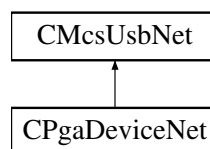
**11.91.2.2** `~CPeristalticPumpDeviceNet()` `~CPeristalticPumpDeviceNet` (  
     `void` )

### 11.91.3 Property Documentation

**11.91.3.1** `McsBus_MotorControl` `CMcsBus_MotorControlNet`^ `McsBus_MotorControl` [get]

## 11.92 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

#### 11.92.1 Constructor & Destructor Documentation

**11.92.1.1** `CPgaDeviceNet()` `CPgaDeviceNet` ( )

**11.92.1.2** `~CPgaDeviceNet()` `~CPgaDeviceNet ( )`

## 11.92.2 Member Function Documentation

**11.92.2.1** `ApplyGains()` `uint32_t ApplyGains ( )`

**11.92.2.2** `DefineAmplification()` `uint32_t DefineAmplification (`  
    `int index,`  
    `int amplification,`  
    `int poti1,`  
    `int poti2 )`

**11.92.2.3** `DefineFrequencyRange()` `uint32_t DefineFrequencyRange (`  
    `int index,`  
    `int low,`  
    `int high,`  
    `int channels,`  
    `int gain )`

**11.92.2.4** `DefineNumAmplifications()` `uint32_t DefineNumAmplifications (`  
    `int number )`

**11.92.2.5** `DefineNumFrequencyRanges()` `uint32_t DefineNumFrequencyRanges (`  
    `int rnum )`

**11.92.2.6** `GetAmplification()` `uint32_t GetAmplification (`  
    `int index,`  
    `[System::Runtime::InteropServices::Out] int% amplification,`  
    `[System::Runtime::InteropServices::Out] int% poti1,`  
    `[System::Runtime::InteropServices::Out] int% poti2 )`



**11.92.2.7 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* )

**11.92.2.8 GetGain()** uint32\_t GetGain (   
     int *channel*,   
     [System::Runtime::InteropServices::Out] int% *Gain*,   
     [System::Runtime::InteropServices::Out] int% *potil*,   
     [System::Runtime::InteropServices::Out] int% *poti2* )

**11.92.2.9 GetNumAmplifications()** uint32\_t GetNumAmplifications (   
     [System::Runtime::InteropServices::Out] int% *number* )

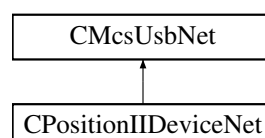
**11.92.2.10 GetNumFrequencyRanges()** uint32\_t GetNumFrequencyRanges (   
     [System::Runtime::InteropServices::Out] int% *numRanges* )

**11.92.2.11 SetGain()** uint32\_t SetGain (   
     int *channel*,   
     int *Gain*,   
     int *potil*,   
     int *poti2* )

## 11.93 CPositionIIDeviceNet Class Reference

[CPositionIIDeviceNet](#) is the class to control PositionII devices

Inheritance diagram for CPositionIIDeviceNet:



## Public Member Functions

- [CPositionIIDeviceNet](#) ()  
*Initializes a new instance of the [CPositionIIDeviceNet](#) class.*
- virtual [~CPositionIIDeviceNet](#) ()
- [!CPositionIIDeviceNet](#) ()
- uint32\_t [GetCoilCommunication](#) (uint16\_t coil)  
*get if the communication to the coil is working*
- uint32\_t [GetOnOff](#) (uint16\_t coil)  
*get if the coil is switched on/off*
- void [SwitchOnOff](#) (uint16\_t coil, uint32\_t on)  
*switched the 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

### 11.93.1 Detailed Description

[CPositionIIDeviceNet](#) is the class to control PositionII devices

### 11.93.2 Constructor & Destructor Documentation

#### 11.93.2.1 CPositionIIDeviceNet() [CPositionIIDeviceNet](#) ( )

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

#### 11.93.2.2 ~CPositionIIDeviceNet() [virtual](#) [~CPositionIIDeviceNet](#) ( ) [virtual]

#### 11.93.2.3 ~!CPositionIIDeviceNet() [!CPositionIIDeviceNet](#) ( )

### 11.93.3 Member Function Documentation

#### 11.93.3.1 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

**11.93.3.2 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 |

**11.93.3.3 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         |

**11.93.3.4 GetImplantCurrentSetpoint()** uint32\_t GetImplantCurrentSetpoint (

```

 uint16_t coil)

```

sets the implant current setpoint

## Parameters

|             |          |
|-------------|----------|
| <i>coil</i> | the coil |
|-------------|----------|

## Returns

the current

**11.93.3.5 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

**11.93.3.6 GetImplantState()** `uint32_t GetImplantState (`  
`uint16_t coil )`

gets the implantat state

## Parameters

|             |          |
|-------------|----------|
| <i>coil</i> | the coil |
|-------------|----------|

## Returns

the state

**11.93.3.7 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

**11.93.3.8 GetPowerStrength()** `uint32_t GetPowerStrength (`  
`uint16_t coil )`

sets the power for the trigger pulses

**Parameters**

|             |          |
|-------------|----------|
| <i>coil</i> | the coil |
|-------------|----------|

**Returns**

the power

**11.93.3.9 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 |

**11.93.3.10 GetStateDebugData()** `uint32_t GetStateDebugData (`  
`uint16_t coil )`

get the debug queue state

**Parameters**

|             |          |
|-------------|----------|
| <i>coil</i> | the coil |
|-------------|----------|

**Returns**

the state

**11.93.3.11 GetStateEventData()** `uint32_t GetStateEventData ( )`

get the event queue state

**Returns**

the state

**11.93.3.12 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 |

**11.93.3.13 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 |

**11.93.3.14 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 |

**11.93.3.15 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 |

**11.93.3.16 SetStateEventData()** `void SetStateEventData (`  
    `uint32_t state )`

clears/starts/stops the event queue for a certain coil

## Parameters

|              |                  |
|--------------|------------------|
| <i>state</i> | clear/start/stop |
|--------------|------------------|

**11.93.3.17 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 |

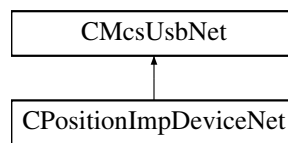
### 11.93.4 Property Documentation

#### 11.93.4.1 RFFunction `CRFFFunctionNet^ RFFunction [get]`

## 11.94 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

#### 11.94.1 Detailed Description

[CPositionImpDeviceNet](#) is the class to access the Position/Imp devices

## 11.94.2 Constructor & Destructor Documentation

### 11.94.2.1 CPositionImpDeviceNet() `CPositionImpDeviceNet ( )`

Initializes a new instance of the `CPositionImpDeviceNet` class.

### 11.94.2.2 ~CPositionImpDeviceNet() `virtual ~CPositionImpDeviceNet ( ) [virtual]`

### 11.94.2.3 "!CPositionImpDeviceNet() `!CPositionImpDeviceNet ( )`

## 11.94.3 Member Function Documentation

### 11.94.3.1 ConnectedImp() `uint32_t ConnectedImp ( )`

The ID of the connected Imp device

#### Returns

The ID

### 11.94.3.2 ConnectImp() `void ConnectImp ( uint32_t id )`

Connect to a Imp device with a certain ID

#### Parameters

|           |        |
|-----------|--------|
| <i>id</i> | The ID |
|-----------|--------|

### 11.94.3.3 GetDeviceList() `uint32_t GetDeviceList ( int32_t index )`

Gets the device list

**Parameters**

|              |           |
|--------------|-----------|
| <i>index</i> | the index |
|--------------|-----------|

**Returns**

the ID

**11.94.3.4 GetImpId()** `uint32_t GetImpId ( )`

Gets the ID of the impedance measure device

**Returns**

the ID

**11.94.3.5 GetRFFrequency()** `int32_t GetRFFrequency ( )`

Gets currently used RF frequency

**Returns**

The frequency

**11.94.3.6 SetDeviceList()** `void SetDeviceList (`  
    `int32_t index,`  
    `uint32_t id )`

Sets the device list

**Parameters**

|              |           |
|--------------|-----------|
| <i>index</i> | the index |
| <i>id</i>    | the ID    |

**11.94.3.7 SetImpId()** `void SetImpId (`  
    `uint32_t id )`

Sets the ID of the impedance measure device

## Parameters

|           |        |
|-----------|--------|
| <i>id</i> | the ID |
|-----------|--------|

**11.94.3.8 SetRFFrequency()** `void SetRFFrequency (`  
`int32_t frequency )`

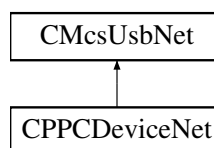
Sets the current RF frequency

## Parameters

|                  |               |
|------------------|---------------|
| <i>frequency</i> | The frequency |
|------------------|---------------|

## 11.95 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

#### 11.95.1 Constructor & Destructor Documentation

**11.95.1.1 CPPCDeviceNet()** `CPPCDeviceNet (`  
`void )`

## 11.95.2 Property Documentation

**11.95.2.1 McsBus** [CMcsBusNet](#)<sup>^</sup> McsBus [get]

**11.95.2.2 McsBus\_MotorControl** [CMcsBus\\_MotorControlNet](#)<sup>^</sup> McsBus\_MotorControl [get]

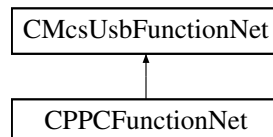
**11.95.2.3 McsBus\_Sensor** [CMcsBus\\_SensorNet](#)<sup>^</sup> McsBus\_Sensor [get]

**11.95.2.4 PPCFunction** [CPPPCFunctionNet](#)<sup>^</sup> PPCFunction [get]

## 11.96 CPPCFunctionNet Class Reference

[CPPCFunctionNet](#) is the class to access the PPC (high precision Patch Peristaltic patch Pump

Inheritance diagram for CPPCFunctionNet:



### Public Member Functions

- [CPPCFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> pPPCFunctionPointer↔ Container)  
*Initializes a new instance of the [CPPCFunctionNet](#) class.*
- [CPPCFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- virtual [~CPPCFunctionNet](#) ()
- [!CPPCFunctionNet](#) ()
- int [GetPumpSpeedUnit](#) (uint16\_t channel)  
*Reads the Pump Speed Unit*
- void [SetPumpSpeedUnit](#) (uint16\_t channel, int SpeedUnit)  
*Writes the Pump Speed Unit*
- [PP\\_Pump\\_Mode\\_Type\\_EnumNet](#) [GetPumpModeType](#) (uint16\_t channel)  
*Reads the Pump Mode Type.*
- void [SetPumpModeType](#) (uint16\_t channel, [PP\\_Pump\\_Mode\\_Type\\_EnumNet](#) PumpMode)  
*Writes the config string from the device.*

- void [GetAnalogVoltageRange](#) (uint16\_t channel, [System::Runtime::InteropServices::Out]uint16\_t% min\_voltage, [System::Runtime::InteropServices::Out]uint16\_t% max\_voltage)  
*Reads the Analog Input Voltage Range*
- void [SetAnalogVoltageRange](#) (uint16\_t channel, uint16\_t min\_voltage, uint16\_t max\_voltage)  
*Writes the Analog Input Voltage Range*
- void [GetPressureRange](#) (uint16\_t channel, [System::Runtime::InteropServices::Out]int32\_t% lower\_pressure, [System::Runtime::InteropServices::Out]int32\_t% upper\_pressure)  
*Get the pressure range that is used between the analog voltage or the digital states*
- void [SetPressureRange](#) (uint16\_t channel, int32\_t lower\_pressure, int32\_t upper\_pressure)  
*Get the pressure range that is used between the analog voltage or the digital states*
- uint16\_t [GetSupplyVoltage](#) ()  
*Reads the current supply voltage in mV*
- uint16\_t [GetAnalogVoltage](#) (uint16\_t channel)  
*Reads the current analog voltage*
- uint16\_t [GetDigitalIn](#) (uint16\_t channel)  
*Reads the digital input state*
- int [GetValveActive](#) (uint16\_t valve)  
*Gets the valve active/inactive state*
- void [SetValveActive](#) (uint16\_t valve, int valveActive)  
*Sets the valve active/inactive state*
- void [SetPressureOffset](#) ()  
*Sets the pressure offset*
- void [LoadPressure](#) (int32\_t pressure, uint32\_t options)  
*Loads the reservoir with a pressure*
- void [IsBusy](#) ([System::Runtime::InteropServices::Out]int16\_t% task, [System::Runtime::InteropServices::Out]int16\_t% wait)  
*Is the PPC busy with a task*
- void [FirePressurePulse](#) (int32\_t duration, int32\_t nextpressure)  
*Fire a pressure pulse from the reservoir*
- int32\_t [MeasureReservoir](#) ()  
*Measures the reservoir pressure*

## Additional Inherited Members

### 11.96.1 Detailed Description

[CPPCFunctionNet](#) is the class to access the PPC (high precision Patch Peristaltic patch Pump

### 11.96.2 Constructor & Destructor Documentation

**11.96.2.1 CPPCFunctionNet()** [1/2] [CPPCFunctionNet](#) (  
     CMcsUsbNet^ mcsusb,  
     CMcsUsbFunctionPointerContainer^ pPPCFunctionPointerContainer )

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

**11.96.2.2 CPPCFunctionNet()** [2/2] `CPPCFunctionNet (`  
`CMcsUsbNet^ mcsusb )`

**11.96.2.3 ~CPPCFunctionNet()** `virtual ~CPPCFunctionNet ( ) [virtual]`

**11.96.2.4 "!CPPCFunctionNet()** `!CPPCFunctionNet ( )`

### 11.96.3 Member Function Documentation

**11.96.3.1 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                |

**11.96.3.2 GetAnalogVoltage()** `uint16_t GetAnalogVoltage (`  
`uint16_t channel )`

Reads the current analog voltage

#### Parameters

|                |                    |
|----------------|--------------------|
| <i>channel</i> | The Channel Number |
|----------------|--------------------|

#### Returns

The Analog Voltage

**11.96.3.3 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 |

**11.96.3.4 GetDigitalIn()** `uint16_t GetDigitalIn (`  
`uint16_t channel )`

Reads the digital input state

**Parameters**

|                |                    |
|----------------|--------------------|
| <i>channel</i> | The Channel Number |
|----------------|--------------------|

**Returns**

The Digital State

**11.96.3.5 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 |

**11.96.3.6 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.

**11.96.3.7 GetPumpSpeedUnit()** `int GetPumpSpeedUnit ( uint16_t channel )`

Reads the Pump Speed Unit

**Parameters**

|                |                    |
|----------------|--------------------|
| <i>channel</i> | The Channel Number |
|----------------|--------------------|

**Returns**

The Speed Unit

**11.96.3.8 GetSupplyVoltage()** `uint16_t GetSupplyVoltage ( )`

Reads the current supply voltage in mV

**Returns**

The supply voltage

**11.96.3.9 GetValveActive()** `int GetValveActive ( uint16_t valve )`

Gets the valve active/inactive state

**Parameters**

|              |                  |
|--------------|------------------|
| <i>valve</i> | The valve number |
|--------------|------------------|

**Returns**

The valve state

**11.96.3.10 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 |

**11.96.3.11 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 |

**11.96.3.12 MeasureReservoir()** `int32_t MeasureReservoir ( )`

Measures the reservoir pressure

## Returns

The pressure

**11.96.3.13 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 |

**11.96.3.14 SetPressureOffset()** `void SetPressureOffset ( )`

Sets the pressure offset

**11.96.3.15 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 |

**11.96.3.16 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. |

**11.96.3.17 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     |

**11.96.3.18 SetValveActive()** `void SetValveActive (`  
    `uint16_t valve,`  
    `int valveActive )`

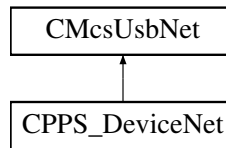
Sets the valve active/inactive state

**Parameters**

|                    |                  |
|--------------------|------------------|
| <i>valve</i>       | The valve number |
| <i>valveActive</i> | The valve state  |

## 11.97 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

#### 11.97.1 Constructor & Destructor Documentation

**11.97.1.1** [CPPS\\_DeviceNet](#)() [CPPS\\_DeviceNet](#) (  
void )

#### 11.97.2 Property Documentation

**11.97.2.1** **McsBus** [CMcsBusNet](#)<sup>^</sup> [McsBus](#) [get]

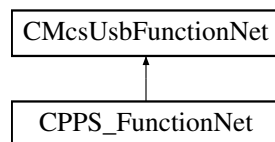
**11.97.2.2** **McsBus\_MotorControl** [CMcsBus\\_MotorControlNet](#)<sup>^</sup> [McsBus\\_MotorControl](#) [get]

**11.97.2.3** **McsBus\_Sensor** [CMcsBus\\_SensorNet](#)<sup>^</sup> [McsBus\\_Sensor](#) [get]

**11.97.2.4 PPS\_Function** `CPPS_FunctionNet^ PPS_Function [get]`

## 11.98 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 GetDigitalIn (unsigned short index)`
- `unsigned short GetBubbleState ()`

### Additional Inherited Members

#### 11.98.1 Constructor & Destructor Documentation



**11.98.1.1 CPPS\_FunctionNet()** [1/2] `CPPS_FunctionNet (`  
    `CMcsUsbNet^ mcsusb,`  
    `CMcsUsbFunctionPointerContainer^ cPPS_FunctionPointerContainer )`

**11.98.1.2 CPPS\_FunctionNet()** [2/2] `CPPS_FunctionNet (`  
    `CMcsUsbNet^ mcsusb )`

## 11.98.2 Member Function Documentation

**11.98.2.1 GetAnalogVoltage()** `unsigned short GetAnalogVoltage (`  
    `unsigned short index )`

**11.98.2.2 GetAnalogVoltages()** `void GetAnalogVoltages (`  
    `unsigned short index,`  
    `[System::Runtime::InteropServices::Out] unsigned short% minvoltage,`  
    `[System::Runtime::InteropServices::Out] unsigned short% maxvoltage )`

**11.98.2.3 GetBubbleState()** `unsigned short GetBubbleState ( )`

**11.98.2.4 GetDigitalIn()** `unsigned short GetDigitalIn (`  
    `unsigned short index )`

**11.98.2.5 GetPumpCouple()** `unsigned int GetPumpCouple ( )`

**11.98.2.6 GetPumpEnableSpeedRatio()** `unsigned int GetPumpEnableSpeedRatio ( )`

**11.98.2.7 GetPumpFastOnOff()** `unsigned int GetPumpFastOnOff (`  
    `unsigned short index )`

**11.98.2.8 GetPumpFastSpeed()** short GetPumpFastSpeed (   
 unsigned short *index* )

**11.98.2.9 GetPumpFunctionSpeeds()** void GetPumpFunctionSpeeds (   
 unsigned short *index*,   
 [System::Runtime::InteropServices::Out] short% *offspeed*,   
 [System::Runtime::InteropServices::Out] short% *onspeed* )

**11.98.2.10 GetPumpManualOnOff()** unsigned int GetPumpManualOnOff (   
 unsigned short *index* )

**11.98.2.11 GetPumpMaxSpeed()** unsigned short GetPumpMaxSpeed (   
 unsigned short *index* )

**11.98.2.12 GetPumpModeType()** [PP\\_Pump\\_Mode\\_Type\\_EnumNet](#) GetPumpModeType (   
 unsigned short *index* )

**11.98.2.13 GetPumpSpeedRatio()** int GetPumpSpeedRatio ( )

**11.98.2.14 GetPumpSpeedUnit()** int GetPumpSpeedUnit (   
 unsigned short *index* )

**11.98.2.15 GetSupplyVoltage()** unsigned short GetSupplyVoltage ( )

**11.98.2.16 GetUseBubble()** unsigned int GetUseBubble (   
 unsigned short *index* )

**11.98.2.17 SetAnalogVoltages()** void SetAnalogVoltages (   
    unsigned short *index*,  
    unsigned short *minvoltage*,  
    unsigned short *maxvoltage* )

**11.98.2.18 SetPumpCouple()** void SetPumpCouple (   
    unsigned int *i* )

**11.98.2.19 SetPumpEnableSpeedRatio()** void SetPumpEnableSpeedRatio (   
    unsigned int *enable* )

**11.98.2.20 SetPumpFastOnOff()** void SetPumpFastOnOff (   
    unsigned short *index*,  
    unsigned int *onoff* )

**11.98.2.21 SetPumpFastSpeed()** void SetPumpFastSpeed (   
    unsigned short *index*,  
    short *fastspeed* )

**11.98.2.22 SetPumpFunctionSpeeds()** void SetPumpFunctionSpeeds (   
    unsigned short *index*,  
    short *offspeed*,  
    short *onspeed* )

**11.98.2.23 SetPumpManualOnOff()** void SetPumpManualOnOff (   
    unsigned short *index*,  
    unsigned int *onoff* )

**11.98.2.24 SetPumpMaxSpeed()** void SetPumpMaxSpeed (   
    unsigned short *index*,  
    unsigned short *maxspeed* )

**11.98.2.25 SetPumpModeType()** `void SetPumpModeType (`  
    `unsigned short index,`  
    `PP_Pump_Mode_Type_EnumNet type )`

**11.98.2.26 SetPumpSpeedRatio()** `void SetPumpSpeedRatio (`  
    `int ratio )`

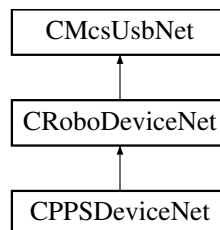
**11.98.2.27 SetPumpSpeedUnit()** `void SetPumpSpeedUnit (`  
    `unsigned short index,`  
    `int speedunit )`

**11.98.2.28 SetUseBubble()** `void SetUseBubble (`  
    `unsigned short index,`  
    `unsigned int usebubble )`

## 11.99 CPPSDeviceNet Class Reference

CPPS4plus1DeviceNet is the to control the MCS HLA device

Inheritance diagram for CPPSDeviceNet:



### Public Member Functions

- [CPPSDeviceNet](#) (void)

### Additional Inherited Members

#### 11.99.1 Detailed Description

CPPS4plus1DeviceNet is the to control the MCS HLA device

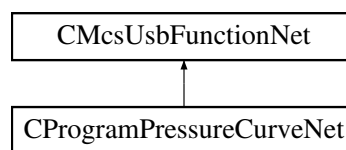
## 11.99.2 Constructor & Destructor Documentation

**11.99.2.1 CPPSDeviceNet()** `CPPSDeviceNet ( void )`

## 11.100 CProgramPressureCurveNet Class Reference

`CProgramPressureCurveNet` is the class to program pressure curves

Inheritance diagram for `CProgramPressureCurveNet`:



### Public Member Functions

- `CProgramPressureCurveNet (CMcsUsbNet^ mcsusb)`  
*Initializes a new instance of the `CPulseGeneratorFunctionNet` class.*
- virtual `~CProgramPressureCurveNet (void)`
- `!CProgramPressureCurveNet (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

#### 11.100.1 Detailed Description

`CProgramPressureCurveNet` is the class to program pressure curves

#### 11.100.2 Constructor & Destructor Documentation

**11.100.2.1 CProgramPressureCurveNet()** `CProgramPressureCurveNet ( CMcsUsbNet^ mcsusb )`

Initializes a new instance of the `CPulseGeneratorFunctionNet` class.

**11.100.2.2** `~CProgramPressureCurveNet()` `virtual ~CProgramPressureCurveNet (`  
`void ) [virtual]`

**11.100.2.3** `!CProgramPressureCurveNet()` `!CProgramPressureCurveNet (`  
`void )`

### 11.100.3 Member Function Documentation

**11.100.3.1** `GetRepeats()` `unsigned int GetRepeats (`  
`unsigned char busnumber,`  
`unsigned char busaddress,`  
`int32_t channel )`

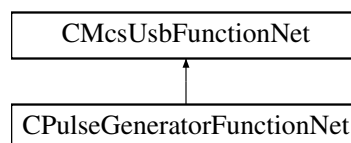
**11.100.3.2** `Program()` `void Program (`  
`unsigned char busnumber,`  
`unsigned char busaddress,`  
`int32_t channel,`  
`array< int32_t >^ pressures,`  
`array< int32_t >^ steps,`  
`array< int16_t >^ durations )`

**11.100.3.3** `SetRepeats()` `void SetRepeats (`  
`unsigned char busnumber,`  
`unsigned char busaddress,`  
`int32_t channel,`  
`uint32_t repeats )`

## 11.101 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][PulseGenerator\\_Mode\\_EnumNet](#)% mode, [System::Runtime::InteropServices::Out][int32\\_t](#)% digitalchannel)  
*Reads the generator mode*
- void [SetModeSelect](#) ([int32\\_t](#) generator\_number, [PulseGenerator\\_Mode\\_EnumNet](#) mode, [int32\\_t](#) digitalchannel)  
*Writes the generator mode*

## Additional Inherited Members

### 11.101.1 Detailed Description

[CPulseGeneratorFunctionNet](#) is the class to control the pulse generator for video tracking

### 11.101.2 Constructor & Destructor Documentation

**11.101.2.1 [CPulseGeneratorFunctionNet](#)() [1/2]** [CPulseGeneratorFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb,  
[CMcsUsbFunctionPointerContainer](#)<sup>^</sup> pPulseGeneratorFunctionPointerContainer )

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

**11.101.2.2 [CPulseGeneratorFunctionNet](#)() [2/2]** [CPulseGeneratorFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb )

**11.101.2.3 [~CPulseGeneratorFunctionNet](#)()** virtual [~CPulseGeneratorFunctionNet](#) ( ) [virtual]

**11.101.2.4** `!!CPulseGeneratorFunctionNet() !CPulseGeneratorFunctionNet ( )`

### 11.101.3 Member Function Documentation

**11.101.3.1 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 |

**11.101.3.2 GetPeriod()** `int32_t GetPeriod (`  
    `int32_t generator_number )`

Reads the generator period

#### Parameters

|                         |                      |
|-------------------------|----------------------|
| <i>generator_number</i> | The generator number |
|-------------------------|----------------------|

#### Returns

The period

**11.101.3.3 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

**11.101.3.4 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 |

**11.101.3.5 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           |

**11.101.3.6 SetPulseLength()** void SetPulseLength (   
     int32\_t *generator\_number*,   
     int32\_t *pulselength\_in\_10us* )

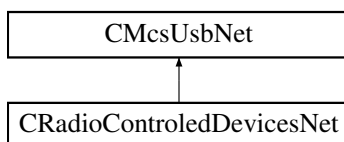
Writes the generator pulse length

**Parameters**

|                            |                      |
|----------------------------|----------------------|
| <i>generator_number</i>    | The generator number |
| <i>pulselength_in_10us</i> | The pulse length     |

**11.102 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

#### 11.102.1 Constructor & Destructor Documentation

**11.102.1.1 CRadioControlledDevicesNet()** [1/2] [CRadioControlledDevicesNet](#) (  
void )

**11.102.1.2 CRadioControlledDevicesNet()** [2/2] [CRadioControlledDevicesNet](#) (  
CRadioControlledDevices \* *pRadioControlled* ) [protected]

#### 11.102.2 Member Function Documentation

**11.102.2.1 ConnectDevice()** void [ConnectDevice](#) (  
unsigned short *sn* )

**11.102.2.2 DisConnectDevice()** void [DisConnectDevice](#) ( )

**11.102.2.3 GetDeviceNames()** `array<unsigned short> ^ GetDeviceNames ( )`

**11.102.2.4 GetFrequency()** `unsigned short GetFrequency ( )`

**11.102.2.5 HasRadioControl()** `bool HasRadioControl ( )`

**11.102.2.6 SetFrequency()** `void SetFrequency (   
 unsigned short frequency )`

**11.102.2.7 StillConnected()** `bool StillConnected ( )`

## 11.103 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](#)

### 11.103.1 Constructor & Destructor Documentation

**11.103.1.1 CRegionOfInterestRect()** `CRegionOfInterestRect (   
 int left,   
 int top,   
 int right,   
 int bottom )`

### 11.103.2 Member Function Documentation

**11.103.2.1 DeepCopy()** [CRegionOfInterestRect](#) ^ DeepCopy ( )

### 11.103.3 Member Data Documentation

**11.103.3.1 m\_Bottom** [int](#) m\_Bottom

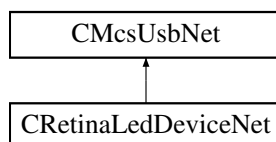
**11.103.3.2 m\_Left** [int](#) m\_Left

**11.103.3.3 m\_Right** [int](#) m\_Right

**11.103.3.4 m\_Top** [int](#) m\_Top

## 11.104 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

### 11.104.1 Constructor & Destructor Documentation

**11.104.1.1 CRetinaLedDeviceNet()** `CRetinaLedDeviceNet ( )`

**11.104.1.2 ~CRetinaLedDeviceNet()** `~CRetinaLedDeviceNet ( )`

### 11.104.2 Member Function Documentation

**11.104.2.1 AddLoopEntry()** `unsigned int AddLoopEntry ( unsigned short repeats, unsigned short steps_back )`

**11.104.2.2 AddTableEntry()** `unsigned int AddTableEntry ( unsigned long long pattern )`

**11.104.2.3 ClearTable()** `unsigned int ClearTable ( )`

**11.104.2.4 GetTablepointer()** `unsigned int GetTablepointer ( int % position )`

**11.104.2.5 SetLED()** `unsigned int SetLED ( unsigned long long pattern )`

**11.104.2.6 SetLumi()** `unsigned int SetLumi ( int lumi )`

**11.104.2.7 SetPersistency()** unsigned int SetPersistency (   
 unsigned int *persistency* )

**11.104.2.8 SetRepeat()** unsigned int SetRepeat (   
 int *repeat* )

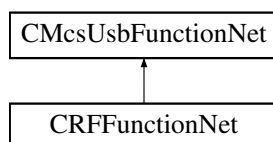
**11.104.2.9 SetTablepointer()** unsigned int SetTablepointer (   
 int *position* )

**11.104.2.10 SetTrigger()** unsigned int SetTrigger (   
 int *enable* )

## 11.105 CRFFunctionNet Class Reference

[CRFFunctionNet](#) is the class to control RF devices

Inheritance diagram for CRFFunctionNet:



### Public Member Functions

- [CRFFunctionNet](#) (CMcsUsbNet<sup>^</sup> mcsusb, CMcsUsbFunctionPointerContainer<sup>^</sup> pRFFunctionPointerContainer)  
*Initializes a new instance of the [CRFFunctionNet](#) class.*
- [CRFFunctionNet](#) (CMcsUsbNet<sup>^</sup> mcsusb)
- virtual ~[CRFFunctionNet](#) ()
- ![CRFFunctionNet](#) ()
- uint32\_t [GetBaseFrequency](#) (CFirmwareDestinationNet destination)  
*gets the base advertise frequency*
- void [SetBaseFrequency](#) (CFirmwareDestinationNet destination, uint32\_t frequency)  
*sets the base advertise frequency*
- uint32\_t [GetWorkingFrequency](#) ()  
*gets the working frequency*
- void [SetWorkingFrequency](#) (uint32\_t frequency)  
*sets the working frequency*
- array< uint32\_t > <sup>^</sup> [GetAvailableDeviceListEx](#) (int list\_Length)  
*get a list of available devices*
- array< uint32\_t > <sup>^</sup> [GetAvailableDeviceList](#) ()

- get a list of available devices*
- array< uint32\_t > ^ [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

### 11.105.1 Detailed Description

[CRFFunctionNet](#) is the class to control RF devices

### 11.105.2 Constructor & Destructor Documentation

**11.105.2.1 CRFFunctionNet()** [1/2] [CRFFunctionNet](#) (   
[CMcsUsbNet](#)^ mcsusb,   
[CMcsUsbFunctionPointerContainer](#)^ pRFFunctionPointerContainer )

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

**11.105.2.2 CRFFunctionNet()** [2/2] [CRFFunctionNet](#) (   
[CMcsUsbNet](#)^ mcsusb )

**11.105.2.3 ~CRFFunctionNet()** virtual [~CRFFunctionNet](#) ( ) [virtual]

**11.105.2.4 "!CRFFunctionNet()** [!CRFFunctionNet](#) ( )

### 11.105.3 Member Function Documentation

**11.105.3.1 Connect()** void [Connect](#) (   
uint32\_t sn )

connect to a RF device, use 0 to disconnect

**Parameters**

|           |                   |
|-----------|-------------------|
| <i>sn</i> | the serial number |
|-----------|-------------------|

**11.105.3.2 GetAvailableDeviceList()** `array<uint32_t> ^ GetAvailableDeviceList ( )`

get a list of available devices

**Returns**

array of devices

**11.105.3.3 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

**11.105.3.4 GetAvailableStateList()** `array<uint32_t> ^ GetAvailableStateList ( )`

get a list of the states of the available devices

**Returns**

array of states

**11.105.3.5 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

**11.105.3.6 GetBaseFrequency()** `uint32_t GetBaseFrequency ( CFirmwareDestinationNet destination )`

gets the base advertise frequency

**Parameters**

|                    |                          |
|--------------------|--------------------------|
| <i>destination</i> | the destination to query |
|--------------------|--------------------------|

**Returns**

the frequency

**11.105.3.7 GetConnectedDevice()** `uint32_t GetConnectedDevice ( )`

get connect RF device, 0 = no device connected

**Returns**

the serial number

**11.105.3.8 GetState()** `uint32_t GetState ( )`

get connection state

**Returns**

the state

**11.105.3.9 GetTestMode()** `uint32_t GetTestMode ( )`

gets test mode

**Returns**

the mode

**11.105.3.10 GetWorkingFrequency()** `uint32_t GetWorkingFrequency ( )`

gets the working frequency

**Returns**

the frequency

**11.105.3.11 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          |

**11.105.3.12 SetTestMode()** `void SetTestMode (   
    uint32_t mode )`

set test mode

**Parameters**

|             |          |
|-------------|----------|
| <i>mode</i> | the mode |
|-------------|----------|

**11.105.3.13 SetWorkingFrequency()** `void SetWorkingFrequency (   
    uint32_t frequency )`

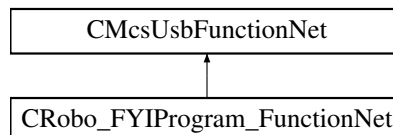
sets the working frequency

## Parameters

|                  |               |
|------------------|---------------|
| <i>frequency</i> | the frequency |
|------------------|---------------|

## 11.106 CRobo\_FYIProgram\_FunctionNet Class Reference

Inheritance diagram for CRobo\_FYIProgram\_FunctionNet:



## Public Member Functions

- [CRobo\\_FYIProgram\\_FunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> robo\_↔ FYIProgram\_FunctionPointerContainer)
- [CRobo\\_FYIProgram\\_FunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- void [SetValve1](#) (unsigned char index, unsigned int valve1)
- unsigned int [GetValve1](#) (unsigned char index)
- void [SetValve2](#) (unsigned char index, unsigned int valve2)
- unsigned int [GetValve2](#) (unsigned char index)
- void [SetLength](#) (unsigned char index, int length)
- int [GetLength](#) (unsigned char index)
- void [Start](#) ()
- int [GetState](#) ()

## Additional Inherited Members

## 11.106.1 Constructor &amp; Destructor Documentation

**11.106.1.1 CRobo\_FYIProgram\_FunctionNet()** [1/2] [CRobo\\_FYIProgram\\_FunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb,  
[CMcsUsbFunctionPointerContainer](#)<sup>^</sup> robo\_FYIProgram\_FunctionPointerContainer )

**11.106.1.2 CRobo\_FYIProgram\_FunctionNet()** [2/2] [CRobo\\_FYIProgram\\_FunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb )

## 11.106.2 Member Function Documentation

**11.106.2.1 GetLength()** `int GetLength (`  
`unsigned char index )`

**11.106.2.2 GetState()** `int GetState ( )`

**11.106.2.3 GetValve1()** `unsigned int GetValve1 (`  
`unsigned char index )`

**11.106.2.4 GetValve2()** `unsigned int GetValve2 (`  
`unsigned char index )`

**11.106.2.5 SetLength()** `void SetLength (`  
`unsigned char index,`  
`int length )`

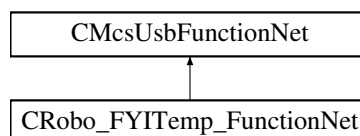
**11.106.2.6 SetValve1()** `void SetValve1 (`  
`unsigned char index,`  
`unsigned int valve1 )`

**11.106.2.7 SetValve2()** `void SetValve2 (`  
`unsigned char index,`  
`unsigned int valve2 )`

**11.106.2.8 Start()** `void Start ( )`

## 11.107 CRobo\_FYITemp\_FunctionNet Class Reference

Inheritance diagram for CRobo\_FYITemp\_FunctionNet:



## Public Member Functions

- [CRobo\\_FYITemp\\_FunctionNet](#) ([CMcsUsbNet](#)^ 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

### 11.107.1 Constructor & Destructor Documentation

**11.107.1.1 CRobo\_FYITemp\_FunctionNet()** [CRobo\\_FYITemp\\_FunctionNet](#) (  
    [CMcsUsbNet](#)^ mcsusb )

### 11.107.2 Member Function Documentation

**11.107.2.1 GetICoeff()** int GetICoeff (  
    unsigned char index )

**11.107.2.2 GetMaxPower()** int GetMaxPower (  
    unsigned char index )

**11.107.2.3 GetPCoeff()** int GetPCoeff (  
    unsigned char index )

**11.107.2.4 GetRegulatorOnOff()** int GetRegulatorOnOff (  
    unsigned char index )

**11.107.2.5 GetSollTemp()** `int GetSollTemp (`  
    `unsigned char index )`

**11.107.2.6 SetICoeff()** `void SetICoeff (`  
    `unsigned char index,`  
    `int icoeff )`

**11.107.2.7 SetMaxPower()** `void SetMaxPower (`  
    `unsigned char index,`  
    `int power )`

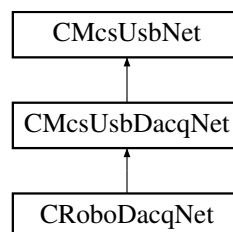
**11.107.2.8 SetPCoeff()** `void SetPCoeff (`  
    `unsigned char index,`  
    `int pcoeff )`

**11.107.2.9 SetRegulatorOnOff()** `void SetRegulatorOnOff (`  
    `unsigned char index,`  
    `int onoff )`

**11.107.2.10 SetSollTemp()** `void SetSollTemp (`  
    `unsigned char index,`  
    `int temp )`

## 11.108 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 [SetTriggerMaskValue](#) (unsigned int mask, unsigned int value, unsigned int virtualDevice)
- 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 [GetUJOffset](#) ()
- int [GetUCOffset](#) ()
- int [GetICOffset](#) ()
- int [GetCrossTalkOffset](#) ()
- int [GetXGain](#) ()
- int [GetCrossTalkOptimum](#) ()
- unsigned int [GetRecordingNumber](#) ()
- int [GetResistanceC](#) ()
- int [GetResistanceV](#) ()
- int [GetCapacityC](#) ()
- int [GetCapacityV](#) ()
- int [GetCapacityX](#) ()
- int [GetUV](#) ()
- int [GetUC](#) ()
- int [GetIC](#) ()
- int [GetNUV\\_MS](#) ()
- int [GetNUC\\_MS](#) ()
- int [GetNIC\\_MS](#) ()
- void [SetAllDigout](#) (uint32\_t value)
- uint32\_t [GetAllDigout](#) ()
- void [SetCommand](#) (unsigned char command, int value)
- int [GetCommand](#) (unsigned char command)
- void [SetDigout](#) (uint16\_t index, bool enable)
- bool [GetDigout](#) (uint16\_t index)
- void [TableDefBegin](#) ()
- void [TableDefEnd](#) ()
- void [Table\\_Wait](#) (unsigned int tableWait)
- void [SetDownsampleFactor](#) (int index, int downsample\_factor)
- void [SetFilterCoeffs](#) (int index, array< int > ^ coeffs)
- 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](#) ()

### Static Public Attributes

- static const unsigned int [TriggerMask\\_Default](#) = 0xFF00
- static const unsigned int [TriggerValue\\_MoveAbs](#) = COMMAND\_ROBO\_MOVEABS
- static const unsigned int [TriggerValue\\_StartQueue](#) = COMMAND\_ROBO\_QUEUE
- static const unsigned int [VirtualDevice\\_ContinuousDacq](#) = 0
- static const unsigned int [VirtualDevice\\_TableRun](#) = 1

### Additional Inherited Members

#### 11.108.1 Constructor & Destructor Documentation

**11.108.1.1 CRoboDacqNet()** [1/2] [CRoboDacqNet](#) (  
void )

**11.108.1.2 CRoboDacqNet()** [2/2] [CRoboDacqNet](#) (  
[CRoboDeviceNet](#)^ *robodevice* )

#### 11.108.2 Member Function Documentation

**11.108.2.1 CancelTableLoop()** void [CancelTableLoop](#) ( )

**11.108.2.2 CancelTableLoopAndStopTable()** void [CancelTableLoopAndStopTable](#) ( )

**11.108.2.3 ClampAmpRestart()** void [ClampAmpRestart](#) ( )

**11.108.2.4 DoRamp()** `void DoRamp (`  
    `int startValue,`  
    `int endValue,`  
    `int duration,`  
    `int mode )`

**11.108.2.5 Emu\_GetCellCapacity()** `int Emu_GetCellCapacity ( )`

**11.108.2.6 Emu\_GetCellPotential()** `int Emu_GetCellPotential ( )`

**11.108.2.7 Emu\_GetCellResists()** `int Emu_GetCellResists ( )`

**11.108.2.8 Emu\_GetElectrodeResists()** `int Emu_GetElectrodeResists ( )`

**11.108.2.9 Emu\_GetNoise()** `int Emu_GetNoise ( )`

**11.108.2.10 Emu\_SetCellCapacity()** `void Emu_SetCellCapacity (`  
    `int emuCellCapacity )`

**11.108.2.11 Emu\_SetCellPotential()** `void Emu_SetCellPotential (`  
    `int emuCellPotential )`

**11.108.2.12 Emu\_SetCellResists()** `void Emu_SetCellResists (`  
    `int emuCellResist )`

**11.108.2.13 Emu\_SetElectrodeResists()** `void Emu_SetElectrodeResists (`  
    `int emuElectrodeResist )`

**11.108.2.14 Emu\_SetNoise()** void Emu\_SetNoise (   
int *emuNoise* )

**11.108.2.15 GetAllDigout()** uint32\_t GetAllDigout ( )

**11.108.2.16 GetCapacityC()** int GetCapacityC ( )

**11.108.2.17 GetCapacityV()** int GetCapacityV ( )

**11.108.2.18 GetCapacityX()** int GetCapacityX ( )

**11.108.2.19 GetClampAmpSerialNumber()** unsigned int GetClampAmpSerialNumber ( )

**11.108.2.20 GetCommand()** int GetCommand (   
unsigned char *command* )

**11.108.2.21 GetConfigurationBit()** bool GetConfigurationBit (   
unsigned short *bit* )

**11.108.2.22 GetConfigurationBitAxc()** bool GetConfigurationBitAxc ( )

**11.108.2.23 GetConfigurationBitBlu\_Led()** bool GetConfigurationBitBlu\_Led ( )

**11.108.2.24 GetConfigurationBitBlu\_LedToggleFast()** bool GetConfigurationBitBlu\_LedToggleFast ( )

**11.108.2.25    GetConfigurationBitBlu\_LedToggleSlow()**    `bool GetConfigurationBitBlu_LedToggleSlow ( )`

**11.108.2.26    GetConfigurationBitCC\_Gen()**    `bool GetConfigurationBitCC_Gen ( )`

**11.108.2.27    GetConfigurationBitCV\_Gen()**    `bool GetConfigurationBitCV_Gen ( )`

**11.108.2.28    GetConfigurationBitRC\_Gen()**    `bool GetConfigurationBitRC_Gen ( )`

**11.108.2.29    GetConfigurationBitRed\_Led()**    `bool GetConfigurationBitRed_Led ( )`

**11.108.2.30    GetConfigurationBitRed\_LedSaturation()**    `bool GetConfigurationBitRed_LedSaturation ( )`

**11.108.2.31    GetConfigurationBitRed\_LedToggleFast()**    `bool GetConfigurationBitRed_LedToggleFast ( )`

**11.108.2.32    GetConfigurationBitRed\_LedToggleSlow()**    `bool GetConfigurationBitRed_LedToggleSlow ( )`

**11.108.2.33    GetConfigurationBitRelais()**    `bool GetConfigurationBitRelais ( )`

**11.108.2.34    GetConfigurationBitRV\_Gen()**    `bool GetConfigurationBitRV_Gen ( )`

**11.108.2.35    GetConfigurationBits()**    `unsigned int GetConfigurationBits ( )`

**11.108.2.36 GetConfigurationBitStream()** `bool GetConfigurationBitStream ( )`

**11.108.2.37 GetConfigurationBitSupply()** `bool GetConfigurationBitSupply ( )`

**11.108.2.38 GetCrossTalkOffset()** `int GetCrossTalkOffset ( )`

**11.108.2.39 GetCrossTalkOptimum()** `int GetCrossTalkOptimum ( )`

**11.108.2.40 GetDigout()** `bool GetDigout (`  
`uint16_t index )`

**11.108.2.41 GetDisplayText()** `String ^ GetDisplayText (`  
`int index )`

**11.108.2.42 GetDownsampleFactor()** `int GetDownsampleFactor (`  
`int index )`

**11.108.2.43 GetFilter()** `int GetFilter ( )`

**11.108.2.44 GetFilterCoeffs()** `array<int> ^ GetFilterCoeffs (`  
`int index )`

**11.108.2.45 GetIC()** `int GetIC ( )`

**11.108.2.46 GetIClamp()** `int GetIClamp ( )`

**11.108.2.47 GetICOffset()** `int GetICOffset ( )`

**11.108.2.48 GetIGain()** `int GetIGain ( )`

**11.108.2.49 GetNIC\_MS()** `int GetNIC_MS ( )`

**11.108.2.50 GetNUC\_MS()** `int GetNUC_MS ( )`

**11.108.2.51 GetNUV\_MS()** `int GetNUV_MS ( )`

**11.108.2.52 GetPGain()** `int GetPGain ( )`

**11.108.2.53 GetRecordingNumber()** `unsigned int GetRecordingNumber ( )`

**11.108.2.54 GetResistanceC()** `int GetResistanceC ( )`

**11.108.2.55 GetResistanceV()** `int GetResistanceV ( )`

**11.108.2.56 GetScreen()** `int GetScreen ( )`

**11.108.2.57 GetSimulation()** `unsigned int GetSimulation ( )`

**11.108.2.58 GetUC()** `int GetUC ( )`

**11.108.2.59 GetUClamp()** `int GetUClamp ( )`

**11.108.2.60 GetUCOffset()** `int GetUCOffset ( )`

**11.108.2.61 GetUpdateDisplay()** `int GetUpdateDisplay ( )`

**11.108.2.62 GetUV()** `int GetUV ( )`

**11.108.2.63 GetUVOffset()** `int GetUVOffset ( )`

**11.108.2.64 GetXGain()** `int GetXGain ( )`

**11.108.2.65 RunTable()** [1/2] `void RunTable ( )`

**11.108.2.66 RunTable()** [2/2] `void RunTable (`  
`int timeout )`

**11.108.2.67 SetAllDigout()** `void SetAllDigout (`  
`uint32_t value )`

**11.108.2.68 SetCommand()** `void SetCommand (`  
`unsigned char command,`  
`int value )`

**11.108.2.69 SetConfigurationBit()** void SetConfigurationBit (   
    unsigned short *bit*,   
    bool *value* )

**11.108.2.70 SetConfigurationBitAxc()** void SetConfigurationBitAxc (   
    bool *value* )

**11.108.2.71 SetConfigurationBitBlu\_Led()** void SetConfigurationBitBlu\_Led (   
    bool *value* )

**11.108.2.72 SetConfigurationBitBlu\_LedToggleFast()** void SetConfigurationBitBlu\_LedToggleFast (   
    bool *value* )

**11.108.2.73 SetConfigurationBitBlu\_LedToggleSlow()** void SetConfigurationBitBlu\_LedToggleSlow (   
    bool *value* )

**11.108.2.74 SetConfigurationBitCC\_Gen()** void SetConfigurationBitCC\_Gen (   
    bool *value* )

**11.108.2.75 SetConfigurationBitCV\_Gen()** void SetConfigurationBitCV\_Gen (   
    bool *value* )

**11.108.2.76 SetConfigurationBitRC\_Gen()** void SetConfigurationBitRC\_Gen (   
    bool *value* )

**11.108.2.77 SetConfigurationBitRed\_Led()** void SetConfigurationBitRed\_Led (   
    bool *value* )



**11.108.2.78 SetConfigurationBitRed\_LedSaturation()** void SetConfigurationBitRed\_LedSaturation ( bool *value* )

**11.108.2.79 SetConfigurationBitRed\_LedToggleFast()** void SetConfigurationBitRed\_LedToggleFast ( bool *value* )

**11.108.2.80 SetConfigurationBitRed\_LedToggleSlow()** void SetConfigurationBitRed\_LedToggleSlow ( bool *value* )

**11.108.2.81 SetConfigurationBitRelais()** void SetConfigurationBitRelais ( bool *value* )

**11.108.2.82 SetConfigurationBitRV\_Gen()** void SetConfigurationBitRV\_Gen ( bool *value* )

**11.108.2.83 SetConfigurationBitStream()** void SetConfigurationBitStream ( bool *value* )

**11.108.2.84 SetConfigurationBitSupply()** void SetConfigurationBitSupply ( bool *value* )

**11.108.2.85 SetCrossTalkOffset()** void SetCrossTalkOffset ( int *CrossTalk* )

**11.108.2.86 SetCrossTalkOptimum()** void SetCrossTalkOptimum ( int *cxOptimum* )

**11.108.2.87 SetDigout()** void SetDigout (   
    uint16\_t *index*,  
    bool *enable* )

**11.108.2.88 SetDisplayText()** void SetDisplayText (   
    int *index*,  
    String^ *displayText* )

**11.108.2.89 SetDownsampleFactor()** void SetDownsampleFactor (   
    int *index*,  
    int *downsample\_factor* )

**11.108.2.90 SetFilter()** void SetFilter (   
    int *filter* )

**11.108.2.91 SetFilterCoeffs()** void SetFilterCoeffs (   
    int *index*,  
    array< int >^ *coeffs* )

**11.108.2.92 SetIClamp()** void SetIClamp (   
    int *iClamp* )

**11.108.2.93 SetICOffset()** void SetICOffset (   
    int *ICOffset* )

**11.108.2.94 SetIGain()** void SetIGain (   
    int *iGain* )

**11.108.2.95 SetNoFilterCoeffs()** void SetNoFilterCoeffs (   
    int *index* )

**11.108.2.96 SetPGain()** void SetPGain (  
int *pGain* )

**11.108.2.97 SetRecordingNumber()** void SetRecordingNumber (  
unsigned int *recordingNumber* )

**11.108.2.98 SetScreen()** void SetScreen (  
int *screen* )

**11.108.2.99 SetSimulation()** void SetSimulation (  
unsigned int *enable* )

**11.108.2.100 SetTriggerMaskValue()** void SetTriggerMaskValue (  
unsigned int *mask*,  
unsigned int *value*,  
unsigned int *virtualDevice* )

**11.108.2.101 SetUClamp()** void SetUClamp (  
int *uClamp* )

**11.108.2.102 SetUCOffset()** void SetUCOffset (  
int *UCOffset* )

**11.108.2.103 SetUVOffset()** void SetUVOffset (  
int *UVOffset* )

**11.108.2.104 SetXGain()** void SetXGain (  
int *xGain* )

**11.108.2.105 StopTable()** [1/2] `void StopTable ( )`

**11.108.2.106 StopTable()** [2/2] `void StopTable (`  
`int timeout )`

**11.108.2.107 Table\_Wait()** `void Table_Wait (`  
`unsigned int tableWait )`

**11.108.2.108 TableDefBegin()** `void TableDefBegin ( )`

**11.108.2.109 TableDefEnd()** `void TableDefEnd ( )`

**11.108.2.110 UpdateDisplay()** `void UpdateDisplay ( )`

### 11.108.3 Member Data Documentation

**11.108.3.1 TriggerMask\_Default** `const unsigned int TriggerMask_Default = 0xFF00 [static]`

**11.108.3.2 TriggerValue\_MoveAbs** `const unsigned int TriggerValue_MoveAbs = COMMAND_ROBO_↔`  
`MOVEABS [static]`

**11.108.3.3 TriggerValue\_StartQueue** `const unsigned int TriggerValue_StartQueue = COMMAND_ROBO_↔`  
`QUEUE [static]`

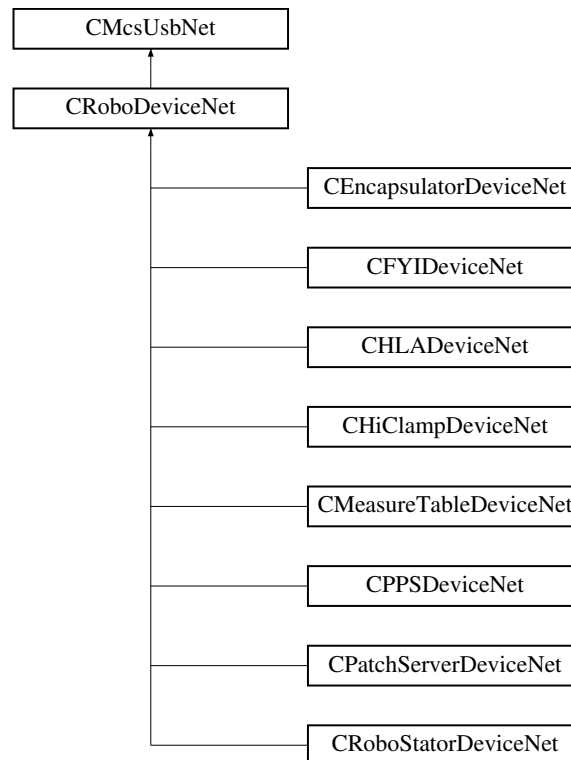
**11.108.3.4 VirtualDevice\_ContinousDacq** `const unsigned int VirtualDevice_ContinousDacq = 0`  
`[static]`

**11.108.3.5 VirtualDevice\_TableRun** `const unsigned int VirtualDevice_TableRun = 1 [static]`

## 11.109 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)  
*Moves the motor to the new absolute position*

- void [MoveAbs](#) (unsigned char busaddress, char axes, int x, int y, int timeout)  
*Moves the motor to the new absolute position*
- void [MoveAbs](#) (unsigned char busaddress, char axes, array< int >^ pos)  
*Moves the motor to the new absolute position*
- void [MoveAbs](#) (unsigned char busaddress, char axes, array< int >^ pos, 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)
- bool [IsQueueEnabled](#) ()
- void [EnableQueue](#) (bool enable)
- bool [IsQueueStarted](#) ()
- void [StartQueue](#) (bool start)
- void [WaitTimer](#) (uint32\_t waittime, 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](#) ()
- void [NullCommand](#) (unsigned int marker)
- 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

### 11.109.1 Detailed Description

[CRoboDeviceNet](#) is the base class for all Robo platform based devices

### 11.109.2 Constructor & Destructor Documentation

**11.109.2.1 [CRoboDeviceNet](#)()** [CRoboDeviceNet](#) (  
void )

**11.109.2.2 [~CRoboDeviceNet](#)()** [~CRoboDeviceNet](#) (  
void )

### 11.109.3 Member Function Documentation

**11.109.3.1 [CancelPoolLoop](#)()** void [CancelPoolLoop](#) ( )



**11.109.3.2 CancelPoolLoopAndStopMovement()** `void CancelPoolLoopAndStopMovement ( )`

**11.109.3.3 EnableQueue()** `void EnableQueue (`  
`bool enable )`

**11.109.3.4 FindReference() [1/2]** `void FindReference (`  
`unsigned char busaddress,`  
`char axes )`

**11.109.3.5 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) |

**11.109.3.6 GetAirpressure()** `unsigned int GetAirpressure ( )`

**11.109.3.7 GetAirpressureLimit()** `void GetAirpressureLimit (`  
`[System::Runtime::InteropServices::Out] unsigned int% lowerpressure,`  
`[System::Runtime::InteropServices::Out] unsigned int% upperpressure )`

**11.109.3.8 GetAirValve()** `unsigned int GetAirValve ( )`

**11.109.3.9 GetCurrentAirvalve()** `unsigned int GetCurrentAirvalve ( )`

**11.109.3.10 GetCurrentAirvalveLimit()** void GetCurrentAirvalveLimit (   
 [System::Runtime::InteropServices::Out] unsigned int% *lowercurrent*,   
 [System::Runtime::InteropServices::Out] unsigned int% *uppercurrent* )

**11.109.3.11 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 |

**11.109.3.12 GetErrorAirpressure()** unsigned int GetErrorAirpressure ( )

**11.109.3.13 GetErrorCurrentAirvalve()** unsigned int GetErrorCurrentAirvalve ( )

**11.109.3.14 GetErrorVoltage12V()** unsigned int GetErrorVoltage12V ( )

**11.109.3.15 GetErrorVoltage5V()** unsigned int GetErrorVoltage5V ( )

**11.109.3.16 GetErrorVoltageAirvalve()** unsigned int GetErrorVoltageAirvalve ( )

**11.109.3.17 GetErrorVoltageRs485A()** unsigned int GetErrorVoltageRs485A ( )

**11.109.3.18 GetErrorVoltageRs485B()** unsigned int GetErrorVoltageRs485B ( )

**11.109.3.19 GetErrorVoltageValves()** unsigned int GetErrorVoltageValves ( )

**11.109.3.20 GetInMovement()** bool GetInMovement ( )

Low level command, gets the internal state "In Movement"

**11.109.3.21 GetMinPressure()** int GetMinPressure ( )

**11.109.3.22 GetMovementError()** uint32\_t GetMovementError ( )

Low level command, gets the error of the last movement end

**11.109.3.23 GetVoltage12V()** unsigned int GetVoltage12V ( )

**11.109.3.24 GetVoltage12VLimit()** void GetVoltage12VLimit (   
 [System::Runtime::InteropServices::Out] unsigned int% *lowervoltage*,   
 [System::Runtime::InteropServices::Out] unsigned int% *uppervoltage* )

**11.109.3.25 GetVoltage5V()** unsigned int GetVoltage5V ( )

**11.109.3.26 GetVoltage5VLimit()** void GetVoltage5VLimit (   
 [System::Runtime::InteropServices::Out] unsigned int% *lowervoltage*,   
 [System::Runtime::InteropServices::Out] unsigned int% *uppervoltage* )

**11.109.3.27 GetVoltageAirvalve()** unsigned int GetVoltageAirvalve ( )

**11.109.3.28 GetVoltageAirvalveLimit()** void GetVoltageAirvalveLimit (   
 [System::Runtime::InteropServices::Out] unsigned int% *lowervoltage*,   
 [System::Runtime::InteropServices::Out] unsigned int% *uppervoltage* )

**11.109.3.29 GetVoltageRs485A()** unsigned int GetVoltageRs485A ( )

**11.109.3.30 GetVoltageRs485ALimit()** void GetVoltageRs485ALimit (   
 [System::Runtime::InteropServices::Out] unsigned int% *lowervoltage*,   
 [System::Runtime::InteropServices::Out] unsigned int% *uppervoltage* )

**11.109.3.31 GetVoltageRs485B()** unsigned int GetVoltageRs485B ( )

**11.109.3.32 GetVoltageRs485BLimit()** void GetVoltageRs485BLimit (   
 [System::Runtime::InteropServices::Out] unsigned int% *lowervoltage*,   
 [System::Runtime::InteropServices::Out] unsigned int% *uppervoltage* )

**11.109.3.33 GetVoltageValves()** unsigned int GetVoltageValves ( )

**11.109.3.34 GetVoltageValvesLimit()** void GetVoltageValvesLimit (   
 [System::Runtime::InteropServices::Out] unsigned int% *lowervoltage*,   
 [System::Runtime::InteropServices::Out] unsigned int% *uppervoltage* )

**11.109.3.35 IsQueueEnabled()** bool IsQueueEnabled ( )

**11.109.3.36 IsQueueStarted()** bool IsQueueStarted ( )

**11.109.3.37 MoveAbs()** [1/4] void MoveAbs (   
 unsigned char *busaddress*,   
 char *axes*,   
 array< int >^ *pos* )

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>pos</i>        | Positions of the axis 0 to 3, if pattern in axes is set |

**11.109.3.38 MoveAbs() [2/4]** `void MoveAbs (`  
     unsigned char *busaddress*,  
     char *axes*,  
     array< int >^ *pos*,  
     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>pos</i>        | Positions of the axis 0 to 3, if pattern in axes is set               |
| <i>timeout</i>    | Timeout of maximal waiting for the end of the command (-1 is forever) |

**11.109.3.39 MoveAbs() [3/4]** `void MoveAbs (`  
     unsigned char *busaddress*,  
     char *axes*,  
     int *x*,  
     int *y* )

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 |

**11.109.3.40 MoveAbs() [4/4]** `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) |

**11.109.3.41 NullCommand()** `void NullCommand (`  
    `unsigned int marker )`

**11.109.3.42 SetAirpressureLimit()** `void SetAirpressureLimit (`  
    `unsigned int lowerpressure,`  
    `unsigned int upperpressure )`

**11.109.3.43 SetAirValve()** `void SetAirValve (`  
    `unsigned int onoff )`

**11.109.3.44 SetCurrentAirvalveLimit()** `void SetCurrentAirvalveLimit (`  
    `unsigned int lowercurrent,`  
    `unsigned int uppercurrent )`

**11.109.3.45 SetCurrentAndAir() [1/2]** `void SetCurrentAndAir (`  
    `unsigned char busaddress,`  
    `char axes,`  
    `unsigned short onoff )`

**11.109.3.46 SetCurrentAndAir() [2/2]** `void SetCurrentAndAir (`  
    `unsigned char busaddress,`  
    `char axes,`  
    `unsigned short onoff,`  
    `int timeout )`

**11.109.3.47 SetInMovement()** `void SetInMovement ( )`

Low level command, sets the internal state to "In Movement"

**11.109.3.48 SetMinPressure()** `void SetMinPressure (`  
`int pressure )`

**11.109.3.49 SetVoltage12VLimit()** `void SetVoltage12VLimit (`  
`unsigned int lowervoltage,`  
`unsigned int uppervoltage )`

**11.109.3.50 SetVoltage5VLimit()** `void SetVoltage5VLimit (`  
`unsigned int lowervoltage,`  
`unsigned int uppervoltage )`

**11.109.3.51 SetVoltageAirvalveLimit()** `void SetVoltageAirvalveLimit (`  
`unsigned int lowervoltage,`  
`unsigned int uppervoltage )`

**11.109.3.52 SetVoltageRs485ALimit()** `void SetVoltageRs485ALimit (`  
`unsigned int lowervoltage,`  
`unsigned int uppervoltage )`

**11.109.3.53 SetVoltageRs485BLimit()** `void SetVoltageRs485BLimit (`  
`unsigned int lowervoltage,`  
`unsigned int uppervoltage )`

**11.109.3.54 SetVoltageValvesLimit()** `void SetVoltageValvesLimit (`  
`unsigned int lowervoltage,`  
`unsigned int uppervoltage )`

**11.109.3.55 StartQueue()** `void StartQueue (`  
    `bool start )`

**11.109.3.56 StopMovement() [1/2]** `void StopMovement (`  
    `unsigned char busaddress,`  
    `char axes )`

**11.109.3.57 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) |

**11.109.3.58 WaitTimer()** `void WaitTimer (`  
    `uint32_t waittime,`  
    `int timeout )`

## 11.109.4 Member Data Documentation

**11.109.4.1 Axes\_I** `const char Axes_I = 2 [static]`

Bit pattern for i axis for axes argument

**11.109.4.2 Axes\_X** `const char Axes_X = 1 [static]`

Bit pattern for x axis for axes argument



**11.109.4.3 Axes\_Y** `const char Axes_Y = 2 [static]`

Bit pattern for y axis for axes argument

**11.109.4.4 Axes\_Z** `const char Axes_Z = 1 [static]`

Bit pattern for z axis for axes argument

**11.109.4.5 Axis\_I** `const byte Axis_I = 1 [static]`

Axis number of i for axis argument

**11.109.4.6 Axis\_X** `const byte Axis_X = 0 [static]`

Axis number of x for axis argument

**11.109.4.7 Axis\_Y** `const byte Axis_Y = 1 [static]`

Axis number of y for axis argument

**11.109.4.8 Axis\_Z** `const byte Axis_Z = 0 [static]`

Axis number of z for axis argument

**11.109.4.9 McsBus\_XY** `const byte McsBus_XY = 1 [static]`

McsBus address for the xy-plane

**11.109.4.10 McsBus\_ZI** `const byte McsBus_ZI = 2 [static]`

McsBus address for the z and i axes

**11.109.4.11 RoboError\_AnotherMaster** `const uint32_t RoboError_AnotherMaster = ( (0xA0110000L) | 0x0005 ) [static]`

**11.109.4.12 RoboError\_Base** `const uint32_t RoboError_Base = (0xA0110000L) [static]`

**11.109.4.13 RoboError\_CannotEscapeEndSwitch** `const uint32_t RoboError_CannotEscapeEndSwitch = ( (0xA0110000L) | 0x0009 ) [static]`

**11.109.4.14 RoboError\_CommandAlreadyInProgress** `const uint32_t RoboError_CommandAlreadyInProgress = ( (0xA0110000L) | 0x000A ) [static]`

**11.109.4.15 RoboError\_CommandNotPossible** `const uint32_t RoboError_CommandNotPossible = ( (0xA0110000L) | 0x0014 ) [static]`

**11.109.4.16 RoboError\_CommunicationTimeout** `const uint32_t RoboError_CommunicationTimeout = ( (0xA0110000L) | 0x0004 ) [static]`

**11.109.4.17 RoboError\_DacqNotReady** `const uint32_t RoboError_DacqNotReady = ( (0xA0110000L) | 0x0015 ) [static]`

**11.109.4.18 RoboError\_DLLMovementTimeout** `const uint32_t RoboError_DLLMovementTimeout = ( (0xA0110000L) | 0x1001 ) [static]`

**11.109.4.19 RoboError\_FindReferenceMethod** `const uint32_t RoboError_FindReferenceMethod = ( (0xA0110000L) | 0x0006 ) [static]`

**11.109.4.20 RoboError\_GilsonCommandPending** `const uint32_t RoboError_GilsonCommandPending = ( (0xA0110000L) | 0x0011 ) [static]`

**11.109.4.21 RoboError\_GilsonTimeout** `const uint32_t RoboError_GilsonTimeout = ( (0xA0110000L) | 0x000F ) [static]`

**11.109.4.22 RoboError\_GilsonWrondID** `const uint32_t RoboError_GilsonWrondID = ( (0xA0110000L) | 0x0010 ) [static]`

**11.109.4.23 RoboError\_McsBus\_UnknownCommand** `const uint32_t RoboError_McsBus_UnknownCommand = ( (0xA0110000L) | 0x003F ) [static]`

**11.109.4.24 RoboError\_NoEndSwitch** `const uint32_t RoboError_NoEndSwitch = ( (0xA0110000L) | 0x0008 ) [static]`

**11.109.4.25 RoboError\_NoMoreData** `const uint32_t RoboError_NoMoreData = ( (0xA0110000L) | 0x0016 ) [static]`

**11.109.4.26 RoboError\_NoReference** `const uint32_t RoboError_NoReference = ( (0xA0110000L) | 0x000B ) [static]`

**11.109.4.27 RoboError\_NoSpeedOrAcceleration** `const uint32_t RoboError_NoSpeedOrAcceleration = ( (0xA0110000L) | 0x0007 ) [static]`

**11.109.4.28 RoboError\_OverPressure** `const uint32_t RoboError_OverPressure = ( (0xA0110000L) | 0x000C ) [static]`

**11.109.4.29 RoboError\_ParameterNotAllowed** `const uint32_t RoboError_ParameterNotAllowed = ( (0xA0110000L) | 0x0012 ) [static]`

**11.109.4.30 RoboError\_PeristalticTimeout** `const uint32_t RoboError_PeristalticTimeout = ( (0xA0110000L) | 0x000E ) [static]`

**11.109.4.31 RoboError\_Phase0OutOfRange** `const uint32_t RoboError_Phase0OutOfRange = ( (0x←A0110000L) | 0x000D ) [static]`

**11.109.4.32 RoboError\_PollLoopCanceled** `const uint32_t RoboError_PollLoopCanceled = ( (0x←A0110000L) | 0x1002) [static]`

**11.109.4.33 RoboError\_PollLoopCanceledAndStopMovement** `const uint32_t RoboError_PollLoop←CanceledAndStopMovement = ( (0xA0110000L) | 0x1003) [static]`

**11.109.4.34 RoboError\_Pressure** `const uint32_t RoboError_Pressure = ( (0xA0110000L) | 0x0002 ) [static]`

**11.109.4.35 RoboError\_RangeExceeded** `const uint32_t RoboError_RangeExceeded = ( (0xA0110000L) | 0x0003 ) [static]`

**11.109.4.36 RoboError\_StateChangeNotPossible** `const uint32_t RoboError_StateChangeNotPossible = ( (0xA0110000L) | 0x0013 ) [static]`

**11.109.4.37 RoboError\_Timeout** `const uint32_t RoboError_Timeout = ( (0xA0110000L) | 0x0001 ) [static]`

**11.109.4.38 RoboError\_UnknownCommand** `const uint32_t RoboError_UnknownCommand = ( (0x←A0110000L) ) [static]`

## 11.109.5 Property Documentation

**11.109.5.1 McsBus** `CMcsBusNet^ McsBus [get]`

**11.109.5.2 McsBus\_MotorControl** [CMcsBus\\_MotorControlNet](#)<sup>^</sup> McsBus\_MotorControl [get]

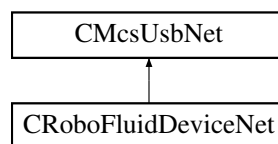
**11.109.5.3 RoboMainLowLevelCommand** [RoboMainLowLevelCommands](#)<sup>^</sup> RoboMainLowLevelCommand [get]

## 11.109.6 Event Documentation

**11.109.6.1 RoboStatusEvent** [RoboStatusEventDelegate](#)<sup>^</sup> RoboStatusEvent

## 11.110 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](#)<sup>^</sup> [m\\_pMcsBus\\_MotorControlNet](#)

## Properties

- [CMcsBus\\_MotorControlNet](#)<sup>^</sup> [McsBus\\_MotorControl](#) [get]

## Additional Inherited Members

### 11.110.1 Constructor & Destructor Documentation

**11.110.1.1 CRoboFluidDeviceNet()** [CRoboFluidDeviceNet](#) (  
void )

**11.110.1.2 ~CRoboFluidDeviceNet()** [~CRoboFluidDeviceNet](#) (  
void )

### 11.110.2 Member Function Documentation

**11.110.2.1 CloseAllValves()** void CloseAllValves ( )

**11.110.2.2 GetPumpSpeed()** short GetPumpSpeed (  
int *index* )

**11.110.2.3 GetSingleValve()** bool GetSingleValve (  
int *valve* )

Query the state of a valve.

#### Parameters

|              |                     |
|--------------|---------------------|
| <i>valve</i> | number of valve /*! |
|--------------|---------------------|

#### Returns

state of the valve

**11.110.2.4 GetValve()** `int GetValve ( )`

Query the state of the values.

**Returns**

the current state of the valves as a bit pattern.

**11.110.2.5 IsPumpMotorOn()** `bool IsPumpMotorOn (`  
`int index )`**11.110.2.6 PumpOff()** `void PumpOff (`  
`int index )`**11.110.2.7 PumpOn()** `void PumpOn (`  
`int index,`  
`short speed )`**11.110.2.8 SetPumpSpeed()** `void SetPumpSpeed (`  
`int index,`  
`short speed )`**11.110.2.9 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           |

**11.110.2.10 SetValve()** `void SetValve (`  
`int value )`

Open or Close valves.

## Parameters

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

## 11.110.3 Member Data Documentation

**11.110.3.1** `m_pMcsBus_MotorControlNet` [CMcsBus\\_MotorControlNet](#) ^ m\_pMcsBus\_MotorControlNet  
[protected]

**11.110.3.2** `m_pRoboFluidDevice` [CRoboFluidDevice\\*](#) m\_pRoboFluidDevice [protected]

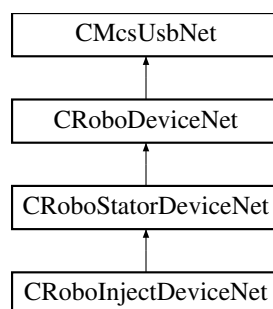
## 11.110.4 Property Documentation

**11.110.4.1** `McsBus_MotorControl` [CMcsBus\\_MotorControlNet](#) ^ [McsBus\\_MotorControl](#) [get]

## 11.111 CRoboInjectDeviceNet Class Reference

[CRoboInjectDeviceNet](#) is the to control the MCS RoboInject device

Inheritance diagram for CRoboInjectDeviceNet:



## Public Member Functions

- [CRoboInjectDeviceNet](#) (void)



## Additional Inherited Members

### 11.111.1 Detailed Description

[CRobolInjectDeviceNet](#) is the to control the MCS RobolInject device

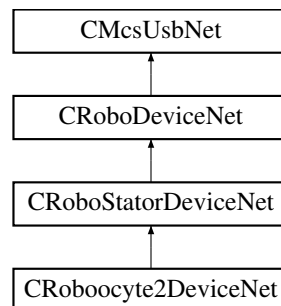
### 11.111.2 Constructor & Destructor Documentation

**11.111.2.1 CRobolInjectDeviceNet()** [CRobolInjectDeviceNet](#) (  
void )

## 11.112 CRoboocyte2DeviceNet Class Reference

[CRoboocyte2DeviceNet](#) is the class to control the MCS Roboocyte2 device

Inheritance diagram for CRoboocyte2DeviceNet:



## Public Member Functions

- [CRoboocyte2DeviceNet](#) (void)
- void [SetAxisLED](#) (bool onoff)
- bool [GetAxisLED](#) ()
- [CRoboDacqNet](#) ^ [GetRoboDacq](#) ()
- [CRoboFluidDeviceNet](#) ^ [GetRoboFluidDevice](#) ()
- [CGilsonDeviceNet](#) ^ [GetGilsonDevice](#) ()
- [CMcsBus\\_ExtensionNet](#) ^ [GetMcsBus\\_Extension](#) ()

## Additional Inherited Members

### 11.112.1 Detailed Description

[CRoboocyte2DeviceNet](#) is the class to control the MCS Roboocyte2 device

### 11.112.2 Constructor & Destructor Documentation

**11.112.2.1 CRoboocyte2DeviceNet()** `CRoboocyte2DeviceNet ( void )`

### 11.112.3 Member Function Documentation

**11.112.3.1 GetAxisLED()** `bool GetAxisLED ( )`

**11.112.3.2 GetGilsonDevice()** `CGilsonDeviceNet ^ GetGilsonDevice ( )`

**11.112.3.3 GetMcsBus\_Extension()** `CMcsBus_ExtensionNet ^ GetMcsBus_Extension ( )`

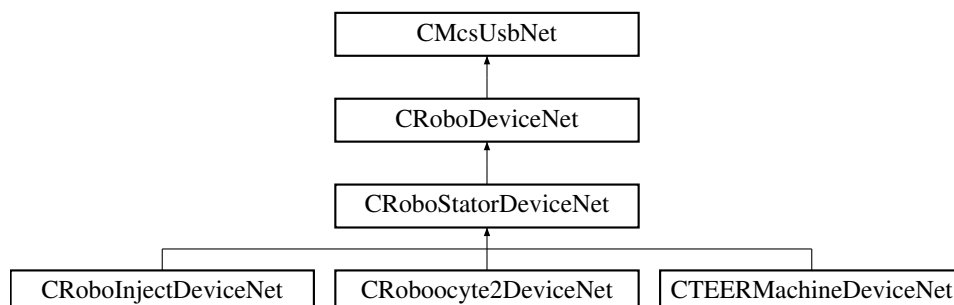
**11.112.3.4 GetRoboDacq()** `CRoboDacqNet ^ GetRoboDacq ( )`

**11.112.3.5 GetRoboFluidDevice()** `CRoboFluidDeviceNet ^ GetRoboFluidDevice ( )`

**11.112.3.6 SetAxisLED()** `void SetAxisLED ( bool onoff )`

## 11.113 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

### 11.113.1 Constructor & Destructor Documentation

**11.113.1.1 CRoboStatorDeviceNet()** `CRoboStatorDeviceNet ( void )`

### 11.113.2 Member Function Documentation

**11.113.2.1 FindReferenceI()** [1/2] `void FindReferenceI ( )`

**11.113.2.2 FindReferenceI()** [2/2] `void FindReferenceI ( int timeout )`

**11.113.2.3 FindReferenceXY()** [1/2] `void FindReferenceXY ( )`

**11.113.2.4 FindReferenceXY()** [2/2] `void FindReferenceXY ( int timeout )`

**11.113.2.5 FindReferenceZ()** [1/2] `void FindReferenceZ ( )`

**11.113.2.6 FindReferenceZ()** [2/2] `void FindReferenceZ ( int timeout )`

**11.113.2.7 GetCurrentPositionI()** `void GetCurrentPositionI ( [System::Runtime::InteropServices::Out] int% i )`

**11.113.2.8 GetCurrentPositionXY()** void GetCurrentPositionXY (   
 [System::Runtime::InteropServices::Out] int% x,   
 [System::Runtime::InteropServices::Out] int% y )

**11.113.2.9 GetCurrentPositionZ()** void GetCurrentPositionZ (   
 [System::Runtime::InteropServices::Out] int% z )

**11.113.2.10 HasRefI()** unsigned char HasRefI ( )

**11.113.2.11 HasRefXY()** unsigned char HasRefXY ( )

**11.113.2.12 HasRefZ()** unsigned char HasRefZ ( )

**11.113.2.13 MoveAbsI()** [1/2] void MoveAbsI (   
 int i )

**11.113.2.14 MoveAbsI()** [2/2] void MoveAbsI (   
 int i,   
 int timeout )

**11.113.2.15 MoveAbsXY()** [1/2] void MoveAbsXY (   
 int x,   
 int y )

**11.113.2.16 MoveAbsXY()** [2/2] void MoveAbsXY (   
 int x,   
 int y,   
 int timeout )

**11.113.2.17 MoveAbsZ()** [1/2] void MoveAbsZ (  
int z )

**11.113.2.18 MoveAbsZ()** [2/2] void MoveAbsZ (  
int z,  
int timeout )

**11.113.2.19 SetAccelerationI()** void SetAccelerationI (  
int a )

**11.113.2.20 SetAccelerationNativeI()** void SetAccelerationNativeI (  
int a )

**11.113.2.21 SetAccelerationNativeXY()** void SetAccelerationNativeXY (  
int a )

**11.113.2.22 SetAccelerationNativeZ()** void SetAccelerationNativeZ (  
int a )

**11.113.2.23 SetAccelerationXY()** void SetAccelerationXY (  
int a )

**11.113.2.24 SetAccelerationZ()** void SetAccelerationZ (  
int a )

**11.113.2.25 SetCurrentAndAirXY()** [1/2] void SetCurrentAndAirXY (  
unsigned short onoff )

**11.113.2.26 SetCurrentAndAirXY()** [2/2] void SetCurrentAndAirXY (  
    unsigned short *onoff*,  
    int *timeout* )

**11.113.2.27 SetSpeedI()** void SetSpeedI (  
    int *v* )

**11.113.2.28 SetSpeedNativeI()** void SetSpeedNativeI (  
    int *v* )

**11.113.2.29 SetSpeedNativeXY()** void SetSpeedNativeXY (  
    int *v* )

**11.113.2.30 SetSpeedNativeZ()** void SetSpeedNativeZ (  
    int *v* )

**11.113.2.31 SetSpeedXY()** void SetSpeedXY (  
    int *v* )

**11.113.2.32 SetSpeedZ()** void SetSpeedZ (  
    int *v* )

**11.113.2.33 SetVelocityI()** void SetVelocityI (  
    int *v* )

**11.113.2.34 SetVelocityXY()** void SetVelocityXY (  
    int *v* )

**11.113.2.35 SetVelocityZ()** `void SetVelocityZ (`  
    `int v )`

**11.113.2.36 StopMovementI()** [1/2] `void StopMovementI ( )`

**11.113.2.37 StopMovementI()** [2/2] `void StopMovementI (`  
    `int timeout )`

**11.113.2.38 StopMovementXY()** [1/2] `void StopMovementXY ( )`

**11.113.2.39 StopMovementXY()** [2/2] `void StopMovementXY (`  
    `int timeout )`

**11.113.2.40 StopMovementZ()** [1/2] `void StopMovementZ ( )`

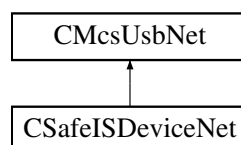
**11.113.2.41 StopMovementZ()** [2/2] `void StopMovementZ (`  
    `int timeout )`

### 11.113.3 Property Documentation

**11.113.3.1 RoboMainStatorLowLevelCommand** `RoboMainStatorLowLevelCommands^ RoboMainStator←`  
`LowLevelCommand [get]`

## 11.114 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](#)^ [RoboDevice](#) [get]  
*Gets the [CRoboDeviceNet](#). Use this to control the syringe.*
- [CFluidControlDeviceNet](#)^ [FluidControlDevice](#) [get]  
*Gets the [CFluidControlDeviceNet](#). Use this to control the valves. Only SetSingleValve is implemented for [CSafeISDeviceNet](#).*
- [CMcsUsbDacqNet](#)^ [DacqDevice](#) [get]  
*Gets the [CMcsUsbDacqNet](#). Use this to control the data aquisition.*

## Additional Inherited Members

### 11.114.1 Detailed Description

### 11.114.2 Constructor & Destructor Documentation

#### 11.114.2.1 CSafeISDeviceNet() [CSafeISDeviceNet](#) (void )

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

#### 11.114.2.2 ~CSafeISDeviceNet() [~CSafeISDeviceNet](#) (void )

Releases unmanaged resources and performs other cleanup operations before the [CSafeISDeviceNet](#) is reclaimed by garbage collection.

### 11.114.3 Member Function Documentation

**11.114.3.1 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 |
|-----------------|------------------------------------------------------------------------------|

**11.114.3.2 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. |
|------------------|----------------------------------|

**11.114.3.3 SetDacMode()** `void SetDacMode (`  
`unsigned char mode )`

Sets the DAC mode.

#### Parameters

|             |                                           |
|-------------|-------------------------------------------|
| <i>mode</i> | The mode: 0 = Impedance ; 1 = Amperometry |
|-------------|-------------------------------------------|

**11.114.3.4 SetDacPeriode()** `void SetDacPeriode (`  
`unsigned int periode )`

Sets the DAC periode.

#### Parameters

|                |                               |
|----------------|-------------------------------|
| <i>periode</i> | The periode in units of 10µs. |
|----------------|-------------------------------|

**11.114.3.5 SetDacPulseform()** `void SetDacPulseform (`  
`array< short >^ pulseform )`

Sets the DAC pulseform.

#### Parameters

|                                  |                |
|----------------------------------|----------------|
| <i><a href="#">pulseform</a></i> | The pulseform. |
|----------------------------------|----------------|

**11.114.3.6 SetSwitches()** `void SetSwitches (`  
`unsigned short switches )`

Sets the switches for all electrodes on the device. Do not use during measurement

#### Parameters

|                                 |                                              |
|---------------------------------|----------------------------------------------|
| <i><a href="#">switches</a></i> | The switches: See Schematics for the meaning |
|---------------------------------|----------------------------------------------|

### 11.114.4 Property Documentation

**11.114.4.1 DacqDevice** [CMcsUsbDacqNet](#)^ DacqDevice [get]

Gets the [CMcsUsbDacqNet](#). Use this to control the data aquisition.

**11.114.4.2 FluidControlDevice** [CFluidControlDeviceNet](#)^ FluidControlDevice [get]

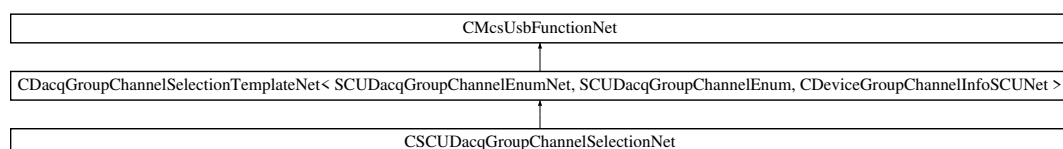
Gets the [CFluidControlDeviceNet](#). Use this to control the valves. Only SetSingleValve is implemented for [CSafeISDeviceNet](#).

**11.114.4.3 RoboDevice** [CRoboDeviceNet](#)^ RoboDevice [get]

Gets the [CRoboDeviceNet](#). Use this to control the syringe.

## 11.115 CSCUDacqGroupChannelSelectionNet Class Reference

Inheritance diagram for CSCUDacqGroupChannelSelectionNet:



## Public Member Functions

- [CSCUDacqGroupChannelSelectionNet](#) ([CMcsUsbNet](#)^ mcsusb)

## Additional Inherited Members

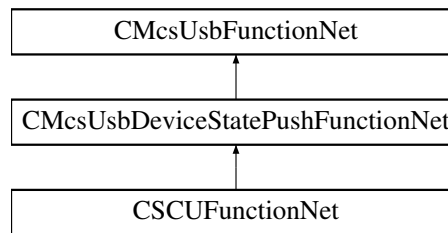
### 11.115.1 Constructor & Destructor Documentation

#### 11.115.1.1 CSCUDacqGroupChannelSelectionNet() [CSCUDacqGroupChannelSelectionNet](#) ([CMcsUsbNet](#)^ mcsusb )

## 11.116 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\\_HeadstageEnumNet](#) [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.*
- void [SetHeadstageNumberOfAnalogChannelsPermanent](#) (uint32\_t Headstage, uint32\_t NumberOfchannels)  
*Sets the number of analog channels permanent 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.*
- void [SetHeadstageSampleratePermanent](#) (uint32\_t Headstage, uint32\_t Samplerate)  
*Sets the samplerate permanent on a given headstage.*
- uint32\_t [GetHeadstageLinkSpeed](#) (uint32\_t Headstage)  
*Gets the Link speed of a given headstage.*
- void [SetHeadstageLinkSpeedPermanent](#) (uint32\_t Headstage, uint32\_t LinkSpeed)  
*Sets the Link speed permanent on a given headstage.*
- uint32\_t [GetHeadstageFrameCyclesToCompare](#) (uint32\_t Headstage)  
*Gets the frame cycles to compare of a given headstage.*
- void [SetHeadstageFrameCyclesToComparePermanent](#) (uint32\_t Headstage, uint32\_t FrameCycles)  
*Sets the frame cycles to compare permanent on 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*
- void [SetDockSmartMultiwellPlatePowerstate](#) (uint32\_t Dock, uint32\_t SMWP, bool PowerIsOn)  
*Sets the power of the selected smart multiwell plate in the selected dock*
- bool [GetDockSmartMultiwellPlatePowerstate](#) (uint32\_t Dock, uint32\_t SMWP)  
*Gets the power of the selected smart multiwell plate in the selected dock*
- uint32\_t [GetAvailableDockSmartMultiwellPlates](#) (uint32\_t Dock)  
*Gets a bitmap of available smart multiwell plates in a dock.*
- void [SetDockSmartMultiwellPlateWellMux](#) (uint32\_t Dock, uint32\_t SMWP, int32\_t WellMux)  
*Sets the well multiplexer of the selected smart multiwell plate in the selected dock*
- int32\_t [GetDockSmartMultiwellPlateWellMux](#) (uint32\_t Dock, uint32\_t SMWP)  
*Gets the well multiplexer of the selected smart multiwell plate in the selected dock*

## 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

### 11.116.1 Detailed Description

[CSCUFunctionNet](#) is the class to control the SCU device

### 11.116.2 Constructor & Destructor Documentation

**11.116.2.1 CSCUFunctionNet()** [1/2] `CSCUFunctionNet (   
 CMcsUsbNet^ mcsusb,   
 CMcsUsbFunctionPointerContainer^ pSCUFunctionPointerContainer )`

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

**11.116.2.2 CSCUFunctionNet()** [2/2] `CSCUFunctionNet (   
 CMcsUsbNet^ mcsusb )`

**11.116.2.3 ~CSCUFunctionNet()** `virtual ~CSCUFunctionNet ( ) [virtual]`

**11.116.2.4 !CSCUFunctionNet()** `!CSCUFunctionNet ( )`

### 11.116.3 Member Function Documentation

**11.116.3.1 AutomaticAnalogOut()** `void AutomaticAnalogOut (   
 bool automatic )`

Sets automatic source channel selection

#### Parameters

|                  |           |
|------------------|-----------|
| <i>automatic</i> | Automatic |
|------------------|-----------|

**11.116.3.2 EnableAnalogOut()** `void EnableAnalogOut (`

```
bool enable)
```

Enables AnalogOut globally

#### Parameters

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

#### 11.116.3.3 GetAnalogOutADCRange() `uint32_t GetAnalogOutADCRange ( )`

Gets the analog out ADC range

#### Returns

Range

#### 11.116.3.4 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

#### 11.116.3.5 GetAnalogOutDACRange() `AnalogOut_DAC_Range_EnumNet GetAnalogOutDACRange ( )`

Gets the analog out DAC range

#### Returns

Range

#### 11.116.3.6 GetAvailableDockSmartMultiwellPlates() `uint32_t GetAvailableDockSmartMultiwellPlates ( uint32_t Dock )`

Gets a bitmap of available smart multiwell plates in a dock.



## Parameters

|             |                 |
|-------------|-----------------|
| <i>Dock</i> | select the dock |
|-------------|-----------------|

## Returns

The bitmap of available smart multiwell plates.

**11.116.3.7 GetAvailableHeadstages()** `uint32_t GetAvailableHeadstages ( )`

Gets a bitmap of available headstages.

## Returns

The bitmap of available headstages.

**11.116.3.8 GetDockSmartMultiwellPlatePowerstate()** `bool GetDockSmartMultiwellPlatePowerstate (   
uint32_t Dock,   
uint32_t SMWP )`

Gets the power of the selected smart multiwell plate in the selected dock

## Parameters

|             |                                  |
|-------------|----------------------------------|
| <i>Dock</i> | select the dock                  |
| <i>SMWP</i> | select the smart multiwell plate |

## Returns

returns the powerstate of selected smart multiwellplate in selected dock

**11.116.3.9 GetDockSmartMultiwellPlateWellMux()** `int32_t GetDockSmartMultiwellPlateWellMux (   
uint32_t Dock,   
uint32_t SMWP )`

Gets the well multiplexer of the selected smart multiwell plate in the selected dock

## Parameters

|             |                                  |
|-------------|----------------------------------|
| <i>Dock</i> | select the dock                  |
| <i>SMWP</i> | select the smart multiwell plate |

**Returns**

returns the selected well of selected smart multiwellplate in selected dock, -1 means Off

**11.116.3.10 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

**11.116.3.11 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

**11.116.3.12 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.

**11.116.3.13 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.

**11.116.3.14 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.

**11.116.3.15 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.

**11.116.3.16 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.

**11.116.3.17 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.

**11.116.3.18 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.

**11.116.3.19 GetHeadstageFrameCyclesToCompare()** `uint32_t GetHeadstageFrameCyclesToCompare ( uint32_t Headstage )`

Gets the frame cycles to compare of a given headstage.

**Parameters**

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

**Returns**

The samplerate in Hz for the given headstage.

**11.116.3.20 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.

**11.116.3.21 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.

**11.116.3.22 GetHeadstageLinkSpeed()** `uint32_t GetHeadstageLinkSpeed (`  
`uint32_t Headstage )`

Gets the Link speed of a given headstage.

**Parameters**

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

**Returns**

The samplerate in Hz for the given headstage.

**11.116.3.23 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.

**11.116.3.24 GetHeadstageNumberOfStimulationChannels()** `uint32_t GetHeadstageNumberOfStimulation↵`  
`Channels (`  
`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.

**11.116.3.25 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.

**11.116.3.26 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.

**11.116.3.27 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.

**11.116.3.28 GetMaxNumberOfHeadstages()** `uint32_t GetMaxNumberOfHeadstages ( )`

Gets the maximal number of headstages.

**Returns**

The maximal number of headstages.

**11.116.3.29 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.

**11.116.3.30 GetReferenceElectrodeMode()** `ReferenceElectrodeModeEnumNet GetReferenceElectrodeMode ( uint32_t Headstage )`

Gets the mode for the reference electrode

**Parameters**

|                  |                      |
|------------------|----------------------|
| <i>Headstage</i> | The headstage number |
|------------------|----------------------|

**Returns**

The mode

**11.116.3.31 GetReferenceElectrodeSwitchState()** `ReferenceElectrodeSwitchPositionEnumNet GetReferenceElectrodeSwitchState ( uint32_t Headstage )`

Gets the position of the switch for the reference electrode

**Parameters**

|                  |                      |
|------------------|----------------------|
| <i>Headstage</i> | The headstage number |
|------------------|----------------------|

**Returns**

The switch position



**11.116.3.32 HasAnalogOut()** `bool HasAnalogOut ( )`

Has AnalogOut hardware

**Returns**

Enabled

**11.116.3.33 HasGalvanicIsolation()** `bool HasGalvanicIsolation ( )`

Has galvanic isolated hardware

**Returns**

Enabled

**11.116.3.34 HasHSPowerSwitch()** `bool HasHSPowerSwitch ( )`

Has SCU HS power switch

**Returns**

Has Switch

**11.116.3.35 IsAnalogOutEnabled()** `bool IsAnalogOutEnabled ( )`

Is AnalogOut enabled

**Returns**

Enabled

**11.116.3.36 IsAutomaticAnalogOut()** `bool IsAutomaticAnalogOut ( )`

Is Automatic source channel selection selected

**Returns**

Automatic

**11.116.3.37 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.

**11.116.3.38 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.

**11.116.3.39 IsInDacqLegacyMode()** `bool IsInDacqLegacyMode ( )`

Is the SCU in legacy mode

**Returns**

Is Enabled

**11.116.3.40 OnGetAvailableHeadstages()** `delegate void OnGetAvailableHeadstages (`  
    `uint32_t AvailableHeadstages )`

**11.116.3.41 OnIsHeadstageAvailable()** `delegate void OnIsHeadstageAvailable (`  
    `uint32_t Headstage,`  
    `bool available )`

**11.116.3.42 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. |

**11.116.3.43 SetAnalogOutADCRange()** `void SetAnalogOutADCRange (`  
`uint32_t range )`

Sets the analog out ADC range

## Parameters

|              |       |
|--------------|-------|
| <i>range</i> | Range |
|--------------|-------|

**11.116.3.44 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 |

**11.116.3.45 SetAnalogOutDACRange()** `void SetAnalogOutDACRange (`  
`AnalogOut\_DAC\_Range\_EnumNet range )`

Sets the analog out DAC range

## Parameters

|              |       |
|--------------|-------|
| <i>range</i> | Range |
|--------------|-------|

**11.116.3.46 SetDacqLegacyMode()** `void SetDacqLegacyMode (`  
`bool enable )`

Enable the SCU legacy mode

## Parameters

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

**11.116.3.47 SetDockSmartMultiwellPlatePowerstate()** `void SetDockSmartMultiwellPlatePowerstate (`  
     `uint32_t Dock,`  
     `uint32_t SMWP,`  
     `bool PowerIsOn )`

Sets the power of the selected smart multiwell plate in the selected dock

## Parameters

|                  |                                                              |
|------------------|--------------------------------------------------------------|
| <i>Dock</i>      | select the dock                                              |
| <i>SMWP</i>      | select the smart multiwell plate                             |
| <i>PowerIsOn</i> | powerstate of selected smart multiwellplate in selected dock |

**11.116.3.48 SetDockSmartMultiwellPlateWellMux()** `void SetDockSmartMultiwellPlateWellMux (`  
     `uint32_t Dock,`  
     `uint32_t SMWP,`  
     `int32_t WellMux )`

Sets the well multiplexer of the selected smart multiwell plate in the selected dock

## Parameters

|                |                                                                             |
|----------------|-----------------------------------------------------------------------------|
| <i>Dock</i>    | select the dock                                                             |
| <i>SMWP</i>    | select the smart multiwell plate                                            |
| <i>WellMux</i> | select well of selected smart multiwellplate in selected dock, -1 means Off |

**11.116.3.49 SetHeadstageFrameCyclesToComparePermanent()** `void SetHeadstageFrameCyclesTo↵`  
`ComparePermanent (`  
     `uint32_t Headstage,`  
     `uint32_t FrameCycles )`

Sets the frame cycles to compare permanent on a given headstage.

## Parameters

|                    |                                               |
|--------------------|-----------------------------------------------|
| <i>Headstage</i>   | The headstage number                          |
| <i>FrameCycles</i> | The samplerate in Hz for the given headstage. |

**11.116.3.50 SetHeadstageLinkSpeedPermanent()** `void SetHeadstageLinkSpeedPermanent (`  
     `uint32_t Headstage,`  
     `uint32_t LinkSpeed )`

Sets the Link speed permanent on a given headstage.

**Parameters**

|                  |                                               |
|------------------|-----------------------------------------------|
| <i>Headstage</i> | The headstage number                          |
| <i>LinkSpeed</i> | The samplerate in Hz for the given headstage. |

**11.116.3.51 SetHeadstageNumberOfAnalogChannelsPermanent()** `void SetHeadstageNumberOfAnalog↵`  
`ChannelsPermanent (`  
     `uint32_t Headstage,`  
     `uint32_t NumberOfchannels )`

Sets the number of analog channels permanent for a given headstage.

**Parameters**

|                         |                                                             |
|-------------------------|-------------------------------------------------------------|
| <i>Headstage</i>        | The headstage number                                        |
| <i>NumberOfchannels</i> | The number of analog channels the headstage has to transmit |

**11.116.3.52 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. |

**11.116.3.53 SetHeadstageSampleratePermanent()** `void SetHeadstageSampleratePermanent (`  
     `uint32_t Headstage,`  
     `uint32_t Samplerate )`

Sets the samplerate permanent on a given headstage.

**Parameters**

|                   |                                               |
|-------------------|-----------------------------------------------|
| <i>Headstage</i>  | The headstage number                          |
| <i>Samplerate</i> | The samplerate in Hz for the given headstage. |

**11.116.3.54 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             |

**11.116.3.55 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  |

**11.116.4 Event Documentation**

**11.116.4.1 GetAvailableHeadstagesEvent** `OnGetAvailableHeadstages^` `GetAvailableHeadstagesEvent`  
[add], [remove], [raise]

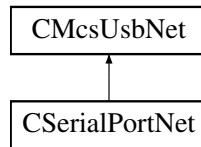
Event fires when the bitmap of available headstages has changed

**11.116.4.2 IsHeadstageAvailableEvent** `OnIsHeadstageAvailable^` `IsHeadstageAvailableEvent` [add],  
[remove], [raise]

Event fires when 'true' if the headstage is connected for the headstage to query has changed

## 11.117 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

#### 11.117.1 Constructor & Destructor Documentation

**11.117.1.1 CSerialPortNet()** [CSerialPortNet](#) (  
void )

#### 11.117.2 Member Function Documentation

**11.117.2.1 GetBytesAvailable()** int [GetBytesAvailable](#) (  
void )

**11.117.2.2 Receive()** [1/2] array<byte> ^ [Receive](#) (  
int length )

**11.117.2.3 Receive()** [2/2] array<byte> ^ [Receive](#) (  
void )

**11.117.2.4 ReceiveString()** [1/2] `String ^ ReceiveString (`  
`int length )`

**11.117.2.5 ReceiveString()** [2/2] `String ^ ReceiveString (`  
`void )`

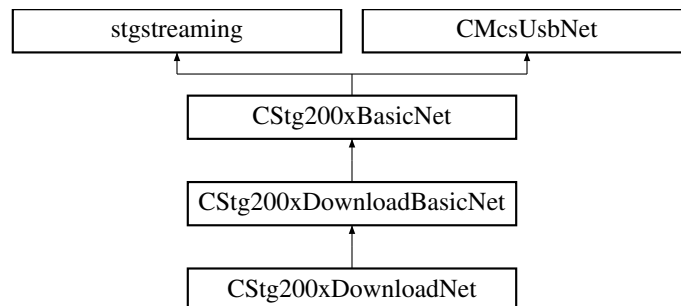
**11.117.2.6 Send()** [1/2] `void Send (`  
`array< byte >^ buffer )`

**11.117.2.7 Send()** [2/2] `void Send (`  
`String^ command )`

## 11.118 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.*
- virtual int32\_t [GetDACResolution](#) ()  
*Gets number of bits of the DAC resolution.*
- virtual int32\_t [GetTimeResolutionInNanoSeconds](#) ()  
*Gets the time resolution of the STG.*
- 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.*
- virtual int32\_t [GetCurrentResolutionInPicoAmp](#) (uint32\_t channel)  
*Gets the Current Resolution of the specified channel in Picoamps.*
- 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 System::Collections::Generic::List< int32\_t > ^ [GetVoltageRangeListInMicroVolt](#) (uint32\_t channel)  
*Gets a list of current ranges supported by the device (STG5 only).*

- virtual uint32\_t [GetNumberOfVoltageRangeIndexes](#) ()  
*Gets the number of voltage ranges (STG5 only).*
- virtual uint32\_t [GetVoltageRangeInMicroVoltByIndex](#) (uint32\_t channel, uint32\_t index)  
*Gets the voltage range for the given channel and index (STG5 only).*
- virtual uint32\_t [GetVoltageRangeSelectedIndex](#) (uint32\_t channel)  
*Gets the currently selected range index for the voltage output (not used yet).*
- virtual void [SetVoltageRangeSelectedIndex](#) (uint32\_t channel, uint32\_t rangeIndex)  
*Sets the range index for the voltage output (not used yet).*
- virtual System::Collections::Generic::List< int32\_t > ^ [GetCurrentRangeListInNanoAmp](#) (uint32\_t channel)  
*Gets a list of current ranges supported by the device (STG5 only).*
- virtual uint32\_t [GetNumberOfCurrentRangeIndexes](#) ()  
*Gets the number of current ranges (STG5 only).*
- virtual uint32\_t [GetCurrentRangeInNanoAmpByIndex](#) (uint32\_t channel, uint32\_t index)  
*Gets the current range for the given channel and index (STG5 only).*
- virtual uint32\_t [GetCurrentRangeSelectedIndex](#) (uint32\_t channel)  
*Gets the currently selected range index for the current output (STG5 only).*
- virtual void [SetCurrentRangeSelectedIndex](#) (uint32\_t channel, uint32\_t rangeIndex)  
*Sets the range index for the current output (STG5 only).*
- virtual void [SetMeasurementMode](#) (unsigned int channel)  
*Sets a channel to measurement mode (STG3008-FA).*
- virtual void [SetFAAmplification](#) (Stg3008FilterAmpAmplificationEnumNet amplification)  
*Sets the amplification of the filter amplifier (STG3008-FA).*
- virtual [Stg3008FilterAmpAmplificationEnumNet](#) [GetFAAmplification](#) ()  
*Gets the currently selected amplification of the filter amplifier (STG3008-FA).*
- bool [GetCanDisableCalibration](#) (uint32\_t channel)  
*Gets whether the channel can disable autocalibration.*
- bool [GetCanDisableCurrentswitch](#) (uint32\_t channel)  
*Gets whether the channel can disable its current switch.*
- bool [GetHasLowCurrentswitchLatency](#) (uint32\_t channel)  
*Gets whether the channel has a low latency current switch.*
- virtual void [SetAutocalibrationDisabled](#) (unsigned int channel, bool disable)  
*Sets the autocalibration configuration.*
- virtual bool [GetAutocalibrationDisabled](#) (unsigned int channel)  
*Gets the autocalibration configuration.*
- virtual void [SetElectrodeMode](#) (uint32\_t electrode, array< [ElectrodeModeEnumNet](#) >^ mode)  
*Puts an electrode in either automatic or manual mode.*
- virtual void [SetElectrodeMode](#) (uint32\_t electrode, [ElectrodeModeEnumNet](#) mode)  
*Puts an electrode in either automatic or manual mode.*
- virtual void [SetElectrodeMode](#) (uint32\_t Scu\_HS, uint32\_t electrode, array< [ElectrodeModeEnumNet](#) >^ mode)  
*Puts an electrode in either automatic or manual mode.*
- virtual void [SetElectrodeMode](#) (uint32\_t Scu\_HS, uint32\_t electrode, [ElectrodeModeEnumNet](#) mode)  
*Puts an electrode in either automatic or manual mode.*
- virtual uint32\_t [GetElectrodeMode](#) (uint32\_t electrode)  
*Gets the mode an electrode is in.*
- virtual uint32\_t [GetElectrodeMode](#) (uint32\_t Scu\_HS, uint32\_t electrode)  
*Gets the mode an electrode is in.*
- virtual void [SetElectrodeDacMux](#) (uint32\_t electrode, uint32\_t listmodelIndex, array< [ElectrodeDacMuxEnumNet](#) >^ dacMux)  
*Defines the DAC to use for an electrode.*

- virtual void [SetElectrodeDacMux](#) (uint32\_t electrode, uint32\_t listmodelIndex, [ElectrodeDacMuxEnumNet](#) dacMux)  
*Defines the DAC to use for an electrode.*
- virtual void [SetElectrodeDacMux](#) (uint32\_t Scu\_HS, uint32\_t electrode, uint32\_t listmodelIndex, [ElectrodeDacMuxEnumNet](#) dacMux)  
*Defines the DAC to use for an electrode.*
- virtual void [SetElectrodeDacMux](#) (uint32\_t Scu\_HS, uint32\_t electrode, uint32\_t listmodelIndex, array< [ElectrodeDacMuxEnumNet](#) >^ dacMux)  
*Defines the DAC to use for an electrode.*
- virtual [ElectrodeDacMuxEnumNet](#) [GetElectrodeDacMux](#) (uint32\_t electrode, uint32\_t listmodelIndex)  
*Gets the DAC which is used for an electrode.*
- virtual [ElectrodeDacMuxEnumNet](#) [GetElectrodeDacMux](#) (uint32\_t Scu\_HS, uint32\_t electrode, uint32\_t listmodelIndex)  
*Gets the DAC which is used for an electrode.*
- virtual void [SetElectrodeEnable](#) (uint32\_t electrode, uint32\_t listmodelIndex, array< bool >^ enable)  
*Enables or disables the stimulation switch for an electrode.*
- virtual void [SetElectrodeEnable](#) (uint32\_t electrode, uint32\_t listmodelIndex, bool enable)  
*Enables or disables the stimulation switch for an electrode.*
- virtual void [SetElectrodeEnable](#) (uint32\_t Scu\_HS, uint32\_t electrode, uint32\_t listmodelIndex, bool enable)  
*Enables or disables the stimulation switch for an electrode.*
- virtual void [SetElectrodeEnable](#) (uint32\_t Scu\_HS, uint32\_t electrode, uint32\_t listmodelIndex, array< bool >^ enable)  
*Enables or disables the stimulation switch for an electrode.*
- virtual bool [GetElectrodeEnable](#) (uint32\_t electrode, uint32\_t listmodelIndex)  
*Gets whether an electrode is enabled or disabled for stimulation.*
- virtual bool [GetElectrodeEnable](#) (uint32\_t Scu\_HS, uint32\_t electrode, uint32\_t listmodelIndex)  
*Gets whether an electrode is enabled or disabled for stimulation.*
- virtual void [SetExternalElectrodeEnable](#) (uint32\_t electrode, uint32\_t listmodelIndex, array< bool >^ enable)  
*Enables or disables the stimulation switch for an external electrode.*
- virtual void [SetExternalElectrodeEnable](#) (uint32\_t electrode, uint32\_t listmodelIndex, bool enable)  
*Enables or disables the stimulation switch for an external electrode.*
- virtual bool [GetExternalElectrodeEnable](#) (uint32\_t electrode, uint32\_t listmodelIndex)  
*Gets whether an electrode is enabled or disabled for stimulation.*
- virtual void [SetBlankingEnable](#) (uint32\_t electrode, bool enable)  
*Defines whether an electrode should be blanked while stimulation is in progress.*
- virtual void [SetBlankingEnable](#) (uint32\_t electrode, array< bool >^ enable)  
*Defines whether an electrode should be blanked while stimulation is in progress.*
- virtual void [SetBlankingEnable](#) (uint32\_t Scu\_HS, uint32\_t electrode, bool enable)  
*Defines whether an electrode should be blanked while stimulation is in progress.*
- virtual void [SetBlankingEnable](#) (uint32\_t Scu\_HS, uint32\_t electrode, array< bool >^ enable)  
*Defines whether an electrode should be blanked while stimulation is in progress.*
- virtual bool [GetBlankingEnable](#) (uint32\_t electrode)  
*Gets whether an electrode should be blanked while stimulation is in progress.*
- virtual bool [GetBlankingEnable](#) (uint32\_t Scu\_HS, uint32\_t electrode)  
*Gets whether an electrode should be blanked while stimulation is in progress.*
- virtual void [SetEnableAmplifierProtectionSwitch](#) (uint32\_t electrode, bool enable)  
*Defines whether the Amplifier Protection Switch is openend while stimulation is in progress.*
- virtual void [SetEnableAmplifierProtectionSwitch](#) (uint32\_t electrode, array< bool >^ enable)  
*Defines whether the Amplifier Protection Switch is openend while stimulation is in progress.*
- virtual void [SetEnableAmplifierProtectionSwitch](#) (uint32\_t Scu\_HS, uint32\_t electrode, bool enable)  
*Defines whether the Amplifier Protection Switch is openend while stimulation is in progress.*

- virtual void [SetEnableAmplifierProtectionSwitch](#) (uint32\_t Scu\_HS, uint32\_t electrode, array< bool >^ enable)  
*Defines whether the Amplifier Protection Switch is openend while stimulation is in progress.*
- virtual bool [GetEnableAmplifierProtectionSwitch](#) (uint32\_t electrode)  
*Gets whether the Amplifier Protection Switch is openend while stimulation is in progress.*
- virtual bool [GetEnableAmplifierProtectionSwitch](#) (uint32\_t Scu\_HS, uint32\_t electrode)  
*Gets whether the Amplifier Protection Switch is openend while stimulation is in progress.*
- virtual uint32\_t [GetNumberOfStimulationElectrodes](#) ()
- template<typename digitalsourceenum >  
virtual void [SetTriggerSource](#) (unsigned int triggernum, [DigitalSource](#)< digitalsourceenum >^ triggersource, int bitnum\_offset)
- virtual void [SetTriggerSource](#) (unsigned int triggernum, [TriggerSourceEnumNet](#) triggersource, int bitnum\_offset)
- virtual void [SetTriggerSource](#) (unsigned int triggernum, [TriggerSourceEnumNet](#) triggersource)
- virtual [TriggerSourceEnumNet](#) [GetTriggerSource](#) (unsigned int triggernum)
- virtual void [SetListmodeIndexRange](#) (unsigned int electrodeGroup, unsigned int startIndex, unsigned int endIndex, unsigned int mode)  
*Define the range of list mode indexes to use for the given electrode group.*
- virtual void [GetListmodeIndexRange](#) (unsigned int electrodeGroup, unsigned int &startIndex, unsigned int &endIndex, unsigned int &mode)  
*Query the range of list mode indexes to use for the given electrode group.*
- virtual void [SetListmodeTriggerSource](#) (unsigned int electrodeGroup, [TriggerSourceEnumNet](#) triggersource)  
*Define the signal which triggers the transition from one list mode entry to the next. After reaching the last entry in the list, the first entry is selected. For triggersource use the Enum which corresponds to the device in use, for example use SCUDigitalSourceEnumNet and cast to TriggerSourceEnumNet if working with an SCU device.*
- virtual void [SetListmodeTriggerSource](#) (unsigned int electrodeGroup, [TriggerSourceEnumNet](#) triggersource, int bitnumOffset)  
*Define the signal which triggers the transition from one list mode entry to the next. After reaching the last entry in the list, the first entry is selected. For triggersource use the Enum which corresponds to the device in use, for example use SCUDigitalSourceEnumNet and cast to TriggerSourceEnumNet if working with an SCU device.*
- virtual [TriggerSourceEnumNet](#) [GetListmodeTriggerSource](#) (unsigned int electrodeGroup)  
*Query the currently active signal which triggers the transition from one list mode entry to the next. For triggersource use the Enum which corresponds to the device in use, for example use SCUDigitalSourceEnumNet and cast to TriggerSourceEnumNet if working with an SCU device.*
- virtual void [ListModeSendStart](#) (unsigned int electrodeGroupMask)  
*Activate (arm) the Listmode for the selected electrode groups.*
- virtual void [ListModeSendStop](#) (unsigned int electrodeGroupMask)  
*Deactivate the Listmode for the selected electrode groups.*
- 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.*

- virtual bool [HasDigitalPort](#) ()

*Queries if the device has a digital port.*

- virtual bool [HasFilterAmp](#) ()

*Queries if the device has a filter amplifier. Currently the STG3008-FA supports this feature.*

- virtual bool [LegacyHasOutputMap](#) ()

*Queries if the device can map its stimulation channels. Legacy command, do not use.*

## Additional Inherited Members

### 11.118.1 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.

### 11.118.2 Constructor & Destructor Documentation

**11.118.2.1** [~CStg200xBasicNet\(\)](#) virtual [~CStg200xBasicNet](#) ( ) [virtual]

The destructor.

### 11.118.3 Member Function Documentation

**11.118.3.1** [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.

**11.118.3.2 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.

**11.118.3.3 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.

**11.118.3.4 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.

**11.118.3.5 GetCanDisableCalibration()** `bool GetCanDisableCalibration ( uint32_t channel )`

Gets whether the channel can disable autocalibration.

## Parameters

|                |                               |
|----------------|-------------------------------|
| <i>channel</i> | The channel which is queried. |
|----------------|-------------------------------|

## Returns

True if autocalibration can be disabled.

**11.118.3.6 GetCanDisableCurrentswitch()** `bool GetCanDisableCurrentswitch ( uint32_t channel )`

Gets whether the channel can disable its current switch.

## Parameters

|                |                               |
|----------------|-------------------------------|
| <i>channel</i> | The channel which is queried. |
|----------------|-------------------------------|

## Returns

True if the current switch can be disabled.

**11.118.3.7 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.

**11.118.3.8 GetCurrentRangeInNanoAmpByIndex()** `virtual uint32_t GetCurrentRangeInNanoAmpByIndex (`  
`uint32_t channel,`  
`uint32_t index ) [virtual]`

Gets the current range for the given channel and index (STG5 only).

**Parameters**

|                |              |
|----------------|--------------|
| <i>channel</i> | The channel. |
| <i>index</i>   | The channel. |

**Returns**

The current range in uA.

**11.118.3.9 GetCurrentRangeListInNanoAmp()** `virtual System::Collections::Generic::List<int32_t> ^ GetCurrentRangeListInNanoAmp (`  
`uint32_t channel ) [virtual]`

Gets a list of current ranges supported by the device (STG5 only).

**11.118.3.10 GetCurrentRangeSelectedIndex()** `virtual uint32_t GetCurrentRangeSelectedIndex (`  
`uint32_t channel ) [virtual]`

Gets the currently selected range index for the current output (STG5 only).

**Parameters**

|                |                        |
|----------------|------------------------|
| <i>channel</i> | The channel to change. |
|----------------|------------------------|

**Returns**

The currently selected range index.

**11.118.3.11 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.

**11.118.3.12 GetCurrentResolutionInPicoAmp()** `virtual int32_t GetCurrentResolutionInPicoAmp ( uint32_t channel ) [virtual]`

Gets the Current Resolution of the specified channel in Picoamps.

**Parameters**

|                |                           |
|----------------|---------------------------|
| <i>channel</i> | Channel which is queried. |
|----------------|---------------------------|

**Returns**

The Current Resolution of the specified channel in Picoamps.

**11.118.3.13 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.

**11.118.3.14 GetDACResolution()** `virtual int32_t GetDACResolution ( ) [virtual]`

Gets number of bits of the DAC resolution.

**Returns**

The DAC resolution in bits.

**11.118.3.15 GetDiginValue()** `virtual uint32_t GetDiginValue ( ) [virtual]`

Gets the Value on the digital input port.

**Returns**

The current value on the digital inputs.

**11.118.3.16 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.

**11.118.3.17 GetDigoutValue()** `virtual uint32_t GetDigoutValue ( ) [virtual]`

Gets the Value on the digital output port.

**Returns**

The current value on the digital outputs.

**11.118.3.18 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.

**11.118.3.19 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.

**11.118.3.20 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.

**11.118.3.21 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.

**11.118.3.22 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.

**11.118.3.23 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.

**11.118.3.24 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.

**11.118.3.25 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.

**11.118.3.26 GetExternalElectrodeEnable()** virtual bool GetExternalElectrodeEnable (   
     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.

**11.118.3.27 GetFAAmplification()** virtual [Stg3008FilterAmpAmplificationEnumNet](#) GetFAAmplification   
 ( ) [virtual]

Gets the currently selected amplification of the filter amplifier (STG3008-FA).

**Returns**

True the currently selected amplification.

**11.118.3.28 GetHasLowCurrentswitchLatency()** bool GetHasLowCurrentswitchLatency (   
     uint32\_t *channel* )

Gets whether the channel has a low latency current switch.

**Parameters**

|                |                               |
|----------------|-------------------------------|
| <i>channel</i> | The channel which is queried. |
|----------------|-------------------------------|

**Returns**

True if the channel has a low latency current switch.

**11.118.3.29 GetHeadstage()** `virtual uint32_t GetHeadstage ( ) [virtual]`

**11.118.3.30 GetListmodeIndexRange()** `virtual void GetListmodeIndexRange ( unsigned int electrodeGroup, unsigned int & startIndex, unsigned int & endIndex, unsigned int & mode ) [virtual]`

Query the range of list mode indexes to use for the given electrode group.

#### Parameters

|                       |                                                    |
|-----------------------|----------------------------------------------------|
| <i>electrodeGroup</i> | The electrodegroup for which the range is queried. |
|-----------------------|----------------------------------------------------|

#### Parameters

|                   |                                                             |
|-------------------|-------------------------------------------------------------|
| <i>startIndex</i> | The index of the first active element in the listmode list. |
| <i>endIndex</i>   | The index of the last active element in the listmode list.  |
| <i>mode</i>       | 0 for "start with startIndex", 1 for "start with endIndex". |

**11.118.3.31 GetListmodeTriggerSource()** `virtual TriggerSourceEnumNet GetListmodeTriggerSource ( unsigned int electrodeGroup ) [virtual]`

Query the currently active signal which triggers the transition from one list mode entry to the next. For triggersource use the Enum which corresponds to the device in use, for example use SCUDigitalSourceEnumNet and cast to TriggerSourceEnumNet if working with an SCU device.

#### Parameters

|                       |                                                            |
|-----------------------|------------------------------------------------------------|
| <i>electrodeGroup</i> | The electrodegroup for which the triggersource is queried. |
|-----------------------|------------------------------------------------------------|

#### Returns

One of the possible sources for the transition.

**11.118.3.32 GetNumberOfAnalogChannels()** `virtual uint32_t GetNumberOfAnalogChannels ( ) [virtual]`

Gets the Number of available analog channels of the device.

**Returns**

The number of analog channels.

**11.118.3.33 GetNumberOfCurrentRangeIndexes()** `virtual uint32_t GetNumberOfCurrentRangeIndexes ( ) [virtual]`

Gets the number of current ranges (STG5 only).

**Returns**

The number of current ranges available on the device.

**11.118.3.34 GetNumberOfHWDACPaths()** `virtual uint32_t GetNumberOfHWDACPaths ( ) [virtual]`

Gets the Number of HW Stimulation DACs of the device.

**Returns**

The number of independent HW Stimulation outputs.

**11.118.3.35 GetNumberOfStimulationElectrodes()** `virtual uint32_t GetNumberOfStimulationElectrodes ( ) [virtual]`

**11.118.3.36 GetNumberOfStimulationSourcesPerElectrode()** `virtual uint32_t GetNumberOfStimulationSourcesPerElectrode ( ) [virtual]`

Gets the number of stimulation sources (DACs) per electrode.

**Returns**

The number of stimulation sources (DACs) per electrode.

**11.118.3.37 GetNumberOfSyncoutChannels()** `virtual uint32_t GetNumberOfSyncoutChannels ( ) [virtual]`

Gets the Number of available syncout channels of the device.

**Returns**

The number of analog channels.



**11.118.3.38 GetNumberOfTriggerInputs()** `virtual uint32_t GetNumberOfTriggerInputs ( ) [virtual]`

Gets the Number of trigger inputs of the device.

#### Returns

The number of trigger inputs.

**11.118.3.39 GetNumberOfVoltageRangeIndexes()** `virtual uint32_t GetNumberOfVoltageRangeIndexes ( ) [virtual]`

Gets the number of voltage ranges (STG5 only).

#### Returns

The number of voltage ranges available on the device.

**11.118.3.40 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.

**11.118.3.41 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.                                    |

**11.118.3.42 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.   |

**11.118.3.43 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.        |

**11.118.3.44 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.

**11.118.3.45 GetTimeResolutionInNanoSeconds()** virtual int32\_t GetTimeResolutionInNanoSeconds (

```
) [virtual]
```

Gets the time resolution of the STG.

**Returns**

The time resolution in nanoseconds.

**11.118.3.46 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.

**11.118.3.47 GetTriggerSource()** `virtual TriggerSourceEnumNet GetTriggerSource ( unsigned int triggernum ) [virtual]`**11.118.3.48 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.

**11.118.3.49 GetVoltageRangeInMicroVoltByIndex()** `virtual uint32_t GetVoltageRangeInMicroVoltByIndex ( uint32_t channel, uint32_t index ) [virtual]`

Gets the voltage range for the given channel and index (STG5 only).

**Parameters**

|                |              |
|----------------|--------------|
| <i>channel</i> | The channel. |
| <i>index</i>   | The channel. |

**Returns**

The voltage range.

**11.118.3.50 GetVoltageRangeListInMicroVolt()** `virtual System::Collections::Generic::List<int32_t> ^ GetVoltageRangeListInMicroVolt ( uint32_t channel ) [virtual]`

Gets a list of current ranges supported by the device (STG5 only).

**11.118.3.51 GetVoltageRangeSelectedIndex()** `virtual uint32_t GetVoltageRangeSelectedIndex ( uint32_t channel ) [virtual]`

Gets the currently selected range index for the voltage output (not used yet).

**Parameters**

|                |                        |
|----------------|------------------------|
| <i>channel</i> | The channel to change. |
|----------------|------------------------|

**Returns**

The currently selected range index.

**11.118.3.52 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.

**11.118.3.53 HasDigitalPort()** `virtual bool HasDigitalPort ( ) [virtual]`

Queries if the device has a digital port.

**Returns**

true if the device has a digital port.

**11.118.3.54 HasFilterAmp()** `virtual bool HasFilterAmp ( ) [virtual]`

Queries if the device has a filter amplifier. Currently the STG3008-FA supports this feature.

#### Returns

true if the device has a filter amplifier.

**11.118.3.55 LegacyHasOutputMap()** `virtual bool LegacyHasOutputMap ( ) [virtual]`

Queries if the device can map its stimulation channels. Legacy command, do not use.

#### Returns

true if the device can map its stimulation channels.

**11.118.3.56 ListModeSendStart()** `virtual void ListModeSendStart ( unsigned int electrodeGroupMask ) [virtual]`

Activate (arm) the Listmode for the selected electrode groups.

#### Parameters

|                           |                                                                      |
|---------------------------|----------------------------------------------------------------------|
| <i>electrodeGroupMask</i> | The bitmask of electrode groups for which the listmode is activated. |
|---------------------------|----------------------------------------------------------------------|

**11.118.3.57 ListModeSendStop()** `virtual void ListModeSendStop ( unsigned int electrodeGroupMask ) [virtual]`

Deactivate the Listmode for the selected electrode groups.

#### Parameters

|                           |                                                                        |
|---------------------------|------------------------------------------------------------------------|
| <i>electrodeGroupMask</i> | The bitmask of electrode groups for which the listmode is deactivated. |
|---------------------------|------------------------------------------------------------------------|

**11.118.3.58 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. |
|-------------------|---------------------------------------------|

**11.118.3.59 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. |
|-------------------|---------------------------------------------|

**11.118.3.60 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.. |

**11.118.3.61 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. |

**11.118.3.62 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. |

**11.118.3.63 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. |

**11.118.3.64 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. |

**11.118.3.65 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. |

**11.118.3.66 SetCurrentMode()** [1/2] `virtual void SetCurrentMode ( ) [virtual]`

Sets all channels to current mode (STG3008-FA and STG400x only).

**11.118.3.67 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. |
|----------------|------------------------|

**11.118.3.68 SetCurrentRangeSelectedIndex()** `virtual void SetCurrentRangeSelectedIndex ( uint32_t channel, uint32_t rangeIndex ) [virtual]`

Sets the range index for the current output (STG5 only).

## Parameters

|                   |                        |
|-------------------|------------------------|
| <i>channel</i>    | The channel to change. |
| <i>rangeIndex</i> | The new range index.   |

**11.118.3.69 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. |

**11.118.3.70 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. |
|-------------------|-------------------------|

**11.118.3.71 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. |
|--------------------|---------------------------------------|

**11.118.3.72 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 <a href="#">ElectrodeDacMuxEnumNet.Stg1</a> (1), <a href="#">ElectrodeDacMuxEnumNet.Stg2</a> (2) or <a href="#">ElectrodeDacMuxEnumNet.Stg3</a> (3). To ground an electrode, use <a href="#">ElectrodeDacMuxEnumNet.Ground</a> (0). |

**11.118.3.73 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 <a href="#">ElectrodeDacMuxEnumNet.Stg1</a> (1), <a href="#">ElectrodeDacMuxEnumNet.Stg2</a> (2) or <a href="#">ElectrodeDacMuxEnumNet.Stg3</a> (3). To ground an electrode, use <a href="#">ElectrodeDacMuxEnumNet.Ground</a> (0). |

**11.118.3.74 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 <a href="#">ElectrodeDacMuxEnumNet.Stg1</a> (1), <a href="#">ElectrodeDacMuxEnumNet.Stg2</a> (2) or <a href="#">ElectrodeDacMuxEnumNet.Stg3</a> (3). To ground an electrode, use <a href="#">ElectrodeDacMuxEnumNet.Ground</a> (0). |

**11.118.3.75 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 <a href="#">ElectrodeDacMuxEnumNet.Stg1</a> (1), <a href="#">ElectrodeDacMuxEnumNet.Stg2</a> (2) or <a href="#">ElectrodeDacMuxEnumNet.Stg3</a> (3). To ground an electrode, use <a href="#">ElectrodeDacMuxEnumNet.Ground</a> (0). |

**11.118.3.76 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. |

**11.118.3.77 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. |

**11.118.3.78 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. |

**11.118.3.79 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. |

**11.118.3.80 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.

**11.118.3.81 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. |
|-------------|----------------------------------------|

**11.118.3.82 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.

**11.118.3.83 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. |
|-------------|----------------------------------------|

**11.118.3.84 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. |

**11.118.3.85 SetEnableAmplifierProtectionSwitch()** [2/4] virtual void SetEnableAmplifierProtectionSwitch (

```

 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. |

**11.118.3.86 SetEnableAmplifierProtectionSwitch()** [3/4] virtual void SetEnableAmplifierProtectionSwitch (

```

 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. |

**11.118.3.87 SetEnableAmplifierProtectionSwitch()** [4/4] virtual void SetEnableAmplifierProtectionSwitch (

```

 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. |

**11.118.3.88 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. |

**11.118.3.89 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. |

**11.118.3.90 SetFAAmplification()** virtual void SetFAAmplification (   
     *Stg3008FilterAmpAmplificationEnumNet* *amplification* ) [virtual]

Sets the amplification of the filter amplifier (STG3008-FA).

**11.118.3.91 SetHeadstage()** virtual void SetHeadstage (   
     unsigned int *headstage* ) [virtual]

**11.118.3.92 SetListmodeIndexRange()** virtual void SetListmodeIndexRange (   
     unsigned int *electrodeGroup*,   
     unsigned int *startIndex*,   
     unsigned int *endIndex*,   
     unsigned int *mode* ) [virtual]

Define the range of list mode indexes to use for the given electrode group.

#### Parameters

|                       |                                                    |
|-----------------------|----------------------------------------------------|
| <i>electrodeGroup</i> | The electrodegroup for which the range is defined. |
|-----------------------|----------------------------------------------------|

#### Parameters

|                   |                                                             |
|-------------------|-------------------------------------------------------------|
| <i>startIndex</i> | The index of the first active element in the listmode list. |
| <i>endIndex</i>   | The index of the last active element in the listmode list.  |
| <i>mode</i>       | 0 for "start with startIndex", 1 for "start with endIndex". |

**11.118.3.93 SetListmodeTriggerSource()** [1/2] virtual void SetListmodeTriggerSource (   
     unsigned int *electrodeGroup*,   
     *TriggerSourceEnumNet* *triggersource* ) [virtual]

Define the signal which triggers the transition from one list mode entry to the next. After reaching the last entry in the list, the first entry is selected. For triggersource use the Enum which corresponds to the device in use, for example use *SCUDigitalSourceEnumNet* and cast to *TriggerSourceEnumNet* if working with an SCU device.

#### Parameters

|                       |                                                            |
|-----------------------|------------------------------------------------------------|
| <i>electrodeGroup</i> | The electrodegroup for which the triggersource is defined. |
|-----------------------|------------------------------------------------------------|

## Parameters

|                      |                                                 |
|----------------------|-------------------------------------------------|
| <i>triggersource</i> | One of the possible sources for the transition. |
|----------------------|-------------------------------------------------|

**11.118.3.94 SetListmodeTriggerSource()** [2/2] virtual void SetListmodeTriggerSource ( unsigned int *electrodeGroup*, *TriggerSourceEnumNet* *triggersource*, int *bitnumOffset* ) [virtual]

Define the signal which triggers the transition from one list mode entry to the next. After reaching the last entry in the list, the first entry is selected. For *triggersource* use the Enum which corresponds to the device in use, for example use *SCUDigitalSourceEnumNet* and cast to *TriggerSourceEnumNet* if working with an SCU device.

## Parameters

|                       |                                                                   |
|-----------------------|-------------------------------------------------------------------|
| <i>electrodeGroup</i> | The electrodegroup for which the <i>triggersource</i> is defined. |
|-----------------------|-------------------------------------------------------------------|

## Parameters

|                      |                                                                      |
|----------------------|----------------------------------------------------------------------|
| <i>triggersource</i> | One of the possible sources for the transition.                      |
| <i>bitnumOffset</i>  | Number to add to the numeric value of the <i>triggersource</i> enum. |

**11.118.3.95 SetMeasurementMode()** virtual void SetMeasurementMode ( unsigned int *channel* ) [virtual]

Sets a channel to measurement mode (STG3008-FA).

## Parameters

|                |                        |
|----------------|------------------------|
| <i>channel</i> | The channel to change. |
|----------------|------------------------|

**11.118.3.96 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. |
|-------------|----------------------------|

**11.118.3.97 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. |

**11.118.3.98 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. |

**11.118.3.99 SetTriggerSource() [1/3]** `virtual void SetTriggerSource (`  
    `unsigned int triggernum,`  
    `DigitalSource< digitalsourceenum >^ triggersource,`  
    `int bitnum_offset ) [virtual]`

**11.118.3.100 SetTriggerSource() [2/3]** `virtual void SetTriggerSource (`  
    `unsigned int triggernum,`  
    `TriggerSourceEnumNet triggersource ) [virtual]`

**11.118.3.101 SetTriggerSource() [3/3]** `virtual void SetTriggerSource (`  
    `unsigned int triggernum,`  
    `TriggerSourceEnumNet triggersource,`  
    `int bitnum_offset ) [virtual]`

**11.118.3.102 SetVoltageMode()** [1/2] `virtual void SetVoltageMode ( ) [virtual]`

Sets all channels to voltage mode (STG3008-FA and STG400x only).

**11.118.3.103 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. |
|----------------|------------------------|

**11.118.3.104 SetVoltageRangeSelectedIndex()** `virtual void SetVoltageRangeSelectedIndex ( uint32_t channel, uint32_t rangeIndex ) [virtual]`

Sets the range index for the voltage output (not used yet).

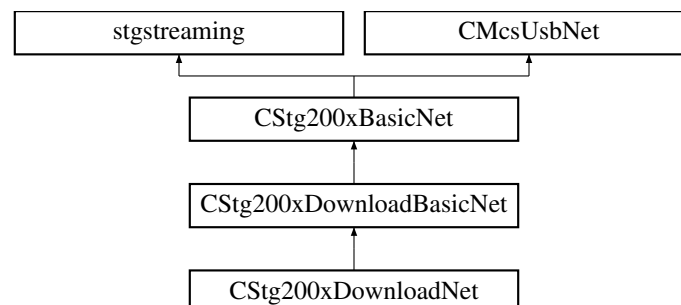
#### Parameters

|                   |                        |
|-------------------|------------------------|
| <i>channel</i>    | The channel to change. |
| <i>rangeIndex</i> | The new range index.   |

## 11.119 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 **CSig200xDownloadBasicNet::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

### 11.119.1 Detailed Description

[CStg200xDownloadBasicNet](#) is the base class to control the download mode of the MCS STG device.

### 11.119.2 Member Function Documentation

**11.119.2.1 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. |
|----------------|---------------------------------|

**11.119.2.2 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. |
|----------------|-----------------------------------------|

**11.119.2.3 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()`

**11.119.2.4 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.

**11.119.2.5 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.

**11.119.2.6 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.

**11.119.2.7 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.

**11.119.2.8 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. |

**11.119.2.9 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. |

**11.119.2.10 ResetStatus()** `void ResetStatus (`  
     `uint32_t triggermap )`

Reset the status flag.

#### Parameters

|                   |                                                  |
|-------------------|--------------------------------------------------|
| <i>triggermap</i> | bitmap of trigger for which to reset the status. |
|-------------------|--------------------------------------------------|

**11.119.2.11 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. |

**11.119.2.12 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. |

**11.119.2.13 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. |

**11.119.2.14 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. |

**11.119.2.15 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. |

**11.119.2.16 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. |

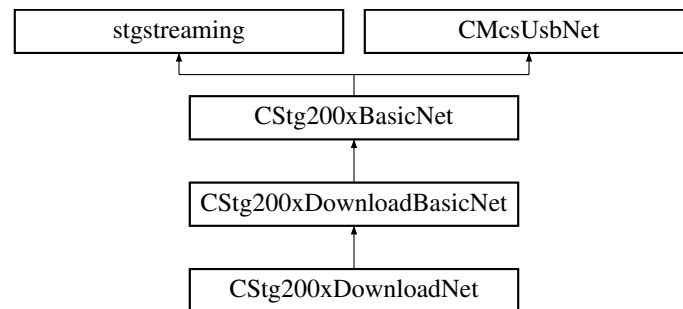
### 11.119.3 Property Documentation

#### 11.119.3.1 Stimulus `CStimulusFunctionNet^` Stimulus [get]

## 11.120 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^ pollStatus)`  
*Use this constructor if you want to use the status callback.*
- `~CStg200xDownloadNet ()`
- 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.*
- 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.*
- 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.*
- void `SendPreparedData` (uint32\_t channel, array< uint8\_t >^ pData, STG\_DestinationEnumNet destType, Net)
- void `AppendPreparedData` (uint32\_t channel, array< uint8\_t >^ pData, STG\_DestinationEnumNet destType, Net)
- void `SegmentDefine` (array< uint32\_t >^ segment\_list)  
*Defines the segment memory layout of the STG.*
- void `SegmentStart` (uint32\_t triggermap, uint32\_t segment, Stg200xSegmentFlagsEnumNet segmentflags)



*Switchs segment and starts trigger.*

- void [SegmentSelect](#) (uint32\_t segment, [Stg200xSegmentFlagsEnumNet](#) segmentflags)

*Switchs segment.*

- void [EnableMultiFileMode](#) ([Stg200xMultiFileSubmodeEnumNet](#) 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

### 11.120.1 Detailed Description

Main class for the STG download mode This class implements the STG download mode interface.

### 11.120.2 Constructor & Destructor Documentation

#### 11.120.2.1 CStg200xDownloadNet() [1/2] [CStg200xDownloadNet](#) ( )

Use this constructor if you do not want to use the status callback.

#### 11.120.2.2 CStg200xDownloadNet() [2/2] [CStg200xDownloadNet](#) ( [OnStgPollStatus](#) ^ *pollStatus* )

Use this constructor if you want to use the status callback.

#### 11.120.2.3 ~CStg200xDownloadNet() [~CStg200xDownloadNet](#) ( )

### 11.120.3 Member Function Documentation

**11.120.3.1 AppendPreparedData()** `void AppendPreparedData (`  
`uint32_t channel,`  
`array< uint8_t >^ pData,`  
`STG_DestinationEnumNet destTypeNet )`

**11.120.3.2 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  $\pm 2000$  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. |

**11.120.3.3 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.

**11.120.3.4 EnableMultiFileMode()** `void EnableMultiFileMode (   
Stg200xMultiFileSubmodeEnumNet 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. |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**11.120.3.5 GetModuleCurrent()** `uint32_t GetModuleCurrent (   
unsigned int channel )`

**11.120.3.6 GetModuleTemp()** `int32_t GetModuleTemp (   
unsigned int channel )`

**11.120.3.7 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  $\mu$ 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 $\mu\text{V}$ and nA in voltage and current mode, respectively. |
|------------------|--------------------------------------------------------------------------------------------------|

**Parameters**

|                 |                                                                             |
|-----------------|-----------------------------------------------------------------------------|
| <i>duration</i> | A list of durations in units of $\mu\text{s}$ .                             |
| <i>destType</i> | specifies wheather the data is for syncout, current or voltage stimulation. |

**11.120.3.8 PrepareAndSendData()** `void PrepareAndSendData (`  
    `uint32_t channel,`  
    `array< int32_t >^ amplitude,`  
    `array< uint64_t >^ duration,`  
    `STG_DestinationEnumNet destType )`

Prepare and send data to a given channel on the STG. Previous data sent to that channel is erased first.

Each datapoint is represented by an signed 32bit integer value. When using voltage stimulation, the values are in multiple of 1  $\mu\text{V}$ , thus the possible range is  $\pm 2000\text{ V}$ . When using current stimulation, the values are in multiple of 1 nA, this the possible range is  $\pm 2000\text{ mA}$ .

The duration is given as a list of 64 bit integers. Durations are given in units of  $\mu\text{s}$ . The STG has a resolution of 20  $\mu\text{s}$ .

Blocks of data which should repeat can be defined by prepending such a block with an entry in the arrays where both amplitude and duration is zero. The end of such an block is marked by an entry where the duration is set to zero and the amplitude beeing set to the number of times the block should run. Blocks can be nested.

**Parameters**

|                |                                     |
|----------------|-------------------------------------|
| <i>channel</i> | The channel number to send data to. |
|----------------|-------------------------------------|

**Parameters**

|                  |                                                                                                  |
|------------------|--------------------------------------------------------------------------------------------------|
| <i>amplitude</i> | A list of amplitudes in units of $\mu\text{V}$ and nA in voltage and current mode, respectively. |
|------------------|--------------------------------------------------------------------------------------------------|

## Parameters

|                 |                                                                             |
|-----------------|-----------------------------------------------------------------------------|
| <i>duration</i> | A list of durations in units of $\mu$ s.                                    |
| <i>destType</i> | specifies wheather the data is for syncout, current or voltage stimulation. |

**11.120.3.9 QueryTriggerstatus()** `StgStatusNet ^ QueryTriggerstatus ( )`**11.120.3.10 SegmentDefine()** `void SegmentDefine ( 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). |
|---------------------|---------------------------------------------|

**11.120.3.11 SegmentSelect()** `void SegmentSelect ( 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. |
|---------------------|-------------------------------------------------------------------------------------------|

**11.120.3.12 SegmentStart()** `void SegmentStart (`  
    `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. |
|---------------------|-------------------------------------------------------------------------------------------|

**11.120.3.13 SendPreparedData()** `void SendPreparedData (`  
    `uint32_t channel,`  
    `array< uint8_t >^ pData,`  
    `STG_DestinationEnumNet destTypeNet )`

**11.120.3.14 SetOutputMap()** `void SetOutputMap (`  
    `array< uint32_t >^ ChannelLayout )`

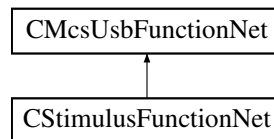
## 11.120.4 Event Documentation

**11.120.4.1 MwPollStatusEvent** `OnMwPollStatus^ MwPollStatusEvent [add], [remove], [raise]`

**11.120.4.2 Stg200xPollStatusEvent** [OnStgPollStatus](#)<sup>^</sup> [Stg200xPollStatusEvent](#) [add], [remove], [raise]

## 11.121 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](#) ^ device\_data\_and\_unrolled, [STG\\_DestinationEnumNet](#) destType)
- [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.*
- 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.*
- int [GetCurrentResolutionInPicoAmp](#) (uint32\_t channel)
 

*Gets the Current Resolution of the specified channel in Picoamps.*
- 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

### 11.121.1 Constructor & Destructor Documentation



**11.121.1.1 CStimulusFunctionNet()** [1/2] `CStimulusFunctionNet (`  
`CMcsUsbNet^ mcsusb,`  
`CMcsUsbFunctionPointerContainer^ stimulusFunctionPointerContainer )`

**11.121.1.2 CStimulusFunctionNet()** [2/2] `CStimulusFunctionNet (`  
`CMcsUsbNet^ mcsusb )`

## 11.121.2 Member Function Documentation

**11.121.2.1 ClearChannel\_PrepareAndSendData()** `void ClearChannel_PrepareAndSendData (`  
`uint32_t channel,`  
`array< int32_t >^ amplitude,`  
`array< uint64_t >^ duration,`  
`STG_DestinationEnumNet destType,`  
`bool doClear )`

**11.121.2.2 ClearChannelData()** `void ClearChannelData (`  
`int channel )`

Delete a Stimulus Pattern from STG memory

### Parameters

|                |                                 |
|----------------|---------------------------------|
| <i>channel</i> | specifies the channel to clear. |
|----------------|---------------------------------|

**11.121.2.3 ClearMultiplexedData()** `void ClearMultiplexedData ( )`

Clears the Stimulation Memory in the STG device.

**11.121.2.4 ClearSyncData()** `void ClearSyncData (`  
`int channel )`

Delete a Syncout Pattern from STG memory

### Parameters

|                |                                 |
|----------------|---------------------------------|
| <i>channel</i> | specifies the channel to clear. |
|----------------|---------------------------------|

```

11.121.2.5 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 1 means that the stimulation is active during that time. A value 0 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>Syncout</i>  | A list of syncouts as int32. They define additional syncout.                 |
| <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.

**11.121.2.6 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.

**11.121.2.7 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.

**11.121.2.8 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.

**11.121.2.9 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.

**11.121.2.10 GetCurrentResolutionInPicoAmp()** `int GetCurrentResolutionInPicoAmp (`  
`uint32_t channel )`

Gets the Current Resolution of the specified channel in Picoamps.

## Parameters

|                |                           |
|----------------|---------------------------|
| <i>channel</i> | Channel which is queried. |
|----------------|---------------------------|

## Returns

The Current Resolution of the specified channel in Picoamps.

**11.121.2.11 GetDACResolution()** `int GetDACResolution ( )`

Gets number of bits of the DAC resolution.

## Returns

The DAC resolution in bits.

**11.121.2.12 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`

**11.121.2.13 GetNumberOfAnalogChannels()** `int GetNumberOfAnalogChannels ( )`

Get the number of STG channels.

## Returns

The number of STG channels.

**11.121.2.14 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.

**11.121.2.15 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 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 += 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 $\mu\text{V}$ and nA in voltage and current mode, respectively. |
|------------------|--------------------------------------------------------------------------------------------------|

## Parameters

|                 |                                                                             |
|-----------------|-----------------------------------------------------------------------------|
| <i>duration</i> | A list of durations in units of $\mu\text{s}$ .                             |
| <i>destType</i> | specifies wheather the data is for syncout, current or voltage stimulation. |

## Returns

Error Status. 0 on success.

**11.121.2.18 PrepareAndSendData()** `void PrepareAndSendData (`  
`uint32_t channel,`  
`array< int32_t >^ amplitude,`  
`array< uint64_t >^ duration,`  
`STG_DestinationEnumNet destType )`

Prepare and send data to a given channel on the STG. Previous data sent to that channel is erased first.

Each datapoint is represented by an signed 32bit integer value. When using voltage stimulation, the values are in multiple of 1  $\mu\text{V}$ , thus the possible range is  $\pm 2000 \text{ V}$ . When using current stimulation, the values are in multiple of 1 nA, this the possible range is  $\pm 2000 \text{ mA}$ .

The duration is given as a list of 64 bit integers. Durations are given in units of  $\mu\text{s}$ . The STG has a resolution of 20  $\mu\text{s}$ .

Blocks of data which should repeat can be defined by prepending such a block with an entry in the arrays where both amplitude and duration is zero. The end of such an block is marked by an entry where the duration is set to zero and the amplitude beeing set to the number of times the block should run. Blocks can be nested.

## Parameters

|                |                                     |
|----------------|-------------------------------------|
| <i>channel</i> | The channel number to send data to. |
|----------------|-------------------------------------|

## Parameters

|                  |                                                                                                  |
|------------------|--------------------------------------------------------------------------------------------------|
| <i>amplitude</i> | A list of amplitudes in units of $\mu\text{V}$ and nA in voltage and current mode, respectively. |
|------------------|--------------------------------------------------------------------------------------------------|

**Parameters**

|                 |                                                                             |
|-----------------|-----------------------------------------------------------------------------|
| <i>duration</i> | A list of durations in units of $\mu$ s.                                    |
| <i>destType</i> | specifies wheather the data is for syncout, current or voltage stimulation. |

**Returns**

Error Status. 0 on success.

**11.121.2.19 PrepareData()** `StimulusDeviceDataAndUnrolledData ^ PrepareData (`  
    `int channel,`  
    `array< int32_t >^ amplitude,`  
    `array< uint64_t >^ duration,`  
    `STG_DestinationEnumNet destType )`

**11.121.2.20 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. |
|-------------|---------------------------|

**11.121.2.21 SendPreparedData()** `void SendPreparedData (`  
    `int channel,`  
    `StimulusDeviceDataAndUnrolledData^ device_data_and_unrolled,`  
    `STG_DestinationEnumNet destType )`

**11.121.2.22 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. |
|-------------------|---------------------------------------------|

**11.121.2.23 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. |
|-------------------|---------------------------------------------|

**11.121.2.24 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.. |

**11.121.2.25 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. |

**11.121.2.26 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. |

**11.121.2.27 StartPoll()** `void StartPoll ( )`

Starts the interrupt fetching thread and delivers events

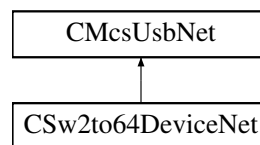
**11.121.2.28 StopPoll()** `void StopPoll ( )`

Stops the interrupt fetching thread and delivers events

**11.121.3 Event Documentation****11.121.3.1 PollStatusEvent** `OnStgPollStatus^ PollStatusEvent`**11.122 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****11.122.1 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.

## 11.122.2 Constructor & Destructor Documentation

**11.122.2.1 CSw2to64DeviceNet()** `CSw2to64DeviceNet ( )`

**11.122.2.2 ~CSw2to64DeviceNet()** `~CSw2to64DeviceNet ( )`

## 11.122.3 Member Function Documentation

**11.122.3.1 GetChannel()** `unsigned char GetChannel ( unsigned short index )`

Gets one current switch position.

### Parameters

|           |              |                                                    |
|-----------|--------------|----------------------------------------------------|
| <i>in</i> | <i>index</i> | number of channel to read the switch position from |
|-----------|--------------|----------------------------------------------------|

### Returns

switch position of desired channel

**11.122.3.2 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

**11.122.3.3 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

**11.122.3.4 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                    |

**11.122.3.5 SetChannels()** `void SetChannels (`  
`array< unsigned char >^ pattern )`

Sets the switch positions from a char array.

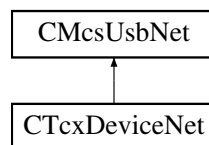
## Parameters

|    |                |                                                                                               |
|----|----------------|-----------------------------------------------------------------------------------------------|
| in | <i>pattern</i> | array of char with the size of the number of channels, each char has the setting of a channel |
|----|----------------|-----------------------------------------------------------------------------------------------|

## 11.123 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 whether 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 temperature of the mainboard in units of 0.1 °C.*
- unsigned int [GetVolti](#) (unsigned int channel)
- unsigned int [GetNumDevices](#) ()
- void [SetSetpoint](#) (unsigned int channel, int sp)  
*Sets the target temperature 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 temperature 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

### 11.123.1 Detailed Description

Class to control a Temperature Controller (TCX)

### 11.123.2 Constructor & Destructor Documentation

**11.123.2.1 CTcxDeviceNet()** `CTcxDeviceNet ( )`

Initializes a new instance of `CTcxDeviceNet` class.

**11.123.2.2 ~CTcxDeviceNet()** `~CTcxDeviceNet ( )`**11.123.3 Member Function Documentation****11.123.3.1 CalibrateThermocouple()** `void CalibrateThermocouple ( unsigned int channel )`**11.123.3.2 FactoryReset()** `void FactoryReset ( )`**11.123.3.3 GetBoardTemp()** `unsigned int GetBoardTemp ( )`

Gets the temperate of the mainboard in units of 0.1 °C.

**11.123.3.4 GetCalibration()** `int GetCalibration ( unsigned int channel )`**11.123.3.5 GetCalibrationDecp()** `int GetCalibrationDecp ( unsigned int channel )`**11.123.3.6 GetCalibrationMax()** `int GetCalibrationMax ( unsigned int channel )`**11.123.3.7 GetCalibrationMin()** `int GetCalibrationMin ( unsigned int channel )`



**11.123.3.8 GetCurrent()** `int GetCurrent (`  
`unsigned int channel )`

**11.123.3.9 GetD()** `int GetD (`  
`unsigned int device )`

Gets the D-coefficient of the specified device.

**11.123.3.10 GetDDecp()** `int GetDDecp (`  
`unsigned int device )`

**11.123.3.11 GetDevice()** `int GetDevice (`  
`unsigned int channel )`

**11.123.3.12 GetDeviceType()** `TcxDeviceTypeEnumNet GetDeviceType ( )`

**11.123.3.13 GetDevname()** `String ^ GetDevname (`  
`unsigned int device )`

**11.123.3.14 GetDMax()** `int GetDMax (`  
`unsigned int device )`

**11.123.3.15 GetDMin()** `int GetDMin (`  
`unsigned int device )`

**11.123.3.16 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 %.

**11.123.3.17 GetEnableHeaterLimit()** `bool GetEnableHeaterLimit ( unsigned int device )`

**11.123.3.18 GetEnableThermocouple()** `bool GetEnableThermocouple ( unsigned int device )`

**11.123.3.19 GetHasThermocouple()** `bool GetHasThermocouple ( )`

Gets whether the device supports a thermocouple.

**11.123.3.20 GetHeaterLimit()** `int GetHeaterLimit ( unsigned int device )`

Gets the temperature limit of the specified heater in units of 0.1 °C.

**11.123.3.21 GetHeaterTemp()** `int GetHeaterTemp ( unsigned int channel )`

Gets the temperature of the specified heater in units of 0.1 °C.

**11.123.3.22 GetI()** `int GetI ( unsigned int device )`

Gets the I-coefficient of the specified device.

**11.123.3.23 GetIDecp()** `int GetIDecp ( unsigned int device )`

**11.123.3.24 GetIMax()** `int GetIMax ( unsigned int device )`

**11.123.3.25 GetIMin()** `int GetIMin ( unsigned int device )`

**11.123.3.26 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.

**11.123.3.27 GetMaxHeaterPowerMultiwell()** `double GetMaxHeaterPowerMultiwell ( )`

queries the max. heater power that the Multiwell temperature controller will apply; unit: W; useful range: 5.2W..7.6W

**11.123.3.28 GetMaxP()** `int GetMaxP ( unsigned int device )`

Gets the maximum heater power of the specified device.

**11.123.3.29 GetMaxpDecp()** `int GetMaxpDecp ( unsigned int device )`

**11.123.3.30 GetMaxpMax()** `int GetMaxpMax (`  
`unsigned int device )`

**11.123.3.31 GetMaxpMin()** `int GetMaxpMin (`  
`unsigned int device )`

**11.123.3.32 GetNumControlChannels()** `unsigned int GetNumControlChannels ( )`

Gets the number of channels the device can control/regulate.

**11.123.3.33 GetNumDevices()** `unsigned int GetNumDevices ( )`

**11.123.3.34 GetNumMeasureChannels()** `unsigned int GetNumMeasureChannels ( )`

Gets the number of channels the device can measure.

**11.123.3.35 GetOnOff()** `int GetOnOff (`  
`unsigned int channel )`

Gets if the specified channel is on or off.

**11.123.3.36 GetP()** `int GetP (`  
`unsigned int device )`

Gets the P-coefficient of the specified device.

**11.123.3.37 GetPDecp()** `int GetPDecp (`  
`unsigned int device )`

**11.123.3.38 GetPMax()** `int GetPMax (`  
`unsigned int device )`

**11.123.3.39 GetPMin()** `int GetPMin (`  
`unsigned int device )`

**11.123.3.40 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.

**11.123.3.41 GetPwrOut()** `int GetPwrOut ( unsigned int channel )`

**11.123.3.42 GetPwrSet()** `int GetPwrSet ( unsigned int channel )`

**11.123.3.43 GetRes1()** `int GetRes1 ( unsigned int channel )`

**11.123.3.44 GetRes2()** `int GetRes2 ( unsigned int channel )`

**11.123.3.45 GetResS()** `int GetResS ( unsigned int channel )`

**11.123.3.46 GetResX()** `int GetResX ( unsigned int channel )`

**11.123.3.47 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.

**11.123.3.48 GetSensorType()** `TcxSensorTypeEnumNet` GetSensorType (   
 unsigned int *device* )

**11.123.3.49 GetSetpoint()** `int` GetSetpoint (   
 unsigned int *channel* )

Gets the target temperate of specified channel in units of 0.1 °C.

**11.123.3.50 GetSetpointDecp()** `int` GetSetpointDecp (   
 unsigned int *channel* )

**11.123.3.51 GetSetpointMax()** `int` GetSetpointMax (   
 unsigned int *channel* )

**11.123.3.52 GetSetpointMin()** `int` GetSetpointMin (   
 unsigned int *channel* )

**11.123.3.53 GetThermocoupleCalibration()** `int` GetThermocoupleCalibration (   
 unsigned int *channel* )

**11.123.3.54 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.

**11.123.3.55 GetThermocoupleReferenceTemp()** `int GetThermocoupleReferenceTemp ( unsigned int channel )`

**11.123.3.56 GetThermocoupleTemp()** `int GetThermocoupleTemp ( unsigned int channel )`

**11.123.3.57 GetThermocoupleTempAbs()** `int GetThermocoupleTempAbs ( unsigned int channel )`

**11.123.3.58 GetUnit()** `String ^ GetUnit ( unsigned int channel )`

**11.123.3.59 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.

**11.123.3.60 GetValue()** `int GetValue (`  
`unsigned int channel )`

Gets the temperate of the specified channel in units of 0.1 °C.

**11.123.3.61 GetValueHires()** `int GetValueHires (`  
`unsigned int channel )`

Gets the temperate of the specified channel in units of 0.01 °C.

**11.123.3.62 GetVolti()** `unsigned int GetVolti (`  
`unsigned int channel )`

**11.123.3.63 SetCalibration()** `void SetCalibration (`  
`unsigned int channel,`  
`int calib )`

**11.123.3.64 SetD()** `void SetD (`  
`unsigned int device,`  
`int d_coeff )`

Sets the D-coefficient of the specified device.

**11.123.3.65 SetDevice()** `void SetDevice (`  
`unsigned int channel,`  
`int device )`

**11.123.3.66 SetDeviceType()** `void SetDeviceType (`  
`TcxDeviceTypeEnumNet devicetype )`

**11.123.3.67 SetDevname()** `void SetDevname (`  
`unsigned int device,`  
`String^ Devicename )`



**11.123.3.68 SetEnableHeaterLimit()** `void SetEnableHeaterLimit (`  
     `unsigned int device,`  
     `bool enable )`

**11.123.3.69 SetEnableThermocouple()** `void SetEnableThermocouple (`  
     `unsigned int device,`  
     `bool enable )`

**11.123.3.70 SetHeaterLimit()** `void SetHeaterLimit (`  
     `unsigned int device,`  
     `int heater_limit )`

**11.123.3.71 SetI()** `void SetI (`  
     `unsigned int device,`  
     `int i_coeff )`

Sets the I-coefficient of the specified device.

**11.123.3.72 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

**11.123.3.73 SetMaxP()** `void SetMaxP (`  
     `unsigned int device,`  
     `int maxp )`

Sets the maximum heater power of the specified device.

**11.123.3.74 SetOnOff()** `void SetOnOff (`  
     `unsigned int channel,`  
     `bool on )`

Switches the specified channel on or off.

#### Parameters

|                |                     |
|----------------|---------------------|
| <i>channel</i> | The channel number. |
|----------------|---------------------|

**11.123.3.75 SetP()** `void SetP (`  
    `unsigned int device,`  
    `int p_coeff )`

Sets the P-coefficient of the specified device.

**11.123.3.76 SetSensorType()** `void SetSensorType (`  
    `unsigned int device,`  
    `TcxSensorTypeEnumNet type )`

**11.123.3.77 SetSetpoint()** `void SetSetpoint (`  
    `unsigned int channel,`  
    `int sp )`

Sets the target temperate of specified channel in units of 0.1 °C.

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

Sets the proportional constant for the thermocouple.

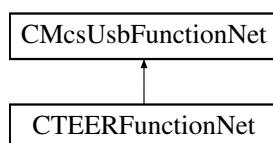
#### Parameters

|                |                                              |
|----------------|----------------------------------------------|
| <i>channel</i> | Thermocouple channel number.                 |
| <i>value</i>   | Proportinal constant in Nanovolt per Kelvin. |

## 11.124 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> pTEERFunctionPointer↔  
Container)  
*Initializes a new instance of the [CTEERFunctionNet](#) class.*
- [CTEERFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- virtual [~CTEERFunctionNet](#) ()
- [ICTEERFunctionNet](#) ()
- [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 resistance 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

### 11.124.1 Detailed Description

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

### 11.124.2 Constructor & Destructor Documentation

**11.124.2.1 CTEERFunctionNet()** [1/2] [CTEERFunctionNet](#) (  
     CMcsUsbNet^ mcsusb,  
     CMcsUsbFunctionPointerContainer^ pTEERFunctionPointerContainer )

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

**11.124.2.2 CTEERFunctionNet()** [2/2] `CTEERFunctionNet ( CMcsUsbNet^ mcsusb )`

**11.124.2.3 ~CTEERFunctionNet()** `virtual ~CTEERFunctionNet ( ) [virtual]`

**11.124.2.4 !CTEERFunctionNet()** `!CTEERFunctionNet ( )`

### 11.124.3 Member Function Documentation

**11.124.3.1 CancelInternalCalibration()** `void CancelInternalCalibration ( )`

in case the internal calibration "hangs", this will cancel it

**11.124.3.2 GetAdapterCode()** `uint32_t GetAdapterCode ( )`

gets the adapter code

#### Returns

the adapter code

**11.124.3.3 GetAdcOffsetU1()** `int GetAdcOffsetU1 ( )`

returns ADC offset of U1 channel (result of internal calibration)

#### Returns

the ADC offset for U1

**11.124.3.4 GetAdcOffsetU2()** `int GetAdcOffsetU2 ( )`

returns ADC offset of U2 channel (result of internal calibration)

#### Returns

the ADC offset for U2

**11.124.3.5 GetAmplitude\_nA()** `uint32_t GetAmplitude_nA ( )`

gets TEER stimulation amplitude in nA

**Returns**

current stimulation amplitude in nA

**11.124.3.6 GetBytesPerSample()** `uint32_t GetBytesPerSample ( )`

private function to be called internally only

**Returns**

not documented

**11.124.3.7 GetClampMode()** `TeerClampModeEnumNet GetClampMode ( )`

gets TEER clamp mode (voltage/current)

**Returns**

current TEER clamp mode

**11.124.3.8 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 |

**11.124.3.9 GetCurrentEnable()** `bool GetCurrentEnable ( )`

when disabled, no current will flow through chamber

**Returns**

false when disabled, true when enabled

**11.124.3.10 GetDacZero()** `int GetDacZero ( )`

returns DAC-offset (result of internal calibration); use to check for plausibility only

**Returns**

the DAC offset

**11.124.3.11 GetFrameErrorCounter()** `int GetFrameErrorCounter ( )`

returns number of times (since bootup) sample memory got overwritten

**Returns**

the number of errors

**11.124.3.12 GetLiquidResistance()** `int32_t GetLiquidResistance ( )`

gets the resistance of the liquid in ohms

**Returns**

the resistance in ohms

**11.124.3.13 GetMaxChunkSize\_Byte()** `uint32_t GetMaxChunkSize_Byte ( )`

private function to be called internally only

**Returns**

not documented

**11.124.3.14 GetNumberOfAvailableSamples()** `uint32_t GetNumberOfAvailableSamples ( )`

private function to be called internally only

**Returns**

not documented

**11.124.3.15 GetPeriod\_us()** `uint32_t GetPeriod_us ( )`

gets the period of TEER stimulation in us

**Returns**

the period in us

**11.124.3.16 GetRotaryPositionCode()** `uint32_t GetRotaryPositionCode ( )`

gets the rotary position code

**Returns**

the rotary position code

**11.124.3.17 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

**11.124.3.18 GetSampleRate()** `int GetSampleRate ( )`

returns sample rate in Hz



**Returns**

the sample rate in Hz

**11.124.3.19 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

**11.124.3.20 GetScaleFactorU1()** `int GetScaleFactorU1 ( )`

returns U1 scale factor times  $10^6$  (result of internal calibration)

**Returns**

the U1 scale factor

**11.124.3.21 GetScaleFactorU2()** `int GetScaleFactorU2 ( )`

returns U2 scale factor times  $10^6$  (result of internal calibration)

**Returns**

the U2 scale factor

**11.124.3.22 GetUptimeSeconds()** `int32_t GetUptimeSeconds ( )`

returns time in seconds since device was powered up

**Returns**

seconds since power-on

**11.124.3.23 GetWaveform()** `TeerWaveformEnumNet GetWaveform ( )`

gets TEER stimulation waveform (sine/rect)

**Returns**

waveform enum

**11.124.3.24 IsInternalCalibrationFinished()** `bool IsInternalCalibrationFinished ( )`

queries whether internal calibration has finished

**Returns**

true if calibration has finished

**11.124.3.25 IsSamplingFinished()** `uint32_t IsSamplingFinished ( )`

returns false iff stimulation/sampling is going on, otherwise true

**Returns**

true if sampling is finished

**11.124.3.26 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 |
|---------------------|---------------------------------|

**11.124.3.27 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 |
|-----------------------|---------------------------------------------|

**11.124.3.28 SetClampMode()** `void SetClampMode (`  
`TeerClampModeEnumNet ClampMode )`

sets TEER clamp mode (voltage/current)

Parameters

|                  |                     |
|------------------|---------------------|
| <i>ClampMode</i> | new TEER clamp mode |
|------------------|---------------------|

**11.124.3.29 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 |

**11.124.3.30 SetCurrentEnable()** `void SetCurrentEnable (`  
`bool NewCurrentEnable )`

when disabled, no current will flow through chamber

Parameters

|                         |                                        |
|-------------------------|----------------------------------------|
| <i>NewCurrentEnable</i> | disabled when false, enabled when true |
|-------------------------|----------------------------------------|

**11.124.3.31 SetExternalLED()** `void SetExternalLED (`  
`uint32_t NewState )`

sets the external LED

Parameters

|                 |       |
|-----------------|-------|
| <i>NewState</i> | state |
|-----------------|-------|

**11.124.3.32 SetLiquidResistance()** `void SetLiquidResistance (`  
`int32_t NewLiquidResistance_Ohm )`

sets the resistance of the liquid in ohms

**Parameters**

|                                |                        |
|--------------------------------|------------------------|
| <i>NewLiquidResistance_Ohm</i> | the resistance in ohms |
|--------------------------------|------------------------|

**11.124.3.33 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 |
|------------------|------------------|

**11.124.3.34 SetWaveform()** `void SetWaveform (`  
`TeerWaveformEnumNet Waveform )`

sets TEER stimulation waveform (sine/rect)

**Parameters**

|                 |               |
|-----------------|---------------|
| <i>Waveform</i> | waveform enum |
|-----------------|---------------|

**11.124.3.35 StartInternalCalibration()** `void StartInternalCalibration ( )`

starts determination of internal DAC-offset; result is used internally; NON-BLOCKING call

**11.124.3.36 StartSampling()** `void StartSampling (`  
`uint32_t NumberOfCycles )`

starts TEER stimulation (duration: n cycles) and samples during last cycle

## Parameters

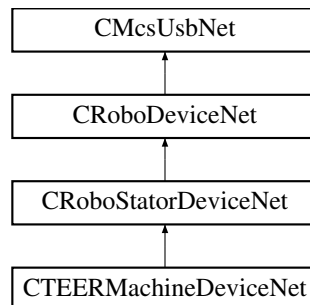
|                       |                                                              |
|-----------------------|--------------------------------------------------------------|
| <i>NumberOfCycles</i> | number of cycles (sine or rect) to output (0 - loop forever) |
|-----------------------|--------------------------------------------------------------|

**11.124.3.37 StopSampling()** void StopSampling ( )

stops TEER stimulation and sampling

**11.125 CTEERMachineDeviceNet Class Reference**

Inheritance diagram for CTEERMachineDeviceNet:

**Public Member Functions**

- [CTEERMachineDeviceNet](#) ( )
- [~CTEERMachineDeviceNet](#) ( )

**Properties**

- [CTEERFunctionNet](#)<sup>^</sup> [TEERFunctionNet](#) [get]

**Additional Inherited Members****11.125.1 Constructor & Destructor Documentation****11.125.1.1 CTEERMachineDeviceNet()** [CTEERMachineDeviceNet](#) ( )**11.125.1.2 ~CTEERMachineDeviceNet()** [~CTEERMachineDeviceNet](#) ( )

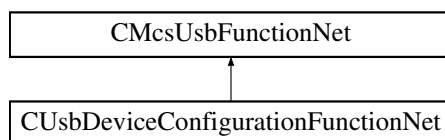
## 11.125.2 Property Documentation

### 11.125.2.1 TEERFunctionNet `CTEERFunctionNet^ TEERFunctionNet [get]`

## 11.126 CUsbDeviceConfigurationFunctionNet Class Reference

`CUsbDeviceConfigurationFunctionNet` is the class to configure the USB firmware

Inheritance diagram for `CUsbDeviceConfigurationFunctionNet`:



### Public Member Functions

- `CUsbDeviceConfigurationFunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ pUsbDeviceConfigurationFunctionPointerContainer)`  
*Initializes a new instance of the `CUsbDeviceConfigurationFunctionNet` class.*
- `CUsbDeviceConfigurationFunctionNet (CMcsUsbNet^ mcsusb)`
- `virtual ~CUsbDeviceConfigurationFunctionNet ()`
- `!CUsbDeviceConfigurationFunctionNet ()`
- `void SetDeviceName (String^ name)`  
*sets the USB device name for configurable devices*
- `void SetDeviceId (ProductIdEnumNet id)`  
*sets the USB device name for configurable devices*

### Additional Inherited Members

### 11.126.1 Detailed Description

`CUsbDeviceConfigurationFunctionNet` is the class to configure the USB firmware

### 11.126.2 Constructor & Destructor Documentation

#### 11.126.2.1 CUsbDeviceConfigurationFunctionNet() [1/2] `CUsbDeviceConfigurationFunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ pUsbDeviceConfigurationFunctionPointerContainer)`

Initializes a new instance of the `CUsbDeviceConfigurationFunctionNet` class.

**11.126.2.2 CUsbDeviceConfigurationFunctionNet()** [2/2] `CUsbDeviceConfigurationFunctionNet ( CMcsUsbNet^ mcsusb )`

**11.126.2.3 ~CUsbDeviceConfigurationFunctionNet()** `virtual ~CUsbDeviceConfigurationFunctionNet ( ) [virtual]`

**11.126.2.4 !CUsbDeviceConfigurationFunctionNet()** `!CUsbDeviceConfigurationFunctionNet ( )`

### 11.126.3 Member Function Documentation

**11.126.3.1 SetDeviceId()** `void SetDeviceId ( ProductIdEnumNet id )`

sets the USB device name for configurable devices

#### Parameters

|           |  |
|-----------|--|
| <i>id</i> |  |
|-----------|--|

**11.126.3.2 SetDeviceName()** `void SetDeviceName ( String^ name )`

sets the USB device name for configurable devices

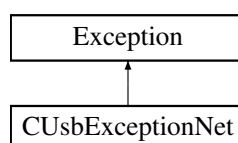
#### Parameters

|             |  |
|-------------|--|
| <i>name</i> |  |
|-------------|--|

## 11.127 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]

### 11.127.1 Detailed Description

Exception class that is thrown in case of an USB error.

### 11.127.2 Constructor & Destructor Documentation

#### 11.127.2.1 CUsbExceptionNet() [1/2] [CUsbExceptionNet](#) ( uint32\_t status )

Constructor of a CUsbException.

##### Parameters

|               |                   |
|---------------|-------------------|
| <i>status</i> | the status number |
|---------------|-------------------|

#### 11.127.2.2 CUsbExceptionNet() [2/2] [CUsbExceptionNet](#) ( uint32\_t status, String^ message )

### 11.127.3 Property Documentation

#### 11.127.3.1 Status uint32\_t Status [get]

## 11.128 CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet Class Reference

## Public Member Functions

- [CVoltageRangeInfoNet](#) (int vr, String^ vrString)



**Public Attributes**

- int [VoltageRangeInMicroVolt](#)
- String ^ [VoltageRangeDisplayStringMilliVolt](#)

**11.128.1 Constructor & Destructor Documentation**

**11.128.1.1 CVoltageRangeInfoNet()** [CVoltageRangeInfoNet](#) (  
     int *vr*,  
     String^ *vrString* )

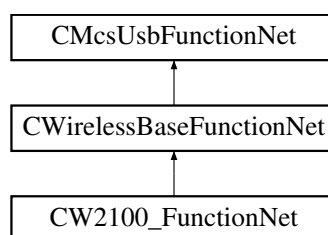
**11.128.2 Member Data Documentation**

**11.128.2.1 VoltageRangeDisplayStringMilliVolt** String ^ [VoltageRangeDisplayStringMilliVolt](#)

**11.128.2.2 VoltageRangeInMicroVolt** int [VoltageRangeInMicroVolt](#)

**11.129 CW2100\_FunctionNet Class Reference**

Inheritance diagram for CW2100\_FunctionNet:

**Classes**

- struct [AudioChannelsNet](#)

## Public Member Functions

- [CW2100\\_FunctionNet](#) ([CMcsUsbNet](#)^ mcsusb, [CMcsUsbFunctionPointerContainer](#)^ w2100\_Function↔  
PointerContainer)
- [CW2100\\_FunctionNet](#) ([CMcsUsbNet](#)^ mcsusb)
- [array< HeadStageIDType ^ > ^ GetAvailableHeadstages](#) (unsigned int max\_length)
- [void SelectHeadstage](#) (unsigned int IDorEntry, int TimeSlotNr)
- [void DeselectHeadstage](#) (int TimeSlotNr)
- [void DeselectAllHeadstages](#) ()
- [HeadStageIDTypeState ^ GetSelectedHeadstageState](#) (int TimeSlotNr)
- [BatteryState ^ GetBatteryState](#) (int TimeSlotNr)
- [System::String ^ GetUserDefinedName](#) (unsigned short ID)
- [System::String ^ GetUserDefinedNameFromSelectedHS](#) (int TimeSlotNr)
- [System::String ^ GetUserDefinedNameCache](#) (unsigned short ID)
- [uint32\\_t GetUserDefinedNameCache](#) (unsigned short ID, [System::Runtime::InteropServices::Out]System↔  
::String^% Name)
- [W2100\\_StimulusParametersNet ^ GetStiumlusParameters](#) (unsigned short ID)
- [W2100\\_StimulusParametersNet ^ GetStimulusParametersFromSelectedHS](#) (int TimeSlotNr)
- [W2100\\_StimulusParametersNet ^ GetStimulusParametersCache](#) (unsigned int typeValue)
- [uint32\\_t GetStimulusParametersCache](#) (unsigned int typeValue, [System::Runtime::InteropServices::↔  
Out]W2100\_StimulusParametersNet^% StimulusParameters)
- [void SetSelectedChannels](#) (array< BYTE >^ 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](#)<sup>^</sup> [Stimulator](#) [get]
- [CPulseGeneratorFunctionNet](#)<sup>^</sup> [PulseGenerator](#) [get]

**Additional Inherited Members****11.129.1 Constructor & Destructor Documentation**

**11.129.1.1 CW2100\_FunctionNet()** [1/2] [CW2100\\_FunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> *mcsusb*,  
[CMcsUsbFunctionPointerContainer](#)<sup>^</sup> *w2100\_FunctionPointerContainer* )

**11.129.1.2 CW2100\_FunctionNet()** [2/2] [CW2100\\_FunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> *mcsusb* )

**11.129.2 Member Function Documentation**

**11.129.2.1 ClearStimulusParametersCache()** [1/2] static void [ClearStimulusParametersCache](#) ( )  
[static]

**11.129.2.2 ClearStimulusParametersCache()** [2/2] static void [ClearStimulusParametersCache](#) (  
unsigned short *ID* ) [static]

**11.129.2.3 ClearUserDefinedNameCache()** [1/2] static void [ClearUserDefinedNameCache](#) ( ) [static]

**11.129.2.4 ClearUserDefinedNameCache()** [2/2] static void ClearUserDefinedNameCache ( unsigned short *ID* ) [static]

**11.129.2.5 DeselectAllHeadstages()** void DeselectAllHeadstages ( )

**11.129.2.6 DeselectHeadstage()** void DeselectHeadstage ( int *TimeSlotNr* )

**11.129.2.7 GetAccelGyroCurrentRate()** int GetAccelGyroCurrentRate ( int *TimeSlotNr* )

**11.129.2.8 GetAccelGyroDesiredRate()** int GetAccelGyroDesiredRate ( int *TimeSlotNr* )

**11.129.2.9 GetAccelGyroEnabled()** [W2100\\_Accel\\_Gyro\\_Select\\_EnumNet](#) GetAccelGyroEnabled ( int *TimeSlotNr* )

**11.129.2.10 GetAccelRange()** int GetAccelRange ( int *TimeSlotNr* )

**11.129.2.11 GetAnalogOutChannel()** unsigned int GetAnalogOutChannel ( [System::Runtime::InteropServices::Out] int % *automatic*, unsigned short *index* )

**11.129.2.12 GetAnalogOutFilter()** array<unsigned int> ^ GetAnalogOutFilter ( [System::Runtime::InteropServices::Out] int % *automatic* )

**11.129.2.13 GetAudioChannels()** array<[AudioChannelsNet](#)^> ^ GetAudioChannels ( )

**11.129.2.14 GetAvailableHeadstages()** array<HeadStageIDType^> ^ GetAvailableHeadstages (   
 unsigned int *max\_length* )

**11.129.2.15 GetBatteryState()** BatteryState ^ GetBatteryState (   
 int *TimeSlotNr* )

**11.129.2.16 GetDacRange()** AnalogOut\_DAC\_Range\_EnumNet GetDacRange ( )

**11.129.2.17 GetFilterProperties()** array<CFilterPropertyNet^> ^ GetFilterProperties (   
 W2100DacqGroupChannelEnumNet *GroupID* )

**11.129.2.18 GetFilterProperty()** CFilterPropertyNet ^ GetFilterProperty (   
 W2100DacqGroupChannelEnumNet *GroupID*,   
 unsigned int *index* )

**11.129.2.19 GetFPGA FirmwareType()** unsigned int GetFPGA FirmwareType (   
 int *TimeSlotNr* )

**11.129.2.20 GetGyroRange()** int GetGyroRange (   
 int *TimeSlotNr* )

**11.129.2.21 GetHeadstageOnOff()** unsigned short GetHeadstageOnOff (   
 int *TimeSlotNr* )

**11.129.2.22 GetHeadstageSamplingActive()** bool GetHeadstageSamplingActive (   
 int *TimeSlotNr* )

**11.129.2.23 GetMultiHeadstageMode()** bool GetMultiHeadstageMode ( )

**11.129.2.24 GetPicFirmwareType()** unsigned int GetPicFirmwareType (  
int *TimeSlotNr* )

**11.129.2.25 GetSelectedChannels()** array<BYTE> ^ GetSelectedChannels (  
int *TimeSlotNr* )

**11.129.2.26 GetSelectedHeadstageState()** HeadStageIDTypeState ^ GetSelectedHeadstageState (  
int *TimeSlotNr* )

**11.129.2.27 GetStimulusParametersCache()** [1/2] W2100\_StimulusParametersNet ^ GetStimulus↔  
ParametersCache (  
unsigned int *typeValue* )

**11.129.2.28 GetStimulusParametersCache()** [2/2] uint32\_t GetStimulusParametersCache (  
unsigned int *typeValue*,  
[System::Runtime::InteropServices::Out] W2100\_StimulusParametersNet^% *Stimulus↔*  
*Parameters* )

**11.129.2.29 GetStimulusParametersFromSelectedHS()** W2100\_StimulusParametersNet ^ GetStimulus↔  
ParametersFromSelectedHS (  
int *TimeSlotNr* )

**11.129.2.30 GetStiumlusParameters()** W2100\_StimulusParametersNet ^ GetStiumlusParameters (  
unsigned short *ID* )

**11.129.2.31 GetUserDefinedName()** System::String ^ GetUserDefinedName (  
unsigned short *ID* )

**11.129.2.32 GetUserDefinedNameCache()** [1/2] System::String ^ GetUserDefinedNameCache (  
unsigned short *ID* )

**11.129.2.33 GetUserDefinedNameCache()** [2/2] uint32\_t GetUserDefinedNameCache ( unsigned short *ID*, [System::Runtime::InteropServices::Out] System::String^% *Name* )

**11.129.2.34 GetUserDefinedNameFromSelectedHS()** System::String ^ GetUserDefinedNameFromSelectedHS ( int *TimeSlotNr* )

**11.129.2.35 SelectHeadstage()** void SelectHeadstage ( unsigned int *IDorEntry*, int *TimeSlotNr* )

**11.129.2.36 SetAccelGyroDesiredRate()** void SetAccelGyroDesiredRate ( int *rate*, int *TimeSlotNr* )

**11.129.2.37 SetAccelGyroEnabled()** void SetAccelGyroEnabled ( W2100\_Accel\_Gyro\_Select\_EnumNet *enable*, int *TimeSlotNr* )

**11.129.2.38 SetAccelRange()** void SetAccelRange ( int *range*, int *TimeSlotNr* )

**11.129.2.39 SetAnalogOutChannel()** void SetAnalogOutChannel ( int *automatic*, unsigned short *index*, unsigned int *Channel* )

**11.129.2.40 SetAnalogOutFilter()** void SetAnalogOutFilter ( int *automatic*, array< unsigned int >^ *Coeffs* )

**11.129.2.41 SetAudioChannels()** void SetAudioChannels (   
array< [AudioChannelsNet](#)> channels )

**11.129.2.42 SetDacRange()** void SetDacRange (   
[AnalogOut\\_DAC\\_Range\\_EnumNet](#) range )

**11.129.2.43 SetGyroRange()** void SetGyroRange (   
int range,   
int TimeSlotNr )

**11.129.2.44 SetHeadstageOnOff()** void SetHeadstageOnOff (   
unsigned short On,   
int TimeSlotNr )

**11.129.2.45 SetHeadstageSamplingActive()** void SetHeadstageSamplingActive (   
bool Active,   
int TimeSlotNr )

**11.129.2.46 SetHeadstageToSleep()** void SetHeadstageToSleep (   
unsigned int Sleep16ms,   
int TimeSlotNr )

**11.129.2.47 SetMultiHeadstageMode()** void SetMultiHeadstageMode (   
bool Mode )

**11.129.2.48 SetSelectedChannels()** void SetSelectedChannels (   
array< BYTE > channels,   
int TimeSlotNr )

## 11.129.3 Property Documentation

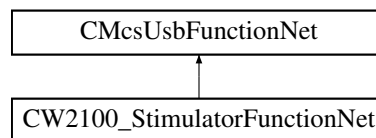


**11.129.3.1 PulseGenerator** [CPulseGeneratorFunctionNet](#)<sup>^</sup> PulseGenerator [get]

**11.129.3.2 Stimulator** [CW2100\\_StimulatorFunctionNet](#)<sup>^</sup> Stimulator [get]

## 11.130 CW2100\_StimulatorFunctionNet Class Reference

Inheritance diagram for CW2100\_StimulatorFunctionNet:



### Public Member Functions

- [CW2100\\_StimulatorFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- void [SendStart](#) (uint32\_t triggermap)  
*Start (Trigger) the STG. The startup delay is in the range of a few ms.*
- void [SendStop](#) (uint32\_t triggermap)  
*Stop some or all triggers of the STG.*
- [CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData](#)<sup>^</sup> [PrepareData](#) (int channel, array< int32\_t ><sup>^</sup> amplitude, array< uint64\_t ><sup>^</sup> duration, [STG\\_DestinationEnumNet](#) destType, uint32\_t repeat)
- [CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData](#)<sup>^</sup> [PrepareDataSync](#) (int channel, array< int32\_t ><sup>^</sup> amplitude, array< uint32\_t ><sup>^</sup> Sync, array< uint64\_t ><sup>^</sup> duration, [STG\\_DestinationEnumNet](#) destType, uint32\_t repeat)
- void [SendPreparedData](#) (int channel, [CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData](#)<sup>^</sup> device\_data\_and\_unrolled, [STG\\_DestinationEnumNet](#) destType)
- void [ClearChannelData](#) (int channel)  
*Delete a Stimulus Pattern from STG memory*
- int [GetDACResolution](#) ()  
*Gets number of bits of the DAC resolution.*
- int [GetTimeResolutionInNanoSeconds](#) ()  
*Gets number of bits of the DAC resolution.*
- int [GetVoltageRangeInMicroVolt](#) (uint32\_t channel)  
*Gets the Voltage Range of the specified channel in Microvolts.*
- int [GetVoltageResolutionInMicroVolt](#) (uint32\_t channel)  
*Gets the Voltage Resolution of the specified channel in Microvolts.*
- int [GetCurrentRangeInNanoAmp](#) (uint32\_t channel)  
*Gets the Current Range of the specified channel in Nanoamps.*
- int [GetCurrentResolutionInNanoAmp](#) (uint32\_t channel)  
*Gets the Current Resolution of the specified channel in Nanoamps.*
- int [GetCurrentResolutionInPicoAmp](#) (uint32\_t channel)  
*Gets the Current Resolution of the specified channel in Picoamps.*
- 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

#### 11.130.1 Constructor & Destructor Documentation

**11.130.1.1 [CW2100\\_StimulatorFunctionNet\(\)](#)** [CW2100\\_StimulatorFunctionNet](#) ( [CMcsUsbNet](#)<sup>^</sup> *mcsusb* )

#### 11.130.2 Member Function Documentation

**11.130.2.1 [ClearChannelData\(\)](#)** void [ClearChannelData](#) ( int *channel* )

Delete a Stimulus Pattern from STG memory

#### Parameters

|                |                                 |
|----------------|---------------------------------|
| <i>channel</i> | specifies the channel to clear. |
|----------------|---------------------------------|

**11.130.2.2 GetBoostAlwaysOnMode()** `uint32_t GetBoostAlwaysOnMode ( )`

**11.130.2.3 GetBoostPreTime()** `uint32_t GetBoostPreTime ( )`

**11.130.2.4 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.

**11.130.2.5 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.

**11.130.2.6 GetCurrentResolutionInPicoAmp()** `int GetCurrentResolutionInPicoAmp (   
uint32_t channel )`

Gets the Current Resolution of the specified channel in Picoamps.

**Parameters**

|                |                           |
|----------------|---------------------------|
| <i>channel</i> | Channel which is queried. |
|----------------|---------------------------|

**Returns**

The Current Resolution of the specified channel in Picoamps.

**11.130.2.7 GetDACResolution()** `int GetDACResolution ( )`

Gets number of bits of the DAC resolution.

**Returns**

The DAC resolution in bits.

**11.130.2.8 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 )`

**11.130.2.9 GetDigitalStimulatorTriggerSlope()** `DigitalStimulatorTriggerSlopeEnumNet GetDigital↔`  
`StimulatorTriggerSlope (`  
    `int TimeSlotNr,`  
    `DigitalStimulatorTriggerEventEnumNet trigger_event,`  
    `int trigger_number )`

**11.130.2.10 GetNumberOfAnalogChannels()** `uint32_t GetNumberOfAnalogChannels ( )`

**11.130.2.11 GetNumberOfSyncoutChannels()** `uint32_t GetNumberOfSyncoutChannels ( )`

**11.130.2.12 GetNumberOfTriggerInputs()** `uint32_t GetNumberOfTriggerInputs ( )`

**11.130.2.13 GetStimulationPatternMemory()** `uint32_t GetStimulationPatternMemory ( )`

**11.130.2.14 GetTimeResolutionInNanoSeconds()** `int GetTimeResolutionInNanoSeconds ( )`

Gets number of bits of the DAC resolution.

#### Returns

The time resolution in ns.

**11.130.2.15 GetTimeSlot()** `int GetTimeSlot ( )`

**11.130.2.16 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.

**11.130.2.17 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.

**11.130.2.18 PrepareData()** `CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData ^ PrepareData (`  
`int channel,`  
`array< int32_t >^ amplitude,`  
`array< uint64_t >^ duration,`  
`STG_DestinationEnumNet destType,`  
`uint32_t repeat )`

**11.130.2.19 PrepareDataSync()** `CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData ^ PrepareDataSync (`  
`int channel,`  
`array< int32_t >^ amplitude,`  
`array< uint32_t >^ Sync,`  
`array< uint64_t >^ duration,`  
`STG_DestinationEnumNet destType,`  
`uint32_t repeat )`

**11.130.2.20 SelectTimeSlot()** `void SelectTimeSlot (`  
`int TimeSlotNr )`

**11.130.2.21 SendPreparedData()** `void SendPreparedData (`  
`int channel,`  
`CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData^ device_data_and_unrolled,`  
`STG_DestinationEnumNet destType )`

**11.130.2.22 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. |
|-------------------|---------------------------------------------|

**11.130.2.23 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. |
|-------------------|---------------------------------------------|

**11.130.2.24 SetDigitalStimulatorTrigger()** void SetDigitalStimulatorTrigger (   
     int *TimeSlotNr*,   
     DigitalStimulatorTriggerEventEnumNet *trigger\_event*,   
     int *trigger\_number*,   
     W2100DigitalSourceEnumNet *digstream\_source*,   
     int *bitnumber\_offset* )

**11.130.2.25 SetDigitalStimulatorTriggerSlope()** void SetDigitalStimulatorTriggerSlope (   
     int *TimeSlotNr*,   
     DigitalStimulatorTriggerEventEnumNet *trigger\_event*,   
     int *trigger\_number*,   
     DigitalStimulatorTriggerSlopeEnumNet *slope* )

**11.130.2.26 StartPoll()** void StartPoll ( )

**11.130.2.27 StopPoll()** void StopPoll ( )

**11.130.3 Member Data Documentation**

**11.130.3.1 BOOST\_BIT** const uint32\_t BOOST\_BIT = (1 << 0) [static]

**11.130.3.2 GND\_SWITCH\_BIT** const uint32\_t GND\_SWITCH\_BIT = (1 << 1) [static]

**11.130.3.3 SYNC\_BIT0** const uint32\_t SYNC\_BIT0 = (1 << 2) [static]

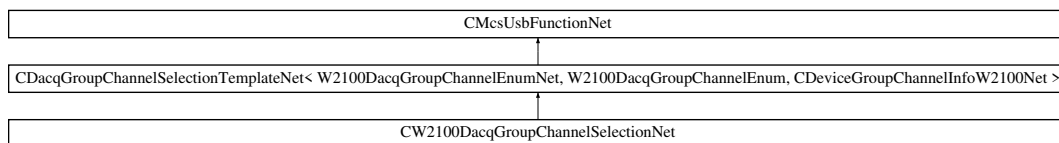
**11.130.3.4 SYNC\_BIT1** const uint32\_t SYNC\_BIT1 = (1 << 3) [static]

## 11.130.4 Event Documentation

### 11.130.4.1 PollStatusEvent `OnStgPollStatus^ PollStatusEvent`

## 11.131 CW2100DacqGroupChannelSelectionNet Class Reference

Inheritance diagram for CW2100DacqGroupChannelSelectionNet:



### Public Member Functions

- `CW2100DacqGroupChannelSelectionNet (CMcsUsbNet^ mcsusb)`

### Additional Inherited Members

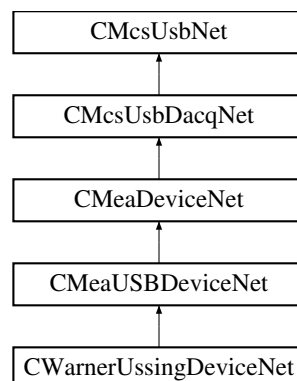
### 11.131.1 Constructor & Destructor Documentation

#### 11.131.1.1 CW2100DacqGroupChannelSelectionNet() `CW2100DacqGroupChannelSelectionNet (CMcsUsbNet^ mcsusb)`

## 11.132 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

### 11.132.1 Detailed Description

[CWarnerUssingDeviceNet](#) is the class to control the Ussing device

### 11.132.2 Constructor & Destructor Documentation

#### 11.132.2.1 CWarnerUssingDeviceNet() [CWarnerUssingDeviceNet](#) ( )

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

#### 11.132.2.2 ~CWarnerUssingDeviceNet() virtual [~CWarnerUssingDeviceNet](#) ( ) [virtual]

#### 11.132.2.3 "!CWarnerUssingDeviceNet() [!CWarnerUssingDeviceNet](#) ( )

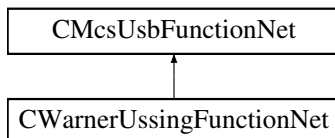
### 11.132.3 Property Documentation

#### 11.132.3.1 WarnerUssingFunction [CWarnerUssingFunctionNet](#)<sup>^</sup> [WarnerUssingFunction](#) [get]

## 11.133 CWarnerUssingFunctionNet Class Reference

[CWarnerUssingFunctionNet](#) is the class to control the Ussing device

Inheritance diagram for CWarnerUssingFunctionNet:



### Public Member Functions

- [CWarnerUssingFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> pWarner↔  
UssingFunctionPointerContainer)  
*Initializes a new instance of the [CWarnerUssingFunctionNet](#) class.*
- [CWarnerUssingFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- virtual [~CWarnerUssingFunctionNet](#) ()
- [ICWarnerUssingFunctionNet](#) ()
- [int32\\_t](#) [GetChannelsCountOfChamber](#) ([int32\\_t](#) ChamberId)  
*gets number of channels in datastream from chamber amp with given index*
- [int32\\_t](#) [GetNumberOfHardwareSlotsForChambers](#) ()  
*gets number of physical hardware slots for chambers amps*
- [int32\\_t](#) [GetNumberOfAvailableChambers](#) ()  
*gets number of actually connected chamber amps*
- bool [IsChamberAvailable](#) ([int32\\_t](#) ChamberId)  
*checks whether chamber amp is connected to slot*
- void [SetPulse](#) ([int32\\_t](#) ChamberId, [UssingClampModeEnumNet](#) StgMode, [int32\\_t](#) NumberOfRepetitions,  
array< [int](#) ><sup>^</sup> Amplitudes, array< [int](#) ><sup>^</sup> Durations, array< [int](#) ><sup>^</sup> 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*
- bool [WasInternalCalibrationSuccessful](#) ([int32\\_t](#) ChamberId)

- checks whether internal calibration was successful, i.e. no strange reference voltage readings etc...*
- `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 Uc is above, current source is overloaded*
- `bool CompensateElectrodeOffset (int32_t ChamberId)`  
*blocking call to compensate electrode offset of one chamber; returns true when successful*
- `bool WaitForChamber (int32_t ChamberId)`  
*blocking call that waits for chamber boot-up calibration to complete*
- `bool WaitForAllChambers ()`  
*blocking call that waits for ALL chambers' boot-up calibration to complete*

## Additional Inherited Members

### 11.133.1 Detailed Description

[CWarnerUssingFunctionNet](#) is the class to control the Ussing device

### 11.133.2 Constructor & Destructor Documentation

**11.133.2.1 CWarnerUssingFunctionNet()** [1/2] [CWarnerUssingFunctionNet](#) (   
 [CMcsUsbNet](#)<sup>^</sup> *mcsusb*,   
 [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> *pWarnerUssingFunctionPointerContainer* )

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

**11.133.2.2 CWarnerUssingFunctionNet()** [2/2] [CWarnerUssingFunctionNet](#) (   
 [CMcsUsbNet](#)<sup>^</sup> *mcsusb* )

**11.133.2.3 ~CWarnerUssingFunctionNet()** `virtual ~CWarnerUssingFunctionNet ( ) [virtual]`

**11.133.2.4 "!CWarnerUssingFunctionNet()** `!CWarnerUssingFunctionNet ( )`

### 11.133.3 Member Function Documentation

**11.133.3.1 CompensateElectrodeOffset()** `bool CompensateElectrodeOffset (   
 int32_t ChamberId )`

blocking call to compensate electrode offset of one chamber; returns true when successful

#### Parameters

|                        |                                             |
|------------------------|---------------------------------------------|
| <i>Chamber↔<br/>Id</i> | index of hardware chamber slot (zero-based) |
|------------------------|---------------------------------------------|

**Returns**

true if compensation succeeded

**11.133.3.2 GetAvailableChambers()** `array<int> ^ GetAvailableChambers ( )`

returns array with (zero-based) ChamberIds of all available chambers

**11.133.3.3 GetChannelsCountOfChamber()** `int32_t GetChannelsCountOfChamber (   
int32_t ChamberId )`

gets number of channels in datastream from chamber amp with given index

**Parameters**

|                  |                                             |
|------------------|---------------------------------------------|
| <i>ChamberId</i> | index of hardware chamber slot (zero-based) |
|------------------|---------------------------------------------|

**Returns**

return value of zero means that amp is not placed

**11.133.3.4 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>ChamberId</i> | index of hardware chamber slot (zero-based) |
|------------------|---------------------------------------------|

**Returns**

the current clamp mode

**11.133.3.5 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

**11.133.3.6 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

**11.133.3.7 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

**11.133.3.8 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

**11.133.3.9 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

**11.133.3.10 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

**11.133.3.11 GetLiquidResistance()** `int32_t GetLiquidResistance (int32_t ChamberId )`

gets the resistance of the liquid

**Parameters**

|                        |                                             |
|------------------------|---------------------------------------------|
| <i>Chamber↔<br/>Id</i> | index of hardware chamber slot (zero-based) |
|------------------------|---------------------------------------------|

**Returns**

the liquid resistance in ohm

**11.133.3.12 GetLowCurrentRange()** `uint32_t GetLowCurrentRange (`  
`int32_t ChamberId )`

query the range of the low current mode

**Parameters**

|                        |                                             |
|------------------------|---------------------------------------------|
| <i>Chamber↔<br/>Id</i> | index of hardware chamber slot (zero-based) |
|------------------------|---------------------------------------------|

**Returns**

low current range in nA

**11.133.3.13 GetNumberOfAvailableChambers()** `int32_t GetNumberOfAvailableChambers ( )`

gets number of actually connected chamber amps

**Returns**

the number of actually connected chambers

**11.133.3.14 GetNumberOfHardwareSlotsForChambers()** `int32_t GetNumberOfHardwareSlotsFor↔`  
`Chambers ( )`

gets number of physical hardware slots for chambers amps

**Returns**

the number of hardware chamber slots on the backplane

**11.133.3.15 GetU1Offset()** `int32_t GetU1Offset (`  
`int32_t ChamberId )`

- diagnostic function only -



## Parameters

|                        |                                             |
|------------------------|---------------------------------------------|
| <i>Chamber↔<br/>Id</i> | index of hardware chamber slot (zero-based) |
|------------------------|---------------------------------------------|

## Returns

U1 offset

**11.133.3.16 GetU1Reference()** `int32_t GetU1Reference (`  
`int32_t ChamberId )`

- diagnostic function only -

## Parameters

|                        |                                             |
|------------------------|---------------------------------------------|
| <i>Chamber↔<br/>Id</i> | index of hardware chamber slot (zero-based) |
|------------------------|---------------------------------------------|

## Returns

U1 reference

**11.133.3.17 GetU2Offset()** `int32_t GetU2Offset (`  
`int32_t ChamberId )`

- diagnostic function only -

## Parameters

|                        |                                             |
|------------------------|---------------------------------------------|
| <i>Chamber↔<br/>Id</i> | index of hardware chamber slot (zero-based) |
|------------------------|---------------------------------------------|

## Returns

U2 offset

**11.133.3.18 GetU2Reference()** `int32_t GetU2Reference (`  
`int32_t ChamberId )`

- diagnostic function only -

**Parameters**

|                        |                                             |
|------------------------|---------------------------------------------|
| <i>Chamber↔<br/>Id</i> | index of hardware chamber slot (zero-based) |
|------------------------|---------------------------------------------|

**Returns**

U2 reference

```
11.133.3.19 GetUnitDescription() String ^ GetUnitDescription (
 int32_t ChamberId,
 int32_t ChannelId)
```

gets the description for the unit

**Parameters**

|                        |                                             |
|------------------------|---------------------------------------------|
| <i>Chamber↔<br/>Id</i> | index of hardware chamber slot (zero-based) |
| <i>ChannelId</i>       | index of channel (zero-based)               |

**Returns**

the description of the unix

```
11.133.3.20 GetUnitExponent() int32_t GetUnitExponent (
 int32_t ChamberId,
 int32_t ChannelId)
```

gets the unit exponent for specified chamber and channel

**Parameters**

|                        |                                             |
|------------------------|---------------------------------------------|
| <i>Chamber↔<br/>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

**11.133.3.21 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

**11.133.3.22 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

**11.133.3.23 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

**11.133.3.24 GetVoltageClampControllerParam\_D()** `uint32_t GetVoltageClampControllerParam_D (`  
`int32_t ChamberId )`

gets D value of PID controller;

**Parameters**

|                        |                                             |
|------------------------|---------------------------------------------|
| <i>Chamber↔<br/>Id</i> | index of hardware chamber slot (zero-based) |
|------------------------|---------------------------------------------|

**Returns**

the D value

**11.133.3.25 GetVoltageClampControllerParam\_I()** `uint32_t GetVoltageClampControllerParam_I (`  
`int32_t ChamberId )`

gets I value of PID controller;

**Parameters**

|                        |                                             |
|------------------------|---------------------------------------------|
| <i>Chamber↔<br/>Id</i> | index of hardware chamber slot (zero-based) |
|------------------------|---------------------------------------------|

**Returns**

the I value

**11.133.3.26 GetVoltageClampControllerParam\_P()** `uint32_t GetVoltageClampControllerParam_P (`  
`int32_t ChamberId )`

gets P value of PID controller;

**Parameters**

|                        |                                             |
|------------------------|---------------------------------------------|
| <i>Chamber↔<br/>Id</i> | index of hardware chamber slot (zero-based) |
|------------------------|---------------------------------------------|

**Returns**

the P value

**11.133.3.27 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

**11.133.3.28 IsHighCurrentMode()** `bool IsHighCurrentMode (`  
`int32_t ChamberId )`

#### Parameters

|                  |                                             |
|------------------|---------------------------------------------|
| <i>ChamberId</i> | index of hardware chamber slot (zero-based) |
|------------------|---------------------------------------------|

#### Returns

true if in high current mode

**11.133.3.29 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

**11.133.3.30 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

**11.133.3.31 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                       |

**11.133.3.32 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 |

**11.133.3.33 SetHighCurrentMode()** `void SetHighCurrentMode (`  
    `int32_t ChamberId )`

switch to high-current mode

## Parameters

|                        |                                             |
|------------------------|---------------------------------------------|
| <i>Chamber↵<br/>Id</i> | index of hardware chamber slot (zero-based) |
|------------------------|---------------------------------------------|

**11.133.3.34 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                              |

**11.133.3.35 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                |

**11.133.3.36 SetLowCurrentMode()** void SetLowCurrentMode (   
     int32\_t *ChamberId* )

switch to low-current mode

## Parameters

|                        |                                             |
|------------------------|---------------------------------------------|
| <i>Chamber↵<br/>Id</i> | index of hardware chamber slot (zero-based) |
|------------------------|---------------------------------------------|

**11.133.3.37 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")        |

**11.133.3.38 SetVoltageClampControllerParam\_D()** void SetVoltageClampControllerParam\_D (   
int32\_t ChamberId,   
uint32\_t D )

sets D value of PID controller;

#### Parameters

|                        |                                             |
|------------------------|---------------------------------------------|
| <i>Chamber↔<br/>Id</i> | index of hardware chamber slot (zero-based) |
| <i>D</i>               | useful range: 0..700                        |

**11.133.3.39 SetVoltageClampControllerParam\_I()** void SetVoltageClampControllerParam\_I (   
int32\_t ChamberId,   
uint32\_t I )

sets I value of PID controller;

#### Parameters

|                        |                                             |
|------------------------|---------------------------------------------|
| <i>Chamber↔<br/>Id</i> | index of hardware chamber slot (zero-based) |
| <i>I</i>               | useful range: 80000..120000                 |



**11.133.3.40 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                        |

**11.133.3.41 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...)

**11.133.3.42 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...)

**11.133.3.43 WasInternalCalibrationSuccessful()** `bool WasInternalCalibrationSuccessful (`  
    `int32_t ChamberId )`

checks whether internal calibration was successful, i.e. no strange reference voltage readings etc...

#### Parameters

|                  |                                             |
|------------------|---------------------------------------------|
| <i>ChamberId</i> | index of hardware chamber slot (zero-based) |
|------------------|---------------------------------------------|

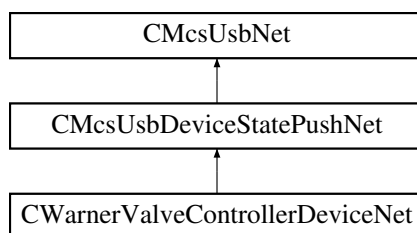
## Returns

true if successful

## 11.134 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](#) ()
- [!CWarnerValveControllerDeviceNet](#) ()
- int [GetValveActive](#) (uint16\_t valve)
- *Gets the valve active/inactive state*
- void [SetValveActive](#) (uint16\_t valve, int valveActive)
- *Sets the valve active/inactive state*
- uint32\_t [GetValvesActiveMap](#) ()
- *Gets the valves active/inactive states*

- void [SetValvesActiveMap](#) (uint32\_t valvesActive)  
*Sets the valve active/inactive state*
- int32\_t [GetValveManualState](#) (uint16\_t valve)  
*Gets the valve manual on/off state*
- void [SetValveManualState](#) (uint16\_t valve, int32\_t valveManualState)  
*Sets the valve manual on/off state*
- uint32\_t [GetValvesManualStateMap](#) ()  
*Gets the valves manual on/off states*
- void [SetValvesManualStateMap](#) (uint32\_t valveaManualState)  
*Sets the valve manual on/off state*
- int32\_t [GetValveManualGroup](#) (uint16\_t valve)  
*Gets the valve manual group*
- void [SetValveManualGroup](#) (uint16\_t valve, int32\_t valveManualGroup)  
*Sets the valve manual group*
- [WvcValveModeEnumNet](#) [GetValveMode](#) (uint16\_t valve)  
*Reads the valve mode*
- void [SetValveMode](#) (uint16\_t valve, [WvcValveModeEnumNet](#) ValveMode)  
*Writes the valve mode*
- int32\_t [GetAnalogThresholdLow](#) (uint16\_t valve)  
*Gets the lower threshold for the analog in port per valve*
- void [SetAnalogThresholdLow](#) (uint16\_t valve, int32\_t threshold)  
*Sets the lower threshold for the analog in port per valve*
- int32\_t [GetAnalogThresholdHigh](#) (uint16\_t valve)  
*Gets the upper threshold for the analog in port per valve*
- void [SetAnalogThresholdHigh](#) (uint16\_t valve, int32\_t threshold)  
*Sets the upper threshold for the analog in port per valve*
- [PortDirectionEnumNet](#) [GetDigitalPortDirection](#) (uint16\_t port)  
*Gets the direction of a digital port*
- void [SetDigitalPortDirection](#) (uint16\_t port, [PortDirectionEnumNet](#) direction)  
*Sets the direction of a digital port*
- bool [IsValveDigitalInInverted](#) (uint16\_t valve)  
*Is digital in inverted*
- void [SetValveDigitalInInvert](#) (uint16\_t valve, bool isInverted)  
*Invert digital in*
- uint32\_t [GetValveDigitalInPort](#) (uint16\_t valve)  
*Gets the number of the digital in port which is mapped to a valve*
- void [SetValveDigitalInPort](#) (uint16\_t valve, uint32\_t digitalInPort)  
*Map a digital in port to a valve*
- bool [IsDigitalOutPortInverted](#) (uint16\_t digitalOutPort)  
*Gets the number of the valve which is mapped to a digital out port*
- void [SetDigitalOutPortInvert](#) (uint16\_t digitalOutPort, bool isInverted)  
*Map a valve to a digital out port*
- uint32\_t [GetDigitalOutPortValve](#) (uint16\_t digitalOutPort)  
*Gets the number of the valve which is mapped to a digital out port*
- void [SetDigitalOutPortValve](#) (uint16\_t digitalOutPort, uint32\_t valve)  
*Map a valve to a digital out port*
- void [SetDefault](#) ()  
*Sets the settings of the valve controller to default*
- bool [IsValveOpen](#) (uint16\_t valve)  
*Is valve open*
- bool [IsValveOpenInDigitalMode](#) (uint16\_t valve)

- True, if the valve would be open when the device is in digital mode*
  - bool [IsValveOpenInAnalogMode](#) (uint16\_t valve)
- True, if the valve would be open when the device is in analog mode*
  - int32\_t [GetAnalogVoltage](#) ()
- Reads the voltage on the analog in port*
  - void [GetValveTableEntry](#) (uint16\_t valve, uint16\_t index, [System::Runtime::InteropServices::Out]uint32\_t% duration, [System::Runtime::InteropServices::Out]bool% state)
- Read an entry from the valve protocol table*
  - void [SetValveTableEntry](#) (uint16\_t valve, uint16\_t index, uint32\_t duration, bool state)
- Write an entry to the valve protocol table*
  - void [ClearValveTable](#) (uint16\_t valve)
- Clear the valve protocol table*
  - void [LoadValveTable](#) ()
- Load the current table from permanent memory*
  - void [StoreValveTable](#) ()
- Store the current table in permanent memory*
  - String ^ [GetTableNamebyIndex](#) (uint16\_t tableNumber)
- Get the name of a protocol table*
  - String ^ [GetTableName](#) ()
- Get the name of the current protocol table*
  - void [SetTableName](#) (String^ tableName)
- Set the name of the current protocol table*
  - uint32\_t [GetActiveRunningTableNumber](#) ()
- Gets the number of the table that is active for running*
  - void [SetActiveRunningTableNumber](#) (uint32\_t tableNumber)
- Sets the number of the tanle that is active for running*
  - uint32\_t [GetCurrentEditTableNumber](#) ()
- Gets the number of the table that is current for editing*
  - void [SetCurrentEditTableNumber](#) (uint32\_t tableNumber)
- Sets the number of the table that is current for editing*
  - void [ClearTableName](#) ()
- Clear the name of current protocol table*
  - void [SetTableStep](#) (uint16\_t valve, int32\_t steps)
- Skips the table protocol for a valve by steps*
  - void [SetTableStepAll](#) (int32\_t steps)
- Skips the table protocol for all valves by steps*
  - int32\_t [GetTotalNumberOfValves](#) ()
- Get the total number of valves in the system*
  - int32\_t [GetTotalNumberOfDigitalPorts](#) ()
- Get the total number of digital ports in the system*
  - int32\_t [GetTotalTableSize](#) ()
- Get the total table size in the system*
  - int32\_t [GetTotalNumberOfTables](#) ()
- Get the total number of tables in the system*
  - int32\_t [GetCurrentNumberOfValves](#) ()
- Get the current number of valves connected to the system*
  - uint32\_t [GetValveBoardRevision](#) ()
- Gets the revision code of the valve board*
  - 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*
- void [GetValveCurrent](#) ([System::Runtime::InteropServices::Out]int16\_t% switch\_current, [System::Runtime::InteropServices::Out]int16\_t% hold\_current)
- Gets the valve currents*
- void [SetValveCurrent](#) (int16\_t switch\_current, int16\_t hold\_current)
- Sets the valve currents different from the default*

## Events

- [OnGetValveActive](#) ^ [GetValveActiveEvent](#) [add, remove, raise]
- Event fires when the valve state for the valve number has changed*
- [OnGetValveManualState](#) ^ [GetValveManualStateEvent](#) [add, remove, raise]
- Event fires when the manual valve state for the valve number has changed*
- [OnGetValveManualGroup](#) ^ [GetValveManualGroupEvent](#) [add, remove, raise]
- Event fires when the manual valve group for the valve number has changed*
- [OnGetValveMode](#) ^ [GetValveModeEvent](#) [add, remove, raise]
- Event fires when the valve mode for the valve number has changed*
- [OnGetAnalogThresholdLow](#) ^ [GetAnalogThresholdLowEvent](#) [add, remove, raise]
- Event fires when the threshold in mV for the valve number has changed*
- [OnGetAnalogThresholdHigh](#) ^ [GetAnalogThresholdHighEvent](#) [add, remove, raise]
- Event fires when the threshold in mV for the valve number has changed*
- [OnGetDigitalPortDirection](#) ^ [GetDigitalPortDirectionEvent](#) [add, remove, raise]
- Event fires when the direction for the port number has changed*
- [OnIsValveDigitalInInverted](#) ^ [IsValveDigitalInInvertedEvent](#) [add, remove, raise]
- Event fires when is inverted for the valve number has changed*
- [OnGetValveDigitalInPort](#) ^ [GetValveDigitalInPortEvent](#) [add, remove, raise]
- Event fires when the digital in port for the valve number has changed*
- [OnIsDigitalOutPortInverted](#) ^ [IsDigitalOutPortInvertedEvent](#) [add, remove, raise]
- Event fires when is inverted for the digital out port has changed*
- [OnGetDigitalOutPortValve](#) ^ [GetDigitalOutPortValveEvent](#) [add, remove, raise]
- Event fires when the valve number for the digital out port has changed*
- [OnIsValveOpen](#) ^ [IsValveOpenEvent](#) [add, remove, raise]
- Event fires when is open for the valve number has changed*
- [OnIsValveOpenInDigitalMode](#) ^ [IsValveOpenInDigitalModeEvent](#) [add, remove, raise]
- Event fires when is open for the valve number has changed*
- [OnIsValveOpenInAnalogMode](#) ^ [IsValveOpenInAnalogModeEvent](#) [add, remove, raise]
- Event fires when is open for the valve number has changed*
- [OnGetAnalogVoltage](#) ^ [GetAnalogVoltageEvent](#) [add, remove, raise]
- Event fires when the voltage in mV has changed*
- [OnTableEntryChanged](#) ^ [TableEntryChangedEvent](#) [add, remove, raise]
- Event fires when an entry of a table changed*
- [OnGetTableNamebyIndex](#) ^ [GetTableNamebyIndexEvent](#) [add, remove, raise]
- Event fires when the name of the table for the table number has changed*
- [OnGetActiveRunningTableNumber](#) ^ [GetActiveRunningTableNumberEvent](#) [add, remove, raise]

- Event fires when the table number has changed*
- [OnGetCurrentNumberOfValves](#)^ [GetCurrentNumberOfValvesEvent](#) [add, remove, raise]
- Event fires when the number of valves has changed*
- [OnGetValveBoardRevision](#)^ [GetValveBoardRevisionEvent](#) [add, remove, raise]
- Event fires when the revision code has changed*
- [OnGetValveLedOn](#)^ [GetValveLedOnEvent](#) [add, remove, raise]
- Event fires when the LED state has changed*
- [OnGetDisplayMode](#)^ [GetDisplayModeEvent](#) [add, remove, raise]
- Event fires when the display mode has changed*

## Additional Inherited Members

### 11.134.1 Detailed Description

[CWarnerValveControllerDeviceNet](#) is the class to access the Warner Valve Controller

### 11.134.2 Constructor & Destructor Documentation

**11.134.2.1 CWarnerValveControllerDeviceNet()** [CWarnerValveControllerDeviceNet](#) ( )

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

**11.134.2.2 ~CWarnerValveControllerDeviceNet()** virtual [~CWarnerValveControllerDeviceNet](#) ( )  
[virtual]

**11.134.2.3 !CWarnerValveControllerDeviceNet()** [!CWarnerValveControllerDeviceNet](#) ( )

### 11.134.3 Member Function Documentation

**11.134.3.1 ClearTableName()** void ClearTableName ( )

Clear the name of current protocol table

**11.134.3.2 ClearValveTable()** void ClearValveTable (   
uint16\_t valve )

Clear the valve protocol table

## Parameters

|              |                  |
|--------------|------------------|
| <i>valve</i> | The valve number |
|--------------|------------------|

**11.134.3.3 GetActiveRunningTableNumber()** `uint32_t GetActiveRunningTableNumber ( )`

Gets the number of the table that is active for running

## Returns

The table number

**11.134.3.4 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

**11.134.3.5 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

**11.134.3.6 GetAnalogVoltage()** `int32_t GetAnalogVoltage ( )`

Reads the voltage on the analog in port

**Returns**

The voltage in mV

**11.134.3.7 GetCurrentEditTableNumber()** `uint32_t GetCurrentEditTableNumber ( )`

Gets the number of the table that is current for editing

**Returns**

The table number

**11.134.3.8 GetCurrentNumberOfValves()** `int32_t GetCurrentNumberOfValves ( )`

Get the current number of valves connected to the system

**Returns**

The number of valves

**11.134.3.9 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

**11.134.3.10 GetDigitalPortDirection()** `PortDirectionEnumNet GetDigitalPortDirection (   
uint16_t port )`

Gets the direction of a digital port



**Parameters**

|             |                 |
|-------------|-----------------|
| <i>port</i> | The port number |
|-------------|-----------------|

**Returns**

the direction

**11.134.3.11 GetDisplayMode()** `WvcDisplayModeEnumNet GetDisplayMode ( )`

Reads the display mode

**Returns**

The display mode

**11.134.3.12 GetTableName()** `String ^ GetTableName ( )`

Get the name of the current protocol table

**Returns**

The name of the table

**11.134.3.13 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

**11.134.3.14 GetTotalNumberOfDigitalPorts()** `int32_t GetTotalNumberOfDigitalPorts ( )`

Get the total number of digital ports in the system

**Returns**

The number of digital ports

**11.134.3.15 GetTotalNumberOfTables()** `int32_t GetTotalNumberOfTables ( )`

Get the total number of tables in the system

**Returns**

The number of tables

**11.134.3.16 GetTotalNumberOfValves()** `int32_t GetTotalNumberOfValves ( )`

Get the total number of valves in the system

**Returns**

The number of valves

**11.134.3.17 GetTotalTableSize()** `int32_t GetTotalTableSize ( )`

Get the total table size in the system

**Returns**

The table size

**11.134.3.18 GetValveActive()** `int GetValveActive (   
 uint16_t valve )`

Gets the valve active/inactive state

**Parameters**

|              |                  |
|--------------|------------------|
| <i>valve</i> | The valve number |
|--------------|------------------|

**Returns**

The valve state

**11.134.3.19 GetValveBoardRevision()** `uint32_t GetValveBoardRevision ( )`

Gets the revision code of the valve board

**Returns**

The revision code

**11.134.3.20 GetValveBoardRevisionString()** `String ^ GetValveBoardRevisionString ( )`

Gets the revision name of the valve board

**Returns**

The revision name

**11.134.3.21 GetValveCurrent()** `void GetValveCurrent ( [System::Runtime::InteropServices::Out] int16_t% switch_current, [System::Runtime::InteropServices::Out] int16_t% hold_current )`

Gets the valve currents

**Parameters**

|                       |                                   |
|-----------------------|-----------------------------------|
| <i>switch_current</i> | The switch current (in DAC units) |
| <i>hold_current</i>   | The hold current (in DAC units)   |

**11.134.3.22 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

**11.134.3.23 GetValveLedOn()** `bool GetValveLedOn ( )`

Gets the LED state of the valve board

**Returns**

The LED state

**11.134.3.24 GetValveManualGroup()** `int32_t GetValveManualGroup ( uint16_t valve )`

Gets the valve manual group

**Parameters**

|              |                  |
|--------------|------------------|
| <i>valve</i> | The valve number |
|--------------|------------------|

**Returns**

The manual valve group

**11.134.3.25 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

**11.134.3.26 GetValveMode()** `WvcValveModeEnumNet GetValveMode ( uint16_t valve )`

Reads the valve mode

**Parameters**

|              |                  |
|--------------|------------------|
| <i>valve</i> | The valve number |
|--------------|------------------|

**Returns**

The valve mode

**11.134.3.27 GetValvesActiveMap()** uint32\_t GetValvesActiveMap ( )

Gets the valves active/inactive states

**Returns**

The valves states

**11.134.3.28 GetValvesManualStateMap()** uint32\_t GetValvesManualStateMap ( )

Gets the valves manual on/off states

**Returns**

The manual valves states

**11.134.3.29 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              |

**11.134.3.30 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

**11.134.3.31 IsValveDigitalInInverted()** `bool IsValveDigitalInInverted (`  
`uint16_t valve )`

Is digital in inverted

**Parameters**

|              |                  |
|--------------|------------------|
| <i>valve</i> | The valve number |
|--------------|------------------|

**Returns**

is inverted

**11.134.3.32 IsValveOpen()** `bool IsValveOpen (`  
`uint16_t valve )`

Is valve open

**Parameters**

|              |                  |
|--------------|------------------|
| <i>valve</i> | The valve number |
|--------------|------------------|

**Returns**

is open

**11.134.3.33 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

**11.134.3.34 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

**11.134.3.35 LoadValveTable()** `void LoadValveTable ( )`

Load the current table from permanent memory

**11.134.3.36 OnGetActiveRunningTableNumber()** `delegate void OnGetActiveRunningTableNumber ( uint32_t tableNumber )`

**11.134.3.37 OnGetAnalogThresholdHigh()** `delegate void OnGetAnalogThresholdHigh ( uint16_t valve, int32_t threshold )`

**11.134.3.38 OnGetAnalogThresholdLow()** `delegate void OnGetAnalogThresholdLow ( uint16_t valve, int32_t threshold )`

**11.134.3.39 OnGetAnalogVoltage()** `delegate void OnGetAnalogVoltage ( int32_t voltage )`

**11.134.3.40 OnGetCurrentNumberOfValves()** delegate void OnGetCurrentNumberOfValves (   
int32\_t numberOfValves )

**11.134.3.41 OnGetDigitalOutPortValve()** delegate void OnGetDigitalOutPortValve (   
uint16\_t digitalOutPort,   
uint32\_t valve )

**11.134.3.42 OnGetDigitalPortDirection()** delegate void OnGetDigitalPortDirection (   
uint16\_t port,   
PortDirectionEnumNet direction )

**11.134.3.43 OnGetDisplayMode()** delegate void OnGetDisplayMode (   
WvcDisplayModeEnumNet DisplayMode )

**11.134.3.44 OnGetTableNamebyIndex()** delegate void OnGetTableNamebyIndex (   
uint16\_t tableNumber,   
String^ tableName )

**11.134.3.45 OnGetValveActive()** delegate void OnGetValveActive (   
uint16\_t valve,   
int valveActive )

**11.134.3.46 OnGetValveBoardRevision()** delegate void OnGetValveBoardRevision (   
uint32\_t revision )

**11.134.3.47 OnGetValveDigitalInPort()** delegate void OnGetValveDigitalInPort (   
uint16\_t valve,   
uint32\_t digitalInPort )

**11.134.3.48 OnGetValveLedOn()** delegate void OnGetValveLedOn (   
bool ledon )



**11.134.3.49 OnGetValveManualGroup()** delegate void OnGetValveManualGroup (   
 uint16\_t valve,   
 int32\_t valveManualGroup )

**11.134.3.50 OnGetValveManualState()** delegate void OnGetValveManualState (   
 uint16\_t valve,   
 int32\_t valveManualState )

**11.134.3.51 OnGetValveMode()** delegate void OnGetValveMode (   
 uint16\_t valve,   
 WvcValveModeEnumNet ValveMode )

**11.134.3.52 OnIsDigitalOutPortInverted()** delegate void OnIsDigitalOutPortInverted (   
 uint16\_t digitalOutPort,   
 bool isInverted )

**11.134.3.53 OnIsValveDigitalInInverted()** delegate void OnIsValveDigitalInInverted (   
 uint16\_t valve,   
 bool isInverted )

**11.134.3.54 OnIsValveOpen()** delegate void OnIsValveOpen (   
 uint16\_t valve,   
 bool valveOpen )

**11.134.3.55 OnIsValveOpenInAnalogMode()** delegate void OnIsValveOpenInAnalogMode (   
 uint16\_t valve,   
 bool valveOpen )

**11.134.3.56 OnIsValveOpenInDigitalMode()** delegate void OnIsValveOpenInDigitalMode (   
 uint16\_t valve,   
 bool valveOpen )

**11.134.3.57 OnTableEntryChanged()** delegate void OnTableEntryChanged (   
 uint16\_t tableNumber )

**11.134.3.58 SetActiveRunningTableNumber()** void SetActiveRunningTableNumber (   
 uint32\_t tableNumber )

Sets the number of the tanle that is active for running

**Parameters**

|                    |                  |
|--------------------|------------------|
| <i>tableNumber</i> | The table number |
|--------------------|------------------|

**11.134.3.59 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 |

**11.134.3.60 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 |

**11.134.3.61 SetCurrentEditTableNumber()** `void SetCurrentEditTableNumber (`  
    `uint32_t tableNumber )`

Sets the number of the table that is current for editing

**Parameters**

|                    |                  |
|--------------------|------------------|
| <i>tableNumber</i> | The table number |
|--------------------|------------------|

**11.134.3.62 SetDefault()** `void SetDefault ( )`

Sets the settings of the valve controller to default

**11.134.3.63 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 |

**11.134.3.64 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     |

**11.134.3.65 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   |

**11.134.3.66 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 |

**11.134.3.67 SetTableName()** `void SetTableName (`  
    `String^ tableName )`

Set the name of the current protocol table

**Parameters**

|                  |                       |
|------------------|-----------------------|
| <i>tableName</i> | The name of the table |
|------------------|-----------------------|

**11.134.3.68 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  |

**11.134.3.69 SetTableStepAll()** `void SetTableStepAll (`  
    `int32_t steps )`

Skips the table protocol for all valves by steps

**Parameters**

|              |                 |
|--------------|-----------------|
| <i>steps</i> | Number of steps |
|--------------|-----------------|

**11.134.3.70 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  |

**11.134.3.71 SetValveCurrent()** `void SetValveCurrent (`  
    `int16_t switch_current,`  
    `int16_t hold_current )`

Sets the valve currents different from the default

Parameters

|                       |                                                                       |
|-----------------------|-----------------------------------------------------------------------|
| <i>switch_current</i> | The switch current (in DAC units); -1 sets the device default current |
| <i>hold_current</i>   | The hold current (in DAC units); -1 sets the device default current   |

**11.134.3.72 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 |

**11.134.3.73 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 |

**11.134.3.74 SetValveLedOn()** `void SetValveLedOn (`  
    `bool ledon )`

Gets the LED state of the valve board

Parameters

|              |               |
|--------------|---------------|
| <i>ledon</i> | The LED state |
|--------------|---------------|

**11.134.3.75 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 |

**11.134.3.76 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 |

**11.134.3.77 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   |

**11.134.3.78 SetValvesActiveMap()** `void SetValvesActiveMap (`  
    `uint32_t valvesActive )`

Sets the valve active/inactive state

**Parameters**

|                     |                   |
|---------------------|-------------------|
| <i>valvesActive</i> | The valves states |
|---------------------|-------------------|

**11.134.3.79 SetValvesManualStateMap()** `void SetValvesManualStateMap (`  
`uint32_t valveaManualState )`

Sets the valve manual on/off state

#### Parameters

|                          |                          |
|--------------------------|--------------------------|
| <i>valveaManualState</i> | The manual valves states |
|--------------------------|--------------------------|

**11.134.3.80 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              |

**11.134.3.81 StoreValveTable()** `void StoreValveTable ( )`

Store the current table in permanent memory

## 11.134.4 Event Documentation

**11.134.4.1 GetActiveRunningTableNumberEvent** `OnGetActiveRunningTableNumber^ GetActiveRunning↔`  
`TableNumberEvent [add], [remove], [raise]`

Event fires when the table number has changed

**11.134.4.2 GetAnalogThresholdHighEvent** `OnGetAnalogThresholdHigh^ GetAnalogThresholdHighEvent`  
`[add], [remove], [raise]`

Event fires when the threshold in mV for the valve number has changed

**11.134.4.3 GetAnalogThresholdLowEvent** [OnGetAnalogThresholdLow](#)<sup>^</sup> GetAnalogThresholdLowEvent  
[add], [remove], [raise]

Event fires when the threshold in mV for the valve number has changed

**11.134.4.4 GetAnalogVoltageEvent** [OnGetAnalogVoltage](#)<sup>^</sup> GetAnalogVoltageEvent [add], [remove],  
[raise]

Event fires when the voltage in mV has changed

**11.134.4.5 GetCurrentNumberOfValvesEvent** [OnGetCurrentNumberOfValves](#)<sup>^</sup> GetCurrentNumberOf↔  
ValvesEvent [add], [remove], [raise]

Event fires when the number of valves has changed

**11.134.4.6 GetDigitalOutPortValveEvent** [OnGetDigitalOutPortValve](#)<sup>^</sup> GetDigitalOutPortValveEvent  
[add], [remove], [raise]

Event fires when the valve number for the digital out port has changed

**11.134.4.7 GetDigitalPortDirectionEvent** [OnGetDigitalPortDirection](#)<sup>^</sup> GetDigitalPortDirectionEvent  
[add], [remove], [raise]

Event fires when the direction for the port number has changed

**11.134.4.8 GetDisplayModeEvent** [OnGetDisplayMode](#)<sup>^</sup> GetDisplayModeEvent [add], [remove], [raise]

Event fires when the display mode has changed

**11.134.4.9 GetTableNamebyIndexEvent** [OnGetTableNamebyIndex](#)<sup>^</sup> GetTableNamebyIndexEvent [add],  
[remove], [raise]

Event fires when the name of the table for the table number has changed



**11.134.4.10 GetValveActiveEvent** [OnGetValveActive](#)<sup>^</sup> GetValveActiveEvent [add], [remove], [raise]

Event fires when the valve state for the valve number has changed

**11.134.4.11 GetValveBoardRevisionEvent** [OnGetValveBoardRevision](#)<sup>^</sup> GetValveBoardRevisionEvent [add], [remove], [raise]

Event fires when the revision code has changed

**11.134.4.12 GetValveDigitalInPortEvent** [OnGetValveDigitalInPort](#)<sup>^</sup> GetValveDigitalInPortEvent [add], [remove], [raise]

Event fires when the digital in port for the valve number has changed

**11.134.4.13 GetValveLedOnEvent** [OnGetValveLedOn](#)<sup>^</sup> GetValveLedOnEvent [add], [remove], [raise]

Event fires when the LED state has changed

**11.134.4.14 GetValveManualGroupEvent** [OnGetValveManualGroup](#)<sup>^</sup> GetValveManualGroupEvent [add], [remove], [raise]

Event fires when the manual valve group for the valve number has changed

**11.134.4.15 GetValveManualStateEvent** [OnGetValveManualState](#)<sup>^</sup> GetValveManualStateEvent [add], [remove], [raise]

Event fires when the manual valve state for the valve number has changed

**11.134.4.16 GetValveModeEvent** [OnGetValveMode](#)<sup>^</sup> GetValveModeEvent [add], [remove], [raise]

Event fires when the valve mode for the valve number has changed

**11.134.4.17 IsDigitalOutPortInvertedEvent** [OnIsDigitalOutPortInverted](#)<sup>^</sup> IsDigitalOutPortInverted↔ Event [add], [remove], [raise]

Event fires when is inverted for the digital out port has changed

**11.134.4.18 IsValveDigitalInInvertedEvent** [OnIsValveDigitalInInverted](#)<sup>^</sup> IsValveDigitalInInverted↔  
Event [add], [remove], [raise]

Event fires when is inverted for the valve number has changed

**11.134.4.19 IsValveOpenEvent** [OnIsValveOpen](#)<sup>^</sup> IsValveOpenEvent [add], [remove], [raise]

Event fires when is open for the valve number has changed

**11.134.4.20 IsValveOpenInAnalogModeEvent** [OnIsValveOpenInAnalogMode](#)<sup>^</sup> IsValveOpenInAnalogMode↔  
Event [add], [remove], [raise]

Event fires when is open for the valve number has changed

**11.134.4.21 IsValveOpenInDigitalModeEvent** [OnIsValveOpenInDigitalMode](#)<sup>^</sup> IsValveOpenInDigital↔  
ModeEvent [add], [remove], [raise]

Event fires when is open for the valve number has changed

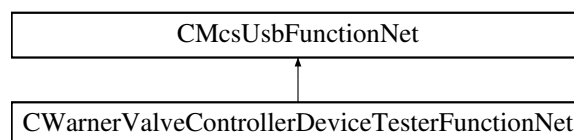
**11.134.4.22 TableEntryChangedEvent** [OnTableEntryChanged](#)<sup>^</sup> TableEntryChangedEvent [add], [remove], [raise]

Event fires when an entry of a table changed

## 11.135 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](#) (int32\_t direction)  
*Sets the direction of the IO test ports*

## Additional Inherited Members

### 11.135.1 Detailed Description

[CWarnerValveControllerDeviceTesterFunctionNet](#) is the class to access the functions for the Warner Valve Controller Device Tester

### 11.135.2 Constructor & Destructor Documentation

**11.135.2.1 CWarnerValveControllerDeviceTesterFunctionNet() [1/2]** [CWarnerValveControllerDeviceTesterFunctionNet](#)  
 (  
     [CMcsUsbNet](#)<sup>^</sup> mcsusb,  
     [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> pWarnerValveControllerDeviceTesterFunction↔  
     PointerContainer )

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

**11.135.2.2 CWarnerValveControllerDeviceTesterFunctionNet() [2/2]** [CWarnerValveControllerDeviceTesterFunctionNet](#)  
 (  
     [CMcsUsbNet](#)<sup>^</sup> mcsusb )

**11.135.2.3** `~CWarnerValveControllerDeviceTesterFunctionNet()` `virtual ~CWarnerValveControllerDeviceTesterFunctionNet()`  
( ) [virtual]

**11.135.2.4** `!CWarnerValveControllerDeviceTesterFunctionNet()` `!CWarnerValveControllerDeviceTesterFunctionNet()`  
( )

### 11.135.3 Member Function Documentation

**11.135.3.1** `GetIO()` `uint32_t GetIO ( )`

Gets the output from the io ports

#### Returns

The manual valves states

**11.135.3.2** `GetSync()` `uint32_t GetSync ( )`

Gets the output from the sync port

#### Returns

The sync state

**11.135.3.3** `SetADC()` `void SetADC (`  
`uint32_t onoff )`

Sets the ADC port of the tester

#### Parameters

|              |                |
|--------------|----------------|
| <i>onoff</i> | The port state |
|--------------|----------------|

**11.135.3.4** `SetIO()` `void SetIO (`  
`uint32_t io )`

Sets the input to the io ports

## Parameters

|           |                          |
|-----------|--------------------------|
| <i>io</i> | The manual valves states |
|-----------|--------------------------|

**11.135.3.5 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 |
|------------------|-------------------------------------|

**11.135.3.6 SetTrigger()** `void SetTrigger (`  
`uint32_t trigger )`

Sets the input to the trigger port

## Parameters

|                |                   |
|----------------|-------------------|
| <i>trigger</i> | The trigger state |
|----------------|-------------------|

**11.135.3.7 SetTriggerSyncDirection()** `void SetTriggerSyncDirection (`  
`uint32_t direction )`

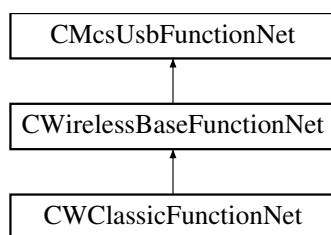
Sets the direction of the trigger/sync test port

## Parameters

|                  |                           |
|------------------|---------------------------|
| <i>direction</i> | The direction: 1=IN 0=OUT |
|------------------|---------------------------|

## 11.136 CWClassicFunctionNet Class Reference

Inheritance diagram for CWClassicFunctionNet:



## Public Member Functions

- [CWClassicFunctionNet](#) ([CMcsUsbNet](#)^ mcsusb, [CMcsUsbFunctionPointerContainer](#)^ wClassicFuntion↔  
PointerContainer)
- [CWClassicFunctionNet](#) ([CMcsUsbNet](#)^ mcsusb)
- [uint32\\_t ResetChannelmap](#) (unsigned int virtualDevice)
- [uint32\\_t SetChannelmap](#) (unsigned char position, unsigned char channel, unsigned int Device)
- void [SetHWSelectedChannels](#) (array< bool >^ 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

### 11.136.1 Constructor & Destructor Documentation

**11.136.1.1 CWClassicFunctionNet()** [1/2] `CWClassicFunctionNet ( CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ wClassicFuntionPointerContainer )`

**11.136.1.2 CWClassicFunctionNet()** [2/2] `CWClassicFunctionNet ( CMcsUsbNet^ mcsusb )`

## 11.136.2 Member Function Documentation

**11.136.2.1 GetFilterParametersHeadstage()** `array<int> ^ GetFilterParametersHeadstage ( unsigned short index )`

**11.136.2.2 GetHasChecksum()** `unsigned int GetHasChecksum ( unsigned int Device )`

**11.136.2.3 GetHasRedLedHeadstage()** `bool GetHasRedLedHeadstage ( )`

**11.136.2.4 GetHeadstageOnOff()** `USHORT GetHeadstageOnOff ( )`

**11.136.2.5 GetResetFilter()** `unsigned int GetResetFilter ( unsigned int Device )`

**11.136.2.6 GetRFConnectionStatus()** `unsigned int GetRFConnectionStatus ( )`

**11.136.2.7 GetRFFrequencyHeadstage()** `unsigned short GetRFFrequencyHeadstage ( uint8_t receiver_nb )`

**11.136.2.8 GetRFFrequencyReceiver()** unsigned short GetRFFrequencyReceiver (   
     uint8\_t *receiver\_nb*,  
     uint8\_t *configuration* )

**11.136.2.9 GetRFPower()** unsigned short GetRFPower ( )

**11.136.2.10 GetScanHeadstagesResult()** uint8\_t GetScanHeadstagesResult (   
     int *max\_wait\_for\_ms* )

**11.136.2.11 GetSelectedHeadstage()** uint8\_t GetSelectedHeadstage ( )

**11.136.2.12 GetSerialNumberHeadstage()** unsigned short GetSerialNumberHeadstage ( )

**11.136.2.13 GetWPADebugMode()** unsigned int GetWPADebugMode (   
     unsigned int *Device* )

**11.136.2.14 GetWPAType()** unsigned short GetWPAType (   
     unsigned int *Device* )

**11.136.2.15 ResetChannelmap()** uint32\_t ResetChannelmap (   
     unsigned int *virtualDevice* )

**11.136.2.16 ScanForHeadstages()** void ScanForHeadstages ( )

**11.136.2.17 SetChannelmap()** uint32\_t SetChannelmap (   
     unsigned char *position*,  
     unsigned char *channel*,  
     unsigned int *Device* )



**11.136.2.18 SetFilterParametersHeadstage()** void SetFilterParametersHeadstage (   
 unsigned short *index*,  
 array< int >^ *buffer* )

**11.136.2.19 SetHasChecksum()** void SetHasChecksum (   
 unsigned int *has*,  
 unsigned int *Device* )

**11.136.2.20 SetHeadstageOnOff()** void SetHeadstageOnOff (   
 uint16\_t *onoff* )

**11.136.2.21 SetHWSelectedChannels()** void SetHWSelectedChannels (   
 array< bool >^ *channels*,  
 unsigned int *Device* )

**11.136.2.22 SetResetFilter()** void SetResetFilter (   
 unsigned int *reset*,  
 unsigned int *Device* )

**11.136.2.23 SetRFFrequencyHeadstage()** void SetRFFrequencyHeadstage (   
 uint8\_t *receiver\_nb*,  
 unsigned short *frequency* )

**11.136.2.24 SetRFFrequencyReceiver()** void SetRFFrequencyReceiver (   
 uint8\_t *receiver\_nb*,  
 uint8\_t *configuration*,  
 unsigned short *frequency* )

**11.136.2.25 SetRFFrequencyReceiverEeprom()** void SetRFFrequencyReceiverEeprom (   
 uint8\_t *receiver\_nb*,  
 uint8\_t *configuration*,  
 unsigned short *frequency* )

**11.136.2.26 SetRFLostBehaviour()** `void SetRFLostBehaviour (`  
     `uint8_t stoponfailure,`  
     `unsigned int Device )`

**11.136.2.27 SetRFPower()** `void SetRFPower (`  
     `unsigned short power )`

**11.136.2.28 SetSelectedHeadstage()** `void SetSelectedHeadstage (`  
     `uint8_t number )`

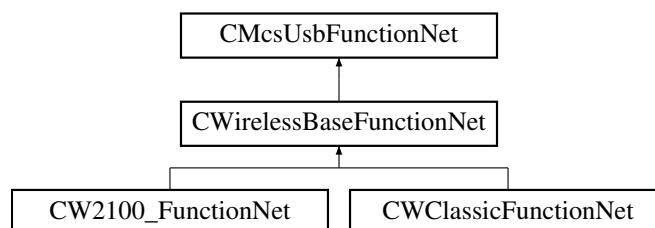
**11.136.2.29 SetSerialNumberHeadstage()** `void SetSerialNumberHeadstage (`  
     `unsigned short number )`

**11.136.2.30 SetWPADebugMode()** `void SetWPADebugMode (`  
     `unsigned int mode,`  
     `unsigned int Device )`

**11.136.2.31 SetWPAType()** `void SetWPAType (`  
     `unsigned short type,`  
     `unsigned int Device )`

## 11.137 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 ^ [CreateWirelessHeadstageSerialNumberString](#) (unsigned short ID)

## Additional Inherited Members

### 11.137.1 Constructor & Destructor Documentation

**11.137.1.1 CWirelessBaseFunctionNet()** [CWirelessBaseFunctionNet](#) (   
[CMcsUsbNet](#)^ *mcsusb*,   
[CMcsUsbFunctionPointerContainer](#)^ *mcsusbfunction* )

### 11.137.2 Member Function Documentation

**11.137.2.1 CreateWirelessHeadstageSerialNumberString()** static String ^ [CreateWirelessHeadstageSerialNumberString](#) (   
 unsigned short *ID* ) [static]

## 11.138 DevicIdNet Struct Reference

Device Id.

## Public Member Functions

- [DevicIdNet](#) ()
- [DevicIdNet](#) ([VendorIdEnumNet](#) vendor, [ProductIdEnumNet](#) product, int bcd, [McsBusTypeEnumNet](#) bustype)
- [DevicIdNet](#) ([DevicIdNet](#)% deviceId)
- [DevicIdNet](#) operator= ([DevicIdNet](#)% deviceId)

## Public Attributes

- [VendorIdEnumNet](#) IdVendor
- [ProductIdEnumNet](#) IdProduct
- int [BcdDevice](#)
- [McsBusTypeEnumNet](#) BusType

### 11.138.1 Detailed Description

Device Id.

## 11.138.2 Constructor & Destructor Documentation

**11.138.2.1 DeviceIdNet()** [1/3] `DeviceIdNet ( )`

**11.138.2.2 DeviceIdNet()** [2/3] `DeviceIdNet (`  
    `VendorIdEnumNet vendor,`  
    `ProductIdEnumNet product,`  
    `int bcd,`  
    `McsBusTypeEnumNet bustype )`

**11.138.2.3 DeviceIdNet()** [3/3] `DeviceIdNet (`  
    `DeviceIdNet% deviceId )`

## 11.138.3 Member Function Documentation

**11.138.3.1 operator=()** `DeviceIdNet operator= (`  
    `DeviceIdNet% deviceId )`

## 11.138.4 Member Data Documentation

**11.138.4.1 BcdDevice** `int BcdDevice`

**11.138.4.2 BusType** `McsBusTypeEnumNet BusType`

**11.138.4.3 IdProduct** `ProductIdEnumNet IdProduct`

**11.138.4.4 IdVendor** `VendorIdEnumNet IdVendor`

## 11.139 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]

### 11.139.1 Constructor & Destructor Documentation

**11.139.1.1 DigitalSource()** [1/2] [DigitalSource](#) ( )

**11.139.1.2 DigitalSource()** [2/2] [DigitalSource](#) (  
digitalsourceenum *source* )

### 11.139.2 Member Function Documentation

**11.139.2.1 MaxBitNumber()** [1/2] int [MaxBitNumber](#) ( )

**11.139.2.2 MaxBitNumber()** [2/2] int [MaxBitNumber](#) (  
digitalsourceenum *Source* )

**11.139.2.3 MaxBitNumberStatic()** static int [MaxBitNumberStatic](#) (  
digitalsourceenum *Source* ) [static]

**11.139.2.4 size()** `static int size ( ) [static]`

### 11.139.3 Property Documentation

**11.139.3.1 Source** `digitalsourceenum Source [get], [set]`

## 11.140 DigitalSourceGeneral Class Reference

### Public Member Functions

- [DigitalSourceGeneral](#) (Type^ type)
- [DigitalSourceGeneral](#) (Type^ type, int [Source](#))
- int [MaxBitNumber](#) ()
- int [MaxBitNumber](#) (int [Source](#))

### Static Public Member Functions

- static int [MaxBitNumber](#) (Type^ type, int [Source](#))
- static int [size](#) (Type^ type)

### Properties

- int [Source](#) [get, set]

### 11.140.1 Constructor & Destructor Documentation

**11.140.1.1 DigitalSourceGeneral()** [1/2] [DigitalSourceGeneral](#) (  
Type^ type )

**11.140.1.2 DigitalSourceGeneral()** [2/2] [DigitalSourceGeneral](#) (  
Type^ type,  
int Source )

### 11.140.2 Member Function Documentation

**11.140.2.1 MaxBitNumber()** [1/3] `int MaxBitNumber ( )`

**11.140.2.2 MaxBitNumber()** [2/3] `int MaxBitNumber (`  
`int Source )`

**11.140.2.3 MaxBitNumber()** [3/3] `static int MaxBitNumber (`  
`Type^ type,`  
`int Source ) [static]`

**11.140.2.4 size()** `static int size (`  
`Type^ type ) [static]`

### 11.140.3 Property Documentation

**11.140.3.1 Source** `int Source [get], [set]`

## 11.141 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)

#### 11.141.1 Detailed Description

Class gives firmware versions of the device's firmware destinations.

#### 11.141.2 Constructor & Destructor Documentation

##### 11.141.2.1 [DriverVersionNet\(\)](#) [DriverVersionNet](#) ( )

Constructor.

##### 11.141.2.2 [~DriverVersionNet\(\)](#) [~DriverVersionNet](#) ( )

Destructor.

#### 11.141.3 Member Function Documentation



**11.141.3.1 DriverVersionNet::FormatVersion()** `static String ^ DriverVersionNet::FormatVersion ( unsigned int v ) [static]`

**11.141.3.2 GetDestinationCode()** `CFirmwareDestinationNet GetDestinationCode ( unsigned int index )`

Get CFirmwareDestinationNet.

**Parameters**

|              |                                  |
|--------------|----------------------------------|
| <i>index</i> | by index of firmware destination |
|--------------|----------------------------------|

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

Get firmware destination name.

**Parameters**

|             |                            |
|-------------|----------------------------|
| <i>dest</i> | by CFirmwareDestinationNet |
|-------------|----------------------------|

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

Get firmware destination name.

**Parameters**

|              |                                  |
|--------------|----------------------------------|
| <i>index</i> | by index of firmware destination |
|--------------|----------------------------------|

**11.141.3.5 GetMajor()** [1/2] `unsigned int GetMajor ( CFirmwareDestinationNet dest )`

Get the major version number of firmware destination.

**Parameters**

|             |                            |
|-------------|----------------------------|
| <i>dest</i> | by CFirmwareDestinationNet |
|-------------|----------------------------|

**11.141.3.6 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 |
|--------------|----------------------------------|

**11.141.3.7 GetMinor()** [1/2] unsigned int GetMinor (   
 CFirmwareDestinationNet dest )

Get the minor version number of firmware destination.

Parameters

|             |                            |
|-------------|----------------------------|
| <i>dest</i> | by CFirmwareDestinationNet |
|-------------|----------------------------|

**11.141.3.8 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 |
|--------------|----------------------------------|

**11.141.3.9 GetNumEntries()** unsigned int GetNumEntries ( )

Get the number of available firmware destinations.

**11.141.3.10 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 |
|-------------|----------------------------|

**11.141.3.11 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 |
|--------------|----------------------------------|

**11.141.3.12 GetStatus()** [1/2] unsigned int GetStatus (   
CFirmwareDestinationNet *dest* )

Get status of firmware destination.

**Parameters**

|             |                            |
|-------------|----------------------------|
| <i>dest</i> | by CFirmwareDestinationNet |
|-------------|----------------------------|

**11.141.3.13 GetStatus()** [2/2] unsigned int GetStatus (   
unsigned int *index* )

Get status of firmware destination.

**Parameters**

|              |                                  |
|--------------|----------------------------------|
| <i>index</i> | by index of firmware destination |
|--------------|----------------------------------|

**11.141.3.14 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 |
|-------------|----------------------------|

**11.141.3.15 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 |
|--------------|----------------------------------|

**11.141.3.16 GetVersionString() [1/2]** `String ^ GetVersionString ( CFirmwareDestinationNet dest )`

Get the version as a string in the format Major.Minor.

Parameters

|             |                            |
|-------------|----------------------------|
| <i>dest</i> | by CFirmwareDestinationNet |
|-------------|----------------------------|

**11.141.3.17 GetVersionString() [2/2]** `String ^ GetVersionString ( unsigned int index )`

Get the version as a string in the format Major.Minor.

Parameters

|              |                      |
|--------------|----------------------|
| <i>index</i> | by index of firmware |
|--------------|----------------------|

## 11.142 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" )

#### 11.142.1 Member Data Documentation

**11.142.1.1 Altera** String ^ Altera = gcnew String( "Altera" ) [static]

**11.142.1.2 Bootstrap** String ^ Bootstrap = gcnew String( "Bootstrap" ) [static]

**11.142.1.3 BUS1\_MCSBUS1** String ^ BUS1\_MCSBUS1 = gcnew String( "Bus1McsBus1" ) [static]

**11.142.1.4 BUS1\_MCSBUS2** String ^ BUS1\_MCSBUS2 = gcnew String( "Bus1McsBus2" ) [static]

**11.142.1.5 DSP** String ^ DSP = gcnew String( "DSP" ) [static]

**11.142.1.6 FPGA2** String ^ FPGA2 = gcnew String( "FPGA2" ) [static]

**11.142.1.7 FPGA3** String ^ FPGA3 = gcnew String( "FPGA3" ) [static]

**11.142.1.8 FPGA4** String ^ FPGA4 = gcnew String( "FPGA4" ) [static]

**11.142.1.9 FPGA5** String ^ FPGA5 = gcnew String( "FPGA5" ) [static]

**11.142.1.10 FPGA6** `String ^ FPGA6 = gcnew String( "FPGA6" ) [static]`

**11.142.1.11 MCSBUS1** `String ^ MCSBUS1 = gcnew String( "McsBus1" ) [static]`

**11.142.1.12 MCSBUS10** `String ^ MCSBUS10 = gcnew String( "McsBus10" ) [static]`

**11.142.1.13 MCSBUS11** `String ^ MCSBUS11 = gcnew String( "McsBus11" ) [static]`

**11.142.1.14 MCSBUS12** `String ^ MCSBUS12 = gcnew String( "McsBus12" ) [static]`

**11.142.1.15 MCSBUS13** `String ^ MCSBUS13 = gcnew String( "McsBus13" ) [static]`

**11.142.1.16 MCSBUS2** `String ^ MCSBUS2 = gcnew String( "McsBus2" ) [static]`

**11.142.1.17 MCSBUS3** `String ^ MCSBUS3 = gcnew String( "McsBus3" ) [static]`

**11.142.1.18 MCSBUS4** `String ^ MCSBUS4 = gcnew String( "McsBus4" ) [static]`

**11.142.1.19 MCSBUS5** `String ^ MCSBUS5 = gcnew String( "McsBus5" ) [static]`

**11.142.1.20 MCSBUS6** `String ^ MCSBUS6 = gcnew String( "McsBus6" ) [static]`

**11.142.1.21 MCSBUS7** `String ^ MCSBUS7 = gcnew String( "McsBus7" ) [static]`

**11.142.1.22 MCSBUS8** `String ^ MCSBUS8 = gcnew String( "McsBus8" ) [static]`

**11.142.1.23 MCSBUS9** `String ^ MCSBUS9 = gcnew String( "McsBus9" ) [static]`

**11.142.1.24 MCU1** `String ^ MCU1 = gcnew String( "MCU1" ) [static]`

**11.142.1.25 PIC** `String ^ PIC = gcnew String( "PIC" ) [static]`

**11.142.1.26 PIC2** `String ^ PIC2 = gcnew String( "PIC2" ) [static]`

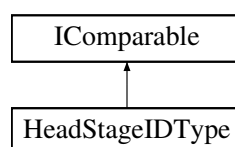
**11.142.1.27 PIC3** `String ^ PIC3 = gcnew String( "PIC3" ) [static]`

**11.142.1.28 PIC4** `String ^ PIC4 = gcnew String( "PIC4" ) [static]`

**11.142.1.29 USB** `String ^ USB = gcnew String( "USB" ) [static]`

## 11.143 HeadStageIDType Class Reference

Inheritance diagram for HeadStageIDType:





## Public Types

- enum class [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]

## 11.143.1 Member Enumeration Documentation

### 11.143.1.1 HeadstageTypeEnum enum [HeadstageTypeEnum](#) [strong]

#### Enumerator

|                                       |  |
|---------------------------------------|--|
| <a href="#">Unknown</a>               |  |
| <a href="#">MeasuringOnly</a>         |  |
| <a href="#">OpticalStimulation</a>    |  |
| <a href="#">ElectricalStimulation</a> |  |

## 11.143.2 Constructor & Destructor Documentation

**11.143.2.1 HeadStageIDType()** `HeadStageIDType (`  
    `unsigned int entry,`  
    `CW2100_FunctionNet^ device )`

### 11.143.3 Member Function Documentation

**11.143.3.1 CompareTo()** `virtual Int32 CompareTo (`  
    `Object^ obj ) [virtual]`

**11.143.3.2 Equals()** `virtual bool Equals (`  
    `Object^ obj ) [override], [virtual]`

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

### 11.143.4 Property Documentation

**11.143.4.1 Entry** `unsigned int Entry [get]`

**11.143.4.2 HasIMU** `bool HasIMU [get]`

**11.143.4.3 HasOptoCurrentMeasurement** `bool HasOptoCurrentMeasurement [get]`

**11.143.4.4 HeadstageType** `HeadstageTypeEnum HeadstageType [get]`

**11.143.4.5 ID** `unsigned short ID [get]`

**11.143.4.6 NumberOfAnalogChannels** `int NumberOfAnalogChannels [get]`

**11.143.4.7 NumberOfStimulationChannels** `int NumberOfStimulationChannels [get]`

**11.143.4.8 SN** `System:: String^ SN [get]`

**11.143.4.9 StimulusParameters** `W2100_StimulusParametersNet^ StimulusParameters [get]`

**11.143.4.10 Type** `System:: String^ Type [get]`

**11.143.4.11 TypeValue** `unsigned int TypeValue [get]`

**11.143.4.12 UserDefinedName** `System:: String^ UserDefinedName [get]`

**11.143.4.13 Valid** `bool Valid [get]`

**11.143.4.14 W16IsW14** `bool W16IsW14 [get]`

## 11.144 HeadstageIDTypeObject Class Reference

### Public Member Functions

- [HeadstageIDTypeObject](#) ([HeadStageIDType](#)^ idType)
- virtual `String` ^ [ToString](#) () override
- virtual `bool` [Equals](#) ([Object](#)^ obj) override
- virtual `int` [GetHashCode](#) () override

## Public Attributes

- [HeadStageIDType](#) ^ [\\_IdType](#)
- String ^ [\\_AdditionalText](#)

## Properties

- [HeadStageIDType](#) ^ [IdType](#) [get]
- String ^ [AdditionalText](#) [get, set]

## 11.144.1 Constructor & Destructor Documentation

**11.144.1.1 HeadstageIDTypeObject()** [HeadstageIDTypeObject](#) (  
    [HeadStageIDType](#) ^ *idType* )

## 11.144.2 Member Function Documentation

**11.144.2.1 Equals()** virtual bool Equals (  
    Object ^ *obj* ) [override], [virtual]

**11.144.2.2 GetHashCode()** virtual int GetHashCode ( ) [override], [virtual]

**11.144.2.3 ToString()** virtual String ^ ToString ( ) [override], [virtual]

## 11.144.3 Member Data Documentation

**11.144.3.1 \_AdditionalText** String ^ [\\_AdditionalText](#)

**11.144.3.2 \_IdType** [HeadStageIDType](#) ^ [\\_IdType](#)

#### 11.144.4 Property Documentation

**11.144.4.1 AdditionalText** `String^ AdditionalText [get], [set]`

**11.144.4.2 IdType** `HeadStageIDType^ IdType [get]`

### 11.145 HeadStageIDTypeState Class Reference

#### Properties

- unsigned int [State](#) [get]
- [HeadStageIDType^](#) [IdType](#) [get]
- bool [ControlState](#) [get]
- bool [DataState](#) [get]

#### 11.145.1 Property Documentation

**11.145.1.1 ControlState** `bool ControlState [get]`

**11.145.1.2 DataState** `bool DataState [get]`

**11.145.1.3 IdType** `HeadStageIDType^ IdType [get]`

**11.145.1.4 State** `unsigned int State [get]`

## 11.146 mkfilterNet Class Reference

### Static Public Member Functions

- static int [mkfilter](#) (String<sup>^</sup> *filtertype*, double *value*, String<sup>^</sup> *passtype*, int *order*, double *alpha1*, double *alpha2*, [System::Runtime::InteropServices::Out] array< double ><sup>^</sup>% *xcoeffs*, [System::Runtime::InteropServices::Out] array< double ><sup>^</sup>% *ycoeffs*)
- static int [mkfilter\\_MCS](#) (int *SamplesPerSecond*, double *R1*, double *R2*, double *C*, double *Amplification*, double *Correction*, [System::Runtime::InteropServices::Out] array< double ><sup>^</sup>% *xcoeffs*, [System::Runtime::InteropServices::Out] array< double ><sup>^</sup>% *ycoeffs*)
- static int [mkfilter\\_MCS](#) (int *SamplesPerSecond*, double *R1*, double *R2*, double *C*, double *Correction*, [System::Runtime::InteropServices::Out] array< double ><sup>^</sup>% *xcoeffs*, [System::Runtime::InteropServices::Out] array< double ><sup>^</sup>% *ycoeffs*)
- static int [mkfilter\\_MCS\\_k](#) (int *SamplesPerSecond*, double *R1*, double *R2*, double *C*, double *Amplification*, double *Correction*, [System::Runtime::InteropServices::Out] array< double ><sup>^</sup>% *coeffs*)
- static int [mkfilter\\_MCS\\_k](#) (int *SamplesPerSecond*, double *R1*, double *R2*, double *C*, double *Correction*, [System::Runtime::InteropServices::Out] array< double ><sup>^</sup>% *coeffs*)
- static void [mkfilter\\_coef\\_in\\_one\\_set](#) (int *n*, [System::Runtime::InteropServices::In] array< double ><sup>^</sup> *xcoeffs*, [System::Runtime::InteropServices::In] array< double ><sup>^</sup> *ycoeffs*, [System::Runtime::InteropServices::Out] array< double ><sup>^</sup>% *out\_coeffs*)
- static void [mkfilter\\_scale\\_coef\\_in\\_one\\_set](#) (int *n*, double *scale*, [System::Runtime::InteropServices::In] array< double ><sup>^</sup> *xcoeffs*, [System::Runtime::InteropServices::In] array< double ><sup>^</sup> *ycoeffs*, [System::Runtime::InteropServices::Out] array< double ><sup>^</sup>% *out\_coeffs*)
- static void [mkfilter\\_normalize\\_coeffs\\_short](#) (short *maxvalue*, [System::Runtime::InteropServices::In] array< double ><sup>^</sup> *coeffs*, [System::Runtime::InteropServices::Out] array< short ><sup>^</sup>% *out\_coeffs*)
- static void [mkfilter\\_normalize\\_coeffs\\_int](#) (int *maxvalue*, [System::Runtime::InteropServices::In] array< double ><sup>^</sup> *coeffs*, [System::Runtime::InteropServices::Out] array< int ><sup>^</sup>% *out\_coeffs*)
- static void [mkfilter\\_normalize\\_scale\\_coeffs\\_int](#) (int *maxvalue*, [System::Runtime::InteropServices::In] array< double ><sup>^</sup> *coeffs*, [System::Runtime::InteropServices::Out] array< int ><sup>^</sup>% *out\_coeffs*)
- static double [mkfilter\\_highpass\\_coeff](#) (int *SamplesPerSecond*, double *Frequency*)
- static double [mkfilter\\_highpass\\_k](#) (int *SamplesPerSecond*, double *Frequency*)
- static double [mkfilter\\_highpass\\_frequency\\_from\\_coeff](#) (int *SamplesPerSecond*, double *coeff*)
- static double [mkfilter\\_highpass\\_frequency\\_from\\_k](#) (int *SamplesPerSecond*, double *k*)

### 11.146.1 Member Function Documentation

**11.146.1.1 mkfilter()** static int `mkfilter` (  
     String<sup>^</sup> *filtertype*,  
     double *value*,  
     String<sup>^</sup> *passtype*,  
     int *order*,  
     double *alpha1*,  
     double *alpha2*,  
     [System::Runtime::InteropServices::Out] array< double ><sup>^</sup>% *xcoeffs*,  
     [System::Runtime::InteropServices::Out] array< double ><sup>^</sup>% *ycoeffs* ) [static]

**11.146.1.2 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]

**11.146.1.3 mkfilter\_highpass\_coeff()** static double mkfilter\_highpass\_coeff (   
     int *SamplesPerSecond*,  
     double *Frequency* ) [static]

**11.146.1.4 mkfilter\_highpass\_frequency\_from\_coeff()** static double mkfilter\_highpass\_frequency\_from↵  
 from\_coeff (   
     int *SamplesPerSecond*,  
     double *coeff* ) [static]

**11.146.1.5 mkfilter\_highpass\_frequency\_from\_k()** static double mkfilter\_highpass\_frequency\_from↵  
 \_k (   
     int *SamplesPerSecond*,  
     double *k* ) [static]

**11.146.1.6 mkfilter\_highpass\_k()** static double mkfilter\_highpass\_k (   
     int *SamplesPerSecond*,  
     double *Frequency* ) [static]

**11.146.1.7 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]

**11.146.1.8 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]

- 11.146.1.9 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]
- 11.146.1.10 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]
- 11.146.1.11 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]
- 11.146.1.12 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]
- 11.146.1.13 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]
- 11.146.1.14 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]



## 11.147 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)

### 11.147.1 Member Function Documentation

**11.147.1.1 FindReferencePhase0()** [1/2] `void FindReferencePhase0 (`  
    `unsigned char busaddress,`  
    `char axes )`

**11.147.1.2 FindReferencePhase0()** [2/2] `void FindReferencePhase0 (`  
    `unsigned char busaddress,`  
    `char axes,`  
    `int timeout )`

**11.147.1.3 GetAxisConfig()** `unsigned int GetAxisConfig (`  
    `unsigned char busaddress,`  
    `char axes )`

**11.147.1.4 GetHWConfig()** `unsigned int GetHWConfig ( )`

**11.147.1.5 GetHWRevision()** `unsigned int GetHWRevision ( )`

**11.147.1.6 GetMaxNoPressure()** `unsigned int GetMaxNoPressure ( )`

**11.147.1.7 GetMaxNoPressureWaitTime()** `unsigned int GetMaxNoPressureWaitTime ( )`

**11.147.1.8 GetMaxPressureWaitTime()** `unsigned int GetMaxPressureWaitTime ( )`

**11.147.1.9 GetMinNoPressureWaitTime()** `unsigned int GetMinNoPressureWaitTime ( )`

**11.147.1.10 GetMinPressure()** unsigned int GetMinPressure ( )

**11.147.1.11 GetMinPressureWaitTime()** unsigned int GetMinPressureWaitTime ( )

**11.147.1.12 GetParameter() [1/2]** void GetParameter (   
 unsigned short *command*,   
 unsigned short *index*,   
 [System::Runtime::InteropServices::Out] unsigned int% *value* )

**11.147.1.13 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* )

**11.147.1.14 GetPhases()** void GetPhases (   
 unsigned char *busaddress*,   
 char *axes*,   
 [System::Runtime::InteropServices::Out] unsigned short% *phase0*,   
 [System::Runtime::InteropServices::Out] unsigned short% *lastphase* )

**11.147.1.15 GetSearchReferenceFastAccel()** unsigned short GetSearchReferenceFastAccel (   
 unsigned char *busaddress*,   
 char *axes* )

**11.147.1.16 GetSearchReferenceFastSpeed()** unsigned short GetSearchReferenceFastSpeed (   
 unsigned char *busaddress*,   
 char *axes* )

**11.147.1.17 GetSearchReferenceFineAccel()** unsigned short GetSearchReferenceFineAccel (   
 unsigned char *busaddress*,   
 char *axes* )

**11.147.1.18 GetSearchReferenceFineSpeed()** unsigned short GetSearchReferenceFineSpeed (   
 unsigned char *busaddress*,  
 char *axes* )

**11.147.1.19 GetSearchReferenceMethod()** unsigned int GetSearchReferenceMethod (   
 unsigned char *busaddress*,  
 char *axes* )

**11.147.1.20 GetSearchReferenceMoveOut()** int GetSearchReferenceMoveOut (   
 unsigned char *busaddress*,  
 char *axes* )

**11.147.1.21 GetSearchReferenceOffsetPos()** int GetSearchReferenceOffsetPos (   
 unsigned char *busaddress*,  
 char *axes* )

**11.147.1.22 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                               |

**11.147.1.23 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                               |

**11.147.1.24 HasRef()** unsigned char HasRef (   
 unsigned char *busaddress*,   
 char *axes* )

**11.147.1.25 SetAxisConfig()** void SetAxisConfig (   
 unsigned char *busaddress*,   
 char *axes*,   
 unsigned int *config* )

**11.147.1.26 SetHWConfig()** void SetHWConfig (   
 unsigned int *config* )

**11.147.1.27 SetHWRevision()** void SetHWRevision (   
 unsigned int *revision* )

**11.147.1.28 SetMaxNoPressure()** void SetMaxNoPressure (   
 unsigned int *pressure* )

**11.147.1.29 SetMaxNoPressureWaitTime()** void SetMaxNoPressureWaitTime (   
 unsigned int *t* )

**11.147.1.30 SetMaxPressureWaitTime()** void SetMaxPressureWaitTime (   
 unsigned int *t* )

**11.147.1.31 SetMinNoPressureWaitTime()** void SetMinNoPressureWaitTime (   
 unsigned int *t* )

**11.147.1.32 SetMinPressure()** void SetMinPressure (   
 unsigned int *pressure* )

**11.147.1.33 SetMinPressureWaitTime()** void SetMinPressureWaitTime (   
 unsigned int *t* )

**11.147.1.34 SetParameter()** [1/2] void SetParameter (   
 unsigned short *command*,   
 unsigned short *index*,   
 unsigned int *value* )

**11.147.1.35 SetParameter()** [2/2] void SetParameter (   
 unsigned short *command*,   
 unsigned short *index*,   
 unsigned int *value1*,   
 unsigned int *value2* )

**11.147.1.36 SetSearchReferenceFastAccel()** void SetSearchReferenceFastAccel (   
 unsigned char *busaddress*,   
 char *axes*,   
 unsigned short *accel* )

**11.147.1.37 SetSearchReferenceFastSpeed()** void SetSearchReferenceFastSpeed (   
 unsigned char *busaddress*,   
 char *axes*,   
 unsigned short *speed* )

**11.147.1.38 SetSearchReferenceFineAccel()** void SetSearchReferenceFineAccel (   
 unsigned char *busaddress*,   
 char *axes*,   
 unsigned short *accel* )

**11.147.1.39 SetSearchReferenceFineSpeed()** void SetSearchReferenceFineSpeed (   
 unsigned char *busaddress*,   
 char *axes*,   
 unsigned short *speed* )

**11.147.1.40 SetSearchReferenceMethod()** `void SetSearchReferenceMethod (`  
     `unsigned char busaddress,`  
     `char axes,`  
     `unsigned int method )`

**11.147.1.41 SetSearchReferenceMoveOut()** `void SetSearchReferenceMoveOut (`  
     `unsigned char busaddress,`  
     `char axes,`  
     `int move )`

**11.147.1.42 SetSearchReferenceOffsetPos()** `void SetSearchReferenceOffsetPos (`  
     `unsigned char busaddress,`  
     `char axes,`  
     `int offsetpos )`

**11.147.1.43 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                         |

**11.147.1.44 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                         |

## 11.148 CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands Class Reference

### Public Member Functions

- void [FindReferencePhase0XY](#) ()
- void [FindReferencePhase0XY](#) (int timeout)

#### 11.148.1 Member Function Documentation

**11.148.1.1 [FindReferencePhase0XY\(\)](#) [1/2]** `void FindReferencePhase0XY ( )`

**11.148.1.2 [FindReferencePhase0XY\(\)](#) [2/2]** `void FindReferencePhase0XY (   
int timeout )`

## 11.149 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](#)

#### 11.149.1 Constructor & Destructor Documentation

**11.149.1.1 [s\\_FilterAttributesNet\(\)](#)** `s_FilterAttributesNet (   
s_FilterAttributes attrib )`

#### 11.149.2 Member Function Documentation



**11.149.2.1 ToCpp()** `s_FilterAttributes ToCpp ( )`

### 11.149.3 Member Data Documentation

**11.149.3.1 CommaPositionA** `uint32_t CommaPositionA`

**11.149.3.2 CommaPositionB** `uint32_t CommaPositionB`

**11.149.3.3 PostCommaA** `uint32_t PostCommaA`

**11.149.3.4 PostCommaB** `uint32_t PostCommaB`

**11.149.3.5 PreCommaA** `uint32_t PreCommaA`

**11.149.3.6 PreCommaB** `uint32_t PreCommaB`

## 11.150 CMeaAudioFunctionNet::s\_setaudionet Struct Reference

### Public Attributes

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

### 11.150.1 Member Data Documentation

**11.150.1.1 amplification** `int amplification`

**11.150.1.2 channel** `int channel`

## 11.151 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]

### 11.151.1 Constructor & Destructor Documentation

**11.151.1.1 SidebandData()** [SidebandData](#) ( )

**11.151.1.2 ~SidebandData()** [~SidebandData](#) ( )

Destructor: called by Dispose()

**11.151.1.3 "!SidebandData()** [!SidebandData](#) ( )

Finalizer: called by GC before collecting

### 11.151.2 Property Documentation

**11.151.2.1 Duration** `array< uint64_t >^` [Duration](#) [get]

**11.151.2.2 Sideband** `array< int32_t >^` [Sideband](#) [get]

## 11.152 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](#)

### 11.152.1 Member Function Documentation

**11.152.1.1 FromIntPtr()** static [StgStatusNet](#) ^ FromIntPtr ( IntPtr stgstatus ) [static]

**11.152.1.2 FromPtr()** static [StgStatusNet](#) ^ FromPtr ( stgstatus\_t \* stgstatus ) [static]

### 11.152.2 Member Data Documentation

**11.152.2.1 ListOfChangedTriggers** array<uint32\_t> ^ ListOfChangedTriggers

**11.152.2.2 TiggerStatus** array<[Stg200xTriggerStatusEnumNet](#)> ^ TiggerStatus

## 11.153 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]

### 11.153.1 Constructor & Destructor Documentation

**11.153.1.1 StimulusDeviceDataAndUnrolledData()** [StimulusDeviceDataAndUnrolledData](#) ( )

**11.153.1.2 ~StimulusDeviceDataAndUnrolledData()** [~StimulusDeviceDataAndUnrolledData](#) ( )

Destructor: called by Dispose()

**11.153.1.3 ~!StimulusDeviceDataAndUnrolledData()** [!StimulusDeviceDataAndUnrolledData](#) ( )

Finalizer: called by GC before collecting

### 11.153.2 Property Documentation

**11.153.2.1 DeviceData** array< uint8\_t >^ DeviceData [get]

**11.153.2.2 DeviceDataLength** int DeviceDataLength [get]

**11.153.2.3 UnrolledAmplitude** array< int32\_t >^ UnrolledAmplitude [get]

**11.153.2.4 UnrolledDuration** array< uint64\_t >^ UnrolledDuration [get]

**11.153.2.5 UnrolledSync** `array< uint32_t>^ UnrolledSync [get]`

## 11.154 usbSetupPacket\_t Class Reference

### Public Attributes

- `uint8_t` [bmRequestType](#)
- `uint8_t` [bRequest](#)
- `uint16_t` [wValue](#)
- `uint16_t` [wIndex](#)
- `uint16_t` [wLength](#)

### 11.154.1 Member Data Documentation

**11.154.1.1 bmRequestType** `uint8_t bmRequestType`

**11.154.1.2 bRequest** `uint8_t bRequest`

**11.154.1.3 wIndex** `uint16_t wIndex`

**11.154.1.4 wLength** `uint16_t wLength`

**11.154.1.5 wValue** `uint16_t wValue`

## 11.155 W2100\_StimulusParametersNet Struct Reference

### Public Attributes

- `int` [DACResolution](#)
- `int` [TimeResolutionInNanoSeconds](#)
- `int` [VoltageRangeInMicroVolt](#)
- `int` [VoltageResolutionInMicroVolt](#)
- `int` [CurrentRangeInNanoAmp](#)
- `int` [CurrentResolutionInNanoAmp](#)

### 11.155.1 Member Data Documentation

**11.155.1.1 CurrentRangeInNanoAmp** `int CurrentRangeInNanoAmp`

**11.155.1.2 CurrentResolutionInNanoAmp** `int CurrentResolutionInNanoAmp`

**11.155.1.3 DACResolution** `int DACResolution`

**11.155.1.4 TimeResolutionInNanoSeconds** `int TimeResolutionInNanoSeconds`

**11.155.1.5 VoltageRangeInMicroVolt** `int VoltageRangeInMicroVolt`

**11.155.1.6 VoltageResolutionInMicroVolt** `int VoltageResolutionInMicroVolt`

## Index

- !CDacCalibrationFunctionNet
  - CDacCalibrationFunctionNet, [121](#)
- !CDigOutStimulatorFunctionNet
  - CDigOutStimulatorFunctionNet, [131](#)
- !CEXternDTesterDeviceNet
  - CEXternDTesterDeviceNet, [136](#)
- !CGrapheneFunctionNet
  - CGrapheneFunctionNet, [177](#)
- !CInterfaceboard2FunctionNet
  - CInterfaceboard2FunctionNet, [220](#)
- !CInterfaceboardFunctionNet
  - CInterfaceboardFunctionNet, [222](#)
- !CLIH3DeviceNet
  - CLIH3DeviceNet, [225](#)
- !CMEA2100x256FunctionNet
  - CMEA2100x256FunctionNet, [340](#)
- !CMcsUsbFunctionNet
  - CMcsUsbFunctionNet, [309](#)
- !CMcsUsbListNet
  - CMcsUsbListNet, [315](#)
- !CMcsUsbNet
  - CMcsUsbNet, [321](#)
- !CMeFunctionNet
  - CMeFunctionNet, [374](#)
- !CMeaCleanDeviceNet
  - CMeaCleanDeviceNet, [345](#)
- !CMeaCoatDeviceNet
  - CMeaCoatDeviceNet, [349](#)
- !CMultiBatteryChargerDeviceNet
  - CMultiBatteryChargerDeviceNet, [376](#)
- !CMultiwellCallbackFunctionNet
  - CMultiwellCallbackFunctionNet, [383](#)
- !CMultiwellDeviceNet
  - CMultiwellDeviceNet, [385](#)
- !CMultiwellOptoStimFunctionNet
  - CMultiwellOptoStimFunctionNet, [392](#)
- !CPPCFunctionNet
  - CPPCFunctionNet, [426](#)
- !CPedoterDeviceNet
  - CPedoterDeviceNet, [407](#)
- !CPositionIIDeviceNet
  - CPositionIIDeviceNet, [413](#)
- !CPositionImpDeviceNet
  - CPositionImpDeviceNet, [421](#)
- !CProgramPressureCurveNet
  - CProgramPressureCurveNet, [440](#)
- !CPulseGeneratorFunctionNet
  - CPulseGeneratorFunctionNet, [441](#)
- !CRFFFunctionNet
  - CRFFFunctionNet, [449](#)
- !CSCUFunctionNet
  - CSCUFunctionNet, [505](#)
- !CTEERFunctionNet
  - CTEERFunctionNet, [607](#)
- !CUsbDeviceConfigurationFunctionNet, [617](#)
- !CWarnerUssingDeviceNet
  - CWarnerUssingDeviceNet, [635](#)
- !CWarnerUssingFunctionNet
  - CWarnerUssingFunctionNet, [638](#)
- !CWarnerValveControllerDeviceNet
  - CWarnerValveControllerDeviceNet, [656](#)
- !CWarnerValveControllerDeviceTesterFunctionNet
  - CWarnerValveControllerDeviceTesterFunctionNet, [678](#)
- !SidebandData
  - CStimulusFunctionNet::SidebandData, [716](#)
- !StimulusDeviceDataAndUnrolledData
  - CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, [718](#)
- \_AdditionalText
  - HeadstageIDTypeObject, [702](#)
- \_IdType
  - HeadstageIDTypeObject, [702](#)
- ~CCMOSMeaDeviceNet
  - CCMOSMeaDeviceNet, [115](#)
- ~CChannelTestDeviceNet
  - CChannelTestDeviceNet, [103](#)
- ~CCreateFilterNet
  - CCreateFilterNet, [119](#)
- ~CDacCalibrationFunctionNet
  - CDacCalibrationFunctionNet, [121](#)
- ~CDigOutStimulatorFunctionNet
  - CDigOutStimulatorFunctionNet, [131](#)
- ~CEXternDTesterDeviceNet
  - CEXternDTesterDeviceNet, [136](#)
- ~CFilterCoefficientsNet
  - CFilterCoefficientsNet, [138](#)
- ~CFilterPropertyNet
  - CFilterPropertyNet, [143](#)
- ~CFluidControlDeviceNet
  - CFluidControlDeviceNet, [146](#)
- ~CGenericDevelopDeviceNet
  - CGenericDevelopDeviceNet, [160](#)
- ~CGilsonDeviceNet
  - CGilsonDeviceNet, [172](#)
- ~CGrapheneASICDeviceNet
  - CGrapheneASICDeviceNet, [173](#)
- ~CGrapheneFunctionNet
  - CGrapheneFunctionNet, [176](#)
- ~CInterfaceboard2FunctionNet
  - CInterfaceboard2FunctionNet, [220](#)
- ~CInterfaceboardFunctionNet
  - CInterfaceboardFunctionNet, [222](#)
- ~CLIH3DeviceNet
  - CLIH3DeviceNet, [225](#)
- ~CMEA2100x256FunctionNet
  - CMEA2100x256FunctionNet, [340](#)
- ~CMcsBusNet
  - CMcsBusNet, [272](#)

- ~CMcsBus\_AxisParametersNet
    - CMcsBus\_AxisParametersNet, [233](#)
  - ~CMcsBus\_ExtensionNet
    - CMcsBus\_ExtensionNet, [235](#)
  - ~CMcsBus\_FYIExtensionNet
    - CMcsBus\_FYIExtensionNet, [236](#)
  - ~CMcsBus\_MotorControlNet
    - CMcsBus\_MotorControlNet, [240](#)
  - ~CMcsBus\_SensorNet
    - CMcsBus\_SensorNet, [256](#)
  - ~CMcsBus\_TempSensorNet
    - CMcsBus\_TempSensorNet, [265](#)
  - ~CMcsBus\_VoltageModeNet
    - CMcsBus\_VoltageModeNet, [268](#)
  - ~CMcsUsbDacqNet
    - CMcsUsbDacqNet, [279](#)
  - ~CMcsUsbFactoryNet
    - CMcsUsbFactoryNet, [302](#)
  - ~CMcsUsbFunctionNet
    - CMcsUsbFunctionNet, [309](#)
  - ~CMcsUsbListEntryNet
    - CMcsUsbListEntryNet, [311](#)
  - ~CMcsUsbListNet
    - CMcsUsbListNet, [315](#)
  - ~CMcsUsbNet
    - CMcsUsbNet, [321](#)
  - ~CMeFunctionNet
    - CMeFunctionNet, [374](#)
  - ~CMeaCleanDeviceNet
    - CMeaCleanDeviceNet, [344](#)
  - ~CMeaCoatDeviceNet
    - CMeaCoatDeviceNet, [349](#)
  - ~CMeaDeviceNet
    - CMeaDeviceNet, [355](#)
  - ~CMealImpedanceDeviceNet
    - CMealImpedanceDeviceNet, [368](#)
  - ~CMeaSwitchDeviceNet
    - CMeaSwitchDeviceNet, [371](#)
  - ~CMeaUSBDeviceNet
    - CMeaUSBDeviceNet, [373](#)
  - ~CMultiBatteryChargerDeviceNet
    - CMultiBatteryChargerDeviceNet, [376](#)
  - ~CMultiwellCallbackFunctionNet
    - CMultiwellCallbackFunctionNet, [383](#)
  - ~CMultiwellDeviceNet
    - CMultiwellDeviceNet, [385](#)
  - ~CMultiwellOptoStimFunctionNet
    - CMultiwellOptoStimFunctionNet, [392](#)
  - ~CNF\_GenDeviceNet
    - CNF\_GenDeviceNet, [396](#)
  - ~CokuvisionStimulatorDeviceNet
    - CokuvisionStimulatorDeviceNet, [402](#)
  - ~CPPCFunctionNet
    - CPPCFunctionNet, [426](#)
  - ~CPathIdentDeviceNet
    - CPathIdentDeviceNet, [406](#)
  - ~CPedoterDeviceNet
    - CPedoterDeviceNet, [407](#)
  - ~CPeristalticPumpDeviceNet
    - CPeristalticPumpDeviceNet, [408](#)
  - ~CPgaDeviceNet
    - CPgaDeviceNet, [409](#)
  - ~CPositionIIDeviceNet
    - CPositionIIDeviceNet, [413](#)
  - ~CPositionImpDeviceNet
    - CPositionImpDeviceNet, [421](#)
  - ~CProgramPressureCurveNet
    - CProgramPressureCurveNet, [439](#)
  - ~CPulseGeneratorFunctionNet
    - CPulseGeneratorFunctionNet, [441](#)
  - ~CRFFFunctionNet
    - CRFFFunctionNet, [449](#)
  - ~CRetinaLedDeviceNet
    - CRetinaLedDeviceNet, [447](#)
  - ~CRoboDeviceNet
    - CRoboDeviceNet, [474](#)
  - ~CRoboFluidDeviceNet
    - CRoboFluidDeviceNet, [488](#)
  - ~CSCUFunctionNet
    - CSCUFunctionNet, [505](#)
  - ~CSafeISDeviceNet
    - CSafeISDeviceNet, [499](#)
  - ~CStg200xBasicNet
    - CStg200xBasicNet, [527](#)
  - ~CStg200xDownloadNet
    - CStg200xDownloadNet, [571](#)
  - ~CSw2to64DeviceNet
    - CSw2to64DeviceNet, [590](#)
  - ~CTEERFunctionNet
    - CTEERFunctionNet, [607](#)
  - ~CTEERMachineDeviceNet
    - CTEERMachineDeviceNet, [615](#)
  - ~CTcxDeviceNet
    - CTcxDeviceNet, [594](#)
  - ~CUsbDeviceConfigurationFunctionNet
    - CUsbDeviceConfigurationFunctionNet, [617](#)
  - ~CWarnerUssingDeviceNet
    - CWarnerUssingDeviceNet, [635](#)
  - ~CWarnerUssingFunctionNet
    - CWarnerUssingFunctionNet, [638](#)
  - ~CWarnerValveControllerDeviceNet
    - CWarnerValveControllerDeviceNet, [656](#)
  - ~CWarnerValveControllerDeviceTesterFunctionNet
    - CWarnerValveControllerDeviceTesterFunctionNet, [677](#)
  - ~DriverVersionNet
    - DriverVersionNet, [690](#)
  - ~SidebandData
    - CStimulusFunctionNet::SidebandData, [716](#)
  - ~StimulusDeviceDataAndUnrolledData
    - CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, [718](#)
- A
- CFilterCoefficientsNet, [139](#)
  - Mcs::Usb, [68](#)
- AccelOnly



- Mcs::Usb, [90](#)
- AdapterTypeEnumNet
  - Mcs::Usb, [52](#)
- AddBlock
  - CMcsUsbDacqNet::ChannelBlockNet, [189](#)
- AddBlocksAndChannels
  - CMcsUsbDacqNet::ChannelBlockNet, [189](#), [190](#)
- AddChannel
  - CMcsUsbDacqNet::ChannelBlockNet, [192](#)
- AdditionalText
  - HeadstageIDTypeObject, [703](#)
- AddLoopEntry
  - CRetinaLedDeviceNet, [447](#)
- AddSelectedChannelsQueue
  - CMcsUsbDacqNet::ChannelBlockNet, [192](#), [193](#)
- AddSoftwareKey
  - CMcsUsbNet, [321](#)
- AddTableEntry
  - CRetinaLedDeviceNet, [447](#)
- ALA\_VC3
  - Mcs::Usb, [76](#), [90](#)
- ALA\_VC3\_DEVICE
  - Mcs::Usb, [63](#)
- ALTERA
  - Mcs::Usb, [55](#)
- Altera
  - FirmwareDestinationNames, [696](#)
- ALTERA\_BASE
  - Mcs::Usb, [56](#)
- ALTERA\_BOOTSTRAP
  - Mcs::Usb, [56](#)
- ALTERA\_GOLD
  - Mcs::Usb, [55](#)
- ALTERA\_TARGET1
  - Mcs::Usb, [57](#)
- ALTERA\_TARGET2
  - Mcs::Usb, [57](#)
- ALTERA\_TARGET3
  - Mcs::Usb, [57](#)
- AlwaysOn
  - Mcs::Usb, [79](#)
- Ampere
  - Mcs::Usb, [53](#), [90](#)
- amplification
  - CMeaAudioFunctionNet::s\_setaudionet, [715](#)
  - CW2100\_FunctionNet::AudioChannelsNet, [95](#)
- AmplifierSettle
  - CIntanMea\_FunctionNet, [218](#)
- AMS\_Dongle
  - Mcs::Usb, [77](#)
- Analog
  - Mcs::Usb, [76](#), [93](#)
- AnalogGain
  - CMeaDeviceNet, [360](#)
- AnalogGroup
  - Mcs::Usb, [61](#)
- AnalogOut\_DAC\_Range\_EnumNet
  - Mcs::Usb, [53](#)
- AnalogSource\_HS1
  - Mcs::Usb, [53](#)
- AnalogSource\_HS2
  - Mcs::Usb, [53](#)
- AnalogSource\_IF
  - Mcs::Usb, [53](#)
- AnalogSourceEnumNet
  - Mcs::Usb, [53](#)
- AnalogUnitEnumNet
  - Mcs::Usb, [53](#)
- Any
  - Mcs::Usb, [76](#), [90](#)
- AppendPreparedData
  - CStg200xDownloadNet, [571](#)
- ApplyGains
  - CPgaDeviceNet, [410](#)
- AreTransistorVoltagesSet
  - CCMOSMea\_FunctionNet, [106](#)
- Armed
  - Mcs::Usb, [84](#)
- ASMedia
  - Mcs::Usb, [89](#)
- AssociateToThis
  - CMcsUsbNet, [321](#)
- AudioTestChannelGroup
  - Mcs::Usb, [61](#), [70](#), [81](#), [92](#)
- AutomaticAnalogOut
  - CSCUFunctionNet, [505](#)
- Aux
  - Mcs::Usb, [63](#)
- AuxIn
  - Mcs::Usb, [57](#), [70](#), [72](#), [74](#), [82](#), [85](#), [92](#)
- AuxPort
  - Mcs::Usb, [59](#)
- AvailFrames
  - CMcsUsbDacqNet::ChannelBlockNet, [194](#)
- Axes\_I
  - CRoboDeviceNet, [482](#)
- Axes\_X
  - CRoboDeviceNet, [482](#)
- Axes\_Y
  - CRoboDeviceNet, [482](#)
- Axes\_Z
  - CRoboDeviceNet, [483](#)
- Axis\_I
  - CRoboDeviceNet, [483](#)
- Axis\_X
  - CRoboDeviceNet, [483](#)
- Axis\_Y
  - CRoboDeviceNet, [483](#)
- Axis\_Z
  - CRoboDeviceNet, [483](#)
- B
  - CFilterCoefficientsNet, [139](#)
  - Mcs::Usb, [68](#)
- BatteryState, [95](#)
  - Charge, [95](#)
  - ChargeRegionString, [95](#)

- ChargeString, [95](#)
- Voltage, [96](#)
- VoltageString, [96](#)
- BcdDevice
  - DeviceIdNet, [686](#)
- BeginImpedanceCheck
  - CIntanMea\_FunctionNet, [218](#)
- Bessel
  - Mcs::Usb, [66](#)
- Bessel3dbFilterBandPassNet, [96](#)
  - Bessel3dbFilterBandPassNet, [96](#)
- Bessel3dbFilterHighPassNet, [96](#)
  - Bessel3dbFilterHighPassNet, [97](#)
- Bessel3dbFilterLowPassNet, [97](#)
  - Bessel3dbFilterLowPassNet, [97](#)
- BesselConstDelayFilterBandPassNet, [98](#)
  - BesselConstDelayFilterBandPassNet, [98](#)
- BesselConstDelayFilterHighPassNet, [98](#)
  - BesselConstDelayFilterHighPassNet, [98](#)
- BesselConstDelayFilterLowPassNet, [99](#)
  - BesselConstDelayFilterLowPassNet, [99](#)
- BesselLegacyFilterHighPassNet, [99](#)
  - BesselLegacyFilterHighPassNet, [100](#)
- BesselLegacyFilterLowPassNet, [100](#)
  - BesselLegacyFilterLowPassNet, [100](#)
- BMI
  - Mcs::Usb, [86](#)
- bmRequestType
  - usbSetupPacket\_t, [719](#)
- BOOST\_BIT
  - CW2100\_StimulatorFunctionNet, [633](#)
- Bootstrap
  - FirmwareDestinationNames, [696](#)
  - Mcs::Usb, [55](#)
- BootstrapOtherCypress
  - Mcs::Usb, [55](#)
- Both
  - Mcs::Usb, [90](#)
- Break
  - Mcs::Usb, [80](#)
- bRequest
  - usbSetupPacket\_t, [719](#)
- BurnAdcOffset
  - COctoPotDeviceNet, [398](#)
- BurnDacOffset
  - CDacCalibrationFunctionNet, [121](#)
  - COctoPotDeviceNet, [398](#)
- BUS0MCSBUS0
  - Mcs::Usb, [54](#)
- BUS0MCSBUS1
  - Mcs::Usb, [54](#)
- BUS0MCSBUS10
  - Mcs::Usb, [54](#)
- BUS0MCSBUS11
  - Mcs::Usb, [54](#)
- BUS0MCSBUS12
  - Mcs::Usb, [54](#)
- BUS0MCSBUS13
  - Mcs::Usb, [54](#)
- BUS0MCSBUS14
  - Mcs::Usb, [54](#)
- BUS0MCSBUS15
  - Mcs::Usb, [54](#)
- BUS0MCSBUS2
  - Mcs::Usb, [54](#)
- BUS0MCSBUS3
  - Mcs::Usb, [54](#)
- BUS0MCSBUS4
  - Mcs::Usb, [54](#)
- BUS0MCSBUS5
  - Mcs::Usb, [54](#)
- BUS0MCSBUS6
  - Mcs::Usb, [54](#)
- BUS0MCSBUS7
  - Mcs::Usb, [54](#)
- BUS0MCSBUS8
  - Mcs::Usb, [54](#)
- BUS0MCSBUS9
  - Mcs::Usb, [54](#)
- BUS1\_MCSBUS1
  - FirmwareDestinationNames, [696](#)
- BUS1\_MCSBUS2
  - FirmwareDestinationNames, [696](#)
- BUS1MCSBUS0
  - Mcs::Usb, [54](#)
- BUS1MCSBUS1
  - Mcs::Usb, [54](#)
- BUS1MCSBUS10
  - Mcs::Usb, [54](#)
- BUS1MCSBUS11
  - Mcs::Usb, [54](#)
- BUS1MCSBUS12
  - Mcs::Usb, [54](#)
- BUS1MCSBUS13
  - Mcs::Usb, [54](#)
- BUS1MCSBUS14
  - Mcs::Usb, [54](#)
- BUS1MCSBUS15
  - Mcs::Usb, [54](#)
- BUS1MCSBUS2
  - Mcs::Usb, [54](#)
- BUS1MCSBUS3
  - Mcs::Usb, [54](#)
- BUS1MCSBUS4
  - Mcs::Usb, [54](#)
- BUS1MCSBUS5
  - Mcs::Usb, [54](#)
- BUS1MCSBUS6
  - Mcs::Usb, [54](#)
- BUS1MCSBUS7
  - Mcs::Usb, [54](#)
- BUS1MCSBUS8
  - Mcs::Usb, [54](#)
- BUS1MCSBUS9
  - Mcs::Usb, [54](#)
- BUS2MCSBUS0
  - Mcs::Usb, [54](#)

- Mcs::Usb, [55](#)
- BUS2MCSBUS1
  - Mcs::Usb, [54](#)
- BUS2MCSBUS10
  - Mcs::Usb, [55](#)
- BUS2MCSBUS11
  - Mcs::Usb, [55](#)
- BUS2MCSBUS12
  - Mcs::Usb, [55](#)
- BUS2MCSBUS13
  - Mcs::Usb, [55](#)
- BUS2MCSBUS14
  - Mcs::Usb, [55](#)
- BUS2MCSBUS15
  - Mcs::Usb, [55](#)
- BUS2MCSBUS2
  - Mcs::Usb, [55](#)
- BUS2MCSBUS3
  - Mcs::Usb, [55](#)
- BUS2MCSBUS4
  - Mcs::Usb, [55](#)
- BUS2MCSBUS5
  - Mcs::Usb, [55](#)
- BUS2MCSBUS6
  - Mcs::Usb, [55](#)
- BUS2MCSBUS7
  - Mcs::Usb, [55](#)
- BUS2MCSBUS8
  - Mcs::Usb, [55](#)
- BUS2MCSBUS9
  - Mcs::Usb, [55](#)
- BUSNUMBER0
  - Mcs::Usb, [54](#)
- BUSNUMBER1
  - Mcs::Usb, [54](#)
- BUSNUMBER2
  - Mcs::Usb, [54](#)
- BusType
  - DeviceIdNet, [686](#)
- Butterworth
  - Mcs::Usb, [66](#)
- ButterworthFilterBandPassNet, [100](#)
  - ButterworthFilterBandPassNet, [101](#)
- ButterworthFilterHighPassNet, [101](#)
  - ButterworthFilterHighPassNet, [101](#)
- ButterworthFilterLowPassNet, [102](#)
  - ButterworthFilterLowPassNet, [102](#)
- CalibrateThermocouple
  - CFluidControlDeviceNet, [146](#)
  - CTcxDeviceNet, [594](#)
- Campden\_Ci4600EphysVideoDataIntegrator
  - Mcs::Usb, [76](#)
- CancelInternalCalibration
  - CTEERFunctionNet, [607](#)
- CancelPoolLoop
  - CRoboDeviceNet, [474](#)
- CancelPoolLoopAndStopMovement
  - CRoboDeviceNet, [474](#)
- CancelTableLoop
  - CRoboDacqNet, [459](#)
- CancelTableLoopAndStopTable
  - CRoboDacqNet, [459](#)
- CapacityTest
  - CMultiBatteryChargerDeviceNet, [376](#)
- CatchAmp
  - Mcs::Usb, [75](#)
- CatchAmpGetAdcMean
  - CMcsBus\_SensorNet, [256](#)
- CatchAmpGetAdcValue
  - CMcsBus\_SensorNet, [257](#)
- CatchAmpGetAdcValueH
  - CMcsBus\_SensorNet, [257](#)
- CatchAmpGetAdcValueL
  - CMcsBus\_SensorNet, [257](#)
- CatchAmpGetDacAmplitude
  - CMcsBus\_SensorNet, [257](#)
- CatchAmpGetDacEnable
  - CMcsBus\_SensorNet, [257](#)
- CatchAmpGetDacOffset
  - CMcsBus\_SensorNet, [257](#)
- CatchAmpGetPwmEnable
  - CMcsBus\_SensorNet, [257](#)
- CatchAmpSetDacAmplitude
  - CMcsBus\_SensorNet, [257](#)
- CatchAmpSetDacEnable
  - CMcsBus\_SensorNet, [258](#)
- CatchAmpSetDacOffset
  - CMcsBus\_SensorNet, [258](#)
- CatchAmpSetPwmEnable
  - CMcsBus\_SensorNet, [258](#)
- CChannelTestDeviceNet, [102](#)
  - ~CChannelTestDeviceNet, [103](#)
  - CChannelTestDeviceNet, [103](#)
  - SetAmplitude, [103](#)
  - SetAttenuation, [103](#)
  - SetFrequency, [103](#)
  - SetWaveform, [103](#)
- CCMOSMea\_FunctionNet, [103](#)
  - AreTransistorVoltagesSet, [106](#)
  - CCMOSMea\_FunctionNet, [105](#), [106](#)
  - ClearSTGOutput, [106](#)
  - DetectChipType, [106](#)
  - EnableChannelsInGroup, [106](#)
  - GetADCInputOffset, [106](#)
  - GetBath, [106](#)
  - GetBathMode, [106](#)
  - GetEnabledChannelsInGroup, [107](#)
  - GetGate, [107](#)
  - GetGNDI, [107](#)
  - GetGroupADCBits, [107](#)
  - GetGroupChannelBitmaskBySelect, [107](#)
  - GetGroupChannelBitmaskHS1NCBathCurrent, [107](#), [108](#)
  - GetGroupChannelBitmaskHS1NCCol2Current, [108](#)
  - GetGroupChannelBitmaskHS1NChipTemp, [108](#)

- GetGroupChannelBitmaskHS1Sidebands, 108
- GetGroupChannelBitmaskHS1TriggerStatus, 108, 109
- GetGroupChannelBitmaskIFDigChannels, 109
- GetGroupChannelBitmaskInterfaceADC, 109
- GetGroupChannelBitmaskPacketFrameContext, 109
- GetGroupChannelBitmaskSTG1DACSignal, 109, 110
- GetGroupDCOffset, 110
- GetGroupID, 110
- GetGroupNumberOfChannels, 110
- GetGroupResolutionPerDigit, 110
- GetGroupSampleSize, 111
- GetGroupType, 111
- GetGroupUnit, 111
- GetMaxNumOfColumns, 111
- GetNeurochipMemoryData, 111
- GetNeurochipMemorySize, 112
- GetNumberOfSupportedGroups, 112
- GetSourceBulk, 112
- GetSourceDrain, 112
- GetSourceGate, 112
- GetStimulusSites, 112
- GetVDD3I, 112
- GetVDDI, 112
- IsChipPowered, 112
- IsGateFloating, 112
- PowerChip, 113
- SetADCInputOffset, 113
- SetBath, 113
- SetBathMode, 113
- SetGate, 113
- SetGateFloating, 113
- SetGateToVOP, 113
- SetNeurochipMemoryData, 113
- SetSourceBulk, 113
- SetSourceDrain, 114
- SetSourceGate, 114
- SetStimulusSites, 114
- UpdateTransistorVoltages, 114
- VOPSTimerSetResetTimes, 114
- CCMOSMeaDeviceNet, 114
  - ~CCMOSMeaDeviceNet, 115
  - CCMOSMeaDeviceNet, 115
  - CMosMea, 117
  - GetAvailableBaseSamplerates, 115
  - GetBaseSamplerate, 116
  - GetChannelDataI16, 116
  - GetChannelDataI32, 116
  - GetChannelDataUI16, 116
  - GetChannelDataUI32, 116
  - GetCMOSDataDictionary, 116
  - GetMaxReadableColumns, 116
  - SetBaseSamplerate, 117
  - SetRegionOfInterests, 117
  - Stimulus, 117
  - UpdateChannelBlock, 117
- CCMOSMeaDeviceNet::CRegionOfInterestRect, 445
  - CRegionOfInterestRect, 445
  - DeepCopy, 445
  - m\_Bottom, 446
  - m\_Left, 446
  - m\_Right, 446
  - m\_Top, 446
- CCreateFilterNet, 117
  - ~CCreateFilterNet, 119
  - CCreateFilterNet, 119
  - CutoffOrCenterFrequency, 119
  - FindFilter, 119
  - GetBiQuad, 119
  - GetBiQuads, 119
  - GetCpp, 119
  - NumCoefSets, 120
  - Order, 120
  - SampleRate, 120
  - Scale, 120
  - WidthFrequency, 120
- CDacCalibrationFunctionNet, 120
  - !CDacCalibrationFunctionNet, 121
  - ~CDacCalibrationFunctionNet, 121
  - BurnDacOffset, 121
  - CDacCalibrationFunctionNet, 121
  - GetDacOffset, 121
  - SetDacOffset, 122
- CDacqGroupChannelGenericSelectionNet, 122
  - CDacqGroupChannelGenericSelectionNet, 122
- CDacqGroupChannelSelectionNet, 123
  - CDacqGroupChannelSelectionNet, 123
- CDacqGroupChannelSelectionTemplateNet
  - CDacqGroupChannelSelectionTemplateNet< Dac-qGroupChannelEnumTemplateNet, Dac-qGroupChannelEnumTemplate, CDevice-GroupChannelInfoTemplateNet >, 124
- CDacqGroupChannelSelectionTemplateNet< Dac-qGroupChannelEnumTemplateNet, Dac-qGroupChannelEnumTemplate, CDevice-GroupChannelInfoTemplateNet >, 123
  - CDacqGroupChannelSelectionTemplateNet, 124
  - EnableChannelsInGroup, 124
  - GetDeviceGroupChannelInfos, 124
  - GetEnabledChannelsInGroup, 124, 125
  - GetGroupID, 125
  - GetGroupNumberOfChannels, 125
  - GetGroupSampleSize, 125
  - GetGroupType, 125
  - GetNumberOfSupportedGroups, 126
- CDeviceGroupChannelInfoGenericNet, 126
  - CDeviceGroupChannelInfoGenericNet, 126
- CDeviceGroupChannelInfoMEA2100\_256Net, 126
  - CDeviceGroupChannelInfoMEA2100\_256Net, 127
- CDeviceGroupChannelInfoNet, 127
  - CDeviceGroupChannelInfoNet, 127
- CDeviceGroupChannelInfoSCUNet, 128
  - CDeviceGroupChannelInfoSCUNet, 128
- CDeviceGroupChannelInfoTemplateNet

- CDeviceGroupChannelInfoTemplateNet< Dacq-GroupChannelEnumTemplateNet >, 128
- CDeviceGroupChannelInfoTemplateNet< Dacq-GroupChannelEnumTemplateNet >, 128
- CDeviceGroupChannelInfoTemplateNet, 128
- GroupID, 129
- GroupType, 129
- NumberOfChannels, 129
- CDeviceGroupChannelInfoW2100Net, 129
- CDeviceGroupChannelInfoW2100Net, 129
- CDigOutStimulatorFunctionNet, 130
- !CDigOutStimulatorFunctionNet, 131
- ~CDigOutStimulatorFunctionNet, 131
- CDigOutStimulatorFunctionNet, 131
- ClearChannel, 131
- GetGlobalRepeat, 131
- GetNumberOfChannels, 132
- GetStartTriggerSlope, 132
- GetStopTriggerSlope, 132
- PrepareChannelData, 132
- SendChannelData, 133
- SetGlobalRepeat, 133
- SetStartTriggerSlope, 133
- SetStopTriggerSlope, 134
- CEncapsulatorDeviceNet, 134
- CEncapsulatorDeviceNet, 134
- GetRoboFluidDevice, 135
- CExternDTesterDeviceNet, 135
- !CExternDTesterDeviceNet, 136
- ~CExternDTesterDeviceNet, 136
- CExternDTesterDeviceNet, 135
- Read, 136
- Read2, 136
- Write, 136
- Write2, 137
- CFilterCoefficientsNet, 137
- ~CFilterCoefficientsNet, 138
- A, 139
- B, 139
- CFilterCoefficientsNet, 137, 138
- GetUIntA, 138
- GetUIntB, 138
- GetUInts, 138
- IsEqual, 138
- CFilterCoefficientsNet::s\_FilterAttributesNet, 714
- CommaPositionA, 715
- CommaPositionB, 715
- PostCommaA, 715
- PostCommaB, 715
- PreCommaA, 715
- PreCommaB, 715
- s\_FilterAttributesNet, 714
- ToCpp, 714
- CFilterConfigurationNet, 139
- CFilterConfigurationNet, 139
- EraseFilterParameterPermanent, 140
- GetDefaultFilterAttributes, 140
- GetFilterAttributes, 140
- GetHighpassFilterEnable, 140
- ResetHighpassFilter, 140
- SetFilterParameter, 140
- SetFilterParameterPermanent, 141
- SetHighpassFilterEnable, 141
- CFilterConfigurationRegisterNet, 141
- CFilterConfigurationRegisterNet, 141
- EraseFilterParameterPermanent, 142
- SetFilterParameter, 142
- SetFilterParameterPermanent, 142
- CFilterPropertyNet, 143
- ~CFilterPropertyNet, 143
- CFilterPropertyNet, 143
- CornerFrequency, 144
- CornerFrequencycHz, 144
- FilterActive, 144
- FilterBand, 144
- FilterFamily, 144
- FilterType, 144
- Order, 144
- ToString, 143
- CFirmwareDestinationNet
- Mcs::Usb, 53
- CFluidControlDeviceNet, 144
- ~CFluidControlDeviceNet, 146
- CalibrateThermocouple, 146
- CFluidControlDeviceNet, 146
- GetAdc, 146
- GetDigin, 146
- GetDigout, 147
- GetPWM, 147
- GetReferenceTemperature, 147
- GetSingleValve, 147
- GetThermocoupleCalibration, 148
- GetThermocoupleNanovoltPerKelvin, 148
- GetThermocoupleTemperature, 148
- GetValve, 149
- McsBus\_VoltageMode, 151
- SetDigout, 149
- SetPWM, 149
- SetSingleValve, 149
- SetThermocoupleNanovoltPerKelvin, 151
- SetValve, 151
- CFYIDeviceNet, 151
- CFYIDeviceNet, 152
- FYIProgram, 152
- FYITemp, 152
- Sensor, 152
- CGenericDevelopDeviceNet, 153
- ~CGenericDevelopDeviceNet, 160
- CGenericDevelopDeviceNet, 160
- ClosePipe, 160
- FindEndpoints, 160
- GetBuffer, 160
- GetByteBuffer, 161
- GetIntBuffer, 161
- GetShortBuffer, 162
- GetUByteBuffer, 163

- GetUIntBuffer, 163
- GetUShortBuffer, 164
- OpenPipe, 165
- ReadPipe, 165
- ResetPipe, 165
- SetBuffer, 166
- SetByteBuffer, 166
- SetIntBuffer, 166
- SetShortBuffer, 167
- SetUByteBuffer, 168
- SetUIntBuffer, 168
- SetUShortBuffer, 169
- SetValue, 169
- VendorInRequest, 170
- VendorOutRequest, 170
- WritePipe, 170
- CGilsonDeviceNet, 171
  - ~CGilsonDeviceNet, 172
  - CGilsonDeviceNet, 171
  - ConnectSlave, 172
  - GetLastAnswer, 172
  - m\_pGilsonDevice, 172
  - SendBuffered, 172
  - SendImmediate, 172
  - SendImmediateGetResponse, 172
- CGrapheneASICDeviceNet, 173
  - ~CGrapheneASICDeviceNet, 173
  - CGrapheneASICDeviceNet, 173
  - GetAvailableBaseSamplerates, 173
  - GetBaseSamplerate, 174
  - SetBaseSamplerate, 174
  - SetRegionOfInterests, 174
- CGrapheneFunctionNet, 174
  - ICGrapheneFunctionNet, 177
  - ~CGrapheneFunctionNet, 176
  - CGrapheneFunctionNet, 176
  - GetADCGain, 177
  - GetADCOffset, 177, 178
  - GetCur2VolResistance, 178
  - GetDACOffset, 179
  - GetVdsVgs, 179
  - GetVdVs, 180
  - GetVdVsDAC, 180, 181
  - GetVoltageRange, 181
  - GetVoltageReached, 181
  - GetVoltageResolution, 182
  - SetADCGain, 182, 183
  - SetADCOffset, 183
  - SetDACOffset, 183, 184
  - SetVdsVgs, 184
  - SetVdVs, 185
  - SetVdVsDAC, 185
  - SetVoltageRange, 186
  - SetVoltageResolution, 186
- ChangeSerialNumber
  - CMcsUsbFactoryNet, 302
- channel
  - CMeaAudioFunctionNet::s\_setaudionet, 715
  - CW2100\_FunctionNet::AudioChannelsNet, 95
- ChannelBlock
  - CMcsUsbDacqNet, 297
- ChannelBlockNet
  - CMcsUsbDacqNet::ChannelBlockNet, 189
- ChannelBlockTypeNet
  - Mcs::Usb, 57
- channeldata\_current
  - Mcs::Usb, 85
- channeldata\_current\_always\_boost
  - Mcs::Usb, 85
- channeldata\_current\_always\_boost\_own\_sync
  - Mcs::Usb, 85
- channeldata\_current\_own\_boost\_gnd\_sync
  - Mcs::Usb, 85
- channeldata\_current\_own\_sync
  - Mcs::Usb, 85
- channeldata\_positive\_current
  - Mcs::Usb, 85
- channeldata\_positive\_current\_own\_boost\_gnd\_sync
  - Mcs::Usb, 85
- channeldata\_positive\_current\_own\_sync
  - Mcs::Usb, 85
- channeldata\_positive\_voltage
  - Mcs::Usb, 85
- channeldata\_voltage
  - Mcs::Usb, 85
- ChannelDataEvent
  - CMcsUsbDacqNet, 297
- ChannelPIC
  - Mcs::Usb, 55
- ChannelReset
  - CMultiBatteryChargerDeviceNet, 376
- ChannelTest
  - Mcs::Usb, 77
- Charge
  - BatteryState, 95
- ChargeRegionString
  - BatteryState, 95
- ChargeString
  - BatteryState, 95
- ChecksumAndPacketCounter
  - Mcs::Usb, 60
- CHiClampDeviceNet, 211
  - CHiClampDeviceNet, 212
  - RoboDacq, 212
- CHLADacqNet, 212
  - CHLADacqNet, 213
- CHLADeviceNet, 213
  - CHLADeviceNet, 213
  - HLADacq, 214
  - SerialPort, 214
- CHWInfo
  - CMcsUsbDacqNet::CHWInfo, 214
- Ci4600Intan
  - Mcs::Usb, 52
- CIntanMea\_FunctionNet, 217
  - AmplifierSettle, 218



- BeginImpedanceCheck, 218
- CIntanMea\_FunctionNet, 217
- GetDSPHighPassByIndex, 218
- GetImpedanceResult, 218
- GetIntanRegister, 218
- GetLowerFrequencyByIndex, 218
- GetUpperFrequencyByIndex, 218
- SetBandwidthByIndex, 218
- SetDiagnosticMode, 218
- SetDSPHighPassByIndex, 219
- SetIntanRegister, 219
- CInterfaceboard2FunctionNet, 219
  - !CInterfaceboard2FunctionNet, 220
  - ~CInterfaceboard2FunctionNet, 220
  - CInterfaceboard2FunctionNet, 220
  - GetIoVoltage, 220
  - SetIoVoltage, 220
- CInterfaceboardFunctionNet, 221
  - !CInterfaceboardFunctionNet, 222
  - ~CInterfaceboardFunctionNet, 222
  - CInterfaceboardFunctionNet, 221
  - GetCardinalDacqSamplerate, 222
  - GetCardinalStgOutputrate, 222
  - SetCardinalDacqSamplerate, 222
  - SetCardinalStgOutputrate, 222
- ClampAmpRestart
  - CRoboDacqNet, 459
- ClampModeCurrent
  - Mcs::Usb, 86
- ClampModeInternalCalibration
  - Mcs::Usb, 86
- ClampModeOpen
  - Mcs::Usb, 86
- ClampModeVoltage
  - Mcs::Usb, 86
- ClearBuffers
  - CMcsUsbDacqNet, 279
- ClearChannel
  - CDigOutStimulatorFunctionNet, 131
- ClearChannel\_PrepAndSendData
  - CStg200xDownloadNet, 572
  - CStimulusFunctionNet, 579
- ClearChannelData
  - CStg200xDownloadBasicNet, 563
  - CStimulusFunctionNet, 579
  - CW2100\_StimulatorFunctionNet, 628
- ClearMultiplexedData
  - CStimulusFunctionNet, 579
- ClearSTGOutput
  - CCMOSMea\_FunctionNet, 106
- ClearStimulusParametersCache
  - CW2100\_FunctionNet, 621
- ClearSyncData
  - CStg200xDownloadBasicNet, 563
  - CStimulusFunctionNet, 579
- ClearTable
  - CRetinalLedDeviceNet, 447
- ClearTableName
  - CWarnerValveControllerDeviceNet, 656
- ClearUserDefinedNameCache
  - CW2100\_FunctionNet, 621
- ClearValveTable
  - CWarnerValveControllerDeviceNet, 656
- CLIH3DeviceNet, 223
  - !CLIH3DeviceNet, 225
  - ~CLIH3DeviceNet, 225
  - CLIH3DeviceNet, 225
  - DummyCommand, 225
  - EnableUserTrigger, 225
  - ErasePermanentAdcOffset, 226
  - ErasePermanentDacOffset, 226
  - GetAdcOffset, 226
  - GetAudioOutDacParameter, 226
  - GetCardinalDacqSamplerate, 227
  - GetCardinalStgOutputrate, 227
  - GetDacIdleValue, 227
  - GetDacOffset, 227
  - GetDacqRunStatus, 228
  - GetDacUseldleValue, 228
  - GetDigInState, 228
  - GetEepromPage, 228
  - GetSampleInterval, 229
  - IsUserTriggerEnabled, 229
  - ReadClipping, 229
  - ReadUARTData, 229
  - SendCommand, 230
  - SetAdcOffset, 230
  - SetAdcOffsetPermanent, 230
  - SetAudioOutDacParameter, 230
  - SetDacIdleValue, 231
  - SetDacOffset, 231
  - SetDacOffsetPermanent, 231
  - SetDacUseldleValue, 231
  - SetDigOutState, 232
  - SetEepromPage, 232
  - SetSampleInterval, 232
  - StimulusFunction, 233
  - WriteUARTData, 232
- Close
  - Mcs::Usb, 75
- CloseAllValves
  - CRoboFluidDeviceNet, 488
- ClosePipe
  - CGenericDevelopDeviceNet, 160
- ClosePlateClamp
  - CMultiwellDeviceNet, 386
- CMcsBus\_AxisParametersNet, 233
  - ~CMcsBus\_AxisParametersNet, 233
  - CMcsBus\_AxisParametersNet, 233
  - GetAxisParametersSignedEeprom, 234
  - GetAxisParametersUnsignedEeprom, 234
  - SetAxisParametersEeprom, 234
- CMcsBus\_ExtensionNet, 234
  - ~CMcsBus\_ExtensionNet, 235
  - CMcsBus\_ExtensionNet, 235
  - GetLEDSwitch, 235

- SetLEDSwitch, 235
- CMcsBus\_FYIExtensionNet, 235
  - ~CMcsBus\_FYIExtensionNet, 236
  - CMcsBus\_FYIExtensionNet, 236
  - GetDIO, 236
  - GetSingleHeater, 236
  - GetValves, 236
  - SetDIO, 236
  - SetSingleHeater, 237
  - SetValves, 237
- CMcsBus\_MotorControlNet, 237
  - ~CMcsBus\_MotorControlNet, 240
  - CMcsBus\_MotorControlNet, 240
  - GetMCAcceleration, 240
  - GetMCAccelerationEeprom, 240
  - GetMCAccelerationShortCommand, 241
  - GetMCAXisRevisionEeprom, 241
  - GetMCBreakCurrent, 241
  - GetMCBreakCurrentEeprom, 241
  - GetMCConfig, 241
  - GetMCConfigEeprom, 241
  - GetMCCurrent, 241
  - GetMCCurrentEeprom, 242
  - GetMCCurrentMode, 242
  - GetMCCurrentModeEeprom, 242
  - GetMCCurrentModeShortCommand, 242
  - GetMCCurrentPosition, 242
  - GetMCCurrentShortCommand, 242
  - GetMCCurrentSpeed, 242
  - GetMCMaxAcceleration, 243
  - GetMCMaxAccelerationEeprom, 243
  - GetMCMaxCurrent, 243
  - GetMCMaxCurrentEeprom, 243
  - GetMCMaxSpeed, 243
  - GetMCMaxSpeedEeprom, 243
  - GetMCMaxTravel, 243
  - GetMCMaxTravelEeprom, 244
  - GetMCMaxTravelShortCommand, 244
  - GetMCMovement, 244
  - GetMCNewPosition, 244
  - GetMCOutputOnOff, 244
  - GetMCPhase, 244
  - GetMCPhaseOffset, 244
  - GetMCReference, 245
  - GetMCReferenceCurrent, 245
  - GetMCReferenceCurrentEeprom, 245
  - GetMCRegulatorGain, 245
  - GetMCRegulatorGainEeprom, 245
  - GetMCScalingFactor, 245
  - GetMCScalingFactorEeprom, 245
  - GetMCSpeed, 246
  - GetMCSpeedEeprom, 246
  - GetMCSpeedShortCommand, 246
  - GetMCSpeedUnitEeprom, 246
  - GetMCStandbyCurrent, 246
  - GetMCStandbyCurrentEeprom, 246
  - GetMCStandbyTime, 246
  - GetMCStandbyTimeEeprom, 247
- GetSubChannel, 247
- SetMCAcceleration, 247
- SetMCAccelerationEeprom, 247
- SetMCAccelerationShortCommand, 247
- SetMCAXisRevisionEeprom, 247
- SetMCBreakCurrent, 247
- SetMCBreakCurrentEeprom, 248
- SetMCConfig, 248
- SetMCConfigEeprom, 248
- SetMCCurrent, 248
- SetMCCurrentEeprom, 248
- SetMCCurrentMode, 248
- SetMCCurrentModeEeprom, 249
- SetMCCurrentModeShortCommand, 249
- SetMCCurrentPosition, 249
- SetMCCurrentShortCommand, 249
- SetMCMaxAcceleration, 249
- SetMCMaxAccelerationEeprom, 249
- SetMCMaxCurrent, 250
- SetMCMaxCurrentEeprom, 250
- SetMCMaxSpeed, 250
- SetMCMaxSpeedEeprom, 250
- SetMCMaxTravel, 250
- SetMCMaxTravelEeprom, 250
- SetMCMaxTravelShortCommand, 251
- SetMCNewPosition, 251
- SetMCOutputOnOff, 251
- SetMCReference, 251
- SetMCReferenceCurrent, 251
- SetMCReferenceCurrentEeprom, 251
- SetMCRegulatorGain, 252
- SetMCRegulatorGainEeprom, 252
- SetMCRotation, 252
- SetMCScalingFactor, 252
- SetMCScalingFactorEeprom, 252
- SetMCSpeed, 252
- SetMCSpeedEeprom, 253
- SetMCSpeedShortCommand, 253
- SetMCSpeedUnitEeprom, 253
- SetMCStandbyCurrent, 253
- SetMCStandbyCurrentEeprom, 253
- SetMCStandbyTime, 253
- SetMCStandbyTimeEeprom, 254
- SetSubChannel, 254
- StartMCMovement, 254
- StopMCMovement, 254
- CMcsBus\_SensorNet, 254
  - ~CMcsBus\_SensorNet, 256
  - CatchAmpGetAdcMean, 256
  - CatchAmpGetAdcValue, 257
  - CatchAmpGetAdcValueH, 257
  - CatchAmpGetAdcValueL, 257
  - CatchAmpGetDacAmplitude, 257
  - CatchAmpGetDacEnable, 257
  - CatchAmpGetDacOffset, 257
  - CatchAmpGetPwmEnable, 257
  - CatchAmpSetDacAmplitude, 257
  - CatchAmpSetDacEnable, 258



- CatchAmpSetDacOffset, [258](#)
- CatchAmpSetPwmEnable, [258](#)
- CMcsBus\_SensorNet, [256](#)
- Get2AnalogInput, [258](#)
- Get2DigitalInput, [258](#)
- Get4ADC, [258](#)
- Get4ADCAverage, [258](#)
- Get4ADCCatchampAverageShift, [258](#)
- Get4ADCMode, [259](#)
- Get4DAC, [259](#)
- GetADCs, [259](#)
- GetADCsLoop, [259](#)
- GetBubbleStatus, [259](#)
- GetDACs, [259](#)
- GetDetectionThreshold, [259](#)
- GetDetectorValue, [259](#)
- GetLatency, [260](#)
- GetLatencyCounter, [260](#)
- GetMinimalThreshold, [260](#)
- GetMovePump, [260](#)
- GetPiezoState, [260](#)
- GetPressure, [260](#)
- GetPressureOffset, [261](#)
- GetRegulationTimeouts, [261](#)
- GetRegulatorFactor, [261](#)
- GetRegulatorOnOff, [261](#)
- GetRegulatorStatus, [261](#)
- GetRotatePump, [261](#)
- GetSamplePeriode, [262](#)
- GetSollPressure, [262](#)
- GetSyncState, [262](#)
- Set4ADCCatchampAverageShift, [262](#)
- Set4ADCMode, [262](#)
- Set4DAC, [262](#)
- SetDACs, [262](#)
- SetDetectionThreshold, [263](#)
- SetLatency, [263](#)
- SetMinimalThreshold, [263](#)
- SetMovePump, [263](#)
- SetPiezoState, [263](#)
- SetPressureOffset, [263](#)
- SetRegulationTimeouts, [263](#)
- SetRegulatorFactor, [264](#)
- SetRegulatorOnOff, [264](#)
- SetRotatePump, [264](#)
- SetSamplePeriode, [264](#)
- SetSollPressure, [264](#)
- StartSync, [264](#)
- TactSwitchGetState, [264](#)
- TactSwitchSetDisplay, [265](#)
- CMcsBus\_TempSensorNet, [265](#)
  - ~CMcsBus\_TempSensorNet, [265](#)
  - CMcsBus\_TempSensorNet, [265](#)
  - GetNanoVoltsPerKelvin, [266](#)
  - GetTemperatur, [266](#)
  - GetThermoOffset, [266](#)
  - GetThermoTemp, [266](#)
  - GetThermoVoltage, [266](#)
  - SetNanoVoltsPerKelvin, [266](#)
  - SetThermoOffset, [266](#)
- CMcsBus\_VoltageModeNet, [267](#)
  - ~CMcsBus\_VoltageModeNet, [268](#)
  - CMcsBus\_VoltageModeNet, [268](#)
  - GetVMMaxNegativeCurrent, [268](#)
  - GetVMMaxNegativeCurrentEeprom, [268](#)
  - GetVMMaxNegativeVoltage, [268](#)
  - GetVMMaxNegativeVoltageEeprom, [268](#)
  - GetVMMaxPositiveCurrent, [269](#)
  - GetVMMaxPositiveCurrentEeprom, [269](#)
  - GetVMMaxPositiveVoltage, [269](#)
  - GetVMMaxPositiveVoltageEeprom, [269](#)
  - GetVMOutputOnOff, [269](#)
  - GetVMVoltage, [269](#)
  - SetVMMaxNegativeCurrent, [269](#)
  - SetVMMaxNegativeCurrentEeprom, [270](#)
  - SetVMMaxNegativeVoltage, [270](#)
  - SetVMMaxNegativeVoltageEeprom, [270](#)
  - SetVMMaxPositiveCurrent, [270](#)
  - SetVMMaxPositiveCurrentEeprom, [270](#)
  - SetVMMaxPositiveVoltage, [270](#)
  - SetVMMaxPositiveVoltageEeprom, [271](#)
  - SetVMOutputOnOff, [271](#)
  - SetVMVoltage, [271](#)
- CMcsBusNet, [271](#)
  - ~CMcsBusNet, [272](#)
  - CMcsBusNet, [272](#)
  - CMcsBusNet::GetMode, [272](#)
  - CMcsBusNet::GetModeEeprom, [272](#)
  - CMcsBusNet::SetMode, [273](#)
  - CMcsBusNet::SetModeEeprom, [273](#)
  - GetBusAddress, [273](#)
  - GetBusAddressEeprom, [273](#)
  - GetCommand, [273](#), [274](#)
  - GetHWRevisionEeprom, [274](#)
  - SetBusAddress, [274](#)
  - SetBusAddressEeprom, [274](#)
  - SetCommand, [274](#), [275](#)
  - SetHWRevisionEeprom, [275](#)
- CMcsBusNet::GetMode
  - CMcsBusNet, [272](#)
- CMcsBusNet::GetModeEeprom
  - CMcsBusNet, [272](#)
- CMcsBusNet::SetMode
  - CMcsBusNet, [273](#)
- CMcsBusNet::SetModeEeprom
  - CMcsBusNet, [273](#)
- CMcsUsbDacqNet, [275](#)
  - ~CMcsUsbDacqNet, [279](#)
  - ChannelBlock, [297](#)
  - ChannelDataEvent, [297](#)
  - ClearBuffers, [299](#)
  - CMcsUsbDacqNet, [279](#)
  - CMcsUsbDacqNet::GetFilterProperties, [279](#)
  - Error\_Callback\_Aquisition\_Stopped, [297](#)
  - Error\_Callback\_Data\_lost, [297](#)
  - Error\_Callback\_Frames\_Lost, [297](#)

- Error\_Callback\_Packet\_Error, 297
- Error\_Callback\_Queue\_Full, 297
- Error\_Callback\_RingQueue\_Full, 297
- ErrorEvent, 298
- GetAdapterType, 280
- GetAdcDataFormat, 280
- GetAdcZero, 280
- GetAnalogValueUnit, 280
- GetChannelDataFillSize, 280
- GetChannelLayout, 280
- GetChannelsInBlock, 281
- GetDataFormat, 281
- GetDataMode, 281
- GetDigitalSource, 281–284
- GetErrorMessage, 284
- GetFilterProperty, 284
- GetHardwareMaxRange, 284
- GetHardwareMinRange, 284
- GetMaxSamplingFrequency, 284
- GetMeaLayout, 285
- GetMinSamplingFrequencyStepsize, 285
- GetNumberOfDataBits, 285
- GetPoti, 285
- GetResolutionPerDigit, 285
- GetSamplerate, 286
- GetSampleratePerFramerate, 286
- GetVoltageRangeIndex, 286
- GetVoltageRangeInMicroVolt, 286
- GetVoltageRangeInMilliVolt, 286
- HWInfo, 287
- Samplerate, 297
- SendStartDacq, 287
- SendStartStgAndDacq, 287
- SendStopDacq, 287, 288
- SendStopStgAndDacq, 288
- SendStopStgAndDacqWithOptions, 288
- SetDataMode, 289
- SetDigitalSource, 289–291
- SetPoti, 292
- SetSamplerate, 292
- SetVoltageRangeByIndex, 292
- StartDacq, 292, 293
- StartLoop, 294, 295
- StopDacq, 296
- StopLoop, 296
- CMcsUsbDacqNet::ChannelBlockNet, 187
  - AddBlock, 189
  - AddBlocksAndChannels, 189, 190
  - AddChannel, 192
  - AddSelectedChannelsQueue, 192, 193
  - AvailFrames, 194
  - ChannelBlockNet, 189
  - GetChannel, 194
  - GetCheckChecksum, 195
  - GetGroupChannelDataI16, 195
  - GetGroupChannelDataI32, 195
  - GetGroupChannelDataUI16, 196
  - GetGroupChannelDataUI32, 196
- GetNListTrue, 197
- GetNumberOfQueues, 197
- Init, 197
- ReadAsFrameArrayI16, 197
- ReadAsFrameArrayI32, 198
- ReadAsFrameArrayUI16, 199
- ReadAsFrameArrayUI32, 199
- ReadFramesDictI16, 200
- ReadFramesDictI32, 201
- ReadFramesDictUI16, 201
- ReadFramesDictUI32, 202
- ReadFramesI16, 202
- ReadFramesI32, 203
- ReadFramesUI16, 204
- ReadFramesUI32, 204
- SetCheckChecksum, 205
- SetCommonThreshold, 205
- SetQueueSize, 205
- SetQueueSizeAll, 205
- SetSelectedChannels, 205, 206
- SetSelectedChannelsQueue, 207, 208
- SetSelectedData, 209, 210
- SetupGroupDacqQueue, 211
- UpdateQueueSize, 211
- CMcsUsbDacqNet::CHWInfo, 214
  - CHWInfo, 214
  - GetAvailableSampleRates, 215
  - GetAvailableVoltageRangesInMicroVolt, 215
  - GetAvailableVoltageRangesInMicroVoltAndStringsInMilliVolt, 215
  - GetNumberOfHWADCCChannels, 216
  - GetNumberOfHWDigitalChannels, 216
  - IsDigitalChannelDedicated, 216
- CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet, 618
  - CVoltageRangeInfoNet, 619
  - VoltageRangeDisplayStringMilliVolt, 619
  - VoltageRangeInMicroVolt, 619
- CMcsUsbDacqNet::GetFilterProperties
  - CMcsUsbDacqNet, 279
- CMcsUsbDeviceStatePushFunctionNet, 298
  - CMcsUsbDeviceStatePushFunctionNet, 298
  - McsUsbDeviceStateEvent, 299
  - TriggerStatus, 298
- CMcsUsbDeviceStatePushNet, 299
  - CMcsUsbDeviceStatePushNet, 299
  - McsUsbDeviceStateEvent, 300
  - TriggerStatus, 300
- CMcsUsbFactoryNet, 300
  - ~CMcsUsbFactoryNet, 302
  - ChangeSerialNumber, 302
  - CMcsUsbFactoryNet, 301
  - Coldstart, 302
  - CompareFirmware, 302
  - DownloadFirmware, 302
  - FindFirmwareVersionMagicInBuffer, 302
  - FX3MCSDDataAddress, 307
  - FX3MCSDDataDeviceIdOffset, 308

- FX3MCSDDataIFB1ImageOffset, 308
- FX3MCSDDataIFB2ImageOffset, 308
- FX3MCSDDataVersionOffset, 308
- GetChecksumFromFX3Image, 302
- GetDestination, 303
- GetDestinationDisplayLabel, 303
- GetDestinationName, 303
- GetDestinationSerialNumber, 303
- GetDestinationTargetAddress, 303
- GetFirmwareVersionFromFile, 304
- GetFirmwareVersionFromHexFile, 304
- GetNumDestinations, 304
- GetUSBDeviceIDFromFX3Image, 304
- GetUserCodeFromBitFile, 304
- GetUserCodeFromFlash, 304
- GetXilinxFlashOffset, 304
- GetXilinxFlashReadCommand, 305
- LoadUserFirmware, 305
- ReadBlockFromFlash, 305
- ReadBlockFromIFBGlobalEEPROM, 305
- ReadBlockFromNVMEM, 306
- SetDestinationSerialNumber, 306
- UpdateFirmware, 306, 307
- CMcsUsbFunctionNet, 308
  - !CMcsUsbFunctionNet, 309
  - ~CMcsUsbFunctionNet, 309
  - CMcsUsbFunctionNet, 309
  - m\_pMcsUsb, 310
  - m\_pMcsUsbFunction, 310
  - ThrowCUsbExceptionNetOnError, 309
- CMcsUsbFunctionPointerContainer, 310
- CMcsUsbListEntryNet, 310
  - ~CMcsUsbListEntryNet, 311
  - DeviceId, 313
  - DeviceName, 313
  - Equals, 311
  - GetEntry, 311, 312
  - GetEntryCount, 312
  - HwVersion, 313
  - Manufacturer, 313
  - Product, 314
  - SerialNumber, 314
  - SetStringFormat, 313
  - ToString, 313
- CMcsUsbListNet, 314
  - !CMcsUsbListNet, 315
  - ~CMcsUsbListNet, 315
  - CMcsUsbListNet, 315
  - Count, 316
  - DeviceArrival, 316
  - DeviceRemoval, 316
  - GetNumberOfDevices, 315
  - GetUsbListEntries, 315
  - GetUsbListEntry, 315
  - IsDeviceTypeOf, 316
  - SetStringFormat, 316
- CMcsUsbNet, 317
  - !CMcsUsbNet, 321
  - ~CMcsUsbNet, 321
  - AddSoftwareKey, 321
  - AssociateToThis, 321
  - CMcsUsbNet, 321
  - Connect, 321, 322
  - CyclePort, 323
  - Disconnect, 323
  - EmptyKey, 323
  - EnableExceptions, 323
  - EraseEEPROMRegisterPreconfig, 323
  - GetConfiguration, 324
  - GetDeviceCannotStallOutRequests, 324
  - GetDeviceCapableSpeed, 324
  - GetDeviceEnum, 324
  - GetDeviceId, 324
  - GetDeviceRootHubVendorEnum, 324
  - GetDeviceRootHubVendorID, 324
  - GetDeviceRootHubVendorName, 324
  - GetDeviceSpeed, 325
  - GetErrorText, 325
  - GetFirmwareVersion, 325
  - GetHardwareRevision, 325
  - GetHeadstageActive, 326
  - GetHeadstageID, 326
  - GetHeadstagePresent, 326
  - GetLastUSBError, 326
  - GetMea21UsbPort, 327
  - GetNumConfigurations, 327
  - GetSerialNumber, 327
  - GetSoftwareKey, 327
  - GetSoftwareKeyString, 327
  - GetStatus, 327
  - GetStatusOfLastCommand, 327
  - GetUsbListEntry, 328
  - GetVersion, 328
  - HasSoftwareKey, 328
  - IsConnected, 328
  - IsDeviceHighSpeed, 328
  - IsDeviceHighSpeedCapable, 328
  - IsExceptionsEnabled, 329
  - ModifyRegister, 329
  - MultibootGetCypressImageId, 329
  - MultibootGetImageId, 329
  - MultibootGetSelectedImage, 329
  - MultibootSelectImage, 329
  - ReadEEPROMRegisterPreconfig, 330
  - ReadRegister, 330
  - ReadRegister32, 330
  - ReadRegisterTimeSlot, 330
  - RemoveSoftwareKey, 331
  - RescanHeadstage, 331
  - SerialNumber, 338
  - SetConfiguration, 331
  - SetSoftwareKey, 331
  - Status\_AlreadyConfigured, 333
  - Status\_BadStartFrame, 334
  - Status\_Btstuff, 334
  - Status\_BufferOverrun, 334

- Status\_BufferUnderrun, 334
- Status\_Canceled, 334
- Status\_Canceling, 334
- Status\_ConnectedPipes, 334
- Status\_ControlNotOwned, 334
- Status\_Crc, 334
- Status\_DataOverrun, 334
- Status\_DataToggleMismatch, 334
- Status\_DataUnderrun, 335
- Status\_DeviceLocked, 335
- Status\_DeviceNotFound, 335
- Status\_DeviceRemoved, 335
- Status\_DevNotResponding, 335
- Status\_EndpointHalted, 335
- Status\_ErrorBusy, 335
- Status\_ErrorShortTransfer, 335
- Status\_Fifo, 335
- Status\_FrameControlOwned, 335
- Status\_InternalHcError, 335
- Status\_InvalidDeviceHandle, 336
- Status\_InvalidHandle, 336
- Status\_InvalidParameter, 336
- Status\_InvalidPipeHandle, 336
- Status\_InvalidUrbFunction, 336
- Status\_IoPending, 336
- Status\_IoTimeout, 336
- Status\_IsochRequestFailed, 336
- Status\_LastUsbErrorMismatch, 336
- Status\_NoBandwidth, 336
- Status\_NoMemory, 337
- Status\_NoSuchDevice, 337
- Status\_NotAccessed, 337
- Status\_NotSupported, 337
- Status\_PidCheckFailure, 337
- Status\_PipeNotLinked, 337
- Status\_RequestFailed, 337
- Status\_RequestMutexFailed, 337
- Status\_RequestMutexTimeout, 337
- Status\_Stall, 337
- Status\_Unconfigured, 337
- Status\_UnexpectedPid, 338
- ThrowCUsbExceptionNetOnError, 331
- TxnGetSerialNumber, 331
- TxnSetSerialNumber, 331
- TxnTestMemoryReadAndCheck, 331
- TxnTestMemoryWrite, 332
- ValidKey, 332
- WPAError\_ScanningIsPending, 338
- WriteEepromRegisterPreconfig, 332
- WriteRegister, 332, 333
- WriteRegister32, 333
- WriteRegisterArray, 333
- WriteRegisterTimeSlot, 333
- WriteRegisterValue, 333
- CMcsUsbPointerContainer, 338
- CMEA2100\_256DacqGroupChannelSelectionNet, 338
  - CMEA2100\_256DacqGroupChannelSelectionNet, 338
- CMEA2100x256FunctionNet, 339
  - !CMEA2100x256FunctionNet, 340
  - ~CMEA2100x256FunctionNet, 340
  - CMEA2100x256FunctionNet, 339
  - GetLayoutConfiguration, 340
  - SetLayoutConfiguration, 340
- CMeaAudioFunctionNet, 340
  - CMeaAudioFunctionNet, 341
  - GetAudioChannels, 341, 342
  - GetNumberOfAudioChannels, 342
  - SetAudioChannels, 342, 343
- CMeaAudioFunctionNet::s\_setaudionet, 715
  - amplification, 715
  - channel, 715
- CMeaCleanDeviceNet, 343
  - !CMeaCleanDeviceNet, 345
  - ~CMeaCleanDeviceNet, 344
  - CMeaCleanDeviceNet, 344
  - GetCycle, 345
  - GetCycles, 345
  - GetMaxVoltage, 345
  - GetMinVoltage, 345
  - GetOutputVoltage, 345
  - GetSlope, 346
  - IsRunning, 346
  - SetCycles, 346
  - SetMaxVoltage, 346
  - SetMinVoltage, 347
  - SetSlope, 347
  - Start, 347
  - Stop, 347
- CMeaCoatDeviceNet, 347
  - !CMeaCoatDeviceNet, 349
  - ~CMeaCoatDeviceNet, 349
  - CMeaCoatDeviceNet, 349
  - GetCurrentCycle, 349
  - GetCycles, 349
  - GetDuration, 349
  - GetMaxCurrent, 349
  - GetOffsetCurrent, 350
  - GetOutputCurrent, 350
  - GetPauseDuration, 350
  - GetSlope, 350
  - GetTimeInPause, 350
  - GetTimeInPlateau, 351
  - IsRunning, 351
  - SetCycles, 351
  - SetDuration, 351
  - SetMaxCurrent, 352
  - SetOffsetCurrent, 352
  - SetPauseDuration, 352
  - SetSlope, 352
  - Start, 352
  - Stop, 353
- CMeaDeviceNet, 353
  - ~CMeaDeviceNet, 355
  - AnalogGain, 360
  - CMeaDeviceNet, 354, 355

- EnableChecksum, 355
- EnableDigitalIn, 356
- EnableTimestamp, 356
- Gain, 360
- GetAnalogGain, 357
- GetEnumerationSpeed, 357
- GetGain, 357
- MeaAudioFunctionNet, 361
- MeaDigitalDataFunctionNet, 361
- MeaFeedbackFunctionNet, 361
- MeFunctionNet, 361
- SetDigitalOut, 357
- SetNumberOfAnalogChannels, 358
- SetNumberOfChannels, 359
- SetTriggerMaskValue, 359
- SetTriggerPeriod, 360
- W2100\_FunctionNet, 361
- WClassicFunctionNet, 361
- CMeaDigitalDataFunctionNet, 361
  - CMeaDigitalDataFunctionNet, 362
  - GetDigitalData, 362
  - SetDigitalData, 362, 363
- CMeaFeedbackFunctionNet, 363
  - CMeaFeedbackFunctionNet, 364
  - FeedbackGetSampleTimerCount, 364
  - FeedbackGetTriggerTotzeitFactor, 364
  - FeedbackSetAnalogSource, 365
  - FeedbackSetChannelFilter, 365
  - FeedbackSetCreateFilter, 365
  - FeedbackSetDigitalMapping, 365
  - FeedbackSetFeedback, 365
  - FeedbackSetFilterOff, 365
  - FeedbackSetFilterParameter, 365
  - FeedbackSetFilterParameter32, 365
  - FeedbackSetGlobalChannelFilter, 366
  - FeedbackSetIIRFilterParameter, 366
  - FeedbackSetLogic, 366
  - FeedbackSetMkFilter, 366
  - FeedbackSetNumberOfLogics, 366
  - FeedbackSetNumberOfRateCounter, 366
  - FeedbackSetNumberOfRateDetectors, 366
  - FeedbackSetNumberOfSpikeDetectors, 367
  - FeedbackSetNumberOfTriggers, 367
  - FeedbackSetRateCounter, 367
  - FeedbackSetRateDetector, 367
  - FeedbackSetSpikeDetectorThreshold, 367
  - FeedbackSetTrigger, 367
  - FeedbackSetTriggerTotzeitFactor, 367
- CMealImpedanceDeviceNet, 368
  - ~CMealImpedanceDeviceNet, 368
  - CMealImpedanceDeviceNet, 368
  - GetAdapterCode, 368
  - GetArraySize, 368
  - GetImpedanceTestFrequency, 369
  - GetReady, 369
  - GetResult, 369
  - SetImpedanceTestFrequency, 369
  - StartMeasurement, 369
- CMeasureTableDeviceNet, 369
  - CMeasureTableDeviceNet, 370
  - Sensor, 370
- CMeaSwitchDeviceNet, 370
  - ~CMeaSwitchDeviceNet, 371
  - CMeaSwitchDeviceNet, 371
  - GetNumber, 371
  - GetPattern, 371
  - GetPatternBool, 371
  - SetPattern, 371
  - SetPatternBool, 372
- CMeaUSBDeviceNet, 372
  - ~CMeaUSBDeviceNet, 373
  - CMeaUSBDeviceNet, 372, 373
  - SetVoltageRangeForIndex, 373
- CMeFunctionNet, 373
  - !CMeFunctionNet, 374
  - ~CMeFunctionNet, 374
  - CMeFunctionNet, 374
  - SetTransformer, 374
- CmosMea
  - CCMOSMeaDeviceNet, 117
- CmosMea
  - Mcs::Usb, 68
- CMOSMEA5000DigitalSourceEnumNet
  - Mcs::Usb, 57
- CMOSMeaBathModeEnumNet
  - Mcs::Usb, 58
- CmosMeaHeadstage
  - Mcs::Usb, 67
- CMOSMeaHeadstage1NCBathCurrentEnumNet
  - Mcs::Usb, 58
- CMOSMeaHeadstage1NCCol2CurrentEnumNet
  - Mcs::Usb, 58
- CMOSMeaHeadstage1NChipTempEnumNet
  - Mcs::Usb, 58
- CMOSMeaHS1SidebandEnumNet
  - Mcs::Usb, 58
- CMOSMeaHS1TriggerStatusEnumNet
  - Mcs::Usb, 59
- CmosmealFB2
  - Mcs::Usb, 67
- CMOSMealFDigChannelEnumNet
  - Mcs::Usb, 59
- CMOSMealInterfaceADCEnumNet
  - Mcs::Usb, 59
- CmosMealInterfaceboard
  - Mcs::Usb, 67
- CMOSMealPacketFrameContextGroupEnumNet
  - Mcs::Usb, 60
- CMOSMealSTG1DACSignalEnumNet
  - Mcs::Usb, 60
- CMOSMealValueUnitEnumNet
  - Mcs::Usb, 60
- CMultiBatteryChargerDeviceNet, 374
  - !CMultiBatteryChargerDeviceNet, 376
  - ~CMultiBatteryChargerDeviceNet, 376
  - CapacityTest, 376



- ChannelReset, 376
- CMultiBatteryChargerDeviceNet, 376
- GetBatteryVoltage, 377
- GetChannels, 377
- GetChannelState, 377
- GetChargeCapacity, 377
- GetChargeCurrent, 378
- GetChargingMode, 378
- GetChargingPCoefficient, 378
- GetDischargeCapacity, 378
- GetDischargeCurrent, 379
- GetDischargeCurrentSetPoint, 379
- GetFinalDischargeVoltage, 379
- GetRatedCapacity, 380
- SetChargingMode, 380
- SetChargingPCoefficient, 380
- SetDischargeCurrentSetPoint, 380
- SetFinalDischargeVoltage, 381
- SetRatedCapacity, 381
- SetRatedCapacityVolatile, 381
- CMultiwellCallbackFunctionNet, 382
  - ICMultiwellCallbackFunctionNet, 383
  - ~CMultiwellCallbackFunctionNet, 383
  - CMultiwellCallbackFunctionNet, 382
  - GetPlateClampStateByHeadstage, 383
  - GetPlateClampStateByHeadstageEvent, 383
  - OnGetPlateClampStateByHeadstage, 383
- CMultiwellDeviceNet, 384
  - ICMultiwellDeviceNet, 385
  - ~CMultiwellDeviceNet, 385
  - ClosePlateClamp, 386
  - CMultiwellDeviceNet, 385
  - GetPlateClampLockState, 386
  - GetPlateClampState, 386
  - GetPlateMux, 386, 387
  - GetPlateType, 387
  - GetPowerMuxPlate, 387
  - GetTouchPadEnable, 388
  - GetVolatileClampOffset, 388
  - IsPlateTypeValid, 388
  - LockPlateClamp, 389
  - OpenPlateClamp, 389
  - SetPlateMux, 389
  - SetPlateType, 390
  - SetPowerMuxPlate, 390
  - SetTouchPadEnable, 390
  - SetVolatileClampOffset, 391
  - StopPlateClamp, 391
  - UnlockPlateClamp, 391
- CMultiwellOptoStimFunctionNet, 391
  - ICMultiwellOptoStimFunctionNet, 392
  - ~CMultiwellOptoStimFunctionNet, 392
  - CMultiwellOptoStimFunctionNet, 392
  - GetAbsMaxCurrentInMicroAmp, 392
  - GetColorRgb, 393
  - GetColorStr, 393
  - GetMaxDurationHighCurrentInMicroSec, 393
  - GetMaxDutyCycleHighCurrent, 393
  - GetPermanentCurrentInMicroAmp, 394
  - GetWaveLengthInNanometer, 394
  - SetAbsMaxCurrentInMicroAmp, 394
  - SetColorRgb, 395
  - SetColorStr, 395
  - SetMaxDurationHighCurrentInMicroSec, 395
  - SetMaxDutyCycleHighCurrent, 395
  - SetPermanentCurrentInMicroAmp, 395
  - SetWaveLengthInNanometer, 396
- CNF\_GenDeviceNet, 396
  - ~CNF\_GenDeviceNet, 396
  - CNF\_GenDeviceNet, 396
  - Set\_Values, 397
- COctoPotDeviceNet, 397
  - BurnAdcOffset, 398
  - BurnDacOffset, 398
  - COctoPotDeviceNet, 398
  - EnableChecksum, 398
  - EnableDigitalIn, 398
  - EnableTimestamp, 398
  - GetAdcOffset, 398
  - GetDacOffset, 398
  - PatternListStart, 399
  - RampStart, 399
  - ResetAdcOffset, 399
  - ResetDacOffset, 399
  - SetAdcOffset, 399
  - SetAmplificationSwitch, 399
  - SetBathclamp, 399
  - SetChannelSwitch, 399
  - SetDacAutoControl, 399
  - SetDacOffset, 400
  - SetDacValue, 400
  - SetNumberOfChannels, 400
  - SetOutputRate, 400
  - SetPatternListEntry, 400
  - SetPidParameter, 400
  - SetRampParameter, 400
  - SetSineParameter, 400
  - SineStart, 401
- CokuvisionStimulatorDeviceNet, 401
  - ~CokuvisionStimulatorDeviceNet, 402
  - CokuvisionStimulatorDeviceNet, 402
  - GetCheckVoltage, 402
  - GetCurrentFactor, 402
  - GetDACOffset, 402
  - GetMaxPower, 402
  - GetMaxVoltage, 402
  - GetPulseform, 402
  - GetRTC, 403
  - GetStimulatorStatus, 403
  - GetVoltage, 403
  - SetCheckVoltage, 403
  - SetCurrentFactor, 403
  - SetDACOffset, 403
  - SetMaxPower, 404
  - SetMaxVoltage, 404
  - SetPulseform, 404

- SetRTC, 404
- Coldstart
  - CMcsUsbFactoryNet, 302
- CommaPositionA
  - CFilterCoefficientsNet::s\_FilterAttributesNet, 715
  - Mcs::Usb, 65
- CommaPositionB
  - CFilterCoefficientsNet::s\_FilterAttributesNet, 715
  - Mcs::Usb, 65
- CompareFirmware
  - CMcsUsbFactoryNet, 302
- CompareTo
  - HeadStageIDType, 700
- CompensateElectrodeOffset
  - CWarnerUssingFunctionNet, 638
- Connect
  - CMcsUsbNet, 321, 322
  - CRFFunctionNet, 449
- ConnectDevice
  - CRadioControlledDevicesNet, 444
- ConnectedImp
  - CPositionImpDeviceNet, 421
- ConnectImp
  - CPositionImpDeviceNet, 421
- ConnectSlave
  - CGilsonDeviceNet, 172
- ControlState
  - HeadStageIDTypeState, 703
- CornerFrequency
  - CFilterPropertyNet, 144
- CornerFrequencymHz
  - CFilterPropertyNet, 144
- Count
  - CMcsUsbListNet, 316
- CPatchServerDeviceNet, 404
  - CPatchServerDeviceNet, 405
  - Sensor, 405
- CPathIdentDeviceNet, 405
  - ~CPathIdentDeviceNet, 406
  - CPathIdentDeviceNet, 406
  - Measure, 406
  - Set\_Values, 406
- CPedoterDeviceNet, 406
  - !CPedoterDeviceNet, 407
  - ~CPedoterDeviceNet, 407
  - CPedoterDeviceNet, 407
  - GetCommand, 407
  - SetCommand, 407
- CPeristalticPumpDeviceNet, 408
  - ~CPeristalticPumpDeviceNet, 408
  - CPeristalticPumpDeviceNet, 408
  - McsBus\_MotorControl, 409
- CPgaDeviceNet, 409
  - ~CPgaDeviceNet, 409
  - ApplyGains, 410
  - CPgaDeviceNet, 409
  - DefineAmplification, 410
  - DefineFrequencyRange, 410
  - DefineNumAmplifications, 410
  - DefineNumFrequencyRanges, 410
  - GetAmplification, 410
  - GetFrequencyRange, 410
  - GetGain, 411
  - GetNumAmplifications, 411
  - GetNumFrequencyRanges, 411
  - SetGain, 411
- CPositionIIDeviceNet, 411
  - !CPositionIIDeviceNet, 413
  - ~CPositionIIDeviceNet, 413
  - CPositionIIDeviceNet, 413
  - GetCoilCommunication, 413
  - GetDebugData, 413
  - GetEventData, 414
  - GetImplantCurrentSetpoint, 414
  - GetImplantResult, 415
  - GetImplantState, 415
  - GetOnOff, 415
  - GetPowerStrength, 416
  - GetRTC, 416
  - GetStateDebugData, 416
  - GetStateEventData, 417
  - RFFunction, 420
  - SetImplantCurrentSetpoint, 417
  - SetPowerStrength, 417
  - SetRTC, 417
  - SetStateDebugData, 419
  - SetStateEventData, 419
  - SwitchOnOff, 419
- CPositionImpDeviceNet, 420
  - !CPositionImpDeviceNet, 421
  - ~CPositionImpDeviceNet, 421
  - ConnectedImp, 421
  - ConnectImp, 421
  - CPositionImpDeviceNet, 421
  - GetDeviceList, 421
  - GetImpld, 422
  - GetRFFrequency, 422
  - SetDeviceList, 422
  - SetImpld, 422
  - SetRFFrequency, 423
- CPPCDeviceNet, 423
  - CPPCDeviceNet, 423
  - McsBus, 424
  - McsBus\_MotorControl, 424
  - McsBus\_Sensor, 424
  - PPCFunction, 424
- CPPPCFunctionNet, 424
  - !CPPPCFunctionNet, 426
  - ~CPPPCFunctionNet, 426
  - CPPPCFunctionNet, 425
  - FirePressurePulse, 426
  - GetAnalogVoltage, 426
  - GetAnalogVoltageRange, 426
  - GetDigitalIn, 428
  - GetPressureRange, 428
  - GetPumpModeType, 428

- GetPumpSpeedUnit, [429](#)
- GetSupplyVoltage, [429](#)
- GetValveActive, [429](#)
- IsBusy, [429](#)
- LoadPressure, [431](#)
- MeasureReservoir, [431](#)
- SetAnalogVoltageRange, [431](#)
- SetPressureOffset, [431](#)
- SetPressureRange, [431](#)
- SetPumpModeType, [432](#)
- SetPumpSpeedUnit, [432](#)
- SetValveActive, [432](#)
- CPPS\_DeviceNet, [433](#)
  - CPPS\_DeviceNet, [433](#)
  - McsBus, [433](#)
  - McsBus\_MotorControl, [433](#)
  - McsBus\_Sensor, [433](#)
  - PPS\_Function, [433](#)
- CPPS\_FunctionNet, [434](#)
  - CPPS\_FunctionNet, [434](#), [435](#)
  - GetAnalogVoltage, [435](#)
  - GetAnalogVoltages, [435](#)
  - GetBubbleState, [435](#)
  - GetDigitalIn, [435](#)
  - GetPumpCouple, [435](#)
  - GetPumpEnableSpeedRatio, [435](#)
  - GetPumpFastOnOff, [435](#)
  - GetPumpFastSpeed, [435](#)
  - GetPumpFunctionSpeeds, [436](#)
  - GetPumpManualOnOff, [436](#)
  - GetPumpMaxSpeed, [436](#)
  - GetPumpModeType, [436](#)
  - GetPumpSpeedRatio, [436](#)
  - GetPumpSpeedUnit, [436](#)
  - GetSupplyVoltage, [436](#)
  - GetUseBubble, [436](#)
  - SetAnalogVoltages, [436](#)
  - SetPumpCouple, [437](#)
  - SetPumpEnableSpeedRatio, [437](#)
  - SetPumpFastOnOff, [437](#)
  - SetPumpFastSpeed, [437](#)
  - SetPumpFunctionSpeeds, [437](#)
  - SetPumpManualOnOff, [437](#)
  - SetPumpMaxSpeed, [437](#)
  - SetPumpModeType, [437](#)
  - SetPumpSpeedRatio, [438](#)
  - SetPumpSpeedUnit, [438](#)
  - SetUseBubble, [438](#)
- CPPSDeviceNet, [438](#)
  - CPPSDeviceNet, [439](#)
- CProgramPressureCurveNet, [439](#)
  - !CProgramPressureCurveNet, [440](#)
  - ~CProgramPressureCurveNet, [439](#)
  - CProgramPressureCurveNet, [439](#)
  - GetRepeats, [440](#)
  - Program, [440](#)
  - SetRepeats, [440](#)
- CPulseGeneratorFunctionNet, [440](#)
  - !CPulseGeneratorFunctionNet, [441](#)
  - ~CPulseGeneratorFunctionNet, [441](#)
  - CPulseGeneratorFunctionNet, [441](#)
  - GetModeSelect, [442](#)
  - GetPeriod, [442](#)
  - GetPulseLength, [442](#)
  - SetModeSelect, [443](#)
  - SetPeriod, [443](#)
  - SetPulseLength, [443](#)
- CRadioControlledDevicesNet, [443](#)
  - ConnectDevice, [444](#)
  - CRadioControlledDevicesNet, [444](#)
  - DisconnectDevice, [444](#)
  - GetDeviceNames, [444](#)
  - GetFrequency, [445](#)
  - HasRadioControl, [445](#)
  - SetFrequency, [445](#)
  - StillConnected, [445](#)
- CreateSideband
  - CStimulusFunctionNet, [580](#)
- CreateWirelessHeadstageSerialNumberString
  - CWirelessBaseFunctionNet, [685](#)
- CRegionOfInterestRect
  - CCMOSMeaDeviceNet::CRegionOfInterestRect, [445](#)
- CRetinaLedDeviceNet, [446](#)
  - ~CRetinaLedDeviceNet, [447](#)
  - AddLoopEntry, [447](#)
  - AddTableEntry, [447](#)
  - ClearTable, [447](#)
  - CRetinaLedDeviceNet, [447](#)
  - GetTablepointer, [447](#)
  - SetLED, [447](#)
  - SetLumi, [447](#)
  - SetPersistency, [447](#)
  - SetRepeat, [448](#)
  - SetTablepointer, [448](#)
  - SetTrigger, [448](#)
- CRFFFunctionNet, [448](#)
  - !CRFFFunctionNet, [449](#)
  - ~CRFFFunctionNet, [449](#)
  - Connect, [449](#)
  - CRFFFunctionNet, [449](#)
  - GetAvailableDeviceList, [450](#)
  - GetAvailableDeviceListEx, [450](#)
  - GetAvailableStateList, [450](#)
  - GetAvailableStateListEx, [450](#)
  - GetBaseFrequency, [451](#)
  - GetConnectedDevice, [451](#)
  - GetState, [451](#)
  - GetTestMode, [451](#)
  - GetWorkingFrequency, [452](#)
  - SetBaseFrequency, [452](#)
  - SetTestMode, [452](#)
  - SetWorkingFrequency, [452](#)
- CRobo\_FYIProgram\_FunctionNet, [453](#)
  - CRobo\_FYIProgram\_FunctionNet, [453](#)
  - GetLength, [453](#)



- GetState, [454](#)
- GetValve1, [454](#)
- GetValve2, [454](#)
- SetLength, [454](#)
- SetValve1, [454](#)
- SetValve2, [454](#)
- Start, [454](#)
- CRobo\_FYITemp\_FunctionNet, [454](#)
- CRobo\_FYITemp\_FunctionNet, [455](#)
- GetICoeff, [455](#)
- GetMaxPower, [455](#)
- GetPCoeff, [455](#)
- GetRegulatorOnOff, [455](#)
- GetSollTemp, [455](#)
- SetICoeff, [456](#)
- SetMaxPower, [456](#)
- SetPCoeff, [456](#)
- SetRegulatorOnOff, [456](#)
- SetSollTemp, [456](#)
- CRoboDacqNet, [456](#)
- CancelTableLoop, [459](#)
- CancelTableLoopAndStopTable, [459](#)
- ClampAmpRestart, [459](#)
- CRoboDacqNet, [459](#)
- DoRamp, [459](#)
- Emu\_GetCellCapacity, [460](#)
- Emu\_GetCellPotential, [460](#)
- Emu\_GetCellResists, [460](#)
- Emu\_GetElectrodeResists, [460](#)
- Emu\_GetNoise, [460](#)
- Emu\_SetCellCapacity, [460](#)
- Emu\_SetCellPotential, [460](#)
- Emu\_SetCellResists, [460](#)
- Emu\_SetElectrodeResists, [460](#)
- Emu\_SetNoise, [460](#)
- GetAllDigout, [461](#)
- GetCapacityC, [461](#)
- GetCapacityV, [461](#)
- GetCapacityX, [461](#)
- GetClampAmpSerialNumber, [461](#)
- GetCommand, [461](#)
- GetConfigurationBit, [461](#)
- GetConfigurationBitAxc, [461](#)
- GetConfigurationBitBlu\_Led, [461](#)
- GetConfigurationBitBlu\_LedToggleFast, [461](#)
- GetConfigurationBitBlu\_LedToggleSlow, [461](#)
- GetConfigurationBitCC\_Gen, [462](#)
- GetConfigurationBitCV\_Gen, [462](#)
- GetConfigurationBitRC\_Gen, [462](#)
- GetConfigurationBitRed\_Led, [462](#)
- GetConfigurationBitRed\_LedSaturation, [462](#)
- GetConfigurationBitRed\_LedToggleFast, [462](#)
- GetConfigurationBitRed\_LedToggleSlow, [462](#)
- GetConfigurationBitRelais, [462](#)
- GetConfigurationBitRV\_Gen, [462](#)
- GetConfigurationBits, [462](#)
- GetConfigurationBitStream, [462](#)
- GetConfigurationBitSupply, [463](#)
- GetCrossTalkOffset, [463](#)
- GetCrossTalkOptimum, [463](#)
- GetDigout, [463](#)
- GetDisplayText, [463](#)
- GetDownsampleFactor, [463](#)
- GetFilter, [463](#)
- GetFilterCoeffs, [463](#)
- GetIC, [463](#)
- GetIClamp, [463](#)
- GetICOffset, [463](#)
- GetIGain, [464](#)
- GetNIC\_MS, [464](#)
- GetNUC\_MS, [464](#)
- GetNUV\_MS, [464](#)
- GetPGain, [464](#)
- GetRecordingNumber, [464](#)
- GetResistanceC, [464](#)
- GetResistanceV, [464](#)
- GetScreen, [464](#)
- GetSimulation, [464](#)
- GetUC, [464](#)
- GetUClamp, [465](#)
- GetUCOffset, [465](#)
- GetUpdateDisplay, [465](#)
- GetUV, [465](#)
- GetUVOffset, [465](#)
- GetXGain, [465](#)
- RunTable, [465](#)
- SetAllDigout, [465](#)
- SetCommand, [465](#)
- SetConfigurationBit, [465](#)
- SetConfigurationBitAxc, [466](#)
- SetConfigurationBitBlu\_Led, [466](#)
- SetConfigurationBitBlu\_LedToggleFast, [466](#)
- SetConfigurationBitBlu\_LedToggleSlow, [466](#)
- SetConfigurationBitCC\_Gen, [466](#)
- SetConfigurationBitCV\_Gen, [466](#)
- SetConfigurationBitRC\_Gen, [466](#)
- SetConfigurationBitRed\_Led, [466](#)
- SetConfigurationBitRed\_LedSaturation, [466](#)
- SetConfigurationBitRed\_LedToggleFast, [467](#)
- SetConfigurationBitRed\_LedToggleSlow, [467](#)
- SetConfigurationBitRelais, [467](#)
- SetConfigurationBitRV\_Gen, [467](#)
- SetConfigurationBitStream, [467](#)
- SetConfigurationBitSupply, [467](#)
- SetCrossTalkOffset, [467](#)
- SetCrossTalkOptimum, [467](#)
- SetDigout, [467](#)
- SetDisplayText, [468](#)
- SetDownsampleFactor, [468](#)
- SetFilter, [468](#)
- SetFilterCoeffs, [468](#)
- SetIClamp, [468](#)
- SetICOffset, [468](#)
- SetIGain, [468](#)
- SetNoFilterCoeffs, [468](#)
- SetPGain, [468](#)

- SetRecordingNumber, [469](#)
- SetScreen, [469](#)
- SetSimulation, [469](#)
- SetTriggerMaskValue, [469](#)
- SetUClamp, [469](#)
- SetUCOffset, [469](#)
- SetUVOffset, [469](#)
- SetXGain, [469](#)
- StopTable, [469](#), [470](#)
- Table\_Wait, [470](#)
- TableDefBegin, [470](#)
- TableDefEnd, [470](#)
- TriggerMask\_Default, [470](#)
- TriggerValue\_MoveAbs, [470](#)
- TriggerValue\_StartQueue, [470](#)
- UpdateDisplay, [470](#)
- VirtualDevice\_ContinuousDacq, [470](#)
- VirtualDevice\_TableRun, [470](#)
- CRoboDeviceNet, [471](#)
  - ~CRoboDeviceNet, [474](#)
  - Axes\_I, [482](#)
  - Axes\_X, [482](#)
  - Axes\_Y, [482](#)
  - Axes\_Z, [483](#)
  - Axis\_I, [483](#)
  - Axis\_X, [483](#)
  - Axis\_Y, [483](#)
  - Axis\_Z, [483](#)
  - CancelPoolLoop, [474](#)
  - CancelPoolLoopAndStopMovement, [474](#)
  - CRoboDeviceNet, [474](#)
  - EnableQueue, [475](#)
  - FindReference, [475](#)
  - GetAirpressure, [475](#)
  - GetAirpressureLimit, [475](#)
  - GetAirValve, [475](#)
  - GetCurrentAirvalve, [475](#)
  - GetCurrentAirvalveLimit, [475](#)
  - GetCurrentPosition, [476](#)
  - GetErrorAirpressure, [476](#)
  - GetErrorCurrentAirvalve, [476](#)
  - GetErrorVoltage12V, [476](#)
  - GetErrorVoltage5V, [476](#)
  - GetErrorVoltageAirvalve, [476](#)
  - GetErrorVoltageRs485A, [476](#)
  - GetErrorVoltageRs485B, [476](#)
  - GetErrorVoltageValves, [477](#)
  - GetInMovement, [477](#)
  - GetMinPressure, [477](#)
  - GetMovementError, [477](#)
  - GetVoltage12V, [477](#)
  - GetVoltage12VLimit, [477](#)
  - GetVoltage5V, [477](#)
  - GetVoltage5VLimit, [477](#)
  - GetVoltageAirvalve, [477](#)
  - GetVoltageAirvalveLimit, [477](#)
  - GetVoltageRs485A, [478](#)
  - GetVoltageRs485ALimit, [478](#)
  - GetVoltageRs485B, [478](#)
  - GetVoltageRs485BLimit, [478](#)
  - GetVoltageValves, [478](#)
  - GetVoltageValvesLimit, [478](#)
  - IsQueueEnabled, [478](#)
  - IsQueueStarted, [478](#)
  - McsBus, [486](#)
  - McsBus\_MotorControl, [486](#)
  - McsBus\_XY, [483](#)
  - McsBus\_ZI, [483](#)
  - MoveAbs, [478](#), [479](#)
  - NullCommand, [480](#)
  - RoboError\_AnotherMaster, [483](#)
  - RoboError\_Base, [484](#)
  - RoboError\_CannotEscapeEndSwitch, [484](#)
  - RoboError\_CommandAlreadyInProgress, [484](#)
  - RoboError\_CommandNotPossible, [484](#)
  - RoboError\_CommunicationTimeout, [484](#)
  - RoboError\_DacqNotReady, [484](#)
  - RoboError\_DLLMovementTimeout, [484](#)
  - RoboError\_FindReferenceMethod, [484](#)
  - RoboError\_GilsonCommandPending, [484](#)
  - RoboError\_GilsonTimeout, [484](#)
  - RoboError\_GilsonWrondID, [485](#)
  - RoboError\_McsBus\_UnknownCommand, [485](#)
  - RoboError\_NoEndSwitch, [485](#)
  - RoboError\_NoMoreData, [485](#)
  - RoboError\_NoReference, [485](#)
  - RoboError\_NoSpeedOrAcceleration, [485](#)
  - RoboError\_OverPressure, [485](#)
  - RoboError\_ParameterNotAllowed, [485](#)
  - RoboError\_PeristalticTimeout, [485](#)
  - RoboError\_Phase0OutOfRange, [485](#)
  - RoboError\_PollLoopCanceled, [486](#)
  - RoboError\_PollLoopCanceledAndStopMovement, [486](#)
  - RoboError\_Pressure, [486](#)
  - RoboError\_RangeExceeded, [486](#)
  - RoboError\_StateChangeNotPossible, [486](#)
  - RoboError\_Timeout, [486](#)
  - RoboError\_UnknownCommand, [486](#)
  - RoboMainLowLevelCommand, [487](#)
  - RoboStatusEvent, [487](#)
  - SetAirpressureLimit, [480](#)
  - SetAirValve, [480](#)
  - SetCurrentAirvalveLimit, [480](#)
  - SetCurrentAndAir, [480](#)
  - SetInMovement, [480](#)
  - SetMinPressure, [481](#)
  - SetVoltage12VLimit, [481](#)
  - SetVoltage5VLimit, [481](#)
  - SetVoltageAirvalveLimit, [481](#)
  - SetVoltageRs485ALimit, [481](#)
  - SetVoltageRs485BLimit, [481](#)
  - SetVoltageValvesLimit, [481](#)
  - StartQueue, [481](#)
  - StopMovement, [482](#)
  - WaitTimer, [482](#)

- CRoboDeviceNet::RoboMainLowLevelCommands, 707
  - FindReferencePhase0, 708
  - GetAxisConfig, 708
  - GetHWConfig, 708
  - GetHWRevision, 708
  - GetMaxNoPressure, 708
  - GetMaxNoPressureWaitTime, 708
  - GetMaxPressureWaitTime, 708
  - GetMinNoPressureWaitTime, 708
  - GetMinPressure, 708
  - GetMinPressureWaitTime, 709
  - GetParameter, 709
  - GetPhases, 709
  - GetSearchReferenceFastAccel, 709
  - GetSearchReferenceFastSpeed, 709
  - GetSearchReferenceFineAccel, 709
  - GetSearchReferenceFineSpeed, 709
  - GetSearchReferenceMethod, 710
  - GetSearchReferenceMoveOut, 710
  - GetSearchReferenceOffsetPos, 710
  - GetUserParameter, 710
  - HasRef, 711
  - SetAxisConfig, 711
  - SetHWConfig, 711
  - SetHWRevision, 711
  - SetMaxNoPressure, 711
  - SetMaxNoPressureWaitTime, 711
  - SetMaxPressureWaitTime, 711
  - SetMinNoPressureWaitTime, 711
  - SetMinPressure, 711
  - SetMinPressureWaitTime, 711
  - SetParameter, 712
  - SetSearchReferenceFastAccel, 712
  - SetSearchReferenceFastSpeed, 712
  - SetSearchReferenceFineAccel, 712
  - SetSearchReferenceFineSpeed, 712
  - SetSearchReferenceMethod, 712
  - SetSearchReferenceMoveOut, 713
  - SetSearchReferenceOffsetPos, 713
  - SetUserParameter, 713
- CRoboFluidDeviceNet, 487
  - ~CRoboFluidDeviceNet, 488
  - CloseAllValves, 488
  - CRoboFluidDeviceNet, 488
  - GetPumpSpeed, 488
  - GetSingleValve, 488
  - GetValve, 488
  - IsPumpMotorOn, 489
  - m\_pMcsBus\_MotorControlNet, 490
  - m\_pRoboFluidDevice, 490
  - McsBus\_MotorControl, 490
  - PumpOff, 489
  - PumpOn, 489
  - SetPumpSpeed, 489
  - SetSingleValve, 489
  - SetValve, 489
- CRoboInjectDeviceNet, 490
  - CRoboInjectDeviceNet, 491
- CRoboocyte2DeviceNet, 491
  - CRoboocyte2DeviceNet, 492
  - GetAxisLED, 492
  - GetGilsonDevice, 492
  - GetMcsBus\_Extension, 492
  - GetRoboDacq, 492
  - GetRoboFluidDevice, 492
  - SetAxisLED, 492
- CRoboStatorDeviceNet, 492
  - CRoboStatorDeviceNet, 494
  - FindReferenceI, 494
  - FindReferenceXY, 494
  - FindReferenceZ, 494
  - GetCurrentPositionI, 494
  - GetCurrentPositionXY, 494
  - GetCurrentPositionZ, 495
  - HasRefl, 495
  - HasRefXY, 495
  - HasRefZ, 495
  - MoveAbsI, 495
  - MoveAbsXY, 495
  - MoveAbsZ, 495, 496
  - RoboMainStatorLowLevelCommand, 498
  - SetAccelerationI, 496
  - SetAccelerationNativeI, 496
  - SetAccelerationNativeXY, 496
  - SetAccelerationNativeZ, 496
  - SetAccelerationXY, 496
  - SetAccelerationZ, 496
  - SetCurrentAndAirXY, 496
  - SetSpeedI, 497
  - SetSpeedNativeI, 497
  - SetSpeedNativeXY, 497
  - SetSpeedNativeZ, 497
  - SetSpeedXY, 497
  - SetSpeedZ, 497
  - SetVelocityI, 497
  - SetVelocityXY, 497
  - SetVelocityZ, 497
  - StopMovementI, 498
  - StopMovementXY, 498
  - StopMovementZ, 498
- CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands, 714
  - FindReferencePhase0XY, 714
- CSafeISDeviceNet, 498
  - ~CSafeISDeviceNet, 499
  - CSafeISDeviceNet, 499
  - DacqDevice, 501
  - FluidControlDevice, 501
  - RoboDevice, 501
  - SetAdcChannels, 500
  - SetAdcSamplePos, 500
  - SetDacMode, 500
  - SetDacPeriode, 500
  - SetDacPulseform, 500
  - SetSwitches, 501
- csCapacityTestDischarge

- Mcs::Usb, 68
- csCapacityTestPrecharge
  - Mcs::Usb, 68
- csCharge
  - Mcs::Usb, 68
- CSCUDacqGroupChannelSelectionNet, 501
  - CSCUDacqGroupChannelSelectionNet, 502
- CSCUFunctionNet, 502
  - !CSCUFunctionNet, 505
  - ~CSCUFunctionNet, 505
  - AutomaticAnalogOut, 505
  - CSCUFunctionNet, 505
  - EnableAnalogOut, 505
  - GetAnalogOutADCRange, 506
  - GetAnalogOutChannels, 506
  - GetAnalogOutDACRange, 506
  - GetAvailableDockSmartMultiwellPlates, 506
  - GetAvailableHeadstages, 507
  - GetAvailableHeadstagesEvent, 520
  - GetDockSmartMultiwellPlatePowerstate, 507
  - GetDockSmartMultiwellPlateWellMux, 507
  - GetFilterProperties, 508
  - GetFilterProperty, 508
  - GetHeadstageAdcBits, 508
  - GetHeadstageAdcRangeInMicroVolt, 509
  - GetHeadstageDacBits, 509
  - GetHeadstageDacCurrentRangeInMicroAmpere, 509
  - GetHeadstageDacCurrentResolutionInNanoAmpere, 510
  - GetHeadstageDacVoltageRangeInMilliVolt, 510
  - GetHeadstageDacVoltageResolutionInMicroVolt, 510
  - GetHeadstageFrameCyclesToCompare, 511
  - GetHeadstageGainInPer mille, 511
  - GetHeadstageID, 511
  - GetHeadstageLinkSpeed, 512
  - GetHeadstageNumberOfAnalogChannels, 512
  - GetHeadstageNumberOfStimulationChannels, 512
  - GetHeadstagePowerStateAtStart, 512
  - GetHeadstageSamplerate, 513
  - GetHeadstageSerialNumber, 513
  - GetMaxNumberOfHeadstages, 513
  - GetMaxStimulusChannelsPerHeadstage, 514
  - GetReferenceElectrodeMode, 514
  - GetReferenceElectrodeSwitchState, 514
  - HasAnalogOut, 514
  - HasGalvanicIsolation, 515
  - HasHSPowerSwitch, 515
  - IsAnalogOutEnabled, 515
  - IsAutomaticAnalogOut, 515
  - IsHeadstageAvailable, 515
  - IsHeadstageAvailableEvent, 520
  - IsHSPowered, 516
  - IsInDacqLegacyMode, 516
  - OnGetAvailableHeadstages, 516
  - OnIsHeadstageAvailable, 516
  - PowerHS, 516
  - SetAnalogOutADCRange, 517
  - SetAnalogOutChannels, 517
  - SetAnalogOutDACRange, 517
  - SetDacqLegacyMode, 517
  - SetDockSmartMultiwellPlatePowerstate, 518
  - SetDockSmartMultiwellPlateWellMux, 518
  - SetHeadstageFrameCyclesToComparePermanent, 518
  - SetHeadstageLinkSpeedPermanent, 519
  - SetHeadstageNumberOfAnalogChannelsPermanent, 519
  - SetHeadstagePowerStateAtStart, 519
  - SetHeadstageSampleratePermanent, 519
  - SetReferenceElectrodeMode, 520
  - SetReferenceElectrodeSwitchState, 520
- csDischarge
  - Mcs::Usb, 69
- CSerialPortNet, 521
  - CSerialPortNet, 521
  - GetBytesAvailable, 521
  - Receive, 521
  - ReceiveString, 521, 522
  - Send, 522
- csError
  - Mcs::Usb, 69
- csIdleChargeFinished
  - Mcs::Usb, 68
- csIdleNoBattery
  - Mcs::Usb, 68
- csRefreshBattery
  - Mcs::Usb, 68
- CStg200xBasicNet, 522
  - ~CStg200xBasicNet, 527
  - GetAutocalibrationDisabled, 527
  - GetAvailableMemory, 528
  - GetBlankingEnable, 528
  - GetCanDisableCalibration, 529
  - GetCanDisableCurrentswitch, 529
  - GetCurrentRangeInNanoAmp, 529
  - GetCurrentRangeInNanoAmpByIndex, 530
  - GetCurrentRangeListInNanoAmp, 530
  - GetCurrentRangeSelectedIndex, 530
  - GetCurrentResolutionInNanoAmp, 530
  - GetCurrentResolutionInPicoAmp, 531
  - GetDacAmplificationFactor, 531
  - GetDACResolution, 531
  - GetDiginValue, 531
  - GetDigoutMode, 532
  - GetDigoutValue, 532
  - GetElectrodeDacMux, 532
  - GetElectrodeEnable, 533
  - GetElectrodeMode, 534
  - GetEnableAmplifierProtectionSwitch, 535
  - GetExternalElectrodeEnable, 536
  - GetFAAmplification, 536
  - GetHasLowCurrentswitchLatency, 536
  - GetHeadstage, 536
  - GetListModelIndexRange, 537

- GetListmodeTriggerSource, [537](#)
- GetNumberOfAnalogChannels, [537](#)
- GetNumberOfCurrentRangeIndexes, [538](#)
- GetNumberOfHWDACPaths, [538](#)
- GetNumberOfStimulationElectrodes, [538](#)
- GetNumberOfStimulationSourcesPerElectrode, [538](#)
- GetNumberOfSyncoutChannels, [538](#)
- GetNumberOfTriggerInputs, [538](#)
- GetNumberOfVoltageRangeIndexes, [539](#)
- GetOutputRate, [539](#)
- GetStgProgramInfo, [539](#)
- GetStgVersionInfo, [540](#)
- GetSyncoutMap, [540](#)
- GetTimeResolutionInNanoSeconds, [540](#)
- GetTotalMemory, [541](#)
- GetTriggerSource, [541](#)
- GetVoltageRangeInMicroVolt, [541](#)
- GetVoltageRangeInMicroVoltByIndex, [541](#)
- GetVoltageRangeListInMicroVolt, [542](#)
- GetVoltageRangeSelectedIndex, [542](#)
- GetVoltageResolutionInMicroVolt, [542](#)
- HasDigitalPort, [542](#)
- HasFilterAmp, [542](#)
- LegacyHasOutputMap, [543](#)
- ListModeSendStart, [543](#)
- ListModeSendStop, [543](#)
- SendStart, [543](#)
- SendStop, [544](#)
- SetAutocalibrationDisabled, [544](#)
- SetBlankingEnable, [544](#), [545](#)
- SetCurrentMode, [546](#)
- SetCurrentRangeSelectedIndex, [546](#)
- SetDacAmplificationFactor, [546](#)
- SetDigoutMode, [548](#)
- SetDigoutValue, [548](#)
- SetElectrodeDacMux, [548–550](#)
- SetElectrodeEnable, [550–552](#)
- SetElectrodeMode, [552](#), [554](#), [555](#)
- SetEnableAmplifierProtectionSwitch, [555](#), [556](#)
- SetExternalElectrodeEnable, [557](#)
- SetFAAmplification, [557](#)
- SetHeadstage, [558](#)
- SetListmodeIndexRange, [558](#)
- SetListmodeTriggerSource, [558](#), [559](#)
- SetMeasurementMode, [559](#)
- SetOutputRate, [559](#)
- SetStgProgramInfo, [560](#)
- SetSyncoutMap, [560](#)
- SetTriggerSource, [560](#)
- SetVoltageMode, [560](#), [561](#)
- SetVoltageRangeSelectedIndex, [561](#)
- CStg200xDownloadBasicNet, [561](#)
  - ClearChannelData, [563](#)
  - ClearSyncData, [563](#)
  - DisableAutoReset, [563](#)
  - EnableAutoReset, [563](#)
  - ForceStatusEvent, [564](#)
  - GetMemoryUsageDAC, [564](#)
  - GetMemoryUsageSyncout, [564](#)
  - GetSweepCount, [564](#)
  - GetTrigger, [565](#)
  - ResetStatus, [565](#)
  - SendChannelData, [566](#)
  - SendSyncData, [566](#)
  - SetupRetriggerMode, [566](#), [567](#)
  - SetupTrigger, [567](#)
  - SetupTriggerSingle, [569](#)
  - Stimulus, [570](#)
- CStg200xDownloadNet, [570](#)
  - ~CStg200xDownloadNet, [571](#)
  - AppendPreparedData, [571](#)
  - ClearChannel\_PrepAndSendData, [572](#)
  - CStg200xDownloadNet, [571](#)
  - DisableMultiFileMode, [572](#)
  - EnableMultiFileMode, [573](#)
  - GetModuleCurrent, [573](#)
  - GetModuleTemp, [573](#)
  - MwPollStatusEvent, [576](#)
  - PrepareAndAppendData, [573](#)
  - PrepareAndSendData, [574](#)
  - QueryTriggerstatus, [575](#)
  - SegmentDefine, [575](#)
  - SegmentSelect, [575](#)
  - SegmentStart, [576](#)
  - SendPreparedData, [576](#)
  - SetOutputMap, [576](#)
  - Stg200xPollStatusEvent, [576](#)
- CStimulusFunctionNet, [577](#)
  - ClearChannel\_PrepAndSendData, [579](#)
  - ClearChannelData, [579](#)
  - ClearMultiplexedData, [579](#)
  - ClearSyncData, [579](#)
  - CreateSideband, [580](#)
  - CStimulusFunctionNet, [578](#), [579](#)
  - ForceStatusEvent, [580](#)
  - GetAvailableMemory, [581](#)
  - GetCurrentRangeInNanoAmp, [581](#)
  - GetCurrentResolutionInNanoAmp, [581](#)
  - GetCurrentResolutionInPicoAmp, [581](#)
  - GetDACResolution, [583](#)
  - GetMultiplexedDataChannelsInBlock, [583](#)
  - GetNumberOfAnalogChannels, [583](#)
  - GetTotalMemory, [583](#)
  - GetVoltageRangeInMicroVolt, [583](#)
  - GetVoltageResolutionInMicroVolt, [584](#)
  - PollStatusEvent, [589](#)
  - PrepareAndAppendData, [584](#)
  - PrepareAndSendData, [585](#)
  - PrepareData, [586](#)
  - SendMultiplexedData, [586](#)
  - SendPreparedData, [586](#)
  - SendStart, [586](#)
  - SendStop, [587](#)
  - SetupTrigger, [587](#)
  - SetupTriggerSingle, [588](#)

- StartPoll, 588
- StopPoll, 588
- CStimulusFunctionNet::SidebandData, 716
  - !SidebandData, 716
  - ~SidebandData, 716
  - Duration, 716
  - Sideband, 716
  - SidebandData, 716
- CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, 717
  - !StimulusDeviceDataAndUnrolledData, 718
  - ~StimulusDeviceDataAndUnrolledData, 718
  - DeviceData, 718
  - DeviceDataLength, 718
  - StimulusDeviceDataAndUnrolledData, 718
  - UnrolledAmplitude, 718
  - UnrolledDuration, 718
  - UnrolledSync, 718
- CSw2to64DeviceNet, 589
  - ~CSw2to64DeviceNet, 590
  - CSw2to64DeviceNet, 590
  - GetChannel, 590
  - GetChannels, 590
  - GetNumber, 590
  - SetChannel, 590
  - SetChannels, 591
- CTcxDeviceNet, 591
  - ~CTcxDeviceNet, 594
  - CalibrateThermocouple, 594
  - CTcxDeviceNet, 593
  - FactoryReset, 594
  - GetBoardTemp, 594
  - GetCalibration, 594
  - GetCalibrationDecp, 594
  - GetCalibrationMax, 594
  - GetCalibrationMin, 594
  - GetCurrent, 594
  - GetD, 595
  - GetDDecp, 595
  - GetDevice, 595
  - GetDeviceType, 595
  - GetDevname, 595
  - GetDMax, 595
  - GetDMin, 595
  - GetDuty, 595
  - GetEnableHeaterLimit, 596
  - GetEnableThermocouple, 596
  - GetHasThermocouple, 596
  - GetHeaterLimit, 596
  - GetHeaterTemp, 596
  - GetI, 596
  - GetIDecp, 596
  - GetIMax, 597
  - GetIMin, 597
  - GetIOut, 597
  - GetMaxHeaterPowerMultiwell, 597
  - GetMaxP, 597
  - GetMaxpDecp, 597
  - GetMaxpMax, 597
  - GetMaxpMin, 598
  - GetNumControlChannels, 598
  - GetNumDevices, 598
  - GetNumMeasureChannels, 598
  - GetOnOff, 598
  - GetP, 598
  - GetPDecp, 598
  - GetPMax, 598
  - GetPMin, 598
  - GetPOut, 598
  - GetPwrOut, 599
  - GetPwrSet, 599
  - GetRes1, 599
  - GetRes2, 599
  - GetResS, 599
  - GetResX, 599
  - GetROut, 599
  - GetSensorType, 600
  - GetSetpoint, 600
  - GetSetpointDecp, 600
  - GetSetpointMax, 600
  - GetSetpointMin, 600
  - GetThermocoupleCalibration, 600
  - GetThermocoupleNanovoltPerKelvin, 600
  - GetThermocoupleReferenceTemp, 601
  - GetThermocoupleTemp, 601
  - GetThermocoupleTempAbs, 601
  - GetUnit, 601
  - GetUOut, 601
  - GetValue, 601
  - GetValueHires, 602
  - GetVolti, 602
  - SetCalibration, 602
  - SetD, 602
  - SetDevice, 602
  - SetDeviceType, 602
  - SetDevname, 602
  - SetEnableHeaterLimit, 602
  - SetEnableThermocouple, 603
  - SetHeaterLimit, 603
  - SetI, 603
  - SetMaxHeaterPowerMultiwell, 603
  - SetMaxP, 603
  - SetOnOff, 603
  - SetP, 604
  - SetSensorType, 604
  - SetSetpoint, 604
  - SetThermocoupleNanovoltPerKelvin, 604
- CTEERFunctionNet, 604
  - !CTEERFunctionNet, 607
  - ~CTEERFunctionNet, 607
  - CancelInternalCalibration, 607
  - CTEERFunctionNet, 606
  - GetAdapterCode, 607
  - GetAdcOffsetU1, 607
  - GetAdcOffsetU2, 607
  - GetAmplitude\_nA, 607



- GetBytesPerSample, [608](#)
- GetClampMode, [608](#)
- GetControllerParams, [608](#)
- GetCurrentEnable, [608](#)
- GetDacZero, [609](#)
- GetFrameErrorCounter, [609](#)
- GetLiquidResistance, [609](#)
- GetMaxChunkSize\_Byte, [609](#)
- GetNumberOfAvailableSamples, [609](#)
- GetPeriod\_us, [610](#)
- GetRotaryPositionCode, [610](#)
- GetSampleBufferChunk, [610](#)
- GetSampleRate, [610](#)
- GetSampleVoltageBuffer\_uV, [611](#)
- GetScaleFactorU1, [611](#)
- GetScaleFactorU2, [611](#)
- GetUptimeSeconds, [611](#)
- GetWaveform, [611](#)
- IsInternalCalibrationFinished, [612](#)
- IsSamplingFinished, [612](#)
- SetAmplitude\_nA, [612](#)
- SetBufferIndex, [612](#)
- SetClampMode, [613](#)
- SetControllerParams, [613](#)
- SetCurrentEnable, [613](#)
- SetExternalLED, [613](#)
- SetLiquidResistance, [614](#)
- SetPeriod\_us, [614](#)
- SetWaveform, [614](#)
- StartInternalCalibration, [614](#)
- StartSampling, [614](#)
- StopSampling, [615](#)
- CTEERMachineDeviceNet, [615](#)
  - ~CTEERMachineDeviceNet, [615](#)
- CTEERMachineDeviceNet, [615](#)
- TEERFunctionNet, [616](#)
- CurrentClamp
  - Mcs::Usb, [89](#)
- CurrentMeasure
  - Mcs::Usb, [58](#)
- CurrentRangeInNanoAmp
  - W2100\_StimulusParametersNet, [720](#)
- CurrentResolutionInNanoAmp
  - W2100\_StimulusParametersNet, [720](#)
- CUsbDeviceConfigurationFunctionNet, [616](#)
  - !CUsbDeviceConfigurationFunctionNet, [617](#)
  - ~CUsbDeviceConfigurationFunctionNet, [617](#)
  - CUsbDeviceConfigurationFunctionNet, [616](#)
  - SetDeviceId, [617](#)
  - SetDeviceName, [617](#)
- CUsbExceptionNet, [617](#)
  - CUsbExceptionNet, [618](#)
  - Status, [618](#)
- CutoffOrCenterFrequency
  - CCreateFilterNet, [119](#)
- CVoltageRangeInfoNet
  - CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet, [619](#)
- CW2100\_FunctionNet, [619](#)
  - ClearStimulusParametersCache, [621](#)
  - ClearUserDefinedNameCache, [621](#)
  - CW2100\_FunctionNet, [621](#)
  - DeselectAllHeadstages, [622](#)
  - DeselectHeadstage, [622](#)
  - GetAccelGyroCurrentRate, [622](#)
  - GetAccelGyroDesiredRate, [622](#)
  - GetAccelGyroEnabled, [622](#)
  - GetAccelRange, [622](#)
  - GetAnalogOutChannel, [622](#)
  - GetAnalogOutFilter, [622](#)
  - GetAudioChannels, [622](#)
  - GetAvailableHeadstages, [622](#)
  - GetBatteryState, [623](#)
  - GetDacRange, [623](#)
  - GetFilterProperties, [623](#)
  - GetFilterProperty, [623](#)
  - GetFPGA FirmwareType, [623](#)
  - GetGyroRange, [623](#)
  - GetHeadstageOnOff, [623](#)
  - GetHeadstageSamplingActive, [623](#)
  - GetMultiHeadstageMode, [623](#)
  - GetPicFirmwareType, [623](#)
  - GetSelectedChannels, [624](#)
  - GetSelectedHeadstageState, [624](#)
  - GetStimulusParametersCache, [624](#)
  - GetStimulusParametersFromSelectedHS, [624](#)
  - GetStiumlusParameters, [624](#)
  - GetUserDefinedName, [624](#)
  - GetUserDefinedNameCache, [624](#)
  - GetUserDefinedNameFromSelectedHS, [625](#)
  - PulseGenerator, [626](#)
  - SelectHeadstage, [625](#)
  - SetAccelGyroDesiredRate, [625](#)
  - SetAccelGyroEnabled, [625](#)
  - SetAccelRange, [625](#)
  - SetAnalogOutChannel, [625](#)
  - SetAnalogOutFilter, [625](#)
  - SetAudioChannels, [625](#)
  - SetDacRange, [626](#)
  - SetGyroRange, [626](#)
  - SetHeadstageOnOff, [626](#)
  - SetHeadstageSamplingActive, [626](#)
  - SetHeadstageToSleep, [626](#)
  - SetMultiHeadstageMode, [626](#)
  - SetSelectedChannels, [626](#)
  - Stimulator, [627](#)
- CW2100\_FunctionNet::AudioChannelsNet, [95](#)
  - amplification, [95](#)
  - channel, [95](#)
  - dacqgroup, [95](#)
- CW2100\_StimulatorFunctionNet, [627](#)
  - BOOST\_BIT, [633](#)
  - ClearChannelData, [628](#)
  - CW2100\_StimulatorFunctionNet, [628](#)
  - GetBoostAlwaysOnMode, [629](#)
  - GetBoostPreTime, [629](#)

- GetCurrentRangeInNanoAmp, 629
- GetCurrentResolutionInNanoAmp, 629
- GetCurrentResolutionInPicoAmp, 629
- GetDACResolution, 630
- GetDigitalStimulatorTrigger, 630
- GetDigitalStimulatorTriggerSlope, 630
- GetNumberOfAnalogChannels, 630
- GetNumberOfSyncoutChannels, 630
- GetNumberOfTriggerInputs, 630
- GetStimulationPatternMemory, 630
- GetTimeResolutionInNanoSeconds, 631
- GetTimeSlot, 631
- GetVoltageRangeInMicroVolt, 631
- GetVoltageResolutionInMicroVolt, 631
- GND\_SWITCH\_BIT, 633
- PollStatusEvent, 634
- PrepareData, 631
- PrepareDataSync, 632
- SelectTimeSlot, 632
- SendPreparedData, 632
- SendStart, 632
- SendStop, 632
- SetDigitalStimulatorTrigger, 633
- SetDigitalStimulatorTriggerSlope, 633
- StartPoll, 633
- StopPoll, 633
- SYNC\_BIT0, 633
- SYNC\_BIT1, 633
- CW2100DacqGroupChannelSelectionNet, 634
  - CW2100DacqGroupChannelSelectionNet, 634
- CWarnerUssingDeviceNet, 634
  - ICWarnerUssingDeviceNet, 635
  - ~CWarnerUssingDeviceNet, 635
  - CWarnerUssingDeviceNet, 635
  - WarnerUssingFunction, 635
- CWarnerUssingFunctionNet, 636
  - ICWarnerUssingFunctionNet, 638
  - ~CWarnerUssingFunctionNet, 638
  - CompensateElectrodeOffset, 638
  - CWarnerUssingFunctionNet, 638
  - GetAvailableChambers, 639
  - GetChannelsCountOfChamber, 639
  - GetClampMode, 639
  - GetComplianceVoltageThreshold, 639
  - GetDacPampsPerDigitHighCurrentRange, 640
  - GetDacPampsPerDigitLowCurrentRange, 640
  - GetDacZero, 640
  - GetHighCurrentRange, 641
  - GetIdleModeOffset, 641
  - GetLiquidResistance, 641
  - GetLowCurrentRange, 642
  - GetNumberOfAvailableChambers, 642
  - GetNumberOfHardwareSlotsForChambers, 642
  - GetU1Offset, 642
  - GetU1Reference, 643
  - GetU2Offset, 643
  - GetU2Reference, 643
  - GetUnitDescription, 644
  - GetUnitExponent, 644
  - GetUnitName, 644
  - GetUnitsPerDigit, 645
  - GetUptimeSeconds, 645
  - GetVoltageClampControllerParam\_D, 645
  - GetVoltageClampControllerParam\_I, 646
  - GetVoltageClampControllerParam\_P, 646
  - IsChamberAvailable, 646
  - IsHighCurrentMode, 647
  - IsInternalCalibrationFinished, 647
  - IsPulseEnabled, 647
  - SetClampMode, 648
  - SetEnablePulse, 648
  - SetHighCurrentMode, 648
  - SetIdleModeOffset, 649
  - SetLiquidResistance, 649
  - SetLowCurrentMode, 649
  - SetPulse, 649
  - SetVoltageClampControllerParam\_D, 650
  - SetVoltageClampControllerParam\_I, 650
  - SetVoltageClampControllerParam\_P, 650
  - WaitForAllChambers, 651
  - WaitForChamber, 651
  - WasInternalCalibrationSuccessful, 651
- CWarnerValveControllerDeviceNet, 652
  - ICWarnerValveControllerDeviceNet, 656
  - ~CWarnerValveControllerDeviceNet, 656
  - ClearTableName, 656
  - ClearValveTable, 656
  - CWarnerValveControllerDeviceNet, 656
  - GetActiveRunningTableNumber, 657
  - GetActiveRunningTableNumberEvent, 673
  - GetAnalogThresholdHigh, 657
  - GetAnalogThresholdHighEvent, 673
  - GetAnalogThresholdLow, 657
  - GetAnalogThresholdLowEvent, 673
  - GetAnalogVoltage, 657
  - GetAnalogVoltageEvent, 674
  - GetCurrentEditTableNumber, 658
  - GetCurrentNumberOfValves, 658
  - GetCurrentNumberOfValvesEvent, 674
  - GetDigitalOutPortValve, 658
  - GetDigitalOutPortValveEvent, 674
  - GetDigitalPortDirection, 658
  - GetDigitalPortDirectionEvent, 674
  - GetDisplayMode, 659
  - GetDisplayModeEvent, 674
  - GetTableName, 659
  - GetTableNamebyIndex, 659
  - GetTableNamebyIndexEvent, 674
  - GetTotalNumberOfDigitalPorts, 659
  - GetTotalNumberOfTables, 660
  - GetTotalNumberOfValves, 660
  - GetTotalTableSize, 660
  - GetValveActive, 660
  - GetValveActiveEvent, 674
  - GetValveBoardRevision, 660
  - GetValveBoardRevisionEvent, 675



- GetValveBoardRevisionString, 661
- GetValveCurrent, 661
- GetValveDigitalInPort, 661
- GetValveDigitalInPortEvent, 675
- GetValveLedOn, 661
- GetValveLedOnEvent, 675
- GetValveManualGroup, 662
- GetValveManualGroupEvent, 675
- GetValveManualState, 662
- GetValveManualStateEvent, 675
- GetValveMode, 662
- GetValveModeEvent, 675
- GetValvesActiveMap, 663
- GetValvesManualStateMap, 663
- GetValveTableEntry, 663
- IsDigitalOutPortInverted, 663
- IsDigitalOutPortInvertedEvent, 675
- IsValveDigitalInInverted, 664
- IsValveDigitalInInvertedEvent, 675
- IsValveOpen, 664
- IsValveOpenEvent, 676
- IsValveOpenInAnalogMode, 664
- IsValveOpenInAnalogModeEvent, 676
- IsValveOpenInDigitalMode, 665
- IsValveOpenInDigitalModeEvent, 676
- LoadValveTable, 665
- OnGetActiveRunningTableNumber, 665
- OnGetAnalogThresholdHigh, 665
- OnGetAnalogThresholdLow, 665
- OnGetAnalogVoltage, 665
- OnGetCurrentNumberOfValves, 665
- OnGetDigitalOutPortValve, 666
- OnGetDigitalPortDirection, 666
- OnGetDisplayMode, 666
- OnGetTableNamebyIndex, 666
- OnGetValveActive, 666
- OnGetValveBoardRevision, 666
- OnGetValveDigitalInPort, 666
- OnGetValveLedOn, 666
- OnGetValveManualGroup, 666
- OnGetValveManualState, 667
- OnGetValveMode, 667
- OnIsDigitalOutPortInverted, 667
- OnIsValveDigitalInInverted, 667
- OnIsValveOpen, 667
- OnIsValveOpenInAnalogMode, 667
- OnIsValveOpenInDigitalMode, 667
- OnTableEntryChanged, 667
- SetActiveRunningTableNumber, 667
- SetAnalogThresholdHigh, 668
- SetAnalogThresholdLow, 668
- SetCurrentEditTableNumber, 668
- SetDefault, 668
- SetDigitalOutPortInvert, 668
- SetDigitalOutPortValve, 669
- SetDigitalPortDirection, 669
- SetDisplayMode, 669
- SetTableName, 670
- SetTableStep, 670
- SetTableStepAll, 670
- SetValveActive, 670
- SetValveCurrent, 671
- SetValveDigitalInInvert, 671
- SetValveDigitalInPort, 671
- SetValveLedOn, 671
- SetValveManualGroup, 672
- SetValveManualState, 672
- SetValveMode, 672
- SetValvesActiveMap, 672
- SetValvesManualStateMap, 673
- SetValveTableEntry, 673
- StoreValveTable, 673
- TableEntryChangedEvent, 676
- CWarnerValveControllerDeviceTesterFunctionNet, 676
  - !CWarnerValveControllerDeviceTesterFunctionNet, 678
  - ~CWarnerValveControllerDeviceTesterFunctionNet, 677
- CWarnerValveControllerDeviceTesterFunctionNet, 677
- GetIO, 678
- GetSync, 678
- SetADC, 678
- SetIO, 678
- SetIODirection, 679
- SetTrigger, 679
- SetTriggerSyncDirection, 679
- CWClassicFunctionNet, 679
  - CWClassicFunctionNet, 680, 681
  - GetFilterParametersHeadstage, 681
  - GetHasChecksum, 681
  - GetHasRedLedHeadstage, 681
  - GetHeadstageOnOff, 681
  - GetResetFilter, 681
  - GetRFConnectionStatus, 681
  - GetRFFrequencyHeadstage, 681
  - GetRFFrequencyReceiver, 681
  - GetRFPower, 682
  - GetScanHeadstagesResult, 682
  - GetSelectedHeadstage, 682
  - GetSerialNumberHeadstage, 682
  - GetWPADebugMode, 682
  - GetWPAType, 682
  - ResetChannelmap, 682
  - ScanForHeadstages, 682
  - SetChannelmap, 682
  - SetFilterParametersHeadstage, 682
  - SetHasChecksum, 683
  - SetHeadstageOnOff, 683
  - SetHWSelectedChannels, 683
  - SetResetFilter, 683
  - SetRFFrequencyHeadstage, 683
  - SetRFFrequencyReceiver, 683
  - SetRFFrequencyReceiverEeprom, 683
  - SetRFLostBehaviour, 683
  - SetRFPower, 684

- SetSelectedHeadstage, [684](#)
- SetSerialNumberHeadstage, [684](#)
- SetWPADebugMode, [684](#)
- SetWPAType, [684](#)
- CWirelessBaseFunctionNet, [684](#)
  - CreateWirelessHeadstageSerialNumberString, [685](#)
  - CWirelessBaseFunctionNet, [685](#)
- CyclePort
  - CMcsUsbNet, [323](#)
- Cypress
  - Mcs::Usb, [90](#)
- Cypress\_FX1
  - Mcs::Usb, [76](#)
- Cypress\_FX2
  - Mcs::Usb, [76](#)
- Cypress\_FX3
  - Mcs::Usb, [76](#)
- DAC1Channel
  - Mcs::Usb, [60](#)
- DAC2Channel
  - Mcs::Usb, [60](#)
- DAC3Channel
  - Mcs::Usb, [60](#)
- DAC4Channel
  - Mcs::Usb, [60](#)
- DACQ1DigitalGroup
  - Mcs::Usb, [61](#)
- DacqDevice
  - CSafeISDeviceNet, [501](#)
- dacqgroup
  - CW2100\_FunctionNet::AudioChannelsNet, [95](#)
- DacqGroupChannelEnumNet
  - Mcs::Usb, [61](#)
- DacqMeaGroupTypeEnumNet
  - Mcs::Usb, [61](#)
- DacqTrigger
  - Mcs::Usb, [64](#)
- DACResolution
  - W2100\_StimulusParametersNet, [720](#)
- DataModeEnumNet
  - Mcs::Usb, [61](#)
- DataState
  - HeadStageIDTypeState, [703](#)
- DeepCopy
  - CCMOSMeaDeviceNet::CRegionOfInterestRect, [445](#)
- DefineAmplification
  - CPgaDeviceNet, [410](#)
- DefineFrequencyRange
  - CPgaDeviceNet, [410](#)
- DefineNumAmplifications
  - CPgaDeviceNet, [410](#)
- DefineNumFrequencyRanges
  - CPgaDeviceNet, [410](#)
- DeselectAllHeadstages
  - CW2100\_FunctionNet, [622](#)
- DeselectHeadstage
  - CW2100\_FunctionNet, [622](#)
- DEST\_FX3\_TARGET\_MASK
  - Mcs::Usb, [56](#)
- DEST\_TARGET1
  - Mcs::Usb, [56](#)
- DEST\_TARGET10
  - Mcs::Usb, [56](#)
- DEST\_TARGET11
  - Mcs::Usb, [56](#)
- DEST\_TARGET12
  - Mcs::Usb, [56](#)
- DEST\_TARGET13
  - Mcs::Usb, [56](#)
- DEST\_TARGET14
  - Mcs::Usb, [56](#)
- DEST\_TARGET15
  - Mcs::Usb, [56](#)
- DEST\_TARGET2
  - Mcs::Usb, [56](#)
- DEST\_TARGET3
  - Mcs::Usb, [56](#)
- DEST\_TARGET4
  - Mcs::Usb, [56](#)
- DEST\_TARGET5
  - Mcs::Usb, [56](#)
- DEST\_TARGET6
  - Mcs::Usb, [56](#)
- DEST\_TARGET7
  - Mcs::Usb, [56](#)
- DEST\_TARGET8
  - Mcs::Usb, [56](#)
- DEST\_TARGET9
  - Mcs::Usb, [56](#)
- DEST\_TARGET\_MASK
  - Mcs::Usb, [56](#)
- DetectChipType
  - CCMOSMea\_FunctionNet, [106](#)
- DEVICE\_NOT\_FOUND
  - Mcs::Usb, [65](#)
- DeviceArrival
  - CMcsUsbListNet, [316](#)
- DeviceData
  - CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, [718](#)
- DeviceDataLength
  - CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, [718](#)
- DeviceEnumNet
  - Mcs::Usb, [62](#)
- DeviceHasNoHeadstage
  - Mcs::Usb, [67](#), [68](#), [81](#)
- DeviceId
  - CMcsUsbListEntryNet, [313](#)
- DeviceIdNet, [685](#)
  - BcdDevice, [686](#)
  - BusType, [686](#)
  - DeviceIdNet, [686](#)
  - IdProduct, [686](#)

- IdVendor, [686](#)
- operator=, [686](#)
- DeviceName
  - CMcsUsbListEntryNet, [313](#)
- DeviceNotConnected
  - Mcs::Usb, [66](#), [68](#), [81](#)
- DeviceRemoval
  - CMcsUsbListNet, [316](#)
- DeviceRunStatus
  - Mcs::Usb, [57](#), [70](#), [73](#), [74](#), [82](#), [85](#), [92](#)
- DigDataFromReceiver
  - Mcs::Usb, [92](#)
- Digital
  - Mcs::Usb, [76](#), [93](#)
- DigitalData
  - Mcs::Usb, [57](#), [71](#), [73](#), [74](#), [82](#), [85](#), [92](#)
- DigitalDatastreamEnableEnumNet
  - Mcs::Usb, [63](#)
- DigitalGroup
  - Mcs::Usb, [61](#)
- DigitalIn
  - Mcs::Usb, [57](#), [63](#), [70](#), [72](#), [73](#), [82](#), [85](#), [92](#)
- DigitalInOfOutPort
  - Mcs::Usb, [57](#), [70](#), [72](#), [73](#), [82](#), [85](#), [92](#)
- DigitalInPort
  - Mcs::Usb, [59](#)
- DigitalInReservd
  - Mcs::Usb, [63](#)
- DigitalMux
  - Mcs::Usb, [59](#)
- DigitalOut
  - Mcs::Usb, [63](#)
- DigitalOutReg
  - Mcs::Usb, [59](#)
- DigitalOutReserved
  - Mcs::Usb, [63](#)
- DigitalOutStimulator
  - Mcs::Usb, [57](#), [71](#), [73](#), [74](#), [82](#), [85](#), [92](#)
- DigitalPulse
  - Mcs::Usb, [57](#), [70](#), [72](#), [73](#), [82](#), [85](#), [92](#)
- DigitalReg
  - Mcs::Usb, [59](#)
- DigitalSource
  - DigitalSource< digitalsourceenum >, [687](#)
- DigitalSource< digitalsourceenum >, [687](#)
  - DigitalSource, [687](#)
  - MaxBitNumber, [687](#)
  - MaxBitNumberStatic, [687](#)
  - size, [687](#)
  - Source, [688](#)
- DigitalSourceGeneral, [688](#)
  - DigitalSourceGeneral, [688](#)
  - MaxBitNumber, [688](#), [689](#)
  - size, [689](#)
  - Source, [689](#)
- DigitalStimulatorTriggerEventEnumNet
  - Mcs::Usb, [63](#)
- DigitalStimulatorTriggerSlopeEnumNet
  - Mcs::Usb, [64](#)
- DigitalTargetEnumNet
  - Mcs::Usb, [64](#)
- Digout
  - Mcs::Usb, [64](#)
- DigOutStim
  - Mcs::Usb, [63](#)
- DigOutStimulatorStartTrigger
  - Mcs::Usb, [64](#)
- DigOutStimulatorStopTrigger
  - Mcs::Usb, [64](#)
- Digstream
  - Mcs::Usb, [64](#)
- DigStreamFromReceiver
  - Mcs::Usb, [92](#)
- DigStreamToReceiver
  - Mcs::Usb, [64](#)
- Dilutor
  - Mcs::Usb, [78](#)
- DisableAutoReset
  - CStg200xDownloadBasicNet, [563](#)
- DisableMultiFileMode
  - CStg200xDownloadNet, [572](#)
- Disconnect
  - CMcsUsbNet, [323](#)
- DisConnectDevice
  - CRadioControlledDevicesNet, [444](#)
- DongleS
  - Mcs::Usb, [77](#)
- DoRamp
  - CRoboDacqNet, [459](#)
- Dotriapot
  - Mcs::Usb, [76](#)
- DoubleToInt
  - Mcs::Usb, [66](#)
- DownloadFirmware
  - CMcsUsbFactoryNet, [302](#)
- DownloadOnly
  - Mcs::Usb, [84](#)
- DriverVersionNet, [689](#)
  - ~DriverVersionNet, [690](#)
  - DriverVersionNet, [690](#)
  - DriverVersionNet::FormatVersion, [690](#)
  - GetDestinationCode, [691](#)
  - GetDestinationName, [692](#)
  - GetMajor, [692](#)
  - GetMinor, [693](#)
  - GetNumEntries, [693](#)
  - GetSerialNumber, [693](#)
  - GetStatus, [694](#)
  - GetVersionInt, [694](#)
  - GetVersionString, [695](#)
- DriverVersionNet::FormatVersion
  - DriverVersionNet, [690](#)
- DSP
  - FirmwareDestinationNames, [696](#)
  - Mcs::Usb, [53](#)
- DSPAnalogGroup

- Mcs::Usb, [70](#), [81](#)
- DSPDataGroup
  - Mcs::Usb, [61](#), [91](#)
- DSPDigitalGroup
  - Mcs::Usb, [70](#), [81](#)
- DummyCommand
  - CLIH3DeviceNet, [225](#)
- Duration
  - CStimulusFunctionNet::SidebandData, [716](#)
- eCube
  - Mcs::Usb, [78](#), [81](#)
- eCubeHeadstage
  - Mcs::Usb, [67](#)
- ElectricalStimulation
  - HeadStageIDType, [699](#)
- ElectrodeDacMuxEnumNet
  - Mcs::Usb, [64](#)
- ElectrodeModeEnumNet
  - Mcs::Usb, [65](#)
- ElectrodeOffset
  - Mcs::Usb, [89](#)
- emAutomatic
  - Mcs::Usb, [65](#)
- emManual
  - Mcs::Usb, [65](#)
- EmptyKey
  - CMcsUsbNet, [323](#)
- Emu\_GetCellCapacity
  - CRoboDacqNet, [460](#)
- Emu\_GetCellPotential
  - CRoboDacqNet, [460](#)
- Emu\_GetCellResists
  - CRoboDacqNet, [460](#)
- Emu\_GetElectrodeResists
  - CRoboDacqNet, [460](#)
- Emu\_GetNoise
  - CRoboDacqNet, [460](#)
- Emu\_SetCellCapacity
  - CRoboDacqNet, [460](#)
- Emu\_SetCellPotential
  - CRoboDacqNet, [460](#)
- Emu\_SetCellResists
  - CRoboDacqNet, [460](#)
- Emu\_SetElectrodeResists
  - CRoboDacqNet, [460](#)
- Emu\_SetNoise
  - CRoboDacqNet, [460](#)
- EnableAnalogOut
  - CSCUFunctionNet, [505](#)
- EnableAutoReset
  - CStg200xDownloadBasicNet, [563](#)
- EnableChannelsInGroup
  - CCMOSMea\_FunctionNet, [106](#)
- CDacqGroupChannelSelectionTemplateNet< Dac-  
qGroupChannelEnumTemplateNet, Dac-  
qGroupChannelEnumTemplate, CDevice-  
GroupChannelInfoTemplateNet >, [124](#)
- EnableChecksum
  - CMeaDeviceNet, [355](#)
  - COctoPotDeviceNet, [398](#)
- EnableDigitalIn
  - CMeaDeviceNet, [356](#)
  - COctoPotDeviceNet, [398](#)
- EnableExceptions
  - CMcsUsbNet, [323](#)
- EnableMultiFileMode
  - CStg200xDownloadNet, [573](#)
- EnableQueue
  - CRoboDeviceNet, [475](#)
- EnableTimestamp
  - CMeaDeviceNet, [356](#)
  - COctoPotDeviceNet, [398](#)
- EnableUserTrigger
  - CLIH3DeviceNet, [225](#)
- Encapsulator
  - Mcs::Usb, [78](#)
- enCMosMeaChipType
  - Mcs::Usb, [65](#)
- EnSTG200x\_STATUS
  - Mcs::Usb, [65](#)
- Entry
  - HeadStageIDType, [700](#)
- EOAndCRC
  - Mcs::Usb, [60](#)
- Equals
  - CMcsUsbListEntryNet, [311](#)
  - HeadStageIDType, [700](#)
  - HeadstageIDTypeObject, [702](#)
- EraseEepromRegisterPreconfig
  - CMcsUsbNet, [323](#)
- EraseFilterParameterPermanent
  - CFilterConfigurationNet, [140](#)
  - CFilterConfigurationRegisterNet, [142](#)
- ErasePermanentAdcOffset
  - CLIH3DeviceNet, [226](#)
- ErasePermanentDacOffset
  - CLIH3DeviceNet, [226](#)
- Error\_Callback\_Aquisition\_Stopped
  - CMcsUsbDacqNet, [297](#)
- Error\_Callback\_Data\_lost
  - CMcsUsbDacqNet, [297](#)
- Error\_Callback\_Frames\_Lost
  - CMcsUsbDacqNet, [297](#)
- Error\_Callback\_Packet\_Error
  - CMcsUsbDacqNet, [297](#)
- Error\_Callback\_Queue\_Full
  - CMcsUsbDacqNet, [297](#)
- Error\_Callback\_RingQueue\_Full
  - CMcsUsbDacqNet, [297](#)
- ErrorEvent
  - CMcsUsbDacqNet, [298](#)
- ExtendedMultiFileMode
  - Mcs::Usb, [83](#)
- ExternBCTester
  - Mcs::Usb, [77](#)
- ExternDTester

- Mcs::Usb, [77](#)
- ExternSTester
  - Mcs::Usb, [77](#)
- FactoryReset
  - CTcxDeviceNet, [594](#)
- Falling
  - Mcs::Usb, [64](#)
- FCB
  - Mcs::Usb, [77](#)
- FCX
  - Mcs::Usb, [77](#)
- Feedback
  - Mcs::Usb, [57](#), [70](#), [72](#), [74](#), [82](#), [85](#), [92](#)
- FeedbackGetSampleTimerCount
  - CMeaFeedbackFunctionNet, [364](#)
- FeedbackGetTriggerTotzeitFactor
  - CMeaFeedbackFunctionNet, [364](#)
- FeedbackHigh
  - Mcs::Usb, [63](#)
- FeedbackLow
  - Mcs::Usb, [63](#)
- FeedbackReg
  - Mcs::Usb, [59](#)
- FeedbackSetAnalogSource
  - CMeaFeedbackFunctionNet, [365](#)
- FeedbackSetChannelFilter
  - CMeaFeedbackFunctionNet, [365](#)
- FeedbackSetCreateFilter
  - CMeaFeedbackFunctionNet, [365](#)
- FeedbackSetDigitalMapping
  - CMeaFeedbackFunctionNet, [365](#)
- FeedbackSetFeedback
  - CMeaFeedbackFunctionNet, [365](#)
- FeedbackSetFilterOff
  - CMeaFeedbackFunctionNet, [365](#)
- FeedbackSetFilterParameter
  - CMeaFeedbackFunctionNet, [365](#)
- FeedbackSetFilterParameter32
  - CMeaFeedbackFunctionNet, [365](#)
- FeedbackSetGlobalChannelFilter
  - CMeaFeedbackFunctionNet, [366](#)
- FeedbackSetIIRFilterParameter
  - CMeaFeedbackFunctionNet, [366](#)
- FeedbackSetLogic
  - CMeaFeedbackFunctionNet, [366](#)
- FeedbackSetMkFilter
  - CMeaFeedbackFunctionNet, [366](#)
- FeedbackSetNumberOfLogics
  - CMeaFeedbackFunctionNet, [366](#)
- FeedbackSetNumberOfRateCounter
  - CMeaFeedbackFunctionNet, [366](#)
- FeedbackSetNumberOfRateDetectors
  - CMeaFeedbackFunctionNet, [366](#)
- FeedbackSetNumberOfSpikeDetectors
  - CMeaFeedbackFunctionNet, [367](#)
- FeedbackSetNumberOfTriggers
  - CMeaFeedbackFunctionNet, [367](#)
- FeedbackSetRateCounter
  - CMeaFeedbackFunctionNet, [367](#)
- FeedbackSetRateDetector
  - CMeaFeedbackFunctionNet, [367](#)
- FeedbackSetSpikeDetectorThreshold
  - CMeaFeedbackFunctionNet, [367](#)
- FeedbackSetTrigger
  - CMeaFeedbackFunctionNet, [367](#)
- FeedbackSetTriggerTotzeitFactor
  - CMeaFeedbackFunctionNet, [367](#)
- FilterActive
  - CFilterPropertyNet, [144](#)
- FilterAttributeEnumNet
  - Mcs::Usb, [65](#)
- FilterBand
  - CFilterPropertyNet, [144](#)
- FilterBandEnumNet
  - Mcs::Usb, [66](#)
- FilterCalculationDirectionEnumNet
  - Mcs::Usb, [66](#)
- FilterFamily
  - CFilterPropertyNet, [144](#)
- FilterFamilyEnumNet
  - Mcs::Usb, [66](#)
- FilterType
  - CFilterPropertyNet, [144](#)
- FilterTypeEnumNet
  - Mcs::Usb, [66](#)
- FindEndpoints
  - CGenericDevelopDeviceNet, [160](#)
- FindFilter
  - CCreateFilterNet, [119](#)
- FindFirmwareVersionMagicInBuffer
  - CMcsUsbFactoryNet, [302](#)
- FindReference
  - CRoboDeviceNet, [475](#)
- FindReferenceI
  - CRoboStatorDeviceNet, [494](#)
- FindReferencePhase0
  - CRoboDeviceNet::RoboMainLowLevelCommands, [708](#)
- FindReferencePhase0XY
  - CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands, [714](#)
- FindReferenceXY
  - CRoboStatorDeviceNet, [494](#)
- FindReferenceZ
  - CRoboStatorDeviceNet, [494](#)
- Finished
  - Mcs::Usb, [84](#)
- FirePressurePulse
  - CPPCFunctionNet, [426](#)
- FirmwareDestinationNames, [695](#)
  - Altera, [696](#)
  - Bootstrap, [696](#)
  - BUS1\_MCSBUS1, [696](#)
  - BUS1\_MCSBUS2, [696](#)
  - DSP, [696](#)
  - FPGA2, [696](#)

FPGA3, [696](#)  
 FPGA4, [696](#)  
 FPGA5, [696](#)  
 FPGA6, [696](#)  
 MCSBUS1, [697](#)  
 MCSBUS10, [697](#)  
 MCSBUS11, [697](#)  
 MCSBUS12, [697](#)  
 MCSBUS13, [697](#)  
 MCSBUS2, [697](#)  
 MCSBUS3, [697](#)  
 MCSBUS4, [697](#)  
 MCSBUS5, [697](#)  
 MCSBUS6, [697](#)  
 MCSBUS7, [697](#)  
 MCSBUS8, [698](#)  
 MCSBUS9, [698](#)  
 MCU1, [698](#)  
 PIC, [698](#)  
 PIC2, [698](#)  
 PIC3, [698](#)  
 PIC4, [698](#)  
 USB, [698](#)  
 FluidControlDevice  
     CSafeISDeviceNet, [501](#)  
 ForceStatusEvent  
     CStg200xDownloadBasicNet, [564](#)  
     CStimulusFunctionNet, [580](#)  
 FPGA10  
     Mcs::Usb, [55](#)  
 FPGA10\_BASE  
     Mcs::Usb, [56](#)  
 FPGA10\_GOLD  
     Mcs::Usb, [56](#)  
 FPGA11  
     Mcs::Usb, [55](#)  
 FPGA11\_BASE  
     Mcs::Usb, [56](#)  
 FPGA11\_GOLD  
     Mcs::Usb, [56](#)  
 FPGA12  
     Mcs::Usb, [55](#)  
 FPGA12\_BASE  
     Mcs::Usb, [56](#)  
 FPGA12\_GOLD  
     Mcs::Usb, [56](#)  
 FPGA13  
     Mcs::Usb, [55](#)  
 FPGA13\_BASE  
     Mcs::Usb, [56](#)  
 FPGA13\_GOLD  
     Mcs::Usb, [56](#)  
 FPGA14  
     Mcs::Usb, [55](#)  
 FPGA14\_BASE  
     Mcs::Usb, [56](#)  
 FPGA14\_GOLD  
     Mcs::Usb, [56](#)  
 FPGA15  
     Mcs::Usb, [55](#)  
 FPGA15\_BASE  
     Mcs::Usb, [56](#)  
 FPGA15\_GOLD  
     Mcs::Usb, [56](#)  
 FPGA16  
     Mcs::Usb, [55](#)  
 FPGA16\_BASE  
     Mcs::Usb, [56](#)  
 FPGA16\_GOLD  
     Mcs::Usb, [56](#)  
 FPGA2  
     FirmwareDestinationNames, [696](#)  
     Mcs::Usb, [55](#)  
 FPGA2\_BASE  
     Mcs::Usb, [56](#)  
 FPGA2\_GOLD  
     Mcs::Usb, [55](#)  
 FPGA3  
     FirmwareDestinationNames, [696](#)  
     Mcs::Usb, [55](#)  
 FPGA3\_BASE  
     Mcs::Usb, [56](#)  
 FPGA3\_GOLD  
     Mcs::Usb, [55](#)  
 FPGA4  
     FirmwareDestinationNames, [696](#)  
     Mcs::Usb, [55](#)  
 FPGA4\_BASE  
     Mcs::Usb, [56](#)  
 FPGA4\_GOLD  
     Mcs::Usb, [56](#)  
 FPGA5  
     FirmwareDestinationNames, [696](#)  
     Mcs::Usb, [55](#)  
 FPGA5\_BASE  
     Mcs::Usb, [56](#)  
 FPGA5\_GOLD  
     Mcs::Usb, [56](#)  
 FPGA6  
     FirmwareDestinationNames, [696](#)  
     Mcs::Usb, [55](#)  
 FPGA6\_BASE  
     Mcs::Usb, [56](#)  
 FPGA6\_GOLD  
     Mcs::Usb, [56](#)  
 FPGA7  
     Mcs::Usb, [55](#)  
 FPGA7\_BASE  
     Mcs::Usb, [56](#)  
 FPGA7\_GOLD  
     Mcs::Usb, [56](#)  
 FPGA8  
     Mcs::Usb, [55](#)  
 FPGA8\_BASE  
     Mcs::Usb, [56](#)  
 FPGA8\_GOLD

- Mcs::Usb, [56](#)
- FPGA9
  - Mcs::Usb, [55](#)
- FPGA9\_BASE
  - Mcs::Usb, [56](#)
- FPGA9\_GOLD
  - Mcs::Usb, [56](#)
- FPGA\_BASE
  - Mcs::Usb, [56](#)
- FPGA\_BOOTSTRAP
  - Mcs::Usb, [56](#)
- FPGA\_GOLD
  - Mcs::Usb, [55](#)
- FPGA\_NORMAL
  - Mcs::Usb, [53](#)
- FpgaldEnumNet
  - Mcs::Usb, [66](#)
- FrameContextGroup
  - Mcs::Usb, [61](#)
- FromIntPtr
  - StgStatusNet, [717](#)
- FromPtr
  - StgStatusNet, [717](#)
- FullCharge
  - Mcs::Usb, [69](#)
- FullSpeed
  - Mcs::Usb, [70](#)
- FunkDongleS
  - Mcs::Usb, [77](#)
- FX3MCSDataAddress
  - CMcsUsbFactoryNet, [307](#)
- FX3MCSDataDeviceIdOffset
  - CMcsUsbFactoryNet, [308](#)
- FX3MCSDataIFB1ImageOffset
  - CMcsUsbFactoryNet, [308](#)
- FX3MCSDataIFB2ImageOffset
  - CMcsUsbFactoryNet, [308](#)
- FX3MCSDataVersionOffset
  - CMcsUsbFactoryNet, [308](#)
- FYIPProgram
  - CFYIDeviceNet, [152](#)
- FYITemp
  - CFYIDeviceNet, [152](#)
- Gain
  - CMeaDeviceNet, [360](#)
- Gain100
  - Mcs::Usb, [84](#)
- Gain1000
  - Mcs::Usb, [84](#)
- Gain200
  - Mcs::Usb, [84](#)
- Gain500
  - Mcs::Usb, [84](#)
- Gate
  - Mcs::Usb, [80](#)
- Gated\_High\_Active
  - Mcs::Usb, [79](#)
- Gated\_Low\_Active
  - Mcs::Usb, [79](#)
- GE2100
  - Mcs::Usb, [78](#)
- Generic
  - Mcs::Usb, [77](#)
- Get2AnalogInput
  - CMcsBus\_SensorNet, [258](#)
- Get2DigitalInput
  - CMcsBus\_SensorNet, [258](#)
- Get4ADC
  - CMcsBus\_SensorNet, [258](#)
- Get4ADCAverage
  - CMcsBus\_SensorNet, [258](#)
- Get4ADCCatchampAverageShift
  - CMcsBus\_SensorNet, [258](#)
- Get4ADCMode
  - CMcsBus\_SensorNet, [259](#)
- Get4DAC
  - CMcsBus\_SensorNet, [259](#)
- GetAbsMaxCurrentInMicroAmp
  - CMultiwellOptoStimFunctionNet, [392](#)
- GetAccelGyroCurrentRate
  - CW2100\_FunctionNet, [622](#)
- GetAccelGyroDesiredRate
  - CW2100\_FunctionNet, [622](#)
- GetAccelGyroEnabled
  - CW2100\_FunctionNet, [622](#)
- GetAccelRange
  - CW2100\_FunctionNet, [622](#)
- GetActiveRunningTableNumber
  - CWarnerValveControllerDeviceNet, [657](#)
- GetActiveRunningTableNumberEvent
  - CWarnerValveControllerDeviceNet, [673](#)
- GetAdapterCode
  - CMealImpedanceDeviceNet, [368](#)
  - CTEERFunctionNet, [607](#)
- GetAdapterType
  - CMcsUsbDacqNet, [280](#)
- GetAdc
  - CFluidControlDeviceNet, [146](#)
- GetAdcDataFormat
  - CMcsUsbDacqNet, [280](#)
- GetADCGain
  - CGrapheneFunctionNet, [177](#)
- GetADCInputOffset
  - CCMOSMea\_FunctionNet, [106](#)
- GetADCOffset
  - CGrapheneFunctionNet, [177](#), [178](#)
- GetAdcOffset
  - CLIH3DeviceNet, [226](#)
  - COctoPotDeviceNet, [398](#)
- GetAdcOffsetU1
  - CTEERFunctionNet, [607](#)
- GetAdcOffsetU2
  - CTEERFunctionNet, [607](#)
- GetADCs
  - CMcsBus\_SensorNet, [259](#)
- GetADCsLoop



- CMcsBus\_SensorNet, [259](#)
- GetAdcZero
  - CMcsUsbDacqNet, [280](#)
- GetAirpressure
  - CRoboDeviceNet, [475](#)
- GetAirpressureLimit
  - CRoboDeviceNet, [475](#)
- GetAirValve
  - CRoboDeviceNet, [475](#)
- GetAlIDigout
  - CRoboDacqNet, [461](#)
- GetAmplification
  - CPgaDeviceNet, [410](#)
- GetAmplitude\_nA
  - CTEERFunctionNet, [607](#)
- GetAnalogGain
  - CMeaDeviceNet, [357](#)
- GetAnalogOutADCRange
  - CSCUFunctionNet, [506](#)
- GetAnalogOutChannel
  - CW2100\_FunctionNet, [622](#)
- GetAnalogOutChannels
  - CSCUFunctionNet, [506](#)
- GetAnalogOutDACRange
  - CSCUFunctionNet, [506](#)
- GetAnalogOutFilter
  - CW2100\_FunctionNet, [622](#)
- GetAnalogThresholdHigh
  - CWarnerValveControllerDeviceNet, [657](#)
- GetAnalogThresholdHighEvent
  - CWarnerValveControllerDeviceNet, [673](#)
- GetAnalogThresholdLow
  - CWarnerValveControllerDeviceNet, [657](#)
- GetAnalogThresholdLowEvent
  - CWarnerValveControllerDeviceNet, [673](#)
- GetAnalogValueUnit
  - CMcsUsbDacqNet, [280](#)
- GetAnalogVoltage
  - CPPCFunctionNet, [426](#)
  - CPPS\_FunctionNet, [435](#)
  - CWarnerValveControllerDeviceNet, [657](#)
- GetAnalogVoltageEvent
  - CWarnerValveControllerDeviceNet, [674](#)
- GetAnalogVoltageRange
  - CPPCFunctionNet, [426](#)
- GetAnalogVoltages
  - CPPS\_FunctionNet, [435](#)
- GetArraySize
  - CMealImpedanceDeviceNet, [368](#)
- GetAudioChannels
  - CMeaAudioFunctionNet, [341](#), [342](#)
  - CW2100\_FunctionNet, [622](#)
- GetAudioOutDacParameter
  - CLIH3DeviceNet, [226](#)
- GetAutocalibrationDisabled
  - CStg200xBasicNet, [527](#)
- GetAvailableBaseSamplerates
  - CCMOSMeaDeviceNet, [115](#)
- CGrapheneASICDeviceNet, [173](#)
- GetAvailableChambers
  - CWarnerUssingFunctionNet, [639](#)
- GetAvailableDeviceList
  - CRFFFunctionNet, [450](#)
- GetAvailableDeviceListEx
  - CRFFFunctionNet, [450](#)
- GetAvailableDockSmartMultiwellPlates
  - CSCUFunctionNet, [506](#)
- GetAvailableHeadstages
  - CSCUFunctionNet, [507](#)
  - CW2100\_FunctionNet, [622](#)
- GetAvailableHeadstagesEvent
  - CSCUFunctionNet, [520](#)
- GetAvailableMemory
  - CStg200xBasicNet, [528](#)
  - CStimulusFunctionNet, [581](#)
- GetAvailableSampleRates
  - CMcsUsbDacqNet::CHWInfo, [215](#)
- GetAvailableStateList
  - CRFFFunctionNet, [450](#)
- GetAvailableStateListEx
  - CRFFFunctionNet, [450](#)
- GetAvailableVoltageRangesInMicroVolt
  - CMcsUsbDacqNet::CHWInfo, [215](#)
- GetAvailableVoltageRangesInMicroVoltAndStringsInMilliVolt
  - CMcsUsbDacqNet::CHWInfo, [215](#)
- GetAxisConfig
  - CRoboDeviceNet::RoboMainLowLevelCommands, [708](#)
- GetAxisLED
  - CRoboocyte2DeviceNet, [492](#)
- GetAxisParametersSignedEeprom
  - CMcsBus\_AxisParametersNet, [234](#)
- GetAxisParametersUnsignedEeprom
  - CMcsBus\_AxisParametersNet, [234](#)
- GetBaseFrequency
  - CRFFFunctionNet, [451](#)
- GetBaseSamplerate
  - CCMOSMeaDeviceNet, [116](#)
  - CGrapheneASICDeviceNet, [174](#)
- GetBath
  - CCMOSMea\_FunctionNet, [106](#)
- GetBathMode
  - CCMOSMea\_FunctionNet, [106](#)
- GetBatteryState
  - CW2100\_FunctionNet, [623](#)
- GetBatteryVoltage
  - CMultiBatteryChargerDeviceNet, [377](#)
- GetBiQuad
  - CCreateFilterNet, [119](#)
- GetBiQuads
  - CCreateFilterNet, [119](#)
- GetBlankingEnable
  - CStg200xBasicNet, [528](#)
- GetBoardTemp
  - CTcxDeviceNet, [594](#)
- GetBoostAlwaysOnMode



- CW2100\_StimulatorFunctionNet, 629
- GetBoostPreTime
  - CW2100\_StimulatorFunctionNet, 629
- GetBubbleState
  - CPPS\_FunctionNet, 435
- GetBubbleStatus
  - CMcsBus\_SensorNet, 259
- GetBuffer
  - CGenericDevelopDeviceNet, 160
- GetBusAddress
  - CMcsBusNet, 273
- GetBusAddressEeprom
  - CMcsBusNet, 273
- GetByteBuffer
  - CGenericDevelopDeviceNet, 161
- GetBytesAvailable
  - CSerialPortNet, 521
- GetBytesPerSample
  - CTEERFunctionNet, 608
- GetCalibration
  - CTcxDeviceNet, 594
- GetCalibrationDecp
  - CTcxDeviceNet, 594
- GetCalibrationMax
  - CTcxDeviceNet, 594
- GetCalibrationMin
  - CTcxDeviceNet, 594
- GetCanDisableCalibration
  - CStg200xBasicNet, 529
- GetCanDisableCurrentswitch
  - CStg200xBasicNet, 529
- GetCapacityC
  - CRoboDacqNet, 461
- GetCapacityV
  - CRoboDacqNet, 461
- GetCapacityX
  - CRoboDacqNet, 461
- GetCardinalDacqSamplerate
  - CInterfaceboardFunctionNet, 222
  - CLIH3DeviceNet, 227
- GetCardinalStgOutputrate
  - CInterfaceboardFunctionNet, 222
  - CLIH3DeviceNet, 227
- GetChannel
  - CMcsUsbDacqNet::ChannelBlockNet, 194
  - CSw2to64DeviceNet, 590
- GetChannelDataFillSize
  - CMcsUsbDacqNet, 280
- GetChannelDataI16
  - CCMOSMeaDeviceNet, 116
- GetChannelDataI32
  - CCMOSMeaDeviceNet, 116
- GetChannelDataUI16
  - CCMOSMeaDeviceNet, 116
- GetChannelDataUI32
  - CCMOSMeaDeviceNet, 116
- GetChannelLayout
  - CMcsUsbDacqNet, 280
- GetChannels
  - CMultiBatteryChargerDeviceNet, 377
  - CSw2to64DeviceNet, 590
- GetChannelsCountOfChamber
  - CWarnerUssingFunctionNet, 639
- GetChannelsInBlock
  - CMcsUsbDacqNet, 281
- GetChannelState
  - CMultiBatteryChargerDeviceNet, 377
- GetChargeCapacity
  - CMultiBatteryChargerDeviceNet, 377
- GetChargeCurrent
  - CMultiBatteryChargerDeviceNet, 378
- GetChargingMode
  - CMultiBatteryChargerDeviceNet, 378
- GetChargingPCoefficient
  - CMultiBatteryChargerDeviceNet, 378
- GetCheckChecksum
  - CMcsUsbDacqNet::ChannelBlockNet, 195
- GetChecksumFromFX3Image
  - CMcsUsbFactoryNet, 302
- GetCheckVoltage
  - COkuvisionStimulatorDeviceNet, 402
- GetClampAmpSerialNumber
  - CRoboDacqNet, 461
- GetClampMode
  - CTEERFunctionNet, 608
  - CWarnerUssingFunctionNet, 639
- GetCMOSDataDictionary
  - CCMOSMeaDeviceNet, 116
- GetCoilCommunication
  - CPositionIIDeviceNet, 413
- GetColorRgb
  - CMultiwellOptoStimFunctionNet, 393
- GetColorStr
  - CMultiwellOptoStimFunctionNet, 393
- GetCommand
  - CMcsBusNet, 273, 274
  - CPedoterDeviceNet, 407
  - CRoboDacqNet, 461
- GetComplianceVoltageThreshold
  - CWarnerUssingFunctionNet, 639
- GetConfiguration
  - CMcsUsbNet, 324
- GetConfigurationBit
  - CRoboDacqNet, 461
- GetConfigurationBitAxc
  - CRoboDacqNet, 461
- GetConfigurationBitBlu\_Led
  - CRoboDacqNet, 461
- GetConfigurationBitBlu\_LedToggleFast
  - CRoboDacqNet, 461
- GetConfigurationBitBlu\_LedToggleSlow
  - CRoboDacqNet, 461
- GetConfigurationBitCC\_Gen
  - CRoboDacqNet, 462
- GetConfigurationBitCV\_Gen
  - CRoboDacqNet, 462

GetConfigurationBitRC\_Gen  
     CRoboDacqNet, [462](#)  
 GetConfigurationBitRed\_Led  
     CRoboDacqNet, [462](#)  
 GetConfigurationBitRed\_LedSaturation  
     CRoboDacqNet, [462](#)  
 GetConfigurationBitRed\_LedToggleFast  
     CRoboDacqNet, [462](#)  
 GetConfigurationBitRed\_LedToggleSlow  
     CRoboDacqNet, [462](#)  
 GetConfigurationBitRelais  
     CRoboDacqNet, [462](#)  
 GetConfigurationBitRV\_Gen  
     CRoboDacqNet, [462](#)  
 GetConfigurationBits  
     CRoboDacqNet, [462](#)  
 GetConfigurationBitStream  
     CRoboDacqNet, [462](#)  
 GetConfigurationBitSupply  
     CRoboDacqNet, [463](#)  
 GetConnectedDevice  
     CRFFunctionNet, [451](#)  
 GetControllerParams  
     CTEERFunctionNet, [608](#)  
 GetCpp  
     CCreateFilterNet, [119](#)  
 GetCrossTalkOffset  
     CRoboDacqNet, [463](#)  
 GetCrossTalkOptimum  
     CRoboDacqNet, [463](#)  
 GetCur2VolResistance  
     CGrapheneFunctionNet, [178](#)  
 GetCurrent  
     CTcxDeviceNet, [594](#)  
 GetCurrentAirvalve  
     CRoboDeviceNet, [475](#)  
 GetCurrentAirvalveLimit  
     CRoboDeviceNet, [475](#)  
 GetCurrentCycle  
     CMeaCoatDeviceNet, [349](#)  
 GetCurrentEditTableNumber  
     CWarnerValveControllerDeviceNet, [658](#)  
 GetCurrentEnable  
     CTEERFunctionNet, [608](#)  
 GetCurrentFactor  
     COKuvisionStimulatorDeviceNet, [402](#)  
 GetCurrentNumberOfValves  
     CWarnerValveControllerDeviceNet, [658](#)  
 GetCurrentNumberOfValvesEvent  
     CWarnerValveControllerDeviceNet, [674](#)  
 GetCurrentPosition  
     CRoboDeviceNet, [476](#)  
 GetCurrentPositionI  
     CRoboStatorDeviceNet, [494](#)  
 GetCurrentPositionXY  
     CRoboStatorDeviceNet, [494](#)  
 GetCurrentPositionZ  
     CRoboStatorDeviceNet, [495](#)  
 GetCurrentRangeInNanoAmp  
     CStg200xBasicNet, [529](#)  
     CStimulusFunctionNet, [581](#)  
     CW2100\_StimulatorFunctionNet, [629](#)  
 GetCurrentRangeInNanoAmpByIndex  
     CStg200xBasicNet, [530](#)  
 GetCurrentRangeListInNanoAmp  
     CStg200xBasicNet, [530](#)  
 GetCurrentRangeSelectedIndex  
     CStg200xBasicNet, [530](#)  
 GetCurrentResolutionInNanoAmp  
     CStg200xBasicNet, [530](#)  
     CStimulusFunctionNet, [581](#)  
     CW2100\_StimulatorFunctionNet, [629](#)  
 GetCurrentResolutionInPicoAmp  
     CStg200xBasicNet, [531](#)  
     CStimulusFunctionNet, [581](#)  
     CW2100\_StimulatorFunctionNet, [629](#)  
 GetCycle  
     CMeaCleanDeviceNet, [345](#)  
 GetCycles  
     CMeaCleanDeviceNet, [345](#)  
     CMeaCoatDeviceNet, [349](#)  
 GetD  
     CTcxDeviceNet, [595](#)  
 GetDacAmplificationFactor  
     CStg200xBasicNet, [531](#)  
 GetDacIdleValue  
     CLIH3DeviceNet, [227](#)  
 GetDACOffset  
     CGrapheneFunctionNet, [179](#)  
     COKuvisionStimulatorDeviceNet, [402](#)  
 GetDacOffset  
     CDacCalibrationFunctionNet, [121](#)  
     CLIH3DeviceNet, [227](#)  
     COctoPotDeviceNet, [398](#)  
 GetDacPampsPerDigitHighCurrentRange  
     CWarnerUssingFunctionNet, [640](#)  
 GetDacPampsPerDigitLowCurrentRange  
     CWarnerUssingFunctionNet, [640](#)  
 GetDacqRunStatus  
     CLIH3DeviceNet, [228](#)  
 GetDacRange  
     CW2100\_FunctionNet, [623](#)  
 GetDACResolution  
     CStg200xBasicNet, [531](#)  
     CStimulusFunctionNet, [583](#)  
     CW2100\_StimulatorFunctionNet, [630](#)  
 GetDACs  
     CMcsBus\_SensorNet, [259](#)  
 GetDacUseIdleValue  
     CLIH3DeviceNet, [228](#)  
 GetDacZero  
     CTEERFunctionNet, [609](#)  
     CWarnerUssingFunctionNet, [640](#)  
 GetDataFormat  
     CMcsUsbDacqNet, [281](#)  
 GetDataMode

- CMcsUsbDacqNet, [281](#)
- GetDDecp
  - CTcxDeviceNet, [595](#)
- GetDebugData
  - CPositionIIDeviceNet, [413](#)
- GetDefaultFilterAttributes
  - CFilterConfigurationNet, [140](#)
- GetDestination
  - CMcsUsbFactoryNet, [303](#)
- GetDestinationCode
  - DriverVersionNet, [691](#)
- GetDestinationDisplayLabel
  - CMcsUsbFactoryNet, [303](#)
- GetDestinationName
  - CMcsUsbFactoryNet, [303](#)
  - DriverVersionNet, [692](#)
- GetDestinationSerialNumber
  - CMcsUsbFactoryNet, [303](#)
- GetDestinationTargetAddress
  - CMcsUsbFactoryNet, [303](#)
- GetDetectionThreshold
  - CMcsBus\_SensorNet, [259](#)
- GetDetectorValue
  - CMcsBus\_SensorNet, [259](#)
- GetDevice
  - CTcxDeviceNet, [595](#)
- GetDeviceCannotStallOutRequests
  - CMcsUsbNet, [324](#)
- GetDeviceCapableSpeed
  - CMcsUsbNet, [324](#)
- GetDeviceEnum
  - CMcsUsbNet, [324](#)
- GetDeviceGroupChannelInfos
  - CDacqGroupChannelSelectionTemplateNet< Dac-  
qGroupChannelEnumTemplateNet, Dac-  
qGroupChannelEnumTemplate, CDevice-  
GroupChannelInfoTemplateNet >, [124](#)
- GetDeviceId
  - CMcsUsbNet, [324](#)
- GetDeviceList
  - CPositionImpDeviceNet, [421](#)
- GetDeviceNames
  - CRadioControlledDevicesNet, [444](#)
- GetDeviceRootHubVendorEnum
  - CMcsUsbNet, [324](#)
- GetDeviceRootHubVendorID
  - CMcsUsbNet, [324](#)
- GetDeviceRootHubVendorName
  - CMcsUsbNet, [324](#)
- GetDeviceSpeed
  - CMcsUsbNet, [325](#)
- GetDeviceType
  - CTcxDeviceNet, [595](#)
- GetDevname
  - CTcxDeviceNet, [595](#)
- GetDigin
  - CFluidControlDeviceNet, [146](#)
- GetDigInState
  - CLIH3DeviceNet, [228](#)
- GetDiginValue
  - CStg200xBasicNet, [531](#)
- GetDigitalData
  - CMeaDigitalDataFunctionNet, [362](#)
- GetDigitalIn
  - CPPCFunctionNet, [428](#)
  - CPPS\_FunctionNet, [435](#)
- GetDigitalOutPortValve
  - CWarnerValveControllerDeviceNet, [658](#)
- GetDigitalOutPortValveEvent
  - CWarnerValveControllerDeviceNet, [674](#)
- GetDigitalPortDirection
  - CWarnerValveControllerDeviceNet, [658](#)
- GetDigitalPortDirectionEvent
  - CWarnerValveControllerDeviceNet, [674](#)
- GetDigitalSource
  - CMcsUsbDacqNet, [281–284](#)
- GetDigitalStimulatorTrigger
  - CW2100\_StimulatorFunctionNet, [630](#)
- GetDigitalStimulatorTriggerSlope
  - CW2100\_StimulatorFunctionNet, [630](#)
- GetDigout
  - CFluidControlDeviceNet, [147](#)
  - CRoboDacqNet, [463](#)
- GetDigoutMode
  - CStg200xBasicNet, [532](#)
- GetDigoutValue
  - CStg200xBasicNet, [532](#)
- GetDIO
  - CMcsBus\_FYIExtensionNet, [236](#)
- GetDischargeCapacity
  - CMultiBatteryChargerDeviceNet, [378](#)
- GetDischargeCurrent
  - CMultiBatteryChargerDeviceNet, [379](#)
- GetDischargeCurrentSetPoint
  - CMultiBatteryChargerDeviceNet, [379](#)
- GetDisplayMode
  - CWarnerValveControllerDeviceNet, [659](#)
- GetDisplayModeEvent
  - CWarnerValveControllerDeviceNet, [674](#)
- GetDisplayText
  - CRoboDacqNet, [463](#)
- GetDMax
  - CTcxDeviceNet, [595](#)
- GetDMin
  - CTcxDeviceNet, [595](#)
- GetDockSmartMultiwellPlatePowerstate
  - CSCUFunctionNet, [507](#)
- GetDockSmartMultiwellPlateWellMux
  - CSCUFunctionNet, [507](#)
- GetDownsampleFactor
  - CRoboDacqNet, [463](#)
- GetDSPHighPassByIndex
  - CIntanMea\_FunctionNet, [218](#)
- GetDuration
  - CMeaCoatDeviceNet, [349](#)
- GetDuty

- CTcxDeviceNet, 595
- GetEEpromPage
  - CLIH3DeviceNet, 228
- GetElectrodeDacMux
  - CStg200xBasicNet, 532
- GetElectrodeEnable
  - CStg200xBasicNet, 533
- GetElectrodeMode
  - CStg200xBasicNet, 534
- GetEnableAmplifierProtectionSwitch
  - CStg200xBasicNet, 535
- GetEnabledChannelsInGroup
  - CCMOSMea\_FunctionNet, 107
  - CDacqGroupChannelSelectionTemplateNet< Dac-  
qGroupChannelEnumTemplateNet, Dac-  
qGroupChannelEnumTemplate, CDevice-  
GroupChannelInfoTemplateNet >, 124, 125
- GetEnableHeaterLimit
  - CTcxDeviceNet, 596
- GetEnableThermocouple
  - CTcxDeviceNet, 596
- GetEntry
  - CMcsUsbListEntryNet, 311, 312
- GetEntryCount
  - CMcsUsbListEntryNet, 312
- GetEnumerationSpeed
  - CMeaDeviceNet, 357
- GetErrorAirpressure
  - CRoboDeviceNet, 476
- GetErrorCurrentAirvalve
  - CRoboDeviceNet, 476
- GetErrorMessage
  - CMcsUsbDacqNet, 284
- GetErrorText
  - CMcsUsbNet, 325
- GetErrorVoltage12V
  - CRoboDeviceNet, 476
- GetErrorVoltage5V
  - CRoboDeviceNet, 476
- GetErrorVoltageAirvalve
  - CRoboDeviceNet, 476
- GetErrorVoltageRs485A
  - CRoboDeviceNet, 476
- GetErrorVoltageRs485B
  - CRoboDeviceNet, 476
- GetErrorVoltageValves
  - CRoboDeviceNet, 477
- GetEventData
  - CPositionIIDeviceNet, 414
- GetExternalElectrodeEnable
  - CStg200xBasicNet, 536
- GetFAAmplification
  - CStg200xBasicNet, 536
- GetFilter
  - CRoboDacqNet, 463
- GetFilterAttributes
  - CFilterConfigurationNet, 140
- GetFilterCoeffs
  - CRoboDacqNet, 463
- GetFilterParametersHeadstage
  - CWClassicFunctionNet, 681
- GetFilterProperties
  - CSCUFunctionNet, 508
  - CW2100\_FunctionNet, 623
- GetFilterProperty
  - CMcsUsbDacqNet, 284
  - CSCUFunctionNet, 508
  - CW2100\_FunctionNet, 623
- GetFinalDischargeVoltage
  - CMultiBatteryChargerDeviceNet, 379
- GetFirmwareVersion
  - CMcsUsbNet, 325
- GetFirmwareVersionFromFile
  - CMcsUsbFactoryNet, 304
- GetFirmwareVersionFromHexFile
  - CMcsUsbFactoryNet, 304
- GetFPGA Firmware Type
  - CW2100\_FunctionNet, 623
- GetFrameErrorCounter
  - CTEERFunctionNet, 609
- GetFrequency
  - CRadioControlledDevicesNet, 445
- GetFrequencyRange
  - CPgaDeviceNet, 410
- GetGain
  - CMeaDeviceNet, 357
  - CPgaDeviceNet, 411
- GetGate
  - CCMOSMea\_FunctionNet, 107
- GetGilsonDevice
  - CRoboocyte2DeviceNet, 492
- GetGlobalRepeat
  - CDigOutStimulatorFunctionNet, 131
- GetGNDI
  - CCMOSMea\_FunctionNet, 107
- GetGroupADCBits
  - CCMOSMea\_FunctionNet, 107
- GetGroupChannelBitmaskBySelect
  - CCMOSMea\_FunctionNet, 107
- GetGroupChannelBitmaskHS1NCBathCurrent
  - CCMOSMea\_FunctionNet, 107, 108
- GetGroupChannelBitmaskHS1NCCol2Current
  - CCMOSMea\_FunctionNet, 108
- GetGroupChannelBitmaskHS1NChipTemp
  - CCMOSMea\_FunctionNet, 108
- GetGroupChannelBitmaskHS1Sidebands
  - CCMOSMea\_FunctionNet, 108
- GetGroupChannelBitmaskHS1TriggerStatus
  - CCMOSMea\_FunctionNet, 108, 109
- GetGroupChannelBitmaskIFDigChannels
  - CCMOSMea\_FunctionNet, 109
- GetGroupChannelBitmaskInterfaceADC
  - CCMOSMea\_FunctionNet, 109
- GetGroupChannelBitmaskPacketFrameContext
  - CCMOSMea\_FunctionNet, 109
- GetGroupChannelBitmaskSTG1DACSignal

- CCMOSMea\_FunctionNet, [109](#), [110](#)
- GetGroupChannelDataI16
  - CMcsUsbDacqNet::ChannelBlockNet, [195](#)
- GetGroupChannelDataI32
  - CMcsUsbDacqNet::ChannelBlockNet, [195](#)
- GetGroupChannelDataUI16
  - CMcsUsbDacqNet::ChannelBlockNet, [196](#)
- GetGroupChannelDataUI32
  - CMcsUsbDacqNet::ChannelBlockNet, [196](#)
- GetGroupDCOffset
  - CCMOSMea\_FunctionNet, [110](#)
- GetGroupID
  - CCMOSMea\_FunctionNet, [110](#)
  - CDacqGroupChannelSelectionTemplateNet< Dac-qGroupChannelEnumTemplateNet, Dac-qGroupChannelEnumTemplate, CDevice-GroupChannelInfoTemplateNet >, [125](#)
- GetGroupNumberOfChannels
  - CCMOSMea\_FunctionNet, [110](#)
  - CDacqGroupChannelSelectionTemplateNet< Dac-qGroupChannelEnumTemplateNet, Dac-qGroupChannelEnumTemplate, CDevice-GroupChannelInfoTemplateNet >, [125](#)
- GetGroupResolutionPerDigit
  - CCMOSMea\_FunctionNet, [110](#)
- GetGroupSampleSize
  - CCMOSMea\_FunctionNet, [111](#)
  - CDacqGroupChannelSelectionTemplateNet< Dac-qGroupChannelEnumTemplateNet, Dac-qGroupChannelEnumTemplate, CDevice-GroupChannelInfoTemplateNet >, [125](#)
- GetGroupType
  - CCMOSMea\_FunctionNet, [111](#)
  - CDacqGroupChannelSelectionTemplateNet< Dac-qGroupChannelEnumTemplateNet, Dac-qGroupChannelEnumTemplate, CDevice-GroupChannelInfoTemplateNet >, [125](#)
- GetGroupUnit
  - CCMOSMea\_FunctionNet, [111](#)
- GetGyroRange
  - CW2100\_FunctionNet, [623](#)
- GetHardwareMaxRange
  - CMcsUsbDacqNet, [284](#)
- GetHardwareMinRange
  - CMcsUsbDacqNet, [284](#)
- GetHardwareRevision
  - CMcsUsbNet, [325](#)
- GetHasChecksum
  - CWClassicFunctionNet, [681](#)
- GetHashCode
  - HeadstageIDTypeObject, [702](#)
- GetHasLowCurrentswitchLatency
  - CStg200xBasicNet, [536](#)
- GetHasRedLedHeadstage
  - CWClassicFunctionNet, [681](#)
- GetHasThermocouple
  - CTcxDeviceNet, [596](#)
- GetHeadstage
  - CStg200xBasicNet, [536](#)
- GetHeadstageActive
  - CMcsUsbNet, [326](#)
- GetHeadstageAdcBits
  - CSCUFunctionNet, [508](#)
- GetHeadstageAdcRangeInMicroVolt
  - CSCUFunctionNet, [509](#)
- GetHeadstageDacBits
  - CSCUFunctionNet, [509](#)
- GetHeadstageDacCurrentRangeInMicroAmpere
  - CSCUFunctionNet, [509](#)
- GetHeadstageDacCurrentResolutionInNanoAmpere
  - CSCUFunctionNet, [510](#)
- GetHeadstageDacVoltageRangeInMilliVolt
  - CSCUFunctionNet, [510](#)
- GetHeadstageDacVoltageResolutionInMicroVolt
  - CSCUFunctionNet, [510](#)
- GetHeadstageFrameCyclesToCompare
  - CSCUFunctionNet, [511](#)
- GetHeadstageGainInPer mille
  - CSCUFunctionNet, [511](#)
- GetHeadstageID
  - CMcsUsbNet, [326](#)
  - CSCUFunctionNet, [511](#)
- GetHeadstageLinkSpeed
  - CSCUFunctionNet, [512](#)
- GetHeadstageNumberOfAnalogChannels
  - CSCUFunctionNet, [512](#)
- GetHeadstageNumberOfStimulationChannels
  - CSCUFunctionNet, [512](#)
- GetHeadstageOnOff
  - CW2100\_FunctionNet, [623](#)
  - CWClassicFunctionNet, [681](#)
- GetHeadstagePowerStateAtStart
  - CSCUFunctionNet, [512](#)
- GetHeadstagePresent
  - CMcsUsbNet, [326](#)
- GetHeadstageSamplerate
  - CSCUFunctionNet, [513](#)
- GetHeadstageSamplingActive
  - CW2100\_FunctionNet, [623](#)
- GetHeadstageSerialNumber
  - CSCUFunctionNet, [513](#)
- GetHeaterLimit
  - CTcxDeviceNet, [596](#)
- GetHeaterTemp
  - CTcxDeviceNet, [596](#)
- GetHighCurrentRange
  - CWarnerUssingFunctionNet, [641](#)
- GetHighpassFilterEnable
  - CFilterConfigurationNet, [140](#)
- GetHWConfig
  - CRoboDeviceNet::RoboMainLowLevelCommands, [708](#)
- GetHWRevision
  - CRoboDeviceNet::RoboMainLowLevelCommands, [708](#)
- GetHWRevisionEeprom

- CMcsBusNet, 274
- GetI
  - CTcxDeviceNet, 596
- GetIC
  - CRoboDacqNet, 463
- GetIClamp
  - CRoboDacqNet, 463
- GetICoeff
  - CRobo\_FYITemp\_FunctionNet, 455
- GetICOffset
  - CRoboDacqNet, 463
- GetIDecp
  - CTcxDeviceNet, 596
- GetIdleModeOffset
  - CWarnerUssingFunctionNet, 641
- GetIGain
  - CRoboDacqNet, 464
- GetIMax
  - CTcxDeviceNet, 597
- GetIMin
  - CTcxDeviceNet, 597
- GetImpedanceResult
  - CIntanMea\_FunctionNet, 218
- GetImpedanceTestFrequency
  - CMealImpedanceDeviceNet, 369
- GetImpId
  - CPositionImpDeviceNet, 422
- GetImplantCurrentSetpoint
  - CPositionIIDeviceNet, 414
- GetImplantResult
  - CPositionIIDeviceNet, 415
- GetImplantState
  - CPositionIIDeviceNet, 415
- GetInMovement
  - CRoboDeviceNet, 477
- GetIntanRegister
  - CIntanMea\_FunctionNet, 218
- GetIntBuffer
  - CGenericDevelopDeviceNet, 161
- GetIO
  - CWarnerValveControllerDeviceTesterFunctionNet, 678
- GetIOut
  - CTcxDeviceNet, 597
- GetIoVoltage
  - CInterfaceboard2FunctionNet, 220
- GetLastAnswer
  - CGilsonDeviceNet, 172
- GetLastUSBError
  - CMcsUsbNet, 326
- GetLatency
  - CMcsBus\_SensorNet, 260
- GetLatencyCounter
  - CMcsBus\_SensorNet, 260
- GetLayoutConfiguration
  - CMEA2100x256FunctionNet, 340
- GetLEDSwitch
  - CMcsBus\_ExtensionNet, 235
- GetLength
  - CRobo\_FYIProgram\_FunctionNet, 453
- GetLiquidResistance
  - CTEERFunctionNet, 609
  - CWarnerUssingFunctionNet, 641
- GetListModelIndexRange
  - CStg200xBasicNet, 537
- GetListModelTriggerSource
  - CStg200xBasicNet, 537
- GetLowCurrentRange
  - CWarnerUssingFunctionNet, 642
- GetLowerFrequencyByIndex
  - CIntanMea\_FunctionNet, 218
- GetMajor
  - DriverVersionNet, 692
- GetMaxChunkSize\_Byte
  - CTEERFunctionNet, 609
- GetMaxCurrent
  - CMeaCoatDeviceNet, 349
- GetMaxDurationHighCurrentInMicroSec
  - CMultiwellOptoStimFunctionNet, 393
- GetMaxDutyCycleHighCurrent
  - CMultiwellOptoStimFunctionNet, 393
- GetMaxHeaterPowerMultiwell
  - CTcxDeviceNet, 597
- GetMaxNoPressure
  - CRoboDeviceNet::RoboMainLowLevelCommands, 708
- GetMaxNoPressureWaitTime
  - CRoboDeviceNet::RoboMainLowLevelCommands, 708
- GetMaxNumberOfHeadstages
  - CSCUFunctionNet, 513
- GetMaxNumOfColumns
  - CCMOSMea\_FunctionNet, 111
- GetMaxP
  - CTcxDeviceNet, 597
- GetMaxpDecp
  - CTcxDeviceNet, 597
- GetMaxpMax
  - CTcxDeviceNet, 597
- GetMaxpMin
  - CTcxDeviceNet, 598
- GetMaxPower
  - COKuvisionStimulatorDeviceNet, 402
  - CRobo\_FYITemp\_FunctionNet, 455
- GetMaxPressureWaitTime
  - CRoboDeviceNet::RoboMainLowLevelCommands, 708
- GetMaxReadableColumns
  - CCMOSMeaDeviceNet, 116
- GetMaxSamplingFrequency
  - CMcsUsbDacqNet, 284
- GetMaxStimulusChannelsPerHeadstage
  - CSCUFunctionNet, 514
- GetMaxVoltage
  - CMeaCleanDeviceNet, 345
  - COKuvisionStimulatorDeviceNet, 402



- GetMCAcceleration
  - CMcsBus\_MotorControlNet, [240](#)
- GetMCAccelerationEeprom
  - CMcsBus\_MotorControlNet, [240](#)
- GetMCAccelerationShortCommand
  - CMcsBus\_MotorControlNet, [241](#)
- GetMCAxisRevisionEeprom
  - CMcsBus\_MotorControlNet, [241](#)
- GetMCBreakCurrent
  - CMcsBus\_MotorControlNet, [241](#)
- GetMCBreakCurrentEeprom
  - CMcsBus\_MotorControlNet, [241](#)
- GetMCConfig
  - CMcsBus\_MotorControlNet, [241](#)
- GetMCConfigEeprom
  - CMcsBus\_MotorControlNet, [241](#)
- GetMCCurrent
  - CMcsBus\_MotorControlNet, [241](#)
- GetMCCurrentEeprom
  - CMcsBus\_MotorControlNet, [242](#)
- GetMCCurrentMode
  - CMcsBus\_MotorControlNet, [242](#)
- GetMCCurrentModeEeprom
  - CMcsBus\_MotorControlNet, [242](#)
- GetMCCurrentModeShortCommand
  - CMcsBus\_MotorControlNet, [242](#)
- GetMCCurrentPosition
  - CMcsBus\_MotorControlNet, [242](#)
- GetMCCurrentShortCommand
  - CMcsBus\_MotorControlNet, [242](#)
- GetMCCurrentSpeed
  - CMcsBus\_MotorControlNet, [242](#)
- GetMCMaxAcceleration
  - CMcsBus\_MotorControlNet, [243](#)
- GetMCMaxAccelerationEeprom
  - CMcsBus\_MotorControlNet, [243](#)
- GetMCMaxCurrent
  - CMcsBus\_MotorControlNet, [243](#)
- GetMCMaxCurrentEeprom
  - CMcsBus\_MotorControlNet, [243](#)
- GetMCMaxSpeed
  - CMcsBus\_MotorControlNet, [243](#)
- GetMCMaxSpeedEeprom
  - CMcsBus\_MotorControlNet, [243](#)
- GetMCMaxTravel
  - CMcsBus\_MotorControlNet, [243](#)
- GetMCMaxTravelEeprom
  - CMcsBus\_MotorControlNet, [244](#)
- GetMCMaxTravelShortCommand
  - CMcsBus\_MotorControlNet, [244](#)
- GetMCMovement
  - CMcsBus\_MotorControlNet, [244](#)
- GetMCNewPosition
  - CMcsBus\_MotorControlNet, [244](#)
- GetMCOutputOnOff
  - CMcsBus\_MotorControlNet, [244](#)
- GetMCPhase
  - CMcsBus\_MotorControlNet, [244](#)
- GetMCPhaseOffset
  - CMcsBus\_MotorControlNet, [244](#)
- GetMCReference
  - CMcsBus\_MotorControlNet, [245](#)
- GetMCReferenceCurrent
  - CMcsBus\_MotorControlNet, [245](#)
- GetMCReferenceCurrentEeprom
  - CMcsBus\_MotorControlNet, [245](#)
- GetMCRegulatorGain
  - CMcsBus\_MotorControlNet, [245](#)
- GetMCRegulatorGainEeprom
  - CMcsBus\_MotorControlNet, [245](#)
- GetMcsBus\_Extension
  - CRobocyte2DeviceNet, [492](#)
- GetMCScalingFactor
  - CMcsBus\_MotorControlNet, [245](#)
- GetMCScalingFactorEeprom
  - CMcsBus\_MotorControlNet, [245](#)
- GetMCSpeed
  - CMcsBus\_MotorControlNet, [246](#)
- GetMCSpeedEeprom
  - CMcsBus\_MotorControlNet, [246](#)
- GetMCSpeedShortCommand
  - CMcsBus\_MotorControlNet, [246](#)
- GetMCSpeedUnitEeprom
  - CMcsBus\_MotorControlNet, [246](#)
- GetMCStandbyCurrent
  - CMcsBus\_MotorControlNet, [246](#)
- GetMCStandbyCurrentEeprom
  - CMcsBus\_MotorControlNet, [246](#)
- GetMCStandbyTime
  - CMcsBus\_MotorControlNet, [246](#)
- GetMCStandbyTimeEeprom
  - CMcsBus\_MotorControlNet, [247](#)
- GetMea21UsbPort
  - CMcsUsbNet, [327](#)
- GetMeaLayout
  - CMcsUsbDacqNet, [285](#)
- GetMemoryUsageDAC
  - CStg200xDownloadBasicNet, [564](#)
- GetMemoryUsageSyncout
  - CStg200xDownloadBasicNet, [564](#)
- GetMinimalThreshold
  - CMcsBus\_SensorNet, [260](#)
- GetMinNoPressureWaitTime
  - CRoboDeviceNet::RoboMainLowLevelCommands, [708](#)
- GetMinor
  - DriverVersionNet, [693](#)
- GetMinPressure
  - CRoboDeviceNet, [477](#)
  - CRoboDeviceNet::RoboMainLowLevelCommands, [708](#)
- GetMinPressureWaitTime
  - CRoboDeviceNet::RoboMainLowLevelCommands, [709](#)
- GetMinSamplingFrequencyStepsize
  - CMcsUsbDacqNet, [285](#)

- GetMinVoltage
  - CMeaCleanDeviceNet, [345](#)
- GetModeSelect
  - CPulseGeneratorFunctionNet, [442](#)
- GetModuleCurrent
  - CStg200xDownloadNet, [573](#)
- GetModuleTemp
  - CStg200xDownloadNet, [573](#)
- GetMovementError
  - CRoboDeviceNet, [477](#)
- GetMovePump
  - CMcsBus\_SensorNet, [260](#)
- GetMultiHeadstageMode
  - CW2100\_FunctionNet, [623](#)
- GetMultiplexedDataChannelsInBlock
  - CStimulusFunctionNet, [583](#)
- GetNanoVoltsPerKelvin
  - CMcsBus\_TempSensorNet, [266](#)
- GetNeurochipMemoryData
  - CCMOSMea\_FunctionNet, [111](#)
- GetNeurochipMemorySize
  - CCMOSMea\_FunctionNet, [112](#)
- GetNIC\_MS
  - CRoboDacqNet, [464](#)
- GetNListTrue
  - CMcsUsbDacqNet::ChannelBlockNet, [197](#)
- GetNUC\_MS
  - CRoboDacqNet, [464](#)
- GetNumAmplifications
  - CPgaDeviceNet, [411](#)
- GetNumber
  - CMeaSwitchDeviceNet, [371](#)
  - CSw2to64DeviceNet, [590](#)
- GetNumberOfAnalogChannels
  - CStg200xBasicNet, [537](#)
  - CStimulusFunctionNet, [583](#)
  - CW2100\_StimulatorFunctionNet, [630](#)
- GetNumberOfAudioChannels
  - CMeaAudioFunctionNet, [342](#)
- GetNumberOfAvailableChambers
  - CWarnerUssingFunctionNet, [642](#)
- GetNumberOfAvailableSamples
  - CTEERFunctionNet, [609](#)
- GetNumberOfChannels
  - CDigOutStimulatorFunctionNet, [132](#)
- GetNumberOfCurrentRangeIndexes
  - CStg200xBasicNet, [538](#)
- GetNumberOfDataBits
  - CMcsUsbDacqNet, [285](#)
- GetNumberOfDevices
  - CMcsUsbListNet, [315](#)
- GetNumberOfHardwareSlotsForChambers
  - CWarnerUssingFunctionNet, [642](#)
- GetNumberOfHWADCCChannels
  - CMcsUsbDacqNet::CHWInfo, [216](#)
- GetNumberOfHWDACPaths
  - CStg200xBasicNet, [538](#)
- GetNumberOfHWDigitalChannels
  - CMcsUsbDacqNet::CHWInfo, [216](#)
- GetNumberOfQueues
  - CMcsUsbDacqNet::ChannelBlockNet, [197](#)
- GetNumberOfStimulationElectrodes
  - CStg200xBasicNet, [538](#)
- GetNumberOfStimulationSourcesPerElectrode
  - CStg200xBasicNet, [538](#)
- GetNumberOfSupportedGroups
  - CCMOSMea\_FunctionNet, [112](#)
  - CDacqGroupChannelSelectionTemplateNet< Dac-  
qGroupChannelEnumTemplateNet, Dac-  
qGroupChannelEnumTemplate, CDevice-  
GroupChannelInfoTemplateNet >, [126](#)
- GetNumberOfSyncoutChannels
  - CStg200xBasicNet, [538](#)
  - CW2100\_StimulatorFunctionNet, [630](#)
- GetNumberOfTriggerInputs
  - CStg200xBasicNet, [538](#)
  - CW2100\_StimulatorFunctionNet, [630](#)
- GetNumberOfVoltageRangeIndexes
  - CStg200xBasicNet, [539](#)
- GetNumConfigurations
  - CMcsUsbNet, [327](#)
- GetNumControlChannels
  - CTcxDeviceNet, [598](#)
- GetNumDestinations
  - CMcsUsbFactoryNet, [304](#)
- GetNumDevices
  - CTcxDeviceNet, [598](#)
- GetNumEntries
  - DriverVersionNet, [693](#)
- GetNumFrequencyRanges
  - CPgaDeviceNet, [411](#)
- GetNumMeasureChannels
  - CTcxDeviceNet, [598](#)
- GetNUV\_MS
  - CRoboDacqNet, [464](#)
- GetOffsetCurrent
  - CMeaCoatDeviceNet, [350](#)
- GetOnOff
  - CPositionIIDeviceNet, [415](#)
  - CTcxDeviceNet, [598](#)
- GetOutputCurrent
  - CMeaCoatDeviceNet, [350](#)
- GetOutputRate
  - CStg200xBasicNet, [539](#)
- GetOutputVoltage
  - CMeaCleanDeviceNet, [345](#)
- GetP
  - CTcxDeviceNet, [598](#)
- GetParameter
  - CRoboDeviceNet::RoboMainLowLevelCommands,  
[709](#)
- GetPattern
  - CMeaSwitchDeviceNet, [371](#)
- GetPatternBool
  - CMeaSwitchDeviceNet, [371](#)
- GetPauseDuration



- CMeaCoatDeviceNet, 350
- GetPCoeff
  - CRobo\_FYITemp\_FunctionNet, 455
- GetPDecp
  - CTcxDeviceNet, 598
- GetPeriod
  - CPulseGeneratorFunctionNet, 442
- GetPeriod\_us
  - CTEERFunctionNet, 610
- GetPermanentCurrentInMicroAmp
  - CMultiwellOptoStimFunctionNet, 394
- GetPGain
  - CRoboDacqNet, 464
- GetPhases
  - CRoboDeviceNet::RoboMainLowLevelCommands, 709
- GetPicFirmwareType
  - CW2100\_FunctionNet, 623
- GetPiezoState
  - CMcsBus\_SensorNet, 260
- GetPlateClampLockState
  - CMultiwellDeviceNet, 386
- GetPlateClampState
  - CMultiwellDeviceNet, 386
- GetPlateClampStateByHeadstage
  - CMultiwellCallbackFunctionNet, 383
- GetPlateClampStateByHeadstageEvent
  - CMultiwellCallbackFunctionNet, 383
- GetPlateMux
  - CMultiwellDeviceNet, 386, 387
- GetPlateType
  - CMultiwellDeviceNet, 387
- GetPMax
  - CTcxDeviceNet, 598
- GetPMin
  - CTcxDeviceNet, 598
- GetPoti
  - CMcsUsbDacqNet, 285
- GetPOut
  - CTcxDeviceNet, 598
- GetPowerMuxPlate
  - CMultiwellDeviceNet, 387
- GetPowerStrength
  - CPositionIIDeviceNet, 416
- GetPressure
  - CMcsBus\_SensorNet, 260
- GetPressureOffset
  - CMcsBus\_SensorNet, 261
- GetPressureRange
  - CPPCFunctionNet, 428
- GetPulseform
  - CokuvisionStimulatorDeviceNet, 402
- GetPulseLength
  - CPulseGeneratorFunctionNet, 442
- GetPumpCouple
  - CPPS\_FunctionNet, 435
- GetPumpEnableSpeedRatio
  - CPPS\_FunctionNet, 435
- GetPumpFastOnOff
  - CPPS\_FunctionNet, 435
- GetPumpFastSpeed
  - CPPS\_FunctionNet, 435
- GetPumpFunctionSpeeds
  - CPPS\_FunctionNet, 436
- GetPumpManualOnOff
  - CPPS\_FunctionNet, 436
- GetPumpMaxSpeed
  - CPPS\_FunctionNet, 436
- GetPumpModeType
  - CPPCFunctionNet, 428
  - CPPS\_FunctionNet, 436
- GetPumpSpeed
  - CRoboFluidDeviceNet, 488
- GetPumpSpeedRatio
  - CPPS\_FunctionNet, 436
- GetPumpSpeedUnit
  - CPPCFunctionNet, 429
  - CPPS\_FunctionNet, 436
- GetPWM
  - CFluidControlDeviceNet, 147
- GetPwrOut
  - CTcxDeviceNet, 599
- GetPwrSet
  - CTcxDeviceNet, 599
- GetRatedCapacity
  - CMultiBatteryChargerDeviceNet, 380
- GetReady
  - CMealImpedanceDeviceNet, 369
- GetRecordingNumber
  - CRoboDacqNet, 464
- GetReferenceElectrodeMode
  - CSCUFunctionNet, 514
- GetReferenceElectrodeSwitchState
  - CSCUFunctionNet, 514
- GetReferenceTemperature
  - CFluidControlDeviceNet, 147
- GetRegulationTimeouts
  - CMcsBus\_SensorNet, 261
- GetRegulatorFactor
  - CMcsBus\_SensorNet, 261
- GetRegulatorOnOff
  - CMcsBus\_SensorNet, 261
  - CRobo\_FYITemp\_FunctionNet, 455
- GetRegulatorStatus
  - CMcsBus\_SensorNet, 261
- GetRepeats
  - CProgramPressureCurveNet, 440
- GetRes1
  - CTcxDeviceNet, 599
- GetRes2
  - CTcxDeviceNet, 599
- GetResetFilter
  - CWClassicFunctionNet, 681
- GetResistanceC
  - CRoboDacqNet, 464
- GetResistanceV

- CRoboDacqNet, [464](#)
- GetResolutionPerDigit
  - CMcsUsbDacqNet, [285](#)
- GetResS
  - CTcxDeviceNet, [599](#)
- GetResult
  - CMealImpedanceDeviceNet, [369](#)
- GetResX
  - CTcxDeviceNet, [599](#)
- GetRFConnectionStatus
  - CWClassicFunctionNet, [681](#)
- GetRFFrequency
  - CPositionImpDeviceNet, [422](#)
- GetRFFrequencyHeadstage
  - CWClassicFunctionNet, [681](#)
- GetRFFrequencyReceiver
  - CWClassicFunctionNet, [681](#)
- GetRFPower
  - CWClassicFunctionNet, [682](#)
- GetRoboDacq
  - CRoboocyte2DeviceNet, [492](#)
- GetRoboFluidDevice
  - CEncapsulatorDeviceNet, [135](#)
  - CRoboocyte2DeviceNet, [492](#)
- GetRotaryPositionCode
  - CTEERFunctionNet, [610](#)
- GetRotatePump
  - CMcsBus\_SensorNet, [261](#)
- GetROut
  - CTcxDeviceNet, [599](#)
- GetRTC
  - CokuvisionStimulatorDeviceNet, [403](#)
  - CPositionIIDeviceNet, [416](#)
- GetSampleBufferChunk
  - CTEERFunctionNet, [610](#)
- GetSampleInterval
  - CLIH3DeviceNet, [229](#)
- GetSamplePeriode
  - CMcsBus\_SensorNet, [262](#)
- GetSampleRate
  - CTEERFunctionNet, [610](#)
- GetSamplerate
  - CMcsUsbDacqNet, [286](#)
- GetSampleratePerFramerate
  - CMcsUsbDacqNet, [286](#)
- GetSampleVoltageBuffer\_uV
  - CTEERFunctionNet, [611](#)
- GetScaleFactorU1
  - CTEERFunctionNet, [611](#)
- GetScaleFactorU2
  - CTEERFunctionNet, [611](#)
- GetScanHeadstagesResult
  - CWClassicFunctionNet, [682](#)
- GetScreen
  - CRoboDacqNet, [464](#)
- GetSearchReferenceFastAccel
  - CRoboDeviceNet::RoboMainLowLevelCommands, [709](#)
- GetSearchReferenceFastSpeed
  - CRoboDeviceNet::RoboMainLowLevelCommands, [709](#)
- GetSearchReferenceFineAccel
  - CRoboDeviceNet::RoboMainLowLevelCommands, [709](#)
- GetSearchReferenceFineSpeed
  - CRoboDeviceNet::RoboMainLowLevelCommands, [709](#)
- GetSearchReferenceMethod
  - CRoboDeviceNet::RoboMainLowLevelCommands, [710](#)
- GetSearchReferenceMoveOut
  - CRoboDeviceNet::RoboMainLowLevelCommands, [710](#)
- GetSearchReferenceOffsetPos
  - CRoboDeviceNet::RoboMainLowLevelCommands, [710](#)
- GetSelectedChannels
  - CW2100\_FunctionNet, [624](#)
- GetSelectedHeadstage
  - CWClassicFunctionNet, [682](#)
- GetSelectedHeadstageState
  - CW2100\_FunctionNet, [624](#)
- GetSensorType
  - CTcxDeviceNet, [600](#)
- GetSerialNumber
  - CMcsUsbNet, [327](#)
  - DriverVersionNet, [693](#)
- GetSerialNumberHeadstage
  - CWClassicFunctionNet, [682](#)
- GetSetpoint
  - CTcxDeviceNet, [600](#)
- GetSetpointDecp
  - CTcxDeviceNet, [600](#)
- GetSetpointMax
  - CTcxDeviceNet, [600](#)
- GetSetpointMin
  - CTcxDeviceNet, [600](#)
- GetShortBuffer
  - CGenericDevelopDeviceNet, [162](#)
- GetSimulation
  - CRoboDacqNet, [464](#)
- GetSingleHeater
  - CMcsBus\_FYIExtensionNet, [236](#)
- GetSingleValve
  - CFluidControlDeviceNet, [147](#)
  - CRoboFluidDeviceNet, [488](#)
- GetSlope
  - CMeaCleanDeviceNet, [346](#)
  - CMeaCoatDeviceNet, [350](#)
- GetSoftwareKey
  - CMcsUsbNet, [327](#)
- GetSoftwareKeyString
  - CMcsUsbNet, [327](#)
- GetSolIPressure
  - CMcsBus\_SensorNet, [262](#)
- GetSolITemp

- CRobo\_FYITemp\_FunctionNet, [455](#)
- GetSourceBulk
  - CCMOSMea\_FunctionNet, [112](#)
- GetSourceDrain
  - CCMOSMea\_FunctionNet, [112](#)
- GetSourceGate
  - CCMOSMea\_FunctionNet, [112](#)
- GetStartTriggerSlope
  - CDigOutStimulatorFunctionNet, [132](#)
- GetState
  - CRFFFunctionNet, [451](#)
  - CRobo\_FYIProgram\_FunctionNet, [454](#)
- GetStateDebugData
  - CPositionIIDeviceNet, [416](#)
- GetStateEventData
  - CPositionIIDeviceNet, [417](#)
- GetStatus
  - CMcsUsbNet, [327](#)
  - DriverVersionNet, [694](#)
- GetStatusOfLastCommand
  - CMcsUsbNet, [327](#)
- GetStgProgramInfo
  - CStg200xBasicNet, [539](#)
- GetStgVersionInfo
  - CStg200xBasicNet, [540](#)
- GetStimulationPatternMemory
  - CW2100\_StimulatorFunctionNet, [630](#)
- GetStimulatorStatus
  - CokuvisionStimulatorDeviceNet, [403](#)
- GetStimulusParametersCache
  - CW2100\_FunctionNet, [624](#)
- GetStimulusParametersFromSelectedHS
  - CW2100\_FunctionNet, [624](#)
- GetStimulusSites
  - CCMOSMea\_FunctionNet, [112](#)
- GetStimulusParameters
  - CW2100\_FunctionNet, [624](#)
- GetStopTriggerSlope
  - CDigOutStimulatorFunctionNet, [132](#)
- GetSubChannel
  - CMcsBus\_MotorControlNet, [247](#)
- GetSupplyVoltage
  - CPPCFunctionNet, [429](#)
  - CPPS\_FunctionNet, [436](#)
- GetSweepCount
  - CStg200xDownloadBasicNet, [564](#)
- GetSync
  - CWarnerValveControllerDeviceTesterFunctionNet, [678](#)
- GetSyncoutMap
  - CStg200xBasicNet, [540](#)
- GetSyncState
  - CMcsBus\_SensorNet, [262](#)
- GetTableName
  - CWarnerValveControllerDeviceNet, [659](#)
- GetTableNamebyIndex
  - CWarnerValveControllerDeviceNet, [659](#)
- GetTableNamebyIndexEvent
  - CWarnerValveControllerDeviceNet, [674](#)
- GetTablepointer
  - CRetinaLedDeviceNet, [447](#)
- GetTemperatur
  - CMcsBus\_TempSensorNet, [266](#)
- GetTestMode
  - CRFFFunctionNet, [451](#)
- GetThermocoupleCalibration
  - CFluidControlDeviceNet, [148](#)
- GetThermocoupleNanovoltPerKelvin
  - CFluidControlDeviceNet, [148](#)
- GetThermocoupleReferenceTemp
  - CTcxDeviceNet, [600](#)
- GetThermocoupleTemp
  - CTcxDeviceNet, [601](#)
- GetThermocoupleTempAbs
  - CTcxDeviceNet, [601](#)
- GetThermocoupleTemperature
  - CFluidControlDeviceNet, [148](#)
- GetThermoOffset
  - CMcsBus\_TempSensorNet, [266](#)
- GetThermoTemp
  - CMcsBus\_TempSensorNet, [266](#)
- GetThermoVoltage
  - CMcsBus\_TempSensorNet, [266](#)
- GetTimeInPause
  - CMeaCoatDeviceNet, [350](#)
- GetTimeInPlateau
  - CMeaCoatDeviceNet, [351](#)
- GetTimeResolutionInNanoSeconds
  - CStg200xBasicNet, [540](#)
  - CW2100\_StimulatorFunctionNet, [631](#)
- GetTimeSlot
  - CW2100\_StimulatorFunctionNet, [631](#)
- GetTotalMemory
  - CStg200xBasicNet, [541](#)
  - CStimulusFunctionNet, [583](#)
- GetTotalNumberOfDigitalPorts
  - CWarnerValveControllerDeviceNet, [659](#)
- GetTotalNumberOfTables
  - CWarnerValveControllerDeviceNet, [660](#)
- GetTotalNumberOfValves
  - CWarnerValveControllerDeviceNet, [660](#)
- GetTotalTableSize
  - CWarnerValveControllerDeviceNet, [660](#)
- GetTouchPadEnable
  - CMultiwellDeviceNet, [388](#)
- GetTrigger
  - CStg200xDownloadBasicNet, [565](#)
- GetTriggerSource
  - CStg200xBasicNet, [541](#)
- GetU1Offset
  - CWarnerUssingFunctionNet, [642](#)
- GetU1Reference
  - CWarnerUssingFunctionNet, [643](#)
- GetU2Offset

- CWarnerUssingFunctionNet, 643
- GetU2Reference
  - CWarnerUssingFunctionNet, 643
- GetUByteBuffer
  - CGenericDevelopDeviceNet, 163
- GetUC
  - CRoboDacqNet, 464
- GetUClamp
  - CRoboDacqNet, 465
- GetUCOffset
  - CRoboDacqNet, 465
- GetUIntA
  - CFilterCoefficientsNet, 138
- GetUIntB
  - CFilterCoefficientsNet, 138
- GetUIntBuffer
  - CGenericDevelopDeviceNet, 163
- GetUInts
  - CFilterCoefficientsNet, 138
- GetUnit
  - CTcxDeviceNet, 601
- GetUnitDescription
  - CWarnerUssingFunctionNet, 644
- GetUnitExponent
  - CWarnerUssingFunctionNet, 644
- GetUnitName
  - CWarnerUssingFunctionNet, 644
- GetUnitsPerDigit
  - CWarnerUssingFunctionNet, 645
- GetUOut
  - CTcxDeviceNet, 601
- GetUpdateDisplay
  - CRoboDacqNet, 465
- GetUpperFrequencyByIndex
  - CIntanMea\_FunctionNet, 218
- GetUptimeSeconds
  - CTEERFunctionNet, 611
  - CWarnerUssingFunctionNet, 645
- GetUSBDeviceIDFromFX3Image
  - CMcsUsbFactoryNet, 304
- GetUsbListEntries
  - CMcsUsbListNet, 315
- GetUsbListEntry
  - CMcsUsbListNet, 315
  - CMcsUsbNet, 328
- GetUseBubble
  - CPPS\_FunctionNet, 436
- GetUserCodeFromBitFile
  - CMcsUsbFactoryNet, 304
- GetUserCodeFromFlash
  - CMcsUsbFactoryNet, 304
- GetUserDefinedName
  - CW2100\_FunctionNet, 624
- GetUserDefinedNameCache
  - CW2100\_FunctionNet, 624
- GetUserDefinedNameFromSelectedHS
  - CW2100\_FunctionNet, 625
- GetUserParameter
  - CRoboDeviceNet::RoboMainLowLevelCommands, 710
- GetUShortBuffer
  - CGenericDevelopDeviceNet, 164
- GetUV
  - CRoboDacqNet, 465
- GetUVOffset
  - CRoboDacqNet, 465
- GetValue
  - CTcxDeviceNet, 601
- GetValueHires
  - CTcxDeviceNet, 602
- GetValve
  - CFluidControlDeviceNet, 149
  - CRoboFluidDeviceNet, 488
- GetValve1
  - CRobo\_FYIProgram\_FunctionNet, 454
- GetValve2
  - CRobo\_FYIProgram\_FunctionNet, 454
- GetValveActive
  - CPPCFunctionNet, 429
  - CWarnerValveControllerDeviceNet, 660
- GetValveActiveEvent
  - CWarnerValveControllerDeviceNet, 674
- GetValveBoardRevision
  - CWarnerValveControllerDeviceNet, 660
- GetValveBoardRevisionEvent
  - CWarnerValveControllerDeviceNet, 675
- GetValveBoardRevisionString
  - CWarnerValveControllerDeviceNet, 661
- GetValveCurrent
  - CWarnerValveControllerDeviceNet, 661
- GetValveDigitalInPort
  - CWarnerValveControllerDeviceNet, 661
- GetValveDigitalInPortEvent
  - CWarnerValveControllerDeviceNet, 675
- GetValveLedOn
  - CWarnerValveControllerDeviceNet, 661
- GetValveLedOnEvent
  - CWarnerValveControllerDeviceNet, 675
- GetValveManualGroup
  - CWarnerValveControllerDeviceNet, 662
- GetValveManualGroupEvent
  - CWarnerValveControllerDeviceNet, 675
- GetValveManualState
  - CWarnerValveControllerDeviceNet, 662
- GetValveManualStateEvent
  - CWarnerValveControllerDeviceNet, 675
- GetValveMode
  - CWarnerValveControllerDeviceNet, 662
- GetValveModeEvent
  - CWarnerValveControllerDeviceNet, 675
- GetValves
  - CMcsBus\_FYIExtensionNet, 236
- GetValvesActiveMap
  - CWarnerValveControllerDeviceNet, 663
- GetValvesManualStateMap
  - CWarnerValveControllerDeviceNet, 663

- GetValveTableEntry
  - CWarnerValveControllerDeviceNet, [663](#)
- GetVDD3I
  - CCMOSMea\_FunctionNet, [112](#)
- GetVDDI
  - CCMOSMea\_FunctionNet, [112](#)
- GetVdsVgs
  - CGrapheneFunctionNet, [179](#)
- GetVdVs
  - CGrapheneFunctionNet, [180](#)
- GetVdVsDAC
  - CGrapheneFunctionNet, [180](#), [181](#)
- GetVersion
  - CMcsUsbNet, [328](#)
- GetVersionInt
  - DriverVersionNet, [694](#)
- GetVersionString
  - DriverVersionNet, [695](#)
- GetVMMaxNegativeCurrent
  - CMcsBus\_VoltageModeNet, [268](#)
- GetVMMaxNegativeCurrentEeprom
  - CMcsBus\_VoltageModeNet, [268](#)
- GetVMMaxNegativeVoltage
  - CMcsBus\_VoltageModeNet, [268](#)
- GetVMMaxNegativeVoltageEeprom
  - CMcsBus\_VoltageModeNet, [268](#)
- GetVMMaxPositiveCurrent
  - CMcsBus\_VoltageModeNet, [269](#)
- GetVMMaxPositiveCurrentEeprom
  - CMcsBus\_VoltageModeNet, [269](#)
- GetVMMaxPositiveVoltage
  - CMcsBus\_VoltageModeNet, [269](#)
- GetVMMaxPositiveVoltageEeprom
  - CMcsBus\_VoltageModeNet, [269](#)
- GetVMOutputOnOff
  - CMcsBus\_VoltageModeNet, [269](#)
- GetVMVoltage
  - CMcsBus\_VoltageModeNet, [269](#)
- GetVolatileClampOffset
  - CMultiwellDeviceNet, [388](#)
- GetVoltage
  - COkuvisionStimulatorDeviceNet, [403](#)
- GetVoltage12V
  - CRoboDeviceNet, [477](#)
- GetVoltage12VLimit
  - CRoboDeviceNet, [477](#)
- GetVoltage5V
  - CRoboDeviceNet, [477](#)
- GetVoltage5VLimit
  - CRoboDeviceNet, [477](#)
- GetVoltageAirvalve
  - CRoboDeviceNet, [477](#)
- GetVoltageAirvalveLimit
  - CRoboDeviceNet, [477](#)
- GetVoltageClampControllerParam\_D
  - CWarnerUssingFunctionNet, [645](#)
- GetVoltageClampControllerParam\_I
  - CWarnerUssingFunctionNet, [646](#)
- GetVoltageClampControllerParam\_P
  - CWarnerUssingFunctionNet, [646](#)
- GetVoltageRange
  - CGrapheneFunctionNet, [181](#)
- GetVoltageRangeIndex
  - CMcsUsbDacqNet, [286](#)
- GetVoltageRangeInMicroVolt
  - CMcsUsbDacqNet, [286](#)
  - CStg200xBasicNet, [541](#)
  - CStimulusFunctionNet, [583](#)
  - CW2100\_StimulatorFunctionNet, [631](#)
- GetVoltageRangeInMicroVoltByIndex
  - CStg200xBasicNet, [541](#)
- GetVoltageRangeInMilliVolt
  - CMcsUsbDacqNet, [286](#)
- GetVoltageRangeListInMicroVolt
  - CStg200xBasicNet, [542](#)
- GetVoltageRangeSelectedIndex
  - CStg200xBasicNet, [542](#)
- GetVoltageReached
  - CGrapheneFunctionNet, [181](#)
- GetVoltageResolution
  - CGrapheneFunctionNet, [182](#)
- GetVoltageResolutionInMicroVolt
  - CStg200xBasicNet, [542](#)
  - CStimulusFunctionNet, [584](#)
  - CW2100\_StimulatorFunctionNet, [631](#)
- GetVoltageRs485A
  - CRoboDeviceNet, [478](#)
- GetVoltageRs485ALimit
  - CRoboDeviceNet, [478](#)
- GetVoltageRs485B
  - CRoboDeviceNet, [478](#)
- GetVoltageRs485BLimit
  - CRoboDeviceNet, [478](#)
- GetVoltageValves
  - CRoboDeviceNet, [478](#)
- GetVoltageValvesLimit
  - CRoboDeviceNet, [478](#)
- GetVolti
  - CTcxDeviceNet, [602](#)
- GetWaveform
  - CTEERFunctionNet, [611](#)
- GetWaveLengthInNanometer
  - CMultiwellOptoStimFunctionNet, [394](#)
- GetWorkingFrequency
  - CRFFFunctionNet, [452](#)
- GetWPADebugMode
  - CWClassicFunctionNet, [682](#)
- GetWPAType
  - CWClassicFunctionNet, [682](#)
- GetXGain
  - CRoboDacqNet, [465](#)
- GetXilinxFlashOffset
  - CMcsUsbFactoryNet, [304](#)
- GetXilinxFlashReadCommand
  - CMcsUsbFactoryNet, [305](#)
- GND\_SWITCH\_BIT

- CW2100\_StimulatorFunctionNet, [633](#)
- Graphene\_ASIC
  - Mcs::Usb, [78](#)
- GrapheneASIC
  - Mcs::Usb, [81](#)
- GrapheneASICHeadstage
  - Mcs::Usb, [67](#)
- GrapheneProjectTestDevice
  - Mcs::Usb, [78](#)
- Ground
  - Mcs::Usb, [58](#), [64](#)
- GroupID
  - CDeviceGroupChannelInfoTemplateNet< Dacq-GroupChannelEnumTemplateNet >, [129](#)
- GroupType
  - CDeviceGroupChannelInfoTemplateNet< Dacq-GroupChannelEnumTemplateNet >, [129](#)
- GyroOnly
  - Mcs::Usb, [90](#)
- Hardware
  - Mcs::Usb, [66](#)
- HasAnalogOut
  - CSCUFunctionNet, [514](#)
- HasDigitalPort
  - CStg200xBasicNet, [542](#)
- HasFilterAmp
  - CStg200xBasicNet, [542](#)
- HasGalvanicIsolation
  - CSCUFunctionNet, [515](#)
- HasHSPowerSwitch
  - CSCUFunctionNet, [515](#)
- HasIMU
  - HeadStageIDType, [700](#)
- HasOptoCurrentMeasurement
  - HeadStageIDType, [700](#)
- HasRadioControl
  - CRadioControlledDevicesNet, [445](#)
- HasRef
  - CRoboDeviceNet::RoboMainLowLevelCommands, [711](#)
- HasRefI
  - CRoboStatorDeviceNet, [495](#)
- HasRefXY
  - CRoboStatorDeviceNet, [495](#)
- HasRefZ
  - CRoboStatorDeviceNet, [495](#)
- HasSoftwareKey
  - CMcsUsbNet, [328](#)
- Headstage1NCBathCurrentGroup
  - Mcs::Usb, [61](#)
- Headstage1NCCol2CurrentGroup
  - Mcs::Usb, [61](#)
- Headstage1NChipTempGroup
  - Mcs::Usb, [61](#)
- HeadstageElectrodeGroup
  - Mcs::Usb, [61](#)
- HeadstageIdEnumNet
  - Mcs::Usb, [67](#)
- HeadStageIDType, [698](#)
  - CompareTo, [700](#)
  - ElectricalStimulation, [699](#)
  - Entry, [700](#)
  - Equals, [700](#)
  - HasIMU, [700](#)
  - HasOptoCurrentMeasurement, [700](#)
  - HeadStageIDType, [699](#)
  - HeadstageType, [700](#)
  - HeadstageTypeEnum, [699](#)
  - ID, [700](#)
  - MeasuringOnly, [699](#)
  - NumberOfAnalogChannels, [700](#)
  - NumberOfStimulationChannels, [701](#)
  - OpticalStimulation, [699](#)
  - SN, [701](#)
  - StimulusParameters, [701](#)
  - ToString, [700](#)
  - Type, [701](#)
  - TypeValue, [701](#)
  - Unknown, [699](#)
  - UserDefinedName, [701](#)
  - Valid, [701](#)
  - W16IsW14, [701](#)
- HeadstageIDTypeObject, [701](#)
  - \_AdditionalText, [702](#)
  - \_IdType, [702](#)
  - AdditionalText, [703](#)
  - Equals, [702](#)
  - GetHashCode, [702](#)
  - HeadstageIDTypeObject, [702](#)
  - IdType, [703](#)
  - ToString, [702](#)
- HeadStageIDTypeState, [703](#)
  - ControlState, [703](#)
  - DataState, [703](#)
  - IdType, [703](#)
  - State, [703](#)
- HeadstageType
  - HeadStageIDType, [700](#)
- HeadstageTypeEnum
  - HeadStageIDType, [699](#)
- HEKA\_LIH3\_DEVICE
  - Mcs::Usb, [63](#)
- HekaEPC10Double
  - Mcs::Usb, [76](#)
- HekaEPC10Quadro
  - Mcs::Usb, [76](#)
- HekaEPC10Single
  - Mcs::Usb, [76](#)
- HekaEPC10Triple
  - Mcs::Usb, [76](#)
- HekaEPCLite
  - Mcs::Usb, [76](#)
- HekaITEV100
  - Mcs::Usb, [76](#)
- HekaLIH30
  - Mcs::Usb, [76](#)



HekaLIH408  
Mcs::Usb, [76](#)

HekaLIH816  
Mcs::Usb, [76](#)

HekaPG610  
Mcs::Usb, [76](#)

HekaPG611  
Mcs::Usb, [76](#)

HekaPG612  
Mcs::Usb, [76](#)

HekaPG618  
Mcs::Usb, [76](#)

HekaPG690  
Mcs::Usb, [76](#)

HiClamp  
Mcs::Usb, [78](#)

HiClamp4Uart  
Mcs::Usb, [78](#)

Highpass  
Mcs::Usb, [66](#)

HighSpeed  
Mcs::Usb, [70](#)

HLA  
Mcs::Usb, [76](#)

HLADacq  
CHLADeviceNet, [214](#)

Hs1Digital  
Mcs::Usb, [63](#)

HS1DigitalData1  
Mcs::Usb, [85](#)

HS1ElectrodeGroup  
Mcs::Usb, [70](#)

HS1Sideband1  
Mcs::Usb, [57](#), [71](#), [73](#), [74](#)

HS1Sideband10  
Mcs::Usb, [71](#)

HS1Sideband11  
Mcs::Usb, [71](#)

HS1Sideband12  
Mcs::Usb, [71](#)

HS1Sideband13  
Mcs::Usb, [71](#)

HS1Sideband14  
Mcs::Usb, [71](#)

HS1Sideband15  
Mcs::Usb, [71](#)

HS1Sideband16  
Mcs::Usb, [71](#)

HS1Sideband17  
Mcs::Usb, [71](#)

HS1Sideband18  
Mcs::Usb, [71](#)

HS1Sideband2  
Mcs::Usb, [57](#), [71](#), [73](#), [74](#)

HS1Sideband3  
Mcs::Usb, [57](#), [71](#), [73](#)

HS1Sideband4  
Mcs::Usb, [71](#), [73](#)

HS1Sideband5  
Mcs::Usb, [71](#), [73](#)

HS1Sideband6  
Mcs::Usb, [71](#), [73](#)

HS1Sideband7  
Mcs::Usb, [71](#)

HS1Sideband8  
Mcs::Usb, [71](#)

HS1Sideband9  
Mcs::Usb, [71](#)

Hs1SidebandHigh  
Mcs::Usb, [63](#)

Hs1SidebandLow  
Mcs::Usb, [63](#)

Hs1Trigger  
Mcs::Usb, [63](#)

HS1Trigger10Status  
Mcs::Usb, [71](#)

HS1Trigger11Status  
Mcs::Usb, [71](#)

HS1Trigger12Status  
Mcs::Usb, [71](#)

HS1Trigger13Status  
Mcs::Usb, [71](#)

HS1Trigger14Status  
Mcs::Usb, [71](#)

HS1Trigger15Status  
Mcs::Usb, [71](#)

HS1Trigger16Status  
Mcs::Usb, [71](#)

HS1Trigger17Status  
Mcs::Usb, [71](#)

HS1Trigger18Status  
Mcs::Usb, [71](#)

HS1Trigger1Status  
Mcs::Usb, [57](#), [71](#), [72](#), [74](#)

HS1Trigger2Status  
Mcs::Usb, [57](#), [71](#), [72](#), [74](#)

HS1Trigger3Status  
Mcs::Usb, [57](#), [71](#), [72](#)

HS1Trigger4Status  
Mcs::Usb, [71](#), [72](#)

HS1Trigger5Status  
Mcs::Usb, [71](#), [72](#)

HS1Trigger6Status  
Mcs::Usb, [71](#), [73](#)

HS1Trigger7Status  
Mcs::Usb, [71](#)

HS1Trigger8Status  
Mcs::Usb, [71](#)

HS1Trigger9Status  
Mcs::Usb, [71](#)

Hs2Digital  
Mcs::Usb, [63](#)

HS2DigitalData1  
Mcs::Usb, [85](#)

HS2ElectrodeGroup  
Mcs::Usb, [70](#)

- HS2Sideband1
  - Mcs::Usb, [72–74](#)
- HS2Sideband10
  - Mcs::Usb, [72](#)
- HS2Sideband11
  - Mcs::Usb, [72](#)
- HS2Sideband12
  - Mcs::Usb, [72](#)
- HS2Sideband13
  - Mcs::Usb, [72](#)
- HS2Sideband14
  - Mcs::Usb, [72](#)
- HS2Sideband15
  - Mcs::Usb, [72](#)
- HS2Sideband16
  - Mcs::Usb, [72](#)
- HS2Sideband17
  - Mcs::Usb, [72](#)
- HS2Sideband18
  - Mcs::Usb, [72](#)
- HS2Sideband2
  - Mcs::Usb, [72–74](#)
- HS2Sideband3
  - Mcs::Usb, [72, 73](#)
- HS2Sideband4
  - Mcs::Usb, [72, 73](#)
- HS2Sideband5
  - Mcs::Usb, [72, 73](#)
- HS2Sideband6
  - Mcs::Usb, [72, 73](#)
- HS2Sideband7
  - Mcs::Usb, [72](#)
- HS2Sideband8
  - Mcs::Usb, [72](#)
- HS2Sideband9
  - Mcs::Usb, [72](#)
- Hs2SidebandHigh
  - Mcs::Usb, [63](#)
- Hs2SidebandLow
  - Mcs::Usb, [63](#)
- Hs2Trigger
  - Mcs::Usb, [63](#)
- HS2Trigger10Status
  - Mcs::Usb, [71](#)
- HS2Trigger11Status
  - Mcs::Usb, [72](#)
- HS2Trigger12Status
  - Mcs::Usb, [72](#)
- HS2Trigger13Status
  - Mcs::Usb, [72](#)
- HS2Trigger14Status
  - Mcs::Usb, [72](#)
- HS2Trigger15Status
  - Mcs::Usb, [72](#)
- HS2Trigger16Status
  - Mcs::Usb, [72](#)
- HS2Trigger17Status
  - Mcs::Usb, [72](#)
- HS2Trigger18Status
  - Mcs::Usb, [72](#)
- HS2Trigger1Status
  - Mcs::Usb, [71, 73, 74](#)
- HS2Trigger2Status
  - Mcs::Usb, [71, 73, 74](#)
- HS2Trigger3Status
  - Mcs::Usb, [71, 73](#)
- HS2Trigger4Status
  - Mcs::Usb, [71, 73](#)
- HS2Trigger5Status
  - Mcs::Usb, [71, 73](#)
- HS2Trigger6Status
  - Mcs::Usb, [71, 73](#)
- HS2Trigger7Status
  - Mcs::Usb, [71](#)
- HS2Trigger8Status
  - Mcs::Usb, [71](#)
- HS2Trigger9Status
  - Mcs::Usb, [71](#)
- HWInfo
  - CMcsUsbDacqNet, [287](#)
- HwVersion
  - CMcsUsbListEntryNet, [313](#)
- ID
  - HeadStageIDType, [700](#)
- Idle
  - Mcs::Usb, [84](#)
- IdProduct
  - DeviceIdNet, [686](#)
- IdType
  - HeadstageIDTypeObject, [703](#)
  - HeadStageIDTypeState, [703](#)
- IdVendor
  - DeviceIdNet, [686](#)
- IFB2GoldenInterfaceboard
  - Mcs::Usb, [67](#)
- IFB30GoldenInterfaceboard
  - Mcs::Usb, [67](#)
- IFChannel1
  - Mcs::Usb, [59](#)
- IFChannel2
  - Mcs::Usb, [59](#)
- IFChannel3
  - Mcs::Usb, [59](#)
- IFChannel4
  - Mcs::Usb, [60](#)
- IFChannel5
  - Mcs::Usb, [60](#)
- IFChannel6
  - Mcs::Usb, [60](#)
- IFChannel7
  - Mcs::Usb, [60](#)
- IFChannel8
  - Mcs::Usb, [60](#)
- IFDigChannelsGroup
  - Mcs::Usb, [61, 70, 81, 92](#)
- Ignore



- Mcs::Usb, [80](#)
- IM16KRC
  - Mcs::Usb, [78](#)
- IM16S16KRA
  - Mcs::Usb, [78](#)
- IM16S8KRA
  - Mcs::Usb, [78](#)
- IM64KRB
  - Mcs::Usb, [78](#)
- IM64KRC
  - Mcs::Usb, [78](#)
- Init
  - CMcsUsbDacqNet::ChannelBlockNet, [197](#)
- Input
  - Mcs::Usb, [75](#)
- Intel
  - Mcs::Usb, [89](#)
- InterfaceADCGroup
  - Mcs::Usb, [61](#), [70](#), [81](#), [90](#)
- InterfaceBoard2
  - Mcs::Usb, [67](#)
- IntToDouble
  - Mcs::Usb, [66](#)
- InvitroSignalCollectorUnit
  - Mcs::Usb, [68](#)
- InvivoSignalCollectorUnit
  - Mcs::Usb, [68](#)
- IoVoltageEnumNet
  - Mcs::Usb, [68](#)
- IS32KRA
  - Mcs::Usb, [78](#)
- IsAnalogOutEnabled
  - CSCUFunctionNet, [515](#)
- IsAutomaticAnalogOut
  - CSCUFunctionNet, [515](#)
- IsBusy
  - CPPCFunctionNet, [429](#)
- IsChamberAvailable
  - CWarnerUssingFunctionNet, [646](#)
- IsChipPowered
  - CCMOSMea\_FunctionNet, [112](#)
- IsConnected
  - CMcsUsbNet, [328](#)
- IsDeviceHighSpeed
  - CMcsUsbNet, [328](#)
- IsDeviceHighSpeedCapable
  - CMcsUsbNet, [328](#)
- IsDeviceTypeOf
  - CMcsUsbListNet, [316](#)
- IsDigitalChannelDedicated
  - CMcsUsbDacqNet::CHWInfo, [216](#)
- IsDigitalOutPortInverted
  - CWarnerValveControllerDeviceNet, [663](#)
- IsDigitalOutPortInvertedEvent
  - CWarnerValveControllerDeviceNet, [675](#)
- IsEqual
  - CFilterCoefficientsNet, [138](#)
- IsExceptionsEnabled
- CMcsUsbNet, [329](#)
- IsGateFloating
  - CCMOSMea\_FunctionNet, [112](#)
- IsHeadstageAvailable
  - CSCUFunctionNet, [515](#)
- IsHeadstageAvailableEvent
  - CSCUFunctionNet, [520](#)
- IsHighCurrentMode
  - CWarnerUssingFunctionNet, [647](#)
- IsHSPowered
  - CSCUFunctionNet, [516](#)
- IsInDacqLegacyMode
  - CSCUFunctionNet, [516](#)
- IsInternalCalibrationFinished
  - CTEERFunctionNet, [612](#)
  - CWarnerUssingFunctionNet, [647](#)
- IsPlateTypeValid
  - CMultiwellDeviceNet, [388](#)
- IsPulseEnabled
  - CWarnerUssingFunctionNet, [647](#)
- IsPumpMotorOn
  - CRoboFluidDeviceNet, [489](#)
- IsQueueEnabled
  - CRoboDeviceNet, [478](#)
- IsQueueStarted
  - CRoboDeviceNet, [478](#)
- IsRunning
  - CMeaCleanDeviceNet, [346](#)
  - CMeaCoatDeviceNet, [351](#)
- IsSamplingFinished
  - CTEERFunctionNet, [612](#)
- IsUserTriggerEnabled
  - CLIH3DeviceNet, [229](#)
- IsValveDigitalInInverted
  - CWarnerValveControllerDeviceNet, [664](#)
- IsValveDigitalInInvertedEvent
  - CWarnerValveControllerDeviceNet, [675](#)
- IsValveOpen
  - CWarnerValveControllerDeviceNet, [664](#)
- IsValveOpenEvent
  - CWarnerValveControllerDeviceNet, [676](#)
- IsValveOpenInAnalogMode
  - CWarnerValveControllerDeviceNet, [664](#)
- IsValveOpenInAnalogModeEvent
  - CWarnerValveControllerDeviceNet, [676](#)
- IsValveOpenInDigitalMode
  - CWarnerValveControllerDeviceNet, [665](#)
- IsValveOpenInDigitalModeEvent
  - CWarnerValveControllerDeviceNet, [676](#)
- Kelvin
  - Mcs::Usb, [53](#)
- LastPosition
  - Mcs::Usb, [57](#), [72–74](#), [83](#), [85](#), [92](#)
- LegacyHasOutputMap
  - CStg200xBasicNet, [543](#)
- LegacyMeaUsb
  - Mcs::Usb, [76](#)

- LIH30\_EPC10\_Bus\_EnumNet
  - Mcs::Usb, [68](#)
- LIH30ADCCtrl
  - Mcs::Usb, [67](#), [68](#)
- LIH30ADCModulesGroup
  - Mcs::Usb, [61](#)
- LIH30Interfaceboard
  - Mcs::Usb, [67](#)
- ListModeSendStart
  - CStg200xBasicNet, [543](#)
- ListModeSendStop
  - CStg200xBasicNet, [543](#)
- ListOfChangedTriggers
  - StgStatusNet, [717](#)
- LoadPressure
  - CPPCFunctionNet, [431](#)
- LoadUserFirmware
  - CMcsUsbFactoryNet, [305](#)
- LoadValveTable
  - CWarnerValveControllerDeviceNet, [665](#)
- Lock
  - Mcs::Usb, [75](#)
- LockPlateClamp
  - CMultiwellDeviceNet, [389](#)
- Lowpass
  - Mcs::Usb, [66](#)
- LowSpeed
  - Mcs::Usb, [70](#)
- M4M\_Dock
  - Mcs::Usb, [67](#), [78](#)
- M4M\_ReUse
  - Mcs::Usb, [67](#)
- m\_Bottom
  - CCMOSMeaDeviceNet::CRegionOfInterestRect, [446](#)
- m\_Left
  - CCMOSMeaDeviceNet::CRegionOfInterestRect, [446](#)
- m\_pGilsonDevice
  - CGilsonDeviceNet, [172](#)
- m\_pMcsBus\_MotorControlNet
  - CRoboFluidDeviceNet, [490](#)
- m\_pMcsUsb
  - CMcsUsbFunctionNet, [310](#)
- m\_pMcsUsbFunction
  - CMcsUsbFunctionNet, [310](#)
- m\_pRoboFluidDevice
  - CRoboFluidDeviceNet, [490](#)
- m\_Right
  - CCMOSMeaDeviceNet::CRegionOfInterestRect, [446](#)
- m\_Top
  - CCMOSMeaDeviceNet::CRegionOfInterestRect, [446](#)
- Manual
  - Mcs::Usb, [75](#), [83](#), [93](#)
- Manufacturer
  - CMcsUsbListEntryNet, [313](#)
- MaxBitNumber
  - DigitalSource< digitalsourceenum >, [687](#)
  - DigitalSourceGeneral, [688](#), [689](#)
- MaxBitNumberStatic
  - DigitalSource< digitalsourceenum >, [687](#)
- MBC08
  - Mcs::Usb, [77](#)
- MbcChannelStateEnumNet
  - Mcs::Usb, [68](#)
- MbcChargingModeEnumNet
  - Mcs::Usb, [69](#)
- MbcRatedCapacityEnumNet
  - Mcs::Usb, [69](#)
- MC\_Card
  - Mcs::Usb, [76](#)
- MCS
  - Mcs::Usb, [90](#)
- Mcs, [23](#)
- Mcs::Usb, [23](#)
  - A, [68](#)
  - AccelOnly, [90](#)
  - AdapterTypeEnumNet, [52](#)
  - ALA\_VC3, [76](#), [90](#)
  - ALA\_VC3\_DEVICE, [63](#)
  - ALTERA, [55](#)
  - ALTERA\_BASE, [56](#)
  - ALTERA\_BOOTSTRAP, [56](#)
  - ALTERA\_GOLD, [55](#)
  - ALTERA\_TARGET1, [57](#)
  - ALTERA\_TARGET2, [57](#)
  - ALTERA\_TARGET3, [57](#)
  - AlwaysOn, [79](#)
  - Ampere, [53](#), [90](#)
  - AMS\_Dongle, [77](#)
  - Analog, [76](#), [93](#)
  - AnalogGroup, [61](#)
  - AnalogOut\_DAC\_Range\_EnumNet, [53](#)
  - AnalogSource\_HS1, [53](#)
  - AnalogSource\_HS2, [53](#)
  - AnalogSource\_IF, [53](#)
  - AnalogSourceEnumNet, [53](#)
  - AnalogUnitEnumNet, [53](#)
  - Any, [76](#), [90](#)
  - Armed, [84](#)
  - ASMedia, [89](#)
  - AudioTestChannelGroup, [61](#), [70](#), [81](#), [92](#)
  - Aux, [63](#)
  - AuxIn, [57](#), [70](#), [72](#), [74](#), [82](#), [85](#), [92](#)
  - AuxPort, [59](#)
  - B, [68](#)
  - Bessel, [66](#)
  - BMI, [86](#)
  - Bootstrap, [55](#)
  - BootstrapOtherCypress, [55](#)
  - Both, [90](#)
  - Break, [80](#)
  - BUS0MCSBUS0, [54](#)
  - BUS0MCSBUS1, [54](#)

BUS0MCSBUS10, [54](#)  
BUS0MCSBUS11, [54](#)  
BUS0MCSBUS12, [54](#)  
BUS0MCSBUS13, [54](#)  
BUS0MCSBUS14, [54](#)  
BUS0MCSBUS15, [54](#)  
BUS0MCSBUS2, [54](#)  
BUS0MCSBUS3, [54](#)  
BUS0MCSBUS4, [54](#)  
BUS0MCSBUS5, [54](#)  
BUS0MCSBUS6, [54](#)  
BUS0MCSBUS7, [54](#)  
BUS0MCSBUS8, [54](#)  
BUS0MCSBUS9, [54](#)  
BUS1MCSBUS0, [54](#)  
BUS1MCSBUS1, [54](#)  
BUS1MCSBUS10, [54](#)  
BUS1MCSBUS11, [54](#)  
BUS1MCSBUS12, [54](#)  
BUS1MCSBUS13, [54](#)  
BUS1MCSBUS14, [54](#)  
BUS1MCSBUS15, [54](#)  
BUS1MCSBUS2, [54](#)  
BUS1MCSBUS3, [54](#)  
BUS1MCSBUS4, [54](#)  
BUS1MCSBUS5, [54](#)  
BUS1MCSBUS6, [54](#)  
BUS1MCSBUS7, [54](#)  
BUS1MCSBUS8, [54](#)  
BUS1MCSBUS9, [54](#)  
BUS2MCSBUS0, [55](#)  
BUS2MCSBUS1, [54](#)  
BUS2MCSBUS10, [55](#)  
BUS2MCSBUS11, [55](#)  
BUS2MCSBUS12, [55](#)  
BUS2MCSBUS13, [55](#)  
BUS2MCSBUS14, [55](#)  
BUS2MCSBUS15, [55](#)  
BUS2MCSBUS2, [55](#)  
BUS2MCSBUS3, [55](#)  
BUS2MCSBUS4, [55](#)  
BUS2MCSBUS5, [55](#)  
BUS2MCSBUS6, [55](#)  
BUS2MCSBUS7, [55](#)  
BUS2MCSBUS8, [55](#)  
BUS2MCSBUS9, [55](#)  
BUSNUMBER0, [54](#)  
BUSNUMBER1, [54](#)  
BUSNUMBER2, [54](#)  
Butterworth, [66](#)  
Campden\_Ci4600EphysVideoDataIntegrator, [76](#)  
CatchAmp, [75](#)  
CFirmwareDestinationNet, [53](#)  
ChannelBlockTypeNet, [57](#)  
channeldata\_current, [85](#)  
channeldata\_current\_always\_boost, [85](#)  
channeldata\_current\_always\_boost\_own\_sync, [85](#)  
channeldata\_current\_own\_boost\_gnd\_sync, [85](#)  
channeldata\_current\_own\_sync, [85](#)  
channeldata\_positive\_current, [85](#)  
channeldata\_positive\_current\_own\_boost\_gnd\_sync, [85](#)  
channeldata\_positive\_current\_own\_sync, [85](#)  
channeldata\_positive\_voltage, [85](#)  
channeldata\_voltage, [85](#)  
ChannelPIC, [55](#)  
ChannelTest, [77](#)  
ChecksumAndPacketCounter, [60](#)  
Ci4600Intan, [52](#)  
ClampModeCurrent, [86](#)  
ClampModeInternalCalibration, [86](#)  
ClampModeOpen, [86](#)  
ClampModeVoltage, [86](#)  
Close, [75](#)  
CmosMea, [68](#)  
CMOSMEA5000DigitalSourceEnumNet, [57](#)  
CMOSMeaBathModeEnumNet, [58](#)  
CmosMeaHeadstage, [67](#)  
CMOSMeaHeadstage1NCBathCurrentEnumNet, [58](#)  
CMOSMeaHeadstage1NCCol2CurrentEnumNet, [58](#)  
CMOSMeaHeadstage1NChipTempEnumNet, [58](#)  
CMOSMeaHS1SidebandEnumNet, [58](#)  
CMOSMeaHS1TriggerStatusEnumNet, [59](#)  
CmosmealFB2, [67](#)  
CMOSMealFDigChannelEnumNet, [59](#)  
CMOSMealInterfaceADCEnumNet, [59](#)  
CmosMealInterfaceboard, [67](#)  
CMOSMeaPacketFrameContextGroupEnumNet, [60](#)  
CMOSMeaSTG1DACSignalEnumNet, [60](#)  
CMOSMeaValueUnitEnumNet, [60](#)  
CommaPositionA, [65](#)  
CommaPositionB, [65](#)  
csCapacityTestDischarge, [68](#)  
csCapacityTestPrecharge, [68](#)  
csCharge, [68](#)  
csDischarge, [69](#)  
csError, [69](#)  
csIdleChargeFinished, [68](#)  
csIdleNoBattery, [68](#)  
csRefreshBattery, [68](#)  
CurrentClamp, [89](#)  
CurrentMeasure, [58](#)  
Cypress, [90](#)  
Cypress\_FX1, [76](#)  
Cypress\_FX2, [76](#)  
Cypress\_FX3, [76](#)  
DAC1Channel, [60](#)  
DAC2Channel, [60](#)  
DAC3Channel, [60](#)  
DAC4Channel, [60](#)  
DACQ1DigitalGroup, [61](#)  
DacqGroupChannelEnumNet, [61](#)  
DacqMeaGroupTypeEnumNet, [61](#)

DacqTrigger, 64  
DataModeEnumNet, 61  
DEST\_FX3\_TARGET\_MASK, 56  
DEST\_TARGET1, 56  
DEST\_TARGET10, 56  
DEST\_TARGET11, 56  
DEST\_TARGET12, 56  
DEST\_TARGET13, 56  
DEST\_TARGET14, 56  
DEST\_TARGET15, 56  
DEST\_TARGET2, 56  
DEST\_TARGET3, 56  
DEST\_TARGET4, 56  
DEST\_TARGET5, 56  
DEST\_TARGET6, 56  
DEST\_TARGET7, 56  
DEST\_TARGET8, 56  
DEST\_TARGET9, 56  
DEST\_TARGET\_MASK, 56  
DEVICE\_NOT\_FOUND, 65  
DeviceEnumNet, 62  
DeviceHasNoHeadstage, 67, 68, 81  
DeviceNotConnected, 66, 68, 81  
DeviceRunStatus, 57, 70, 73, 74, 82, 85, 92  
DigDataFromReceiver, 92  
Digital, 76, 93  
DigitalData, 57, 71, 73, 74, 82, 85, 92  
DigitalDatastreamEnableEnumNet, 63  
DigitalGroup, 61  
DigitalIn, 57, 63, 70, 72, 73, 82, 85, 92  
DigitalInOfOutPort, 57, 70, 72, 73, 82, 85, 92  
DigitalInPort, 59  
DigitalInReserverd, 63  
DigitalMux, 59  
DigitalOut, 63  
DigitalOutReg, 59  
DigitalOutReserved, 63  
DigitalOutStimulator, 57, 71, 73, 74, 82, 85, 92  
DigitalPulse, 57, 70, 72, 73, 82, 85, 92  
DigitalReg, 59  
DigitalStimulatorTriggerEventEnumNet, 63  
DigitalStimulatorTriggerSlopeEnumNet, 64  
DigitalTargetEnumNet, 64  
Digout, 64  
DigOutStim, 63  
DigOutStimulatorStartTrigger, 64  
DigOutStimulatorStopTrigger, 64  
Digstream, 64  
DigStreamFromReceiver, 92  
DigStreamToReceiver, 64  
Dilutor, 78  
DongleS, 77  
Dotriapot, 76  
DoubleToInt, 66  
DownloadOnly, 84  
DSP, 53  
DSPAnalogGroup, 70, 81  
DSPDataGroup, 61, 91  
DSPDigitalGroup, 70, 81  
eCube, 78, 81  
eCubeHeadstage, 67  
ElectrodeDacMuxEnumNet, 64  
ElectrodeModeEnumNet, 65  
ElectrodeOffset, 89  
emAutomatic, 65  
emManual, 65  
Encapsulator, 78  
enCMosMeaChipType, 65  
EnSTG200x\_STATUS, 65  
EOAndCRC, 60  
ExtendedMultiFileMode, 83  
ExternBCTester, 77  
ExternDTester, 77  
ExternSTester, 77  
Falling, 64  
FCB, 77  
FCX, 77  
Feedback, 57, 70, 72, 74, 82, 85, 92  
FeedbackHigh, 63  
FeedbackLow, 63  
FeedbackReg, 59  
FilterAttributeEnumNet, 65  
FilterBandEnumNet, 66  
FilterCalculationDirectionEnumNet, 66  
FilterFamilyEnumNet, 66  
FilterTypeEnumNet, 66  
Finished, 84  
FPGA10, 55  
FPGA10\_BASE, 56  
FPGA10\_GOLD, 56  
FPGA11, 55  
FPGA11\_BASE, 56  
FPGA11\_GOLD, 56  
FPGA12, 55  
FPGA12\_BASE, 56  
FPGA12\_GOLD, 56  
FPGA13, 55  
FPGA13\_BASE, 56  
FPGA13\_GOLD, 56  
FPGA14, 55  
FPGA14\_BASE, 56  
FPGA14\_GOLD, 56  
FPGA15, 55  
FPGA15\_BASE, 56  
FPGA15\_GOLD, 56  
FPGA16, 55  
FPGA16\_BASE, 56  
FPGA16\_GOLD, 56  
FPGA2, 55  
FPGA2\_BASE, 56  
FPGA2\_GOLD, 55  
FPGA3, 55  
FPGA3\_BASE, 56  
FPGA3\_GOLD, 55  
FPGA4, 55  
FPGA4\_BASE, 56

FPGA4\_GOLD, 56  
FPGA5, 55  
FPGA5\_BASE, 56  
FPGA5\_GOLD, 56  
FPGA6, 55  
FPGA6\_BASE, 56  
FPGA6\_GOLD, 56  
FPGA7, 55  
FPGA7\_BASE, 56  
FPGA7\_GOLD, 56  
FPGA8, 55  
FPGA8\_BASE, 56  
FPGA8\_GOLD, 56  
FPGA9, 55  
FPGA9\_BASE, 56  
FPGA9\_GOLD, 56  
FPGA\_BASE, 56  
FPGA\_BOOTSTRAP, 56  
FPGA\_GOLD, 55  
FPGA\_NORMAL, 53  
FpgaldEnumNet, 66  
FrameContextGroup, 61  
FullCharge, 69  
FullSpeed, 70  
FunkDongleS, 77  
Gain100, 84  
Gain1000, 84  
Gain200, 84  
Gain500, 84  
Gate, 80  
Gated\_High\_Active, 79  
Gated\_Low\_Active, 79  
GE2100, 78  
Generic, 77  
Graphene\_ASIC, 78  
GrapheneASIC, 81  
GrapheneASICHeadstage, 67  
GrapheneProjectTestDevice, 78  
Ground, 58, 64  
GyroOnly, 90  
Hardware, 66  
Headstage1NCBathCurrentGroup, 61  
Headstage1NCCol2CurrentGroup, 61  
Headstage1NChipTempGroup, 61  
HeadstageElectrodeGroup, 61  
HeadstageIdEnumNet, 67  
HEKA\_LIH3\_DEVICE, 63  
HekaEPC10Double, 76  
HekaEPC10Quadro, 76  
HekaEPC10Single, 76  
HekaEPC10Triple, 76  
HekaEPCLite, 76  
HekaITEV100, 76  
HekaLIH30, 76  
HekaLIH408, 76  
HekaLIH816, 76  
HekaPG610, 76  
HekaPG611, 76  
HekaPG612, 76  
HekaPG618, 76  
HekaPG690, 76  
HiClamp, 78  
HiClamp4Uart, 78  
Highpass, 66  
HighSpeed, 70  
HLA, 76  
Hs1Digital, 63  
HS1DigitalData1, 85  
HS1ElectrodeGroup, 70  
HS1Sideband1, 57, 71, 73, 74  
HS1Sideband10, 71  
HS1Sideband11, 71  
HS1Sideband12, 71  
HS1Sideband13, 71  
HS1Sideband14, 71  
HS1Sideband15, 71  
HS1Sideband16, 71  
HS1Sideband17, 71  
HS1Sideband18, 71  
HS1Sideband2, 57, 71, 73, 74  
HS1Sideband3, 57, 71, 73  
HS1Sideband4, 71, 73  
HS1Sideband5, 71, 73  
HS1Sideband6, 71, 73  
HS1Sideband7, 71  
HS1Sideband8, 71  
HS1Sideband9, 71  
Hs1SidebandHigh, 63  
Hs1SidebandLow, 63  
Hs1Trigger, 63  
HS1Trigger10Status, 71  
HS1Trigger11Status, 71  
HS1Trigger12Status, 71  
HS1Trigger13Status, 71  
HS1Trigger14Status, 71  
HS1Trigger15Status, 71  
HS1Trigger16Status, 71  
HS1Trigger17Status, 71  
HS1Trigger18Status, 71  
HS1Trigger1Status, 57, 71, 72, 74  
HS1Trigger2Status, 57, 71, 72, 74  
HS1Trigger3Status, 57, 71, 72  
HS1Trigger4Status, 71, 72  
HS1Trigger5Status, 71, 72  
HS1Trigger6Status, 71, 73  
HS1Trigger7Status, 71  
HS1Trigger8Status, 71  
HS1Trigger9Status, 71  
Hs2Digital, 63  
HS2DigitalData1, 85  
HS2ElectrodeGroup, 70  
HS2Sideband1, 72–74  
HS2Sideband10, 72  
HS2Sideband11, 72  
HS2Sideband12, 72  
HS2Sideband13, 72





MCS\_PPS5\_DEVICE, [62](#)  
MCS\_PPS\_DEVICE, [62](#)  
MCS\_RETINA\_AMS\_DONGLE, [62](#)  
MCS\_RETINA\_LED\_DEVICE, [62](#)  
MCS\_ROBO\_DEVICE, [62](#)  
MCS\_ROBOINJECT\_DEVICE, [62](#)  
MCS\_ROBOOCYTE2\_DEVICE, [62](#)  
MCS\_SAFEIS\_DEVICE, [62](#)  
MCS\_SMARTIMPLANT\_DEVICE, [62](#)  
MCS\_SOFTWARE\_DONGLE\_DEVICE, [62](#)  
MCS\_STG\_DEVICE, [62](#)  
MCS\_SW2TO64\_DEVICE, [62](#)  
MCS\_TCX\_DEVICE, [62](#)  
MCS\_TERSENS\_DEVICE, [62](#)  
MCS\_UNDEFINED\_BUS, [69](#)  
MCS\_USB\_BUS, [69](#)  
MCSBUS0, [54](#)  
MCSBUS1, [53](#)  
MCSBUS10, [54](#)  
MCSBUS11, [54](#)  
MCSBUS12, [54](#)  
MCSBUS13, [54](#)  
MCSBUS14, [54](#)  
MCSBUS15, [54](#)  
MCSBUS2, [53](#)  
MCSBUS3, [54](#)  
MCSBUS4, [54](#)  
MCSBUS5, [54](#)  
MCSBUS6, [54](#)  
MCSBUS7, [54](#)  
MCSBUS8, [54](#)  
MCSBUS9, [54](#)  
McsBusTypeEnumNet, [69](#)  
McsUsbSpeedEnumNet, [69](#)  
MCU1, [53](#)  
ME128, [77](#)  
ME16, [77](#)  
ME2100, [78](#)  
Me2100\_32PICI40, [81](#)  
Me2100\_32PICI40Headstage, [67](#)  
Me2100\_32Xilinx, [81](#)  
Me2100\_32XilinxHeadstage, [67](#)  
Me2100Graphene16\_32, [81](#)  
Me2100Graphene16\_32Headstage, [67](#)  
Me2100Interfaceboard, [67](#)  
Me2100InvitroSignalCollectorUnit, [67](#)  
Me2100InvivoSignalCollectorUnit, [67](#)  
Me2100UPA32, [81](#)  
Me2100UPA32Headstage, [67](#)  
ME256, [77](#)  
ME32, [77](#)  
ME64, [77](#)  
MEA1060, [77](#)  
MEA120, [52](#)  
MEA2100, [77](#)  
Mea2100, [68](#)  
MEA2100\_256, [78](#)  
Mea2100\_256, [68](#)  
MEA2100\_256DacqGroupChannelEnumNet, [70](#)  
MEA2100\_256DigitalSourceEnumNet, [70](#)  
Mea2100\_256Headstage, [67](#)  
Mea2100\_256Interfaceboard, [67](#)  
MEA2100\_32, [77](#)  
MEA2100\_Lite, [78](#)  
Mea2100\_Lite, [68](#)  
MEA2100\_Mini, [78](#)  
MEA2100\_Mini\_Usb\_develop, [77](#)  
MEA2100BetaScreen, [78](#)  
Mea2100BetaScreen, [81](#)  
Mea2100BetaScreenHeadstage, [67](#)  
MEA2100DigitalSourceEnumNet, [72](#)  
Mea2100Headstage, [67](#)  
Mea2100Interfaceboard, [67](#)  
Mea2100LiteHeadstage, [67](#)  
Mea2100Mini120, [81](#)  
Mea2100Mini120Headstage, [67](#)  
Mea2100Mini60ECP5, [81](#)  
Mea2100Mini60ECP5Headstage, [67](#)  
Mea2100Mini60PICI40, [81](#)  
Mea2100Mini60PICI40Headstage, [67](#)  
Mea2100MultiwellIFB2, [67](#)  
Mea2100STG, [67](#)  
MEA252, [52](#)  
MEA256, [77](#)  
MEA2x32, [52](#)  
MEA2x60, [52](#)  
MEA32, [52](#)  
MEA60, [52](#)  
MEA\_2\_252\_2, [52](#)  
MEA\_2\_252\_2\_6Well, [52](#)  
MEA\_2\_252\_2\_9Well, [52](#)  
MEA\_2\_252\_2\_Test, [52](#)  
MEA\_Clean, [77](#)  
MEA\_Coat, [77](#)  
MEA\_Impedance, [77](#)  
MEA\_Sanofi, [77](#)  
MEA\_Switch, [77](#)  
MEA\_Switch\_2\_1, [77](#)  
MEA\_Switch\_4\_2, [77](#)  
MeaLayoutEnumNet, [73](#)  
MeasureTable, [78](#)  
MicroAmpere, [60](#)  
MilliDegreeCelsius, [60](#)  
mIMEA60, [73](#)  
mIUnknown, [73](#)  
Monitor, [83](#)  
Movement, [80](#)  
Multiboot, [78](#)  
MultiFileMode, [83](#)  
MultipleHandlesOneQueues, [57](#)  
Multiwell, [68](#), [78](#)  
MultiwellI96, [52](#)  
Multiwell\_ICC, [77](#)  
Multiwell\_MEA\_Mini, [78](#)  
MultiwellDigitalSourceEnumNet, [73](#)  
MultiwellHeadstage, [67](#)

MultiwellInterfaceboard, 67  
MultiwellMini, 81  
MultiwellMiniHeadstage, 67  
MultiwellOptoStim, 77  
MultiwellPlateTypeEnumNet, 74  
Mux, 63  
MuxOtherDevice, 63  
Nanion, 86  
NanoAmpere, 60  
NanoVolt, 60  
NCBathCurrent, 58  
NCCol2Current, 58  
NChipTemperature, 58  
Neptun, 78  
NeuroChip, 78  
NeurochipConfig, 78  
NF\_Gen, 78  
NineWell, 85  
nMos16LV, 65  
nMos32LV, 65  
nMos36LN, 65  
nMos64LN, 65  
No\_Plate, 74  
None, 52, 63, 76, 84, 89, 90  
Normal, 75  
NOT\_CONNECTED, 65  
NotApplicable, 52  
NoUnit, 60  
NTC10K, 86  
Octopot, 76  
Off, 79, 80, 90  
off, 79  
OK, 65  
Okuvision\_Stimulator, 77  
OnChannelData, 93  
OnDeviceArrivalRemoval, 93  
One, 57, 70, 72, 74, 82, 85, 92  
OneHandleMultipleQueues, 57  
OneHandleOneQueue, 57  
OnError, 93  
OnMcsUsbDeviceState, 93  
OnMcsUsbDeviceStateCallback, 94  
OnMwPollStatus, 94  
OnStg200xDataHandler, 94  
OnStg200xErrorHandler, 94  
OnStgPollStatus, 94  
OnUpdateFirmwareProgress, 94  
OnUpdateFirmwareStatusChange, 94  
Open, 75  
OpenClamp, 89  
Output, 75  
PacketFrameContextGroup, 61, 70, 81, 92  
PatchServAdcModeEnumNet, 75  
PatchServer, 78  
PathIdent, 78  
PC, 93  
PCI, 90  
PCX, 77  
PeriodicPulse, 63  
PeristalticPump, 77  
PGA, 77  
PIC, 55  
PIC10, 55  
PIC11, 55  
PIC12, 55  
PIC2, 55  
PIC3, 55  
PIC4, 55  
PIC5, 55  
PIC6, 55  
PIC7, 55  
PIC8, 55  
PIC9, 55  
PicoAmpere, 60  
Plate\_24W030MGA, 74  
Plate\_24W300\_30\_1152GBA, 74  
Plate\_24W300\_30GBA, 74  
Plate\_24W300\_30GBB, 74  
Plate\_24W300\_30GMA, 74  
Plate\_24W700\_100FMA, 74  
Plate\_24W700\_100FMB, 74  
Plate\_24W700\_100FMC, 74  
Plate\_24W700\_100PBB, 74  
Plate\_72W500\_100FMA, 74  
Plate\_72W500\_100PMA, 74  
Plate\_96W300\_80\_1152FMA, 74  
Plate\_96W400\_80\_1152FMB, 74  
Plate\_96W700\_100FMA, 74  
Plate\_96W700\_100FMB, 74  
Plate\_96W700\_100GBA, 74  
Plate\_96W700\_100GBB, 74  
Plate\_96W700\_100GBC, 74  
Plate\_96W700\_100GBD, 74  
Plate\_96W700\_100GMA, 74  
Plate\_96W700\_100PBB, 74  
Plate\_Dummy, 74  
Plate\_Dummy\_126, 74  
Plate\_ICB8\_24W700\_100P8A, 74  
Plate\_ICB8\_96W700\_100P8A, 74  
PlateClampEnumNet, 75  
PlateClampLockEnumNet, 75  
PlusMinus10Volts, 53  
PlusMinus2Comma5Volts, 53  
PlusMinus5Volts, 53  
PortDirectionEnumNet, 75  
Pos900, 78  
PositionBase, 78  
PositionIIBase, 78  
PositionIICentralUnit, 78  
PositionImp, 78  
PostCommaA, 65  
PostCommaB, 65  
PP\_Pump\_Mode\_Type\_EnumNet, 75  
PPC, 77  
PPS2, 77  
PPS4plus1, 77



PPS5, [77](#)  
PPS5\_DIG, [77](#)  
PreCommaA, [65](#)  
PreCommaB, [65](#)  
ProductIdEnumNet, [76](#)  
PT100, [86](#)  
PT1000, [86](#)  
PulseGenerator, [57](#), [70](#), [73](#), [74](#), [82](#), [85](#), [92](#)  
PulseGenerator\_Mode\_EnumNet, [79](#)  
rawdata, [85](#)  
RC, [66](#)  
rc100mAh, [69](#)  
rc200mAh, [69](#)  
rc300mAh, [69](#)  
rc30mAh, [69](#)  
rcGreater300mAh, [69](#)  
Rectangle, [87](#)  
Ref16, [79](#)  
Ref24, [79](#)  
Ref32, [79](#)  
Ref8, [79](#)  
Reference, [80](#)  
ReferenceElectrodeModeEnumNet, [79](#)  
ReferenceElectrodeSwitchPositionEnumNet, [79](#)  
RegisterHigh, [63](#)  
RegisterLow, [63](#)  
Regular, [86](#)  
Renesas, [89](#)  
Reserved1, [86](#)  
Reserved2, [86](#)  
Reserved3, [86](#)  
Reserved4, [86](#)  
Reserved5, [86](#)  
Restart, [80](#)  
Retina\_LED, [77](#)  
RetriggerActionEnumNet, [79](#)  
Rising, [64](#)  
RoboCurrentModeEnumNet, [80](#)  
RoboInject, [78](#)  
Roboocyte2, [78](#)  
RoboStatusEventDelegate, [94](#)  
Running, [84](#)  
SafeIS, [78](#)  
SampleDstSize16, [80](#)  
SampleDstSize32, [80](#)  
SampleDstSizeNet, [80](#)  
SampleSize16Signed, [80](#)  
SampleSize16Unsigned, [80](#)  
SampleSize24Signed, [80](#)  
SampleSize24Unsigned, [80](#)  
SampleSize32Signed, [80](#)  
SampleSize32Unsigned, [80](#)  
SampleSize64Signed, [80](#)  
SampleSize64Unsigned, [80](#)  
SampleSizeNet, [80](#)  
SBSVector1, [59](#)  
SBSVector2, [59](#)  
SBSVector3, [59](#)  
SBSVector4, [59](#)  
SCU1ElectrodeGroupHS1, [81](#)  
SCU1ElectrodeGroupHS2, [81](#)  
SCU1ElectrodeGroupHS3, [81](#)  
SCU1ElectrodeGroupHS4, [81](#)  
SCU1HS1Sideband1, [82](#)  
SCU1HS1Sideband1TriggerStatus, [82](#)  
SCU1HS1Sideband2, [82](#)  
SCU1HS1Sideband2TriggerStatus, [82](#)  
SCU1HS2Sideband1, [82](#)  
SCU1HS2Sideband1TriggerStatus, [82](#)  
SCU1HS2Sideband2, [82](#)  
SCU1HS2Sideband2TriggerStatus, [82](#)  
SCU1HS3Sideband1, [82](#)  
SCU1HS3Sideband1TriggerStatus, [82](#)  
SCU1HS3Sideband2, [82](#)  
SCU1HS3Sideband2TriggerStatus, [82](#)  
SCU1HS4Sideband1, [82](#)  
SCU1HS4Sideband1TriggerStatus, [82](#)  
SCU1HS4Sideband2, [82](#)  
SCU1HS4Sideband2TriggerStatus, [82](#)  
SCU1LEDStim1, [82](#)  
SCU1LEDStim1TriggerStatus, [82](#)  
SCU1LEDStim2, [82](#)  
SCU1LEDStim2TriggerStatus, [82](#)  
SCU1LEDStim3, [82](#)  
SCU1LEDStim3TriggerStatus, [82](#)  
SCU1LEDStim4, [82](#)  
SCU1LEDStim4TriggerStatus, [82](#)  
SCU2ElectrodeGroupHS1, [81](#)  
SCU2ElectrodeGroupHS2, [81](#)  
SCU2ElectrodeGroupHS3, [81](#)  
SCU2ElectrodeGroupHS4, [81](#)  
SCU2HS1Sideband1, [83](#)  
SCU2HS1Sideband1TriggerStatus, [82](#)  
SCU2HS1Sideband2, [83](#)  
SCU2HS1Sideband2TriggerStatus, [82](#)  
SCU2HS2Sideband1, [83](#)  
SCU2HS2Sideband1TriggerStatus, [82](#)  
SCU2HS2Sideband2, [83](#)  
SCU2HS2Sideband2TriggerStatus, [82](#)  
SCU2HS3Sideband1, [83](#)  
SCU2HS3Sideband1TriggerStatus, [82](#)  
SCU2HS3Sideband2, [83](#)  
SCU2HS3Sideband2TriggerStatus, [82](#)  
SCU2HS4Sideband1, [83](#)  
SCU2HS4Sideband1TriggerStatus, [82](#)  
SCU2HS4Sideband2, [83](#)  
SCU2HS4Sideband2TriggerStatus, [83](#)  
SCU2LEDStim1, [83](#)  
SCU2LEDStim1TriggerStatus, [83](#)  
SCU2LEDStim2, [83](#)  
SCU2LEDStim2TriggerStatus, [83](#)  
SCU2LEDStim3, [83](#)  
SCU2LEDStim3TriggerStatus, [83](#)  
SCU2LEDStim4, [83](#)  
SCU2LEDStim4TriggerStatus, [83](#)  
SCU\_HeadstageIdEnumNet, [81](#)

- SCUDacqGroupChannelEnumNet, 81
- SCUDigitalSourceEnumNet, 82
- Settings, 93
- Signed\_16bit, 61
- Signed\_24bit, 61
- Signed\_32bit, 61
- Sine, 87
- Single, 80
- SingleWell, 85
- SixWell, 85
- SmartImplant, 78
- SOFAAndCTRLword, 60
- Software, 66
- SoftwareDongle, 78
- Standby, 80, 89
- Start, 64
- State, 90
- STG, 76
- Stg1, 64
- STG1DACSignalGroup, 61, 70, 81
- STG1SidebandsGroup, 61, 70, 81
- STG1TriggerStatusGroup, 61, 70, 81
- Stg2, 64
- Stg200xDigoutModeEnumNet, 83
- Stg200xMultiFileSubmodeEnumNet, 83
- Stg200xSegmentFlagsEnumNet, 84
- Stg200xTriggerStatusEnumNet, 84
- STG2DACSignalGroup, 70, 81
- STG2SidebandsGroup, 70, 81
- STG2TriggerStatusGroup, 70, 81
- Stg3, 64
- STG3008\_FA, 77
- Stg3008FilterAmpAmplificationEnumNet, 84
- STG4002, 76
- STG4002\_opto, 76
- STG4004, 76
- STG4004\_opto, 76
- STG4008, 76
- STG4008\_opto, 76
- STG400x, 76
- STG400x\_opto, 76
- STG5, 76
- STG\_DestinationEnumNet, 84
- StgListModeTrigger, 64
- StgTrigger, 64
- Stimulation, 58
- StimulationLayoutConfigurationEnumNet, 85
- Stop, 64, 75, 80
- StorageCharge, 69
- SubtractFromAll, 79
- SubtractFromAllOther, 79
- SubtractFromReferenceElectrodeOnly, 79
- SubtractionOff, 79
- SuperSpeed, 70
- Sw2to64, 77
- SYNCOUT1, 83
- SYNCOUT2, 83
- SYNCOUT3, 83
- SYNCOUT4, 83
- SYNCOUT5, 83
- SYNCOUT6, 83
- SYNCOUT7, 83
- SYNCOUT8, 83
- syncoutdata, 85
- SyncStart, 84
- Table, 93
- TBSI\_127, 52
- TBSI\_15, 52
- TBSI\_31, 52
- TBSI\_5, 52
- TBSI\_63, 52
- TBSI\_Dacq, 78
- TBSI\_DACQDigitalSourceEnumNet, 85
- TBSI\_Reserved, 52
- TbsiDacq, 68
- TbsiDacqHeadstage, 67
- TbsiDacqInterfaceboard, 67
- TC01, 77
- TC02, 77
- TCX, 77
- TcxDeviceTypeEnumNet, 86
- TcxSensorTypeEnumNet, 86
- TeerClampModeEnumNet, 86
- TeerWaveformEnumNet, 86
- Tersens, 76
- Timestamp, 60
- TouchTest, 93
- Triggerbox\_AMS, 77
- Triggerbox\_AMS3, 77
- Triggerbox\_IMS, 77
- Triggerbox\_R5, 77
- TriggerOnly, 84
- TriggerSourceEnumNet, 87
- TriggerStatus1, 59
- TriggerStatus2, 59
- TriggerStatus3, 59
- TriggerStatus4, 59
- tsAuxIn1, 88
- tsAuxIn2, 88
- tsDACQCy1Dev1Runs, 89
- tsDACQCy1Dev2Runs, 89
- tsDACQCy2Dev1Runs, 89
- tsDACQCy2Dev2Runs, 89
- tsDigitalIn1, 87
- tsDigitalIn10, 87
- tsDigitalIn11, 87
- tsDigitalIn12, 87
- tsDigitalIn13, 87
- tsDigitalIn14, 87
- tsDigitalIn15, 87
- tsDigitalIn16, 87
- tsDigitalIn17, 87
- tsDigitalIn18, 87
- tsDigitalIn19, 87
- tsDigitalIn2, 87
- tsDigitalIn20, 87

- tsDigitalIn21, [87](#)
- tsDigitalIn22, [87](#)
- tsDigitalIn23, [87](#)
- tsDigitalIn24, [87](#)
- tsDigitalIn25, [87](#)
- tsDigitalIn26, [87](#)
- tsDigitalIn27, [87](#)
- tsDigitalIn28, [87](#)
- tsDigitalIn29, [87](#)
- tsDigitalIn3, [87](#)
- tsDigitalIn30, [87](#)
- tsDigitalIn31, [87](#)
- tsDigitalIn32, [87](#)
- tsDigitalIn4, [87](#)
- tsDigitalIn5, [87](#)
- tsDigitalIn6, [87](#)
- tsDigitalIn7, [87](#)
- tsDigitalIn8, [87](#)
- tsDigitalIn9, [87](#)
- tsDigitalPuse0, [88](#)
- tsDigitalPuse1, [88](#)
- tsDigitalPuse10, [88](#)
- tsDigitalPuse11, [88](#)
- tsDigitalPuse12, [88](#)
- tsDigitalPuse13, [88](#)
- tsDigitalPuse14, [88](#)
- tsDigitalPuse15, [88](#)
- tsDigitalPuse16, [88](#)
- tsDigitalPuse17, [88](#)
- tsDigitalPuse18, [89](#)
- tsDigitalPuse19, [89](#)
- tsDigitalPuse2, [88](#)
- tsDigitalPuse20, [89](#)
- tsDigitalPuse21, [89](#)
- tsDigitalPuse22, [89](#)
- tsDigitalPuse23, [89](#)
- tsDigitalPuse24, [89](#)
- tsDigitalPuse25, [89](#)
- tsDigitalPuse26, [89](#)
- tsDigitalPuse27, [89](#)
- tsDigitalPuse28, [89](#)
- tsDigitalPuse29, [89](#)
- tsDigitalPuse3, [88](#)
- tsDigitalPuse30, [89](#)
- tsDigitalPuse31, [89](#)
- tsDigitalPuse4, [88](#)
- tsDigitalPuse5, [88](#)
- tsDigitalPuse6, [88](#)
- tsDigitalPuse7, [88](#)
- tsDigitalPuse8, [88](#)
- tsDigitalPuse9, [88](#)
- tsFeedback1, [87](#)
- tsFeedback10, [88](#)
- tsFeedback11, [88](#)
- tsFeedback12, [88](#)
- tsFeedback13, [88](#)
- tsFeedback14, [88](#)
- tsFeedback15, [88](#)
- tsFeedback16, [88](#)
- tsFeedback17, [88](#)
- tsFeedback18, [88](#)
- tsFeedback19, [88](#)
- tsFeedback2, [87](#)
- tsFeedback20, [88](#)
- tsFeedback21, [88](#)
- tsFeedback22, [88](#)
- tsFeedback23, [88](#)
- tsFeedback24, [88](#)
- tsFeedback25, [88](#)
- tsFeedback26, [88](#)
- tsFeedback27, [88](#)
- tsFeedback28, [88](#)
- tsFeedback29, [88](#)
- tsFeedback3, [87](#)
- tsFeedback30, [88](#)
- tsFeedback31, [88](#)
- tsFeedback32, [88](#)
- tsFeedback4, [88](#)
- tsFeedback5, [88](#)
- tsFeedback6, [88](#)
- tsFeedback7, [88](#)
- tsFeedback8, [88](#)
- tsFeedback9, [88](#)
- tsNone, [87](#)
- tsSidebandBit8, [89](#)
- tsTriggered, [89](#)
- Unknown, [52](#), [53](#), [66](#), [86](#), [89](#)
- unknown, [65](#)
- UnknownDest, [57](#)
- UnknownSpeed, [70](#)
- Unlock, [75](#)
- Unsigned\_16bit, [61](#)
- Unsigned\_24bit, [61](#)
- Unsigned\_32bit, [61](#)
- UpdateTrigger, [84](#)
- USB, [53](#)
- USB\_TARGET1, [57](#)
- USB\_TARGET2, [57](#)
- USB\_TARGET3, [57](#)
- UsbTest, [78](#)
- UsbVendorIdEnumNet, [89](#)
- UssingChamber, [67](#)
- UssingClampModeEnumNet, [89](#)
- UssingRail, [67](#)
- UssingUnitEnumNet, [90](#)
- VendorIdEnumNet, [90](#)
- Volt, [53](#), [90](#)
- Voltage\_3V3, [68](#)
- Voltage\_5V0, [68](#)
- VoltageClamp, [89](#)
- W2100, [78](#)
- W2100\_Accel\_Gyro\_Select\_EnumNet, [90](#)
- W2100DacqGroupChannelEnumNet, [90](#)
- W2100DigitalSourceEnumNet, [92](#)
- W2100IFB2, [67](#)
- W2100Interfaceboard, [67](#)

- W2100WirelessReceiver, [67](#), [68](#)
- W2100WirelessReceiverAnalog, [67](#), [68](#)
- Warner, [86](#)
- Warner\_TEER\_Machine, [79](#)
- Warner\_Ussing, [79](#)
- WARNER\_USSING\_DEVICE, [63](#)
- Warner\_Valve\_Control, [78](#)
- WARNER\_VALVE\_CONTROL\_DEVICE, [63](#)
- Whole\_Cell\_Patch, [78](#)
- WholeCellPatch, [81](#)
- WholeCellPatchHeadstage, [67](#)
- WirelessHeadStageAccDataRE1HS1, [91](#)
- WirelessHeadStageAccDataRE1HS2, [91](#)
- WirelessHeadStageAccDataRE1HS3, [91](#)
- WirelessHeadStageAccDataRE1HS4, [91](#)
- WirelessHeadStageAccDataRE2HS1, [91](#)
- WirelessHeadStageAccDataRE2HS2, [91](#)
- WirelessHeadStageAccDataRE2HS3, [91](#)
- WirelessHeadStageAccDataRE2HS4, [91](#)
- WirelessHeadStageAnalogRE1HS1, [91](#)
- WirelessHeadStageAnalogRE1HS2, [91](#)
- WirelessHeadStageAnalogRE1HS3, [91](#)
- WirelessHeadStageAnalogRE1HS4, [91](#)
- WirelessHeadStageAnalogRE2HS1, [91](#)
- WirelessHeadStageAnalogRE2HS2, [91](#)
- WirelessHeadStageAnalogRE2HS3, [91](#)
- WirelessHeadStageAnalogRE2HS4, [91](#)
- WirelessHeadStageGyroDataRE1HS1, [91](#)
- WirelessHeadStageGyroDataRE1HS2, [91](#)
- WirelessHeadStageGyroDataRE1HS3, [91](#)
- WirelessHeadStageGyroDataRE1HS4, [91](#)
- WirelessHeadStageGyroDataRE2HS1, [91](#)
- WirelessHeadStageGyroDataRE2HS2, [91](#)
- WirelessHeadStageGyroDataRE2HS3, [91](#)
- WirelessHeadStageGyroDataRE2HS4, [91](#)
- WirelessHeadStageOptoStimCurrentRE1HS1, [91](#)
- WirelessHeadStageOptoStimCurrentRE1HS2, [91](#)
- WirelessHeadStageOptoStimCurrentRE1HS3, [91](#)
- WirelessHeadStageOptoStimCurrentRE1HS4, [91](#)
- WirelessHeadStageOptoStimCurrentRE2HS1, [91](#)
- WirelessHeadStageOptoStimCurrentRE2HS2, [91](#)
- WirelessHeadStageOptoStimCurrentRE2HS3, [91](#)
- WirelessHeadStageOptoStimCurrentRE2HS4, [92](#)
- WirelessHeadStageReservedARE1HS1, [91](#)
- WirelessHeadStageReservedARE1HS2, [91](#)
- WirelessHeadStageReservedARE1HS3, [91](#)
- WirelessHeadStageReservedARE1HS4, [91](#)
- WirelessHeadStageReservedARE2HS1, [91](#)
- WirelessHeadStageReservedARE2HS2, [91](#)
- WirelessHeadStageReservedARE2HS3, [92](#)
- WirelessHeadStageReservedARE2HS4, [92](#)
- WirelessHeadStageReservedBRE1HS1, [92](#)
- WirelessHeadStageReservedBRE1HS2, [92](#)
- WirelessHeadStageReservedBRE1HS3, [92](#)
- WirelessHeadStageReservedBRE1HS4, [92](#)
- WirelessHeadStageReservedBRE2HS1, [92](#)
- WirelessHeadStageReservedBRE2HS2, [92](#)
- WirelessHeadStageReservedBRE2HS3, [92](#)
- WirelessHeadStageReservedBRE2HS4, [92](#)
- WirelessHeadStageStatusRE1HS1, [91](#)
- WirelessHeadStageStatusRE1HS2, [91](#)
- WirelessHeadStageStatusRE1HS3, [91](#)
- WirelessHeadStageStatusRE1HS4, [91](#)
- WirelessHeadStageStatusRE2HS1, [91](#)
- WirelessHeadStageStatusRE2HS2, [91](#)
- WirelessHeadStageStatusRE2HS3, [91](#)
- WirelessHeadStageStatusRE2HS4, [91](#)
- WirelessTestAdapter, [52](#)
- Work, [93](#)
- WPA16, [78](#)
- WPA32, [78](#)
- WPA4, [78](#)
- WPA8, [78](#)
- WvcDisplayModeEnumNet, [93](#)
- WvcValveModeEnumNet, [93](#)
- Zero, [57](#), [70](#), [72](#), [74](#), [82](#), [85](#), [92](#)
- MCS\_ANY\_BUS
  - Mcs::Usb, [69](#)
- MCS\_CHANNELTEST\_DEVICE
  - Mcs::Usb, [62](#)
- MCS\_DEVICE\_ANY
  - Mcs::Usb, [62](#)
- MCS\_DEVICE\_NONE
  - Mcs::Usb, [63](#)
- MCS\_DEVICE\_USB
  - Mcs::Usb, [62](#)
- MCS\_DEVICE\_USB\_CYPRESS
  - Mcs::Usb, [63](#)
- MCS\_ENCAPSULATOR\_DEVICE
  - Mcs::Usb, [62](#)
- MCS\_EXTERN\_BC\_TESTER\_DEVICE
  - Mcs::Usb, [62](#)
- MCS\_EXTERN\_D\_TESTER\_DEVICE
  - Mcs::Usb, [62](#)
- MCS\_FCX\_DEVICE
  - Mcs::Usb, [62](#)
- MCS\_FYI\_DEVICE
  - Mcs::Usb, [62](#)
- MCS\_GENERIC\_DEVELOPMENT\_DEVICE
  - Mcs::Usb, [62](#)
- MCS\_HICLAMP\_DEVICE
  - Mcs::Usb, [62](#)
- MCS\_HLA\_DEVICE
  - Mcs::Usb, [62](#)
- MCS\_MBC08\_DEVICE
  - Mcs::Usb, [63](#)
- MCS\_MC\_STIMULUS\_DEVICE
  - Mcs::Usb, [62](#)

MCS\_MCCARD\_DEVICE  
Mcs::Usb, [62](#)

MCS\_MEA\_CLEAN\_DEVICE  
Mcs::Usb, [62](#)

MCS\_MEA\_COAT\_DEVICE  
Mcs::Usb, [62](#)

MCS\_MEA\_DEVICE  
Mcs::Usb, [62](#)

MCS\_MEA\_IMPEDANCE\_DEVICE  
Mcs::Usb, [62](#)

MCS\_MEA\_SWITCH\_DEVICE  
Mcs::Usb, [62](#)

MCS\_MEASURETABLE\_DEVICE  
Mcs::Usb, [62](#)

MCS\_MEAUSB\_DEVICE  
Mcs::Usb, [62](#)

MCS\_NF\_GEN\_DEVICE  
Mcs::Usb, [62](#)

MCS\_OCTOPOT\_DEVICE  
Mcs::Usb, [62](#)

MCS\_OKUVISION\_STIMULATOR\_DEVICE  
Mcs::Usb, [62](#)

MCS\_old  
Mcs::Usb, [90](#)

MCS\_PATCHSERVER\_DEVICE  
Mcs::Usb, [62](#)

MCS\_PATHIDENT\_DEVICE  
Mcs::Usb, [62](#)

MCS\_PCI\_BUS  
Mcs::Usb, [69](#)

MCS\_PCX\_DEVICE  
Mcs::Usb, [62](#)

MCS\_PEDOTER\_DEVICE  
Mcs::Usb, [63](#)

MCS\_PERISTALTIC\_PUMP\_DEVICE  
Mcs::Usb, [62](#)

MCS\_PGA\_DEVICE  
Mcs::Usb, [62](#)

MCS\_PPC\_DEVICE  
Mcs::Usb, [63](#)

MCS\_PPS5\_DEVICE  
Mcs::Usb, [62](#)

MCS\_PPS\_DEVICE  
Mcs::Usb, [62](#)

MCS\_RETINA\_AMS\_DONGLE  
Mcs::Usb, [62](#)

MCS\_RETINA\_LED\_DEVICE  
Mcs::Usb, [62](#)

MCS\_ROBO\_DEVICE  
Mcs::Usb, [62](#)

MCS\_ROBOINJECT\_DEVICE  
Mcs::Usb, [62](#)

MCS\_ROBOOCYTE2\_DEVICE  
Mcs::Usb, [62](#)

MCS\_SAFEIS\_DEVICE  
Mcs::Usb, [62](#)

MCS\_SMARTIMPLANT\_DEVICE  
Mcs::Usb, [62](#)

MCS\_SOFTWARE\_DONGLE\_DEVICE  
Mcs::Usb, [62](#)

MCS\_STG\_DEVICE  
Mcs::Usb, [62](#)

MCS\_SW2TO64\_DEVICE  
Mcs::Usb, [62](#)

MCS\_TCX\_DEVICE  
Mcs::Usb, [62](#)

MCS\_TERSENS\_DEVICE  
Mcs::Usb, [62](#)

MCS\_UNDEFINED\_BUS  
Mcs::Usb, [69](#)

MCS\_USB\_BUS  
Mcs::Usb, [69](#)

McsBus  
CPPCDeviceNet, [424](#)  
CPPS\_DeviceNet, [433](#)  
CRoboDeviceNet, [486](#)

MCSBUS0  
Mcs::Usb, [54](#)

MCSBUS1  
FirmwareDestinationNames, [697](#)  
Mcs::Usb, [53](#)

MCSBUS10  
FirmwareDestinationNames, [697](#)  
Mcs::Usb, [54](#)

MCSBUS11  
FirmwareDestinationNames, [697](#)  
Mcs::Usb, [54](#)

MCSBUS12  
FirmwareDestinationNames, [697](#)  
Mcs::Usb, [54](#)

MCSBUS13  
FirmwareDestinationNames, [697](#)  
Mcs::Usb, [54](#)

MCSBUS14  
Mcs::Usb, [54](#)

MCSBUS15  
Mcs::Usb, [54](#)

MCSBUS2  
FirmwareDestinationNames, [697](#)  
Mcs::Usb, [53](#)

MCSBUS3  
FirmwareDestinationNames, [697](#)  
Mcs::Usb, [54](#)

MCSBUS4  
FirmwareDestinationNames, [697](#)  
Mcs::Usb, [54](#)

MCSBUS5  
FirmwareDestinationNames, [697](#)  
Mcs::Usb, [54](#)

MCSBUS6  
FirmwareDestinationNames, [697](#)  
Mcs::Usb, [54](#)

MCSBUS7  
FirmwareDestinationNames, [697](#)  
Mcs::Usb, [54](#)

MCSBUS8

FirmwareDestinationNames, [698](#)  
 Mcs::Usb, [54](#)  
 MCSBUS9  
 FirmwareDestinationNames, [698](#)  
 Mcs::Usb, [54](#)  
 McsBus\_MotorControl  
 CPeristalticPumpDeviceNet, [409](#)  
 CPPCDeviceNet, [424](#)  
 CPPS\_DeviceNet, [433](#)  
 CRoboDeviceNet, [486](#)  
 CRoboFluidDeviceNet, [490](#)  
 McsBus\_Sensor  
 CPPCDeviceNet, [424](#)  
 CPPS\_DeviceNet, [433](#)  
 McsBus\_VoltageMode  
 CFluidControlDeviceNet, [151](#)  
 McsBus\_XY  
 CRoboDeviceNet, [483](#)  
 McsBus\_ZI  
 CRoboDeviceNet, [483](#)  
 McsBusTypeEnumNet  
 Mcs::Usb, [69](#)  
 McsUsbDeviceStateEvent  
 CMcsUsbDeviceStatePushFunctionNet, [299](#)  
 CMcsUsbDeviceStatePushNet, [300](#)  
 McsUsbSpeedEnumNet  
 Mcs::Usb, [69](#)  
 MCU1  
 FirmwareDestinationNames, [698](#)  
 Mcs::Usb, [53](#)  
 ME128  
 Mcs::Usb, [77](#)  
 ME16  
 Mcs::Usb, [77](#)  
 ME2100  
 Mcs::Usb, [78](#)  
 Me2100\_32PICiCE40  
 Mcs::Usb, [81](#)  
 Me2100\_32PICiCE40Headstage  
 Mcs::Usb, [67](#)  
 Me2100\_32Xilinx  
 Mcs::Usb, [81](#)  
 Me2100\_32XilinxHeadstage  
 Mcs::Usb, [67](#)  
 Me2100Graphene16\_32  
 Mcs::Usb, [81](#)  
 Me2100Graphene16\_32Headstage  
 Mcs::Usb, [67](#)  
 Me2100Interfaceboard  
 Mcs::Usb, [67](#)  
 Me2100InvitroSignalCollectorUnit  
 Mcs::Usb, [67](#)  
 Me2100InvivoSignalCollectorUnit  
 Mcs::Usb, [67](#)  
 Me2100UPA32  
 Mcs::Usb, [81](#)  
 Me2100UPA32Headstage  
 Mcs::Usb, [67](#)  
 ME256  
 Mcs::Usb, [77](#)  
 ME32  
 Mcs::Usb, [77](#)  
 ME64  
 Mcs::Usb, [77](#)  
 MEA1060  
 Mcs::Usb, [77](#)  
 MEA120  
 Mcs::Usb, [52](#)  
 MEA2100  
 Mcs::Usb, [77](#)  
 Mea2100  
 Mcs::Usb, [68](#)  
 MEA2100\_256  
 Mcs::Usb, [78](#)  
 Mea2100\_256  
 Mcs::Usb, [68](#)  
 MEA2100\_256DacqGroupChannelEnumNet  
 Mcs::Usb, [70](#)  
 MEA2100\_256DigitalSourceEnumNet  
 Mcs::Usb, [70](#)  
 Mea2100\_256Headstage  
 Mcs::Usb, [67](#)  
 Mea2100\_256Interfaceboard  
 Mcs::Usb, [67](#)  
 MEA2100\_32  
 Mcs::Usb, [77](#)  
 MEA2100\_Lite  
 Mcs::Usb, [78](#)  
 Mea2100\_Lite  
 Mcs::Usb, [68](#)  
 MEA2100\_Mini  
 Mcs::Usb, [78](#)  
 MEA2100\_Mini\_Usb\_develop  
 Mcs::Usb, [77](#)  
 MEA2100BetaScreen  
 Mcs::Usb, [78](#)  
 Mea2100BetaScreen  
 Mcs::Usb, [81](#)  
 Mea2100BetaScreenHeadstage  
 Mcs::Usb, [67](#)  
 MEA2100DigitalSourceEnumNet  
 Mcs::Usb, [72](#)  
 Mea2100Headstage  
 Mcs::Usb, [67](#)  
 Mea2100Interfaceboard  
 Mcs::Usb, [67](#)  
 Mea2100LiteHeadstage  
 Mcs::Usb, [67](#)  
 Mea2100Mini120  
 Mcs::Usb, [81](#)  
 Mea2100Mini120Headstage  
 Mcs::Usb, [67](#)  
 Mea2100Mini60ECP5  
 Mcs::Usb, [81](#)  
 Mea2100Mini60ECP5Headstage  
 Mcs::Usb, [67](#)

- Mea2100Mini60PICiCE40
  - Mcs::Usb, [81](#)
- Mea2100Mini60PICiCE40Headstage
  - Mcs::Usb, [67](#)
- Mea2100MultiwellIFB2
  - Mcs::Usb, [67](#)
- Mea2100STG
  - Mcs::Usb, [67](#)
- MEA252
  - Mcs::Usb, [52](#)
- MEA256
  - Mcs::Usb, [77](#)
- MEA2x32
  - Mcs::Usb, [52](#)
- MEA2x60
  - Mcs::Usb, [52](#)
- MEA32
  - Mcs::Usb, [52](#)
- MEA60
  - Mcs::Usb, [52](#)
- MEA\_2\_252\_2
  - Mcs::Usb, [52](#)
- MEA\_2\_252\_2\_6Well
  - Mcs::Usb, [52](#)
- MEA\_2\_252\_2\_9Well
  - Mcs::Usb, [52](#)
- MEA\_2\_252\_2\_Test
  - Mcs::Usb, [52](#)
- MEA\_Clean
  - Mcs::Usb, [77](#)
- MEA\_Coat
  - Mcs::Usb, [77](#)
- MEA\_Impedance
  - Mcs::Usb, [77](#)
- MEA\_Sanofi
  - Mcs::Usb, [77](#)
- MEA\_Switch
  - Mcs::Usb, [77](#)
- MEA\_Switch\_2\_1
  - Mcs::Usb, [77](#)
- MEA\_Switch\_4\_2
  - Mcs::Usb, [77](#)
- MeaAudioFunctionNet
  - CMeaDeviceNet, [361](#)
- MeaDigitalDataFunctionNet
  - CMeaDeviceNet, [361](#)
- MeaFeedbackFunctionNet
  - CMeaDeviceNet, [361](#)
- MeaLayoutEnumNet
  - Mcs::Usb, [73](#)
- Measure
  - CPathIdentDeviceNet, [406](#)
- MeasureReservoir
  - CPPCFunctionNet, [431](#)
- MeasureTable
  - Mcs::Usb, [78](#)
- MeasuringOnly
  - HeadStageIDType, [699](#)
- MeFunctionNet
  - CMeaDeviceNet, [361](#)
- MicroAmpere
  - Mcs::Usb, [60](#)
- MilliDegreeCelsius
  - Mcs::Usb, [60](#)
- mkfilter
  - mkfilterNet, [704](#)
- mkfilter\_coef\_in\_one\_set
  - mkfilterNet, [704](#)
- mkfilter\_highpass\_coef
  - mkfilterNet, [705](#)
- mkfilter\_highpass\_frequency\_from\_coef
  - mkfilterNet, [705](#)
- mkfilter\_highpass\_frequency\_from\_k
  - mkfilterNet, [705](#)
- mkfilter\_highpass\_k
  - mkfilterNet, [705](#)
- mkfilter\_MCS
  - mkfilterNet, [705](#)
- mkfilter\_MCS\_k
  - mkfilterNet, [705](#), [706](#)
- mkfilter\_normalize\_coeffs\_int
  - mkfilterNet, [706](#)
- mkfilter\_normalize\_coeffs\_short
  - mkfilterNet, [706](#)
- mkfilter\_normalize\_scale\_coeffs\_int
  - mkfilterNet, [706](#)
- mkfilter\_scale\_coef\_in\_one\_set
  - mkfilterNet, [706](#)
- mkfilterNet, [704](#)
  - mkfilter, [704](#)
  - mkfilter\_coef\_in\_one\_set, [704](#)
  - mkfilter\_highpass\_coef, [705](#)
  - mkfilter\_highpass\_frequency\_from\_coef, [705](#)
  - mkfilter\_highpass\_frequency\_from\_k, [705](#)
  - mkfilter\_highpass\_k, [705](#)
  - mkfilter\_MCS, [705](#)
  - mkfilter\_MCS\_k, [705](#), [706](#)
  - mkfilter\_normalize\_coeffs\_int, [706](#)
  - mkfilter\_normalize\_coeffs\_short, [706](#)
  - mkfilter\_normalize\_scale\_coeffs\_int, [706](#)
  - mkfilter\_scale\_coef\_in\_one\_set, [706](#)
- mIMEA60
  - Mcs::Usb, [73](#)
- mlUnknown
  - Mcs::Usb, [73](#)
- ModifyRegister
  - CMcsUsbNet, [329](#)
- Monitor
  - Mcs::Usb, [83](#)
- MoveAbs
  - CRoboDeviceNet, [478](#), [479](#)
- MoveAbsI
  - CRoboStatorDeviceNet, [495](#)
- MoveAbsXY
  - CRoboStatorDeviceNet, [495](#)
- MoveAbsZ



- CRoboStatorDeviceNet, 495, 496
- Movement
  - Mcs::Usb, 80
- Multiboot
  - Mcs::Usb, 78
- MultibootGetCypressImageId
  - CMcsUsbNet, 329
- MultibootGetImageId
  - CMcsUsbNet, 329
- MultibootGetSelectedImage
  - CMcsUsbNet, 329
- MultibootSelectImage
  - CMcsUsbNet, 329
- MultiFileMode
  - Mcs::Usb, 83
- MultipleHandlesOneQueues
  - Mcs::Usb, 57
- Multiwell
  - Mcs::Usb, 68, 78
- MultiwellI96
  - Mcs::Usb, 52
- Multiwell\_ICC
  - Mcs::Usb, 77
- Multiwell\_MEA\_Mini
  - Mcs::Usb, 78
- MultiwellDigitalSourceEnumNet
  - Mcs::Usb, 73
- MultiwellHeadstage
  - Mcs::Usb, 67
- MultiwellInterfaceboard
  - Mcs::Usb, 67
- MultiwellMini
  - Mcs::Usb, 81
- MultiwellMiniHeadstage
  - Mcs::Usb, 67
- MultiwellOptoStim
  - Mcs::Usb, 77
- MultiwellPlateTypeEnumNet
  - Mcs::Usb, 74
- Mux
  - Mcs::Usb, 63
- MuxOtherDevice
  - Mcs::Usb, 63
- MwPollStatusEvent
  - CStg200xDownloadNet, 576
- Nanion
  - Mcs::Usb, 86
- NanoAmpere
  - Mcs::Usb, 60
- NanoVolt
  - Mcs::Usb, 60
- NCBathCurrent
  - Mcs::Usb, 58
- NCCol2Current
  - Mcs::Usb, 58
- NChipTemperature
  - Mcs::Usb, 58
- Neptun
  - Mcs::Usb, 78
- NeuroChip
  - Mcs::Usb, 78
- NeurochipConfig
  - Mcs::Usb, 78
- NF\_Gen
  - Mcs::Usb, 78
- NineWell
  - Mcs::Usb, 85
- nMos16LV
  - Mcs::Usb, 65
- nMos32LV
  - Mcs::Usb, 65
- nMos36LN
  - Mcs::Usb, 65
- nMos64LN
  - Mcs::Usb, 65
- No\_Plate
  - Mcs::Usb, 74
- None
  - Mcs::Usb, 52, 63, 76, 84, 89, 90
- Normal
  - Mcs::Usb, 75
- NOT\_CONNECTED
  - Mcs::Usb, 65
- NotApplicable
  - Mcs::Usb, 52
- NoUnit
  - Mcs::Usb, 60
- NTC10K
  - Mcs::Usb, 86
- NullCommand
  - CRoboDeviceNet, 480
- NumberOfAnalogChannels
  - HeadStageIDType, 700
- NumberOfChannels
  - CDeviceGroupChannelInfoTemplateNet< Dacq-GroupChannelEnumTemplateNet >, 129
- NumberOfStimulationChannels
  - HeadStageIDType, 701
- NumCoefSets
  - CCreateFilterNet, 120
- Octopot
  - Mcs::Usb, 76
- Off
  - Mcs::Usb, 79, 80, 90
- off
  - Mcs::Usb, 79
- OK
  - Mcs::Usb, 65
- Okuvision\_Stimulator
  - Mcs::Usb, 77
- OnChannelData
  - Mcs::Usb, 93
- OnDeviceArrivalRemoval
  - Mcs::Usb, 93
- One
  - Mcs::Usb, 57, 70, 72, 74, 82, 85, 92



- OneHandleMultipleQueues
  - Mcs::Usb, [57](#)
- OneHandleOneQueue
  - Mcs::Usb, [57](#)
- OnError
  - Mcs::Usb, [93](#)
- OnGetActiveRunningTableNumber
  - CWarnerValveControllerDeviceNet, [665](#)
- OnGetAnalogThresholdHigh
  - CWarnerValveControllerDeviceNet, [665](#)
- OnGetAnalogThresholdLow
  - CWarnerValveControllerDeviceNet, [665](#)
- OnGetAnalogVoltage
  - CWarnerValveControllerDeviceNet, [665](#)
- OnGetAvailableHeadstages
  - CSCUFunctionNet, [516](#)
- OnGetCurrentNumberOfValves
  - CWarnerValveControllerDeviceNet, [665](#)
- OnGetDigitalOutPortValve
  - CWarnerValveControllerDeviceNet, [666](#)
- OnGetDigitalPortDirection
  - CWarnerValveControllerDeviceNet, [666](#)
- OnGetDisplayMode
  - CWarnerValveControllerDeviceNet, [666](#)
- OnGetPlateClampStateByHeadstage
  - CMultiwellCallbackFunctionNet, [383](#)
- OnGetTableNamebyIndex
  - CWarnerValveControllerDeviceNet, [666](#)
- OnGetValveActive
  - CWarnerValveControllerDeviceNet, [666](#)
- OnGetValveBoardRevision
  - CWarnerValveControllerDeviceNet, [666](#)
- OnGetValveDigitalInPort
  - CWarnerValveControllerDeviceNet, [666](#)
- OnGetValveLedOn
  - CWarnerValveControllerDeviceNet, [666](#)
- OnGetValveManualGroup
  - CWarnerValveControllerDeviceNet, [666](#)
- OnGetValveManualState
  - CWarnerValveControllerDeviceNet, [667](#)
- OnGetValveMode
  - CWarnerValveControllerDeviceNet, [667](#)
- OnIsDigitalOutPortInverted
  - CWarnerValveControllerDeviceNet, [667](#)
- OnIsHeadstageAvailable
  - CSCUFunctionNet, [516](#)
- OnIsValveDigitalInInverted
  - CWarnerValveControllerDeviceNet, [667](#)
- OnIsValveOpen
  - CWarnerValveControllerDeviceNet, [667](#)
- OnIsValveOpenInAnalogMode
  - CWarnerValveControllerDeviceNet, [667](#)
- OnIsValveOpenInDigitalMode
  - CWarnerValveControllerDeviceNet, [667](#)
- OnMcsUsbDeviceState
  - Mcs::Usb, [93](#)
- OnMcsUsbDeviceStateCallback
  - Mcs::Usb, [94](#)
- OnMwPollStatus
  - Mcs::Usb, [94](#)
- OnStg200xDataHandler
  - Mcs::Usb, [94](#)
- OnStg200xErrorHandler
  - Mcs::Usb, [94](#)
- OnStgPollStatus
  - Mcs::Usb, [94](#)
- OnTableEntryChanged
  - CWarnerValveControllerDeviceNet, [667](#)
- OnUpdateFirmwareProgress
  - Mcs::Usb, [94](#)
- OnUpdateFirmwareStatusChange
  - Mcs::Usb, [94](#)
- Open
  - Mcs::Usb, [75](#)
- OpenClamp
  - Mcs::Usb, [89](#)
- OpenPipe
  - CGenericDevelopDeviceNet, [165](#)
- OpenPlateClamp
  - CMultiwellDeviceNet, [389](#)
- operator=
  - DeviceldNet, [686](#)
- OpticalStimulation
  - HeadStageIDType, [699](#)
- Order
  - CCreateFilterNet, [120](#)
  - CFilterPropertyNet, [144](#)
- Output
  - Mcs::Usb, [75](#)
- PacketFrameContextGroup
  - Mcs::Usb, [61](#), [70](#), [81](#), [92](#)
- PatchServAdcModeEnumNet
  - Mcs::Usb, [75](#)
- PatchServer
  - Mcs::Usb, [78](#)
- PathIdent
  - Mcs::Usb, [78](#)
- PatternListStart
  - COctoPotDeviceNet, [399](#)
- PC
  - Mcs::Usb, [93](#)
- PCI
  - Mcs::Usb, [90](#)
- PCX
  - Mcs::Usb, [77](#)
- PeriodicPulse
  - Mcs::Usb, [63](#)
- PeristalticPump
  - Mcs::Usb, [77](#)
- PGA
  - Mcs::Usb, [77](#)
- PIC
  - FirmwareDestinationNames, [698](#)
  - Mcs::Usb, [55](#)
- PIC10
  - Mcs::Usb, [55](#)

PIC11  
     Mcs::Usb, [55](#)  
 PIC12  
     Mcs::Usb, [55](#)  
 PIC2  
     FirmwareDestinationNames, [698](#)  
     Mcs::Usb, [55](#)  
 PIC3  
     FirmwareDestinationNames, [698](#)  
     Mcs::Usb, [55](#)  
 PIC4  
     FirmwareDestinationNames, [698](#)  
     Mcs::Usb, [55](#)  
 PIC5  
     Mcs::Usb, [55](#)  
 PIC6  
     Mcs::Usb, [55](#)  
 PIC7  
     Mcs::Usb, [55](#)  
 PIC8  
     Mcs::Usb, [55](#)  
 PIC9  
     Mcs::Usb, [55](#)  
 PicoAmpere  
     Mcs::Usb, [60](#)  
 Plate\_24W030MGA  
     Mcs::Usb, [74](#)  
 Plate\_24W300\_30\_1152GBA  
     Mcs::Usb, [74](#)  
 Plate\_24W300\_30GBA  
     Mcs::Usb, [74](#)  
 Plate\_24W300\_30GBB  
     Mcs::Usb, [74](#)  
 Plate\_24W300\_30GMA  
     Mcs::Usb, [74](#)  
 Plate\_24W700\_100FMA  
     Mcs::Usb, [74](#)  
 Plate\_24W700\_100FMB  
     Mcs::Usb, [74](#)  
 Plate\_24W700\_100FMC  
     Mcs::Usb, [74](#)  
 Plate\_24W700\_100PBB  
     Mcs::Usb, [74](#)  
 Plate\_72W500\_100FMA  
     Mcs::Usb, [74](#)  
 Plate\_72W500\_100PMA  
     Mcs::Usb, [74](#)  
 Plate\_96W300\_80\_1152FMA  
     Mcs::Usb, [74](#)  
 Plate\_96W400\_80\_1152FMB  
     Mcs::Usb, [74](#)  
 Plate\_96W700\_100FMA  
     Mcs::Usb, [74](#)  
 Plate\_96W700\_100FMB  
     Mcs::Usb, [74](#)  
 Plate\_96W700\_100GBA  
     Mcs::Usb, [74](#)  
 Plate\_96W700\_100GBB  
     Mcs::Usb, [74](#)  
 Plate\_96W700\_100GBC  
     Mcs::Usb, [74](#)  
 Plate\_96W700\_100GBD  
     Mcs::Usb, [74](#)  
 Plate\_96W700\_100GMA  
     Mcs::Usb, [74](#)  
 Plate\_96W700\_100PBB  
     Mcs::Usb, [74](#)  
 Plate\_Dummy  
     Mcs::Usb, [74](#)  
 Plate\_Dummy\_126  
     Mcs::Usb, [74](#)  
 Plate\_ICB8\_24W700\_100P8A  
     Mcs::Usb, [74](#)  
 Plate\_ICB8\_96W700\_100P8A  
     Mcs::Usb, [74](#)  
 PlateClampEnumNet  
     Mcs::Usb, [75](#)  
 PlateClampLockEnumNet  
     Mcs::Usb, [75](#)  
 PlusMinus10Volts  
     Mcs::Usb, [53](#)  
 PlusMinus2Comma5Volts  
     Mcs::Usb, [53](#)  
 PlusMinus5Volts  
     Mcs::Usb, [53](#)  
 PollStatusEvent  
     CStimulusFunctionNet, [589](#)  
     CW2100\_StimulatorFunctionNet, [634](#)  
 PortDirectionEnumNet  
     Mcs::Usb, [75](#)  
 Pos900  
     Mcs::Usb, [78](#)  
 PositionBase  
     Mcs::Usb, [78](#)  
 PositionIIBase  
     Mcs::Usb, [78](#)  
 PositionIICentralUnit  
     Mcs::Usb, [78](#)  
 PositionImp  
     Mcs::Usb, [78](#)  
 PostCommaA  
     CFilterCoefficientsNet::s\_FilterAttributesNet, [715](#)  
     Mcs::Usb, [65](#)  
 PostCommaB  
     CFilterCoefficientsNet::s\_FilterAttributesNet, [715](#)  
     Mcs::Usb, [65](#)  
 PowerChip  
     CCMOSMea\_FunctionNet, [113](#)  
 PowerHS  
     CSCUFunctionNet, [516](#)  
 PP\_Pump\_Mode\_Type\_EnumNet  
     Mcs::Usb, [75](#)  
 PPC  
     Mcs::Usb, [77](#)  
 PPCFunction  
     CPPCDeviceNet, [424](#)

PPS2  
     Mcs::Usb, [77](#)  
 PPS4plus1  
     Mcs::Usb, [77](#)  
 PPS5  
     Mcs::Usb, [77](#)  
 PPS5\_DIG  
     Mcs::Usb, [77](#)  
 PPS\_Function  
     CPPS\_DeviceNet, [433](#)  
 PreCommaA  
     CFilterCoefficientsNet::s\_FilterAttributesNet, [715](#)  
     Mcs::Usb, [65](#)  
 PreCommaB  
     CFilterCoefficientsNet::s\_FilterAttributesNet, [715](#)  
     Mcs::Usb, [65](#)  
 PrepareAndAppendData  
     CStg200xDownloadNet, [573](#)  
     CStimulusFunctionNet, [584](#)  
 PrepareAndSendData  
     CStg200xDownloadNet, [574](#)  
     CStimulusFunctionNet, [585](#)  
 PrepareChannelData  
     CDigOutStimulatorFunctionNet, [132](#)  
 PrepareData  
     CStimulusFunctionNet, [586](#)  
     CW2100\_StimulatorFunctionNet, [631](#)  
 PrepareDataSync  
     CW2100\_StimulatorFunctionNet, [632](#)  
 Product  
     CMcsUsbListEntryNet, [314](#)  
 ProductIdEnumNet  
     Mcs::Usb, [76](#)  
 Program  
     CProgramPressureCurveNet, [440](#)  
 PT100  
     Mcs::Usb, [86](#)  
 PT1000  
     Mcs::Usb, [86](#)  
 PulseGenerator  
     CW2100\_FunctionNet, [626](#)  
     Mcs::Usb, [57](#), [70](#), [73](#), [74](#), [82](#), [85](#), [92](#)  
 PulseGenerator\_Mode\_EnumNet  
     Mcs::Usb, [79](#)  
 PumpOff  
     CRoboFluidDeviceNet, [489](#)  
 PumpOn  
     CRoboFluidDeviceNet, [489](#)  
 QueryTriggerstatus  
     CStg200xDownloadNet, [575](#)  
 RampStart  
     COctoPotDeviceNet, [399](#)  
 rawdata  
     Mcs::Usb, [85](#)  
 RC  
     Mcs::Usb, [66](#)  
 rc100mAh  
     Mcs::Usb, [69](#)  
 rc200mAh  
     Mcs::Usb, [69](#)  
 rc300mAh  
     Mcs::Usb, [69](#)  
 rc30mAh  
     Mcs::Usb, [69](#)  
 rcGreater300mAh  
     Mcs::Usb, [69](#)  
 Read  
     CExternDTesterDeviceNet, [136](#)  
 Read2  
     CExternDTesterDeviceNet, [136](#)  
 ReadAsFrameArray16  
     CMcsUsbDacqNet::ChannelBlockNet, [197](#)  
 ReadAsFrameArray132  
     CMcsUsbDacqNet::ChannelBlockNet, [198](#)  
 ReadAsFrameArrayUI16  
     CMcsUsbDacqNet::ChannelBlockNet, [199](#)  
 ReadAsFrameArrayUI32  
     CMcsUsbDacqNet::ChannelBlockNet, [199](#)  
 ReadBlockFromFlash  
     CMcsUsbFactoryNet, [305](#)  
 ReadBlockFromIFBGlobalEEprom  
     CMcsUsbFactoryNet, [305](#)  
 ReadBlockFromNVMEM  
     CMcsUsbFactoryNet, [306](#)  
 ReadClipping  
     CLIH3DeviceNet, [229](#)  
 ReadEepromRegisterPreconfig  
     CMcsUsbNet, [330](#)  
 ReadFramesDictI16  
     CMcsUsbDacqNet::ChannelBlockNet, [200](#)  
 ReadFramesDictI32  
     CMcsUsbDacqNet::ChannelBlockNet, [201](#)  
 ReadFramesDictUI16  
     CMcsUsbDacqNet::ChannelBlockNet, [201](#)  
 ReadFramesDictUI32  
     CMcsUsbDacqNet::ChannelBlockNet, [202](#)  
 ReadFramesI16  
     CMcsUsbDacqNet::ChannelBlockNet, [202](#)  
 ReadFramesI32  
     CMcsUsbDacqNet::ChannelBlockNet, [203](#)  
 ReadFramesUI16  
     CMcsUsbDacqNet::ChannelBlockNet, [204](#)  
 ReadFramesUI32  
     CMcsUsbDacqNet::ChannelBlockNet, [204](#)  
 ReadPipe  
     CGenericDevelopDeviceNet, [165](#)  
 ReadRegister  
     CMcsUsbNet, [330](#)  
 ReadRegister32  
     CMcsUsbNet, [330](#)  
 ReadRegisterTimeSlot  
     CMcsUsbNet, [330](#)  
 ReadUARTData  
     CLIH3DeviceNet, [229](#)  
 Receive

- CSerialPortNet, [521](#)
- ReceiveString
  - CSerialPortNet, [521](#), [522](#)
- Rectangle
  - Mcs::Usb, [87](#)
- Ref16
  - Mcs::Usb, [79](#)
- Ref24
  - Mcs::Usb, [79](#)
- Ref32
  - Mcs::Usb, [79](#)
- Ref8
  - Mcs::Usb, [79](#)
- Reference
  - Mcs::Usb, [80](#)
- ReferenceElectrodeModeEnumNet
  - Mcs::Usb, [79](#)
- ReferenceElectrodeSwitchPositionEnumNet
  - Mcs::Usb, [79](#)
- RegisterHigh
  - Mcs::Usb, [63](#)
- RegisterLow
  - Mcs::Usb, [63](#)
- Regular
  - Mcs::Usb, [86](#)
- RemoveSoftwareKey
  - CMcsUsbNet, [331](#)
- Renasas
  - Mcs::Usb, [89](#)
- RescanHeadstage
  - CMcsUsbNet, [331](#)
- Reserved1
  - Mcs::Usb, [86](#)
- Reserved2
  - Mcs::Usb, [86](#)
- Reserved3
  - Mcs::Usb, [86](#)
- Reserved4
  - Mcs::Usb, [86](#)
- Reserved5
  - Mcs::Usb, [86](#)
- ResetAdcOffset
  - COctoPotDeviceNet, [399](#)
- ResetChannelmap
  - CWClassicFunctionNet, [682](#)
- ResetDacOffset
  - COctoPotDeviceNet, [399](#)
- ResetHighpassFilter
  - CFilterConfigurationNet, [140](#)
- ResetPipe
  - CGenericDevelopDeviceNet, [165](#)
- ResetStatus
  - CStg200xDownloadBasicNet, [565](#)
- Restart
  - Mcs::Usb, [80](#)
- Retina\_LED
  - Mcs::Usb, [77](#)
- RetriggerActionEnumNet
  - Mcs::Usb, [79](#)
- RFFunction
  - CPositionIIDeviceNet, [420](#)
- Rising
  - Mcs::Usb, [64](#)
- RoboCurrentModeEnumNet
  - Mcs::Usb, [80](#)
- RoboDacq
  - CHiClampDeviceNet, [212](#)
- RoboDevice
  - CSafeISDeviceNet, [501](#)
- RoboError\_AnotherMaster
  - CRoboDeviceNet, [483](#)
- RoboError\_Base
  - CRoboDeviceNet, [484](#)
- RoboError\_CannotEscapeEndSwitch
  - CRoboDeviceNet, [484](#)
- RoboError\_CommandAlreadyInProgress
  - CRoboDeviceNet, [484](#)
- RoboError\_CommandNotPossible
  - CRoboDeviceNet, [484](#)
- RoboError\_CommunicationTimeout
  - CRoboDeviceNet, [484](#)
- RoboError\_DacqNotReady
  - CRoboDeviceNet, [484](#)
- RoboError\_DLLMovementTimeout
  - CRoboDeviceNet, [484](#)
- RoboError\_FindReferenceMethod
  - CRoboDeviceNet, [484](#)
- RoboError\_GilsonCommandPending
  - CRoboDeviceNet, [484](#)
- RoboError\_GilsonTimeout
  - CRoboDeviceNet, [484](#)
- RoboError\_GilsonWrondID
  - CRoboDeviceNet, [485](#)
- RoboError\_McsBus\_UnknownCommand
  - CRoboDeviceNet, [485](#)
- RoboError\_NoEndSwitch
  - CRoboDeviceNet, [485](#)
- RoboError\_NoMoreData
  - CRoboDeviceNet, [485](#)
- RoboError\_NoReference
  - CRoboDeviceNet, [485](#)
- RoboError\_NoSpeedOrAcceleration
  - CRoboDeviceNet, [485](#)
- RoboError\_OverPressure
  - CRoboDeviceNet, [485](#)
- RoboError\_ParameterNotAllowed
  - CRoboDeviceNet, [485](#)
- RoboError\_PeristalticTimeout
  - CRoboDeviceNet, [485](#)
- RoboError\_Phase0OutOfRange
  - CRoboDeviceNet, [485](#)
- RoboError\_PollLoopCanceled
  - CRoboDeviceNet, [486](#)
- RoboError\_PollLoopCanceledAndStopMovement
  - CRoboDeviceNet, [486](#)
- RoboError\_Pressure

- CRoboDeviceNet, [486](#)
- RoboError\_RangeExceeded
  - CRoboDeviceNet, [486](#)
- RoboError\_StateChangeNotPossible
  - CRoboDeviceNet, [486](#)
- RoboError\_Timeout
  - CRoboDeviceNet, [486](#)
- RoboError\_UnknownCommand
  - CRoboDeviceNet, [486](#)
- RoboInject
  - Mcs::Usb, [78](#)
- RoboMainLowLevelCommand
  - CRoboDeviceNet, [487](#)
- RoboMainStatorLowLevelCommand
  - CRoboStatorDeviceNet, [498](#)
- Roboocyte2
  - Mcs::Usb, [78](#)
- RoboStatusEvent
  - CRoboDeviceNet, [487](#)
- RoboStatusEventDelegate
  - Mcs::Usb, [94](#)
- Running
  - Mcs::Usb, [84](#)
- RunTable
  - CRoboDacqNet, [465](#)
- s\_FilterAttributesNet
  - CFilterCoefficientsNet::s\_FilterAttributesNet, [714](#)
- SafeIS
  - Mcs::Usb, [78](#)
- SampleDstSize16
  - Mcs::Usb, [80](#)
- SampleDstSize32
  - Mcs::Usb, [80](#)
- SampleDstSizeNet
  - Mcs::Usb, [80](#)
- SampleRate
  - CCreateFilterNet, [120](#)
- Samplerate
  - CMcsUsbDacqNet, [297](#)
- SampleSize16Signed
  - Mcs::Usb, [80](#)
- SampleSize16Unsigned
  - Mcs::Usb, [80](#)
- SampleSize24Signed
  - Mcs::Usb, [80](#)
- SampleSize24Unsigned
  - Mcs::Usb, [80](#)
- SampleSize32Signed
  - Mcs::Usb, [80](#)
- SampleSize32Unsigned
  - Mcs::Usb, [80](#)
- SampleSize64Signed
  - Mcs::Usb, [80](#)
- SampleSize64Unsigned
  - Mcs::Usb, [80](#)
- SampleSizeNet
  - Mcs::Usb, [80](#)
- SBSVector1
  - Mcs::Usb, [59](#)
- SBSVector2
  - Mcs::Usb, [59](#)
- SBSVector3
  - Mcs::Usb, [59](#)
- SBSVector4
  - Mcs::Usb, [59](#)
- Scale
  - CCreateFilterNet, [120](#)
- ScanForHeadstages
  - CWClassicFunctionNet, [682](#)
- SCU1ElectrodeGroupHS1
  - Mcs::Usb, [81](#)
- SCU1ElectrodeGroupHS2
  - Mcs::Usb, [81](#)
- SCU1ElectrodeGroupHS3
  - Mcs::Usb, [81](#)
- SCU1ElectrodeGroupHS4
  - Mcs::Usb, [81](#)
- SCU1HS1Sideband1
  - Mcs::Usb, [82](#)
- SCU1HS1Sideband1TriggerStatus
  - Mcs::Usb, [82](#)
- SCU1HS1Sideband2
  - Mcs::Usb, [82](#)
- SCU1HS1Sideband2TriggerStatus
  - Mcs::Usb, [82](#)
- SCU1HS2Sideband1
  - Mcs::Usb, [82](#)
- SCU1HS2Sideband1TriggerStatus
  - Mcs::Usb, [82](#)
- SCU1HS2Sideband2
  - Mcs::Usb, [82](#)
- SCU1HS2Sideband2TriggerStatus
  - Mcs::Usb, [82](#)
- SCU1HS3Sideband1
  - Mcs::Usb, [82](#)
- SCU1HS3Sideband1TriggerStatus
  - Mcs::Usb, [82](#)
- SCU1HS3Sideband2
  - Mcs::Usb, [82](#)
- SCU1HS3Sideband2TriggerStatus
  - Mcs::Usb, [82](#)
- SCU1HS4Sideband1
  - Mcs::Usb, [82](#)
- SCU1HS4Sideband1TriggerStatus
  - Mcs::Usb, [82](#)
- SCU1HS4Sideband2
  - Mcs::Usb, [82](#)
- SCU1HS4Sideband2TriggerStatus
  - Mcs::Usb, [82](#)
- SCU1LEDStim1
  - Mcs::Usb, [82](#)
- SCU1LEDStim1TriggerStatus
  - Mcs::Usb, [82](#)
- SCU1LEDStim2
  - Mcs::Usb, [82](#)
- SCU1LEDStim2TriggerStatus

Mcs::Usb, [82](#)  
 SCU1LEDStim3  
   Mcs::Usb, [82](#)  
 SCU1LEDStim3TriggerStatus  
   Mcs::Usb, [82](#)  
 SCU1LEDStim4  
   Mcs::Usb, [82](#)  
 SCU1LEDStim4TriggerStatus  
   Mcs::Usb, [82](#)  
 SCU2ElectrodeGroupHS1  
   Mcs::Usb, [81](#)  
 SCU2ElectrodeGroupHS2  
   Mcs::Usb, [81](#)  
 SCU2ElectrodeGroupHS3  
   Mcs::Usb, [81](#)  
 SCU2ElectrodeGroupHS4  
   Mcs::Usb, [81](#)  
 SCU2HS1Sideband1  
   Mcs::Usb, [83](#)  
 SCU2HS1Sideband1TriggerStatus  
   Mcs::Usb, [82](#)  
 SCU2HS1Sideband2  
   Mcs::Usb, [83](#)  
 SCU2HS1Sideband2TriggerStatus  
   Mcs::Usb, [82](#)  
 SCU2HS2Sideband1  
   Mcs::Usb, [83](#)  
 SCU2HS2Sideband1TriggerStatus  
   Mcs::Usb, [82](#)  
 SCU2HS2Sideband2  
   Mcs::Usb, [83](#)  
 SCU2HS2Sideband2TriggerStatus  
   Mcs::Usb, [82](#)  
 SCU2HS3Sideband1  
   Mcs::Usb, [83](#)  
 SCU2HS3Sideband1TriggerStatus  
   Mcs::Usb, [82](#)  
 SCU2HS3Sideband2  
   Mcs::Usb, [83](#)  
 SCU2HS3Sideband2TriggerStatus  
   Mcs::Usb, [82](#)  
 SCU2HS4Sideband1  
   Mcs::Usb, [83](#)  
 SCU2HS4Sideband1TriggerStatus  
   Mcs::Usb, [82](#)  
 SCU2HS4Sideband2  
   Mcs::Usb, [83](#)  
 SCU2HS4Sideband2TriggerStatus  
   Mcs::Usb, [83](#)  
 SCU2LEDStim1  
   Mcs::Usb, [83](#)  
 SCU2LEDStim1TriggerStatus  
   Mcs::Usb, [83](#)  
 SCU2LEDStim2  
   Mcs::Usb, [83](#)  
 SCU2LEDStim2TriggerStatus  
   Mcs::Usb, [83](#)  
 SCU2LEDStim3  
   Mcs::Usb, [83](#)  
 SCU2LEDStim3TriggerStatus  
   Mcs::Usb, [83](#)  
 SCU2LEDStim4  
   Mcs::Usb, [83](#)  
 SCU2LEDStim4TriggerStatus  
   Mcs::Usb, [83](#)  
 SCU\_HeadstageIdEnumNet  
   Mcs::Usb, [81](#)  
 SCUDacqGroupChannelEnumNet  
   Mcs::Usb, [81](#)  
 SCUDigitalSourceEnumNet  
   Mcs::Usb, [82](#)  
 SegmentDefine  
   CStg200xDownloadNet, [575](#)  
 SegmentSelect  
   CStg200xDownloadNet, [575](#)  
 SegmentStart  
   CStg200xDownloadNet, [576](#)  
 SelectHeadstage  
   CW2100\_FunctionNet, [625](#)  
 SelectTimeSlot  
   CW2100\_StimulatorFunctionNet, [632](#)  
 Send  
   CSerialPortNet, [522](#)  
 SendBuffered  
   CGilsonDeviceNet, [172](#)  
 SendChannelData  
   CDigOutStimulatorFunctionNet, [133](#)  
   CStg200xDownloadBasicNet, [566](#)  
 SendCommand  
   CLIH3DeviceNet, [230](#)  
 SendImmediate  
   CGilsonDeviceNet, [172](#)  
 SendImmediateGetResponse  
   CGilsonDeviceNet, [172](#)  
 SendMultiplexedData  
   CStimulusFunctionNet, [586](#)  
 SendPreparedData  
   CStg200xDownloadNet, [576](#)  
   CStimulusFunctionNet, [586](#)  
   CW2100\_StimulatorFunctionNet, [632](#)  
 SendStart  
   CStg200xBasicNet, [543](#)  
   CStimulusFunctionNet, [586](#)  
   CW2100\_StimulatorFunctionNet, [632](#)  
 SendStartDacq  
   CMcsUsbDacqNet, [287](#)  
 SendStartStgAndDacq  
   CMcsUsbDacqNet, [287](#)  
 SendStop  
   CStg200xBasicNet, [544](#)  
   CStimulusFunctionNet, [587](#)  
   CW2100\_StimulatorFunctionNet, [632](#)  
 SendStopDacq  
   CMcsUsbDacqNet, [287](#), [288](#)  
 SendStopStgAndDacq  
   CMcsUsbDacqNet, [288](#)

- SendStopStgAndDacqWithOptions
  - CMcsUsbDacqNet, [288](#)
- SendSyncData
  - CStg200xDownloadBasicNet, [566](#)
- Sensor
  - CFYIDeviceNet, [152](#)
  - CMeasureTableDeviceNet, [370](#)
  - CPatchServerDeviceNet, [405](#)
- SerialNumber
  - CMcsUsbListEntryNet, [314](#)
  - CMcsUsbNet, [338](#)
- SerialPort
  - CHLADeviceNet, [214](#)
- Set4ADCCatchampAverageShift
  - CMcsBus\_SensorNet, [262](#)
- Set4ADCMode
  - CMcsBus\_SensorNet, [262](#)
- Set4DAC
  - CMcsBus\_SensorNet, [262](#)
- Set\_Values
  - CNF\_GenDeviceNet, [397](#)
  - CPathIdentDeviceNet, [406](#)
- SetAbsMaxCurrentInMicroAmp
  - CMultiwellOptoStimFunctionNet, [394](#)
- SetAccelerationI
  - CRoboStatorDeviceNet, [496](#)
- SetAccelerationNativeI
  - CRoboStatorDeviceNet, [496](#)
- SetAccelerationNativeXY
  - CRoboStatorDeviceNet, [496](#)
- SetAccelerationNativeZ
  - CRoboStatorDeviceNet, [496](#)
- SetAccelerationXY
  - CRoboStatorDeviceNet, [496](#)
- SetAccelerationZ
  - CRoboStatorDeviceNet, [496](#)
- SetAccelGyroDesiredRate
  - CW2100\_FunctionNet, [625](#)
- SetAccelGyroEnabled
  - CW2100\_FunctionNet, [625](#)
- SetAccelRange
  - CW2100\_FunctionNet, [625](#)
- SetActiveRunningTableName
  - CWarnerValveControllerDeviceNet, [667](#)
- SetADC
  - CWarnerValveControllerDeviceTesterFunctionNet, [678](#)
- SetAdcChannels
  - CSafeISDeviceNet, [500](#)
- SetADCGain
  - CGrapheneFunctionNet, [182](#), [183](#)
- SetADCInputOffset
  - CCMOSMea\_FunctionNet, [113](#)
- SetADCOffset
  - CGrapheneFunctionNet, [183](#)
- SetAdcOffset
  - CLIH3DeviceNet, [230](#)
  - COctoPotDeviceNet, [399](#)
- SetAdcOffsetPermanent
  - CLIH3DeviceNet, [230](#)
- SetAdcSamplePos
  - CSafeISDeviceNet, [500](#)
- SetAirpressureLimit
  - CRoboDeviceNet, [480](#)
- SetAirValve
  - CRoboDeviceNet, [480](#)
- SetAllDigout
  - CRoboDacqNet, [465](#)
- SetAmplificationSwitch
  - COctoPotDeviceNet, [399](#)
- SetAmplitude
  - CChannelTestDeviceNet, [103](#)
- SetAmplitude\_nA
  - CTEERFunctionNet, [612](#)
- SetAnalogOutADCRange
  - CSCUFunctionNet, [517](#)
- SetAnalogOutChannel
  - CW2100\_FunctionNet, [625](#)
- SetAnalogOutChannels
  - CSCUFunctionNet, [517](#)
- SetAnalogOutDACRange
  - CSCUFunctionNet, [517](#)
- SetAnalogOutFilter
  - CW2100\_FunctionNet, [625](#)
- SetAnalogThresholdHigh
  - CWarnerValveControllerDeviceNet, [668](#)
- SetAnalogThresholdLow
  - CWarnerValveControllerDeviceNet, [668](#)
- SetAnalogVoltageRange
  - CPPCFunctionNet, [431](#)
- SetAnalogVoltages
  - CPPS\_FunctionNet, [436](#)
- SetAttenuation
  - CChannelTestDeviceNet, [103](#)
- SetAudioChannels
  - CMeaAudioFunctionNet, [342](#), [343](#)
  - CW2100\_FunctionNet, [625](#)
- SetAudioOutDacParameter
  - CLIH3DeviceNet, [230](#)
- SetAutocalibrationDisabled
  - CStg200xBasicNet, [544](#)
- SetAxisConfig
  - CRoboDeviceNet::RoboMainLowLevelCommands, [711](#)
- SetAxisLED
  - CRoboocyte2DeviceNet, [492](#)
- SetAxisParametersEeprom
  - CMcsBus\_AxisParametersNet, [234](#)
- SetBandwidthByIndex
  - CIntanMea\_FunctionNet, [218](#)
- SetBaseFrequency
  - CRFFFunctionNet, [452](#)
- SetBaseSamplerate
  - CCMOSMeaDeviceNet, [117](#)
  - CGrapheneASICDeviceNet, [174](#)
- SetBath



- CCMOSMea\_FunctionNet, 113
- SetBathclamp
  - COctoPotDeviceNet, 399
- SetBathMode
  - CCMOSMea\_FunctionNet, 113
- SetBlankingEnable
  - CStg200xBasicNet, 544, 545
- SetBuffer
  - CGenericDevelopDeviceNet, 166
- SetBufferIndex
  - CTEERFunctionNet, 612
- SetBusAddress
  - CMcsBusNet, 274
- SetBusAddressEeprom
  - CMcsBusNet, 274
- SetByteBuffer
  - CGenericDevelopDeviceNet, 166
- SetCalibration
  - CTcxDeviceNet, 602
- SetCardinalDacqSamplerate
  - CInterfaceboardFunctionNet, 222
- SetCardinalStgOutputrate
  - CInterfaceboardFunctionNet, 222
- SetChannel
  - CSw2to64DeviceNet, 590
- SetChannelmap
  - CWClassicFunctionNet, 682
- SetChannels
  - CSw2to64DeviceNet, 591
- SetChannelSwitch
  - COctoPotDeviceNet, 399
- SetChargingMode
  - CMultiBatteryChargerDeviceNet, 380
- SetChargingPCoefficient
  - CMultiBatteryChargerDeviceNet, 380
- SetCheckChecksum
  - CMcsUsbDacqNet::ChannelBlockNet, 205
- SetCheckVoltage
  - CokuvisionStimulatorDeviceNet, 403
- SetClampMode
  - CTEERFunctionNet, 613
  - CWarnerUssingFunctionNet, 648
- SetColorRgb
  - CMultiwellOptoStimFunctionNet, 395
- SetColorStr
  - CMultiwellOptoStimFunctionNet, 395
- SetCommand
  - CMcsBusNet, 274, 275
  - CPedoterDeviceNet, 407
  - CRoboDacqNet, 465
- SetCommonThreshold
  - CMcsUsbDacqNet::ChannelBlockNet, 205
- SetConfiguration
  - CMcsUsbNet, 331
- SetConfigurationBit
  - CRoboDacqNet, 465
- SetConfigurationBitAxc
  - CRoboDacqNet, 466
- SetConfigurationBitBlu\_Led
  - CRoboDacqNet, 466
- SetConfigurationBitBlu\_LedToggleFast
  - CRoboDacqNet, 466
- SetConfigurationBitBlu\_LedToggleSlow
  - CRoboDacqNet, 466
- SetConfigurationBitCC\_Gen
  - CRoboDacqNet, 466
- SetConfigurationBitCV\_Gen
  - CRoboDacqNet, 466
- SetConfigurationBitRC\_Gen
  - CRoboDacqNet, 466
- SetConfigurationBitRed\_Led
  - CRoboDacqNet, 466
- SetConfigurationBitRed\_LedSaturation
  - CRoboDacqNet, 466
- SetConfigurationBitRed\_LedToggleFast
  - CRoboDacqNet, 467
- SetConfigurationBitRed\_LedToggleSlow
  - CRoboDacqNet, 467
- SetConfigurationBitRelais
  - CRoboDacqNet, 467
- SetConfigurationBitRV\_Gen
  - CRoboDacqNet, 467
- SetConfigurationBitStream
  - CRoboDacqNet, 467
- SetConfigurationBitSupply
  - CRoboDacqNet, 467
- SetControllerParams
  - CTEERFunctionNet, 613
- SetCrossTalkOffset
  - CRoboDacqNet, 467
- SetCrossTalkOptimum
  - CRoboDacqNet, 467
- SetCurrentAirvalveLimit
  - CRoboDeviceNet, 480
- SetCurrentAndAir
  - CRoboDeviceNet, 480
- SetCurrentAndAirXY
  - CRoboStatorDeviceNet, 496
- SetCurrentEditTableNumber
  - CWarnerValveControllerDeviceNet, 668
- SetCurrentEnable
  - CTEERFunctionNet, 613
- SetCurrentFactor
  - CokuvisionStimulatorDeviceNet, 403
- SetCurrentMode
  - CStg200xBasicNet, 546
- SetCurrentRangeSelectedIndex
  - CStg200xBasicNet, 546
- SetCycles
  - CMeaCleanDeviceNet, 346
  - CMeaCoatDeviceNet, 351
- SetD
  - CTcxDeviceNet, 602
- SetDacAmplificationFactor
  - CStg200xBasicNet, 546
- SetDacAutoControl



- COctoPotDeviceNet, [399](#)
- SetDacIdleValue
  - CLIH3DeviceNet, [231](#)
- SetDacMode
  - CSafeISDeviceNet, [500](#)
- SetDACOffset
  - CGrapheneFunctionNet, [183](#), [184](#)
  - COkuvisionStimulatorDeviceNet, [403](#)
- SetDacOffset
  - CDacCalibrationFunctionNet, [122](#)
  - CLIH3DeviceNet, [231](#)
  - COctoPotDeviceNet, [400](#)
- SetDacOffsetPermanent
  - CLIH3DeviceNet, [231](#)
- SetDacPeriode
  - CSafeISDeviceNet, [500](#)
- SetDacPulseform
  - CSafeISDeviceNet, [500](#)
- SetDacqLegacyMode
  - CSCUFunctionNet, [517](#)
- SetDacRange
  - CW2100\_FunctionNet, [626](#)
- SetDACs
  - CMcsBus\_SensorNet, [262](#)
- SetDacUseldleValue
  - CLIH3DeviceNet, [231](#)
- SetDacValue
  - COctoPotDeviceNet, [400](#)
- SetDataMode
  - CMcsUsbDacqNet, [289](#)
- SetDefault
  - CWarnerValveControllerDeviceNet, [668](#)
- SetDestinationSerialNumber
  - CMcsUsbFactoryNet, [306](#)
- SetDetectionThreshold
  - CMcsBus\_SensorNet, [263](#)
- SetDevice
  - CTcxDeviceNet, [602](#)
- SetDeviceld
  - CUsbDeviceConfigurationFunctionNet, [617](#)
- SetDeviceList
  - CPositionImpDeviceNet, [422](#)
- SetDeviceName
  - CUsbDeviceConfigurationFunctionNet, [617](#)
- SetDeviceType
  - CTcxDeviceNet, [602](#)
- SetDevname
  - CTcxDeviceNet, [602](#)
- SetDiagnosticMode
  - CIntanMea\_FunctionNet, [218](#)
- SetDigitalData
  - CMeaDigitalDataFunctionNet, [362](#), [363](#)
- SetDigitalOut
  - CMeaDeviceNet, [357](#)
- SetDigitalOutPortInvert
  - CWarnerValveControllerDeviceNet, [668](#)
- SetDigitalOutPortValve
  - CWarnerValveControllerDeviceNet, [669](#)
- SetDigitalPortDirection
  - CWarnerValveControllerDeviceNet, [669](#)
- SetDigitalSource
  - CMcsUsbDacqNet, [289–291](#)
- SetDigitalStimulatorTrigger
  - CW2100\_StimulatorFunctionNet, [633](#)
- SetDigitalStimulatorTriggerSlope
  - CW2100\_StimulatorFunctionNet, [633](#)
- SetDigout
  - CFluidControlDeviceNet, [149](#)
  - CRoboDacqNet, [467](#)
- SetDigoutMode
  - CStg200xBasicNet, [548](#)
- SetDigOutState
  - CLIH3DeviceNet, [232](#)
- SetDigoutValue
  - CStg200xBasicNet, [548](#)
- SetDIO
  - CMcsBus\_FYIExtensionNet, [236](#)
- SetDischargeCurrentSetPoint
  - CMultiBatteryChargerDeviceNet, [380](#)
- SetDisplayMode
  - CWarnerValveControllerDeviceNet, [669](#)
- SetDisplayText
  - CRoboDacqNet, [468](#)
- SetDockSmartMultiwellPlatePowerstate
  - CSCUFunctionNet, [518](#)
- SetDockSmartMultiwellPlateWellMux
  - CSCUFunctionNet, [518](#)
- SetDownsampleFactor
  - CRoboDacqNet, [468](#)
- SetDSPHighPassByIndex
  - CIntanMea\_FunctionNet, [219](#)
- SetDuration
  - CMeaCoatDeviceNet, [351](#)
- SetEEPROMPage
  - CLIH3DeviceNet, [232](#)
- SetElectrodeDacMux
  - CStg200xBasicNet, [548–550](#)
- SetElectrodeEnable
  - CStg200xBasicNet, [550–552](#)
- SetElectrodeMode
  - CStg200xBasicNet, [552](#), [554](#), [555](#)
- SetEnableAmplifierProtectionSwitch
  - CStg200xBasicNet, [555](#), [556](#)
- SetEnableHeaterLimit
  - CTcxDeviceNet, [602](#)
- SetEnablePulse
  - CWarnerUssingFunctionNet, [648](#)
- SetEnableThermocouple
  - CTcxDeviceNet, [603](#)
- SetExternalElectrodeEnable
  - CStg200xBasicNet, [557](#)
- SetExternalLED
  - CTEERFunctionNet, [613](#)
- SetFAAmplification
  - CStg200xBasicNet, [557](#)
- SetFilter

- CRoboDacqNet, 468
- SetFilterCoeffs
  - CRoboDacqNet, 468
- SetFilterParameter
  - CFilterConfigurationNet, 140
  - CFilterConfigurationRegisterNet, 142
- SetFilterParameterPermanent
  - CFilterConfigurationNet, 141
  - CFilterConfigurationRegisterNet, 142
- SetFilterParametersHeadstage
  - CWClassicFunctionNet, 682
- SetFinalDischargeVoltage
  - CMultiBatteryChargerDeviceNet, 381
- SetFrequency
  - CChannelTestDeviceNet, 103
  - CRadioControlledDevicesNet, 445
- SetGain
  - CPgaDeviceNet, 411
- SetGate
  - CCMOSMea\_FunctionNet, 113
- SetGateFloating
  - CCMOSMea\_FunctionNet, 113
- SetGateToVOP
  - CCMOSMea\_FunctionNet, 113
- SetGlobalRepeat
  - CDigOutStimulatorFunctionNet, 133
- SetGyroRange
  - CW2100\_FunctionNet, 626
- SetHasChecksum
  - CWClassicFunctionNet, 683
- SetHeadstage
  - CStg200xBasicNet, 558
- SetHeadstageFrameCyclesToComparePermanent
  - CSCUFunctionNet, 518
- SetHeadstageLinkSpeedPermanent
  - CSCUFunctionNet, 519
- SetHeadstageNumberOfAnalogChannelsPermanent
  - CSCUFunctionNet, 519
- SetHeadstageOnOff
  - CW2100\_FunctionNet, 626
  - CWClassicFunctionNet, 683
- SetHeadstagePowerStateAtStart
  - CSCUFunctionNet, 519
- SetHeadstageSampleratePermanent
  - CSCUFunctionNet, 519
- SetHeadstageSamplingActive
  - CW2100\_FunctionNet, 626
- SetHeadstageToSleep
  - CW2100\_FunctionNet, 626
- SetHeaterLimit
  - CTcxDeviceNet, 603
- SetHighCurrentMode
  - CWarnerUssingFunctionNet, 648
- SetHighpassFilterEnable
  - CFilterConfigurationNet, 141
- SetHWConfig
  - CRoboDeviceNet::RoboMainLowLevelCommands, 711
- SetHWRevision
  - CRoboDeviceNet::RoboMainLowLevelCommands, 711
- SetHWRevisionEeprom
  - CMcsBusNet, 275
- SetHWSelectedChannels
  - CWClassicFunctionNet, 683
- SetI
  - CTcxDeviceNet, 603
- SetIClamp
  - CRoboDacqNet, 468
- SetICoeff
  - CRobo\_FYITemp\_FunctionNet, 456
- SetICOffset
  - CRoboDacqNet, 468
- SetIdleModeOffset
  - CWarnerUssingFunctionNet, 649
- SetIGain
  - CRoboDacqNet, 468
- SetImpedanceTestFrequency
  - CMealImpedanceDeviceNet, 369
- SetImpld
  - CPositionImpDeviceNet, 422
- SetImplantCurrentSetpoint
  - CPositionIIDeviceNet, 417
- SetInMovement
  - CRoboDeviceNet, 480
- SetIntanRegister
  - CIntanMea\_FunctionNet, 219
- SetIntBuffer
  - CGenericDevelopDeviceNet, 166
- SetIO
  - CWarnerValveControllerDeviceTesterFunctionNet, 678
- SetIODirection
  - CWarnerValveControllerDeviceTesterFunctionNet, 679
- SetIoVoltage
  - CInterfaceboard2FunctionNet, 220
- SetLatency
  - CMcsBus\_SensorNet, 263
- SetLayoutConfiguration
  - CMEA2100x256FunctionNet, 340
- SetLED
  - CRetinalLedDeviceNet, 447
- SetLEDSwitch
  - CMcsBus\_ExtensionNet, 235
- SetLength
  - CRobo\_FYIProgram\_FunctionNet, 454
- SetLiquidResistance
  - CTEERFunctionNet, 614
  - CWarnerUssingFunctionNet, 649
- SetListmodeIndexRange
  - CStg200xBasicNet, 558
- SetListmodeTriggerSource
  - CStg200xBasicNet, 558, 559
- SetLowCurrentMode
  - CWarnerUssingFunctionNet, 649

SetLumi  
    CRetinaLedDeviceNet, [447](#)  
SetMaxCurrent  
    CMeaCoatDeviceNet, [352](#)  
SetMaxDurationHighCurrentInMicroSec  
    CMultiwellOptoStimFunctionNet, [395](#)  
SetMaxDutyCycleHighCurrent  
    CMultiwellOptoStimFunctionNet, [395](#)  
SetMaxHeaterPowerMultiwell  
    CTcxDeviceNet, [603](#)  
SetMaxNoPressure  
    CRoboDeviceNet::RoboMainLowLevelCommands, [711](#)  
SetMaxNoPressureWaitTime  
    CRoboDeviceNet::RoboMainLowLevelCommands, [711](#)  
SetMaxP  
    CTcxDeviceNet, [603](#)  
SetMaxPower  
    CokuvisionStimulatorDeviceNet, [404](#)  
    CRobo\_FYITemp\_FunctionNet, [456](#)  
SetMaxPressureWaitTime  
    CRoboDeviceNet::RoboMainLowLevelCommands, [711](#)  
SetMaxVoltage  
    CMeaCleanDeviceNet, [346](#)  
    CokuvisionStimulatorDeviceNet, [404](#)  
SetMCAcceleration  
    CMcsBus\_MotorControlNet, [247](#)  
SetMCAccelerationEeprom  
    CMcsBus\_MotorControlNet, [247](#)  
SetMCAccelerationShortCommand  
    CMcsBus\_MotorControlNet, [247](#)  
SetMCAxisRevisionEeprom  
    CMcsBus\_MotorControlNet, [247](#)  
SetMCBreakCurrent  
    CMcsBus\_MotorControlNet, [247](#)  
SetMCBreakCurrentEeprom  
    CMcsBus\_MotorControlNet, [248](#)  
SetMCConfig  
    CMcsBus\_MotorControlNet, [248](#)  
SetMCConfigEeprom  
    CMcsBus\_MotorControlNet, [248](#)  
SetMCCurrent  
    CMcsBus\_MotorControlNet, [248](#)  
SetMCCurrentEeprom  
    CMcsBus\_MotorControlNet, [248](#)  
SetMCCurrentMode  
    CMcsBus\_MotorControlNet, [248](#)  
SetMCCurrentModeEeprom  
    CMcsBus\_MotorControlNet, [249](#)  
SetMCCurrentModeShortCommand  
    CMcsBus\_MotorControlNet, [249](#)  
SetMCCurrentPosition  
    CMcsBus\_MotorControlNet, [249](#)  
SetMCCurrentShortCommand  
    CMcsBus\_MotorControlNet, [249](#)  
SetMCMaxAcceleration  
    CMcsBus\_MotorControlNet, [249](#)  
SetMCMaxAccelerationEeprom  
    CMcsBus\_MotorControlNet, [249](#)  
SetMCMaxCurrent  
    CMcsBus\_MotorControlNet, [250](#)  
SetMCMaxCurrentEeprom  
    CMcsBus\_MotorControlNet, [250](#)  
SetMCMaxSpeed  
    CMcsBus\_MotorControlNet, [250](#)  
SetMCMaxSpeedEeprom  
    CMcsBus\_MotorControlNet, [250](#)  
SetMCMaxTravel  
    CMcsBus\_MotorControlNet, [250](#)  
SetMCMaxTravelEeprom  
    CMcsBus\_MotorControlNet, [250](#)  
SetMCMaxTravelShortCommand  
    CMcsBus\_MotorControlNet, [251](#)  
SetMCNewPosition  
    CMcsBus\_MotorControlNet, [251](#)  
SetMCOutputOnOff  
    CMcsBus\_MotorControlNet, [251](#)  
SetMCReference  
    CMcsBus\_MotorControlNet, [251](#)  
SetMCReferenceCurrent  
    CMcsBus\_MotorControlNet, [251](#)  
SetMCReferenceCurrentEeprom  
    CMcsBus\_MotorControlNet, [251](#)  
SetMCRegulatorGain  
    CMcsBus\_MotorControlNet, [252](#)  
SetMCRegulatorGainEeprom  
    CMcsBus\_MotorControlNet, [252](#)  
SetMCRotation  
    CMcsBus\_MotorControlNet, [252](#)  
SetMCScalingFactor  
    CMcsBus\_MotorControlNet, [252](#)  
SetMCScalingFactorEeprom  
    CMcsBus\_MotorControlNet, [252](#)  
SetMCSpeed  
    CMcsBus\_MotorControlNet, [252](#)  
SetMCSpeedEeprom  
    CMcsBus\_MotorControlNet, [253](#)  
SetMCSpeedShortCommand  
    CMcsBus\_MotorControlNet, [253](#)  
SetMCSpeedUnitEeprom  
    CMcsBus\_MotorControlNet, [253](#)  
SetMCStandbyCurrent  
    CMcsBus\_MotorControlNet, [253](#)  
SetMCStandbyCurrentEeprom  
    CMcsBus\_MotorControlNet, [253](#)  
SetMCStandbyTime  
    CMcsBus\_MotorControlNet, [253](#)  
SetMCStandbyTimeEeprom  
    CMcsBus\_MotorControlNet, [254](#)  
SetMeasurementMode  
    CStg200xBasicNet, [559](#)  
SetMinimalThreshold  
    CMcsBus\_SensorNet, [263](#)  
SetMinNoPressureWaitTime

- CRoboDeviceNet::RoboMainLowLevelCommands, 711
- SetMinPressure
  - CRoboDeviceNet, 481
  - CRoboDeviceNet::RoboMainLowLevelCommands, 711
- SetMinPressureWaitTime
  - CRoboDeviceNet::RoboMainLowLevelCommands, 711
- SetMinVoltage
  - CMeaCleanDeviceNet, 347
- SetModeSelect
  - CPulseGeneratorFunctionNet, 443
- SetMovePump
  - CMcsBus\_SensorNet, 263
- SetMultiHeadstageMode
  - CW2100\_FunctionNet, 626
- SetNanoVoltsPerKelvin
  - CMcsBus\_TempSensorNet, 266
- SetNeurochipMemoryData
  - CCMOSMea\_FunctionNet, 113
- SetNoFilterCoeffs
  - CRoboDacqNet, 468
- SetNumberOfAnalogChannels
  - CMeaDeviceNet, 358
- SetNumberOfChannels
  - CMeaDeviceNet, 359
  - COctoPotDeviceNet, 400
- SetOffsetCurrent
  - CMeaCoatDeviceNet, 352
- SetOnOff
  - CTcxDeviceNet, 603
- SetOutputMap
  - CStg200xDownloadNet, 576
- SetOutputRate
  - COctoPotDeviceNet, 400
  - CStg200xBasicNet, 559
- SetP
  - CTcxDeviceNet, 604
- SetParameter
  - CRoboDeviceNet::RoboMainLowLevelCommands, 712
- SetPattern
  - CMeaSwitchDeviceNet, 371
- SetPatternBool
  - CMeaSwitchDeviceNet, 372
- SetPatternListEntry
  - COctoPotDeviceNet, 400
- SetPauseDuration
  - CMeaCoatDeviceNet, 352
- SetPCoeff
  - CRobo\_FYITemp\_FunctionNet, 456
- SetPeriod
  - CPulseGeneratorFunctionNet, 443
- SetPeriod\_us
  - CTEERFunctionNet, 614
- SetPermanentCurrentInMicroAmp
  - CMultiwellOptoStimFunctionNet, 395
- SetPersistency
  - CRetinaLedDeviceNet, 447
- SetPGain
  - CRoboDacqNet, 468
- SetPidParameter
  - COctoPotDeviceNet, 400
- SetPiezoState
  - CMcsBus\_SensorNet, 263
- SetPlateMux
  - CMultiwellDeviceNet, 389
- SetPlateType
  - CMultiwellDeviceNet, 390
- SetPoti
  - CMcsUsbDacqNet, 292
- SetPowerMuxPlate
  - CMultiwellDeviceNet, 390
- SetPowerStrength
  - CPositionIIDeviceNet, 417
- SetPressureOffset
  - CMcsBus\_SensorNet, 263
  - CPPCFunctionNet, 431
- SetPressureRange
  - CPPCFunctionNet, 431
- SetPulse
  - CWarnerUssingFunctionNet, 649
- SetPulseform
  - COkuvisionStimulatorDeviceNet, 404
- SetPulseLength
  - CPulseGeneratorFunctionNet, 443
- SetPumpCouple
  - CPPS\_FunctionNet, 437
- SetPumpEnableSpeedRatio
  - CPPS\_FunctionNet, 437
- SetPumpFastOnOff
  - CPPS\_FunctionNet, 437
- SetPumpFastSpeed
  - CPPS\_FunctionNet, 437
- SetPumpFunctionSpeeds
  - CPPS\_FunctionNet, 437
- SetPumpManualOnOff
  - CPPS\_FunctionNet, 437
- SetPumpMaxSpeed
  - CPPS\_FunctionNet, 437
- SetPumpModeType
  - CPPCFunctionNet, 432
  - CPPS\_FunctionNet, 437
- SetPumpSpeed
  - CRoboFluidDeviceNet, 489
- SetPumpSpeedRatio
  - CPPS\_FunctionNet, 438
- SetPumpSpeedUnit
  - CPPCFunctionNet, 432
  - CPPS\_FunctionNet, 438
- SetPWM
  - CFluidControlDeviceNet, 149
- SetQueueSize
  - CMcsUsbDacqNet::ChannelBlockNet, 205
- SetQueueSizeAll

- CMcsUsbDacqNet::ChannelBlockNet, [205](#)
- SetRampParameter
  - COctoPotDeviceNet, [400](#)
- SetRatedCapacity
  - CMultiBatteryChargerDeviceNet, [381](#)
- SetRatedCapacityVolatile
  - CMultiBatteryChargerDeviceNet, [381](#)
- SetRecordingNumber
  - CRoboDacqNet, [469](#)
- SetReferenceElectrodeMode
  - CSCUFunctionNet, [520](#)
- SetReferenceElectrodeSwitchState
  - CSCUFunctionNet, [520](#)
- SetRegionOfInterests
  - CCMOSMeaDeviceNet, [117](#)
  - CGrapheneASICDeviceNet, [174](#)
- SetRegulationTimeouts
  - CMcsBus\_SensorNet, [263](#)
- SetRegulatorFactor
  - CMcsBus\_SensorNet, [264](#)
- SetRegulatorOnOff
  - CMcsBus\_SensorNet, [264](#)
  - CRobo\_FYITemp\_FunctionNet, [456](#)
- SetRepeat
  - CRetinaLedDeviceNet, [448](#)
- SetRepeats
  - CProgramPressureCurveNet, [440](#)
- SetResetFilter
  - CWClassicFunctionNet, [683](#)
- SetRFFrequency
  - CPositionImpDeviceNet, [423](#)
- SetRFFrequencyHeadstage
  - CWClassicFunctionNet, [683](#)
- SetRFFrequencyReceiver
  - CWClassicFunctionNet, [683](#)
- SetRFFrequencyReceiverEeprom
  - CWClassicFunctionNet, [683](#)
- SetRFLostBehaviour
  - CWClassicFunctionNet, [683](#)
- SetRFPower
  - CWClassicFunctionNet, [684](#)
- SetRotatePump
  - CMcsBus\_SensorNet, [264](#)
- SetRTC
  - CokuvisionStimulatorDeviceNet, [404](#)
  - CPositionIIDeviceNet, [417](#)
- SetSampleInterval
  - CLIH3DeviceNet, [232](#)
- SetSamplePeriode
  - CMcsBus\_SensorNet, [264](#)
- SetSamplerate
  - CMcsUsbDacqNet, [292](#)
- SetScreen
  - CRoboDacqNet, [469](#)
- SetSearchReferenceFastAccel
  - CRoboDeviceNet::RoboMainLowLevelCommands, [712](#)
- SetSearchReferenceFastSpeed
  - CRoboDeviceNet::RoboMainLowLevelCommands, [712](#)
- SetSearchReferenceFineAccel
  - CRoboDeviceNet::RoboMainLowLevelCommands, [712](#)
- SetSearchReferenceFineSpeed
  - CRoboDeviceNet::RoboMainLowLevelCommands, [712](#)
- SetSearchReferenceMethod
  - CRoboDeviceNet::RoboMainLowLevelCommands, [712](#)
- SetSearchReferenceMoveOut
  - CRoboDeviceNet::RoboMainLowLevelCommands, [713](#)
- SetSearchReferenceOffsetPos
  - CRoboDeviceNet::RoboMainLowLevelCommands, [713](#)
- SetSelectedChannels
  - CMcsUsbDacqNet::ChannelBlockNet, [205](#), [206](#)
  - CW2100\_FunctionNet, [626](#)
- SetSelectedChannelsQueue
  - CMcsUsbDacqNet::ChannelBlockNet, [207](#), [208](#)
- SetSelectedData
  - CMcsUsbDacqNet::ChannelBlockNet, [209](#), [210](#)
- SetSelectedHeadstage
  - CWClassicFunctionNet, [684](#)
- SetSensorType
  - CTcxDeviceNet, [604](#)
- SetSerialNumberHeadstage
  - CWClassicFunctionNet, [684](#)
- SetSetpoint
  - CTcxDeviceNet, [604](#)
- SetShortBuffer
  - CGenericDevelopDeviceNet, [167](#)
- SetSimulation
  - CRoboDacqNet, [469](#)
- SetSineParameter
  - COctoPotDeviceNet, [400](#)
- SetSingleHeater
  - CMcsBus\_FYIExtensionNet, [237](#)
- SetSingleValve
  - CFluidControlDeviceNet, [149](#)
  - CRoboFluidDeviceNet, [489](#)
- SetSlope
  - CMeaCleanDeviceNet, [347](#)
  - CMeaCoatDeviceNet, [352](#)
- SetSoftwareKey
  - CMcsUsbNet, [331](#)
- SetSollPressure
  - CMcsBus\_SensorNet, [264](#)
- SetSollTemp
  - CRobo\_FYITemp\_FunctionNet, [456](#)
- SetSourceBulk
  - CCMOSMea\_FunctionNet, [113](#)
- SetSourceDrain
  - CCMOSMea\_FunctionNet, [114](#)
- SetSourceGate
  - CCMOSMea\_FunctionNet, [114](#)

- SetSpeedI
  - CRoboStatorDeviceNet, [497](#)
- SetSpeedNativeI
  - CRoboStatorDeviceNet, [497](#)
- SetSpeedNativeXY
  - CRoboStatorDeviceNet, [497](#)
- SetSpeedNativeZ
  - CRoboStatorDeviceNet, [497](#)
- SetSpeedXY
  - CRoboStatorDeviceNet, [497](#)
- SetSpeedZ
  - CRoboStatorDeviceNet, [497](#)
- SetStartTriggerSlope
  - CDigOutStimulatorFunctionNet, [133](#)
- SetStateDebugData
  - CPositionIIDeviceNet, [419](#)
- SetStateEventData
  - CPositionIIDeviceNet, [419](#)
- SetStgProgramInfo
  - CStg200xBasicNet, [560](#)
- SetStimulusSites
  - CCMOSMea\_FunctionNet, [114](#)
- SetStopTriggerSlope
  - CDigOutStimulatorFunctionNet, [134](#)
- SetStringFormat
  - CMcsUsbListEntryNet, [313](#)
  - CMcsUsbListNet, [316](#)
- SetSubChannel
  - CMcsBus\_MotorControlNet, [254](#)
- SetSwitches
  - CSafeISDeviceNet, [501](#)
- SetSyncoutMap
  - CStg200xBasicNet, [560](#)
- SetTableName
  - CWarnerValveControllerDeviceNet, [670](#)
- SetTablepointer
  - CRetinaLedDeviceNet, [448](#)
- SetTableStep
  - CWarnerValveControllerDeviceNet, [670](#)
- SetTableStepAll
  - CWarnerValveControllerDeviceNet, [670](#)
- SetTestMode
  - CRFFFunctionNet, [452](#)
- SetThermocoupleNanovoltPerKelvin
  - CFluidControlDeviceNet, [151](#)
  - CTcxDeviceNet, [604](#)
- SetThermoOffset
  - CMcsBus\_TempSensorNet, [266](#)
- Settings
  - Mcs::Usb, [93](#)
- SetTouchPadEnable
  - CMultiwellDeviceNet, [390](#)
- SetTransformer
  - CMeFunctionNet, [374](#)
- SetTrigger
  - CRetinaLedDeviceNet, [448](#)
  - CWarnerValveControllerDeviceTesterFunctionNet, [679](#)
- SetTriggerMaskValue
  - CMeaDeviceNet, [359](#)
  - CRoboDacqNet, [469](#)
- SetTriggerPeriod
  - CMeaDeviceNet, [360](#)
- SetTriggerSource
  - CStg200xBasicNet, [560](#)
- SetTriggerSyncDirection
  - CWarnerValveControllerDeviceTesterFunctionNet, [679](#)
- SetUByteBuffer
  - CGenericDevelopDeviceNet, [168](#)
- SetUClamp
  - CRoboDacqNet, [469](#)
- SetUCOffset
  - CRoboDacqNet, [469](#)
- SetUIntBuffer
  - CGenericDevelopDeviceNet, [168](#)
- SetupGroupDacqQueue
  - CMcsUsbDacqNet::ChannelBlockNet, [211](#)
- SetupRetriggerMode
  - CStg200xDownloadBasicNet, [566](#), [567](#)
- SetupTrigger
  - CStg200xDownloadBasicNet, [567](#)
  - CStimulusFunctionNet, [587](#)
- SetupTriggerSingle
  - CStg200xDownloadBasicNet, [569](#)
  - CStimulusFunctionNet, [588](#)
- SetUseBubble
  - CPPS\_FunctionNet, [438](#)
- SetUserParameter
  - CRoboDeviceNet::RoboMainLowLevelCommands, [713](#)
- SetUShortBuffer
  - CGenericDevelopDeviceNet, [169](#)
- SetUVOffset
  - CRoboDacqNet, [469](#)
- SetValue
  - CGenericDevelopDeviceNet, [169](#)
- SetValve
  - CFluidControlDeviceNet, [151](#)
  - CRoboFluidDeviceNet, [489](#)
- SetValve1
  - CRobo\_FYIProgram\_FunctionNet, [454](#)
- SetValve2
  - CRobo\_FYIProgram\_FunctionNet, [454](#)
- SetValveActive
  - CPPCFunctionNet, [432](#)
  - CWarnerValveControllerDeviceNet, [670](#)
- SetValveCurrent
  - CWarnerValveControllerDeviceNet, [671](#)
- SetValveDigitalInInvert
  - CWarnerValveControllerDeviceNet, [671](#)
- SetValveDigitalInPort
  - CWarnerValveControllerDeviceNet, [671](#)
- SetValveLedOn
  - CWarnerValveControllerDeviceNet, [671](#)
- SetValveManualGroup



- CWarnerValveControllerDeviceNet, 672
- SetValveManualState
  - CWarnerValveControllerDeviceNet, 672
- SetValveMode
  - CWarnerValveControllerDeviceNet, 672
- SetValves
  - CMcsBus\_FYIExtensionNet, 237
- SetValvesActiveMap
  - CWarnerValveControllerDeviceNet, 672
- SetValvesManualStateMap
  - CWarnerValveControllerDeviceNet, 673
- SetValveTableEntry
  - CWarnerValveControllerDeviceNet, 673
- SetVdsVgs
  - CGrapheneFunctionNet, 184
- SetVdVs
  - CGrapheneFunctionNet, 185
- SetVdVsDAC
  - CGrapheneFunctionNet, 185
- SetVelocityI
  - CRoboStatorDeviceNet, 497
- SetVelocityXY
  - CRoboStatorDeviceNet, 497
- SetVelocityZ
  - CRoboStatorDeviceNet, 497
- SetVMMaxNegativeCurrent
  - CMcsBus\_VoltageModeNet, 269
- SetVMMaxNegativeCurrentEeprom
  - CMcsBus\_VoltageModeNet, 270
- SetVMMaxNegativeVoltage
  - CMcsBus\_VoltageModeNet, 270
- SetVMMaxNegativeVoltageEeprom
  - CMcsBus\_VoltageModeNet, 270
- SetVMMaxPositiveCurrent
  - CMcsBus\_VoltageModeNet, 270
- SetVMMaxPositiveCurrentEeprom
  - CMcsBus\_VoltageModeNet, 270
- SetVMMaxPositiveVoltage
  - CMcsBus\_VoltageModeNet, 270
- SetVMMaxPositiveVoltageEeprom
  - CMcsBus\_VoltageModeNet, 271
- SetVMOutputOnOff
  - CMcsBus\_VoltageModeNet, 271
- SetVMVoltage
  - CMcsBus\_VoltageModeNet, 271
- SetVolatileClampOffset
  - CMultiwellDeviceNet, 391
- SetVoltage12VLimit
  - CRoboDeviceNet, 481
- SetVoltage5VLimit
  - CRoboDeviceNet, 481
- SetVoltageAirvalveLimit
  - CRoboDeviceNet, 481
- SetVoltageClampControllerParam\_D
  - CWarnerUssingFunctionNet, 650
- SetVoltageClampControllerParam\_I
  - CWarnerUssingFunctionNet, 650
- SetVoltageClampControllerParam\_P
  - CWarnerUssingFunctionNet, 650
- SetVoltageMode
  - CStg200xBasicNet, 560, 561
- SetVoltageRange
  - CGrapheneFunctionNet, 186
- SetVoltageRangeByIndex
  - CMcsUsbDacqNet, 292
- SetVoltageRangeForIndex
  - CMeaUSBDeviceNet, 373
- SetVoltageRangeSelectedIndex
  - CStg200xBasicNet, 561
- SetVoltageResolution
  - CGrapheneFunctionNet, 186
- SetVoltageRs485ALimit
  - CRoboDeviceNet, 481
- SetVoltageRs485BLimit
  - CRoboDeviceNet, 481
- SetVoltageValvesLimit
  - CRoboDeviceNet, 481
- SetWaveform
  - CChannelTestDeviceNet, 103
  - CTEERFunctionNet, 614
- SetWaveLengthInNanometer
  - CMultiwellOptoStimFunctionNet, 396
- SetWorkingFrequency
  - CRFFFunctionNet, 452
- SetWPADebugMode
  - CWClassicFunctionNet, 684
- SetWPAType
  - CWClassicFunctionNet, 684
- SetXGain
  - CRoboDacqNet, 469
- Sideband
  - CStimulusFunctionNet::SidebandData, 716
- SidebandData
  - CStimulusFunctionNet::SidebandData, 716
- Signed\_16bit
  - Mcs::Usb, 61
- Signed\_24bit
  - Mcs::Usb, 61
- Signed\_32bit
  - Mcs::Usb, 61
- Sine
  - Mcs::Usb, 87
- SineStart
  - COctoPotDeviceNet, 401
- Single
  - Mcs::Usb, 80
- SingleWell
  - Mcs::Usb, 85
- SixWell
  - Mcs::Usb, 85
- size
  - DigitalSource< digitalsourceenum >, 687
  - DigitalSourceGeneral, 689
- SmartImplant
  - Mcs::Usb, 78
- SN

- HeadStageIDType, [701](#)
- SOFAndCTRLword
  - Mcs::Usb, [60](#)
- Software
  - Mcs::Usb, [66](#)
- SoftwareDongle
  - Mcs::Usb, [78](#)
- Source
  - DigitalSource< digitalsourceenum >, [688](#)
  - DigitalSourceGeneral, [689](#)
- Standby
  - Mcs::Usb, [80, 89](#)
- Start
  - CMeaCleanDeviceNet, [347](#)
  - CMeaCoatDeviceNet, [352](#)
  - CRobo\_FYIPProgram\_FunctionNet, [454](#)
  - Mcs::Usb, [64](#)
- StartDacq
  - CMcsUsbDacqNet, [292, 293](#)
- StartInternalCalibration
  - CTEERFunctionNet, [614](#)
- StartLoop
  - CMcsUsbDacqNet, [294, 295](#)
- StartMCMovement
  - CMcsBus\_MotorControlNet, [254](#)
- StartMeasurement
  - CMeaImpedanceDeviceNet, [369](#)
- StartPoll
  - CStimulusFunctionNet, [588](#)
  - CW2100\_StimulatorFunctionNet, [633](#)
- StartQueue
  - CRoboDeviceNet, [481](#)
- StartSampling
  - CTEERFunctionNet, [614](#)
- StartSync
  - CMcsBus\_SensorNet, [264](#)
- State
  - HeadStageIDTypeState, [703](#)
  - Mcs::Usb, [90](#)
- Status
  - CUsbExceptionNet, [618](#)
- Status\_AlreadyConfigured
  - CMcsUsbNet, [333](#)
- Status\_BadStartFrame
  - CMcsUsbNet, [334](#)
- Status\_Btstuff
  - CMcsUsbNet, [334](#)
- Status\_BufferOverrun
  - CMcsUsbNet, [334](#)
- Status\_BufferUnderrun
  - CMcsUsbNet, [334](#)
- Status\_Canceled
  - CMcsUsbNet, [334](#)
- Status\_Canceling
  - CMcsUsbNet, [334](#)
- Status\_ConnectedPipes
  - CMcsUsbNet, [334](#)
- Status\_ControlNotOwned
  - CMcsUsbNet, [334](#)
- Status\_Crc
  - CMcsUsbNet, [334](#)
- Status\_DataOverrun
  - CMcsUsbNet, [334](#)
- Status\_DataToggleMismatch
  - CMcsUsbNet, [334](#)
- Status\_DataUnderrun
  - CMcsUsbNet, [335](#)
- Status\_DeviceLocked
  - CMcsUsbNet, [335](#)
- Status\_DeviceNotFound
  - CMcsUsbNet, [335](#)
- Status\_DeviceRemoved
  - CMcsUsbNet, [335](#)
- Status\_DevNotResponding
  - CMcsUsbNet, [335](#)
- Status\_EndpointHalted
  - CMcsUsbNet, [335](#)
- Status\_ErrorBusy
  - CMcsUsbNet, [335](#)
- Status\_ErrorShortTransfer
  - CMcsUsbNet, [335](#)
- Status\_Fifo
  - CMcsUsbNet, [335](#)
- Status\_FrameControlOwned
  - CMcsUsbNet, [335](#)
- Status\_InternalHcError
  - CMcsUsbNet, [335](#)
- Status\_InvalidDeviceHandle
  - CMcsUsbNet, [336](#)
- Status\_InvalidHandle
  - CMcsUsbNet, [336](#)
- Status\_InvalidParameter
  - CMcsUsbNet, [336](#)
- Status\_InvalidPipeHandle
  - CMcsUsbNet, [336](#)
- Status\_InvalidUrbFunction
  - CMcsUsbNet, [336](#)
- Status\_IoPending
  - CMcsUsbNet, [336](#)
- Status\_IoTimeout
  - CMcsUsbNet, [336](#)
- Status\_IsochRequestFailed
  - CMcsUsbNet, [336](#)
- Status\_LastUsbErrorMismatch
  - CMcsUsbNet, [336](#)
- Status\_NoBandwidth
  - CMcsUsbNet, [336](#)
- Status\_NoMemory
  - CMcsUsbNet, [337](#)
- Status\_NoSuchDevice
  - CMcsUsbNet, [337](#)
- Status\_NotAccessed
  - CMcsUsbNet, [337](#)
- Status\_NotSupported
  - CMcsUsbNet, [337](#)
- Status\_PidCheckFailure



- CMcsUsbNet, [337](#)
- Status\_PipeNotLinked
  - CMcsUsbNet, [337](#)
- Status\_RequestFailed
  - CMcsUsbNet, [337](#)
- Status\_RequestMutexFailed
  - CMcsUsbNet, [337](#)
- Status\_RequestMutexTimeout
  - CMcsUsbNet, [337](#)
- Status\_Stall
  - CMcsUsbNet, [337](#)
- Status\_Unconfigured
  - CMcsUsbNet, [337](#)
- Status\_UnexpectedPid
  - CMcsUsbNet, [338](#)
- STG
  - Mcs::Usb, [76](#)
- Stg1
  - Mcs::Usb, [64](#)
- STG1DACSignalGroup
  - Mcs::Usb, [61](#), [70](#), [81](#)
- STG1SidebandsGroup
  - Mcs::Usb, [61](#), [70](#), [81](#)
- STG1TriggerStatusGroup
  - Mcs::Usb, [61](#), [70](#), [81](#)
- Stg2
  - Mcs::Usb, [64](#)
- Stg200xDigoutModeEnumNet
  - Mcs::Usb, [83](#)
- Stg200xMultiFileSubmodeEnumNet
  - Mcs::Usb, [83](#)
- Stg200xPollStatusEvent
  - CStg200xDownloadNet, [576](#)
- Stg200xSegmentFlagsEnumNet
  - Mcs::Usb, [84](#)
- Stg200xTriggerStatusEnumNet
  - Mcs::Usb, [84](#)
- STG2DACSignalGroup
  - Mcs::Usb, [70](#), [81](#)
- STG2SidebandsGroup
  - Mcs::Usb, [70](#), [81](#)
- STG2TriggerStatusGroup
  - Mcs::Usb, [70](#), [81](#)
- Stg3
  - Mcs::Usb, [64](#)
- STG3008\_FA
  - Mcs::Usb, [77](#)
- Stg3008FilterAmpAmplificationEnumNet
  - Mcs::Usb, [84](#)
- STG4002
  - Mcs::Usb, [76](#)
- STG4002\_opto
  - Mcs::Usb, [76](#)
- STG4004
  - Mcs::Usb, [76](#)
- STG4004\_opto
  - Mcs::Usb, [76](#)
- STG4008
  - Mcs::Usb, [76](#)
- STG4008\_opto
  - Mcs::Usb, [76](#)
- STG5
  - Mcs::Usb, [76](#)
- STG\_DestinationEnumNet
  - Mcs::Usb, [84](#)
- StgListModeTrigger
  - Mcs::Usb, [64](#)
- StgStatusNet, [717](#)
  - FromIntPtr, [717](#)
  - FromPtr, [717](#)
  - ListOfChangedTriggers, [717](#)
  - TiggerStatus, [717](#)
- StgTrigger
  - Mcs::Usb, [64](#)
- StillConnected
  - CRadioControlledDevicesNet, [445](#)
- Stimulation
  - Mcs::Usb, [58](#)
- StimulationLayoutConfigurationEnumNet
  - Mcs::Usb, [85](#)
- Stimulator
  - CW2100\_FunctionNet, [627](#)
- Stimulus
  - CCMOSMeaDeviceNet, [117](#)
  - CStg200xDownloadBasicNet, [570](#)
- StimulusDeviceDataAndUnrolledData
  - CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, [718](#)
- StimulusFunction
  - CLIH3DeviceNet, [233](#)
- StimulusParameters
  - HeadStageIDType, [701](#)
- Stop
  - CMeaCleanDeviceNet, [347](#)
  - CMeaCoatDeviceNet, [353](#)
  - Mcs::Usb, [64](#), [75](#), [80](#)
- StopDacq
  - CMcsUsbDacqNet, [296](#)
- StopLoop
  - CMcsUsbDacqNet, [296](#)
- StopMCMovement
  - CMcsBus\_MotorControlNet, [254](#)
- StopMovement
  - CRoboDeviceNet, [482](#)
- StopMovementI
  - CRoboStatorDeviceNet, [498](#)
- StopMovementXY
  - CRoboStatorDeviceNet, [498](#)
- StopMovementZ
  - CRoboStatorDeviceNet, [498](#)
- StopPlateClamp
  - CMultiwellDeviceNet, [391](#)

- StopPoll
  - CStimulusFunctionNet, 588
  - CW2100\_StimulatorFunctionNet, 633
- StopSampling
  - CTEERFunctionNet, 615
- StopTable
  - CRoboDacqNet, 469, 470
- StorageCharge
  - Mcs::Usb, 69
- StoreValveTable
  - CWarnerValveControllerDeviceNet, 673
- SubtractFromAll
  - Mcs::Usb, 79
- SubtractFromAllOther
  - Mcs::Usb, 79
- SubtractFromReferenceElectrodeOnly
  - Mcs::Usb, 79
- SubtractionOff
  - Mcs::Usb, 79
- SuperSpeed
  - Mcs::Usb, 70
- Sw2to64
  - Mcs::Usb, 77
- SwitchOnOff
  - CPositionIIDeviceNet, 419
- SYNC\_BIT0
  - CW2100\_StimulatorFunctionNet, 633
- SYNC\_BIT1
  - CW2100\_StimulatorFunctionNet, 633
- SYNCOUT1
  - Mcs::Usb, 83
- SYNCOUT2
  - Mcs::Usb, 83
- SYNCOUT3
  - Mcs::Usb, 83
- SYNCOUT4
  - Mcs::Usb, 83
- SYNCOUT5
  - Mcs::Usb, 83
- SYNCOUT6
  - Mcs::Usb, 83
- SYNCOUT7
  - Mcs::Usb, 83
- SYNCOUT8
  - Mcs::Usb, 83
- syncoutdata
  - Mcs::Usb, 85
- SyncStart
  - Mcs::Usb, 84
- Table
  - Mcs::Usb, 93
- Table\_Wait
  - CRoboDacqNet, 470
- TableDefBegin
  - CRoboDacqNet, 470
- TableDefEnd
  - CRoboDacqNet, 470
- TableEntryChangedEvent
  - CWarnerValveControllerDeviceNet, 676
- TactSwitchGetState
  - CMcsBus\_SensorNet, 264
- TactSwitchSetDisplay
  - CMcsBus\_SensorNet, 265
- TBSI\_127
  - Mcs::Usb, 52
- TBSI\_15
  - Mcs::Usb, 52
- TBSI\_31
  - Mcs::Usb, 52
- TBSI\_5
  - Mcs::Usb, 52
- TBSI\_63
  - Mcs::Usb, 52
- TBSI\_Dacq
  - Mcs::Usb, 78
- TBSI\_DACQDigitalSourceEnumNet
  - Mcs::Usb, 85
- TBSI\_Reserved
  - Mcs::Usb, 52
- TbsiDacq
  - Mcs::Usb, 68
- TbsiDacqHeadstage
  - Mcs::Usb, 67
- TbsiDacqInterfaceboard
  - Mcs::Usb, 67
- TC01
  - Mcs::Usb, 77
- TC02
  - Mcs::Usb, 77
- TCX
  - Mcs::Usb, 77
- TcxDeviceTypeEnumNet
  - Mcs::Usb, 86
- TcxSensorTypeEnumNet
  - Mcs::Usb, 86
- TeerClampModeEnumNet
  - Mcs::Usb, 86
- TEERFunctionNet
  - CTEERMachineDeviceNet, 616
- TeerWaveformEnumNet
  - Mcs::Usb, 86
- Tersens
  - Mcs::Usb, 76
- ThrowCUsbExceptionNetOnError
  - CMcsUsbFunctionNet, 309
  - CMcsUsbNet, 331
- TiggerStatus
  - StgStatusNet, 717
- TimeResolutionInNanoSeconds
  - W2100\_StimulusParametersNet, 720
- Timestamp
  - Mcs::Usb, 60
- ToCpp
  - CFilterCoefficientsNet::s\_FilterAttributesNet, 714
- ToString
  - CFilterPropertyNet, 143

- CMcsUsbListEntryNet, [313](#)
- HeadStageIDType, [700](#)
- HeadstageIDTypeObject, [702](#)
- TouchTest
  - Mcs::Usb, [93](#)
- Triggerbox\_AMS
  - Mcs::Usb, [77](#)
- Triggerbox\_AMS3
  - Mcs::Usb, [77](#)
- Triggerbox\_IMS
  - Mcs::Usb, [77](#)
- Triggerbox\_R5
  - Mcs::Usb, [77](#)
- TriggerMask\_Default
  - CRoboDacqNet, [470](#)
- TriggerOnly
  - Mcs::Usb, [84](#)
- TriggerSourceEnumNet
  - Mcs::Usb, [87](#)
- TriggerStatus
  - CMcsUsbDeviceStatePushFunctionNet, [298](#)
  - CMcsUsbDeviceStatePushNet, [300](#)
- TriggerStatus1
  - Mcs::Usb, [59](#)
- TriggerStatus2
  - Mcs::Usb, [59](#)
- TriggerStatus3
  - Mcs::Usb, [59](#)
- TriggerStatus4
  - Mcs::Usb, [59](#)
- TriggerValue\_MoveAbs
  - CRoboDacqNet, [470](#)
- TriggerValue\_StartQueue
  - CRoboDacqNet, [470](#)
- tsAuxIn1
  - Mcs::Usb, [88](#)
- tsAuxIn2
  - Mcs::Usb, [88](#)
- tsDACQCy1Dev1Runs
  - Mcs::Usb, [89](#)
- tsDACQCy1Dev2Runs
  - Mcs::Usb, [89](#)
- tsDACQCy2Dev1Runs
  - Mcs::Usb, [89](#)
- tsDACQCy2Dev2Runs
  - Mcs::Usb, [89](#)
- tsDigitalIn1
  - Mcs::Usb, [87](#)
- tsDigitalIn10
  - Mcs::Usb, [87](#)
- tsDigitalIn11
  - Mcs::Usb, [87](#)
- tsDigitalIn12
  - Mcs::Usb, [87](#)
- tsDigitalIn13
  - Mcs::Usb, [87](#)
- tsDigitalIn14
  - Mcs::Usb, [87](#)
- tsDigitalIn15
  - Mcs::Usb, [87](#)
- tsDigitalIn16
  - Mcs::Usb, [87](#)
- tsDigitalIn17
  - Mcs::Usb, [87](#)
- tsDigitalIn18
  - Mcs::Usb, [87](#)
- tsDigitalIn19
  - Mcs::Usb, [87](#)
- tsDigitalIn2
  - Mcs::Usb, [87](#)
- tsDigitalIn20
  - Mcs::Usb, [87](#)
- tsDigitalIn21
  - Mcs::Usb, [87](#)
- tsDigitalIn22
  - Mcs::Usb, [87](#)
- tsDigitalIn23
  - Mcs::Usb, [87](#)
- tsDigitalIn24
  - Mcs::Usb, [87](#)
- tsDigitalIn25
  - Mcs::Usb, [87](#)
- tsDigitalIn26
  - Mcs::Usb, [87](#)
- tsDigitalIn27
  - Mcs::Usb, [87](#)
- tsDigitalIn28
  - Mcs::Usb, [87](#)
- tsDigitalIn29
  - Mcs::Usb, [87](#)
- tsDigitalIn3
  - Mcs::Usb, [87](#)
- tsDigitalIn30
  - Mcs::Usb, [87](#)
- tsDigitalIn31
  - Mcs::Usb, [87](#)
- tsDigitalIn32
  - Mcs::Usb, [87](#)
- tsDigitalIn4
  - Mcs::Usb, [87](#)
- tsDigitalIn5
  - Mcs::Usb, [87](#)
- tsDigitalIn6
  - Mcs::Usb, [87](#)
- tsDigitalIn7
  - Mcs::Usb, [87](#)
- tsDigitalIn8
  - Mcs::Usb, [87](#)
- tsDigitalIn9
  - Mcs::Usb, [87](#)
- tsDigitalPuse0
  - Mcs::Usb, [88](#)
- tsDigitalPuse1
  - Mcs::Usb, [88](#)
- tsDigitalPuse10
  - Mcs::Usb, [88](#)

tsDigitalPuse11  
Mcs::Usb, [88](#)  
tsDigitalPuse12  
Mcs::Usb, [88](#)  
tsDigitalPuse13  
Mcs::Usb, [88](#)  
tsDigitalPuse14  
Mcs::Usb, [88](#)  
tsDigitalPuse15  
Mcs::Usb, [88](#)  
tsDigitalPuse16  
Mcs::Usb, [88](#)  
tsDigitalPuse17  
Mcs::Usb, [88](#)  
tsDigitalPuse18  
Mcs::Usb, [89](#)  
tsDigitalPuse19  
Mcs::Usb, [89](#)  
tsDigitalPuse2  
Mcs::Usb, [88](#)  
tsDigitalPuse20  
Mcs::Usb, [89](#)  
tsDigitalPuse21  
Mcs::Usb, [89](#)  
tsDigitalPuse22  
Mcs::Usb, [89](#)  
tsDigitalPuse23  
Mcs::Usb, [89](#)  
tsDigitalPuse24  
Mcs::Usb, [89](#)  
tsDigitalPuse25  
Mcs::Usb, [89](#)  
tsDigitalPuse26  
Mcs::Usb, [89](#)  
tsDigitalPuse27  
Mcs::Usb, [89](#)  
tsDigitalPuse28  
Mcs::Usb, [89](#)  
tsDigitalPuse29  
Mcs::Usb, [89](#)  
tsDigitalPuse3  
Mcs::Usb, [88](#)  
tsDigitalPuse30  
Mcs::Usb, [89](#)  
tsDigitalPuse31  
Mcs::Usb, [89](#)  
tsDigitalPuse4  
Mcs::Usb, [88](#)  
tsDigitalPuse5  
Mcs::Usb, [88](#)  
tsDigitalPuse6  
Mcs::Usb, [88](#)  
tsDigitalPuse7  
Mcs::Usb, [88](#)  
tsDigitalPuse8  
Mcs::Usb, [88](#)  
tsDigitalPuse9  
Mcs::Usb, [88](#)  
tsFeedback1  
Mcs::Usb, [87](#)  
tsFeedback10  
Mcs::Usb, [88](#)  
tsFeedback11  
Mcs::Usb, [88](#)  
tsFeedback12  
Mcs::Usb, [88](#)  
tsFeedback13  
Mcs::Usb, [88](#)  
tsFeedback14  
Mcs::Usb, [88](#)  
tsFeedback15  
Mcs::Usb, [88](#)  
tsFeedback16  
Mcs::Usb, [88](#)  
tsFeedback17  
Mcs::Usb, [88](#)  
tsFeedback18  
Mcs::Usb, [88](#)  
tsFeedback19  
Mcs::Usb, [88](#)  
tsFeedback2  
Mcs::Usb, [87](#)  
tsFeedback20  
Mcs::Usb, [88](#)  
tsFeedback21  
Mcs::Usb, [88](#)  
tsFeedback22  
Mcs::Usb, [88](#)  
tsFeedback23  
Mcs::Usb, [88](#)  
tsFeedback24  
Mcs::Usb, [88](#)  
tsFeedback25  
Mcs::Usb, [88](#)  
tsFeedback26  
Mcs::Usb, [88](#)  
tsFeedback27  
Mcs::Usb, [88](#)  
tsFeedback28  
Mcs::Usb, [88](#)  
tsFeedback29  
Mcs::Usb, [88](#)  
tsFeedback3  
Mcs::Usb, [87](#)  
tsFeedback30  
Mcs::Usb, [88](#)  
tsFeedback31  
Mcs::Usb, [88](#)  
tsFeedback32  
Mcs::Usb, [88](#)  
tsFeedback4  
Mcs::Usb, [88](#)  
tsFeedback5  
Mcs::Usb, [88](#)  
tsFeedback6  
Mcs::Usb, [88](#)

- tsFeedback7
  - Mcs::Usb, [88](#)
- tsFeedback8
  - Mcs::Usb, [88](#)
- tsFeedback9
  - Mcs::Usb, [88](#)
- tsNone
  - Mcs::Usb, [87](#)
- tsSidebandBit8
  - Mcs::Usb, [89](#)
- tsTriggered
  - Mcs::Usb, [89](#)
- TxnGetSerialNumber
  - CMcsUsbNet, [331](#)
- TxnSetSerialNumber
  - CMcsUsbNet, [331](#)
- TxnTestMemoryReadAndCheck
  - CMcsUsbNet, [331](#)
- TxnTestMemoryWrite
  - CMcsUsbNet, [332](#)
- Type
  - HeadStageIDType, [701](#)
- TypeValue
  - HeadStageIDType, [701](#)
- Unknown
  - HeadStageIDType, [699](#)
  - Mcs::Usb, [52](#), [53](#), [66](#), [86](#), [89](#)
- unknown
  - Mcs::Usb, [65](#)
- UnknownDest
  - Mcs::Usb, [57](#)
- UnknownSpeed
  - Mcs::Usb, [70](#)
- Unlock
  - Mcs::Usb, [75](#)
- UnlockPlateClamp
  - CMultiwellDeviceNet, [391](#)
- UnrolledAmplitude
  - CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, [718](#)
- UnrolledDuration
  - CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, [718](#)
- UnrolledSync
  - CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, [718](#)
- Unsigned\_16bit
  - Mcs::Usb, [61](#)
- Unsigned\_24bit
  - Mcs::Usb, [61](#)
- Unsigned\_32bit
  - Mcs::Usb, [61](#)
- UpdateChannelBlock
  - CCMOSMeaDeviceNet, [117](#)
- UpdateDisplay
  - CRoboDacqNet, [470](#)
- UpdateFirmware
  - CMcsUsbFactoryNet, [306](#), [307](#)
- UpdateQueueSize
  - CMcsUsbDacqNet::ChannelBlockNet, [211](#)
- UpdateTransistorVoltages
  - CCMOSMea\_FunctionNet, [114](#)
- UpdateTrigger
  - Mcs::Usb, [84](#)
- USB
  - FirmwareDestinationNames, [698](#)
  - Mcs::Usb, [53](#)
- USB\_TARGET1
  - Mcs::Usb, [57](#)
- USB\_TARGET2
  - Mcs::Usb, [57](#)
- USB\_TARGET3
  - Mcs::Usb, [57](#)
- usbSetupPacket\_t, [719](#)
  - bmRequestType, [719](#)
  - bRequest, [719](#)
  - wIndex, [719](#)
  - wLength, [719](#)
  - wValue, [719](#)
- UsbTest
  - Mcs::Usb, [78](#)
- UsbVendorIdEnumNet
  - Mcs::Usb, [89](#)
- UserDefinedName
  - HeadStageIDType, [701](#)
- UssingChamber
  - Mcs::Usb, [67](#)
- UssingClampModeEnumNet
  - Mcs::Usb, [89](#)
- UssingRail
  - Mcs::Usb, [67](#)
- UssingUnitEnumNet
  - Mcs::Usb, [90](#)
- Valid
  - HeadStageIDType, [701](#)
- ValidKey
  - CMcsUsbNet, [332](#)
- VendorIdEnumNet
  - Mcs::Usb, [90](#)
- VendorInRequest
  - CGenericDevelopDeviceNet, [170](#)
- VendorOutRequest
  - CGenericDevelopDeviceNet, [170](#)
- VirtualDevice\_ContinuousDacq
  - CRoboDacqNet, [470](#)
- VirtualDevice\_TableRun
  - CRoboDacqNet, [470](#)
- Volt
  - Mcs::Usb, [53](#), [90](#)
- Voltage
  - BatteryState, [96](#)
- Voltage\_3V3
  - Mcs::Usb, [68](#)
- Voltage\_5V0
  - Mcs::Usb, [68](#)
- VoltageClamp

- Mcs::Usb, [89](#)
- VoltageRangeDisplayStringMilliVolt
  - CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet, [619](#)
- VoltageRangeInMicroVolt
  - CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet, [619](#)
- W2100\_StimulusParametersNet, [720](#)
- VoltageResolutionInMicroVolt
  - W2100\_StimulusParametersNet, [720](#)
- VoltageString
  - BatteryState, [96](#)
- VOPSTimerSetResetTimes
  - CCMOSMea\_FunctionNet, [114](#)
- W16IsW14
  - HeadStageIDType, [701](#)
- W2100
  - Mcs::Usb, [78](#)
- W2100\_Accel\_Gyro\_Select\_EnumNet
  - Mcs::Usb, [90](#)
- W2100\_FunctionNet
  - CMeaDeviceNet, [361](#)
- W2100\_StimulusParametersNet, [719](#)
  - CurrentRangeInNanoAmp, [720](#)
  - CurrentResolutionInNanoAmp, [720](#)
  - DACResolution, [720](#)
  - TimeResolutionInNanoSeconds, [720](#)
  - VoltageRangeInMicroVolt, [720](#)
  - VoltageResolutionInMicroVolt, [720](#)
- W2100DacqGroupChannelEnumNet
  - Mcs::Usb, [90](#)
- W2100DigitalSourceEnumNet
  - Mcs::Usb, [92](#)
- W2100IFB2
  - Mcs::Usb, [67](#)
- W2100Interfaceboard
  - Mcs::Usb, [67](#)
- W2100WirelessReceiver
  - Mcs::Usb, [67](#), [68](#)
- W2100WirelessReceiverAnalog
  - Mcs::Usb, [67](#), [68](#)
- WaitForAllChambers
  - CWarnerUssingFunctionNet, [651](#)
- WaitForChamber
  - CWarnerUssingFunctionNet, [651](#)
- WaitTimer
  - CRoboDeviceNet, [482](#)
- Warner
  - Mcs::Usb, [86](#)
- Warner\_TEER\_Machine
  - Mcs::Usb, [79](#)
- Warner\_Ussing
  - Mcs::Usb, [79](#)
- WARNER\_USSING\_DEVICE
  - Mcs::Usb, [63](#)
- Warner\_Valve\_Control
  - Mcs::Usb, [78](#)
- WARNER\_VALVE\_CONTROL\_DEVICE
  - Mcs::Usb, [63](#)
- WarnerUssingFunction
  - CWarnerUssingDeviceNet, [635](#)
  - WasInternalCalibrationSuccessful
    - CWarnerUssingFunctionNet, [651](#)
  - WClassicFunctionNet
    - CMeaDeviceNet, [361](#)
- Whole\_Cell\_Patch
  - Mcs::Usb, [78](#)
- WholeCellPatch
  - Mcs::Usb, [81](#)
- WholeCellPatchHeadstage
  - Mcs::Usb, [67](#)
- WidthFrequency
  - CCreateFilterNet, [120](#)
- wIndex
  - usbSetupPacket\_t, [719](#)
- WirelessHeadStageAccDataRE1HS1
  - Mcs::Usb, [91](#)
- WirelessHeadStageAccDataRE1HS2
  - Mcs::Usb, [91](#)
- WirelessHeadStageAccDataRE1HS3
  - Mcs::Usb, [91](#)
- WirelessHeadStageAccDataRE1HS4
  - Mcs::Usb, [91](#)
- WirelessHeadStageAccDataRE2HS1
  - Mcs::Usb, [91](#)
- WirelessHeadStageAccDataRE2HS2
  - Mcs::Usb, [91](#)
- WirelessHeadStageAccDataRE2HS3
  - Mcs::Usb, [91](#)
- WirelessHeadStageAccDataRE2HS4
  - Mcs::Usb, [91](#)
- WirelessHeadStageAnalogRE1HS1
  - Mcs::Usb, [91](#)
- WirelessHeadStageAnalogRE1HS2
  - Mcs::Usb, [91](#)
- WirelessHeadStageAnalogRE1HS3
  - Mcs::Usb, [91](#)
- WirelessHeadStageAnalogRE1HS4
  - Mcs::Usb, [91](#)
- WirelessHeadStageAnalogRE2HS1
  - Mcs::Usb, [91](#)
- WirelessHeadStageAnalogRE2HS2
  - Mcs::Usb, [91](#)
- WirelessHeadStageAnalogRE2HS3
  - Mcs::Usb, [91](#)
- WirelessHeadStageAnalogRE2HS4
  - Mcs::Usb, [91](#)
- WirelessHeadStageGyroDataRE1HS1
  - Mcs::Usb, [91](#)
- WirelessHeadStageGyroDataRE1HS2
  - Mcs::Usb, [91](#)
- WirelessHeadStageGyroDataRE1HS3
  - Mcs::Usb, [91](#)
- WirelessHeadStageGyroDataRE1HS4
  - Mcs::Usb, [91](#)
- WirelessHeadStageGyroDataRE2HS1

- Mcs::Usb, [91](#)
- WirelessHeadStageGyroDataRE2HS2
  - Mcs::Usb, [91](#)
- WirelessHeadStageGyroDataRE2HS3
  - Mcs::Usb, [91](#)
- WirelessHeadStageGyroDataRE2HS4
  - Mcs::Usb, [91](#)
- WirelessHeadStageOptoStimCurrentRE1HS1
  - Mcs::Usb, [91](#)
- WirelessHeadStageOptoStimCurrentRE1HS2
  - Mcs::Usb, [91](#)
- WirelessHeadStageOptoStimCurrentRE1HS3
  - Mcs::Usb, [91](#)
- WirelessHeadStageOptoStimCurrentRE1HS4
  - Mcs::Usb, [91](#)
- WirelessHeadStageOptoStimCurrentRE2HS1
  - Mcs::Usb, [91](#)
- WirelessHeadStageOptoStimCurrentRE2HS2
  - Mcs::Usb, [91](#)
- WirelessHeadStageOptoStimCurrentRE2HS3
  - Mcs::Usb, [91](#)
- WirelessHeadStageOptoStimCurrentRE2HS4
  - Mcs::Usb, [92](#)
- WirelessHeadStageReservedARE1HS1
  - Mcs::Usb, [91](#)
- WirelessHeadStageReservedARE1HS2
  - Mcs::Usb, [91](#)
- WirelessHeadStageReservedARE1HS3
  - Mcs::Usb, [91](#)
- WirelessHeadStageReservedARE1HS4
  - Mcs::Usb, [91](#)
- WirelessHeadStageReservedARE2HS1
  - Mcs::Usb, [91](#)
- WirelessHeadStageReservedARE2HS2
  - Mcs::Usb, [91](#)
- WirelessHeadStageReservedARE2HS3
  - Mcs::Usb, [92](#)
- WirelessHeadStageReservedARE2HS4
  - Mcs::Usb, [92](#)
- WirelessHeadStageReservedBRE1HS1
  - Mcs::Usb, [92](#)
- WirelessHeadStageReservedBRE1HS2
  - Mcs::Usb, [92](#)
- WirelessHeadStageReservedBRE1HS3
  - Mcs::Usb, [92](#)
- WirelessHeadStageReservedBRE1HS4
  - Mcs::Usb, [92](#)
- WirelessHeadStageReservedBRE2HS1
  - Mcs::Usb, [92](#)
- WirelessHeadStageReservedBRE2HS2
  - Mcs::Usb, [92](#)
- WirelessHeadStageReservedBRE2HS3
  - Mcs::Usb, [92](#)
- WirelessHeadStageReservedBRE2HS4
  - Mcs::Usb, [92](#)
- WirelessHeadStageReservedCRE1HS1
  - Mcs::Usb, [92](#)
- WirelessHeadStageReservedCRE1HS2
  - Mcs::Usb, [92](#)
- WirelessHeadStageReservedCRE1HS3
  - Mcs::Usb, [92](#)
- WirelessHeadStageReservedCRE1HS4
  - Mcs::Usb, [92](#)
- WirelessHeadStageReservedCRE2HS1
  - Mcs::Usb, [92](#)
- WirelessHeadStageReservedCRE2HS2
  - Mcs::Usb, [92](#)
- WirelessHeadStageReservedCRE2HS3
  - Mcs::Usb, [92](#)
- WirelessHeadStageReservedCRE2HS4
  - Mcs::Usb, [92](#)
- WirelessHeadStageStatusRE1HS1
  - Mcs::Usb, [91](#)
- WirelessHeadStageStatusRE1HS2
  - Mcs::Usb, [91](#)
- WirelessHeadStageStatusRE1HS3
  - Mcs::Usb, [91](#)
- WirelessHeadStageStatusRE1HS4
  - Mcs::Usb, [91](#)
- WirelessHeadStageStatusRE2HS1
  - Mcs::Usb, [91](#)
- WirelessHeadStageStatusRE2HS2
  - Mcs::Usb, [91](#)
- WirelessHeadStageStatusRE2HS3
  - Mcs::Usb, [91](#)
- WirelessHeadStageStatusRE2HS4
  - Mcs::Usb, [91](#)
- WirelessTestAdapter
  - Mcs::Usb, [52](#)
- wLength
  - usbSetupPacket\_t, [719](#)
- Work
  - Mcs::Usb, [93](#)
- WPA16
  - Mcs::Usb, [78](#)
- WPA32
  - Mcs::Usb, [78](#)
- WPA4
  - Mcs::Usb, [78](#)
- WPA8
  - Mcs::Usb, [78](#)
- WPAError\_ScanningIsPending
  - CMcsUsbNet, [338](#)
- Write
  - CExternDTesterDeviceNet, [136](#)
- Write2
  - CExternDTesterDeviceNet, [137](#)
- WriteEepromRegisterPreconfig
  - CMcsUsbNet, [332](#)
- WritePipe
  - CGenericDevelopDeviceNet, [170](#)
- WriteRegister
  - CMcsUsbNet, [332](#), [333](#)
- WriteRegister32
  - CMcsUsbNet, [333](#)
- WriteRegisterArray

- CMcsUsbNet, [333](#)
- WriteRegisterTimeSlot
  - CMcsUsbNet, [333](#)
- WriteRegisterValue
  - CMcsUsbNet, [333](#)
- WriteUARTData
  - CLIH3DeviceNet, [232](#)
- wValue
  - usbSetupPacket\_t, [719](#)
- WvcDisplayModeEnumNet
  - Mcs::Usb, [93](#)
- WvcValveModeEnumNet
  - Mcs::Usb, [93](#)
- Zero
  - Mcs::Usb, [57](#), [70](#), [72](#), [74](#), [82](#), [85](#), [92](#)