

multichannel* systems

McsUsbNet.dll

Version 5.1.28

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	1
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	2
2.2 MCS-USB-Sw2to64 device	3
3 Function Classes	3
4 Data ACQuisition (DACQ) Devices	4
5 The MCS Robo Device	5
5.1 Introduction	5
6 STG200x & STG400x STimulus Generator	5
6.1 Introduction	5
6.2 Download mode	6
6.2.1 Memory Layout and Trigger Setup	6
6.3 Streaming mode	8
6.3.1 Memory Layout and Trigger Setup	9
7 Namespace Index	11
7.1 Namespace List	11
8 Hierarchical Index	11
8.1 Class Hierarchy	11
9 Class Index	16
9.1 Class List	16
10 Namespace Documentation	22
10.1 Mcs Namespace Reference	22
10.2 Mcs::Usb Namespace Reference	22
10.2.1 Enumeration Type Documentation	51
10.2.2 Function Documentation	92
11 Class Documentation	94
11.1 CW2100_FunctionNet::AudioChannelsNet Struct Reference	94
11.1.1 Member Data Documentation	94
11.2 BatteryState Class Reference	94
11.2.1 Property Documentation	94
11.3 BesselFilterHighPassNet Class Reference	95

11.3.1 Constructor & Destructor Documentation	95
11.4 BesselFilterLowPassNet Class Reference	95
11.4.1 Constructor & Destructor Documentation	96
11.5 ButterworthFilterHighPassNet Class Reference	96
11.5.1 Constructor & Destructor Documentation	96
11.6 ButterworthFilterLowPassNet Class Reference	97
11.6.1 Constructor & Destructor Documentation	97
11.7 CChannelTestDeviceNet Class Reference	97
11.7.1 Constructor & Destructor Documentation	98
11.7.2 Member Function Documentation	98
11.8 CCMOSMea_FunctionNet Class Reference	98
11.8.1 Constructor & Destructor Documentation	100
11.8.2 Member Function Documentation	101
11.9 CCMOSMeaDeviceNet Class Reference	109
11.9.1 Constructor & Destructor Documentation	110
11.9.2 Member Function Documentation	110
11.9.3 Property Documentation	112
11.10 CCreateFilterNet Class Reference	112
11.10.1 Constructor & Destructor Documentation	113
11.10.2 Member Function Documentation	113
11.10.3 Property Documentation	114
11.11 CDacCalibrationFunctionNet Class Reference	114
11.11.1 Detailed Description	115
11.11.2 Constructor & Destructor Documentation	115
11.11.3 Member Function Documentation	115
11.12 CDacqGroupChannelGenericSelectionNet Class Reference	116
11.12.1 Constructor & Destructor Documentation	117
11.13 CDacqGroupChannelSelectionNet Class Reference	117
11.13.1 Constructor & Destructor Documentation	117
11.14 CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet > Class Template Reference	117
11.14.1 Constructor & Destructor Documentation	118
11.14.2 Member Function Documentation	118
11.15 CDeviceGroupChannelInfoGenericNet Class Reference	120
11.15.1 Constructor & Destructor Documentation	120
11.16 CDeviceGroupChannelInfoMEA2100_256Net Class Reference	121
11.16.1 Constructor & Destructor Documentation	121
11.17 CDeviceGroupChannelInfoNet Class Reference	121
11.17.1 Constructor & Destructor Documentation	122
11.18 CDeviceGroupChannelInfoSCUNet Class Reference	122
11.18.1 Constructor & Destructor Documentation	122
11.19 CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumTemplateNet > Class Template Reference	122

11.19.1 Constructor & Destructor Documentation	123
11.19.2 Member Data Documentation	123
11.20 CDeviceGroupChannelInfoW2100Net Class Reference	123
11.20.1 Constructor & Destructor Documentation	124
11.21 CDigOutStimulatorFunctionNet Class Reference	124
11.21.1 Detailed Description	125
11.21.2 Constructor & Destructor Documentation	125
11.21.3 Member Function Documentation	125
11.22 CEncapsulatorDeviceNet Class Reference	128
11.22.1 Detailed Description	128
11.22.2 Constructor & Destructor Documentation	129
11.22.3 Member Function Documentation	129
11.23 CExternDTesterDeviceNet Class Reference	129
11.23.1 Detailed Description	129
11.23.2 Constructor & Destructor Documentation	130
11.23.3 Member Function Documentation	130
11.24 CFilterCoefficientsNet Class Reference	131
11.24.1 Constructor & Destructor Documentation	131
11.24.2 Member Function Documentation	132
11.24.3 Property Documentation	133
11.25 CFilterConfigurationNet Class Reference	133
11.25.1 Constructor & Destructor Documentation	133
11.25.2 Member Function Documentation	134
11.26 CFilterConfigurationRegisterNet Class Reference	135
11.26.1 Constructor & Destructor Documentation	135
11.26.2 Member Function Documentation	135
11.27 CFilterPropertyNet Class Reference	137
11.27.1 Constructor & Destructor Documentation	137
11.27.2 Member Function Documentation	137
11.27.3 Property Documentation	137
11.28 CFluidControlDeviceNet Class Reference	138
11.28.1 Detailed Description	139
11.28.2 Constructor & Destructor Documentation	140
11.28.3 Member Function Documentation	140
11.28.4 Property Documentation	145
11.29 CFYIDeviceNet Class Reference	145
11.29.1 Detailed Description	146
11.29.2 Constructor & Destructor Documentation	146
11.29.3 Property Documentation	146
11.30 CGenericDevelopDeviceNet Class Reference	147
11.30.1 Detailed Description	154
11.30.2 Constructor & Destructor Documentation	154

11.30.3 Member Function Documentation	154
11.31 CGilsonDeviceNet Class Reference	165
11.31.1 Detailed Description	165
11.31.2 Constructor & Destructor Documentation	165
11.31.3 Member Function Documentation	166
11.31.4 Member Data Documentation	166
11.32 CGrapheneASICDeviceNet Class Reference	167
11.32.1 Constructor & Destructor Documentation	167
11.32.2 Member Function Documentation	167
11.33 CGrapheneFunctionNet Class Reference	168
11.33.1 Detailed Description	170
11.33.2 Constructor & Destructor Documentation	170
11.33.3 Member Function Documentation	170
11.34 CHiClampDeviceNet Class Reference	179
11.34.1 Detailed Description	179
11.34.2 Constructor & Destructor Documentation	179
11.34.3 Property Documentation	180
11.35 CHLADacqNet Class Reference	180
11.35.1 Constructor & Destructor Documentation	180
11.36 CHLADeviceNet Class Reference	180
11.36.1 Detailed Description	181
11.36.2 Constructor & Destructor Documentation	181
11.36.3 Property Documentation	181
11.37 CMcsUsbDacqNet::CHWInfo Class Reference	181
11.37.1 Detailed Description	182
11.37.2 Constructor & Destructor Documentation	182
11.37.3 Member Function Documentation	182
11.38 CIntanMea_FunctionNet Class Reference	184
11.38.1 Constructor & Destructor Documentation	185
11.38.2 Member Function Documentation	185
11.39 CInterfaceboard2FunctionNet Class Reference	186
11.39.1 Detailed Description	187
11.39.2 Constructor & Destructor Documentation	187
11.39.3 Member Function Documentation	187
11.40 CInterfaceboardFunctionNet Class Reference	188
11.40.1 Detailed Description	189
11.40.2 Constructor & Destructor Documentation	189
11.40.3 Member Function Documentation	189
11.41 CLIH3DeviceNet Class Reference	190
11.41.1 Detailed Description	192
11.41.2 Constructor & Destructor Documentation	192
11.41.3 Member Function Documentation	192

11.41.4 Property Documentation	199
11.42 CMcsBus_AxisParametersNet Class Reference	199
11.42.1 Constructor & Destructor Documentation	200
11.42.2 Member Function Documentation	200
11.43 CMcsBus_ExtensionNet Class Reference	201
11.43.1 Constructor & Destructor Documentation	201
11.43.2 Member Function Documentation	202
11.44 CMcsBus_FYIExtensionNet Class Reference	202
11.44.1 Constructor & Destructor Documentation	202
11.44.2 Member Function Documentation	203
11.45 CMcsBus_MotorControlNet Class Reference	204
11.45.1 Constructor & Destructor Documentation	207
11.45.2 Member Function Documentation	207
11.46 CMcsBus_SensorNet Class Reference	221
11.46.1 Constructor & Destructor Documentation	223
11.46.2 Member Function Documentation	223
11.47 CMcsBus_TempSensorNet Class Reference	231
11.47.1 Constructor & Destructor Documentation	232
11.47.2 Member Function Documentation	232
11.48 CMcsBus_VoltageModeNet Class Reference	233
11.48.1 Constructor & Destructor Documentation	234
11.48.2 Member Function Documentation	235
11.49 CMcsBusNet Class Reference	238
11.49.1 Constructor & Destructor Documentation	238
11.49.2 Member Function Documentation	239
11.50 CMcsUsbDacqNet Class Reference	242
11.50.1 Detailed Description	248
11.50.2 Constructor & Destructor Documentation	248
11.50.3 Member Function Documentation	248
11.50.4 Member Data Documentation	288
11.50.5 Property Documentation	289
11.50.6 Event Documentation	289
11.51 CMcsUsbDeviceStatePushFunctionNet Class Reference	289
11.51.1 Constructor & Destructor Documentation	290
11.51.2 Member Function Documentation	290
11.51.3 Event Documentation	290
11.52 CMcsUsbDeviceStatePushNet Class Reference	290
11.52.1 Constructor & Destructor Documentation	291
11.52.2 Member Function Documentation	291
11.52.3 Event Documentation	291
11.53 CMcsUsbFactoryNet Class Reference	291
11.53.1 Constructor & Destructor Documentation	293

11.53.2 Member Function Documentation	293
11.53.3 Member Data Documentation	299
11.54 CMcsUsbFunctionNet Class Reference	300
11.54.1 Constructor & Destructor Documentation	300
11.54.2 Member Function Documentation	301
11.54.3 Member Data Documentation	301
11.55 CMcsUsbFunctionPointerContainer Class Reference	301
11.56 CMcsUsbListEntryNet Class Reference	301
11.56.1 Detailed Description	302
11.56.2 Constructor & Destructor Documentation	302
11.56.3 Member Function Documentation	303
11.56.4 Property Documentation	306
11.57 CMcsUsbListNet Class Reference	307
11.57.1 Detailed Description	307
11.57.2 Constructor & Destructor Documentation	307
11.57.3 Member Function Documentation	308
11.57.4 Property Documentation	309
11.57.5 Event Documentation	309
11.58 CMcsUsbNet Class Reference	310
11.58.1 Detailed Description	314
11.58.2 Constructor & Destructor Documentation	314
11.58.3 Member Function Documentation	314
11.58.4 Member Data Documentation	326
11.58.5 Property Documentation	330
11.59 CMcsUsbPointerContainer Class Reference	331
11.60 CMEA2100_256DacqGroupChannelSelectionNet Class Reference	331
11.60.1 Constructor & Destructor Documentation	331
11.61 CMEA2100x256FunctionNet Class Reference	331
11.61.1 Detailed Description	332
11.61.2 Constructor & Destructor Documentation	332
11.61.3 Member Function Documentation	332
11.62 CMeaAudioFunctionNet Class Reference	333
11.62.1 Constructor & Destructor Documentation	333
11.62.2 Member Function Documentation	334
11.63 CMeaCleanDeviceNet Class Reference	336
11.63.1 Detailed Description	337
11.63.2 Constructor & Destructor Documentation	337
11.63.3 Member Function Documentation	337
11.64 CMeaCoatDeviceNet Class Reference	340
11.64.1 Detailed Description	341
11.64.2 Constructor & Destructor Documentation	341
11.64.3 Member Function Documentation	341

11.65 CMeaDeviceNet Class Reference	345
11.65.1 Detailed Description	346
11.65.2 Constructor & Destructor Documentation	347
11.65.3 Member Function Documentation	347
11.65.4 Property Documentation	352
11.66 CMeaDigitalDataFunctionNet Class Reference	353
11.66.1 Constructor & Destructor Documentation	354
11.66.2 Member Function Documentation	354
11.67 CMeaFeedbackFunctionNet Class Reference	355
11.67.1 Constructor & Destructor Documentation	356
11.67.2 Member Function Documentation	356
11.68 CMeaImpedanceDeviceNet Class Reference	359
11.68.1 Constructor & Destructor Documentation	360
11.68.2 Member Function Documentation	360
11.69 CMeasureTableDeviceNet Class Reference	361
11.69.1 Detailed Description	361
11.69.2 Constructor & Destructor Documentation	361
11.69.3 Property Documentation	361
11.70 CMeaSwitchDeviceNet Class Reference	362
11.70.1 Detailed Description	362
11.70.2 Constructor & Destructor Documentation	362
11.70.3 Member Function Documentation	363
11.71 CMeaUSBDeviceNet Class Reference	363
11.71.1 Detailed Description	364
11.71.2 Constructor & Destructor Documentation	364
11.72 CMeFunctionNet Class Reference	365
11.72.1 Detailed Description	365
11.72.2 Constructor & Destructor Documentation	365
11.72.3 Member Function Documentation	366
11.73 CMultiBatteryChargerDeviceNet Class Reference	366
11.73.1 Detailed Description	367
11.73.2 Constructor & Destructor Documentation	367
11.73.3 Member Function Documentation	367
11.74 CMultiwellCallbackFunctionNet Class Reference	374
11.74.1 Detailed Description	375
11.74.2 Constructor & Destructor Documentation	375
11.74.3 Member Function Documentation	375
11.74.4 Event Documentation	376
11.75 CMultiwellDeviceNet Class Reference	376
11.75.1 Detailed Description	377
11.75.2 Constructor & Destructor Documentation	377
11.75.3 Member Function Documentation	378

11.76 CMultiwellOptoStimFunctionNet Class Reference	384
11.76.1 Detailed Description	384
11.76.2 Constructor & Destructor Documentation	384
11.76.3 Member Function Documentation	385
11.77 CNF_GenDeviceNet Class Reference	388
11.77.1 Constructor & Destructor Documentation	389
11.77.2 Member Function Documentation	389
11.78 COctoPotDeviceNet Class Reference	389
11.78.1 Constructor & Destructor Documentation	390
11.78.2 Member Function Documentation	390
11.79 COkuvisionStimulatorDeviceNet Class Reference	393
11.79.1 Constructor & Destructor Documentation	394
11.79.2 Member Function Documentation	394
11.80 CPatchServerDeviceNet Class Reference	397
11.80.1 Detailed Description	397
11.80.2 Constructor & Destructor Documentation	397
11.80.3 Property Documentation	397
11.81 CPathIdentDeviceNet Class Reference	398
11.81.1 Constructor & Destructor Documentation	398
11.81.2 Member Function Documentation	398
11.82 CPedoterDeviceNet Class Reference	399
11.82.1 Detailed Description	399
11.82.2 Constructor & Destructor Documentation	399
11.82.3 Member Function Documentation	399
11.83 CPeristalticPumpDeviceNet Class Reference	400
11.83.1 Detailed Description	400
11.83.2 Constructor & Destructor Documentation	401
11.83.3 Property Documentation	401
11.84 CPgaDeviceNet Class Reference	401
11.84.1 Constructor & Destructor Documentation	402
11.84.2 Member Function Documentation	402
11.85 CPositionIIDeviceNet Class Reference	403
11.85.1 Detailed Description	405
11.85.2 Constructor & Destructor Documentation	405
11.85.3 Member Function Documentation	405
11.85.4 Property Documentation	412
11.86 CPositionImpDeviceNet Class Reference	412
11.86.1 Detailed Description	412
11.86.2 Constructor & Destructor Documentation	413
11.86.3 Member Function Documentation	413
11.87 CPPCDeviceNet Class Reference	415
11.87.1 Constructor & Destructor Documentation	415

11.87.2 Property Documentation	416
11.88 CPPCFunctionNet Class Reference	416
11.88.1 Detailed Description	417
11.88.2 Constructor & Destructor Documentation	417
11.88.3 Member Function Documentation	418
11.89 CPPS_DeviceNet Class Reference	425
11.89.1 Constructor & Destructor Documentation	425
11.89.2 Property Documentation	425
11.90 CPPS_FunctionNet Class Reference	426
11.90.1 Constructor & Destructor Documentation	426
11.90.2 Member Function Documentation	427
11.91 CPPSDeviceNet Class Reference	430
11.91.1 Detailed Description	430
11.91.2 Constructor & Destructor Documentation	431
11.92 CProgramPressureCurveNet Class Reference	431
11.92.1 Detailed Description	431
11.92.2 Constructor & Destructor Documentation	431
11.92.3 Member Function Documentation	432
11.93 CPulseGeneratorFunctionNet Class Reference	432
11.93.1 Detailed Description	433
11.93.2 Constructor & Destructor Documentation	433
11.93.3 Member Function Documentation	434
11.94 CRadioControlledDevicesNet Class Reference	435
11.94.1 Constructor & Destructor Documentation	436
11.94.2 Member Function Documentation	436
11.95 CCMOSMeaDeviceNet::CRegionOfInterestRect Class Reference	437
11.95.1 Constructor & Destructor Documentation	437
11.95.2 Member Function Documentation	437
11.95.3 Member Data Documentation	438
11.96 CRetinaLedDeviceNet Class Reference	438
11.96.1 Constructor & Destructor Documentation	439
11.96.2 Member Function Documentation	439
11.97 CRFFFunctionNet Class Reference	440
11.97.1 Detailed Description	441
11.97.2 Constructor & Destructor Documentation	441
11.97.3 Member Function Documentation	441
11.98 CRobo_FYIPProgram_FunctionNet Class Reference	445
11.98.1 Constructor & Destructor Documentation	445
11.98.2 Member Function Documentation	445
11.99 CRobo_FYITemp_FunctionNet Class Reference	446
11.99.1 Constructor & Destructor Documentation	447
11.99.2 Member Function Documentation	447

11.100 CRoboDacqNet Class Reference	448
11.100.1 Constructor & Destructor Documentation	451
11.100.2 Member Function Documentation	451
11.100.3 Member Data Documentation	462
11.101 CRoboDeviceNet Class Reference	463
11.101.1 Detailed Description	466
11.101.2 Constructor & Destructor Documentation	466
11.101.3 Member Function Documentation	466
11.101.4 Member Data Documentation	474
11.101.5 Property Documentation	478
11.101.6 Event Documentation	479
11.102 CRoboFluidDeviceNet Class Reference	479
11.102.1 Constructor & Destructor Documentation	480
11.102.2 Member Function Documentation	480
11.102.3 Member Data Documentation	482
11.102.4 Property Documentation	482
11.103 CRoboInjectDeviceNet Class Reference	482
11.103.1 Detailed Description	483
11.103.2 Constructor & Destructor Documentation	483
11.104 CRoboocyte2DeviceNet Class Reference	483
11.104.1 Detailed Description	483
11.104.2 Constructor & Destructor Documentation	484
11.104.3 Member Function Documentation	484
11.105 CRoboStatorDeviceNet Class Reference	484
11.105.1 Constructor & Destructor Documentation	486
11.105.2 Member Function Documentation	486
11.105.3 Property Documentation	490
11.106 CSafeISDeviceNet Class Reference	490
11.106.1 Detailed Description	491
11.106.2 Constructor & Destructor Documentation	491
11.106.3 Member Function Documentation	492
11.106.4 Property Documentation	493
11.107 CSCUDacqGroupChannelSelectionNet Class Reference	493
11.107.1 Constructor & Destructor Documentation	494
11.108 CSCUFunctionNet Class Reference	494
11.108.1 Detailed Description	496
11.108.2 Constructor & Destructor Documentation	497
11.108.3 Member Function Documentation	497
11.108.4 Event Documentation	511
11.109 CSerialPortNet Class Reference	511
11.109.1 Constructor & Destructor Documentation	511
11.109.2 Member Function Documentation	512

11.110 CStg200xBasicNet Class Reference	512
11.110.1 Detailed Description	517
11.110.2 Constructor & Destructor Documentation	517
11.110.3 Member Function Documentation	517
11.111 CStg200xDownloadBasicNet Class Reference	551
11.111.1 Detailed Description	552
11.111.2 Member Function Documentation	552
11.111.3 Property Documentation	558
11.112 CStg200xDownloadNet Class Reference	558
11.112.1 Detailed Description	559
11.112.2 Constructor & Destructor Documentation	559
11.112.3 Member Function Documentation	560
11.112.4 Event Documentation	564
11.113 CStimulusFunctionNet Class Reference	565
11.113.1 Constructor & Destructor Documentation	566
11.113.2 Member Function Documentation	567
11.113.3 Event Documentation	576
11.114 CSw2to64DeviceNet Class Reference	576
11.114.1 Detailed Description	576
11.114.2 Constructor & Destructor Documentation	577
11.114.3 Member Function Documentation	577
11.115 CTcxDeviceNet Class Reference	578
11.115.1 Detailed Description	580
11.115.2 Constructor & Destructor Documentation	580
11.115.3 Member Function Documentation	581
11.116 CTEERFunctionNet Class Reference	591
11.116.1 Detailed Description	593
11.116.2 Constructor & Destructor Documentation	593
11.116.3 Member Function Documentation	594
11.117 CTEERMachineDeviceNet Class Reference	602
11.117.1 Constructor & Destructor Documentation	602
11.117.2 Property Documentation	603
11.118 CUsbDeviceConfigurationFunctionNet Class Reference	603
11.118.1 Detailed Description	603
11.118.2 Constructor & Destructor Documentation	603
11.118.3 Member Function Documentation	604
11.119 CUsbExceptionNet Class Reference	604
11.119.1 Detailed Description	605
11.119.2 Constructor & Destructor Documentation	605
11.119.3 Property Documentation	605
11.120 CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet Class Reference	605
11.120.1 Constructor & Destructor Documentation	606

11.120.2 Member Data Documentation	606
11.121 CW2100_FunctionNet Class Reference	606
11.121.1 Constructor & Destructor Documentation	608
11.121.2 Member Function Documentation	608
11.121.3 Property Documentation	613
11.122 CW2100_StimulatorFunctionNet Class Reference	614
11.122.1 Constructor & Destructor Documentation	615
11.122.2 Member Function Documentation	615
11.122.3 Member Data Documentation	620
11.122.4 Event Documentation	620
11.123 CW2100DacqGroupChannelSelectionNet Class Reference	620
11.123.1 Constructor & Destructor Documentation	620
11.124 CWarnerUssingDeviceNet Class Reference	621
11.124.1 Detailed Description	621
11.124.2 Constructor & Destructor Documentation	621
11.124.3 Property Documentation	622
11.125 CWarnerUssingFunctionNet Class Reference	622
11.125.1 Detailed Description	624
11.125.2 Constructor & Destructor Documentation	624
11.125.3 Member Function Documentation	624
11.126 CWarnerValveControllerDeviceNet Class Reference	638
11.126.1 Detailed Description	642
11.126.2 Constructor & Destructor Documentation	643
11.126.3 Member Function Documentation	643
11.126.4 Event Documentation	660
11.127 CWarnerValveControllerDeviceTesterFunctionNet Class Reference	663
11.127.1 Detailed Description	664
11.127.2 Constructor & Destructor Documentation	664
11.127.3 Member Function Documentation	665
11.128 CWClassicFunctionNet Class Reference	666
11.128.1 Constructor & Destructor Documentation	667
11.128.2 Member Function Documentation	668
11.129 CWirelessBaseFunctionNet Class Reference	671
11.129.1 Constructor & Destructor Documentation	672
11.129.2 Member Function Documentation	672
11.130 DeviceIdNet Struct Reference	672
11.130.1 Detailed Description	672
11.130.2 Constructor & Destructor Documentation	673
11.130.3 Member Function Documentation	673
11.130.4 Member Data Documentation	673
11.131 DigitalSource< digitalsourceenum > Class Template Reference	674
11.131.1 Constructor & Destructor Documentation	674

11.131.2 Member Function Documentation	674
11.131.3 Property Documentation	675
11.132 DigitalSourceGeneral Class Reference	675
11.132.1 Constructor & Destructor Documentation	675
11.132.2 Member Function Documentation	675
11.132.3 Property Documentation	676
11.133 DriverVersionNet Class Reference	676
11.133.1 Detailed Description	677
11.133.2 Constructor & Destructor Documentation	677
11.133.3 Member Function Documentation	677
11.134 FirmwareDestinationNames Class Reference	682
11.134.1 Member Data Documentation	683
11.135 HeadStageIDType Class Reference	685
11.135.1 Member Enumeration Documentation	686
11.135.2 Constructor & Destructor Documentation	686
11.135.3 Member Function Documentation	687
11.135.4 Property Documentation	687
11.136 HeadstageIDTypeObject Class Reference	688
11.136.1 Constructor & Destructor Documentation	689
11.136.2 Member Function Documentation	689
11.136.3 Member Data Documentation	689
11.136.4 Property Documentation	690
11.137 HeadStageIDTypeState Class Reference	690
11.137.1 Property Documentation	690
11.138 mkfilterNet Class Reference	691
11.138.1 Member Function Documentation	691
11.139 CRoboDeviceNet::RoboMainLowLevelCommands Class Reference	694
11.139.1 Member Function Documentation	695
11.140 CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands Class Reference	701
11.140.1 Member Function Documentation	701
11.141 CFilterCoefficientsNet::s_FilterAttributesNet Struct Reference	701
11.141.1 Constructor & Destructor Documentation	701
11.141.2 Member Function Documentation	701
11.141.3 Member Data Documentation	702
11.142 CMeaAudioFunctionNet::s_setaudionet Struct Reference	702
11.142.1 Member Data Documentation	702
11.143 CStimulusFunctionNet::SidebandData Class Reference	703
11.143.1 Constructor & Destructor Documentation	703
11.143.2 Property Documentation	703
11.144 StgStatusNet Class Reference	704
11.144.1 Member Function Documentation	704
11.144.2 Member Data Documentation	704

11.145 CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData Class Reference	704
11.145.1 Constructor & Destructor Documentation	705
11.145.2 Property Documentation	705
11.146 usbSetupPacket_t Class Reference	706
11.146.1 Member Data Documentation	706
11.147 W2100_StimulusParametersNet Struct Reference	706
11.147.1 Member Data Documentation	707
Index	709

1 McsUsbNet.dll for MCS USB devices

1.1 Introduction

This DLL provides the .NET interface to MCS devices

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

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

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

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

1.2 System requirements

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

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

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

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

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

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

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

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

1.3 Connecting to an MCS device

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

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

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

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

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

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

2 Device Classes

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

2.1 The MCS FluidControl Device

2.1.1 Introduction

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

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

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

2.1.2 Access to the FluidControl device

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

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

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

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

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

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

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

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

2.2 MCS-USB-Sw2to64 device

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

First construct an object of the class:

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

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

To get the number of channels the device handles:

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

Set all channel switches at once:

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

Get all channel switches at once:

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

Set one channel switch:

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

Get one channel switch:

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

3 Function Classes

- [Mcs.Usb.CCMOSMea_FunctionNet](#)
- [Mcs.Usb.CDacCalibrationFunctionNet](#)
- [Mcs.Usb.CDigOutStimulatorFunctionNet](#)
- [Mcs.Usb.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.CRFFunctionNet](#)
- [Mcs.Usb.CRobo_FYITemp_FunctionNet](#)
- [Mcs.Usb.CRobo_FYIProgram_FunctionNet](#)
- [Mcs.Usb.CSCUFunctionNet](#)
- [Mcs.Usb.CStimulusFunctionNet](#)
- [Mcs.Usb.CTEERFunctionNet](#)
- [Mcs.Usb.CW2100_FunctionNet](#)
- [Mcs.Usb.CW2100_StimulatorFunctionNet](#)
- [Mcs.Usb.CWarnerUssingFunctionNet](#)
- [Mcs.Usb.CWarnerValveControllerDeviceTesterFunctionNet](#)
- [Mcs.Usb.CWClassicFunctionNet](#)
- [Mcs.Usb.CWirelessBaseFunctionNet](#)

4 Data ACQuisition (DACQ) Devices

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

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

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

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

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

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

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

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

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

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

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

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

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

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

see MEA_Recording in the Examples directory.

5 The MCS Robo Device

5.1 Introduction

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

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

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

6 STG200x & STG400x STimulus Generator

6.1 Introduction

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

6.2 Download mode

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

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

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

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

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

6.2.1 Memory Layout and Trigger Setup

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

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

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

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

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

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

```
}
```

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

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

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

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

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

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

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

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

```

        pData[i] = sin >= 0 ? (ushort)sin : (ushort)((int)Math.Abs(sin) +
            (int)Math.Pow(2, DACResolution - 1));
        tData[i] = (UInt64_t)20; // duration in µs
    }
    device.SendChannelData(2, pData, tData);
}
// Data for Sync 0
{
    device.ClearSyncData(0);
    ushort[] pData = new ushort[1000];
    UInt64_t[] tData = new UInt64_t[1000];
    for (int i = 0; i < 1000; i++)
    {
        pData[i] = (ushort)(i&1);
        tData[i] = 20;
    }
    device.SendSyncData(0, pData, tData);
}

```

Start the trigger by pushing the front button or by software

```

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

```

see the StgDownloadExampleNet in the example directory.

6.3 Streaming mode

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

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

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

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

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

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

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

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

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

6.3.1 Memory Layout and Trigger Setup

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

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

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

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

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

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

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

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

```
void dataHandler(uint32_t trigger)
{
    double factor = 1;
    if (trigger == 0) // Callback for Trigger 1
    {
        // Handle Channel 1
        uint32_t channel = 0;
        for ( ; ; )
        {
            uint32_t space = device.GetDataQueueSpace(channel);
            if (space < 1000)
                break;
            short[] data = new short[1000];
            for (int i = 0; i < 1000; i++)
            {
                // Calc Sin-Wave (16 bits) lower bits will be removed according resolution
                double sin = factor * (Math.Pow(2, 16 - 1) - 1.0) *
                    Math.Sin(2.0 * (double)i * Math.PI / (double)1000);
                data[i] = (short)sin;
            }
            uint32_t enqueued = device.EnqueueData(channel, data);
        }
    }
    // Handle Channel 3
    uint32_t channel = 2;
    for ( ; ; )
    {
        uint32_t space = device.GetDataQueueSpace(channel);
        if (space < 700)
            break;
        short[] data = new short[700];
        for (int i = 0; i < 700; i++)
        {
            // Calc Sin-Wave (16 bits) lower bits will be removed according resolution
            double sin = factor * (Math.Pow(2, 16 - 1) - 1.0) *
                Math.Sin(2.0 * (double)i * Math.PI / (double)700);
            data[i] = (short)sin;
        }
        uint32_t enqueued = device.EnqueueData(channel, data);
    }
}
// Handle Syncout 1
uint32_t channel = 0;
for ( ; ; )
{
    uint32_t space = device.GetSyncoutQueueSpace(channel);
    if (space < 1000)
        break;
    ushort[] data = new ushort[1000];
    for (int i = 0; i < 1000; i++)
```

```

        data[i] = (ushort)(i & 1);
        uint32_t enqueued = device.EnqueueSyncout(channel, data);
    }
}
}
}
void errorHandler()
{
}

```

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

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

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

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

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

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

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

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

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

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

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

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

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

```

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

```

or fixed memory sizes:

```

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

```

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

```

uint32_t ntrigger = device.GetNumberOfTriggerInputs(); // obtain number of triggers in this STG
uint[] channelmap = new uint[ntrigger];
uint[] syncoutmap = new uint[ntrigger];
uint[] digoutmap = new uint[ntrigger];
uint[] autostart = new uint[ntrigger];
uint[] callback_threshold = new uint[ntrigger];
for (int i = 0; i < ntrigger; i++)
{
    channelmap[i] = 0;

```



```

    syncoutmap[i] = 0;
    digoutmap[i] = 0;
    autostart[i] = 0;
    callback_threshold[i] = 0;
}
channelmap[0] = 0x1 + 0x4; // Channel 1 und Channel 3 to Trigger 1
syncoutmap[0] = 0x1; // Syncout 1 to Trigger 1
autostart[0] = 1;
callback_threshold[0] = 50; // 50% of buffer size
device.SetupTrigger(channelmap, syncoutmap, digoutmap, autostart, callback_threshold);
device.StartLoop();
System.Threading.Thread.Sleep(1000); // Give StartLoop some time

```

Start Trigger by pushing the front button or by Software

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

see the StgStreamingExampleNet in the example directory.

7 Namespace Index

7.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Mcs	22
Mcs::Usb	22

8 Hierarchical Index

8.1 Class Hierarchy

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

CW2100_FunctionNet::AudioChannelsNet	94
BatteryState	94
CCreateFilterNet	112
BesselFilterHighPassNet	95
BesselFilterLowPassNet	95
ButterworthFilterHighPassNet	96
ButterworthFilterLowPassNet	97
CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumTemplateNet >	122
CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumNet >	122
CDeviceGroupChannelInfoNet	121
CDeviceGroupChannelInfoTemplateNet< int >	122
CDeviceGroupChannelInfoGenericNet	120

CDeviceGroupChannelInfoTemplateNet< MEA2100_256DacqGroupChannelEnumNet >	122
CDeviceGroupChannelInfoMEA2100_256Net	121
CDeviceGroupChannelInfoTemplateNet< SCUDacqGroupChannelEnumNet >	122
CDeviceGroupChannelInfoSCUNet	122
CDeviceGroupChannelInfoTemplateNet< W2100DacqGroupChannelEnumNet >	122
CDeviceGroupChannelInfoW2100Net	123
CFilterCoefficientsNet	131
CFilterPropertyNet	137
CMcsUsbDacqNet::CHWInfo	181
CMcsUsbFunctionNet	300
CDacqGroupChannelSelectionTemplateNet< W2100DacqGroupChannelEnumNet, W2100DacqGroupChannelEnum, CDeviceGroupChannelInfoW2100Net >	117
CW2100DacqGroupChannelSelectionNet	620
CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumNet, DacqGroupChannelEnum, CDeviceGroupChannelInfoNet >	117
CDacqGroupChannelSelectionNet	117
CDacqGroupChannelSelectionTemplateNet< int, int, CDeviceGroupChannelInfoGenericNet >	117
CDacqGroupChannelGenericSelectionNet	116
CDacqGroupChannelSelectionTemplateNet< SCUDacqGroupChannelEnumNet, SCUDacqGroupChannelEnum, CDeviceGroupChannelInfoSCUNet >	117
CSCUDacqGroupChannelSelectionNet	493
CDacqGroupChannelSelectionTemplateNet< MEA2100_256DacqGroupChannelEnumNet, MEA2100_256DacqGroupChannelEnum, CDeviceGroupChannelInfoMEA2100_256Net >	117
CMEA2100_256DacqGroupChannelSelectionNet	331
CCMOSMea_FunctionNet	98
CDacCalibrationFunctionNet	114
CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet >	117
CDigOutStimulatorFunctionNet	124
CFilterConfigurationNet	133
CFilterConfigurationRegisterNet	135
CGrapheneFunctionNet	168
CIntanMea_FunctionNet	184
CInterfaceboardFunctionNet	188

CInterfaceboard2FunctionNet	186
CMEA2100x256FunctionNet	331
CMcsBusNet	238
CMcsBus_AxisParametersNet	199
CMcsBus_ExtensionNet	201
CMcsBus_FYIExtensionNet	202
CMcsBus_MotorControlNet	204
CMcsBus_SensorNet	221
CMcsBus_TempSensorNet	231
CMcsBus_VoltageModeNet	233
CMcsUsbDeviceStatePushFunctionNet	289
CMultiwellCallbackFunctionNet	374
CSCUFunctionNet	494
CMeFunctionNet	365
CMeaAudioFunctionNet	333
CMeaDigitalDataFunctionNet	353
CMeaFeedbackFunctionNet	355
CMultiwellOptoStimFunctionNet	384
CPPCFunctionNet	416
CPPS_FunctionNet	426
CProgramPressureCurveNet	431
CPulseGeneratorFunctionNet	432
CRFFunctionNet	440
CRobo_FYIProgram_FunctionNet	445
CRobo_FYITemp_FunctionNet	446
CStimulusFunctionNet	565
CTEERFunctionNet	591
CUsbDeviceConfigurationFunctionNet	603
CW2100_StimulatorFunctionNet	614
CWarnerUssingFunctionNet	622
CWarnerValveControllerDeviceTesterFunctionNet	663
CWirelessBaseFunctionNet	671

CW2100_FunctionNet	606
CWClassicFunctionNet	666
CMcsUsbFunctionPointerContainer	301
CMcsUsbListEntryNet	301
CMcsUsbListNet	307
CMcsUsbNet	310
CExternDTesterDeviceNet	129
CFluidControlDeviceNet	138
CGenericDevelopDeviceNet	147
CGilsonDeviceNet	165
CMcsUsbDacqNet	242
CMeaDeviceNet	345
CMeaUSBDeviceNet	363
CCMOSMeaDeviceNet	109
CGrapheneASICDeviceNet	167
CHLADacqNet	180
CLIH3DeviceNet	190
CMultiwellDeviceNet	376
CWarnerUssingDeviceNet	621
COctoPotDeviceNet	389
CRoboDacqNet	448
CMcsUsbDeviceStatePushNet	290
CWarnerValveControllerDeviceNet	638
CMcsUsbFactoryNet	291
CMeaCleanDeviceNet	336
CMeaCoatDeviceNet	340
CMeaImpedanceDeviceNet	359
CMeaSwitchDeviceNet	362
CChannelTestDeviceNet	97
CMultiBatteryChargerDeviceNet	366
CNF_GenDeviceNet	388
COkuvisionStimulatorDeviceNet	393

CPPCDeviceNet	415
CPPS_DeviceNet	425
CPathIdentDeviceNet	398
CPedoterDeviceNet	399
CPeristalticPumpDeviceNet	400
CPgaDeviceNet	401
CPositionIIDeviceNet	403
CPositionImpDeviceNet	412
CRadioControlledDevicesNet	435
CRetinaLedDeviceNet	438
CRoboDeviceNet	463
CEncapsulatorDeviceNet	128
CFYIDeviceNet	145
CHLADeviceNet	180
CHiClampDeviceNet	179
CMeasureTableDeviceNet	361
CPPSDeviceNet	430
CPatchServerDeviceNet	397
CRoboStatorDeviceNet	484
CRoboInjectDeviceNet	482
CRoboocyte2DeviceNet	483
CTEERMachineDeviceNet	602
CRoboFluidDeviceNet	479
CSafeISDeviceNet	490
CSerialPortNet	511
CStg200xBasicNet	512
CStg200xDownloadBasicNet	551
CStg200xDownloadNet	558
CSw2to64DeviceNet	576
CTcxDeviceNet	578
CMcsUsbPointerContainer	331
CCMOSMeaDeviceNet::CRegionOfInterestRect	437

CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet	605
DeviceIdNet	672
DigitalSource< digitalsourceenum >	674
DigitalSourceGeneral	675
DriverVersionNet	676
Exception	
CUsbExceptionNet	604
FirmwareDestinationNames	682
HeadstageIDTypeObject	688
HeadStageIDTypeState	690
IComparable	
HeadStageIDType	685
mkfilterNet	691
CRoboDeviceNet::RoboMainLowLevelCommands	694
CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands	701
CFilterCoefficientsNet::s_FilterAttributesNet	701
CMeaAudioFunctionNet::s_setaudionet	702
CStimulusFunctionNet::SidebandData	703
StgStatusNet	704
stgstreaming	
CStg200xBasicNet	512
CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData	704
usbSetupPacket_t	706
W2100_StimulusParametersNet	706

9 Class Index

9.1 Class List

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

CW2100_FunctionNet::AudioChannelsNet	94
BatteryState	94
BesselFilterHighPassNet	95
BesselFilterLowPassNet	95

ButterworthFilterHighPassNet	96
ButterworthFilterLowPassNet	97
CChannelTestDeviceNet	97
CCMOSMea_FunctionNet	98
CCMOSMeaDeviceNet	109
CCreateFilterNet	112
CDacCalibrationFunctionNet	114
CDacqGroupChannelGenericSelectionNet	116
CDacqGroupChannelSelectionNet	117
CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplateNet >	117
CDeviceGroupChannelInfoGenericNet	120
CDeviceGroupChannelInfoMEA2100_256Net	121
CDeviceGroupChannelInfoNet	121
CDeviceGroupChannelInfoSCUNet	122
CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumTemplateNet >	122
CDeviceGroupChannelInfoW2100Net	123
CDigOutStimulatorFunctionNet	
CDigOutStimulatorFunctionNet is the class of the DigOut stimulator function class	124
CEncapsulatorDeviceNet	
CEncapsulatorDeviceNet is the to control the MCS HiClamp device	128
CExternDTesterDeviceNet	
CExternDTesterDeviceNet is the class to access the ExternD Tester (Handheld Device Tester D)	129
CFilterCoefficientsNet	131
CFilterConfigurationNet	133
CFilterConfigurationRegisterNet	135
CFilterPropertyNet	137
CFluidControlDeviceNet	
CFluidControlDeviceNet is the class to control MCS FluidControl (FCB and FCX) device	138
CFYIDeviceNet	
CFYIDeviceNet is the class to control the MCS FYI device	145
CGenericDevelopDeviceNet	
CGenericDevelopDeviceNet is the class to use during development of a new device	147
CGilsonDeviceNet	
CGilsonDeviceNet is the class to control a Gilson device	165

CGrapheneASICDeviceNet	167
CGrapheneFunctionNet	
CGrapheneFunctionNet is the class to control Graphene device functions	168
CHiClampDeviceNet	
CHiClampDeviceNet is the to control the MCS HiClamp device	179
CHLADacqNet	180
CHLADeviceNet	
CHLADeviceNet is the to control the MCS HLA device	180
CMcsUsbDacqNet::CHWInfo	
Class to provide hardware information about the device	181
CIntanMea_FunctionNet	184
CInterfaceboard2FunctionNet	
CInterfaceboard2FunctionNet is the class to control the Interfaceboard	186
CInterfaceboardFunctionNet	
CInterfaceboardFunctionNet is the class to control the Interfaceboard	188
CLIH3DeviceNet	
CLIH3DeviceNet is the class to access the HEKA LIH3 device	190
CMcsBus_AxisParametersNet	199
CMcsBus_ExtensionNet	201
CMcsBus_FYIExtensionNet	202
CMcsBus_MotorControlNet	204
CMcsBus_SensorNet	221
CMcsBus_TempSensorNet	231
CMcsBus_VoltageModeNet	233
CMcsBusNet	238
CMcsUsbDacqNet	
Base class for data acquisition devices	242
CMcsUsbDeviceStatePushFunctionNet	289
CMcsUsbDeviceStatePushNet	290
CMcsUsbFactoryNet	291
CMcsUsbFunctionNet	300
CMcsUsbFunctionPointerContainer	301
CMcsUsbListEntryNet	
McsUsbListEntryNet identifies a connected device	301
CMcsUsbListNet	
Class to handle a list of connected MCS USB devices	307

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	310
CMcsUsbPointerContainer	331
CMEA2100_256DacqGroupChannelSelectionNet	331
CMEA2100x256FunctionNet	
CMEA2100x256FunctionNet is the class to control the MEA2100-256 device needs #include "↔ Stg200xNet.h" to resolve documentation reference	331
CMeaAudioFunctionNet	333
CMeaCleanDeviceNet	
CMeaCleanDeviceNet is the class to access the MEA Clean device	336
CMeaCoatDeviceNet	
CMeaCoatDeviceNet is the class to access the MEA Coat device	340
CMeaDeviceNet	
Base class for MEA data acquisition devices	345
CMeaDigitalDataFunctionNet	353
CMeaFeedbackFunctionNet	355
CMeaImpedanceDeviceNet	359
CMeasureTableDeviceNet	
CMeasureTableDeviceNet is the to control the MCS HLA device	361
CMeaSwitchDeviceNet	
The class to control the USB-MEA-Switch	362
CMeaUSBDeviceNet	
Class for data acquisition via ME and MEA USB amplifiers	363
CMeFunctionNet	365
CMultiBatteryChargerDeviceNet	
CMultiBatteryChargerDeviceNet is the class to access the MBC-08 device	366
CMultiwellCallbackFunctionNet	
CMultiwellCallbackFunctionNet is the class to access the Multiwell-Mini-Stimulator	374
CMultiwellDeviceNet	
CMultiwellDeviceNet is the class to access the Multiwell device	376
CMultiwellOptoStimFunctionNet	
CMultiwellOptoStimFunctionNet is the class to access the optical properties of the Multiwell Optostim device	384
CNF_GenDeviceNet	388
COctoPotDeviceNet	389
COKuvisionStimulatorDeviceNet	393
CPatchServerDeviceNet	
CPatchServerDeviceNet is the class to control the MCS PatchServer device	397

CPathIdentDeviceNet	398
CPedoterDeviceNet	399
CPeristalticPumpDeviceNet	
CPeristalticPumpDeviceNet is the class to control a Persistaltic Pump	400
CPgaDeviceNet	401
CPositionIIDeviceNet	
CPositionIIDeviceNet is the class to control PositionII devices	403
CPositionImpDeviceNet	
CPositionImpDeviceNet is the class to access the Position/Imp devices	412
CPPCDeviceNet	415
CPPCFunctionNet	
CPPCFunctionNet is the class to access the PPC (high precision Patch Peristaltic patch Pump	416
CPPS_DeviceNet	425
CPPS_FunctionNet	426
CPPSDeviceNet	
CPPS4plus1DeviceNet is the to control the MCS HLA device	430
CProgramPressureCurveNet	
CProgramPressureCurveNet is the class to program pressure curves	431
CPulseGeneratorFunctionNet	
CPulseGeneratorFunctionNet is the class to control the pulse generator for video tracking	432
CRadioControlledDevicesNet	435
CCMOSMeaDeviceNet::CRegionOfInterestRect	437
CRetinaLedDeviceNet	438
CRFFFunctionNet	
CRFFFunctionNet is the class to control RF devices	440
CRobo_FYIProgram_FunctionNet	445
CRobo_FYITemp_FunctionNet	446
CRoboDacqNet	448
CRoboDeviceNet	
CRoboDeviceNet is the base class for all Robo platform based devices	463
CRoboFluidDeviceNet	479
CRoboInjectDeviceNet	
CRoboInjectDeviceNet is the to control the MCS RoboInject device	482
CRoboocyte2DeviceNet	
CRoboocyte2DeviceNet is the class to control the MCS Roboocyte2 device	483
CRoboStatorDeviceNet	484

CSafeISDeviceNet 490	
CSCUDacqGroupChannelSelectionNet	493
CSCUFunctionNet CSCUFunctionNet is the class to control the SCU device	494
CSerialPortNet	511
CStg200xBasicNet Base class for the Stg200x	512
CStg200xDownloadBasicNet CStg200xDownloadBasicNet is the base class to control the download mode of the MCS STG device	551
CStg200xDownloadNet Main class for the STG download mode This class implements the STG download mode interface.	558
CStimulusFunctionNet	565
CSw2to64DeviceNet The class to control the MCS-USB-Sw2to64 device	576
CTcxDeviceNet Class to control a Temperature Controller (TCX)	578
CTEERFunctionNet CTEERFunctionNet is the class to control the TEER device	591
CTEERMachineDeviceNet	602
CUsbDeviceConfigurationFunctionNet CUsbDeviceConfigurationFunctionNet is the class to configure the USB firmware	603
CUsbExceptionNet Exception class that is thrown in case of an USB error	604
CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet	605
CW2100_FunctionNet	606
CW2100_StimulatorFunctionNet	614
CW2100DacqGroupChannelSelectionNet	620
CWarnerUssingDeviceNet CWarnerUssingDeviceNet is the class to control the Ussing device	621
CWarnerUssingFunctionNet CWarnerUssingFunctionNet is the class to control the Ussing device	622
CWarnerValveControllerDeviceNet CWarnerValveControllerDeviceNet is the class to access the Warner Valve Controller	638
CWarnerValveControllerDeviceTesterFunctionNet CWarnerValveControllerDeviceTesterFunctionNet is the class to access the functions for the Warner Valve Controller Device Tester	663
CWClassicFunctionNet	666

CWirelessBaseFunctionNet	671
DeviceIdNet	
Device Id	672
DigitalSource< digitalsourceenum >	674
DigitalSourceGeneral	675
DriverVersionNet	
Class gives firmware versions of the device's firmware destinations	676
FirmwareDestinationNames	682
HeadStageIDType	685
HeadstageIDTypeObject	688
HeadStageIDTypeState	690
mkfilterNet	691
CRoboDeviceNet::RoboMainLowLevelCommands	694
CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands	701
CFilterCoefficientsNet::s_FilterAttributesNet	701
CMeaAudioFunctionNet::s_setaudionet	702
CStimulusFunctionNet::SidebandData	703
StgStatusNet	704
CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData	704
usbSetupPacket_t	706
W2100_StimulusParametersNet	706

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 [BesselFilterLowPassNet](#)
- class [BesselFilterHighPassNet](#)
- 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 [DeviceIdNet](#)
 - 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** ,
 - MCS_GENERIC_DEVELOPMENT_DEVICE** ,
 - MCS_DEVICE_USB** ,
 - MCS_MCCARD_DEVICE** ,
 - MCS_STG_DEVICE** ,
 - MCS_MC_STIMULUS_DEVICE** ,
 - MCS_MEASUSB_DEVICE** ,
 - MCS_MEA_DEVICE** ,
 - MCS_OCTOPOT_DEVICE** ,


```

MCS_TERSENS_DEVICE ,
MCS_PGA_DEVICE ,
MCS_PCX_DEVICE ,
MCS_TCX_DEVICE ,
MCS_FCX_DEVICE ,
MCS_RETINA_LED_DEVICE ,
MCS_MEA_SWITCH_DEVICE ,
MCS_MEA_IMPEDANCE_DEVICE ,
MCS_CHANNELTEST_DEVICE ,
MCS_SW2TO64_DEVICE ,
MCS_RETINA_AMS_DONGLE ,
MCS_PATHIDENT_DEVICE ,
MCS_ROBO_DEVICE ,
MCS_ROBOOCYTE2_DEVICE ,
MCS_ROBOINJECT_DEVICE ,
MCS_HICLAMP_DEVICE ,
MCS_PATCHSERVER_DEVICE ,
MCS_ENCAPSULATOR_DEVICE ,
MCS_MEASURETABLE_DEVICE ,
MCS_FYI_DEVICE ,
MCS_HLA_DEVICE ,
MCS_PPS_DEVICE ,
MCS_PPS5_DEVICE ,
MCS_OKUVISION_STIMULATOR_DEVICE ,
MCS_NF_GEN_DEVICE ,
MCS_SAFEIS_DEVICE ,
MCS_PERISTALTIC_PUMP_DEVICE ,
MCS_EXTERN_BC_TESTER_DEVICE ,
MCS_EXTERN_D_TESTER_DEVICE ,
MCS_SOFTWARE_DONGLE_DEVICE ,
MCS_MEA_CLEAN_DEVICE ,
MCS_MEA_COAT_DEVICE ,
MCS_SMARTIMPLANT_DEVICE ,
MCS_MBC08_DEVICE ,
MCS_PEDOTER_DEVICE ,
MCS_PPC_DEVICE ,
WARNER_VALVE_CONTROL_DEVICE = 7000 ,
WARNER_USSING_DEVICE ,
HEKA_LIH3_DEVICE = 8000 ,
ALA_VC3_DEVICE = 9990 ,
MCS_DEVICE_USB_CYPRESS = 9991 }

```

Enumerates the group of MCS devices to connect to.

- enum class [VendorIdEnumNet](#) {
[Any](#) = -1 ,
[None](#) = 0 ,
[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_MEAUSB ,
[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 ,
HekaLIH406 = MCS_PRODUCT_ID_HEKA_LIH_406 ,
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_EXTERNDTESTER ,
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 ,
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 [DigitalSourceEnumNet](#) {


```

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 `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 ,
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 ,
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) ,
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 ,
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) ,
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 [RetriggerActionEnumNet](#) {
[raStop](#) = STG200x_RETRIGGER_STOP ,
[raRestart](#) = STG200x_RETRIGGER_RESTART ,
[raIgnore](#) = STG200x_RETRIGGER_IGNORE ,
[raGate](#) = STG200x_RETRIGGER_GATEMODE ,
[raSingle](#) = 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 [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 `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 ,
[LIH30UserADCGroup](#) = 0x50 ,
[LIH30TestADCGroup](#) = 0x51 ,
[LIH30ADCModulesGroup](#) = 0x52 ,
[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 ,

Generated by Doxygen


```

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 wether 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](#) {
[SOFAndCTRLword](#) = 0x01 ,
[ChecksumAndPacketCounter](#) = 0x02 ,
[Timestamp](#) = 0x04 ,
[EOFAndCRC](#) = 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_100PBA = HS_PLATETYPE_60 ,
    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 ,
    CmosMeaInterfaceboard = 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 ,
CmosmeaIFB2 = FPGA_ID_CMOS_MEA_IFB2 ,
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_TBISI_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_ADC_Channel_EnumNet {
    User_ADC_0 = 0 ,
    User_ADC_1 = 1 ,
    User_ADC_2 = 2 ,
    User_ADC_3 = 3 ,
    User_ADC_4 = 4 ,

```

- ```

Test_ADC_EPC10 = 5 ,
ModulA_ADC0 = 6 ,
ModulA_ADC1 = 7 ,
ModulA_ADC2 = 8 ,
ModulA_ADC3 = 9 ,
ModulB_ADC0 = 10 ,
ModulB_ADC1 = 11 ,
ModulB_ADC2 = 12 ,
ModulB_ADC3 = 13 ,
ModulC_ADC0 = 14 ,
ModulC_ADC1 = 15 ,
ModulC_ADC2 = 16 ,
ModulC_ADC3 = 17 ,
ModulD_ADC0 = 18 ,
ModulD_ADC1 = 19 ,
ModulD_ADC2 = 20 ,
ModulD_ADC3 = 21 }

```
- enum class [LIH30\\_DAC\\_Channel\\_EnumNet](#) {
 

```

User_DAC_0 = 0 ,
User_DAC_1 = 1 ,
User_DAC_2 = 2 ,
Test_DAC_EPC10 = 3 ,
ModulA_DAC0 = 4 ,
ModulA_DAC1 = 5 ,
ModulB_DAC0 = 6 ,
ModulB_DAC1 = 7 ,
ModulC_DAC0 = 8 ,
ModulC_DAC1 = 9 ,
ModulD_DAC0 = 10 ,
ModulD_DAC1 = 11 }

```
  - 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](#)<sup>^</sup> request)
- private delegate void [OnMcsUsbDeviceStateCallback](#) (IntPtr pThis, uint32\_t size, IntPtr buffer)
- public delegate void [OnUpdateFirmwareStatusChange](#) (String<sup>^</sup>)
- public delegate void [OnUpdateFirmwareProgress](#) (int)



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

### 10.2.1 Enumeration Type Documentation

#### 10.2.1.1 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     |                                |
| MCSBUS3     |                                |
| MCSBUS4     |                                |
| MCSBUS5     |                                |
| MCSBUS6     |                                |
| MCSBUS7     |                                |

## Enumerator

|              |  |
|--------------|--|
| 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  |  |
| BUS2MCSBUS2  |  |
| BUS2MCSBUS3  |  |
| BUS2MCSBUS4  |  |
| BUS2MCSBUS5  |  |
| BUS2MCSBUS6  |  |

## Enumerator

|                       |  |
|-----------------------|--|
| 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            |  |
| FPGA4_GOLD            |  |
| FPGA5_GOLD            |  |
| FPGA6_GOLD            |  |
| FPGA7_GOLD            |  |
| FPGA8_GOLD            |  |

## Enumerator

|                      |  |
|----------------------|--|
| 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 |  |
| ALTERA_TARGET1       |  |
| ALTERA_TARGET2       |  |
| ALTERA_TARGET3       |  |
| USB_TARGET1          |  |
| USB_TARGET2          |  |

**Enumerator**

|             |  |
|-------------|--|
| USB_TARGET3 |  |
| UnknownDest |  |

**10.2.1.6 CMOSMeaBathModeEnumNet** enum [CMOSMeaBathModeEnumNet](#) [strong]

Enumerations of CMOS MEA Bath Mode

**Enumerator**

|                |  |
|----------------|--|
| Ground         |  |
| Stimulation    |  |
| CurrentMeasure |  |

**10.2.1.7 CMOSMeaHeadstage1NCBathCurrentEnumNet** enum [CMOSMeaHeadstage1NCBathCurrentEnumNet](#) [strong]

Enumerations of CMOS MEA HS Current Monitoring Channels Group Bitmask

**Enumerator**

|               |  |
|---------------|--|
| NCBathCurrent |  |
|---------------|--|

**10.2.1.8 CMOSMeaHeadstage1NCCol2CurrentEnumNet** enum [CMOSMeaHeadstage1NCCol2CurrentEnumNet](#) [strong]

Enumerations of CMOS MEA HS Current Monitoring Channels Group Bitmask

**Enumerator**

|               |  |
|---------------|--|
| NCCol2Current |  |
|---------------|--|

**10.2.1.9 CMOSMeaHeadstage1NChipTempEnumNet** enum [CMOSMeaHeadstage1NChipTempEnumNet](#) [strong]

Enumerations of CMOS MEA HS Temperature Monitoring Channels Group Bitmask

**Enumerator**

|                  |  |
|------------------|--|
| NChipTemperature |  |
|------------------|--|

**10.2.1.10 CMOSMeaHS1SidebandEnumNet** enum [CMOSMeaHS1SidebandEnumNet](#) [strong]

Enumerations of CMOS MEA HS STG Sideband Channels Group Bitmask

Enumerator

|            |  |
|------------|--|
| SBSVector1 |  |
| SBSVector2 |  |
| SBSVector3 |  |
| SBSVector4 |  |

**10.2.1.11 CMOSMeaHS1TriggerStatusEnumNet** enum [CMOSMeaHS1TriggerStatusEnumNet](#) [strong]

Enumerations of CMOS MEA HS STG Trigger Status Channels Group Bitmask

Enumerator

|                |  |
|----------------|--|
| TriggerStatus1 |  |
| TriggerStatus2 |  |
| TriggerStatus3 |  |
| TriggerStatus4 |  |

**10.2.1.12 CMOSMeaIFDigChannelEnumNet** enum [CMOSMeaIFDigChannelEnumNet](#) [strong]

Enumerations of CMOS MEA IF Digital Channels Group Bitmask

Enumerator

|               |  |
|---------------|--|
| DigitalMux    |  |
| DigitalInPort |  |
| DigitalOutReg |  |
| FeedbackReg   |  |
| DigitalReg    |  |
| AuxPort       |  |

**10.2.1.13 CMOSMeaInterfaceADCEnumNet** enum [CMOSMeaInterfaceADCEnumNet](#) [strong]

Enumerations of CMOS MEA IF Analog Channels Group Bitmask

**Enumerator**

|            |  |
|------------|--|
| IFChannel1 |  |
| IFChannel2 |  |
| IFChannel3 |  |
| IFChannel4 |  |
| IFChannel5 |  |
| IFChannel6 |  |
| IFChannel7 |  |
| IFChannel8 |  |

**10.2.1.14 CMOSMeaPacketFrameContextGroupEnumNet** enum [CMOSMeaPacketFrameContextGroupEnumNet](#) [strong]

Enumerations of CMOS MEA HS STG Trigger Status Channels Group Bitmask

**Enumerator**

|                          |  |
|--------------------------|--|
| SOFAndCTRLword           |  |
| ChecksumAndPacketCounter |  |
| Timestamp                |  |
| EOAndCRC                 |  |

**10.2.1.15 CMOSMeaSTG1DACSignalEnumNet** enum [CMOSMeaSTG1DACSignalEnumNet](#) [strong]

Enumerations of CMOS MEA DAC Stimulation Channels Group Bitmask

**Enumerator**

|             |  |
|-------------|--|
| DAC1Channel |  |
| DAC2Channel |  |
| DAC3Channel |  |
| DAC4Channel |  |

**10.2.1.16 CMOSMeaValueUnitEnumNet** enum [CMOSMeaValueUnitEnumNet](#) [strong]

Enumerations of CMOS MEA Units of Values in Data stream

**Enumerator**

|            |  |
|------------|--|
| NoUnit     |  |
| NanoVolt   |  |
| PicoAmpere |  |



## Enumerator

|                    |  |
|--------------------|--|
| NanoAmpere         |  |
| MicroAmpere        |  |
| MilliDegreeCelsius |  |

**10.2.1.17 DacqGroupChannelEnumNet** enum [DacqGroupChannelEnumNet](#) [strong]

Enumerates the Channel Groups of Datastream

## Enumerator

|                              |  |
|------------------------------|--|
| HeadstageElectrodeGroup      |  |
| InterfaceADCGroup            |  |
| DSPDataGroup                 |  |
| Headstage1NCBathCurrentGroup |  |
| Headstage1NCCol2CurrentGroup |  |
| Headstage1NChipTempGroup     |  |
| STG1DACSignalGroup           |  |
| LIH30UserADCGroup            |  |
| LIH30TestADCGroup            |  |
| LIH30ADCModulesGroup         |  |
| IFDigChannelsGroup           |  |
| STG1SidebandsGroup           |  |
| STG1TriggerStatusGroup       |  |
| DACQ1DigitalGroup            |  |
| AudioTestChannelGroup        |  |
| PacketFrameContextGroup      |  |

**10.2.1.18 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 |  |

**10.2.1.19 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   |  |

**10.2.1.20 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.                       |

## Enumerator

|                                 |                                            |
|---------------------------------|--------------------------------------------|
| 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 ExternBCTest 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.          |
| 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.           |

### 10.2.1.21 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.                                                                                                |

## Enumerator

|                 |                                             |
|-----------------|---------------------------------------------|
| 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. |

### 10.2.1.22 DigitalSourceEnumNet enum `DigitalSourceEnumNet` [strong]

Enumerates the digital source of the MEA2100 device.

## Enumerator

|                      |  |
|----------------------|--|
| DigitalInOfOutPort   |  |
| DigitalIn            |  |
| DigitalPulse         |  |
| Feedback             |  |
| AuxIn                |  |
| Zero                 |  |
| One                  |  |
| HS1Trigger1Status    |  |
| HS1Trigger2Status    |  |
| HS1Trigger3Status    |  |
| HS1Trigger4Status    |  |
| HS1Trigger5Status    |  |
| HS1Trigger6Status    |  |
| HS1Sideband1         |  |
| HS1Sideband2         |  |
| HS1Sideband3         |  |
| HS1Sideband4         |  |
| HS1Sideband5         |  |
| HS1Sideband6         |  |
| HS2Trigger1Status    |  |
| HS2Trigger2Status    |  |
| HS2Trigger3Status    |  |
| HS2Trigger4Status    |  |
| HS2Trigger5Status    |  |
| HS2Trigger6Status    |  |
| HS2Sideband1         |  |
| HS2Sideband2         |  |
| HS2Sideband3         |  |
| HS2Sideband4         |  |
| HS2Sideband5         |  |
| HS2Sideband6         |  |
| PulseGenerator       |  |
| DigitalOutStimulator |  |
| DigitalData          |  |
| DeviceRunStatus      |  |
| LastPosition         |  |

**10.2.1.23 DigitalStimulatorTriggerEventEnumNet** enum [DigitalStimulatorTriggerEventEnumNet](#) [strong]

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

Enumerator

|       |  |
|-------|--|
| Start |  |
| Stop  |  |

**10.2.1.24 DigitalStimulatorTriggerSlopeEnumNet** enum [DigitalStimulatorTriggerSlopeEnumNet](#) [strong]

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

Enumerator

|         |  |
|---------|--|
| Falling |  |
| Rising  |  |

**10.2.1.25 DigitalTargetEnumNet** enum [DigitalTargetEnumNet](#) [strong]

Enumerates the Digital Targets for Digital Sources

Enumerator

|                              |  |
|------------------------------|--|
| Digout                       |  |
| Digstream                    |  |
| DacqTrigger                  |  |
| StgTrigger                   |  |
| StgListModeTrigger           |  |
| DigOutStimulatorStartTrigger |  |
| DigOutStimulatorStopTrigger  |  |
| DigStreamToReceiver          |  |

**10.2.1.26 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.  |

**10.2.1.27 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    |  |

**10.2.1.28 enCMosMeaChipType** enum [enCMosMeaChipType](#) [strong]

## Enumerator

|          |  |
|----------|--|
| unknown  |  |
| nMos16LV |  |
| nMos32LV |  |
| nMos36LN |  |
| nMos64LN |  |

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

## Enumerator

|                  |  |
|------------------|--|
| OK               |  |
| NOT_CONNECTED    |  |
| DEVICE_NOT_FOUND |  |

**10.2.1.30 FilterAttributeEnumNet** enum [FilterAttributeEnumNet](#) [strong]

**Enumerator**

|                |  |
|----------------|--|
| PreCommaB      |  |
| PostCommaB     |  |
| CommaPositionB |  |
| PreCommaA      |  |
| PostCommaA     |  |
| CommaPositionA |  |

**10.2.1.31 FilterBandEnumNet** enum `FilterBandEnumNet` [strong]**Enumerator**

|          |  |
|----------|--|
| Unknown  |  |
| Lowpass  |  |
| Highpass |  |

**10.2.1.32 FilterCalculationDirectionEnumNet** enum `FilterCalculationDirectionEnumNet` [strong]**Enumerator**

|             |  |
|-------------|--|
| DoubleToInt |  |
| IntToDouble |  |

**10.2.1.33 FilterFamilyEnumNet** enum `FilterFamilyEnumNet` [strong]**Enumerator**

|             |  |
|-------------|--|
| Unknown     |  |
| Bessel      |  |
| Butterworth |  |
| RC          |  |

**10.2.1.34 FilterTypeEnumNet** enum `FilterTypeEnumNet` [strong]**Enumerator**

|          |  |
|----------|--|
| Hardware |  |
| Software |  |



**10.2.1.35 FpgaldEnumNet** enum [FpgaIdEnumNet](#) [strong]

Enumerator

|                                  |  |
|----------------------------------|--|
| DeviceNotConnected               |  |
| Mea2100Interfaceboard            |  |
| Mea2100Headstage                 |  |
| Mea2100STG                       |  |
| MultiwellHeadstage               |  |
| MultiwellInterfaceboard          |  |
| TbsiDacqInterfaceboard           |  |
| TbsiDacqHeadstage                |  |
| CmosMeaInterfaceboard            |  |
| CmosMeaHeadstage                 |  |
| 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                      |  |
| Mea2100LiteHeadstage             |  |
| LIH30Interfaceboard              |  |
| LIH30ADCCtrl                     |  |
| UssingRail                       |  |
| UssingChamber                    |  |
| IFB2GoldenInterfaceboard         |  |
| IFB30GoldenInterfaceboard        |  |
| DeviceHasNoHeadstage             |  |

### 10.2.1.36 HeadstageIdEnumNet enum [HeadstageIdEnumNet](#) [strong]

Enumerator

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

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

enumerates the I/O Voltages available on the IFB2

Enumerator

|             |  |
|-------------|--|
| Voltage_3V3 |  |
| Voltage_5V0 |  |

### 10.2.1.38 LIH30\_ADC\_Channel\_EnumNet enum [LIH30\\_ADC\\_Channel\\_EnumNet](#) [strong]

Enumerator

|                |  |
|----------------|--|
| User_ADC_0     |  |
| User_ADC_1     |  |
| User_ADC_2     |  |
| User_ADC_3     |  |
| User_ADC_4     |  |
| Test_ADC_EPC10 |  |
| ModulA_ADC0    |  |
| ModulA_ADC1    |  |
| ModulA_ADC2    |  |
| ModulA_ADC3    |  |
| ModulB_ADC0    |  |

## Enumerator

|             |  |
|-------------|--|
| ModulB_ADC1 |  |
| ModulB_ADC2 |  |
| ModulB_ADC3 |  |
| ModulC_ADC0 |  |
| ModulC_ADC1 |  |
| ModulC_ADC2 |  |
| ModulC_ADC3 |  |
| ModulD_ADC0 |  |
| ModulD_ADC1 |  |
| ModulD_ADC2 |  |
| ModulD_ADC3 |  |

**10.2.1.39 LIH30\_DAC\_Channel\_EnumNet** enum [LIH30\\_DAC\\_Channel\\_EnumNet](#) [strong]

## Enumerator

|                |  |
|----------------|--|
| User_DAC_0     |  |
| User_DAC_1     |  |
| User_DAC_2     |  |
| Test_DAC_EPC10 |  |
| ModulA_DAC0    |  |
| ModulA_DAC1    |  |
| ModulB_DAC0    |  |
| ModulB_DAC1    |  |
| ModulC_DAC0    |  |
| ModulC_DAC1    |  |
| ModulD_DAC0    |  |
| ModulD_DAC1    |  |

**10.2.1.40 LIH30\_EPC10\_Bus\_EnumNet** enum [LIH30\\_EPC10\\_Bus\\_EnumNet](#) [strong]

## Enumerator

|   |  |
|---|--|
| A |  |
| B |  |

**10.2.1.41 MbcChannelStateEnumNet** enum [MbcChannelStateEnumNet](#) [strong]

**Enumerator**

|                         |  |
|-------------------------|--|
| csIdleNoBattery         |  |
| csIdleChargeFinished    |  |
| csCapacityTestPrecharge |  |
| csCapacityTestDischarge |  |
| csRefreshBattery        |  |
| csCharge                |  |
| csDischarge             |  |
| csError                 |  |

**10.2.1.42 MbcChargingModeEnumNet** enum [MbcChargingModeEnumNet](#) [strong]

**Enumerator**

|               |  |
|---------------|--|
| StorageCharge |  |
| FullCharge    |  |

**10.2.1.43 MbcRatedCapacityEnumNet** enum [MbcRatedCapacityEnumNet](#) [strong]

**Enumerator**

|                 |  |
|-----------------|--|
| rc30mAh         |  |
| rc100mAh        |  |
| rc200mAh        |  |
| rc300mAh        |  |
| rcGreater300mAh |  |

**10.2.1.44 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       |  |

**10.2.1.45 McsUsbSpeedEnumNet** enum [McsUsbSpeedEnumNet](#) [strong]

Enumerates the current connection speed of the device

Enumerator

|              |  |
|--------------|--|
| LowSpeed     |  |
| FullSpeed    |  |
| HighSpeed    |  |
| SuperSpeed   |  |
| UnknownSpeed |  |

**10.2.1.46 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 |  |

**10.2.1.47 MEA2100\_256DigitalSourceEnumNet** enum [MEA2100\\_256DigitalSourceEnumNet](#) [strong]

Enumerates the digital source of the MEA2100-256 device.

Enumerator

|                    |  |
|--------------------|--|
| DigitalInOfOutPort |  |
| DigitalIn          |  |
| DigitalPulse       |  |
| Feedback           |  |
| AuxIn              |  |
| Zero               |  |

## Enumerator

|                      |  |
|----------------------|--|
| One                  |  |
| DeviceRunStatus      |  |
| PulseGenerator       |  |
| 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    |  |

## Enumerator

|                    |  |
|--------------------|--|
| HS2Trigger8Status  |  |
| HS2Trigger9Status  |  |
| HS2Trigger10Status |  |
| 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       |  |

**10.2.1.48 MeaLayoutEnumNet** enum [MeaLayoutEnumNet](#) [strong]

Enumerates the MEA layout of the MEA2100 device.

## Enumerator

|           |  |
|-----------|--|
| mlUnknown |  |
| mlMEA60   |  |

**10.2.1.49 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_100PBA     |  |
| Plate_Dummy_126         |  |
| Plate_24W300_30GMA      |  |
| Plate_96W700_100GMA     |  |
| Plate_24W300_30GBA      |  |
| Plate_96W700_100GBA     |  |
| Plate_24W300_30GBB      |  |
| Plate_96W700_100GBB     |  |
| No_Plate                |  |

**10.2.1.50 PatchServAdcModeEnumNet** enum `PatchServAdcModeEnumNet` [strong]

## Enumerator

|          |  |
|----------|--|
| Normal   |  |
| CatchAmp |  |

**10.2.1.51 PlateClampEnumNet** enum `PlateClampEnumNet` [strong]

## Enumerator

|       |  |
|-------|--|
| Close |  |
| Open  |  |
| Stop  |  |

**10.2.1.52 PlateClampLockEnumNet** enum `PlateClampLockEnumNet` [strong]



## Enumerator

|        |  |
|--------|--|
| Lock   |  |
| Unlock |  |

**10.2.1.53 PortDirectionEnumNet** enum `PortDirectionEnumNet` [strong]

enumerates a port direction

## Enumerator

|        |  |
|--------|--|
| Output |  |
| Input  |  |

**10.2.1.54 PP\_Pump\_Mode\_Type\_EnumNet** enum `PP_Pump_Mode_Type_EnumNet` [strong]

## Enumerator

|         |  |
|---------|--|
| Manual  |  |
| Digital |  |
| Analog  |  |

**10.2.1.55 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                        |  |

## Enumerator

|                      |  |
|----------------------|--|
| HekaLIH406           |  |
| 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                 |  |
| 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              |  |

## Enumerator

|                          |  |
|--------------------------|--|
| 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               |  |
| MEA2100_Lite             |  |
| Multiwell                |  |
| MEA2100_256              |  |
| ME2100                   |  |
| MEA2100BetaScreen        |  |
| MEA2100_Mini             |  |
| TBSI_Dacq                |  |
| Multiwell_MEA_Mini       |  |
| Whole_Cell_Patch         |  |
| eCube                    |  |
| Graphene_ASIC            |  |
| GE2100                   |  |
| Multiboot                |  |
| WPA8                     |  |
| WPA4                     |  |
| WPA16                    |  |
| WPA32                    |  |
| W2100                    |  |
| NeuroChip                |  |
| UsbTest                  |  |
| SoftwareDongle           |  |
| PathIdent                |  |
| NF_Gen                   |  |
| SafelS                   |  |
| Encapsulator             |  |
| NeurochipConfig          |  |
| MeasureTable             |  |

## Enumerator

|                           |  |
|---------------------------|--|
| Roboocyte2                |  |
| RoboInject                |  |
| HiClamp                   |  |
| PatchServer               |  |
| Dilutor                   |  |
| HiClamp4Uart              |  |
| IM16S16KRA                |  |
| IM64KRB                   |  |
| IS32KRA                   |  |
| IM64KRC                   |  |
| IM16S8KRA                 |  |
| IM16KRC                   |  |
| SmartImplant              |  |
| PositionImp               |  |
| PositionBase              |  |
| PositionIICentralUnit     |  |
| PositionIIBase            |  |
| GrapheneProjectTestDevice |  |
| Pos900                    |  |
| Neptun                    |  |
| Warner_Valve_Control      |  |
| Warner_TEER_Machine       |  |
| Warner_Ussing             |  |

### 10.2.1.56 PulseGenerator\_Mode\_EnumNet `enum PulseGenerator_Mode_EnumNet [strong]`

## Enumerator

|                   |  |
|-------------------|--|
| Off               |  |
| AlwaysOn          |  |
| Gated_Low_Active  |  |
| Gated_High_Active |  |

### 10.2.1.57 ReferenceElectrodeModeEnumNet `enum ReferenceElectrodeModeEnumNet [strong]`

enumerates the electrode subtraction modes

## Enumerator

|                                    |  |
|------------------------------------|--|
| SubtractionOff                     |  |
| SubtractFromAllOther               |  |
| SubtractFromReferenceElectrodeOnly |  |
| SubtractFromAll                    |  |

**10.2.1.58 ReferenceElectrodeSwitchPositionEnumNet** enum [ReferenceElectrodeSwitchPositionEnumNet](#) [strong]

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

Enumerator

|       |  |
|-------|--|
| off   |  |
| Ref8  |  |
| Ref16 |  |
| Ref24 |  |
| Ref32 |  |

**10.2.1.59 RetriggerActionEnumNet** enum [RetriggerActionEnumNet](#) [strong]

Enumerates possible retrigger actions for STG200x devices.

Enumerator

|           |  |
|-----------|--|
| raStop    |  |
| raRestart |  |
| raIgnore  |  |
| raGate    |  |
| raSingle  |  |

**10.2.1.60 RoboCurrentModeEnumNet** enum [RoboCurrentModeEnumNet](#) [strong]

Enumerator

|           |  |
|-----------|--|
| Off       |  |
| Break     |  |
| Standby   |  |
| Reference |  |
| Movement  |  |

**10.2.1.61 SampleDstSizeNet** enum [SampleDstSizeNet](#) [strong]

Enumerates the destination data format for ChannelBlock functions.

**Enumerator**

|                 |  |
|-----------------|--|
| SampleDstSize16 |  |
| SampleDstSize32 |  |

**10.2.1.62 SampleSizeNet** enum [SampleSizeNet](#) [strong]

Enumerates the data format for ChannelBlock functions.

**Enumerator**

|                      |  |
|----------------------|--|
| SampleSize16Unsigned |  |
| SampleSize16Signed   |  |
| SampleSize24Unsigned |  |
| SampleSize24Signed   |  |
| SampleSize32Unsigned |  |
| SampleSize32Signed   |  |
| SampleSize64Unsigned |  |
| SampleSize64Signed   |  |

**10.2.1.63 SCU\_HeadstageIdEnumNet** enum [SCU\\_HeadstageIdEnumNet](#) [strong]**Enumerator**

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

**10.2.1.64 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 |  |

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

Enumerates the digital source of the SCU device.

## Enumerator

|                      |  |
|----------------------|--|
| DigitalInOfOutPort   |  |
| DigitalIn            |  |
| DigitalPulse         |  |
| Feedback             |  |
| AuxIn                |  |
| Zero                 |  |
| One                  |  |
| PulseGenerator       |  |
| DigitalOutStimulator |  |
| DigitalData          |  |
| DeviceRunStatus      |  |
| HS1Trigger1Status    |  |
| HS1Trigger2Status    |  |
| HS1Trigger3Status    |  |
| HS1Trigger4Status    |  |
| HS1Trigger5Status    |  |
| HS1Trigger6Status    |  |
| HS1Trigger7Status    |  |

## Enumerator

|                    |  |
|--------------------|--|
| HS1Trigger8Status  |  |
| HS1Trigger9Status  |  |
| HS1Trigger10Status |  |
| HS1Trigger11Status |  |
| HS1Trigger12Status |  |
| HS1Sideband1       |  |
| HS1Sideband2       |  |
| HS1Sideband3       |  |
| HS1Sideband4       |  |
| HS1Sideband5       |  |
| HS1Sideband6       |  |
| HS1Sideband7       |  |
| HS1Sideband8       |  |
| HS1Sideband9       |  |
| HS1Sideband10      |  |
| HS1Sideband11      |  |
| HS1Sideband12      |  |
| HS2Trigger1Status  |  |
| HS2Trigger2Status  |  |
| HS2Trigger3Status  |  |
| HS2Trigger4Status  |  |
| HS2Trigger5Status  |  |
| HS2Trigger6Status  |  |
| HS2Trigger7Status  |  |
| HS2Trigger8Status  |  |
| HS2Trigger9Status  |  |
| HS2Trigger10Status |  |
| HS2Trigger11Status |  |
| HS2Trigger12Status |  |
| HS2Sideband1       |  |
| HS2Sideband2       |  |
| HS2Sideband3       |  |
| HS2Sideband4       |  |
| HS2Sideband5       |  |
| HS2Sideband6       |  |
| HS2Sideband7       |  |
| HS2Sideband8       |  |
| HS2Sideband9       |  |
| HS2Sideband10      |  |
| HS2Sideband11      |  |
| HS2Sideband12      |  |
| LastPosition       |  |

#### 10.2.1.66 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                   |

**10.2.1.67 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 SendSegmentStart() until the next sweep has finished. |

**10.2.1.68 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    |  |

**10.2.1.69 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       |  |

**10.2.1.70 StimulationLayoutConfigurationEnumNet** enum [StimulationLayoutConfigurationEnumNet](#) [strong]

enumerates the layout configuration for the MEA2100-256 device

## Enumerator

|            |  |
|------------|--|
| SingleWell |  |
| SixWell    |  |
| NineWell   |  |

**10.2.1.71 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         |  |

**10.2.1.72 TcxDeviceTypeEnumNet** enum [TcxDeviceTypeEnumNet](#) [strong]

Enumerates the type of TCX devices.

Enumerator

|         |  |
|---------|--|
| Unknown |  |
| Regular |  |
| BMI     |  |
| Nanion  |  |
| Warner  |  |

**10.2.1.73 TcxSensorTypeEnumNet** enum [TcxSensorTypeEnumNet](#) [strong]

Enumerates the sensor types for TCX devices

Enumerator

|           |  |
|-----------|--|
| Reserved5 |  |
| Reserved4 |  |
| Reserved3 |  |
| Reserved2 |  |
| Reserved1 |  |
| NTC10K    |  |
| PT1000    |  |
| PT100     |  |

**10.2.1.74 TeerClampModeEnumNet** enum [TeerClampModeEnumNet](#) [strong]

Enumerator

|                              |  |
|------------------------------|--|
| ClampModeVoltage             |  |
| ClampModeCurrent             |  |
| ClampModeOpen                |  |
| ClampModeInternalCalibration |  |

**10.2.1.75 TeerWaveformEnumNet** enum [TeerWaveformEnumNet](#) [strong]

## Enumerator

|           |  |
|-----------|--|
| Rectangle |  |
| Sine      |  |

**10.2.1.76 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 |  |

**10.2.1.77 UsbVendorIdEnumNet** enum `UsbVendorIdEnumNet` [strong]

## Enumerator

|         |  |
|---------|--|
| Unknown |  |
| None    |  |
| Renesas |  |
| ASMedia |  |
| Intel   |  |

**10.2.1.78 UssingClampModeEnumNet** enum `UssingClampModeEnumNet` [strong]

## Enumerator

|                 |  |
|-----------------|--|
| VoltageClamp    |  |
| CurrentClamp    |  |
| OpenClamp       |  |
| Standby         |  |
| ElectrodeOffset |  |

**10.2.1.79 UssingUnitEnumNet** enum [UssingUnitEnumNet](#) [strong]

Enumerator

|        |  |
|--------|--|
| Volt   |  |
| Ampere |  |
| State  |  |

**10.2.1.80 VendorIdEnumNet** enum [VendorIdEnumNet](#) [strong]

Enumerates the group of MCS devices to connect to.

Enumerator

|         |  |
|---------|--|
| Any     |  |
| None    |  |
| MCS     |  |
| PCI     |  |
| Cypress |  |
| ALA_VC3 |  |

**10.2.1.81 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      |  |

**10.2.1.82 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                |  |

**10.2.1.83 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          |  |

**10.2.1.84 WvcDisplayModeEnumNet** enum [WvcDisplayModeEnumNet](#) [strong]

enumerates Wvc display mode

Enumerator

|           |  |
|-----------|--|
| Work      |  |
| PC        |  |
| Settings  |  |
| TouchTest |  |

**10.2.1.85 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](#)<sup>^</sup> *dacq*,   
    int *CbHandle*,   
    int *numFrames* )

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

Delegate to show a device arrival or removal.

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

**10.2.2.4 OnMcsUsbDeviceState()** public delegate void OnMcsUsbDeviceState (   
usbSetupPacket\_t^ request )

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

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

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

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

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

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

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

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

## 11 Class Documentation

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

#### Public Attributes

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

#### 11.1.1 Member Data Documentation

**11.1.1.1 `amplification`** `int` `amplification`

**11.1.1.2 `channel`** `int` `channel`

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

### 11.2 BatteryState Class Reference

#### Properties

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

#### 11.2.1 Property Documentation

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

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

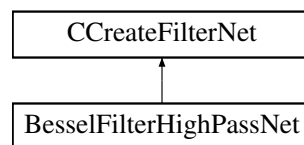
**11.2.1.3 ChargeString** `System:: String^ ChargeString [get]`

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

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

## 11.3 BesselFilterHighPassNet Class Reference

Inheritance diagram for BesselFilterHighPassNet:



### Public Member Functions

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

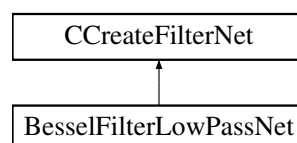
### Additional Inherited Members

#### 11.3.1 Constructor & Destructor Documentation

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

## 11.4 BesselFilterLowPassNet Class Reference

Inheritance diagram for BesselFilterLowPassNet:



## Public Member Functions

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

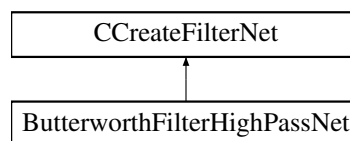
## Additional Inherited Members

### 11.4.1 Constructor & Destructor Documentation

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

## 11.5 ButterworthFilterHighPassNet Class Reference

Inheritance diagram for ButterworthFilterHighPassNet:



## Public Member Functions

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

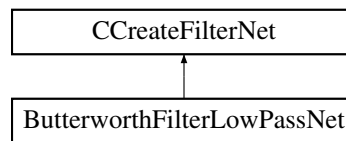
## Additional Inherited Members

### 11.5.1 Constructor & Destructor Documentation

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

## 11.6 ButterworthFilterLowPassNet Class Reference

Inheritance diagram for ButterworthFilterLowPassNet:



### Public Member Functions

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

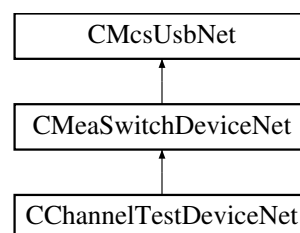
### Additional Inherited Members

#### 11.6.1 Constructor & Destructor Documentation

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

## 11.7 CChannelTestDeviceNet Class Reference

Inheritance diagram for CChannelTestDeviceNet:



### Public Member Functions

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

## Additional Inherited Members

### 11.7.1 Constructor & Destructor Documentation

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

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

### 11.7.2 Member Function Documentation

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

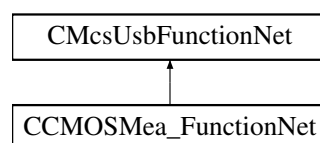
**11.7.2.2 SetAttenuation()** `void SetAttenuation (`  
`unsigned int Attenuation )`

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

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

## 11.8 CCMOSMea\_FunctionNet Class Reference

Inheritance diagram for CCMOSMea\_FunctionNet:





## Public Member Functions

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

- `List< bool > ^ GetEnabledChannelsInGroup (DacqGroupChannelEnumNet GroupID, uint32_t virtualDevice)`
- `SampleSizeNet GetGroupSampleSize (DacqGroupChannelEnumNet GroupID)`
- `SampleSizeNet GetGroupSampleSize (DacqGroupChannelEnumNet GroupID, uint32_t virtualDevice)`
- `uint32_t GetGroupResolutionPerDigit (DacqGroupChannelEnumNet GroupID)`
- `uint32_t GetGroupResolutionPerDigit (DacqGroupChannelEnumNet GroupID, uint32_t virtualDevice)`
- `CMOSMeaValueUnitEnumNet GetGroupUnit (DacqGroupChannelEnumNet GroupID)`
- `CMOSMeaValueUnitEnumNet GetGroupUnit (DacqGroupChannelEnumNet GroupID, uint32_t virtualDevice)`
- `int32_t GetGroupDCOffset (DacqGroupChannelEnumNet GroupID)`
- `int32_t GetGroupDCOffset (DacqGroupChannelEnumNet GroupID, uint32_t virtualDevice)`
- `int32_t GetGroupADCBits (DacqGroupChannelEnumNet GroupID)`
- `int32_t GetGroupADCBits (DacqGroupChannelEnumNet GroupID, uint32_t virtualDevice)`
- `uint32_t GetGroupChannelBitmaskBySelect (DacqGroupChannelEnumNet GroupID, uint32_t Channel↵ Number)`
- `uint32_t GetGroupChannelBitmaskBySelect (DacqGroupChannelEnumNet GroupID, uint32_t Channel↵ Number, uint32_t virtualDevice)`
- `CMOSMeaInterfaceADCEnumNet GetGroupChannelBitmaskInterfaceADC (uint32_t ChannelNumber)`
- `CMOSMeaInterfaceADCEnumNet GetGroupChannelBitmaskInterfaceADC (uint32_t ChannelNumber, uint32_t virtualDevice)`
- `CMOSMeaIFDigChannelEnumNet GetGroupChannelBitmaskIFDigChannels (uint32_t ChannelNumber)`
- `CMOSMeaIFDigChannelEnumNet GetGroupChannelBitmaskIFDigChannels (uint32_t ChannelNumber, uint32_t virtualDevice)`
- `CMOSMeaHeadstage1NCBathCurrentEnumNet GetGroupChannelBitmaskHS1NCBathCurrent (uint32_t ChannelNumber)`
- `CMOSMeaHeadstage1NCBathCurrentEnumNet GetGroupChannelBitmaskHS1NCBathCurrent (uint32_t ChannelNumber, uint32_t virtualDevice)`
- `CMOSMeaHeadstage1NCCol2CurrentEnumNet GetGroupChannelBitmaskHS1NCCol2Current (uint32_t ChannelNumber)`
- `CMOSMeaHeadstage1NCCol2CurrentEnumNet GetGroupChannelBitmaskHS1NCCol2Current (uint32_t ChannelNumber, uint32_t virtualDevice)`
- `CMOSMeaHeadstage1NChipTempEnumNet GetGroupChannelBitmaskHS1NChipTemp (uint32_t Channel↵ Number)`
- `CMOSMeaHeadstage1NChipTempEnumNet GetGroupChannelBitmaskHS1NChipTemp (uint32_t Channel↵ Number, uint32_t virtualDevice)`
- `CMOSMeaSTG1DACSignalEnumNet GetGroupChannelBitmaskSTG1DACSignal (uint32_t 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

### 11.8.1 Constructor & Destructor Documentation

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

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

## 11.8.2 Member Function Documentation

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

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

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

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

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

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

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

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

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

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

**11.8.2.11 GetGate()** `int32_t` GetGate ( )

**11.8.2.12 GetGNDI()** `int32_t` GetGNDI ( )

**11.8.2.13 GetGroupADCBits()** [1/2] `int32_t` GetGroupADCBits (   
 `DacqGroupChannelEnumNet` *GroupID* )

**11.8.2.14 GetGroupADCBits()** [2/2] `int32_t` GetGroupADCBits (   
 `DacqGroupChannelEnumNet` *GroupID*,   
 `uint32_t` *virtualDevice* )

**11.8.2.15 GetGroupChannelBitmaskBySelect()** [1/2] `uint32_t` GetGroupChannelBitmaskBySelect (   
 `DacqGroupChannelEnumNet` *GroupID*,   
 `uint32_t` *ChannelNumber* )

**11.8.2.16 GetGroupChannelBitmaskBySelect()** [2/2] `uint32_t` GetGroupChannelBitmaskBySelect (   
 `DacqGroupChannelEnumNet` *GroupID*,   
 `uint32_t` *ChannelNumber*,   
 `uint32_t` *virtualDevice* )

**11.8.2.17 GetGroupChannelBitmaskHS1NCBathCurrent()** [1/2] [CMOSMeaHeadstage1NCBathCurrentEnumNet](#)

```
GetGroupChannelBitmaskHS1NCBathCurrent (
 uint32_t ChannelNumber)
```

**11.8.2.18 GetGroupChannelBitmaskHS1NCBathCurrent()** [2/2] [CMOSMeaHeadstage1NCBathCurrentEnumNet](#)

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

**11.8.2.19 GetGroupChannelBitmaskHS1NCCol2Current()** [1/2] [CMOSMeaHeadstage1NCCol2CurrentEnumNet](#)

```
GetGroupChannelBitmaskHS1NCCol2Current (
 uint32_t ChannelNumber)
```

**11.8.2.20 GetGroupChannelBitmaskHS1NCCol2Current()** [2/2] [CMOSMeaHeadstage1NCCol2CurrentEnumNet](#)

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

**11.8.2.21 GetGroupChannelBitmaskHS1NChipTemp()** [1/2] [CMOSMeaHeadstage1NChipTempEnumNet](#) Get↔

```
GroupChannelBitmaskHS1NChipTemp (
 uint32_t ChannelNumber)
```

**11.8.2.22 GetGroupChannelBitmaskHS1NChipTemp()** [2/2] [CMOSMeaHeadstage1NChipTempEnumNet](#) Get↔

```
GroupChannelBitmaskHS1NChipTemp (
 uint32_t ChannelNumber,
 uint32_t virtualDevice)
```

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

```
ChannelBitmaskHS1Sidebands (
 uint32_t ChannelNumber)
```

**11.8.2.24 GetGroupChannelBitmaskHS1Sidebands()** [2/2] [CMOSMeaHS1SidebandEnumNet](#) GetGroup↔

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

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

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

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

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

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

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

**11.8.2.31 GetGroupChannelBitmaskPacketFrameContext()** [1/2] [CMOSMeaPacketFrameContextGroupEnumNet](#)  
GetGroupChannelBitmaskPacketFrameContext (   
    uint32\_t *ChannelNumber* )

**11.8.2.32 GetGroupChannelBitmaskPacketFrameContext()** [2/2] [CMOSMeaPacketFrameContextGroupEnumNet](#)  
GetGroupChannelBitmaskPacketFrameContext (   
    uint32\_t *ChannelNumber*,  
    uint32\_t *virtualDevice* )

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

**11.8.2.49 GetMaxNumOfColumns()** `uint32_t GetMaxNumOfColumns (`  
    `uint32_t Samplerate )`

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



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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

**11.8.2.72 SetSourceBulk()** `void SetSourceBulk (`  
`int32_t voltage )`

**11.8.2.73 SetSourceDrain()** `void SetSourceDrain (`  
`int32_t voltage )`

**11.8.2.74 SetSourceGate()** `void SetSourceGate (`  
`int32_t voltage )`

**11.8.2.75 SetStimulusSites()** `void SetStimulusSites (`  
`List< int16_t >^ SwitchPosition )`

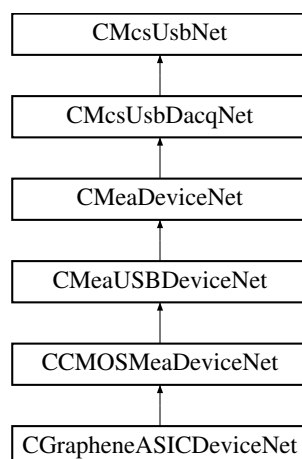
**11.8.2.76 UpdateTransistorVoltages()** `void UpdateTransistorVoltages ( )`

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

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

## 11.9 CCMOSMeaDeviceNet Class Reference

Inheritance diagram for CCMOSMeaDeviceNet:



## Classes

- class [CRegionOfInterestRect](#)

## Public Member Functions

- [CCMOSMeaDeviceNet](#) (void)
- [~CCMOSMeaDeviceNet](#) ()
- virtual void [SetBaseSamplerate](#) (int BaseSamplerate)
- 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

### 11.9.1 Constructor & Destructor Documentation

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

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

### 11.9.2 Member Function Documentation

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

Reimplemented in [CGrapheneASICDeviceNet](#).

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

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

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

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

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

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

**11.9.2.8 GetMaxReadableColumns()** `int GetMaxReadableColumns ( )`

**11.9.2.9 SetBaseSamplerate()** `virtual void SetBaseSamplerate (   
int BaseSamplerate ) [virtual]`

Reimplemented in [CGrapheneASICDeviceNet](#).

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

**11.9.2.11 UpdateChannelBlock()** `void UpdateChannelBlock (   
int queuesize,   
int threshold,   
int channels_in_block )`

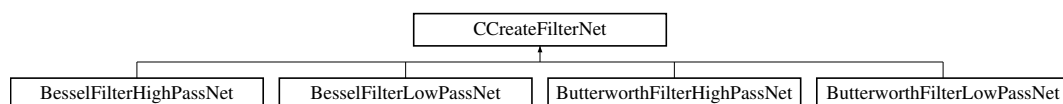
### 11.9.3 Property Documentation

**11.9.3.1 CMosMea** [CCMOSMea\\_FunctionNet](#)^ `CMosMea [get]`

**11.9.3.2 Stimulus** [CStimulusFunctionNet](#)^ `Stimulus [get]`

## 11.10 CCreateFilterNet Class Reference

Inheritance diagram for CCreateFilterNet:



### Public Member Functions

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

**Static Public Member Functions**

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

**Protected Member Functions**

- [CCreateFilterNet](#) (int numCoefSets, CCreateFilter \*pCreateFilter)

**Properties**

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

**11.10.1 Constructor & Destructor Documentation**

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

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

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

**11.10.2 Member Function Documentation**

**11.10.2.1 FindFilter() [1/2]** static int [FindFilter](#) (   
array< array< uint64\_t >>^ *coef*,   
array< [CCreateFilterNet](#)>>^ *param*,   
[CFilterCoefficientsNet::s\\_FilterAttributesNet](#)& *FiltAttr*,   
bool *DoMCSLegacyCompare* ) [static]

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

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

**11.10.2.4 GetBiQuads()** array<CFilterCoefficientsNet^> ^ GetBiQuads ( )

### 11.10.3 Property Documentation

**11.10.3.1 CutoffFrequency** double CutoffFrequency [get]

**11.10.3.2 NumCoefSets** int NumCoefSets [get]

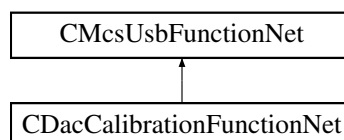
**11.10.3.3 Order** int Order [get]

**11.10.3.4 SampleRate** double SampleRate [get]

**11.10.3.5 Scale** double Scale [get]

## 11.11 CDacCalibrationFunctionNet Class Reference

Inheritance diagram for CDacCalibrationFunctionNet:





**Public Member Functions**

- [CDacCalibrationFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> pDacCalibrationFunctionPointerContainer)  
*Initializes a new instance of the [CDacCalibrationFunctionNet](#) class.*
- [CDacCalibrationFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> 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****11.11.1 Detailed Description****11.11.2 Constructor & Destructor Documentation**

**11.11.2.1 [CDacCalibrationFunctionNet](#)() [1/2]** [CDacCalibrationFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb,  
[CMcsUsbFunctionPointerContainer](#)<sup>^</sup> pDacCalibrationFunctionPointerContainer )

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

**11.11.2.2 [CDacCalibrationFunctionNet](#)() [2/2]** [CDacCalibrationFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb )

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

**11.11.2.4 [!CDacCalibrationFunctionNet](#)()** [!CDacCalibrationFunctionNet](#) ( )

**11.11.3 Member Function Documentation**

**11.11.3.1 [BurnDacOffset](#)()** void [BurnDacOffset](#) (  
uint16\_t dacChannel )

Writes the offset of a DAC channel to permanent memory.

**Parameters**

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

**11.11.3.2 GetDacOffset()** `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.

**11.11.3.3 SetDacOffset()** `void SetDacOffset (`  
     `uint16_t dacChannel,`  
     `int32_t offset )`

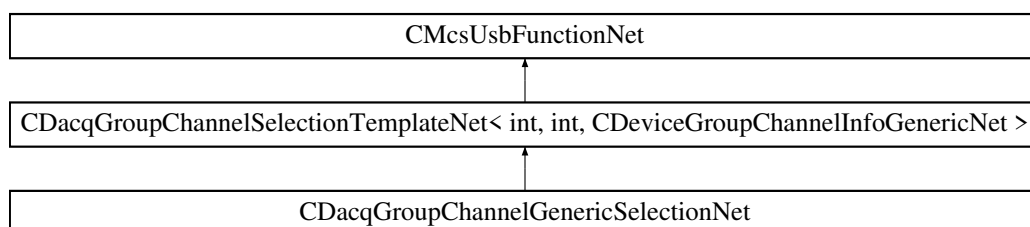
Sets the offset of a DAC channel.

**Parameters**

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

## 11.12 CDacqGroupChannelGenericSelectionNet Class Reference

Inheritance diagram for CDacqGroupChannelGenericSelectionNet:

**Public Member Functions**

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

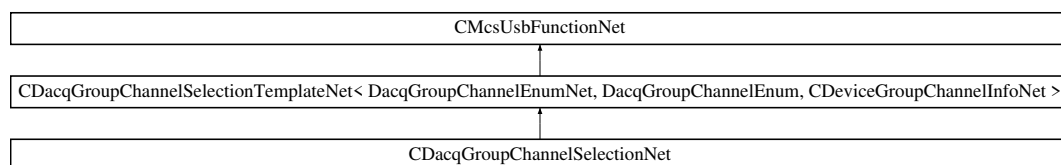
## Additional Inherited Members

### 11.12.1 Constructor & Destructor Documentation

#### 11.12.1.1 CDacqGroupChannelGenericSelectionNet() `CDacqGroupChannelGenericSelectionNet` ( `CMcsUsbNet`^ *mcsusb* )

## 11.13 CDacqGroupChannelSelectionNet Class Reference

Inheritance diagram for CDacqGroupChannelSelectionNet:



## Public Member Functions

- `CDacqGroupChannelSelectionNet` (`CMcsUsbNet`^ *mcsusb*)

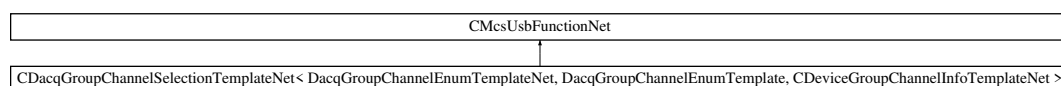
## Additional Inherited Members

### 11.13.1 Constructor & Destructor Documentation

#### 11.13.1.1 CDacqGroupChannelSelectionNet() `CDacqGroupChannelSelectionNet` ( `CMcsUsbNet`^ *mcsusb* )

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

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



## Public Member Functions

- [CDacqGroupChannelSelectionTemplateNet](#) ([CMcsUsbNet](#)<sup>^</sup> *mcsusb*)
- [uint32\\_t GetNumberOfSupportedGroups](#) ()
- [uint32\\_t GetNumberOfSupportedGroups](#) ([uint32\\_t](#) *virtualDevice*)
- [DacqGroupChannelEnumTemplateNet GetGroupID](#) ([uint32\\_t](#) *Index*)
- [DacqGroupChannelEnumTemplateNet GetGroupID](#) ([uint32\\_t](#) *Index*, [uint32\\_t](#) *virtualDevice*)
- [uint32\\_t GetGroupNumberOfChannels](#) ([DacqGroupChannelEnumTemplateNet](#) *GroupID*)
- [uint32\\_t GetGroupNumberOfChannels](#) ([DacqGroupChannelEnumTemplateNet](#) *GroupID*, [uint32\\_t](#) *virtualDevice*)
- [DacqMeaGroupTypeEnumNet GetGroupType](#) ([DacqGroupChannelEnumTemplateNet](#) *GroupID*)
- [DacqMeaGroupTypeEnumNet GetGroupType](#) ([DacqGroupChannelEnumTemplateNet](#) *GroupID*, [uint32\\_t](#) *virtualDevice*)
- [void EnableChannelsInGroup](#) ([DacqGroupChannelEnumTemplateNet](#) *GroupID*, [List< bool >](#)<sup>^</sup> *EnabledChannelsBitMap*)
- [void EnableChannelsInGroup](#) ([DacqGroupChannelEnumTemplateNet](#) *GroupID*, [List< bool >](#)<sup>^</sup> *EnabledChannelsBitMap*, [uint32\\_t](#) *virtualDevice*)
- [List< bool >](#)<sup>^</sup> [GetEnabledChannelsInGroup](#) ([DacqGroupChannelEnumTemplateNet](#) *GroupID*)
- [List< bool >](#)<sup>^</sup> [GetEnabledChannelsInGroup](#) ([DacqGroupChannelEnumTemplateNet](#) *GroupID*, [uint32\\_t](#) *virtualDevice*)
- [SampleSizeNet GetGroupSampleSize](#) ([DacqGroupChannelEnumTemplateNet](#) *GroupID*)
- [SampleSizeNet GetGroupSampleSize](#) ([DacqGroupChannelEnumTemplateNet](#) *GroupID*, [uint32\\_t](#) *virtualDevice*)
- [List< CDeviceGroupChannelInfoTemplateNet](#)<sup>^</sup><sup>></sup> [GetDeviceGroupChannelInfos](#) ()
- [List< CDeviceGroupChannelInfoTemplateNet](#)<sup>^</sup><sup>></sup> [GetDeviceGroupChannelInfos](#) ([uint32\\_t](#) *virtualDevice*)

## Additional Inherited Members

### 11.14.1 Constructor & Destructor Documentation

**11.14.1.1 CDacqGroupChannelSelectionTemplateNet()** [CDacqGroupChannelSelectionTemplateNet](#) ([CMcsUsbNet](#)<sup>^</sup> *mcsusb* )

### 11.14.2 Member Function Documentation

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

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

## 11.14 CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet > Class Template Reference#19

**11.14.2.3 GetDeviceGroupChannelInfos()** [1/2] List<CDeviceGroupChannelInfoTemplateNet^> ^ Get↔  
DeviceGroupChannelInfos ( )

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

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

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

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

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

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

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

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

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

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

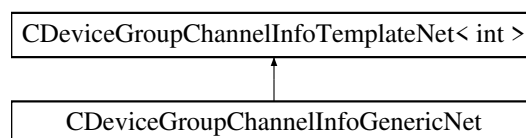
**11.14.2.14 GetGroupType()** [2/2] [DacqMeaGroupTypeEnumNet](#) GetGroupType (   
     DacqGroupChannelEnumTemplateNet GroupID,   
     uint32\_t virtualDevice )

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

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

## 11.15 CDeviceGroupChannelInfoGenericNet Class Reference

Inheritance diagram for CDeviceGroupChannelInfoGenericNet:



### Public Member Functions

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

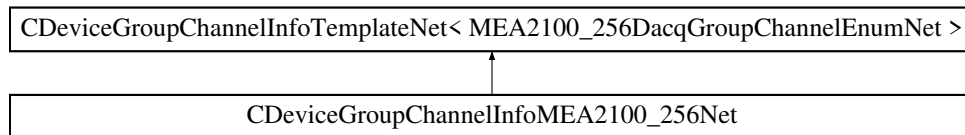
### Additional Inherited Members

#### 11.15.1 Constructor & Destructor Documentation

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

## 11.16 CDeviceGroupChannelInfoMEA2100\_256Net Class Reference

Inheritance diagram for CDeviceGroupChannelInfoMEA2100\_256Net:



### Public Member Functions

- `CDeviceGroupChannelInfoMEA2100_256Net` (`MEA2100_256DacqGroupChannelEnumNet id`, `int channels`, `DacqMeaGroupTypeEnumNet type`)

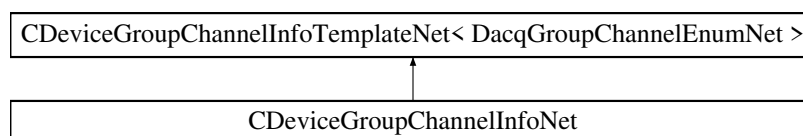
### Additional Inherited Members

#### 11.16.1 Constructor & Destructor Documentation

**11.16.1.1 CDeviceGroupChannelInfoMEA2100\_256Net()** `CDeviceGroupChannelInfoMEA2100_256Net` (   
     `MEA2100_256DacqGroupChannelEnumNet id,`  
     `int channels,`  
     `DacqMeaGroupTypeEnumNet type` )

## 11.17 CDeviceGroupChannelInfoNet Class Reference

Inheritance diagram for CDeviceGroupChannelInfoNet:



### Public Member Functions

- `CDeviceGroupChannelInfoNet` (`DacqGroupChannelEnumNet id`, `int channels`, `DacqMeaGroupTypeEnumNet type`)

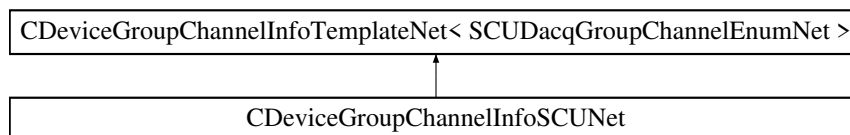
## Additional Inherited Members

### 11.17.1 Constructor & Destructor Documentation

**11.17.1.1 CDeviceGroupChannelInfoNet()** `CDeviceGroupChannelInfoNet` (  
    `DacqGroupChannelEnumNet` *id*,  
    `int` *channels*,  
    `DacqMeaGroupTypeEnumNet` *type* )

## 11.18 CDeviceGroupChannelInfoSCUNet Class Reference

Inheritance diagram for CDeviceGroupChannelInfoSCUNet:



## Public Member Functions

- `CDeviceGroupChannelInfoSCUNet` (`SCUDacqGroupChannelEnumNet` *id*, `int` *channels*, `DacqMeaGroupTypeEnumNet` *type*)

## Additional Inherited Members

### 11.18.1 Constructor & Destructor Documentation

**11.18.1.1 CDeviceGroupChannelInfoSCUNet()** `CDeviceGroupChannelInfoSCUNet` (  
    `SCUDacqGroupChannelEnumNet` *id*,  
    `int` *channels*,  
    `DacqMeaGroupTypeEnumNet` *type* )

## 11.19 CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumTemplateNet > Class Template Reference

## Public Member Functions

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



**Public Attributes**

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

**11.19.1 Constructor & Destructor Documentation**

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

**11.19.2 Member Data Documentation**

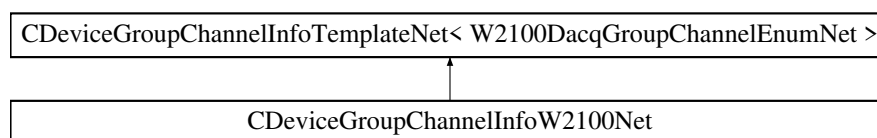
**11.19.2.1 GroupID** [DacqGroupChannelEnumTemplateNet](#) [GroupID](#)

**11.19.2.2 GroupType** [DacqMeaGroupTypeEnumNet](#) [GroupType](#)

**11.19.2.3 NumberOfChannels** int [NumberOfChannels](#)

**11.20 CDeviceGroupChannelInfoW2100Net Class Reference**

Inheritance diagram for CDeviceGroupChannelInfoW2100Net:

**Public Member Functions**

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

## Additional Inherited Members

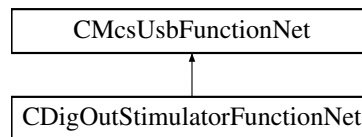
### 11.20.1 Constructor & Destructor Documentation

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

## 11.21 CDigOutStimulatorFunctionNet Class Reference

`CDigOutStimulatorFunctionNet` is the class of the DigOut stimulator function class.

Inheritance diagram for `CDigOutStimulatorFunctionNet`:



### Public Member Functions

- `CDigOutStimulatorFunctionNet (CMcsUsbNet^ 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

### 11.21.1 Detailed Description

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

### 11.21.2 Constructor & Destructor Documentation

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

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

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

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

**11.21.2.4 !CDigOutStimulatorFunctionNet()** [!CDigOutStimulatorFunctionNet](#) ( )

### 11.21.3 Member Function Documentation

**11.21.3.1 ClearChannel()** void ClearChannel (  
    int32\_t *NrChannel* )

clear stimulation pattern

#### Parameters

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

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

get repeat whole stimulation pattern

**Parameters**

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

**Returns**

current value

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

get the number of channels available on the device

**Returns**

the number of channels

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

queries start condition of digital out stimulator

**Parameters**

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

**Returns**

start condition (rising or falling edge)

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

queries stop condition of digital out stimulator

**Parameters**

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

**Returns**

stop condition (rising or falling edge)

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

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

**Parameters**

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

**Returns**

**11.21.3.7 SendChannelData()** `void SendChannelData (`  
`int32_t NrChannel,`  
`CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData^ device_data_and_unrolled`  
`)`

send or append stimulation pattern

**Parameters**

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

**11.21.3.8 SetGlobalRepeat()** `void SetGlobalRepeat (`  
`int32_t NrChannel,`  
`bool value )`

set repeat whole stimulation pattern

**Parameters**

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

**11.21.3.9 SetStartTriggerSlope()** `void SetStartTriggerSlope (`  
`int32_t NrChannel,`  
`DigitalStimulatorTriggerSlopeEnumNet Condition )`

sets start condition of digital out stimulator

Parameters

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

**11.21.3.10 SetStopTriggerSlope()** `void SetStopTriggerSlope (`  
`int32_t NrChannel,`  
`DigitalStimulatorTriggerSlopeEnumNet Condition )`

sets stop condition of digital out stimulator

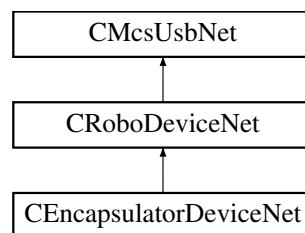
Parameters

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

## 11.22 CEncapsulatorDeviceNet Class Reference

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

Inheritance diagram for CEncapsulatorDeviceNet:



### Public Member Functions

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

### Additional Inherited Members

#### 11.22.1 Detailed Description

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

### 11.22.2 Constructor & Destructor Documentation

**11.22.2.1 CEncapsulatorDeviceNet()** `CEncapsulatorDeviceNet ( void )`

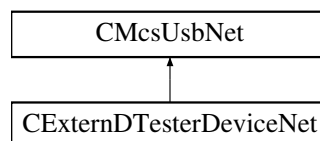
### 11.22.3 Member Function Documentation

**11.22.3.1 GetRoboFluidDevice()** `CRoboFluidDeviceNet ^ GetRoboFluidDevice ( )`

## 11.23 CExternDTesterDeviceNet Class Reference

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

Inheritance diagram for `CExternDTesterDeviceNet`:



### Public Member Functions

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

### Additional Inherited Members

#### 11.23.1 Detailed Description

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

## 11.23.2 Constructor & Destructor Documentation

### 11.23.2.1 CExternDTesterDeviceNet() `CExternDTesterDeviceNet ( )`

Initializes a new instance of the `CExternDTesterDeviceNet` class.

### 11.23.2.2 ~CExternDTesterDeviceNet() `virtual ~CExternDTesterDeviceNet ( ) [virtual]`

### 11.23.2.3 "!CExternDTesterDeviceNet() `!CExternDTesterDeviceNet ( )`

## 11.23.3 Member Function Documentation

### 11.23.3.1 Read() `array<uint8_t> ^ Read ( int configString_Length )`

Reads the config string from the device.

#### Parameters

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

#### Returns

The config string.

### 11.23.3.2 Read2() `String ^ Read2 ( )`

Reads the config string from the device.

#### Returns

The config string.

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

Reads the config string from the device.



## Parameters

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

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

Reads the config string from the device.

## Parameters

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

## 11.24 CFilterCoefficientsNet Class Reference

### Classes

- struct [s\\_FilterAttributesNet](#)

### Public Member Functions

- [CFilterCoefficientsNet](#) ()
- [CFilterCoefficientsNet](#) (double b0, double b1, double b2, double a1, double a2)
- [CFilterCoefficientsNet](#) (double b0, double b1, double a1)
- [CFilterCoefficientsNet](#) (array< double >^ b, array< double >^ a)
- [~CFilterCoefficientsNet](#) ()
- bool [IsEqual](#) (array< uint64\_t >^ coefficients, [s\\_FilterAttributesNet](#)^ FiltAttr)
- bool [IsEqual](#) (array< uint64\_t >^ coefficients, [s\\_FilterAttributesNet](#)^ FiltAttr, bool DoMCSLegacyCompare)
- uint64\_t [GetUIntB](#) (int index, [s\\_FilterAttributesNet](#)^ FiltAttr)
- uint64\_t [GetUIntA](#) (int index, [s\\_FilterAttributesNet](#)^ FiltAttr)

### Properties

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

### 11.24.1 Constructor & Destructor Documentation

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

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

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

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

**11.24.1.5 ~CFilterCoefficientsNet()** `~CFilterCoefficientsNet` ( )

## 11.24.2 Member Function Documentation

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

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

**11.24.2.3 IsEqual()** [1/2] `bool IsEqual` (   
array< uint64\_t >^ *coefficients*,  
`s_FilterAttributesNet`^ *FiltAttr* )

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

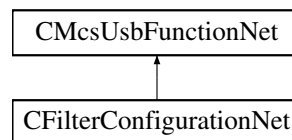
### 11.24.3 Property Documentation

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

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

## 11.25 CFilterConfigurationNet Class Reference

Inheritance diagram for CFilterConfigurationNet:



### Public Member Functions

- `CFilterConfigurationNet (CMcsUsbNet^ 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)`

### Additional Inherited Members

#### 11.25.1 Constructor & Destructor Documentation

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

## 11.25.2 Member Function Documentation

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

**11.25.2.2 GetFilterAttributes()** [1/2] `CFilterCoefficientsNet::s_FilterAttributesNet ^ GetFilterAttributes ( DacqGroupChannelEnumNet GroupID, uint32_t FilterNumber )`

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

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

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

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

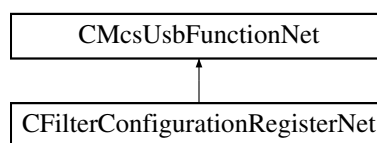
**11.25.2.7 SetFilterParameter()** [2/2] `void SetFilterParameter ( DacqGroupChannelEnumNet GroupID, uint32_t FilterNumber, CFilterCoefficientsNet^ CoefficientsSet1, CFilterCoefficientsNet^ CoefficientsSet2, CFilterPropertyNet^ FilterProp )`

**11.25.2.8 SetFilterParameterPermanent()** void SetFilterParameterPermanent (   
     DacqGroupChannelEnumNet GroupID,   
     uint32\_t FilterNumber )

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

## 11.26 CFilterConfigurationRegisterNet Class Reference

Inheritance diagram for CFilterConfigurationRegisterNet:



### Public Member Functions

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

### Additional Inherited Members

#### 11.26.1 Constructor & Destructor Documentation

**11.26.1.1 CFilterConfigurationRegisterNet()** CFilterConfigurationRegisterNet (   
     CMcsUsbNet^ mcsusb )

#### 11.26.2 Member Function Documentation

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

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

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

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

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

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

## 11.27 CFilterPropertyNet Class Reference

### Public Member Functions

- [CFilterPropertyNet](#) (uint32\_t *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]

### 11.27.1 Constructor & Destructor Documentation

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

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

### 11.27.2 Member Function Documentation

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

### 11.27.3 Property Documentation

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

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

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

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

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

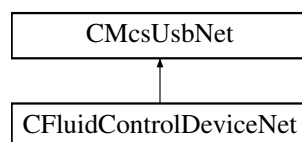
**11.27.3.6 FilterType** `FilterTypeEnumNet FilterType [get]`

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

## 11.28 CFluidControlDeviceNet Class Reference

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

Inheritance diagram for CFluidControlDeviceNet:





## Public Member Functions

- [CFluidControlDeviceNet](#) ()  
*Initialize a new instance of the [CFluidControlDeviceNet](#) class.*
- [~CFluidControlDeviceNet](#) ()  
*Default destructor.*
- void [SetValve](#) (unsigned int value)  
*Open or Close valves.*
- void [SetSingleValve](#) (unsigned short valve, unsigned short onoff)  
*Opens or Closes a valve.*
- void [SetDigout](#) (unsigned int value)  
*Define the pattern on the Digital Output.*
- void [SetPWM](#) (unsigned int channel, unsigned int value)  
*Sets the duty cycle of the PWM output.*
- void [CalibrateThermocouple](#) (unsigned int channel)  
*Calibrates the ADC which is used for the Thermocouple. For the calibration, Short circuit the Thermocouple and use this function to correct a possible offset of the ADC which measures the thermocouple.*
- void [SetThermocoupleNanovoltPerKelvin](#) (unsigned int channel, unsigned int value)  
*Sets the proportional constant for the Thermocouple.*
- unsigned int [GetValve](#) ()  
*Gets the state of the valves.*
- unsigned short [GetSingleValve](#) (unsigned short valve)  
*Gets the state of a valve.*
- unsigned int [GetDigout](#) ()  
*Gets the state of the digital output.*
- unsigned int [GetPWM](#) (unsigned int channel)  
*Gets the state of the PWM output.*
- unsigned int [GetAdc](#) (unsigned int channel)  
*Reads an ADC Value.*
- unsigned int [GetDigin](#) ()  
*Reads the digital input.*
- int [GetThermocoupleTemperature](#) (unsigned int channel)  
*Reads the temperature from Thermocouple. The functions gives the temperature difference between both Thermocouple junctions. To get the absolute temperature, add the reference temperature.*
- int [GetReferenceTemperature](#) (unsigned int channel)  
*Reads the reference temperature for the Thermocouple.*
- unsigned int [GetThermocoupleCalibration](#) (unsigned int channel)  
*Gets the calibration constant for the Thermocouple ADC.*
- unsigned int [GetThermocoupleNanovoltPerKelvin](#) (unsigned int channel)  
*Reads the proportional constant for the Thermocouple.*

## Properties

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

## Additional Inherited Members

### 11.28.1 Detailed Description

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

## 11.28.2 Constructor & Destructor Documentation

### 11.28.2.1 CFluidControlDeviceNet() `CFluidControlDeviceNet ( )`

Initialize a new instance of the `CFluidControlDeviceNet` class.

### 11.28.2.2 ~CFluidControlDeviceNet() `~CFluidControlDeviceNet ( )`

Default destructor.

## 11.28.3 Member Function Documentation

### 11.28.3.1 CalibrateThermocouple() `void CalibrateThermocouple ( unsigned int channel )`

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

#### Parameters

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

### 11.28.3.2 GetAdc() `unsigned int GetAdc ( unsigned int channel )`

Reads an ADC Value.

#### Parameters

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

#### Returns

The current ADC value.

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

Reads the digital input.

**Returns**

The bit pattern of the state of the digital inputs.

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

Gets the state of the digital output.

**Returns**

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

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

Gets the state of the PWM output.

**Returns**

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

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

Reads the reference temperature for the Thermocouple.

**Parameters**

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

**Returns**

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

**11.28.3.7 GetSingleValve()** `unsigned short GetSingleValve (`  
`unsigned short valve )`

Gets the state of a valve.

**Parameters**

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

**Returns**

state of the valve

**11.28.3.8 GetThermocoupleCalibration()** `unsigned int GetThermocoupleCalibration ( unsigned int channel )`

Gets the calibration constant for the Thermocouple ADC.

**Parameters**

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

**Returns**

The calibration constant for the Thermocouple ADC.

**11.28.3.9 GetThermocoupleNanovoltPerKelvin()** `unsigned int GetThermocoupleNanovoltPerKelvin ( unsigned int channel )`

Reads the proportional constant for the Thermocouple.

**Parameters**

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

**Returns**

The proportional constant in Nanovolt per Kelvin.

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

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

## Parameters

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

## Returns

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

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

Gets the state of the valves.

## Returns

The current state of the valves as a bit pattern.

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

Define the pattern on the Digital Output.

## Parameters

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

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

Sets the duty cycle of the PWM output.

## Parameters

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

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

Opens or Closes a valve.

## Parameters

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

## Parameters

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

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

Sets the proportinal constant for the Thermocouple.

## Parameters

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

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

Open or Close valves.

## Parameters

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

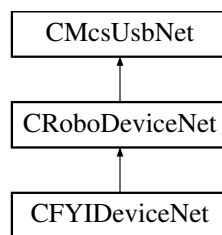
## 11.28.4 Property Documentation

**11.28.4.1 McsBus\_VoltageMode** `CMcsBus_VoltageModeNet^ McsBus_VoltageMode [get]`

## 11.29 CFYIDeviceNet Class Reference

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

Inheritance diagram for CFYIDeviceNet:



### Public Member Functions

- [CFYIDeviceNet](#) (void)

### Properties

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

### Additional Inherited Members

#### 11.29.1 Detailed Description

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

#### 11.29.2 Constructor & Destructor Documentation

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

#### 11.29.3 Property Documentation

**11.29.3.1** [FYIProgram](#) [CRobo\\_FYIProgram\\_FunctionNet](#)^ [FYIProgram](#) [get]

**11.29.3.2** [FYITemp](#) [CRobo\\_FYITemp\\_FunctionNet](#)^ [FYITemp](#) [get]

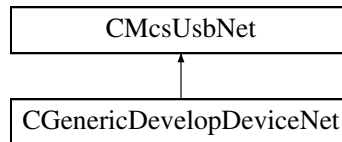


### 11.29.3.3 Sensor `CMcsBus_SensorNet`^ Sensor [get]

## 11.30 CGenericDevelopDeviceNet Class Reference

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

Inheritance diagram for `CGenericDevelopDeviceNet`:



### Public Member Functions

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

#### Parameters

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

#### Parameters

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

- template<typename C >  
void `SetBuffer` (uint16\_t value, uint16\_t index, array< C >^ buffer)
- void `SetUByteBuffer` (uint16\_t value, uint16\_t index, array< unsigned char >^ buffer)  
*Sends an array of type unsigned char to the device.*

#### Parameters

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

#### Parameters

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

*Parameters*

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

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

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

*Parameters*

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

*Parameters*

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

*Parameters*

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

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

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

*Parameters*

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

*Parameters*

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

*Parameters*

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

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

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

*Parameters*

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

*Parameters*

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

*Parameters*

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

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

*Parameters*

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

*Parameters*

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

*Parameters*

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

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

*Parameters*

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

*Parameters*

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

*Parameters*

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

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

**Parameters**

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

**Parameters**

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

**Parameters**

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

**Returns**

*The array of data from the device.*

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

**Parameters**

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

**Parameters**

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

**Parameters**

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

**Returns**

*The array of data from the device.*

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

*Parameters*

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

*Parameters*

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

*Parameters*

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

*Returns*

*The array of data from the device.*

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

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

*Parameters*

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

*Parameters*

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

*Parameters*

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

*Returns*

*The array of data from the device.*

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

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

*Parameters*

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

**Parameters**

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

**Parameters**

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

**Returns**

*The array of data from the device.*

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

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

**Parameters**

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

**Parameters**

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

**Parameters**

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

**Returns**

*The array of data from the device.*

- `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

### 11.30.1 Detailed Description

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

### 11.30.2 Constructor & Destructor Documentation

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

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

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

### 11.30.3 Member Function Documentation

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

Close a Pipe to an USB Endpoint.

#### Parameters

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

**11.30.3.2 FindEndpoints()** array<uint8\_t> ^ FindEndpoints (  
uint8\_t *type*,  
uint8\_t *direction* )



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

**11.30.3.4 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.

**11.30.3.5 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.

**11.30.3.6 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.

**11.30.3.7 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.

**11.30.3.8 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.

```
11.30.3.9 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.

**11.30.3.10 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.

**11.30.3.11 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.

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

Reset a Pipe to an USB Endpoint.

**Parameters**

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

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

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

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

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

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

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

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

**11.30.3.20 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. |
|--------------|---------------------------|

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

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

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

Write data to an USB Endpoint.

**Parameters**

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

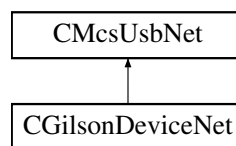
## Parameters

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

## 11.31 CGilsonDeviceNet Class Reference

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

Inheritance diagram for CGilsonDeviceNet:



### Public Member Functions

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

### Protected Attributes

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

### Additional Inherited Members

#### 11.31.1 Detailed Description

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

#### 11.31.2 Constructor & Destructor Documentation

**11.31.2.1 CGilsonDeviceNet()** `CGilsonDeviceNet (`  
`void )`

Initialize a new instance of the `CGilsonDeviceNet` class.

**11.31.2.2 ~CGilsonDeviceNet()** `~CGilsonDeviceNet (`  
`void )`

### 11.31.3 Member Function Documentation

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

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

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

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

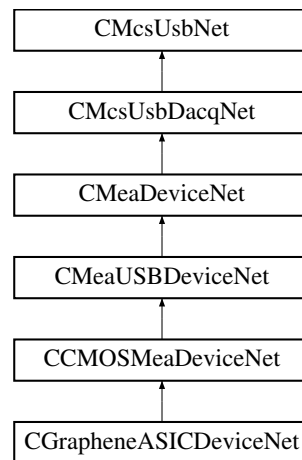
**11.31.3.5 SendImmediateGetResponse()** `String ^ SendImmediateGetResponse (`  
`wchar_t command )`

### 11.31.4 Member Data Documentation

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

## 11.32 CGrapheneASICDeviceNet Class Reference

Inheritance diagram for CGrapheneASICDeviceNet:



### Public Member Functions

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

### Additional Inherited Members

#### 11.32.1 Constructor & Destructor Documentation

**11.32.1.1 CGrapheneASICDeviceNet()** [CGrapheneASICDeviceNet](#) ( void )

**11.32.1.2 ~CGrapheneASICDeviceNet()** [~CGrapheneASICDeviceNet](#) ( )

#### 11.32.2 Member Function Documentation

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

Reimplemented from [CCMOSMeaDeviceNet](#).

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

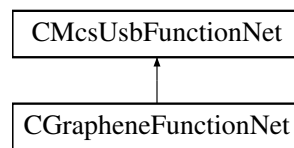
Reimplemented from [CCMOSMeaDeviceNet](#).

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

## 11.33 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*
- 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

### 11.33.1 Detailed Description

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

### 11.33.2 Constructor & Destructor Documentation

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

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

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

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

**11.33.2.4 !CGrapheneFunctionNet()** [!CGrapheneFunctionNet](#) ( )

### 11.33.3 Member Function Documentation

**11.33.3.1 GetCur2VolResistance()** [1/2] [int32\\_t GetCur2VolResistance](#) ( )

Gets the resistance of the current to voltage converter

Returns

The resistance in Ohm

**11.33.3.2 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

**11.33.3.3 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 |

**11.33.3.4 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  |

**11.33.3.5 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 $\mu$ V |
| <i>Vgs</i> | Vgs in $\mu$ V |

**11.33.3.6 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 $\mu\text{V}$    |
| <i>Vgs</i>       | Vgs in $\mu\text{V}$    |

**11.33.3.7 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 $\mu\text{V}$ |
| <i>Vs</i> | Vs in $\mu\text{V}$ |

**11.33.3.8 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 $\mu\text{V}$     |
| <i>Vs</i>        | Vs in $\mu\text{V}$     |

**11.33.3.9 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 |

**11.33.3.10 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         |

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

Gets the voltage range

**Returns**

The voltage range in mV

**11.33.3.12 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

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

Gets the reached voltage

**Returns**

the reached voltage

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

Gets the reached voltage

**Parameters**

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

**Returns**

The reached voltage

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

Gets the voltage resolution

**Returns**

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

**11.33.3.16 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  $\mu\text{V}/\text{digit}$

**11.33.3.17 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 |

**11.33.3.18 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  |

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

Sets Vds and Vgs

Parameters

|            |                      |
|------------|----------------------|
| <i>Vds</i> | Vds in $\mu\text{V}$ |
| <i>Vgs</i> | Vgs in $\mu\text{V}$ |

**11.33.3.20 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 $\mu\text{V}$    |
| <i>Vgs</i>       | Vgs in $\mu\text{V}$    |

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

Sets Vd and Vs

## Parameters

|           |                     |
|-----------|---------------------|
| <i>Vd</i> | Vd in $\mu\text{V}$ |
| <i>Vs</i> | Vs in $\mu\text{V}$ |

**11.33.3.22 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 $\mu\text{V}$     |
| <i>Vs</i>        | Vs in $\mu\text{V}$     |

**11.33.3.23 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 |

**11.33.3.24 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         |

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

Sets the voltage range

Parameters

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

**11.33.3.26 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 |

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

Sets the voltage resolution

Parameters

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



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

Sets the voltage resolution

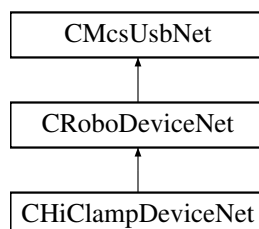
#### Parameters

|                   |                                                      |
|-------------------|------------------------------------------------------|
| <i>Headstage</i>  | The headstage to query.                              |
| <i>resolution</i> | The voltage resolution in $\mu\text{V}/\text{digit}$ |

## 11.34 CHiClampDeviceNet Class Reference

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

Inheritance diagram for CHiClampDeviceNet:



### Public Member Functions

- [CHiClampDeviceNet](#) (void)

### Properties

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

### Additional Inherited Members

#### 11.34.1 Detailed Description

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

#### 11.34.2 Constructor & Destructor Documentation

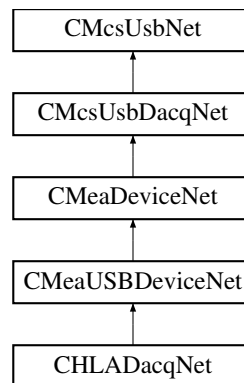
**11.34.2.1 CHiClampDeviceNet()** [CHiClampDeviceNet](#) (  
`void` )

### 11.34.3 Property Documentation

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

## 11.35 CHLADacqNet Class Reference

Inheritance diagram for CHLADacqNet:



### Public Member Functions

- [CHLADacqNet](#) (void)

### Additional Inherited Members

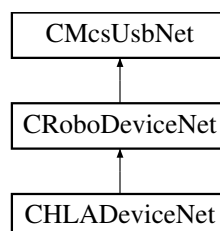
#### 11.35.1 Constructor & Destructor Documentation

##### 11.35.1.1 CHLADacqNet() [CHLADacqNet](#) ( void )

## 11.36 CHLADeviceNet Class Reference

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

Inheritance diagram for CHLADeviceNet:



## Public Member Functions

- [CHLADeviceNet](#) (void)

## Properties

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

## Additional Inherited Members

### 11.36.1 Detailed Description

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

### 11.36.2 Constructor & Destructor Documentation

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

### 11.36.3 Property Documentation

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

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

## 11.37 CMcsUsbDacqNet::CHWInfo Class Reference

Class to provide hardware information about the device.

## Classes

- class [CVoltageRangeInfoNet](#)

## Public Member Functions

- **CHWInfo** (**CMcsUsbDacqNet**<sup>^</sup> device)
- virtual uint32\_t **GetNumberOfHWADCCChannels** ([System::Runtime::InteropServices::Out]int% numberOfChannels)  
*Get the number of analog channels the device supports.*
- virtual uint32\_t **GetNumberOfHWDigitalChannels** ([System::Runtime::InteropServices::Out]int% numberOfChannels)  
*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**<sup>^</sup> > ^ **GetAvailableVoltageRangesInMicroVoltAndStringsInMicroVolt** (int milliGain)  
*Gets a List of voltage ranges the device supports.*

### 11.37.1 Detailed Description

Class to provide hardware information about the device.

### 11.37.2 Constructor & Destructor Documentation

**11.37.2.1 CHWInfo()** **CHWInfo** (**CMcsUsbDacqNet**<sup>^</sup> device )

### 11.37.3 Member Function Documentation

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

**11.37.3.2 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  $\mu\text{V}$ .

**11.37.3.3 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  $\mu\text{V}$ .

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

Get the number of analog channels the device supports.

## Parameters

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

## Returns

Error Status. 0 on success.

**11.37.3.5 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.

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

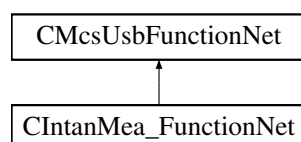
Query if the digital channel replaces an analog channel when enabled (e.g. on MC\_Card) or adds a channel link on USB devices.

#### 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.

## 11.38 CIntanMea\_FunctionNet Class Reference

Inheritance diagram for CIntanMea\_FunctionNet:



#### Public Member Functions

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

## Additional Inherited Members

### 11.38.1 Constructor & Destructor Documentation

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

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

### 11.38.2 Member Function Documentation

**11.38.2.1 AmplifierSettle()** `void AmplifierSettle ( )`

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

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

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

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

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

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

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

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

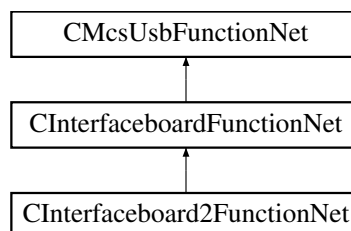
**11.38.2.10 SetDSPHighPassByIndex()** `void SetDSPHighPassByIndex (`  
`int index )`

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

## 11.39 CInterfaceboard2FunctionNet Class Reference

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

Inheritance diagram for CInterfaceboard2FunctionNet:





## Public Member Functions

- [CInterfaceboard2FunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> pInterfaceboard2FunctionPointerContainer)  
*Initializes a new instance of the [CInterfaceboard2FunctionNet](#) class.*
- [CInterfaceboard2FunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> 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

### 11.39.1 Detailed Description

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

### 11.39.2 Constructor & Destructor Documentation

**11.39.2.1 CInterfaceboard2FunctionNet() [1/2]** [CInterfaceboard2FunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb,  
[CMcsUsbFunctionPointerContainer](#)<sup>^</sup> pInterfaceboard2FunctionPointerContainer )

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

**11.39.2.2 CInterfaceboard2FunctionNet() [2/2]** [CInterfaceboard2FunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb )

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

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

### 11.39.3 Member Function Documentation

### 11.39.3.1 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).

### 11.39.3.2 SetIoVoltage() `void SetIoVoltage ( IoVoltageEnumNet ioVoltage )`

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

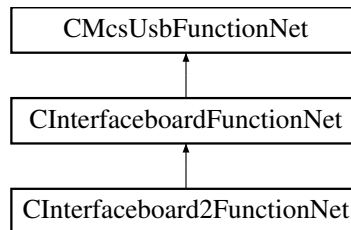
#### Parameters

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

## 11.40 CInterfaceboardFunctionNet Class Reference

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

Inheritance diagram for CInterfaceboardFunctionNet:



### Public Member Functions

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

## Additional Inherited Members

### 11.40.1 Detailed Description

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

### 11.40.2 Constructor & Destructor Documentation

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

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

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

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

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

### 11.40.3 Member Function Documentation

**11.40.3.1 GetCardinalDacqSamplerate()** uint32\_t GetCardinalDacqSamplerate ( )

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

#### Returns

The samplerate in Hz.

**11.40.3.2 GetCardinalStgOutputrate()** uint32\_t GetCardinalStgOutputrate ( )

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

#### Returns

The output rate in Hz.

**11.40.3.3 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. |
|-------------------|-----------------------|

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

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

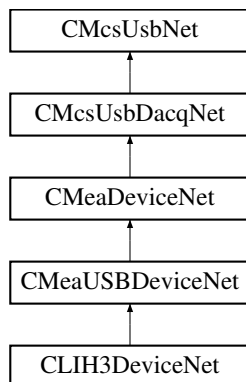
## Parameters

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

## 11.41 CLIH3DeviceNet Class Reference

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

Inheritance diagram for CLIH3DeviceNet:



### Public Member Functions

- [CLIH3DeviceNet](#) ()  
*Initializes a new instance of the [CLIH3DeviceNet](#) class.*
- virtual [~CLIH3DeviceNet](#) ()
- [!CLIH3DeviceNet](#) ()
- void [DummyCommand](#) (uint32\_t dummyParameter)  
*Dummy command to show how to use the DLL.*
- void [SetEEpromPage](#) (uint32\_t EEpromStartAddress, array< int8\_t >^ EEpromData, [LIH30\\_EPC10\\_Bus\\_EnumNet](#) epc10bus)  
*Writes into EEprom on the EPC10 EEPROM*
- array< int8\_t >^ [GetEEpromPage](#) (uint32\_t EEpromStartAddress, int EEpromData\_Length, [LIH30\\_EPC10\\_Bus\\_EnumNet](#) epc10bus)  
*Reads the requested amount of EEprom byte from the EPC10 EEPROM*
- void [SetSampleInterval](#) (uint32\_t SampleInterval)

- Sets the Sample Interval for the DACQ and Stimulation*

  - uint32\_t [GetSampleInterval](#) ()
- Gets the Sample Interval for the DACQ and Stimulation*

  - void [SetAdcOffset](#) (LIH30\_ADC\_Channel\_EnumNet AdcChannel, int32\_t Offset)
- Sets the ADC offset of the DACQ for a single channel*

  - int32\_t [GetAdcOffset](#) (LIH30\_ADC\_Channel\_EnumNet AdcChannel)
- Gets the ADC offset of the DACQ for a single channel*

  - void [SetAdcOffsetPermanent](#) (LIH30\_ADC\_Channel\_EnumNet AdcChannel)
- Writes the ADC offset of the DACQ for a single channel to permanent EEPROM memory*

  - void [ErasePermanentAdcOffset](#) (LIH30\_ADC\_Channel\_EnumNet AdcChannel)
- Deletes the ADC offset of the DACQ for a single channel in permanent EEPROM memory*

  - uint32\_t [ReadClipping](#) (LIH30\_EPC10\_Bus\_EnumNet epc10bus)
- Gets the clipping information*

  - void [SetDigOutState](#) (uint16\_t DigOutState)
- Writes to the LIH30 digital output*

  - uint16\_t [GetDigInState](#) ()
- Reads from the LIH30 digital input*

  - void [SendCommand](#) (LIH30\_EPC10\_Bus\_EnumNet epc10bus, uint16\_t Command)
- Send command to the EPC10*

  - uint16\_t [GetDacqRunStatus](#) ()
- Gets the data acquisition running status*

  - void [SetDacUseldleValue](#) (uint32\_t DacChannel, bool Useldle)
- Sets if the DAC Idle value is used after stimulation*

  - bool [GetDacUseldleValue](#) (uint32\_t DacChannel)
- Gets if the DAC Idle value is used after stimulation*

  - void [SetDacIdleValue](#) (uint32\_t DacChannel, int32\_t IdleValue)
- Sets the DAC Idle value*

  - int32\_t [GetDacIdleValue](#) (uint32\_t DacChannel)
- Gets the DAC Idle value*

  - void [EnableUserTrigger](#) (bool enable)
- Enables the User Trigger*

  - bool [IsUserTriggerEnabled](#) ()
- Is the User Trigger enabled*

  - void [SetDacOffset](#) (LIH30\_DAC\_Channel\_EnumNet DacChannel, int32\_t Offset)
- Sets the offset of a DAC channel.*

  - int32\_t [GetDacOffset](#) (LIH30\_DAC\_Channel\_EnumNet DacChannel)
- Gets the offset of a DAC channel.*

  - void [SetDacOffsetPermanent](#) (LIH30\_DAC\_Channel\_EnumNet DacChannel)
- Writes the DAC offset of the STG for a single channel to permanent EEPROM memory*

  - void [ErasePermanentDacOffset](#) (LIH30\_DAC\_Channel\_EnumNet DacChannel)
- Deletes the DAC offset of the STG for a single channel in permanent EEPROM memory*

  - void [SetAudioOutDacParameter](#) (uint32\_t Frequency, uint32\_t Amplification)
- Sets the parameter of the audio DAC output.*

  - void [GetAudioOutDacParameter](#) ([System::Runtime::InteropServices::Out]uint32\_t% Frequency, [System::Runtime::InteropServices::Out]uint32\_t% Amplification)
- Gets the parameter of the audio DAC output.*

  - String ^ [ReadUARTData](#) ()
- Reads the config string from the device.*

  - void [WriteUARTData](#) (String^ commandString)
- Write the command string to the device.*

## Properties

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

## Additional Inherited Members

### 11.41.1 Detailed Description

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

### 11.41.2 Constructor & Destructor Documentation

#### 11.41.2.1 [CLIH3DeviceNet\(\)](#) [CLIH3DeviceNet](#) ( )

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

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

#### 11.41.2.3 ["!CLIH3DeviceNet\(\)](#) [!CLIH3DeviceNet](#) ( )

### 11.41.3 Member Function Documentation

#### 11.41.3.1 [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 |
|-----------------------|---------------------------------|

#### 11.41.3.2 [EnableUserTrigger\(\)](#) [void EnableUserTrigger](#) ( [bool enable](#) )

Enables the User Trigger

## Parameters

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

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

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

## Parameters

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

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

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

## Parameters

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

**11.41.3.5 GetAdcOffset()** `int32_t GetAdcOffset (   
 LIH30_ADC_Channel_EnumNet AdcChannel )`

Gets the ADC offset of the DACQ for a single channel

## Parameters

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

## Returns

The offset for the given channel number

**11.41.3.6 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) |

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

Gets the DAC Idle value

**Parameters**

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

**Returns**

The idle value

**11.41.3.8 GetDacOffset()** `int32_t GetDacOffset (`  
`LIH30\_DAC\_Channel\_EnumNet DacChannel )`

Gets the offset of a DAC channel.

**Parameters**

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

**Returns**

The offset for the given channel number

**11.41.3.9 GetDacqRunStatus()** `uint16_t GetDacqRunStatus ( )`

Gets the data acquisition running status

**Returns**

The status (1: running / 0: stopped)

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

Gets if the DAC Idle value is used after stimulation



## Parameters

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

## Returns

Use idle value

### 11.41.3.11 GetDigInState() `uint16_t GetDigInState ( )`

Reads from the LIH30 digital input

## Returns

The bit mask defining the digital input state

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

Reads the requested amount of Eeprom byte from the EPC10 EEPROM

## Parameters

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

## Returns

pointer to internal memory for the requested amount of data

### 11.41.3.13 GetSampleInterval() `uint32_t GetSampleInterval ( )`

Gets the Sample Interval for the DACQ and Stimulation

## Returns

Sample Interval configured on the device

**11.41.3.14 IsUserTriggerEnabled()** `bool IsUserTriggerEnabled ( )`

Is the User Trigger enabled

**Returns**

Enabled

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

Gets the clipping information

**Parameters**

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

**Returns**

The clipping value

**11.41.3.16 ReadUARTData()** `String ^ ReadUARTData ( )`

Reads the config string from the device.

**Returns**

The config string.

**11.41.3.17 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   |

**11.41.3.18 SetAdcOffset()** `void SetAdcOffset (`  
    `LIH30_ADC_Channel_EnumNet AdcChannel,`  
    `int32_t Offset )`

Sets the ADC offset of the DACQ for a single channel

**Parameters**

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

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

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

**Parameters**

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

**11.41.3.20 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) |

**11.41.3.21 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  |

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

Sets the offset of a DAC channel.

**Parameters**

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

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

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

**Parameters**

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

**11.41.3.24 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  |

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

Writes to the LIH30 digital output

**Parameters**

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

**11.41.3.26 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                                               |

**11.41.3.27 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 |
|-----------------------|---------------------------------------------------------------------|

**11.41.3.28 WriteUARTData()** void WriteUARTData (   
String^ *commandString* )

Write the command string to the device.

#### Parameters

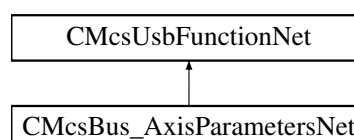
|                      |                    |
|----------------------|--------------------|
| <i>commandString</i> | The config string. |
|----------------------|--------------------|

### 11.41.4 Property Documentation

**11.41.4.1 StimulusFunction** CStimulusFunctionNet^ StimulusFunction [get]

## 11.42 CMcsBus\_AxisParametersNet Class Reference

Inheritance diagram for CMcsBus\_AxisParametersNet:



## Public Member Functions

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

## Additional Inherited Members

### 11.42.1 Constructor & Destructor Documentation

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

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

### 11.42.2 Member Function Documentation

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

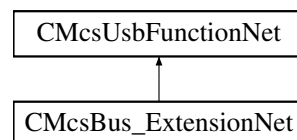
**11.42.2.2 [GetAxisParametersUnsignedEeprom\(\)](#)** unsigned int [GetAxisParametersUnsignedEeprom](#) (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis*,  
    unsigned short *index* )

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

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

## 11.43 CMcsBus\_ExtensionNet Class Reference

Inheritance diagram for CMcsBus\_ExtensionNet:



### Public Member Functions

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

### Additional Inherited Members

#### 11.43.1 Constructor & Destructor Documentation

**11.43.1.1 CMcsBus\_ExtensionNet()** `CMcsBus_ExtensionNet (`  
`CMcsUsbNet^ device )`

**11.43.1.2 ~CMcsBus\_ExtensionNet()** `~CMcsBus_ExtensionNet (`  
`void )`

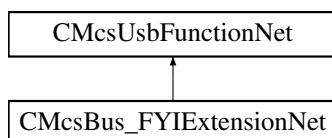
### 11.43.2 Member Function Documentation

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

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

## 11.44 CMcsBus\_FYIExtensionNet Class Reference

Inheritance diagram for CMcsBus\_FYIExtensionNet:



### Public Member Functions

- `CMcsBus_FYIExtensionNet` (`CMcsUsbNet`<sup>^</sup> device)
- `~CMcsBus_FYIExtensionNet` (void)
- void `SetValves` (unsigned char busnumber, unsigned char busaddress, unsigned int states)
- unsigned int `GetValves` (unsigned char busnumber, unsigned char busaddress)
- void `SetDIO` (unsigned char busnumber, unsigned char busaddress, unsigned short io)
- unsigned short `GetDIO` (unsigned char busnumber, unsigned char busaddress)
- void `SetSingleHeater` (unsigned char busnumber, unsigned char busaddress, short index, unsigned short power)
- unsigned short `GetSingleHeater` (unsigned char busnumber, unsigned char busaddress, short index)

### Additional Inherited Members

#### 11.44.1 Constructor & Destructor Documentation

**11.44.1.1 CMcsBus\_FYIExtensionNet()** `CMcsBus_FYIExtensionNet` (   
     `CMcsUsbNet`<sup>^</sup> *device* )



**11.44.1.2** `~CMcsBus_FYIExtensionNet()` `~CMcsBus_FYIExtensionNet` (  
    `void` )

## 11.44.2 Member Function Documentation

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

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

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

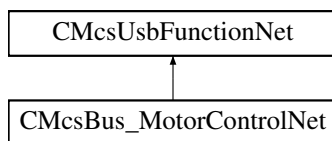
**11.44.2.4** `SetDIO()` `void SetDIO` (  
    `unsigned char busnumber`,  
    `unsigned char busaddress`,  
    `unsigned short io` )

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

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

## 11.45 CMcsBus\_MotorControlNet Class Reference

Inheritance diagram for CMcsBus\_MotorControlNet:



### Public Member Functions

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

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

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

## Additional Inherited Members

### 11.45.1 Constructor & Destructor Documentation

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

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

### 11.45.2 Member Function Documentation

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

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

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

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

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

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

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

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

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

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

**11.45.2.11 GetMCCurrentMode()** [RoboCurrentModeEnumNet](#) GetMCCurrentMode (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis* )

- 11.45.2.12 GetMCCurrentModeEeprom()** [RoboCurrentModeEnumNet](#) GetMCCurrentModeEeprom ( unsigned char *busnumber*, unsigned char *busaddress*, unsigned char *axis* )
- 11.45.2.13 GetMCCurrentModeShortCommand()** [RoboCurrentModeEnumNet](#) GetMCCurrentModeShortCommand ( unsigned char *busnumber*, unsigned char *busaddress*, unsigned char *axis* )
- 11.45.2.14 GetMCCurrentPosition()** int GetMCCurrentPosition ( unsigned char *busnumber*, unsigned char *busaddress*, unsigned char *axis* )
- 11.45.2.15 GetMCCurrentShortCommand()** short GetMCCurrentShortCommand ( unsigned char *busnumber*, unsigned char *busaddress*, unsigned char *axis* )
- 11.45.2.16 GetMCCurrentSpeed()** short GetMCCurrentSpeed ( unsigned char *busnumber*, unsigned char *busaddress*, unsigned char *axis* )
- 11.45.2.17 GetMCMaxAcceleration()** unsigned short GetMCMaxAcceleration ( unsigned char *busnumber*, unsigned char *busaddress*, unsigned char *axis* )
- 11.45.2.18 GetMCMaxAccelerationEeprom()** unsigned short GetMCMaxAccelerationEeprom ( unsigned char *busnumber*, unsigned char *busaddress*, unsigned char *axis* )

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

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

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

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

**11.45.2.23 GetMCMaxTravel()** int GetMCMaxTravel (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

**11.45.2.24 GetMCMaxTravelEeprom()** int GetMCMaxTravelEeprom (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )

**11.45.2.25 GetMCMaxTravelShortCommand()** int GetMCMaxTravelShortCommand (   
 unsigned char *busnumber*,  
 unsigned char *busaddress*,  
 unsigned char *axis* )



- 11.45.2.26 GetMCMovement()** unsigned short GetMCMovement (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )
- 11.45.2.27 GetMCNewPosition()** int GetMCNewPosition (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )
- 11.45.2.28 GetMCOutputOnOff()** bool GetMCOutputOnOff (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )
- 11.45.2.29 GetMCPhase()** unsigned short GetMCPhase (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )
- 11.45.2.30 GetMCPhaseOffset()** unsigned short GetMCPhaseOffset (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )
- 11.45.2.31 GetMCReference()** unsigned char GetMCReference (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis*,   
 [System::Runtime::InteropServices::Out] unsigned char% *switch\_port* )
- 11.45.2.32 GetMCReferenceCurrent()** short GetMCReferenceCurrent (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )

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

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

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

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

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

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

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

- 11.45.2.40 GetMCSpeedShortCommand()** short GetMCSpeedShortCommand (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )
- 11.45.2.41 GetMCSpeedUnitEeprom()** int32\_t GetMCSpeedUnitEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )
- 11.45.2.42 GetMCStandbyCurrent()** short GetMCStandbyCurrent (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )
- 11.45.2.43 GetMCStandbyCurrentEeprom()** short GetMCStandbyCurrentEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )
- 11.45.2.44 GetMCStandbyTime()** short GetMCStandbyTime (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )
- 11.45.2.45 GetMCStandbyTimeEeprom()** short GetMCStandbyTimeEeprom (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )
- 11.45.2.46 GetSubChannel()** unsigned short GetSubChannel (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 unsigned char *axis* )

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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



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

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

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

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

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

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

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

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

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

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

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

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

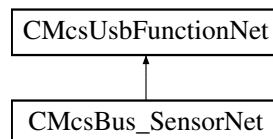
**11.45.2.89 SetSubChannel()** void SetSubChannel (   
     unsigned char *busnumber*,   
     unsigned char *busaddress*,   
     unsigned char *axis*,   
     unsigned short *subchannel* )

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

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

## 11.46 CMcsBus\_SensorNet Class Reference

Inheritance diagram for CMcsBus\_SensorNet:



### Public Member Functions

- [CMcsBus\\_SensorNet](#) ([CMcsUsbNet](#)<sup>^</sup> device)
- [~CMcsBus\\_SensorNet](#) (void)
- void [SetMinimalThreshold](#) (unsigned char busnumber, unsigned char busaddress, unsigned short threshold)
- unsigned short [GetMinimalThreshold](#) (unsigned char busnumber, unsigned char busaddress)
- void [SetDetectionThreshold](#) (unsigned char busnumber, unsigned char busaddress, unsigned short threshold)
- unsigned short [GetDetectionThreshold](#) (unsigned char busnumber, unsigned char busaddress)
- void [SetLatency](#) (unsigned char busnumber, unsigned char busaddress, unsigned short latency)
- unsigned short [GetLatency](#) (unsigned char busnumber, unsigned char busaddress)
- unsigned short [GetBubbleStatus](#) (unsigned char busnumber, unsigned char busaddress)
- unsigned short [GetLatencyCounter](#) (unsigned char busnumber, unsigned char busaddress)
- unsigned short [GetDetectorValue](#) (unsigned char busnumber, unsigned char busaddress)
- array< int > ^ [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

### 11.46.1 Constructor & Destructor Documentation

**11.46.1.1 CMcsBus\_SensorNet()** [CMcsBus\\_SensorNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> *device* )

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

### 11.46.2 Member Function Documentation

**11.46.2.1 CatchAmpGetAdcMean()** int [CatchAmpGetAdcMean](#) (  
unsigned char *busnumber*,  
unsigned char *busaddress* )

**11.46.2.2 CatchAmpGetAdcValue()** int [CatchAmpGetAdcValue](#) (  
unsigned char *busnumber*,  
unsigned char *busaddress* )

**11.46.2.3 CatchAmpGetAdcValueH()** int [CatchAmpGetAdcValueH](#) (  
unsigned char *busnumber*,  
unsigned char *busaddress* )

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

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

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

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

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

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

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

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

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

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

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

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

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

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

**11.46.2.18 Get4ADCMode()** [PatchServAdcModeEnumNet](#) Get4ADCMode (   
    unsigned char *busnumber*,  
    unsigned char *busaddress* )

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

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

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

**11.46.2.22 GetBubbleStatus()** `unsigned short GetBubbleStatus (`  
    `unsigned char busnumber,`  
    `unsigned char busaddress )`

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

**11.46.2.24 GetDetectionThreshold()** `unsigned short GetDetectionThreshold (`  
    `unsigned char busnumber,`  
    `unsigned char busaddress )`

**11.46.2.25 GetDetectorValue()** `unsigned short GetDetectorValue (`  
    `unsigned char busnumber,`  
    `unsigned char busaddress )`

**11.46.2.26 GetLatency()** `unsigned short GetLatency (`  
    `unsigned char busnumber,`  
    `unsigned char busaddress )`

**11.46.2.27 GetLatencyCounter()** `unsigned short GetLatencyCounter (`  
    `unsigned char busnumber,`  
    `unsigned char busaddress )`



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

**11.46.2.29 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* )

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

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

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

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

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

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

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

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

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

**11.46.2.39 GetRotatePump()** short GetRotatePump (   
    unsigned char *busnumber*,   
    unsigned char *busaddress*,   
    unsigned short *index* )

**11.46.2.40 GetSamplePeriode()** unsigned short GetSamplePeriode (   
    unsigned char *busnumber*,   
    unsigned char *busaddress* )

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

**11.46.2.42 GetSyncState()** unsigned short GetSyncState (   
 unsigned char *busnumber*,   
 unsigned char *busaddress* )

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

**11.46.2.58 SetSollPressure()** void SetSollPressure (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned short *index*,  
    int *pressure* )

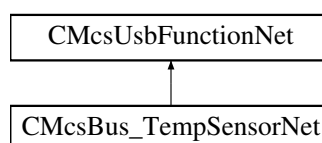
**11.46.2.59 StartSync()** void StartSync (  
    unsigned char *busnumber*,  
    unsigned char *busaddress* )

**11.46.2.60 TactSwitchGetState()** int TactSwitchGetState (  
    unsigned char *busnumber*,  
    unsigned char *busaddress* )

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

## 11.47 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

### 11.47.1 Constructor & Destructor Documentation

**11.47.1.1 [CMcsBus\\_TempSensorNet\(\)](#)** [CMcsBus\\_TempSensorNet](#) (  
    [CMcsUsbNet](#)<sup>^</sup> device )

**11.47.1.2 [~CMcsBus\\_TempSensorNet\(\)](#)** [~CMcsBus\\_TempSensorNet](#) (  
    void )

### 11.47.2 Member Function Documentation

**11.47.2.1 [GetNanoVoltsPerKelvin\(\)](#)** int [GetNanoVoltsPerKelvin](#) (  
    unsigned char *busnumber*,  
    unsigned char *busaddress* )

**11.47.2.2 [GetTemperatur\(\)](#) [1/2]** short [GetTemperatur](#) (  
    unsigned char *busnumber*,  
    unsigned char *busaddress* )

**11.47.2.3 [GetTemperatur\(\)](#) [2/2]** short [GetTemperatur](#) (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    short *index* )

**11.47.2.4 GetThermoOffset()** short GetThermoOffset (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 short *index* )

**11.47.2.5 GetThermoTemp()** short GetThermoTemp (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 short *index* )

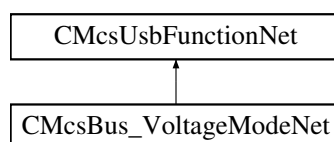
**11.47.2.6 GetThermoVoltage()** short GetThermoVoltage (   
 unsigned char *busnumber*,   
 unsigned char *busaddress*,   
 short *index* )

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

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

## 11.48 CMcsBus\_VoltageModeNet Class Reference

Inheritance diagram for CMcsBus\_VoltageModeNet:



## Public Member Functions

- [CMcsBus\\_VoltageModeNet](#) ([CMcsUsbNet](#)<sup>^</sup> device)
- [~CMcsBus\\_VoltageModeNet](#) (void)
- void [SetVMMMaxPositiveCurrentEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel, short current)
- short [GetVMMMaxPositiveCurrentEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel)
- void [SetVMMMaxPositiveCurrent](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel, short current)
- short [GetVMMMaxPositiveCurrent](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel)
- void [SetVMMMaxNegativeCurrentEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel, short current)
- short [GetVMMMaxNegativeCurrentEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel)
- void [SetVMMMaxNegativeCurrent](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel, short current)
- short [GetVMMMaxNegativeCurrent](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel)
- void [SetVMMMaxPositiveVoltageEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel, short voltage)
- short [GetVMMMaxPositiveVoltageEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel)
- void [SetVMMMaxPositiveVoltage](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel, short voltage)
- short [GetVMMMaxPositiveVoltage](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel)
- void [SetVMMMaxNegativeVoltageEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel, short voltage)
- short [GetVMMMaxNegativeVoltageEeprom](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel)
- void [SetVMMMaxNegativeVoltage](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel, short voltage)
- short [GetVMMMaxNegativeVoltage](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel)
- void [SetVMOOutputOnOff](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel, unsigned short status)
- unsigned short [GetVMOOutputOnOff](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel)
- void [SetVMVoltage](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel, short voltage)
- short [GetVMVoltage](#) (unsigned char busnumber, unsigned char busaddress, unsigned char channel)

## Additional Inherited Members

### 11.48.1 Constructor & Destructor Documentation

#### 11.48.1.1 [CMcsBus\\_VoltageModeNet\(\)](#) [CMcsBus\\_VoltageModeNet](#) ( [CMcsUsbNet](#)<sup>^</sup> device )



**11.48.1.2** `~CMcsBus_VoltageModeNet()` `~CMcsBus_VoltageModeNet` (  
    void )

## 11.48.2 Member Function Documentation

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

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

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

**11.48.2.4** `GetVMMaxNegativeVoltageEeprom()` short `GetVMMaxNegativeVoltageEeprom` (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *channel* )

**11.48.2.5** `GetVMMaxPositiveCurrent()` short `GetVMMaxPositiveCurrent` (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *channel* )

**11.48.2.6** `GetVMMaxPositiveCurrentEeprom()` short `GetVMMaxPositiveCurrentEeprom` (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *channel* )

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

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

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

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

**11.48.2.11 SetVMMaxNegativeCurrent()** void SetVMMaxNegativeCurrent (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *channel*,  
    short *current* )

**11.48.2.12 SetVMMaxNegativeCurrentEeprom()** void SetVMMaxNegativeCurrentEeprom (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *channel*,  
    short *current* )

**11.48.2.13 SetVMMaxNegativeVoltage()** void SetVMMaxNegativeVoltage (   
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *channel*,  
    short *voltage* )

**11.48.2.14 SetVMMaxNegativeVoltageEeprom()** void SetVMMaxNegativeVoltageEeprom (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *channel*,  
    short *voltage* )

**11.48.2.15 SetVMMaxPositiveCurrent()** void SetVMMaxPositiveCurrent (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *channel*,  
    short *current* )

**11.48.2.16 SetVMMaxPositiveCurrentEeprom()** void SetVMMaxPositiveCurrentEeprom (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *channel*,  
    short *current* )

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

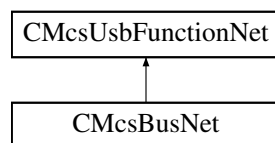
**11.48.2.18 SetVMMaxPositiveVoltageEeprom()** void SetVMMaxPositiveVoltageEeprom (  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *channel*,  
    short *voltage* )

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

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

## 11.49 CMcsBusNet Class Reference

Inheritance diagram for CMcsBusNet:



### Public Member Functions

- [CMcsBusNet](#) ([CMcsUsbNet](#)^ 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

#### 11.49.1 Constructor & Destructor Documentation

**11.49.1.1 CMcsBusNet()** [CMcsBusNet](#) (  
[CMcsUsbNet](#)^ device )

**11.49.1.2** `~CMcsBusNet()` virtual `~CMcsBusNet` (  
void ) [virtual]

## 11.49.2 Member Function Documentation

**11.49.2.1** `CMcsBusNet::GetMode()` unsigned short `CMcsBusNet::GetMode` (  
unsigned char *busnumber*,  
unsigned char *busaddress* )

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

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

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

**11.49.2.5** `GetBusAddress()` unsigned short `GetBusAddress` (  
unsigned char *busnumber*,  
unsigned char *busaddress* )

**11.49.2.6** `GetBusAddressEeprom()` unsigned short `GetBusAddressEeprom` (  
unsigned char *busnumber*,  
unsigned char *busaddress* )

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

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

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

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

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

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

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

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

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

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

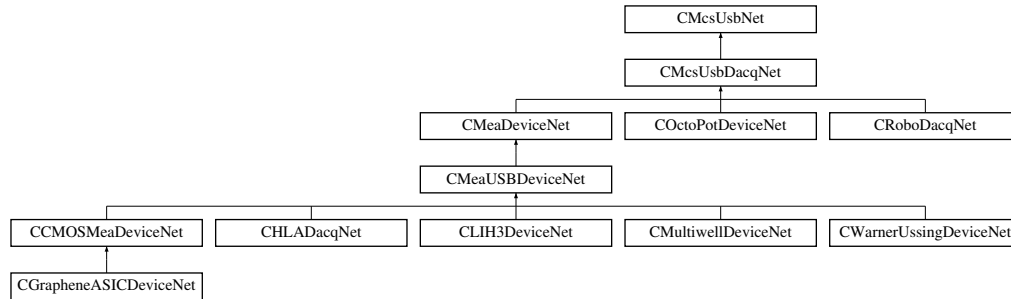
**11.49.2.17 SetCommand() [4/4]** void SetCommand (   
    unsigned char *command*,  
    unsigned char *busnumber*,  
    unsigned char *busaddress*,  
    unsigned char *axis*,  
    unsigned short *value* )

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

## 11.50 CMcsUsbDacqNet Class Reference

Base class for data acquisition devices.

Inheritance diagram for CMcsUsbDacqNet:



### Classes

- class [CHWInfo](#)  
*Class to provide hardware information about the device.*

### Public Member Functions

- [CMcsUsbDacqNet](#) ()
- [~CMcsUsbDacqNet](#) ()
- [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 void SetVoltageRangeInMicroVolt](#) (int32\_t voltageRange, unsigned int virtualDevice)  
*Sets the voltage range on devices which support multiple voltage ranges.*
- [virtual int32\\_t GetVoltageRangeInMicroVolt](#) (unsigned int virtualDevice)  
*Gets the currently selected voltage range on devices which support multiple voltage ranges.*
- [virtual int32\\_t GetVoltageRangeInMilliVolt](#) ()  
*Gets the currently selected voltage range on devices which support multiple voltage ranges.*
- [virtual void SetDataMode](#) (DataModeEnumNet dataMode, unsigned int virtualDevice)  
*Sets the data mode, can be 16, 24 or 32bit, all signed or unsigned on the MEA2100 device.*
- [virtual DataModeEnumNet GetDataMode](#) (unsigned int virtualDevice)  
*Gets the data mode, can be 16, 24 or 32bit, all signed or unsigned on the MEA2100 device.*
- [void SetDigitalSource](#) (DigitalTargetEnumNet digitaltarget, int32\_t NrChannel, DigitalSourceEnumNet source, int bitnumber\_offset)  
*Sets the function/source of an digital output bit.*
- [void SetDigitalSource](#) (DigitalTargetEnumNet digitaltarget, int32\_t NrChannel, W2100DigitalSourceEnumNet source, int bitnumber\_offset)  
*Sets the function/source of an digital output bit.*
- [void SetDigitalSource](#) (DigitalTargetEnumNet digitaltarget, int32\_t NrChannel, SCUDigitalSourceEnumNet source, int bitnumber\_offset)  
*Sets the function/source of an digital output bit.*
- [void SetDigitalSource](#) (DigitalTargetEnumNet digitaltarget, int32\_t NrChannel, MEA2100\_256DigitalSourceEnumNet source, int bitnumber\_offset)



*Sets the function/source of an digital output bit.*

- template<typename digitalsourceenum >  
void **SetDigitalSource** (DigitalTargetEnumNet digitaltarget, int32\_t NrChannel, DigitalSource< digitalsourceenum >^ source, int bitnumber\_offset)

*Sets the function/source of an digital output bit.*

- void **GetDigitalSource** (DigitalTargetEnumNet digitaltarget, int32\_t NrChannel, [System::Runtime::InteropServices::Out]DigitalSourceEnumNet% source, [System::Runtime::InteropServices::Out]int% bitnumber\_offset)

*Gets the function/source of an digital output bit.*

- void **GetDigitalSource** (DigitalTargetEnumNet digitaltarget, int32\_t NrChannel, [System::Runtime::InteropServices::Out]W2100DigitalSourceEnumNet% source, [System::Runtime::InteropServices::Out]int% bitnumber\_offset)

*Gets the function/source of an digital output bit.*

- void **GetDigitalSource** (DigitalTargetEnumNet digitaltarget, int32\_t NrChannel, [System::Runtime::InteropServices::Out]SCUDigitalSourceEnumNet% source, [System::Runtime::InteropServices::Out]int% bitnumber\_offset)

*Gets the function/source of an digital output bit.*

- void **GetDigitalSource** (DigitalTargetEnumNet digitaltarget, int32\_t NrChannel, [System::Runtime::InteropServices::Out]MEA2100\_256DigitalSourceEnumNet% source, [System::Runtime::InteropServices::Out]int% bitnumber\_offset)

*Gets the function/source of an digital output bit.*

- template<typename digitalsourceenum >  
void **GetDigitalSource** (DigitalTargetEnumNet digitaltarget, int32\_t NrChannel, [System::Runtime::InteropServices::Out]DigitalSource< digitalsourceenum >^% source, [System::Runtime::InteropServices::Out]int% bitnumber\_offset)

*Gets the function/source of an digital output bit.*

- virtual AdapterTypeEnumNet **GetAdapterType** ()

*Gets the adapter which is connected to the MEA2100 device.*

- virtual MeaLayoutEnumNet **GetMeaLayout** ()

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

- virtual uint32\_t **GetAdcDataFormat** (uint32\_t virtualDevice)

*Gets the ADC data format, 16 means 16 bits, 24 means 24 bits, 32 means 32 bits.*

- virtual uint32\_t **GetAnalogValueUnit** (uint32\_t virtualDevice, DacqGroupChannelEnumNet group, [System::Runtime::InteropServices::Out] AnalogUnitEnumNet% unit)
- virtual uint32\_t **GetResolutionPerDigit** (uint32\_t virtualDevice, DacqGroupChannelEnumNet group, [System::Runtime::InteropServices::Out] int% res, [System::Runtime::InteropServices::Out] int% resUnit)
- virtual uint32\_t **GetAdcZero** (uint32\_t virtualDevice, DacqGroupChannelEnumNet group, [System::Runtime::InteropServices::Out] int% adcz)
- virtual uint32\_t **GetHardwareMinRange** (uint32\_t virtualDevice, DacqGroupChannelEnumNet group, [System::Runtime::InteropServices::Out] int% r, [System::Runtime::InteropServices::Out] int% rUnit)
- virtual uint32\_t **GetHardwareMaxRange** (uint32\_t virtualDevice, DacqGroupChannelEnumNet group, [System::Runtime::InteropServices::Out] int% r, [System::Runtime::InteropServices::Out] int% rUnit)
- virtual uint32\_t **GetDataFormat** (uint32\_t virtualDevice, DacqGroupChannelEnumNet group, [System::Runtime::InteropServices::Out] int% numberOfBits)
- virtual uint32\_t **GetNumberOfDataBits** (uint32\_t virtualDevice, DacqGroupChannelEnumNet group, [System::Runtime::InteropServices::Out] int% numberOfBits)

*Get the real number of data bits.*

- virtual void **SetSamplerate** (int32\_t rate, unsigned int oversample, unsigned int virtualDevice)

*Sets the sampling frequency of the device.*

- virtual int32\_t **GetSamplerate** (unsigned int virtualDevice)

*Gets the sampling frequency of the device.*

- virtual uint32\_t **GetMaxSamplingFrequency** (int virtualDevice)

*Gets the maximal sampling frequency of the device.*

- virtual uint32\_t **GetMinSamplingFrequencyStepsize** ()

*Gets the minimal sampling frequency step size increment value of the device.*

- virtual int32\_t [GetChannelsInBlock](#) (unsigned int virtualDevice)

*Get the number of 16 bit datawords which will be collected per sample frame, use after the device is configured.*

- virtual 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](#) ()
- virtual void [SetSelectedChannels](#) (int nChannels, int queuesize, int threshold, [SampleSizeNet](#) samplesize, int ChannelsInBlock)

*Create a FIFO queue per channel. Each channel will have its own FIFO and Callback function.*

- virtual void [SetSelectedChannels](#) (int nChannels, int queuesize, int threshold, [SampleSizeNet](#) samplesize, [SampleDstSizeNet](#) sampleDstSize, int ChannelsInBlock)
- virtual void [SetSelectedChannels](#) (array< bool > ^ selectedChannels, int queuesize, int threshold, [SampleSizeNet](#) samplesize, int ChannelsInBlock)

*Create a FIFO queue per channel. Each channel will have its own FIFO and Callback function.*

- virtual void [SetSelectedChannels](#) (array< bool > ^ selectedChannels, int queuesize, int threshold, [SampleSizeNet](#) samplesize, [SampleDstSizeNet](#) sampleDstSize, int ChannelsInBlock)
- virtual void [SetSelectedData](#) (int nChannels, int queuesize, int threshold, [SampleSizeNet](#) samplesize, int ChannelsInBlock)

*Create a common FIFO queue for all channels. Use handle = 0 in the ChannelBlock\_ReadFrames... functions.*

- virtual void [SetSelectedData](#) (int nChannels, int queuesize, int threshold, [SampleSizeNet](#) samplesize, [SampleDstSizeNet](#) sampleDstSize, int ChannelsInBlock)
- virtual void [SetSelectedData](#) (array< bool > ^ selectedChannels, int queuesize, int threshold, [SampleSizeNet](#) samplesize, int ChannelsInBlock)

*Create a common FIFO queue for all channels. Use handle = 0 in the ChannelBlock\_ReadFrames... functions.*

- virtual void [SetSelectedData](#) (array< bool > ^ selectedChannels, int queuesize, int threshold, [SampleSizeNet](#) samplesize, [SampleDstSizeNet](#) sampleDstSize, int ChannelsInBlock)
- virtual int [AddSelectedChannelsQueue](#) (int nByteOffset, int nChannelOffset, int nChannels, int queuesize, int threshold, [SampleSizeNet](#) samplesize)

*Adds a common FIFO queue for all channels. Data in callback will be a list per channel. Use ChannelBlock\_ReadFramesDict... with handle = 0 to read the data.*

- virtual int [AddSelectedChannelsQueue](#) (int nByteOffset, int nChannelOffset, int nChannels, int queuesize, int threshold, [SampleSizeNet](#) samplesize, [SampleDstSizeNet](#) sampleDstSize)
- virtual int [AddSelectedChannelsQueue](#) (int nByteOffset, int nChannelOffset, array< bool > ^ selectedChannels, int queuesize, int threshold, [SampleSizeNet](#) samplesize)

*Adds a common FIFO queue for all channels. Data in callback will be a list per channel. Use ChannelBlock\_ReadFramesDict... with handle = 0 to read the data.*

- virtual int [AddSelectedChannelsQueue](#) (int nByteOffset, int nChannelOffset, array< bool > ^ selectedChannels, int queuesize, int threshold, [SampleSizeNet](#) samplesize, [SampleDstSizeNet](#) sampleDstSize)
- virtual void [SetSelectedChannelsQueue](#) (int nChannels, int queuesize, int threshold, [SampleSizeNet](#) samplesize, int ChannelsInBlock)

*Create a common FIFO queue for all channels. Data in callback will be a list per channel. Use ChannelBlock\_ReadFramesDict... with handle = 0 to read the data.*

- virtual void [SetSelectedChannelsQueue](#) (int nChannels, int queuesize, int threshold, [SampleSizeNet](#) samplesize, [SampleDstSizeNet](#) sampleDstSize, int ChannelsInBlock)
- virtual void [SetSelectedChannelsQueue](#) (array< bool > ^ selectedChannels, int queuesize, int threshold, [SampleSizeNet](#) samplesize, int ChannelsInBlock)

*Create a common FIFO queue for all channels. Data in callback will be a list per channel. Use ChannelBlock\_ReadFramesDict... with handle = 0 to read the data.*

- virtual void [SetSelectedChannelsQueue](#) (array< bool > ^ selectedChannels, int queuesize, int threshold, [SampleSizeNet](#) samplesize, [SampleDstSizeNet](#) sampleDstSize, int ChannelsInBlock)
- virtual uint32\_t [ChannelBlock\\_AvailFrames](#) (int handle)

*Get the number of sample frames already available in the FIFO.*

- virtual uint32\_t [ChannelBlock\\_AvailFrames](#) (int handle, int queue)

- virtual array< uint16\_t > ^ [ChannelBlock\\_ReadFramesUI16](#) (int handle, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)  
*Read data from a FIFO queue in uint16\_t data format*
- virtual void [ChannelBlock\\_ReadFramesUI16](#) (int handle, array< uint16\_t >^ buffer, int frames\_pos, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)  
*Read data from a FIFO queue in uint16\_t data format*
- virtual array< int16\_t > ^ [ChannelBlock\\_ReadFramesI16](#) (int handle, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)  
*Read data from a FIFO queue in int16\_t data format*
- virtual void [ChannelBlock\\_ReadFramesI16](#) (int handle, array< int16\_t >^ buffer, int frames\_pos, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)  
*Read data from a FIFO queue in int16\_t data format*
- virtual array< uint32\_t > ^ [ChannelBlock\\_ReadFramesUI32](#) (int handle, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)  
*Read data from a FIFO queue in uint32\_t data format*
- virtual void [ChannelBlock\\_ReadFramesUI32](#) (int handle, array< uint32\_t >^ buffer, int frames\_pos, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)  
*Read data from a FIFO queue in uint32\_t data format*
- virtual array< int32\_t > ^ [ChannelBlock\\_ReadFramesI32](#) (int handle, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)  
*Read data from a FIFO queue in int32\_t data format*
- virtual void [ChannelBlock\\_ReadFramesI32](#) (int handle, array< int32\_t >^ buffer, int frames\_pos, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)  
*Read data from a FIFO queue in int32\_t data format*
- virtual array< array< uint16\_t >^> ^ [ChannelBlock\\_ReadAsFrameArrayUI16](#) (int handle, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)  
*Read data from a FIFO queue as array of uint16\_t data frame arrays*
- virtual array< array< uint16\_t >^> ^ [ChannelBlock\\_ReadAsFrameArrayUI16](#) (int handle, int queue, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)  
*Read data from a FIFO queue as array of uint16\_t data frame arrays*
- virtual array< array< int16\_t >^> ^ [ChannelBlock\\_ReadAsFrameArrayI16](#) (int handle, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)  
*Read data from a FIFO queue as array of uint16\_t data frame arrays*
- virtual array< array< int16\_t >^> ^ [ChannelBlock\\_ReadAsFrameArrayI16](#) (int handle, int queue, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)  
*Read data from a FIFO queue as array of uint16\_t data frame arrays*
- virtual array< array< uint32\_t >^> ^ [ChannelBlock\\_ReadAsFrameArrayUI32](#) (int handle, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)  
*Read data from a FIFO queue as array of uint16\_t data frame arrays*
- virtual array< array< uint32\_t >^> ^ [ChannelBlock\\_ReadAsFrameArrayUI32](#) (int handle, int queue, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)  
*Read data from a FIFO queue as array of uint16\_t data frame arrays*
- virtual array< array< int32\_t >^> ^ [ChannelBlock\\_ReadAsFrameArrayI32](#) (int handle, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)  
*Read data from a FIFO queue as array of uint16\_t data frame arrays*
- virtual array< array< int32\_t >^> ^ [ChannelBlock\\_ReadAsFrameArrayI32](#) (int handle, int queue, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)  
*Read data from a FIFO queue as array of uint16\_t data frame arrays*
- virtual System::Collections::Generic::Dictionary< int, array< uint16\_t >^> ^ [ChannelBlock\\_ReadFramesDictUI16](#) (int handle, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)  
*Read data from a FIFO queue in uint16\_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number*

- virtual System::Collections::Generic::Dictionary< int, array< int16\_t >^> ^ [ChannelBlock\\_ReadFramesDictI16](#) (int handle, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)  
*Read data from a FIFO queue in int16\_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number*
- virtual System::Collections::Generic::Dictionary< int, array< uint32\_t >^> ^ [ChannelBlock\\_ReadFramesDictUI32](#) (int handle, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)  
*Read data from a FIFO queue in uint32\_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number*
- virtual System::Collections::Generic::Dictionary< int, array< int32\_t >^> ^ [ChannelBlock\\_ReadFramesDictI32](#) (int handle, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)  
*Read data from a FIFO queue in int32\_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number*
- virtual System::Collections::Generic::Dictionary< int, array< uint16\_t >^> ^ [GetGroupChannelDataUI16](#) ([DacqGroupChannelEnumNet](#) group, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)  
*Read data from a FIFO queue in uint16\_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number*
- virtual System::Collections::Generic::Dictionary< int, array< int16\_t >^> ^ [GetGroupChannelDataI16](#) ([DacqGroupChannelEnumNet](#) group, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)  
*Read data from a FIFO queue in int16\_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number*
- virtual System::Collections::Generic::Dictionary< int, array< uint32\_t >^> ^ [GetGroupChannelDataUI32](#) ([DacqGroupChannelEnumNet](#) group, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)  
*Read data from a FIFO queue in uint32\_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number*
- virtual System::Collections::Generic::Dictionary< int, array< int32\_t >^> ^ [GetGroupChannelDataI32](#) ([DacqGroupChannelEnumNet](#) group, int frames, [System::Runtime::InteropServices::Out]int % frames\_ret)  
*Read data from a FIFO queue in int32\_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number*
- void [SetupGroupDacqQueue](#) (int queuesize, int threshold)
- void [SetupGroupDacqQueue](#) (int queuesize, int threshold, unsigned int virtualDevice)
- [CHWInfo](#) ^ [HWInfo](#) ()

### Static Public Attributes

- static const int [Error\\_Callback\\_Queue\\_Full](#) = 0x100
- static const int [Error\\_Callback\\_Aquisition\\_Stopped](#) = 0x200
- static const int [Error\\_Callback\\_Packet\\_Error](#) = 1
- static const int [Error\\_Callback\\_RingQueue\\_Full](#) = 3
- static const int [Error\\_Callback\\_Frames\\_Lost](#) = 4
- static const int [Error\\_Callback\\_Data\\_lost](#) = 5

### Properties

- virtual int [Samplerate](#) [get, set]  
*The sampling frequency of the device in Hz.*

### Events

- [OnChannelData](#) ^ [ChannelDataEvent](#) [add, remove, raise]
- [OnError](#) ^ [ErrorEvent](#) [add, remove, raise]

## Additional Inherited Members

### 11.50.1 Detailed Description

Base class for data acquisition devices.

### 11.50.2 Constructor & Destructor Documentation

**11.50.2.1 CMcsUsbDacqNet()** `CMcsUsbDacqNet ( )`

**11.50.2.2 ~CMcsUsbDacqNet()** `~CMcsUsbDacqNet ( )`

### 11.50.3 Member Function Documentation

**11.50.3.1 AddSelectedChannelsQueue()** [1/4] `virtual int AddSelectedChannelsQueue (`  
    `int nByteOffset,`  
    `int nChannelOffset,`  
    `array< bool >^ selectedChannels,`  
    `int queuesize,`  
    `int threshold,`  
    `SampleSizeNet samplesize ) [virtual]`

Adds a common FIFO queue for all channels. Data in callback will be a list per channel. Use ChannelBlock\_Read↔FramesDict... with handle = 0 to read the data.

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

#### Parameters

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

#### Parameters

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

## Parameters

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

## Parameters

|                  |                                          |
|------------------|------------------------------------------|
| <i>queuesize</i> | Size of sample frames the FIFO can hold. |
|------------------|------------------------------------------|

## Parameters

|                  |                                                                                       |
|------------------|---------------------------------------------------------------------------------------|
| <i>threshold</i> | Number of sample frames the FIFO must acquire before the callback function is called. |
|------------------|---------------------------------------------------------------------------------------|

## Parameters

|                   |                                            |
|-------------------|--------------------------------------------|
| <i>samplesize</i> | size of the datawords, either 16 or 32bit. |
|-------------------|--------------------------------------------|

## Returns

The handle to the Queue.

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

**11.50.3.3 AddSelectedChannelsQueue() [3/4]** virtual int AddSelectedChannelsQueue (   
     int *nByteOffset*,   
     int *nChannelOffset*,   
     int *nChannels*,   
     int *queuesize*,   
     int *threshold*,   
     SampleSizeNet *samplesize* ) [virtual]

Adds a common FIFO queue for all channels. Data in callback will be a list per channel. Use `ChannelBlock_Read↔FramesDict...` with `handle = 0` to read the data.

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

#### Parameters

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

#### Parameters

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

#### Parameters

|                  |                                                 |
|------------------|-------------------------------------------------|
| <i>nChannels</i> | Number of channels to be collected in the FIFO. |
|------------------|-------------------------------------------------|

#### Parameters

|                  |                                          |
|------------------|------------------------------------------|
| <i>queuesize</i> | Size of sample frames the FIFO can hold. |
|------------------|------------------------------------------|

#### Parameters

|                  |                                                                                       |
|------------------|---------------------------------------------------------------------------------------|
| <i>threshold</i> | Number of sample frames the FIFO must acquire before the callback function is called. |
|------------------|---------------------------------------------------------------------------------------|

#### Parameters

|                   |                                            |
|-------------------|--------------------------------------------|
| <i>samplesize</i> | size of the datawords, either 16 or 32bit. |
|-------------------|--------------------------------------------|



**Returns**

The handle to the Queue.

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

**11.50.3.5 ChannelBlock\_AvailFrames()** [1/2] virtual uint32\_t ChannelBlock\_AvailFrames (   
     int *handle* ) [virtual]

Get the number of sample frames already available in the FIFO.

**Parameters**

|               |                                                                                                     |
|---------------|-----------------------------------------------------------------------------------------------------|
| <i>handle</i> | Handle of the FIFO queue. Either zero when the SetSelectedData call was used or the channel number. |
|---------------|-----------------------------------------------------------------------------------------------------|

**Returns**

Number of sample frames available in the FIFO.

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

**11.50.3.7 ChannelBlock\_ReadAsFrameArrayI16()** [1/2] virtual array<array<int16\_t>> ^ Channel↔  
 Block\_ReadAsFrameArrayI16 (   
     int *handle*,   
     int *frames*,   
     [System::Runtime::InteropServices::Out] int % *frames\_ret* ) [virtual]

Read data from a FIFO queue as array of uint16\_t data frame arrays

**Parameters**

|               |                                                                        |
|---------------|------------------------------------------------------------------------|
| <i>handle</i> | Handle of the FIFO queue. Zero when the SetSelectedData call was used. |
|---------------|------------------------------------------------------------------------|

**Parameters**

|               |                                  |
|---------------|----------------------------------|
| <i>frames</i> | Number of sample frames to read. |
|---------------|----------------------------------|

**Parameters**

|                   |                                                                        |
|-------------------|------------------------------------------------------------------------|
| <i>frames_ret</i> | Number of sample frames which were read, might be smaller than frames. |
|-------------------|------------------------------------------------------------------------|

**Returns**

Array of int16\_t frame arrays.

```
11.50.3.8 ChannelBlock_ReadAsFrameArrayI16() [2/2] virtual array<array<int16_t>^> ^ Channel↔
Block_ReadAsFrameArrayI16 (
 int handle,
 int queue,
 int frames,
 [System::Runtime::InteropServices::Out] int % frames_ret) [virtual]
```

Read data from a FIFO queue as array of uint16\_t data frame arrays

**Parameters**

|               |                                                                        |
|---------------|------------------------------------------------------------------------|
| <i>handle</i> | Handle of the FIFO queue. Zero when the SetSelectedData call was used. |
|---------------|------------------------------------------------------------------------|

**Parameters**

|               |                                  |
|---------------|----------------------------------|
| <i>queue</i>  | Number of the sub queue.         |
| <i>frames</i> | Number of sample frames to read. |

## Parameters

|                   |                                                                        |
|-------------------|------------------------------------------------------------------------|
| <i>frames_ret</i> | Number of sample frames which were read, might be smaller than frames. |
|-------------------|------------------------------------------------------------------------|

## Returns

Array of int16\_t frame arrays.

**11.50.3.9 ChannelBlock\_ReadAsFrameArrayI32()** [1/2] `virtual array<array<int32_t>> ^ Channel↔  
Block_ReadAsFrameArrayI32 (`  
`int handle,`  
`int frames,`  
`[System::Runtime::InteropServices::Out] int % frames_ret ) [virtual]`

Read data from a FIFO queue as array of uint16\_t data frame arrays

## Parameters

|               |                                                                        |
|---------------|------------------------------------------------------------------------|
| <i>handle</i> | Handle of the FIFO queue. Zero when the SetSelectedData call was used. |
|---------------|------------------------------------------------------------------------|

## Parameters

|               |                                  |
|---------------|----------------------------------|
| <i>frames</i> | Number of sample frames to read. |
|---------------|----------------------------------|

## Parameters

|                   |                                                                        |
|-------------------|------------------------------------------------------------------------|
| <i>frames_ret</i> | Number of sample frames which were read, might be smaller than frames. |
|-------------------|------------------------------------------------------------------------|

## Returns

Array of int32\_t frame arrays.

**11.50.3.10 ChannelBlock\_ReadAsFrameArrayI32()** [2/2] `virtual array<array<int32_t>^> ^ ChannelBlock_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.

**11.50.3.11 ChannelBlock\_ReadAsFrameArrayUI16()** [1/2] `virtual array<array<uint16_t>^> ^ ChannelBlock_ReadAsFrameArrayUI16 (`  
`int handle,`  
`int frames,`  
`[System::Runtime::InteropServices::Out] int % frames_ret ) [virtual]`

Read data from a FIFO queue as array of uint16\_t data frame arrays

#### Parameters

|               |                                                                        |
|---------------|------------------------------------------------------------------------|
| <i>handle</i> | Handle of the FIFO queue. Zero when the SetSelectedData call was used. |
|---------------|------------------------------------------------------------------------|

#### Parameters

|               |                                  |
|---------------|----------------------------------|
| <i>frames</i> | Number of sample frames to read. |
|---------------|----------------------------------|

## Parameters

|                   |                                                                        |
|-------------------|------------------------------------------------------------------------|
| <i>frames_ret</i> | Number of sample frames which were read, might be smaller than frames. |
|-------------------|------------------------------------------------------------------------|

## Returns

Array of uint16\_t frame arrays.

### 11.50.3.12 ChannelBlock\_ReadAsFrameArrayUI16() [2/2] virtual array<array<uint16\_t>^> ^

```
ChannelBlock_ReadAsFrameArrayUI16 (
 int handle,
 int queue,
 int frames,
 [System::Runtime::InteropServices::Out] int % frames_ret) [virtual]
```

Read data from a FIFO queue as array of uint16\_t data frame arrays

## Parameters

|               |                                                                        |
|---------------|------------------------------------------------------------------------|
| <i>handle</i> | Handle of the FIFO queue. Zero when the SetSelectedData call was used. |
|---------------|------------------------------------------------------------------------|

## Parameters

|               |                                  |
|---------------|----------------------------------|
| <i>queue</i>  | Number of the sub queue.         |
| <i>frames</i> | Number of sample frames to read. |

## Parameters

|                   |                                                                        |
|-------------------|------------------------------------------------------------------------|
| <i>frames_ret</i> | Number of sample frames which were read, might be smaller than frames. |
|-------------------|------------------------------------------------------------------------|

## Returns

Array of uint16\_t frame arrays.

**11.50.3.13 ChannelBlock\_ReadAsFrameArrayUI32()** [1/2] `virtual array<array<uint32_t>> ^`  
`ChannelBlock_ReadAsFrameArrayUI32 (`  
    `int handle,`  
    `int frames,`  
    `[System::Runtime::InteropServices::Out] int % frames_ret ) [virtual]`

Read data from a FIFO queue as array of uint16\_t data frame arrays

#### Parameters

|               |                                                                        |
|---------------|------------------------------------------------------------------------|
| <i>handle</i> | Handle of the FIFO queue. Zero when the SetSelectedData call was used. |
|---------------|------------------------------------------------------------------------|

#### Parameters

|               |                                  |
|---------------|----------------------------------|
| <i>frames</i> | Number of sample frames to read. |
|---------------|----------------------------------|

#### Parameters

|                   |                                                                        |
|-------------------|------------------------------------------------------------------------|
| <i>frames_ret</i> | Number of sample frames which were read, might be smaller than frames. |
|-------------------|------------------------------------------------------------------------|

#### Returns

Array of uint32\_t frame arrays.

**11.50.3.14 ChannelBlock\_ReadAsFrameArrayUI32()** [2/2] `virtual array<array<uint32_t>> ^`  
`ChannelBlock_ReadAsFrameArrayUI32 (`  
    `int handle,`  
    `int queue,`  
    `int frames,`  
    `[System::Runtime::InteropServices::Out] int % frames_ret ) [virtual]`

Read data from a FIFO queue as array of uint16\_t data frame arrays

#### Parameters

|               |                                                                        |
|---------------|------------------------------------------------------------------------|
| <i>handle</i> | Handle of the FIFO queue. Zero when the SetSelectedData call was used. |
|---------------|------------------------------------------------------------------------|

## Parameters

|               |                                  |
|---------------|----------------------------------|
| <i>queue</i>  | Number of the sub queue.         |
| <i>frames</i> | Number of sample frames to read. |

## Parameters

|                   |                                                                        |
|-------------------|------------------------------------------------------------------------|
| <i>frames_ret</i> | Number of sample frames which were read, might be smaller than frames. |
|-------------------|------------------------------------------------------------------------|

## Returns

Array of uint32\_t frame arrays.

**11.50.3.15 ChannelBlock\_ReadFramesDictI16()** virtual System::Collections::Generic::Dictionary<int, array<int16\_t>^> ^ ChannelBlock\_ReadFramesDictI16 ( int *handle*, int *frames*, [System::Runtime::InteropServices::Out] int % *frames\_ret* ) [virtual]

Read data from a FIFO queue in int16\_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number

## Parameters

|               |                                                                                 |
|---------------|---------------------------------------------------------------------------------|
| <i>handle</i> | Handle of the FIFO queue. Zero when the SetSelectedChannelsQueue call was used. |
|---------------|---------------------------------------------------------------------------------|

## Parameters

|                   |                                                                        |
|-------------------|------------------------------------------------------------------------|
| <i>frames</i>     | Number of sample frames to read.                                       |
| <i>frames_ret</i> | Number of sample frames which were read, might be smaller than frames. |

## Returns

Dictionary of int16\_t arrays and hardware channel as key.

```
11.50.3.16 ChannelBlock_ReadFramesDictI32() virtual System::Collections::Generic::Dictionary<int,
array<int32_t>^> ^ ChannelBlock_ReadFramesDictI32 (
 int handle,
 int frames,
 [System::Runtime::InteropServices::Out] int % frames_ret) [virtual]
```

Read data from a FIFO queue in int32\_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number

#### Parameters

|               |                                                                                 |
|---------------|---------------------------------------------------------------------------------|
| <i>handle</i> | Handle of the FIFO queue. Zero when the SetSelectedChannelsQueue call was used. |
|---------------|---------------------------------------------------------------------------------|

#### Parameters

|                   |                                                                        |
|-------------------|------------------------------------------------------------------------|
| <i>frames</i>     | Number of sample frames to read.                                       |
| <i>frames_ret</i> | Number of sample frames which were read, might be smaller than frames. |

#### Returns

Dictionary of int32\_t arrays and hardware channel as key.

```
11.50.3.17 ChannelBlock_ReadFramesDictUI16() virtual System::Collections::Generic::Dictionary<int,
array<uint16_t>^> ^ ChannelBlock_ReadFramesDictUI16 (
 int handle,
 int frames,
 [System::Runtime::InteropServices::Out] int % frames_ret) [virtual]
```

Read data from a FIFO queue in uint16\_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number

#### Parameters

|               |                                                                                 |
|---------------|---------------------------------------------------------------------------------|
| <i>handle</i> | Handle of the FIFO queue. Zero when the SetSelectedChannelsQueue call was used. |
|---------------|---------------------------------------------------------------------------------|

#### Parameters

|                   |                                                                        |
|-------------------|------------------------------------------------------------------------|
| <i>frames</i>     | Number of sample frames to read.                                       |
| <i>frames_ret</i> | Number of sample frames which were read, might be smaller than frames. |



**Returns**

Dictionary of uint16\_t arrays and hardware channel as key.

**11.50.3.18 ChannelBlock\_ReadFramesDictUI32()** `virtual System::Collections::Generic::Dictionary<int, array<uint32_t>^> ^ ChannelBlock_ReadFramesDictUI32 ( int handle, int frames, [System::Runtime::InteropServices::Out] int % frames_ret ) [virtual]`

Read data from a FIFO queue in uint32\_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number

**Parameters**

|               |                                                                                 |
|---------------|---------------------------------------------------------------------------------|
| <i>handle</i> | Handle of the FIFO queue. Zero when the SetSelectedChannelsQueue call was used. |
|---------------|---------------------------------------------------------------------------------|

**Parameters**

|                   |                                                                        |
|-------------------|------------------------------------------------------------------------|
| <i>frames</i>     | Number of sample frames to read.                                       |
| <i>frames_ret</i> | Number of sample frames which were read, might be smaller than frames. |

**Returns**

Dictionary of uint32\_t arrays and hardware channel as key.

**11.50.3.19 ChannelBlock\_ReadFramesI16()** `[1/2] virtual void ChannelBlock_ReadFramesI16 ( int handle, array< int16_t >^ buffer, int frames_pos, int frames, [System::Runtime::InteropServices::Out] int % frames_ret ) [virtual]`

Read data from a FIFO queue in int16\_t data format

**Parameters**

|               |                                                                                                     |
|---------------|-----------------------------------------------------------------------------------------------------|
| <i>handle</i> | Handle of the FIFO queue. Either zero when the SetSelectedData call was used or the channel number. |
|---------------|-----------------------------------------------------------------------------------------------------|

**Parameters**

|                   |                                            |
|-------------------|--------------------------------------------|
| <i>buffer</i>     | Buffer to put the data from the device in. |
| <i>frames_pos</i> | Position in buffer where to put the data.  |
| <i>frames</i>     | Number of sample frames to read.           |

**Parameters**

|                   |                                                                        |
|-------------------|------------------------------------------------------------------------|
| <i>frames_ret</i> | Number of sample frames which were read, might be smaller than frames. |
|-------------------|------------------------------------------------------------------------|

**11.50.3.20 ChannelBlock\_ReadFramesI16()** [2/2] `virtual array<int16_t> ^ ChannelBlock_ReadFramesI16 (`  
`int handle,`  
`int frames,`  
`[System::Runtime::InteropServices::Out] int % frames_ret ) [virtual]`

Read data from a FIFO queue in int16\_t data format

**Parameters**

|               |                                                                                                     |
|---------------|-----------------------------------------------------------------------------------------------------|
| <i>handle</i> | Handle of the FIFO queue. Either zero when the SetSelectedData call was used or the channel number. |
|---------------|-----------------------------------------------------------------------------------------------------|

**Parameters**

|               |                                  |
|---------------|----------------------------------|
| <i>frames</i> | Number of sample frames to read. |
|---------------|----------------------------------|

**Parameters**

|                   |                                                                        |
|-------------------|------------------------------------------------------------------------|
| <i>frames_ret</i> | Number of sample frames which were read, might be smaller than frames. |
|-------------------|------------------------------------------------------------------------|

**11.50.3.21 ChannelBlock\_ReadFramesI32()** [1/2] `virtual void ChannelBlock_ReadFramesI32 (`  
`int handle,`  
`array< int32_t >^ buffer,`  
`int frames_pos,`

```
int frames,
[System::Runtime::InteropServices::Out] int % frames_ret) [virtual]
```

Read data from a FIFO queue in uint32\_t data format

#### Parameters

|               |                                                                                                     |
|---------------|-----------------------------------------------------------------------------------------------------|
| <i>handle</i> | Handle of the FIFO queue. Either zero when the SetSelectedData call was used or the channel number. |
|---------------|-----------------------------------------------------------------------------------------------------|

#### Parameters

|                   |                                            |
|-------------------|--------------------------------------------|
| <i>buffer</i>     | Buffer to put the data from the device in. |
| <i>frames_pos</i> | Position in buffer where to put the data.  |
| <i>frames</i>     | Number of sample frames to read.           |

#### Parameters

|                   |                                                                        |
|-------------------|------------------------------------------------------------------------|
| <i>frames_ret</i> | Number of sample frames which were read, might be smaller than frames. |
|-------------------|------------------------------------------------------------------------|

#### 11.50.3.22 ChannelBlock\_ReadFramesI32() [2/2] virtual array<int32\_t> ^ ChannelBlock\_ReadFramesI32 (

```
int handle,
int frames,
[System::Runtime::InteropServices::Out] int % frames_ret) [virtual]
```

Read data from a FIFO queue in uint32\_t data format

#### Parameters

|               |                                                                                                     |
|---------------|-----------------------------------------------------------------------------------------------------|
| <i>handle</i> | Handle of the FIFO queue. Either zero when the SetSelectedData call was used or the channel number. |
|---------------|-----------------------------------------------------------------------------------------------------|

#### Parameters

|               |                                  |
|---------------|----------------------------------|
| <i>frames</i> | Number of sample frames to read. |
|---------------|----------------------------------|

## Parameters

|                   |                                                                        |
|-------------------|------------------------------------------------------------------------|
| <i>frames_ret</i> | Number of sample frames which were read, might be smaller than frames. |
|-------------------|------------------------------------------------------------------------|

**11.50.3.23 ChannelBlock\_ReadFramesUI16() [1/2]** `virtual void ChannelBlock_ReadFramesUI16 (`  
`int handle,`  
`array< uint16_t >^ buffer,`  
`int frames_pos,`  
`int frames,`  
`[System::Runtime::InteropServices::Out] int % frames_ret ) [virtual]`

Read data from a FIFO queue in uint16\_t data format

## Parameters

|                   |                                                                                                     |
|-------------------|-----------------------------------------------------------------------------------------------------|
| <i>handle</i>     | Handle of the FIFO queue. Either zero when the SetSelectedData call was used or the channel number. |
| <i>buffer</i>     | Buffer to put the data from the device in.                                                          |
| <i>frames_pos</i> | Position in buffer where to put the data.                                                           |
| <i>frames</i>     | Number of sample frames to read.                                                                    |

## Parameters

|                   |                                                                        |
|-------------------|------------------------------------------------------------------------|
| <i>frames_ret</i> | Number of sample frames which were read, might be smaller than frames. |
|-------------------|------------------------------------------------------------------------|

**11.50.3.24 ChannelBlock\_ReadFramesUI16() [2/2]** `virtual array<uint16_t> ^ ChannelBlock_ReadFramesUI16 (`  
`int handle,`  
`int frames,`  
`[System::Runtime::InteropServices::Out] int % frames_ret ) [virtual]`

Read data from a FIFO queue in uint16\_t data format

## Parameters

|               |                                                                                                     |
|---------------|-----------------------------------------------------------------------------------------------------|
| <i>handle</i> | Handle of the FIFO queue. Either zero when the SetSelectedData call was used or the channel number. |
|---------------|-----------------------------------------------------------------------------------------------------|

## Parameters

|               |                                  |
|---------------|----------------------------------|
| <i>frames</i> | Number of sample frames to read. |
|---------------|----------------------------------|

## Parameters

|                   |                                                                        |
|-------------------|------------------------------------------------------------------------|
| <i>frames_ret</i> | Number of sample frames which were read, might be smaller than frames. |
|-------------------|------------------------------------------------------------------------|

## Returns

Array of data from the device.

**11.50.3.25 ChannelBlock\_ReadFramesUI32()** [1/2] `virtual void ChannelBlock_ReadFramesUI32 (`  
`int handle,`  
`array< uint32_t >^ buffer,`  
`int frames_pos,`  
`int frames,`  
`[System::Runtime::InteropServices::Out] int % frames_ret ) [virtual]`

Read data from a FIFO queue in uint32\_t data format

## Parameters

|               |                                                                                                     |
|---------------|-----------------------------------------------------------------------------------------------------|
| <i>handle</i> | Handle of the FIFO queue. Either zero when the SetSelectedData call was used or the channel number. |
|---------------|-----------------------------------------------------------------------------------------------------|

## Parameters

|                   |                                            |
|-------------------|--------------------------------------------|
| <i>buffer</i>     | Buffer to put the data from the device in. |
| <i>frames_pos</i> | Position in buffer where to put the data.  |
| <i>frames</i>     | Number of sample frames to read.           |

## Parameters

|                   |                                                                        |
|-------------------|------------------------------------------------------------------------|
| <i>frames_ret</i> | Number of sample frames which were read, might be smaller than frames. |
|-------------------|------------------------------------------------------------------------|

**11.50.3.26 ChannelBlock\_ReadFramesUI32()** [2/2] virtual array<uint32\_t> ^ ChannelBlock\_Read↔  
FramesUI32 (   
    int *handle*,  
    int *frames*,  
    [System::Runtime::InteropServices::Out] int % *frames\_ret* ) [virtual]

Read data from a FIFO queue in uint32\_t data format

**Parameters**

|               |                                                                                                     |
|---------------|-----------------------------------------------------------------------------------------------------|
| <i>handle</i> | Handle of the FIFO queue. Either zero when the SetSelectedData call was used or the channel number. |
|---------------|-----------------------------------------------------------------------------------------------------|

**Parameters**

|               |                                  |
|---------------|----------------------------------|
| <i>frames</i> | Number of sample frames to read. |
|---------------|----------------------------------|

**Parameters**

|                   |                                                                        |
|-------------------|------------------------------------------------------------------------|
| <i>frames_ret</i> | Number of sample frames which were read, might be smaller than frames. |
|-------------------|------------------------------------------------------------------------|

**11.50.3.27 ClearBuffers()** virtual void ClearBuffers ( ) [virtual]

**11.50.3.28 CMcsUsbDacqNet::GetFilterProperties()** virtual array<CFilterPropertyNet^> ^ CMcs↔  
UsbDacqNet::GetFilterProperties (   
    DacqGroupChannelEnumNet *GroupID* ) [virtual]

**11.50.3.29 GetAdapterType()** virtual AdapterTypeEnumNet GetAdapterType ( ) [virtual]

Gets the adapter which is connected to the MEA2100 device.

**Returns**

AdapterTypeEnumNet which enumerates the possible adapters.

**11.50.3.30 GetAdcDataFormat()** virtual uint32\_t GetAdcDataFormat (   
uint32\_t virtualDevice ) [virtual]

Gets the ADC data format, 16 means 16 bits, 24 means 24 bits, 32 means 32 bits.

#### Returns

The data format in bits.

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

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

**11.50.3.33 GetChannelDataFillSize()** int GetChannelDataFillSize ( )

**11.50.3.34 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]

**11.50.3.35 GetChannelsInBlock()** virtual int32\_t GetChannelsInBlock (   
unsigned int virtualDevice ) [virtual]

Get the number of 16 bit datawords which will be collected per sample frame, use after the device is configured.

#### Returns

Number of 16 bit datawords per sample frame.

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

**11.50.3.37 GetDataMode()** virtual DataModeEnumNet GetDataMode (   
unsigned int virtualDevice ) [virtual]

Gets the data mode, can be 16, 24 or 32bit, all signed or unsigned on the MEA2100 device.

## Parameters

|                      |                        |
|----------------------|------------------------|
| <i>virtualDevice</i> | Virtual device to use. |
|----------------------|------------------------|

## Returns

DataModeEnumNet which enumerates the possible data modes.

**11.50.3.38 GetDigitalSource() [1/5]** `void GetDigitalSource (`  
`DigitalTargetEnumNet digitaltarget,`  
`int32_t NrChannel,`  
`[System::Runtime::InteropServices::Out] DigitalSource< digitalsourceenum >^%`  
`source,`  
`[System::Runtime::InteropServices::Out] int% bitnumber_offset )`

Gets the function/source of an digital output bit.

This is the templated generic implementation.

## Parameters

|                         |                                                    |
|-------------------------|----------------------------------------------------|
| <i>digitaltarget</i>    | The digital target to query.                       |
| <i>NrChannel</i>        | The channel/bit of target to query.                |
| <i>source</i>           | The source/function assignd to the digital target. |
| <i>bitnumber_offset</i> | An offset / bit number with the source/function.   |

**11.50.3.39 GetDigitalSource() [2/5]** `void GetDigitalSource (`  
`DigitalTargetEnumNet digitaltarget,`  
`int32_t NrChannel,`  
`[System::Runtime::InteropServices::Out] DigitalSourceEnumNet% source,`  
`[System::Runtime::InteropServices::Out] int% bitnumber_offset )`

Gets the function/source of an digital output bit.

This overload is for the MEA2100 device.

## Parameters

|                         |                                                    |
|-------------------------|----------------------------------------------------|
| <i>digitaltarget</i>    | The digital target to query.                       |
| <i>NrChannel</i>        | The channel/bit of target to query.                |
| <i>source</i>           | The source/function assignd to the digital target. |
| <i>bitnumber_offset</i> | An offset / bit number with the source/function.   |



**11.50.3.40 GetDigitalSource() [3/5]** `void GetDigitalSource ( DigitalTargetEnumNet digitaltarget, int32_t NrChannel, [System::Runtime::InteropServices::Out] MEA2100_256DigitalSourceEnumNet% source, [System::Runtime::InteropServices::Out] int% bitnumber_offset )`

Gets the function/source of an digital output bit.

This overload is for the MEA2100-256 device.

#### Parameters

|                         |                                                    |
|-------------------------|----------------------------------------------------|
| <i>digitaltarget</i>    | The digital target to query.                       |
| <i>NrChannel</i>        | The channel/bit of target to query.                |
| <i>source</i>           | The source/function assignd to the digital target. |
| <i>bitnumber_offset</i> | An offset / bit number with the source/function.   |

**11.50.3.41 GetDigitalSource() [4/5]** `void GetDigitalSource ( DigitalTargetEnumNet digitaltarget, int32_t NrChannel, [System::Runtime::InteropServices::Out] SCUDigitalSourceEnumNet% source, [System::Runtime::InteropServices::Out] int% bitnumber_offset )`

Gets the function/source of an digital output bit.

This overload is for the SCU device.

#### Parameters

|                         |                                                    |
|-------------------------|----------------------------------------------------|
| <i>digitaltarget</i>    | The digital target to query.                       |
| <i>NrChannel</i>        | The channel/bit of target to query.                |
| <i>source</i>           | The source/function assignd to the digital target. |
| <i>bitnumber_offset</i> | An offset / bit number with the source/function.   |

**11.50.3.42 GetDigitalSource() [5/5]** `void GetDigitalSource ( DigitalTargetEnumNet digitaltarget, int32_t NrChannel, [System::Runtime::InteropServices::Out] W2100DigitalSourceEnumNet% source, [System::Runtime::InteropServices::Out] int% bitnumber_offset )`

Gets the function/source of an digital output bit.

This overload is for the W2100 device.

#### Parameters

|                         |                                                    |
|-------------------------|----------------------------------------------------|
| <i>digitaltarget</i>    | The digital target to query.                       |
| <i>NrChannel</i>        | The channel/bit of target to query.                |
| <i>source</i>           | The source/function assignd to the digital target. |
| <i>bitnumber_offset</i> | An offset / bit number with the source/function.   |

**11.50.3.43 GetErrorMessage()** `uint32_t GetErrorMessage (`  
`[System::Runtime::InteropServices::Out] String^% errorString,`  
`[System::Runtime::InteropServices::Out] int% info )`

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

**11.50.3.45 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.

**11.50.3.46 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.

**11.50.3.47 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.

**11.50.3.48 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.

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

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

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

Gets the maximal sampling frequency of the device.

#### Returns

Sampling frequency in Hz.

**11.50.3.52 GetMeaLayout()** virtual [MeaLayoutEnumNet](#) GetMeaLayout ( ) [virtual]

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

#### Returns

MeaLayoutEnumNet which enumerates the MEA types.

**11.50.3.53 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.

**11.50.3.54 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

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

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

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

Gets the sampling frequency of the device.

#### Returns

Sampling frequency in Hz.

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

**11.50.3.59 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.

**11.50.3.60 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.

**11.50.3.61 HWInfo()** `CHWInfo ^ HWInfo ( )`

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

Start sampling.

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

Start sampling.

**Parameters**

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

**11.50.3.64 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> |  |

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

Stop sampling.

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

Stop sampling.

Parameters

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

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

Stop sampling together with the STG.

Parameters

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

**11.50.3.68 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> |  |
|-----------------------|--|

**11.50.3.69 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.                              |

**11.50.3.70 SetDigitalSource()** [1/5] void SetDigitalSource (   
*DigitalTargetEnumNet* *digitaltarget*,   
int32\_t *NrChannel*,   
*DigitalSource*< digitalsourceenum >^ *source*,   
int *bitnumber\_offset* )

Sets the function/source of an digital output bit.

This is the templated generic implementation.



## Parameters

|                         |                                                      |
|-------------------------|------------------------------------------------------|
| <i>digitaltarget</i>    | The digital target to change.                        |
| <i>NrChannel</i>        | The channel/bit of target to change.                 |
| <i>source</i>           | The source/function to assign to the digital target. |
| <i>bitnumber_offset</i> | An offset / bit number with the source/function.     |

**11.50.3.71 SetDigitalSource() [2/5]** `void SetDigitalSource (`  
`DigitalTargetEnumNet digitaltarget,`  
`int32_t NrChannel,`  
`DigitalSourceEnumNet source,`  
`int bitnumber_offset )`

Sets the function/source of an digital output bit.

This overload is for the MEA2100 device.

## Parameters

|                         |                                                      |
|-------------------------|------------------------------------------------------|
| <i>digitaltarget</i>    | The digital target to change.                        |
| <i>NrChannel</i>        | The channel/bit of target to change.                 |
| <i>source</i>           | The source/function to assign to the digital target. |
| <i>bitnumber_offset</i> | An offset / bit number with the source/function.     |

**11.50.3.72 SetDigitalSource() [3/5]** `void SetDigitalSource (`  
`DigitalTargetEnumNet digitaltarget,`  
`int32_t NrChannel,`  
`MEA2100_256DigitalSourceEnumNet source,`  
`int bitnumber_offset )`

Sets the function/source of an digital output bit.

This overload is for the MEA2100-256 device.

## Parameters

|                         |                                                      |
|-------------------------|------------------------------------------------------|
| <i>digitaltarget</i>    | The digital target to change.                        |
| <i>NrChannel</i>        | The channel/bit of target to change.                 |
| <i>source</i>           | The source/function to assign to the digital target. |
| <i>bitnumber_offset</i> | An offset / bit number with the source/function.     |

**11.50.3.73 SetDigitalSource() [4/5]** `void SetDigitalSource (`  
`DigitalTargetEnumNet digitaltarget,`

```

 int32_t NrChannel,
 SCUDigitalSourceEnumNet source,
 int bitnumber_offset)

```

Sets the function/source of an digital output bit.

This overload is for the SCU device.

#### Parameters

|                         |                                                      |
|-------------------------|------------------------------------------------------|
| <i>digitaltarget</i>    | The digital target to change.                        |
| <i>NrChannel</i>        | The channel/bit of target to change.                 |
| <i>source</i>           | The source/function to assign to the digital target. |
| <i>bitnumber_offset</i> | An offset / bit number with the source/function.     |

**11.50.3.74 SetDigitalSource()** [5/5] `void SetDigitalSource (`  
`DigitalTargetEnumNet digitaltarget,`  
`int32_t NrChannel,`  
`W2100DigitalSourceEnumNet source,`  
`int bitnumber_offset )`

Sets the function/source of an digital output bit.

This overload is for the W2100 device.

#### Parameters

|                         |                                                      |
|-------------------------|------------------------------------------------------|
| <i>digitaltarget</i>    | The digital target to change.                        |
| <i>NrChannel</i>        | The channel/bit of target to change.                 |
| <i>source</i>           | The source/function to assign to the digital target. |
| <i>bitnumber_offset</i> | An offset / bit number with the source/function.     |

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

**11.50.3.76 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. |
|-------------|---------------------------|

**11.50.3.77 SetSelectedChannels() [1/4]** `virtual void SetSelectedChannels (`  
`array< bool >^ selectedChannels,`  
`int queuesize,`  
`int threshold,`  
`SampleSizeNet samplesize,`  
`int ChannelsInBlock ) [virtual]`

Create a FIFO queue per channel. Each channel will have its own FIFO and Callback function.

When using a 32bit sample size, the number obtained from GetChannelsInBlock must be divided by 2 to be used here, since GetChannelsInBlock returns the number of 16 bit datapoints per sample frame, while this functions uses the number of sample frames in its own data format.

## Parameters

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

## Parameters

|                  |                                          |
|------------------|------------------------------------------|
| <i>queuesize</i> | Size of sample frames the FIFO can hold. |
|------------------|------------------------------------------|

## Parameters

|                  |                                                                                       |
|------------------|---------------------------------------------------------------------------------------|
| <i>threshold</i> | Number of sample frames the FIFO must acquire before the callback function is called. |
|------------------|---------------------------------------------------------------------------------------|

## Parameters

|                   |                                            |
|-------------------|--------------------------------------------|
| <i>samplesize</i> | size of the datawords, either 16 or 32bit. |
|-------------------|--------------------------------------------|

## Parameters

|                        |                                         |
|------------------------|-----------------------------------------|
| <i>ChannelsInBlock</i> | value obtained from GetChannelsInBlock. |
|------------------------|-----------------------------------------|

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

**11.50.3.79 SetSelectedChannels() [3/4]** virtual void SetSelectedChannels (   
int *nChannels*,  
int *queuesize*,  
int *threshold*,  
SampleSizeNet *samplesize*,  
int *ChannelsInBlock* ) [virtual]

Create a FIFO queue per channel. Each channel will have its own FIFO and Callback function.

When using a 32bit sample size, the number obtained from GetChannelsInBlock must be divided by 2 to be used here, since GetChannelsInBlock returns the number of 16 bit datapoints per sample frame, while this functions uses the number of sample frames in its own data format.

**Parameters**

|                  |                                                 |
|------------------|-------------------------------------------------|
| <i>nChannels</i> | Number of channels to be collected in the FIFO. |
|------------------|-------------------------------------------------|

**Parameters**

|                  |                                          |
|------------------|------------------------------------------|
| <i>queuesize</i> | Size of sample frames the FIFO can hold. |
|------------------|------------------------------------------|

**Parameters**

|                  |                                                                                        |
|------------------|----------------------------------------------------------------------------------------|
| <i>threshold</i> | Number of samples frames the FIFO must acquire before the callback function is called. |
|------------------|----------------------------------------------------------------------------------------|

**Parameters**

|                        |                                            |
|------------------------|--------------------------------------------|
| <i>samplesize</i>      | size of the datawords, either 16 or 32bit. |
| <i>ChannelsInBlock</i> | value obtained from GetChannelsInBlock.    |

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

```

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

```

**11.50.3.81 SetSelectedChannelsQueue()** [1/4] virtual void SetSelectedChannelsQueue (

```

 array< bool >^ selectedChannels,
 int queuesize,
 int threshold,
 SampleSizeNet samplesize,
 int ChannelsInBlock) [virtual]

```

Create a common FIFO queue for all channels. Data in callback will be a list per channel. Use ChannelBlock\_↔ ReadFramesDict... with handle = 0 to read the data.

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

#### Parameters

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

#### Parameters

|                  |                                          |
|------------------|------------------------------------------|
| <i>queuesize</i> | Size of sample frames the FIFO can hold. |
|------------------|------------------------------------------|

#### Parameters

|                  |                                                                                       |
|------------------|---------------------------------------------------------------------------------------|
| <i>threshold</i> | Number of sample frames the FIFO must acquire before the callback function is called. |
|------------------|---------------------------------------------------------------------------------------|

#### Parameters

|                   |                                            |
|-------------------|--------------------------------------------|
| <i>samplesize</i> | size of the datawords, either 16 or 32bit. |
|-------------------|--------------------------------------------|

**Parameters**

|                        |                                         |
|------------------------|-----------------------------------------|
| <i>ChannelsInBlock</i> | value obtained from GetChannelsInBlock. |
|------------------------|-----------------------------------------|

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

**11.50.3.83 SetSelectedChannelsQueue() [3/4]** virtual void SetSelectedChannelsQueue (   
int *nChannels*,  
int *queuesize*,  
int *threshold*,  
SampleSizeNet *samplesize*,  
int *ChannelsInBlock* ) [virtual]

Create a common FIFO queue for all channels. Data in callback will be a list per channel. Use ChannelBlock\_↔ ReadFramesDict... with handle = 0 to read the data.

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

**Parameters**

|                  |                                                 |
|------------------|-------------------------------------------------|
| <i>nChannels</i> | Number of channels to be collected in the FIFO. |
|------------------|-------------------------------------------------|

**Parameters**

|                  |                                          |
|------------------|------------------------------------------|
| <i>queuesize</i> | Size of sample frames the FIFO can hold. |
|------------------|------------------------------------------|

**Parameters**

|                  |                                                                                       |
|------------------|---------------------------------------------------------------------------------------|
| <i>threshold</i> | Number of sample frames the FIFO must acquire before the callback function is called. |
|------------------|---------------------------------------------------------------------------------------|

## Parameters

|                   |                                            |
|-------------------|--------------------------------------------|
| <i>samplesize</i> | size of the datawords, either 16 or 32bit. |
|-------------------|--------------------------------------------|

## Parameters

|                        |                                         |
|------------------------|-----------------------------------------|
| <i>ChannelsInBlock</i> | value obtained from GetChannelsInBlock. |
|------------------------|-----------------------------------------|

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

**11.50.3.85 SetSelectedData()** [1/4] virtual void SetSelectedData (   
     array< bool >^ *selectedChannels*,  
     int *queuesize*,  
     int *threshold*,  
     SampleSizeNet *samplesize*,  
     int *ChannelsInBlock* ) [virtual]

Create a common FIFO queue for all channels. Use handle = 0 in the ChannelBlock\_ReadFrames... functions.

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

## Parameters

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

## Parameters

|                  |                                          |
|------------------|------------------------------------------|
| <i>queuesize</i> | Size of sample frames the FIFO can hold. |
|------------------|------------------------------------------|

## Parameters

|                  |                                                                                       |
|------------------|---------------------------------------------------------------------------------------|
| <i>threshold</i> | Number of sample frames the FIFO must acquire before the callback function is called. |
|------------------|---------------------------------------------------------------------------------------|

## Parameters

|                        |                                            |
|------------------------|--------------------------------------------|
| <i>samplesize</i>      | size of the datawords, either 16 or 32bit. |
| <i>ChannelsInBlock</i> | value obtained from GetChannelsInBlock.    |

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

**11.50.3.87 SetSelectedData() [3/4]** virtual void SetSelectedData (   
     int nChannels,   
     int queuesize,   
     int threshold,   
     SampleSizeNet samplesize,   
     int ChannelsInBlock ) [virtual]

Create a common FIFO queue for all channels. Use handle = 0 in the ChannelBlock\_ReadFrames... functions.

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

## Parameters

|                  |                                                 |
|------------------|-------------------------------------------------|
| <i>nChannels</i> | Number of channels to be collected in the FIFO. |
|------------------|-------------------------------------------------|

## Parameters

|                  |                                          |
|------------------|------------------------------------------|
| <i>queuesize</i> | Size of sample frames the FIFO can hold. |
|------------------|------------------------------------------|



## Parameters

|                  |                                                                                       |
|------------------|---------------------------------------------------------------------------------------|
| <i>threshold</i> | Number of sample frames the FIFO must acquire before the callback function is called. |
|------------------|---------------------------------------------------------------------------------------|

## Parameters

|                        |                                            |
|------------------------|--------------------------------------------|
| <i>samplesize</i>      | size of the datawords, either 16 or 32bit. |
| <i>ChannelsInBlock</i> | value obtained from GetChannelsInBlock.    |

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

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

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

**11.50.3.91 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. |
|--------------------------|--------------------------------------------------------------------------|

**11.50.3.92 SetVoltageRangeInMicroVolt()** `virtual void SetVoltageRangeInMicroVolt (`  
    `int32_t voltageRange,`  
    `unsigned int virtualDevice ) [virtual]`

Sets the voltage range on devices which support multiple voltage ranges.

**Parameters**

|                     |                                  |
|---------------------|----------------------------------|
| <i>voltageRange</i> | Voltage Range to use in $\mu$ V. |
|---------------------|----------------------------------|

This replaces SetVoltageRange, where the value of the range was in mV!

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

Start the data acquisition thread and sampling.

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

Start the data acquisition thread and sampling.

**Parameters**

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

**11.50.3.95 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. |
|---------------------|----------------------|

**11.50.3.96 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. |
|----------------------|--------------------------|

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

Start the data acquisition thread.

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

Start the data acquisition thread.

**Parameters**

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

**11.50.3.99 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. |
|---------------------|----------------------|

**11.50.3.100 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. |
|----------------------|--------------------------|

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

Stop the data acquisition thread and sampling.

**11.50.3.102 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. |
|----------------------|--------------------------|

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

**11.50.4 Member Data Documentation**

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

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

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

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

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

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

### 11.50.5 Property Documentation

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

The sampling frequency of the device in Hz.

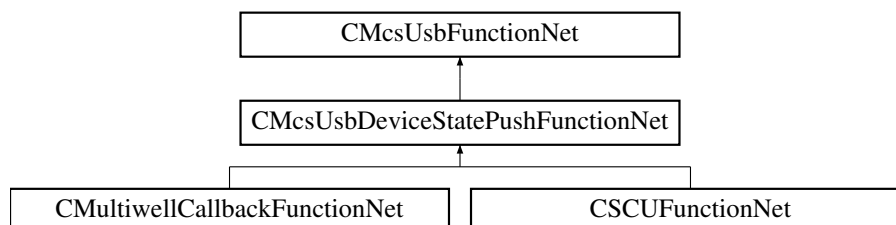
### 11.50.6 Event Documentation

**11.50.6.1 ChannelDataEvent** `OnChannelData^ ChannelDataEvent [add], [remove], [raise]`

**11.50.6.2 ErrorEvent** `OnError^ ErrorEvent [add], [remove], [raise]`

## 11.51 CMcsUsbDeviceStatePushFunctionNet Class Reference

Inheritance diagram for CMcsUsbDeviceStatePushFunctionNet:



### Public Member Functions

- void [TriggerStatus](#) ()

## Protected Member Functions

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

## Events

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

## Additional Inherited Members

### 11.51.1 Constructor & Destructor Documentation

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

### 11.51.2 Member Function Documentation

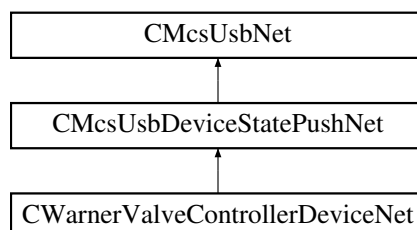
**11.51.2.1 TriggerStatus()** void TriggerStatus ( )

### 11.51.3 Event Documentation

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

## 11.52 CMcsUsbDeviceStatePushNet Class Reference

Inheritance diagram for CMcsUsbDeviceStatePushNet:





**Public Member Functions**

- void [TriggerStatus](#) ()

**Protected Member Functions**

- [CMcsUsbDeviceStatePushNet](#) ([CMcsUsbPointerContainer](#)^ pDevice)

**Events**

- [OnMcsUsbDeviceState](#)^ [McsUsbDeviceStateEvent](#) [add, remove, raise]

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

**11.52.1.1 CMcsUsbDeviceStatePushNet()** [CMcsUsbDeviceStatePushNet](#) (  
[CMcsUsbPointerContainer](#)^ pDevice ) [protected]

**11.52.2 Member Function Documentation**

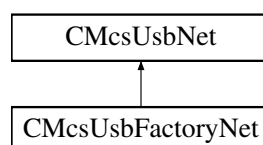
**11.52.2.1 TriggerStatus()** void [TriggerStatus](#) ( )

**11.52.3 Event Documentation**

**11.52.3.1 McsUsbDeviceStateEvent** [OnMcsUsbDeviceState](#)^ [McsUsbDeviceStateEvent](#) [add], [remove], [raise]

**11.53 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

#### 11.53.1 Constructor & Destructor Documentation

**11.53.1.1** [CMcsUsbFactoryNet\(\)](#) [CMcsUsbFactoryNet](#) ( )

**11.53.1.2** [~CMcsUsbFactoryNet\(\)](#) [~CMcsUsbFactoryNet](#) ( )

#### 11.53.2 Member Function Documentation

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

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

**11.53.2.3 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 )

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

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

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

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

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

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

**11.53.2.10 GetDestinationName()** [1/2] `String ^ GetDestinationName ( CFirmwareDestinationNet dest )`

**11.53.2.11 GetDestinationName()** [2/2] `String ^ GetDestinationName ( unsigned int index )`

**11.53.2.12 GetDestinationSerialNumber()** `String ^ GetDestinationSerialNumber ( CFirmwareDestinationNet dest )`

**11.53.2.13 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.

**11.53.2.14 GetFirmwareVersionFromFile()** [1/2] `static bool GetFirmwareVersionFromFile ( String^ FirmwareFile, [System::Runtime::InteropServices::Out] uint32_t% Version ) [static]`

Retrives version info from a Firmware update file.

**11.53.2.15 GetFirmwareVersionFromFile()** [2/2] `static bool GetFirmwareVersionFromFile ( String^ FirmwareFile, [System::Runtime::InteropServices::Out] uint32_t% Version, [System::Runtime::InteropServices::Out] uint32_t% Position ) [static]`

**11.53.2.16 GetFirmwareVersionFromHexFile()** `static bool GetFirmwareVersionFromHexFile ( String^ FirmwareFile, [System::Runtime::InteropServices::Out] uint32_t% Version ) [static]`

**11.53.2.17 GetNumDestinations()** `unsigned int GetNumDestinations ( )`

**11.53.2.18 GetUSBDeviceIDFromFX3Image()** `static uint32_t GetUSBDeviceIDFromFX3Image ( String^ FirmwareFile ) [static]`

**11.53.2.19 GetUsercodeFromBitFile()** `static bool GetUsercodeFromBitFile ( String^ FirmwareFile, [System::Runtime::InteropServices::Out] unsigned int% Usercode ) [static]`

**11.53.2.20 GetUsercodeFromFlash()** `bool GetUsercodeFromFlash ( unsigned int FPGA, unsigned int Address, [System::Runtime::InteropServices::Out] unsigned int% Usercode )`

**11.53.2.21 GetXilinxFlashOffset()** `int32_t GetXilinxFlashOffset ( CFirmwareDestinationNet dest )`

**11.53.2.22 GetXilinxFlashReadCommand()** `uint32_t GetXilinxFlashReadCommand ( CFirmwareDestinationNet dest )`

**11.53.2.23 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> . |
|------------------|------------------------------------------------------------------------|

**11.53.2.24 LoadUserFirmware()** [2/2] `bool LoadUserFirmware (`  
    `String^ FirmwareFile,`  
    `CMcsUsbListEntryNet^ listEntry,`  
    `uint32_t LockMask )`

**11.53.2.25 ReadBlockFromFlash()** [1/2] `array<unsigned char> ^ ReadBlockFromFlash (`  
    `unsigned int FPGA,`  
    `unsigned int Address )`

**11.53.2.26 ReadBlockFromFlash()** [2/2] `void ReadBlockFromFlash (`  
    `unsigned int FPGA,`  
    `unsigned int Address,`  
    `array< unsigned char >^ buffer,`  
    `int position )`

**11.53.2.27 ReadBlockFromIFBGlobalEEProm()** `array<unsigned char> ^ ReadBlockFromIFBGlobal↵`  
`EEProm (`  
    `unsigned int Address )`

**11.53.2.28 ReadBlockFromNVMEM()** `array<unsigned char> ^ ReadBlockFromNVMEM (`  
    `unsigned int FPGA,`  
    `unsigned int Offset,`  
    `unsigned int Address )`

**11.53.2.29 SetDestinationSerialNumber()** `void SetDestinationSerialNumber (`  
    `CFirmwareDestinationNet dest,`  
    `String^ serialnumber )`

**11.53.2.30 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. |
|------------------|-----------------------------------|

**11.53.2.31 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. |
|------------------|-----------------------------------|

**11.53.2.32 UpdateFirmware()** [3/5] `bool UpdateFirmware (`  
    `String^ FirmwareFile,`  
    `CMcsUsbListEntryNet^ listEntry,`  
    `CFirmwareDestinationNet dest,`  
    `bool SkipWait,`  
    `uint32_t LockMask )`

**11.53.2.33 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. |
|---------------------|--------------------------------|

```

11.53.2.34 UpdateFirmware() [5/5] bool UpdateFirmware (
 String^ FirmwareFile,
 CMcsUsbListEntryNet^ listEntry,
 CFirmwareDestinationNet Dest,
 OnUpdateFirmwareStatusChange^ deleg,
 OnUpdateFirmwareProgress^ progress,
 bool SkipWait,
 unsigned int LockMask)

```

### 11.53.3 Member Data Documentation

**11.53.3.1 FX3MCSDDataAddress** const uint32\_t FX3MCSDDataAddress = 0x40037E00 [static]

**11.53.3.2 FX3MCSDDataDeviceIdOffset** const uint32\_t FX3MCSDDataDeviceIdOffset = 0x4 [static]

**11.53.3.3 FX3MCSDDataIFB1ImageOffset** const uint32\_t FX3MCSDDataIFB1ImageOffset = 0x2C [static]

**11.53.3.4 FX3MCSDDataIFB2ImageOffset** const uint32\_t FX3MCSDDataIFB2ImageOffset = 0xC [static]

**11.53.3.5 FX3MCSDDataVersionOffset** const uint32\_t FX3MCSDDataVersionOffset = 0x8 [static]

|                  |                                                                                                                                                          |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| ChcAdFunctionSet |                                                                                                                                                          |
|                  | ChcGroupChannelSelectorTemplateSet, W210DcGroupChannelFrameSet, W210DcGroupChannelFrame, ChcGroupChannelIdW210Set >                                      |
|                  | ChcGroupChannelSelectorTemplateSet, DcGroupChannelFrameSet, DcGroupChannelFrame, ChcGroupChannelIdDcSet >                                                |
|                  | ChcGroupChannelSelectorTemplateSet > in, in, ChcGroupChannelIdGroupSet >                                                                                 |
|                  | ChcGroupChannelSelectorTemplateSet, SCChcGroupChannelFrameSet, SCChcGroupChannelFrame, ChcGroupChannelIdSCSet >                                          |
|                  | ChcGroupChannelSelectorTemplateSet, MEA210, 210DcGroupChannelFrameSet, MEA210, 210DcGroupChannelFrame, ChcGroupChannelIdMEA210, 210DcGroupChannelIdSet > |
|                  | ChcMixerFunctionSet                                                                                                                                      |
|                  | ChcCalibrationFunctionSet                                                                                                                                |
|                  | ChcGroupChannelSelectorTemplateSet, DcGroupChannelFrameTemplateSet, DcGroupChannelFrameTemplate, ChcGroupChannelIdTemplateSet >                          |
|                  | ChcDeInterlacedFunctionSet                                                                                                                               |
|                  | ChcFilterConfigurationSet                                                                                                                                |
|                  | ChcFilterConfigurationFunctionSet                                                                                                                        |
|                  | ChcGammaFunctionSet                                                                                                                                      |
|                  | ChcHdmiFunctionSet                                                                                                                                       |
|                  | ChcHdmiChannelFunctionSet                                                                                                                                |
|                  | ChcHDMI100_210FunctionSet                                                                                                                                |
|                  | ChcHdmiSet                                                                                                                                               |
|                  | ChcHdmi_AudioParameterSet                                                                                                                                |
|                  | ChcHdmi_FunctionSet                                                                                                                                      |
|                  | ChcHdmi_FYIParameterSet                                                                                                                                  |
|                  | ChcHdmi_MonoConnectSet                                                                                                                                   |
|                  | ChcHdmi_SensorSet                                                                                                                                        |
|                  | ChcHdmi_TemplateSet                                                                                                                                      |
|                  | ChcHdmi_VoltageModeSet                                                                                                                                   |
|                  | ChcUsbDevicePushFunctionSet                                                                                                                              |
|                  | ChcFunctionSet                                                                                                                                           |
|                  | ChcMaxLatencyFunctionSet                                                                                                                                 |
|                  | ChcMaxDigitalDelayFunctionSet                                                                                                                            |
|                  | ChcMediaFunctionSet                                                                                                                                      |
|                  | ChcMediaOperationFunctionSet                                                                                                                             |
|                  | ChcMuxFunctionSet                                                                                                                                        |
|                  | ChcOpFunctionSet                                                                                                                                         |
|                  | ChcProgramProcessControlSet                                                                                                                              |
|                  | ChcPulseGeneratorFunctionSet                                                                                                                             |
|                  | ChcPFunctionSet                                                                                                                                          |
|                  | ChcRobt_FYIProgram_FunctionSet                                                                                                                           |
|                  | ChcRobt_FYITemp_FunctionSet                                                                                                                              |
|                  | ChcSensorFunctionSet                                                                                                                                     |
|                  | ChcTEEPFunctionSet                                                                                                                                       |
|                  | ChcMixerConfigurationFunctionSet                                                                                                                         |
|                  | CW210_SensorFunctionSet                                                                                                                                  |
|                  | CWAnalogFunctionSet                                                                                                                                      |
|                  | CWAnalogControlDeviceControlFunctionSet                                                                                                                  |
|                  | ChcHdmiChannelFunctionSet                                                                                                                                |

- **CMcsUsbFunctionNet** (**CMcsUsbNet**<sup>^</sup> mcsusb)
- virtual ~**CMcsUsbFunctionNet** (void)
- **!CMcsUsbFunctionNet** ()
- void **ThrowCUsbExceptionNetOnError** (uint32 t status)

- `CMcUsbFunctionNet (CMcUsbNet^ mcsusb, CMcUsbFunctionPointerContainer^ mcsusbfunction)`

- CMcsUsbNet ^ m\_pMcsUsb
- CMcsUsbFunction \* m\_pMcsUsbFunction

Generated by Doxygen

**11.54.1.1 CMcsUsbFunctionNet()** [1/2] `CMcsUsbFunctionNet (`  
`CMcsUsbNet^ mcsusb )`

**11.54.1.2 ~CMcsUsbFunctionNet()** `virtual ~CMcsUsbFunctionNet (`  
`void ) [virtual]`

**11.54.1.3 "!CMcsUsbFunctionNet()** `!CMcsUsbFunctionNet ( )`

**11.54.1.4 CMcsUsbFunctionNet()** [2/2] `CMcsUsbFunctionNet (`  
`CMcsUsbNet^ mcsusb,`  
`CMcsUsbFunctionPointerContainer^ mcsusbfunction ) [protected]`

## 11.54.2 Member Function Documentation

**11.54.2.1 ThrowCUsbExceptionNetOnError()** `void ThrowCUsbExceptionNetOnError (`  
`uint32_t status )`

## 11.54.3 Member Data Documentation

**11.54.3.1 m\_pMcsUsb** `CMcsUsbNet ^ m_pMcsUsb [protected]`

**11.54.3.2 m\_pMcsUsbFunction** `CMcsUsbFunction* m_pMcsUsbFunction [protected]`

## 11.55 CMcsUsbFunctionPointerContainer Class Reference

## 11.56 CMcsUsbListEntryNet Class Reference

McsUsbListEntryNet identifies a connected device.

## Public Member Functions

- `~CMcsUsbListEntryNet ()`
- virtual bool `Equals (Object^ obj)` override  
*Checks whether 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]

### 11.56.1 Detailed Description

McsUsbListEntryNet identifies a connected device.

### 11.56.2 Constructor & Destructor Documentation

#### 11.56.2.1 `~CMcsUsbListEntryNet()` `~CMcsUsbListEntryNet ()`

### 11.56.3 Member Function Documentation

**11.56.3.1 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. |
|------------|----------------------------------------------------------|

**11.56.3.2 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.

**11.56.3.3 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.

**11.56.3.4 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.

**11.56.3.5 GetEntryCount()** [1/2] `static unsigned int GetEntryCount ( ) [static]`

Returns the number of devices connected to the computer.

## Returns

The number of devices.

**11.56.3.6 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.

**11.56.3.7 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)". |
|---------------|----------------------------------------------------------------|

**11.56.3.8 ToString()** `virtual String ^ ToString ( ) [override], [virtual]`

**11.56.4 Property Documentation**

**11.56.4.1 DeviceId** `DeviceIdNet^ DeviceId [get]`

**11.56.4.2 DeviceName** `String^ DeviceName [get]`

The device name of the device represented by this [CMcsUsbListEntryNet](#).

**11.56.4.3 HwVersion** `String^ HwVersion [get]`

The hardware revision of the device represented by this [CMcsUsbListEntryNet](#).

**11.56.4.4 Manufacturer** `String^ Manufacturer [get]`

The Manufacturer ID of the device represented by this [CMcsUsbListEntryNet](#).

**11.56.4.5 Product** `String^ Product [get]`

The Product ID of the device represented by this [CMcsUsbListEntryNet](#).

**11.56.4.6 SerialNumber** `String^ SerialNumber [get]`

The serial number of the device represented by this [CMcsUsbListEntryNet](#).



## 11.57 CMcsUsbListNet Class Reference

Class to handle a list of connected MCS USB devices.

### Public Member Functions

- [CMcsUsbListNet](#) ([DeviceEnumNet](#) McsUsbDevice)  
*Initializes a new instance of [CMcsUsbListNet](#) class.*
- [CMcsUsbListNet](#) (array< [DeviceIdNet](#)<sup>^</sup>> ^ DeviceIdList)  
*Initializes a new instance of [CMcsUsbListNet](#) class.*
- [~CMcsUsbListNet](#) ()  
*Destructor: called by [Dispose\(\)](#)*
- [!CMcsUsbListNet](#) ()  
*Finalizer: called by GC before collecting*
- void [SetStringFormat](#) (String ^ format)  
*Specify the text the [CMcsUsbListEntryNet.ToString\(\)](#) function should return. The special code N expands to the device name and S expands to the serial number of the device.*
- uint32\_t [GetNumberOfDevices](#) ()  
*Gets the number of devices currently in the list.*
- [CMcsUsbListEntryNet](#) ^ [GetUsbListEntry](#) (unsigned int index)  
*Returns one [CMcsUsbListEntryNet](#) from the list of USB Devices connected to the computer.*
- array< [CMcsUsbListEntryNet](#)<sup>^</sup>> ^ [GetUsbListEntries](#) ()  
*Returns all entries from the list of USB Devices connected to the computer.*
- bool [IsDeviceTypeOf](#) ([CMcsUsbListEntryNet](#)<sup>^</sup> entry, [DeviceEnumNet](#) McsUsbDevice)

### Properties

- uint32\_t [Count](#) [get]  
*Gets the number of devices currently in the list.*

### Events

- [OnDeviceArrivalRemoval](#)<sup>^</sup> [DeviceArrival](#)
- [OnDeviceArrivalRemoval](#)<sup>^</sup> [DeviceRemoval](#)

#### 11.57.1 Detailed Description

Class to handle a list of connected MCS USB devices.

#### 11.57.2 Constructor & Destructor Documentation

##### 11.57.2.1 CMcsUsbListNet() [1/2] [CMcsUsbListNet](#) ( [DeviceEnumNet](#) McsUsbDevice )

Initializes a new instance of [CMcsUsbListNet](#) class.

**11.57.2.2 CMcsUsbListNet()** [2/2] `CMcsUsbListNet ( array< DeviceIdNet^>^ DeviceIdList )`

Initializes a new instance of `CMcsUsbListNet` class.

**11.57.2.3 ~CMcsUsbListNet()** `~CMcsUsbListNet ( )`

Destructor: called by `Dispose()`

**11.57.2.4 ~!CMcsUsbListNet()** `!CMcsUsbListNet ( )`

Finalizer: called by GC before collecting

### 11.57.3 Member Function Documentation

**11.57.3.1 GetNumberOfDevices()** `uint32_t GetNumberOfDevices ( )`

Gets the number of devices currently in the list.

#### Returns

The number of devices currently in the list.

**11.57.3.2 GetUsbListEntries()** `array<CMcsUsbListEntryNet^> ^ GetUsbListEntries ( )`

Returns all entries from the list of USB Devices connected to the computer.

**11.57.3.3 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. |
|--------------|-----------------------------|

**11.57.3.4 IsDeviceTypeOf()** `bool IsDeviceTypeOf (`  
     `CMcsUsbListEntryNet^ entry,`  
     `DeviceEnumNet McsUsbDevice )`

**11.57.3.5 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)". |
|---------------|----------------------------------------------------------------|

## 11.57.4 Property Documentation

**11.57.4.1 Count** `uint32_t Count [get]`

Gets the number of devices currently in the list.

## 11.57.5 Event Documentation

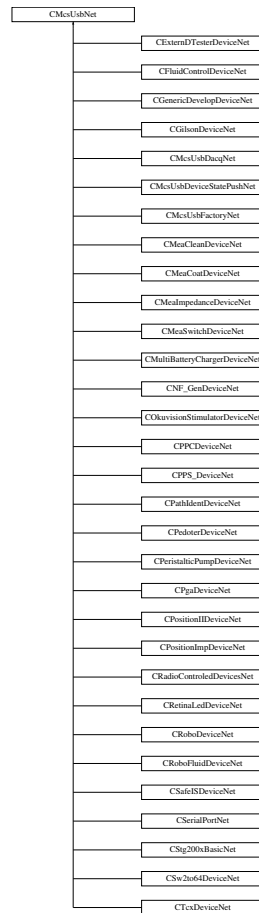
**11.57.5.1 DeviceArrival** `OnDeviceArrivalRemoval^ DeviceArrival`

**11.57.5.2 DeviceRemoval** `OnDeviceArrivalRemoval^ DeviceRemoval`

## 11.58 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)
- bool [ReadEepromRegisterPreconfig](#) (uint32\_t EEPROMBase, uint32\_t DMA\_reg, [System::Runtime::InteropServices::Out]uint32\_t DMA\_value)
- bool [ReadEepromRegisterPreconfig](#) (uint32\_t EEPROMBase, uint32\_t DMA\_reg, [System::Runtime::InteropServices::Out]uint32\_t DMA\_value, uint32\_t EEPROMSize)
- bool [ReadEepromRegisterPreconfig](#) (uint32\_t EEPROMBase, uint32\_t DMA\_reg, [System::Runtime::InteropServices::Out]uint32\_t DMA\_value, uint32\_t EEPROMSize, uint32\_t EepromStartAddress)
- void [WriteEepromRegisterPreconfig](#) (uint32\_t EEPROMBase, uint32\_t DMA\_reg, uint32\_t DMA\_value)
- void [WriteEepromRegisterPreconfig](#) (uint32\_t EEPROMBase, uint32\_t DMA\_reg, uint32\_t DMA\_value, uint32\_t EEPROMSize)
- void [WriteEepromRegisterPreconfig](#) (uint32\_t EEPROMBase, uint32\_t DMA\_reg, uint32\_t DMA\_value, uint32\_t EEPROMSize, uint32\_t EepromStartAddress)
- void [EraseEepromRegisterPreconfig](#) (uint32\_t EEPROMBase, uint32\_t DMA\_reg)
- void [EraseEepromRegisterPreconfig](#) (uint32\_t EEPROMBase, uint32\_t DMA\_reg, uint32\_t EEPROMSize)
- void [EraseEepromRegisterPreconfig](#) (uint32\_t EEPROMBase, uint32\_t DMA\_reg, uint32\_t EEPROMSize, uint32\_t EepromStartAddress)
- unsigned int [GetLastUSBError](#) ()
- void [ThrowCUsbExceptionNetOnError](#) (uint32\_t status)
- bool [GetDeviceCannotStallOutRequests](#) ()
- String ^ [GetHardwareRevision](#) ()
- unsigned int [GetFirmwareVersion](#) (CFirmwareDestinationNet destination)  
*Gets the firmware version for the destination.*
- uint8\_t [GetNumConfigurations](#) ()
- uint8\_t [GetConfiguration](#) ()
- void [SetConfiguration](#) (uint8\_t config)
- uint32\_t [GetDeviceRootHubVendorID](#) ()  
*Gets the Vendor ID of the USB root hub the device is connected to.*
- [UsbVendorIdEnumNet GetDeviceRootHubVendorEnum](#) ()  
*Gets the Vendor ID of the USB root hub the device is connected to.*
- String ^ [GetDeviceRootHubVendorName](#) ()  
*Gets the Vendor Name of the USB root hub the device is connected to.*
- void [EnableExceptions](#) (bool enable)  
*Enables or Disables Exceptions for calls to McsUsb Devices. If Exceptions are disabled, the return value of a command can be queries with the GetStatusOfLastCommand call instead.*
- bool [IsExceptionsEnabled](#) ()
- uint32\_t [GetStatusOfLastCommand](#) ()  
*Gets the status of the last call to the McsUsb Library.*
- uint32\_t [CyclePort](#) ()
- void [AssociateToThis](#) (CMcsUsbNet^ device)

## Static Public Member Functions

- static String ^ [GetErrorText](#) (unsigned int Status)  
*Gets the error text string that belongs to a status number.*

## Static Public Attributes

- static const uint32\_t [Status\\_Crc](#) = (0xE0100001L)
- static const uint32\_t [Status\\_Btstuff](#) = (0xE0100002L)
- static const uint32\_t [Status\\_DataToggleMismatch](#) = (0xE0100003L)
- static const uint32\_t [Status\\_Stall](#) = (0xE0100004L)
- static const uint32\_t [Status\\_DevNotResponding](#) = (0xE0100005L)
- static const uint32\_t [Status\\_PidCheckFailure](#) = (0xE0100006L)
- static const uint32\_t [Status\\_UnexpectedPid](#) = (0xE0100007L)
- static const uint32\_t [Status\\_DataOverrun](#) = (0xE0100008L)
- static const uint32\_t [Status\\_DataUnderrun](#) = (0xE0100009L)
- static const uint32\_t [Status\\_BufferOverrun](#) = (0xE010000CL)
- static const uint32\_t [Status\\_BufferUnderrun](#) = (0xE010000DL)
- static const uint32\_t [Status\\_NotAccessed](#) = (0xE010000FL)
- static const uint32\_t [Status\\_Fifo](#) = (0xE0100010L)
- static const uint32\_t [Status\\_EndpointHalted](#) = (0xE0100030L)
- static const uint32\_t [Status\\_NoMemory](#) = (0xE0100100L)
- static const uint32\_t [Status\\_InvalidUrbFunction](#) = (0xE0100200L)
- static const uint32\_t [Status\\_InvalidParameter](#) = (0xE0100300L)
- static const uint32\_t [Status\\_InvalidDeviceHandle](#) = (0xE0100013L)
- static const uint32\_t [Status\\_InvalidHandle](#) = (0xE0100012L)
- static const uint32\_t [Status\\_ErrorBusy](#) = (0xE0100400L)
- static const uint32\_t [Status\\_RequestFailed](#) = (0xE0100500L)
- static const uint32\_t [Status\\_InvalidPipeHandle](#) = (0xE0100600L)
- static const uint32\_t [Status\\_NoBandwidth](#) = (0xE0100700L)
- static const uint32\_t [Status\\_InternalHcError](#) = (0xE0100800L)
- static const uint32\_t [Status\\_ErrorShortTransfer](#) = (0xE0100900L)
- static const uint32\_t [Status\\_BadStartFrame](#) = (0xE0100A00L)
- static const uint32\_t [Status\\_IsochRequestFailed](#) = (0xE0100B00L)
- static const uint32\_t [Status\\_FrameControlOwned](#) = (0xE0100C00L)
- static const uint32\_t [Status\\_ControlNotOwned](#) = (0xE0100D00L)
- static const uint32\_t [Status\\_Canceled](#) = (0xE0110000L)
- static const uint32\_t [Status\\_Canceling](#) = (0xE0120000L)
- static const uint32\_t [Status\\_AlreadyConfigured](#) = (0xE0110001L)
- static const uint32\_t [Status\\_Unconfigured](#) = (0xE0110002L)
- static const uint32\_t [Status\\_NoSuchDevice](#) = (0xE01F0002L)
- static const uint32\_t [Status\\_DeviceNotFound](#) = (0xE01F0003L)
- static const uint32\_t [Status\\_NotSupported](#) = (0xE01F0005L)
- static const uint32\_t [Status\\_IoPending](#) = (0xE01F0006L)
- static const uint32\_t [Status\\_IoTimeout](#) = (0xE01F0007L)
- static const uint32\_t [Status\\_DeviceRemoved](#) = (0xE01F0008L)
- static const uint32\_t [Status\\_PipeNotLinked](#) = (0xE01F0009L)
- static const uint32\_t [Status\\_ConnectedPipes](#) = (0xE01F000AL)
- static const uint32\_t [Status\\_DeviceLocked](#) = (0xE01F0010L)
- static const uint32\_t [Status\\_RequestMutexTimeout](#) = (0xE01F0020L)
- static const uint32\_t [Status\\_RequestMutexFailed](#) = (0xE01F0021L)
- static const uint32\_t [Status\\_LastUsbErrorMismatch](#) = (0xE01F0022L)
- static const uint32\_t [WPAError\\_ScanningsPending](#) = ( (0xA0220000L) | 0x0036 )

## Properties

- virtual String^ [SerialNumber](#) [get]

### 11.58.1 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.

### 11.58.2 Constructor & Destructor Documentation

#### 11.58.2.1 CMcsUsbNet() [1/2] `CMcsUsbNet ( )`

Initializes a new instance of the base class to handle MCS USB devices.

#### 11.58.2.2 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. |
|----------------|-------------------------------------------|

#### 11.58.2.3 ~CMcsUsbNet() `virtual ~CMcsUsbNet ( ) [virtual]`

#### 11.58.2.4 "!CMcsUsbNet() `!CMcsUsbNet ( )`

### 11.58.3 Member Function Documentation

#### 11.58.3.1 AddSoftwareKey() `void AddSoftwareKey ( String^ key )`

#### 11.58.3.2 AssociateToThis() `void AssociateToThis ( CMcsUsbNet^ device )`



**11.58.3.3 Connect()** [1/4] virtual uint32\_t Connect ( ) [virtual]

Opens a connection to the device.

**Returns**

Error Status. 0 on success.

**11.58.3.4 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.

**11.58.3.5 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.

**11.58.3.6 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.

**11.58.3.7 CyclePort()** `uint32_t CyclePort ( )`

**11.58.3.8 Disconnect()** `virtual void Disconnect ( ) [virtual]`

Disconnect from a device.

**11.58.3.9 EmptyKey()** `bool EmptyKey (`  
    `String^ key )`

**11.58.3.10 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. |
|---------------|----------------------------------------------|

**11.58.3.11 EraseEepromRegisterPreconfig() [1/3]** `void EraseEepromRegisterPreconfig (`  
    `uint32_t EEPROMBase,`  
    `uint32_t DMA_reg )`

**11.58.3.12 EraseEepromRegisterPreconfig() [2/3]** `void EraseEepromRegisterPreconfig (`  
    `uint32_t EEPROMBase,`  
    `uint32_t DMA_reg,`  
    `uint32_t EEPROMSize )`

**11.58.3.13 EraseEepromRegisterPreconfig()** [3/3] void EraseEepromRegisterPreconfig (   
    uint32\_t *EEPROMBase*,   
    uint32\_t *DMA\_reg*,   
    uint32\_t *EEPROMSize*,   
    uint32\_t *EepromStartAddress* )

**11.58.3.14 GetConfiguration()** uint8\_t GetConfiguration ( )

**11.58.3.15 GetDeviceCannotStallOutRequests()** bool GetDeviceCannotStallOutRequests ( )

**11.58.3.16 GetDeviceCapableSpeed()** McsUsbSpeedEnumNet GetDeviceCapableSpeed ( )

**11.58.3.17 GetDeviceEnum()** DeviceEnumNet GetDeviceEnum ( )

**11.58.3.18 GetDeviceId()** DeviceIdNet ^ GetDeviceId ( )

**11.58.3.19 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.

**11.58.3.20 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.

**11.58.3.21 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".

**11.58.3.22 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.

**11.58.3.23 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.

**11.58.3.24 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.

**11.58.3.25 GetHardwareRevision()** `String ^ GetHardwareRevision ( )`

**11.58.3.26 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

**11.58.3.27 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

**11.58.3.28 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

**11.58.3.29 GetLastUSBError()** `unsigned int GetLastUSBError ( )`

**11.58.3.30 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

**11.58.3.31 GetNumConfigurations()** `uint8_t GetNumConfigurations ( )`

**11.58.3.32 GetSerialNumber()** `virtual String ^ GetSerialNumber ( ) [virtual]`

Query the Serial Number of the device.

**Returns**

The Serial Number.

**11.58.3.33 GetSoftwareKey()** `array<BYTE> ^ GetSoftwareKey ( unsigned int index )`

**11.58.3.34 GetSoftwareKeyString()** `[1/2] String ^ GetSoftwareKeyString ( SoftwareKeyProgrammIdsNet::ProgrammIdsNet ProgrammID, uint8_t majorversion )`

**11.58.3.35 GetSoftwareKeyString()** `[2/2] String ^ GetSoftwareKeyString ( uint8_t ProgrammID, uint8_t majorversion )`

**11.58.3.36 GetStatus()** `virtual uint32_t GetStatus ( [System::Runtime::InteropServices::Out] uint32_t% iStatus ) [virtual]`

**11.58.3.37 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.

**11.58.3.38 GetUsbListEntry()** `CMcsUsbListEntryNet ^ GetUsbListEntry ( )`

**11.58.3.39 GetVersion()** [1/2] `DriverVersionNet ^ GetVersion ( )`

**11.58.3.40 GetVersion()** [2/2] `DriverVersionNet ^ GetVersion ( CFirmwareDestinationNet dest )`

**11.58.3.41 HasSoftwareKey()** [1/2] `bool HasSoftwareKey ( SoftwareKeyProgrammIdsNet::ProgrammIdsNet ProgrammID, uint8_t majorversion )`

**11.58.3.42 HasSoftwareKey()** [2/2] `bool HasSoftwareKey ( uint8_t ProgrammID, uint8_t majorversion )`

**11.58.3.43 IsConnected()** `virtual bool IsConnected ( ) [virtual]`

Check if a device is Connected.

**Returns**

true if the device is connected.

**11.58.3.44 IsDeviceHighSpeed()** `bool IsDeviceHighSpeed ( )`

**11.58.3.45 IsDeviceHighSpeedCapable()** `bool IsDeviceHighSpeedCapable ( )`

**11.58.3.46 IsExceptionsEnabled()** `bool IsExceptionsEnabled ( )`

**11.58.3.47 MultibootGetCypressImageId()** `uint32_t MultibootGetCypressImageId ( unsigned int sector )`

Query the multiboot image id of the device located in specified sector (range: 0..9 0..9) of IFB2 Cypress.

**Returns**

The magic ident code of the image.

**11.58.3.48 MultibootGetImageId()** `String ^ MultibootGetImageId ( unsigned int sector )`

Query the multiboot image id of the device located in specified sector (range: 0..2 / 0..9) of IFB1 / IFB2 FPGA.

**Returns**

The magic ident code of the image.

**11.58.3.49 MultibootGetSelectedImage()** `uint32_t MultibootGetSelectedImage ( )`

Gets sector index of selected FPGA boot image on IFB

**Returns**

Sector index of image; range: 0..2

**11.58.3.50 MultibootSelectImage()** `void MultibootSelectImage ( unsigned int sector )`

Select the multiboot image specified by "sector" (range: 0..2) for IFB FPGA.

**Returns**

Throws exception on error.



**11.58.3.51 ReadEepromRegisterPreconfig() [1/3]** `bool ReadEepromRegisterPreconfig (`  
`uint32_t EEPROMBase,`  
`uint32_t DMA_reg,`  
`[System::Runtime::InteropServices::Out] uint32_t% DMA_value )`

**11.58.3.52 ReadEepromRegisterPreconfig() [2/3]** `bool ReadEepromRegisterPreconfig (`  
`uint32_t EEPROMBase,`  
`uint32_t DMA_reg,`  
`[System::Runtime::InteropServices::Out] uint32_t% DMA_value,`  
`uint32_t EEPROMSize )`

**11.58.3.53 ReadEepromRegisterPreconfig() [3/3]** `bool ReadEepromRegisterPreconfig (`  
`uint32_t EEPROMBase,`  
`uint32_t DMA_reg,`  
`[System::Runtime::InteropServices::Out] uint32_t% DMA_value,`  
`uint32_t EEPROMSize,`  
`uint32_t EepromStartAddress )`

**11.58.3.54 ReadRegister() [1/2]** `unsigned int ReadRegister (`  
`unsigned int reg )`

**11.58.3.55 ReadRegister() [2/2]** `array<uint32_t> ^ ReadRegister (`  
`unsigned int reg,`  
`int length )`

**11.58.3.56 ReadRegister32()** `unsigned int ReadRegister32 (`  
`unsigned int adr )`

**11.58.3.57 ReadRegisterTimeSlot()** `unsigned int ReadRegisterTimeSlot (`  
`unsigned int reg,`  
`int TimeSlot )`

**11.58.3.58 RemoveSoftwareKey()** `void RemoveSoftwareKey (`  
`unsigned int index )`

**11.58.3.59 RescanHeadstage()** `void RescanHeadstage (`  
`uint32_t headstage )`

rescans and activates a headstage

## Parameters

|    |                  |                               |
|----|------------------|-------------------------------|
| in | <i>headstage</i> | the headstage number (0 or 1) |
|----|------------------|-------------------------------|

**11.58.3.60 SetConfiguration()** void SetConfiguration (   
     uint8\_t *config* )

**11.58.3.61 SetSoftwareKey()** void SetSoftwareKey (   
     unsigned int *index*,   
     array< BYTE >^ *buffer* )

**11.58.3.62 ThrowCUsbExceptionNetOnError()** void ThrowCUsbExceptionNetOnError (   
     uint32\_t *status* )

**11.58.3.63 TxnGetSerialNumber()** unsigned int TxnGetSerialNumber ( )

**11.58.3.64 TxnSetSerialNumber()** void TxnSetSerialNumber (   
     unsigned int *number* )

**11.58.3.65 TxnTestMemoryReadAndCheck()** unsigned int TxnTestMemoryReadAndCheck (   
     unsigned short *index* )

**11.58.3.66 TxnTestMemoryWrite()** unsigned int TxnTestMemoryWrite (   
     unsigned short *index* )

**11.58.3.67 ValidKey()** [1/2] bool ValidKey (   
     String^ *key*,   
     [System::Runtime::InteropServices::Out] String^% *serial\_number* )

**11.58.3.68 ValidKey()** [2/2] bool ValidKey (   
    String^ key,   
    uint8\_t ProgrammID,   
    uint8\_t majorversion,   
    [System::Runtime::InteropServices::Out] String^% serial\_number )

**11.58.3.69 WriteEepromRegisterPreconfig()** [1/3] void WriteEepromRegisterPreconfig (   
    uint32\_t EEPROMBase,   
    uint32\_t DMA\_reg,   
    uint32\_t DMA\_value )

**11.58.3.70 WriteEepromRegisterPreconfig()** [2/3] void WriteEepromRegisterPreconfig (   
    uint32\_t EEPROMBase,   
    uint32\_t DMA\_reg,   
    uint32\_t DMA\_value,   
    uint32\_t EEPROMSize )

**11.58.3.71 WriteEepromRegisterPreconfig()** [3/3] void WriteEepromRegisterPreconfig (   
    uint32\_t EEPROMBase,   
    uint32\_t DMA\_reg,   
    uint32\_t DMA\_value,   
    uint32\_t EEPROMSize,   
    uint32\_t EepromStartAddress )

**11.58.3.72 WriteRegister()** [1/2] void WriteRegister (   
    unsigned int reg,   
    array< unsigned int >^ values )

**11.58.3.73 WriteRegister()** [2/2] void WriteRegister (   
    unsigned int reg,   
    unsigned int value )

**11.58.3.74 WriteRegister32()** void WriteRegister32 (   
    unsigned int adr,   
    unsigned int value )

**11.58.3.75 WriteRegisterArray()** `void WriteRegisterArray (`  
    `unsigned int reg,`  
    `array< unsigned int >^ values )`

**11.58.3.76 WriteRegisterTimeSlot()** [1/2] `void WriteRegisterTimeSlot (`  
    `unsigned int reg,`  
    `array< unsigned int >^ values,`  
    `int TimeSlot )`

**11.58.3.77 WriteRegisterTimeSlot()** [2/2] `void WriteRegisterTimeSlot (`  
    `unsigned int reg,`  
    `unsigned int value,`  
    `int TimeSlot )`

**11.58.3.78 WriteRegisterValue()** `void WriteRegisterValue (`  
    `unsigned int reg,`  
    `unsigned int value )`

#### 11.58.4 Member Data Documentation

**11.58.4.1 Status\_AlreadyConfigured** `const uint32_t Status_AlreadyConfigured = (0xE0110001L)`  
[static]

**11.58.4.2 Status\_BadStartFrame** `const uint32_t Status_BadStartFrame = (0xE0100A00L)` [static]

**11.58.4.3 Status\_Btstuff** `const uint32_t Status_Btstuff = (0xE0100002L)` [static]

**11.58.4.4 Status\_BufferOverrun** `const uint32_t Status_BufferOverrun = (0xE010000CL)` [static]

**11.58.4.5 Status\_BufferUnderrun** `const uint32_t Status_BufferUnderrun = (0xE010000DL)` [static]

**11.58.4.6 Status\_Canceled** `const uint32_t Status_Canceled = (0xE0110000L) [static]`

**11.58.4.7 Status\_Canceling** `const uint32_t Status_Canceling = (0xE0120000L) [static]`

**11.58.4.8 Status\_ConnectedPipes** `const uint32_t Status_ConnectedPipes = (0xE01F000AL) [static]`

**11.58.4.9 Status\_ControlNotOwned** `const uint32_t Status_ControlNotOwned = (0xE0100D00L) [static]`

**11.58.4.10 Status\_Crc** `const uint32_t Status_Crc = (0xE0100001L) [static]`

**11.58.4.11 Status\_DataOverrun** `const uint32_t Status_DataOverrun = (0xE0100008L) [static]`

**11.58.4.12 Status\_DataToggleMismatch** `const uint32_t Status_DataToggleMismatch = (0xE0100003L) [static]`

**11.58.4.13 Status\_DataUnderrun** `const uint32_t Status_DataUnderrun = (0xE0100009L) [static]`

**11.58.4.14 Status\_DeviceLocked** `const uint32_t Status_DeviceLocked = (0xE01F0010L) [static]`

**11.58.4.15 Status\_DeviceNotFound** `const uint32_t Status_DeviceNotFound = (0xE01F0003L) [static]`

**11.58.4.16 Status\_DeviceRemoved** `const uint32_t Status_DeviceRemoved = (0xE01F0008L) [static]`

**11.58.4.17 Status\_DevNotResponding** `const uint32_t Status_DevNotResponding = (0xE0100005L)`  
[static]

**11.58.4.18 Status\_EndpointHalted** `const uint32_t Status_EndpointHalted = (0xE0100030L)` [static]

**11.58.4.19 Status\_ErrorBusy** `const uint32_t Status_ErrorBusy = (0xE0100400L)` [static]

**11.58.4.20 Status\_ErrorShortTransfer** `const uint32_t Status_ErrorShortTransfer = (0xE0100900L)`  
[static]

**11.58.4.21 Status\_Fifo** `const uint32_t Status_Fifo = (0xE0100010L)` [static]

**11.58.4.22 Status\_FrameControlOwned** `const uint32_t Status_FrameControlOwned = (0xE0100C00L)`  
[static]

**11.58.4.23 Status\_InternalHcError** `const uint32_t Status_InternalHcError = (0xE0100800L)` [static]

**11.58.4.24 Status\_InvalidDeviceHandle** `const uint32_t Status_InvalidDeviceHandle = (0xE0100013L)`  
[static]

**11.58.4.25 Status\_InvalidHandle** `const uint32_t Status_InvalidHandle = (0xE0100012L)` [static]

**11.58.4.26 Status\_InvalidParameter** `const uint32_t Status_InvalidParameter = (0xE0100300L)` [static]

**11.58.4.27 Status\_InvalidPipeHandle** `const uint32_t Status_InvalidPipeHandle = (0xE0100600L)`  
[static]

**11.58.4.28 Status\_InvalidUrbFunction** `const uint32_t Status_InvalidUrbFunction = (0xE0100200L)`  
[static]

**11.58.4.29 Status\_IoPending** `const uint32_t Status_IoPending = (0xE01F0006L)` [static]

**11.58.4.30 Status\_IoTimeout** `const uint32_t Status_IoTimeout = (0xE01F0007L)` [static]

**11.58.4.31 Status\_IsochRequestFailed** `const uint32_t Status_IsochRequestFailed = (0xE0100B00L)`  
[static]

**11.58.4.32 Status\_LastUsbErrorMismatch** `const uint32_t Status_LastUsbErrorMismatch = (0xE01F0022L)` [static]

**11.58.4.33 Status\_NoBandwidth** `const uint32_t Status_NoBandwidth = (0xE0100700L)` [static]

**11.58.4.34 Status\_NoMemory** `const uint32_t Status_NoMemory = (0xE0100100L)` [static]

**11.58.4.35 Status\_NoSuchDevice** `const uint32_t Status_NoSuchDevice = (0xE01F0002L)` [static]

**11.58.4.36 Status\_NotAccessed** `const uint32_t Status_NotAccessed = (0xE010000FL)` [static]

**11.58.4.37 Status\_NotSupported** `const uint32_t Status_NotSupported = (0xE01F0005L)` [static]

**11.58.4.38 Status\_PidCheckFailure** `const uint32_t Status_PidCheckFailure = (0xE0100006L) [static]`

**11.58.4.39 Status\_PipeNotLinked** `const uint32_t Status_PipeNotLinked = (0xE01F0009L) [static]`

**11.58.4.40 Status\_RequestFailed** `const uint32_t Status_RequestFailed = (0xE0100500L) [static]`

**11.58.4.41 Status\_RequestMutexFailed** `const uint32_t Status_RequestMutexFailed = (0xE01F0021L) [static]`

**11.58.4.42 Status\_RequestMutexTimeout** `const uint32_t Status_RequestMutexTimeout = (0xE01↔F0020L) [static]`

**11.58.4.43 Status\_Stall** `const uint32_t Status_Stall = (0xE0100004L) [static]`

**11.58.4.44 Status\_Unconfigured** `const uint32_t Status_Unconfigured = (0xE0110002L) [static]`

**11.58.4.45 Status\_UnexpectedPid** `const uint32_t Status_UnexpectedPid = (0xE0100007L) [static]`

**11.58.4.46 WPAError\_ScanningIsPending** `const uint32_t WPAError_ScanningIsPending = ( (0x↔A0220000L) | 0x0036 ) [static]`

## **11.58.5 Property Documentation**

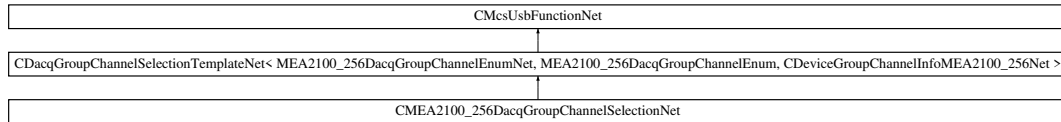
**11.58.5.1 SerialNumber** `virtual String^ SerialNumber [get]`



## 11.59 CMcsUsbPointerContainer Class Reference

### 11.60 CMEA2100\_256DacqGroupChannelSelectionNet Class Reference

Inheritance diagram for CMEA2100\_256DacqGroupChannelSelectionNet:



#### Public Member Functions

- [CMEA2100\\_256DacqGroupChannelSelectionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)

#### Additional Inherited Members

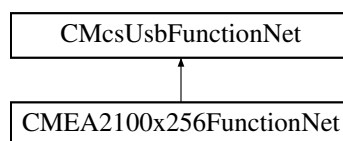
#### 11.60.1 Constructor & Destructor Documentation

**11.60.1.1 CMEA2100\_256DacqGroupChannelSelectionNet()** [CMEA2100\\_256DacqGroupChannelSelectionNet](#)  
(  
    [CMcsUsbNet](#)<sup>^</sup> mcsusb )

### 11.61 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> pMEA2100x256↔  
FunctionPointerContainer)

*Initializes a new instance of the [CMEA2100x256FunctionNet](#) class.*

- [CMEA2100x256FunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- virtual [~CMEA2100x256FunctionNet](#) ()
- [!CMEA2100x256FunctionNet](#) ()
- [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::GetNumberOfStimulat](#)*

- 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::GetNumberOfStimulat](#)*

## Additional Inherited Members

### 11.61.1 Detailed Description

[CMEA2100x256FunctionNet](#) is the class to control the MEA2100-256 device needs #include "Stg200xNet.h" to resolve documentation reference

### 11.61.2 Constructor & Destructor Documentation

**11.61.2.1 CMEA2100x256FunctionNet()** [1/2] [CMEA2100x256FunctionNet](#) (   
 [CMcsUsbNet](#)<sup>^</sup> *mcsusb*,   
 [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> *pMEA2100x256FunctionPointerContainer* )

Initializes a new instance of the [CMEA2100x256FunctionNet](#) class.

**11.61.2.2 CMEA2100x256FunctionNet()** [2/2] [CMEA2100x256FunctionNet](#) (   
 [CMcsUsbNet](#)<sup>^</sup> *mcsusb* )

**11.61.2.3 ~CMEA2100x256FunctionNet()** [virtual ~CMEA2100x256FunctionNet](#) ( ) [virtual]

**11.61.2.4 !"CMEA2100x256FunctionNet()** [!CMEA2100x256FunctionNet](#) ( )

### 11.61.3 Member Function Documentation

**11.61.3.1 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.

**11.61.3.2 SetLayoutConfiguration()** [void SetLayoutConfiguration](#) (   
 [StimulationLayoutConfigurationEnumNet](#) *LayoutConfiguration* )

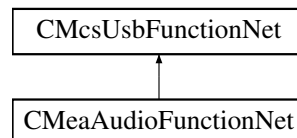
Sets the stimulation layout configuration. Can be single well, 6-well or 9-well. The number of DAC channels available per well is [Mcs::Usb::CStg200xBasicNet::GetNumberOfAnalogChannels](#) divided by [Mcs::Usb::CStg200xBasicNet::GetNumberOfStimulationSourcesPerElectrode](#).

## Parameters

|                            |                                           |
|----------------------------|-------------------------------------------|
| <i>LayoutConfiguration</i> | The new stimulation layout configuration. |
|----------------------------|-------------------------------------------|

## 11.62 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

## 11.62.1 Constructor &amp; Destructor Documentation

- 11.62.1.1 CMeaAudioFunctionNet()** [1/2] [CMeaAudioFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb,  
[CMcsUsbFunctionPointerContainer](#)<sup>^</sup> meaAudioFunctionPointerContainer )

**11.62.1.2 CMeaAudioFunctionNet()** [2/2] `CMeaAudioFunctionNet ( CMcsUsbNet^ mcsusb )`

## 11.62.2 Member Function Documentation

**11.62.2.1 GetAudioChannels()** [1/2] `virtual uint32_t GetAudioChannels ( [System::Runtime::InteropServices::Out] array< s_setaudionet^>^% channels ) [virtual]`

Gets the electrode to monitor and amplification for the audio channels.

### Parameters

|                 |                                                                            |
|-----------------|----------------------------------------------------------------------------|
| <i>channels</i> | Struct which contains the electrode (channel) and amplification on return. |
|-----------------|----------------------------------------------------------------------------|

### Returns

Error Status. 0 on success.

**11.62.2.2 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.

**11.62.2.3 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.

**11.62.2.4 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.

**11.62.2.5 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. |
|----------------------|------------------------|

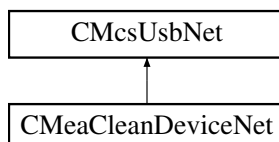
## Returns

Error Status. 0 on success.

## 11.63 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

### 11.63.1 Detailed Description

[CMeaCleanDeviceNet](#) is the class to access the MEA Clean device.

### 11.63.2 Constructor & Destructor Documentation

#### 11.63.2.1 CMeaCleanDeviceNet() [CMeaCleanDeviceNet](#) ( )

Initializes a new instance of the [CMeaCleanDeviceNet](#) class.

#### 11.63.2.2 ~CMeaCleanDeviceNet() [virtual](#) [~CMeaCleanDeviceNet](#) ( ) [\[virtual\]](#)

#### 11.63.2.3 ~!CMeaCleanDeviceNet() [!CMeaCleanDeviceNet](#) ( )

### 11.63.3 Member Function Documentation

#### 11.63.3.1 GetCycle() [int32\\_t](#) [GetCycle](#) ( )

Gets the current cycle.

##### Returns

The cycle number.

#### 11.63.3.2 GetCycles() [uint32\\_t](#) [GetCycles](#) ( )

Gets the number of cycles.

##### Returns

The number of cycles to run for.

**11.63.3.3 GetMaxVoltage()** `int32_t GetMaxVoltage ( )`

Gets the upper voltage level

**Returns**

The upper voltage level in mV.

**11.63.3.4 GetMinVoltage()** `int32_t GetMinVoltage ( )`

Gets the lower voltage level.

**Returns**

The lower voltage level in mV.

**11.63.3.5 GetOutputVoltage()** `int32_t GetOutputVoltage ( )`

Gets the output voltage.

**Returns**

The output voltage in mV.

**11.63.3.6 GetSlope()** `uint32_t GetSlope ( )`

Gets the voltage slope.

**Returns**

The voltage slope in mV/s.

**11.63.3.7 IsRunning()** `bool IsRunning ( )`

Gets if the MEA Clean device is running.

**Returns**

"true" when a run is in progress, otherwise "false".

**11.63.3.8 SetCycles()** `void SetCycles (   
                  uint32_t cycles )`

Sets the number of cycles.



## Parameters

|               |                                            |
|---------------|--------------------------------------------|
| <i>cycles</i> | The number of cycles to run for (0 .. 99). |
|---------------|--------------------------------------------|

**11.63.3.9 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). |
|-------------------|------------------------------------------------|

**11.63.3.10 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). |
|-------------------|------------------------------------------------|

**11.63.3.11 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). |
|---------------------|------------------------------------------------|

**11.63.3.12 Start()** `void Start ( )`

Starts a MEA Clean run.

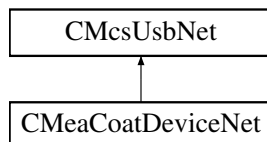
**11.63.3.13 Stop()** `void Stop ( )`

Stops a MEA Clean run.

## 11.64 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

### 11.64.1 Detailed Description

[CMeaCoatDeviceNet](#) is the class to access the MEA Coat device.

### 11.64.2 Constructor & Destructor Documentation

#### 11.64.2.1 CMeaCoatDeviceNet() [CMeaCoatDeviceNet](#) ( )

Initializes a new instance of the [CMeaCoatDeviceNet](#) class.

#### 11.64.2.2 ~CMeaCoatDeviceNet() [virtual](#) [~CMeaCoatDeviceNet](#) ( ) [\[virtual\]](#)

#### 11.64.2.3 "!CMeaCoatDeviceNet() [!CMeaCoatDeviceNet](#) ( )

### 11.64.3 Member Function Documentation

#### 11.64.3.1 GetCurrentCycle() [int32\\_t](#) [GetCurrentCycle](#) ( )

Gets the current cycle.

#### Returns

The cycle number.

**11.64.3.2 GetCycles()** `uint32_t GetCycles ( )`

Gets the number of cycles.

**Returns**

The number of cycles to run for.

**11.64.3.3 GetDuration()** `uint32_t GetDuration ( )`

Gets the duration of a MEA Coat run.

**Returns**

The duration in ms.

**11.64.3.4 GetMaxCurrent()** `uint32_t GetMaxCurrent ( )`

Gets the limit of the current ramp (absolute value).

**Returns**

The limit of the current ramp in pA (absolute value).

**11.64.3.5 GetOffsetCurrent()** `int32_t GetOffsetCurrent ( )`

Gets the offset of the current.

**Returns**

The offset of the current in pA.

**11.64.3.6 GetOutputCurrent()** `int32_t GetOutputCurrent ( )`

Gets the output current.

**Returns**

The output current in pA.

**11.64.3.7 GetPauseDuration()** `uint32_t GetPauseDuration ( )`

Gets the duration of the pause between MEA Coat pulses.

**Returns**

The duration in ms.

**11.64.3.8 GetSlope()** `int32_t GetSlope ( )`

Gets the current slope.

**Returns**

The current slope in pA/s.

**11.64.3.9 GetTimeInPause()** `int32_t GetTimeInPause ( )`

Gets the time in the pause.

**Returns**

The time in the pause in ms.

**11.64.3.10 GetTimeInPlateau()** `int32_t GetTimeInPlateau ( )`

Gets the time in the plateau.

**Returns**

The time in the plateau in ms.

**11.64.3.11 IsRunning()** `bool IsRunning ( )`

Gets if the MEA Clean device is running.

**Returns**

"true" when a run is in progress, otherwise "false".

**11.64.3.12 SetCycles()** `void SetCycles (   
uint32_t cycles )`

Sets the number of cycles.

**Parameters**

|               |                                            |
|---------------|--------------------------------------------|
| <i>cycles</i> | The number of cycles to run for (0 .. 99). |
|---------------|--------------------------------------------|

**11.64.3.13 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). |
|-----------------|---------------------------------------|

**11.64.3.14 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). |
|-------------------|-------------------------------------------------------------------|

**11.64.3.15 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). |
|----------------------|-------------------------------------------------|

**11.64.3.16 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). |
|----------------------|---------------------------------------|

**11.64.3.17 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). |
|---------------------|---------------------------------------------------|

**11.64.3.18 Start()** `void Start ( )`

Starts a MEA Coat run.

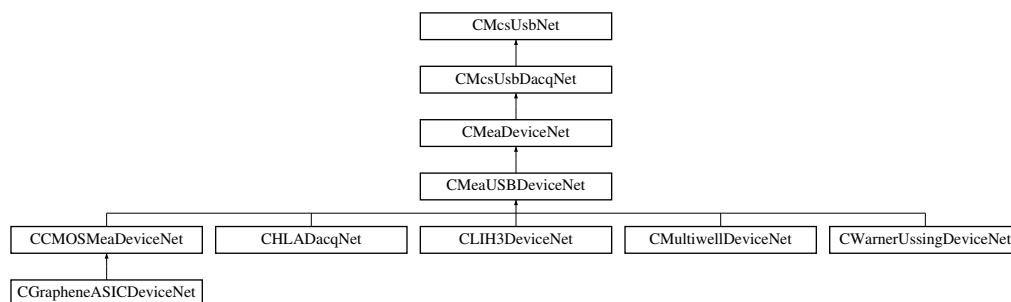
**11.64.3.19 Stop()** `void Stop ( )`

Stops a MEA Coat run.

## 11.65 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

### 11.65.1 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



## 11.65.2 Constructor & Destructor Documentation

### 11.65.2.1 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. |
|----------------|-------------------------------------------|

### 11.65.2.2 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. |
|--------------|------------------------------------------|

### 11.65.2.3 ~CMeaDeviceNet() ~CMeaDeviceNet ( )

## 11.65.3 Member Function Documentation

**11.65.3.1 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.

**11.65.3.2 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.

**11.65.3.3 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.

**11.65.3.4 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.

**11.65.3.5 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.

**11.65.3.6 GetEnumerationSpeed()** `int32_t GetEnumerationSpeed ( )`

**11.65.3.7 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.

**11.65.3.8 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. |
|--------------------|--------------------|

**11.65.3.9 SetNumberOfAnalogChannels()** `virtual uint32_t SetNumberOfAnalogChannels (`  
    `unsigned int NumberOfChannels_HS1,`  
    `unsigned int NumberOfChannels_HS2,`  
    `unsigned int NumberOfChannels_DSP,`  
    `unsigned int NumberOfChannels_IF,`  
    `unsigned int virtualDevice ) [virtual]`

Sets the number of analog channels in the datastream for the MEA2100 device.

**Parameters**

|                             |                                                 |
|-----------------------------|-------------------------------------------------|
| <i>NumberOfChannels_HS1</i> | Number of analog channels from the Headstage 1. |
|-----------------------------|-------------------------------------------------|

**Parameters**

|                             |                                                 |
|-----------------------------|-------------------------------------------------|
| <i>NumberOfChannels_HS2</i> | Number of analog channels from the Headstage 2. |
|-----------------------------|-------------------------------------------------|

**Parameters**

|                             |                                    |
|-----------------------------|------------------------------------|
| <i>NumberOfChannels_DSP</i> | Number of data words from the DSP. |
|-----------------------------|------------------------------------|

**Parameters**

|                              |                                                    |
|------------------------------|----------------------------------------------------|
| <i>NumberOfChannels_↔_IF</i> | Number of analog channels from the Interfaceboard. |
|------------------------------|----------------------------------------------------|

**Parameters**

|                      |                       |
|----------------------|-----------------------|
| <i>virtualDevice</i> | virtualDevice to use. |
|----------------------|-----------------------|

**Returns**

Error Status. 0 on success.

**11.65.3.10 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.

**11.65.3.11 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.

**11.65.3.12 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.

**11.65.3.13 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.

## 11.65.4 Property Documentation

**11.65.4.1 AnalogGain** `virtual int AnalogGain [get]`

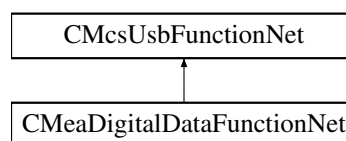
The gain of the analog inputs of the device. Value is gain times 1000, a value of 1000 corresponds to a gain of 1.0.

**11.65.4.2 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.

**11.65.4.3 MeaAudioFunctionNet** `CMeaAudioFunctionNet^ MeaAudioFunctionNet [get]`**11.65.4.4 MeaDigitalDataFunctionNet** `CMeaDigitalDataFunctionNet^ MeaDigitalDataFunctionNet [get]`**11.65.4.5 MeaFeedbackFunctionNet** `CMeaFeedbackFunctionNet^ MeaFeedbackFunctionNet [get]`**11.65.4.6 MeFunctionNet** `CMeFunctionNet^ MeFunctionNet [get]`**11.65.4.7 W2100\_FunctionNet** `CW2100_FunctionNet^ W2100_FunctionNet [get]`**11.65.4.8 WClassicFunctionNet** `CWClassicFunctionNet^ WClassicFunctionNet [get]`**11.66 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

### 11.66.1 Constructor & Destructor Documentation

**11.66.1.1 CMeaDigitalDataFunctionNet()** [1/2] `CMeaDigitalDataFunctionNet (   
CMcsUsbNet^ mcsusb,   
CMcsUsbFunctionPointerContainer^ meaDigitalFunctionPointerContainer )`

**11.66.1.2 CMeaDigitalDataFunctionNet()** [2/2] `CMeaDigitalDataFunctionNet (   
CMcsUsbNet^ mcsusb )`

### 11.66.2 Member Function Documentation

**11.66.2.1 GetDigitalData()** `unsigned int GetDigitalData ( )`

Get the value of the digital output.

#### Returns

Value on the digital data register.

**11.66.2.2 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. |
|--------------|------------|



**11.66.2.3 SetDigitalData()** [2/2] void SetDigitalData (  
     unsigned int *digital\_value*,  
     unsigned int *digital\_value\_mask* )

Generate a value on the digital output.

Parameters

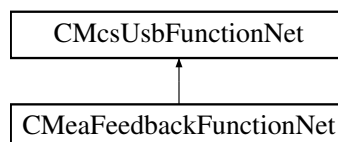
|                      |               |
|----------------------|---------------|
| <i>digital_value</i> | Value to set. |
|----------------------|---------------|

Parameters

|                           |                  |
|---------------------------|------------------|
| <i>digital_value_mask</i> | Mask for change. |
|---------------------------|------------------|

## 11.67 CMeaFeedbackFunctionNet Class Reference

Inheritance diagram for CMeaFeedbackFunctionNet:



### Public Member Functions

- [CMeaFeedbackFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> meaFeedback↔  
FunctionNet)
- [CMeaFeedbackFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- void [FeedbackSetFeedback](#) (unsigned char on, unsigned short digoutmask, unsigned short diginmask)
- unsigned int [FeedbackGetSampleTimerCount](#) ([System::Runtime::InteropServices::Out]unsigned int%  
CurrentCount, [System::Runtime::InteropServices::Out]unsigned int% LastKnownCount, [System::Runtime↔  
::InteropServices::Out]bool% On)
- void [FeedbackSetDigitalMapping](#) (unsigned short channel, unsigned short outmapping, unsigned short in-  
mapping)
- void [FeedbackSetFilterParameter](#) (unsigned char filter, array< short ><sup>^</sup> parameters)
- void [FeedbackSetFilterParameter32](#) (unsigned char filter, array< int ><sup>^</sup> parameters)
- void [FeedbackSetIIRFilterParameter](#) (unsigned char filter, int length, array< double ><sup>^</sup> parameters)
- void [FeedbackSetMkFilter](#) (unsigned char filter, String<sup>^</sup> filtertype, double cheb\_ripple, String<sup>^</sup> passtype, int  
order, double alpha1, double alpha2)
- void [FeedbackSetChannelFilter](#) (short channel, char filter)
- void [FeedbackSetGlobalChannelFilter](#) (char filter, unsigned short firstchannel, unsigned short lastchannel)

- void [FeedbackSetFilterOff](#) ()
- void [FeedbackSetNumberOfSpikeDetectors](#) (unsigned short number)
- void [FeedbackSetSpikeDetectorThreshold](#) (unsigned short position, unsigned short sourcechannel, unsigned short resultchannel, unsigned short trigger, unsigned short totzeit, int threshold1, int threshold2, short slope)
- void [FeedbackSetNumberOfRateCounter](#) (unsigned short number)
- void [FeedbackSetRateCounter](#) (unsigned short position, unsigned short sourcechannel, unsigned short 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)

## Additional Inherited Members

### 11.67.1 Constructor & Destructor Documentation

**11.67.1.1 CMeaFeedbackFunctionNet()** [1/2] [CMeaFeedbackFunctionNet](#) (   
[CMcsUsbNet](#)^ *mcsusb*,   
[CMcsUsbFunctionPointerContainer](#)^ *meaFeedbackFunctionNet* )

**11.67.1.2 CMeaFeedbackFunctionNet()** [2/2] [CMeaFeedbackFunctionNet](#) (   
[CMcsUsbNet](#)^ *mcsusb* )

### 11.67.2 Member Function Documentation

**11.67.2.1 FeedbackGetSampleTimerCount()** unsigned int [FeedbackGetSampleTimerCount](#) (   
[System::Runtime::InteropServices::Out] unsigned int% *CurrentCount*,   
[System::Runtime::InteropServices::Out] unsigned int% *LastKnownCount*,   
[System::Runtime::InteropServices::Out] bool% *On* )

**11.67.2.2 FeedbackSetAnalogSource()** void [FeedbackSetAnalogSource](#) (   
[AnalogSourceEnumNet](#) *AnalogSource*,   
unsigned int *Channels*,   
unsigned int *Offset* )

**11.67.2.3 FeedbackSetChannelFilter()** void FeedbackSetChannelFilter (   
    short *channel*,  
    char *filter* )

**11.67.2.4 FeedbackSetDigitalMapping()** void FeedbackSetDigitalMapping (   
    unsigned short *channel*,  
    unsigned short *outmapping*,  
    unsigned short *inmapping* )

**11.67.2.5 FeedbackSetFeedback()** void FeedbackSetFeedback (   
    unsigned char *on*,  
    unsigned short *digoutmask*,  
    unsigned short *diginmask* )

**11.67.2.6 FeedbackSetFilterOff()** void FeedbackSetFilterOff ( )

**11.67.2.7 FeedbackSetFilterParameter()** void FeedbackSetFilterParameter (   
    unsigned char *filter*,  
    array< short >^ *parameters* )

**11.67.2.8 FeedbackSetFilterParameter32()** void FeedbackSetFilterParameter32 (   
    unsigned char *filter*,  
    array< int >^ *parameters* )

**11.67.2.9 FeedbackSetGlobalChannelFilter()** void FeedbackSetGlobalChannelFilter (   
    char *filter*,  
    unsigned short *firstchannel*,  
    unsigned short *lastchannel* )

**11.67.2.10 FeedbackSetIIRFilterParameter()** void FeedbackSetIIRFilterParameter (   
    unsigned char *filter*,  
    int *length*,  
    array< double >^ *parameters* )

**11.67.2.11 FeedbackSetLogic()** `void FeedbackSetLogic (`  
    `unsigned short position,`  
    `array< unsigned short >^ sourcechannel,`  
    `unsigned short resultchannel,`  
    `unsigned int lookup )`

**11.67.2.12 FeedbackSetMkFilter()** `void FeedbackSetMkFilter (`  
    `unsigned char filter,`  
    `String^ filtertype,`  
    `double cheb_ripple,`  
    `String^ passtype,`  
    `int order,`  
    `double alpha1,`  
    `double alpha2 )`

**11.67.2.13 FeedbackSetNumberOfLogics()** `void FeedbackSetNumberOfLogics (`  
    `unsigned short number )`

**11.67.2.14 FeedbackSetNumberOfRateCounter()** `void FeedbackSetNumberOfRateCounter (`  
    `unsigned short number )`

**11.67.2.15 FeedbackSetNumberOfRateDetectors()** `void FeedbackSetNumberOfRateDetectors (`  
    `unsigned short number )`

**11.67.2.16 FeedbackSetNumberOfSpikeDetectors()** `void FeedbackSetNumberOfSpikeDetectors (`  
    `unsigned short number )`

**11.67.2.17 FeedbackSetNumberOfTriggers()** `void FeedbackSetNumberOfTriggers (`  
    `unsigned short number )`

**11.67.2.18 FeedbackSetRateCounter()** `void FeedbackSetRateCounter (`  
    `unsigned short position,`  
    `unsigned short sourcechannel,`  
    `unsigned short resultchannel )`

**11.67.2.19 FeedbackSetRateDetector()** void FeedbackSetRateDetector (

```

 unsigned short position,
 unsigned short resultchannel,
 unsigned short trigger,
 unsigned short totzeit,
 unsigned short pulses,
 unsigned int duration1,
 unsigned int duration2)

```

**11.67.2.20 FeedbackSetSpikeDetectorThreshold()** void FeedbackSetSpikeDetectorThreshold (

```

 unsigned short position,
 unsigned short sourcechannel,
 unsigned short resultchannel,
 unsigned short trigger,
 unsigned short totzeit,
 int threshold1,
 int threshold2,
 short slope)

```

**11.67.2.21 FeedbackSetTrigger()** void FeedbackSetTrigger (

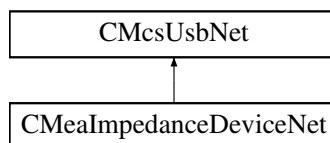
```

 unsigned short position,
 unsigned short sourcechannel,
 unsigned short resultchannel,
 unsigned short trigger,
 unsigned short totzeit)

```

## 11.68 CMealImpedanceDeviceNet Class Reference

Inheritance diagram for CMealImpedanceDeviceNet:



### Public Member Functions

- [CMealImpedanceDeviceNet](#) ()
- [~CMealImpedanceDeviceNet](#) ()
- virtual void [StartMeasurement](#) (unsigned short channel)
- virtual unsigned short [GetReady](#) ()
- virtual unsigned short [GetArraySize](#) ()
- virtual array< unsigned short > ^ [GetResult](#) ()
- unsigned short [GetAdapterCode](#) ()
- virtual unsigned int [GetImpedanceTestFrequency](#) ()
- virtual void [SetImpedanceTestFrequency](#) (unsigned int TestFrequency\_Hertz)

## Additional Inherited Members

### 11.68.1 Constructor & Destructor Documentation

**11.68.1.1 CMeaImpedanceDeviceNet()** `CMeaImpedanceDeviceNet ( )`

**11.68.1.2 ~CMeaImpedanceDeviceNet()** `~CMeaImpedanceDeviceNet ( )`

### 11.68.2 Member Function Documentation

**11.68.2.1 GetAdapterCode()** `unsigned short GetAdapterCode ( )`

**11.68.2.2 GetArraySize()** `virtual unsigned short GetArraySize ( ) [virtual]`

**11.68.2.3 GetImpedanceTestFrequency()** `virtual unsigned int GetImpedanceTestFrequency ( ) [virtual]`

**11.68.2.4 GetReady()** `virtual unsigned short GetReady ( ) [virtual]`

**11.68.2.5 GetResult()** `virtual array<unsigned short> ^ GetResult ( ) [virtual]`

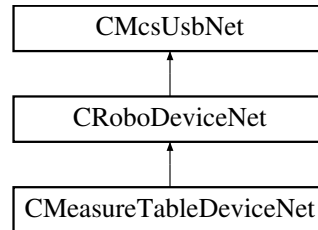
**11.68.2.6 SetImpedanceTestFrequency()** `virtual void SetImpedanceTestFrequency ( unsigned int TestFrequency_Hertz ) [virtual]`

**11.68.2.7 StartMeasurement()** `virtual void StartMeasurement ( unsigned short channel ) [virtual]`

## 11.69 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

#### 11.69.1 Detailed Description

[CMeasureTableDeviceNet](#) is the to control the MCS HLA device

#### 11.69.2 Constructor & Destructor Documentation

**11.69.2.1 CMeasureTableDeviceNet()** [CMeasureTableDeviceNet](#) (  
void )

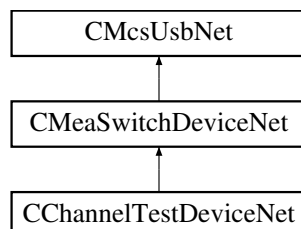
#### 11.69.3 Property Documentation

**11.69.3.1 Sensor** [CMcsBus\\_SensorNet](#)<sup>^</sup> [Sensor](#) [get]

## 11.70 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

#### 11.70.1 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.

#### 11.70.2 Constructor & Destructor Documentation

##### 11.70.2.1 CMeaSwitchDeviceNet() [CMeaSwitchDeviceNet](#) ( )

Constructor.



**11.70.2.2** `~CMeaSwitchDeviceNet()` `~CMeaSwitchDeviceNet ( )`

Destructor.

**11.70.3 Member Function Documentation****11.70.3.1** `GetNumber()` `unsigned short GetNumber ( )`

Gets the number of boards in the device.

The MEA-Switch are delivered with 64 or 128 channels

**11.70.3.2** `GetPattern()` `array<unsigned char> ^ GetPattern ( )`

Gets the pattern of the switches that are currently set in the device as char array.

**11.70.3.3** `GetPatternBool()` `array<bool> ^ GetPatternBool ( )`

Gets the pattern of the switches that are currently set in the device as bools.

**11.70.3.4** `SetPattern()` `void SetPattern ( array< unsigned char >^ pattern )`

Sets the pattern of switches from a char array.

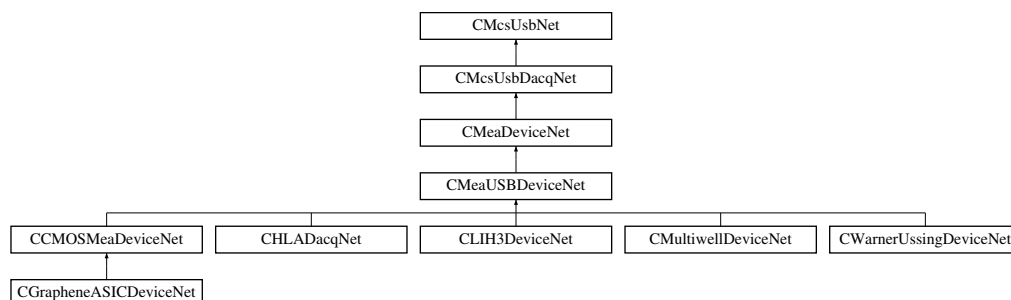
**11.70.3.5** `SetPatternBool()` `void SetPatternBool ( array< bool >^ pattern )`

Sets the pattern of switches from a.

**11.71 CMeaUSBDeviceNet Class Reference**

Class for data acquisition via ME and MEA USB amplifiers

Inheritance diagram for CMeaUSBDeviceNet:



## Public Member Functions

- [CMeaUSBDeviceNet](#) ([OnChannelData](#)<sup>^</sup> channelData, [OnError](#)<sup>^</sup> error)  
*Initializes a new instance of [CMeaDeviceNet](#) class.*
- [CMeaUSBDeviceNet](#) ()  
*Initializes a new instance of [CMeaDeviceNet](#) class.*
- [~CMeaUSBDeviceNet](#) ()

## Additional Inherited Members

### 11.71.1 Detailed Description

Class for data acquisition via ME and MEA USB amplifiers

### 11.71.2 Constructor & Destructor Documentation

#### 11.71.2.1 [CMeaUSBDeviceNet](#)() [1/2] [CMeaUSBDeviceNet](#) ( [OnChannelData](#)<sup>^</sup> channelData, [OnError](#)<sup>^</sup> error )

Initializes a new instance of [CMeaDeviceNet](#) class.

##### Parameters

|                             |                                             |
|-----------------------------|---------------------------------------------|
| <a href="#">channelData</a> | Handler to call when new data is available. |
|-----------------------------|---------------------------------------------|

##### Parameters

|                       |                                       |
|-----------------------|---------------------------------------|
| <a href="#">error</a> | Handler to call when an error occurs. |
|-----------------------|---------------------------------------|

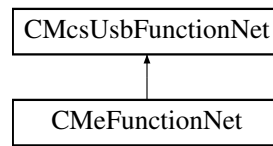
#### 11.71.2.2 [CMeaUSBDeviceNet](#)() [2/2] [CMeaUSBDeviceNet](#) ( )

Initializes a new instance of [CMeaDeviceNet](#) class.

#### 11.71.2.3 [~CMeaUSBDeviceNet](#)() [~CMeaUSBDeviceNet](#) ( )

## 11.72 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

#### 11.72.1 Detailed Description

#### 11.72.2 Constructor & Destructor Documentation

**11.72.2.1 CMeFunctionNet()** [1/2] [CMeFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb,  
[CMcsUsbFunctionPointerContainer](#)<sup>^</sup> meFunctionPointerContainer )

Initializes a new instance of the [CDacCalibrationFunctionNet](#) class.

**11.72.2.2 CMeFunctionNet()** [2/2] [CMeFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb )

**11.72.2.3 ~CMeFunctionNet()** virtual [~CMeFunctionNet](#) (  
 void ) [virtual]

**11.72.2.4 !CMeFunctionNet()** [!CMeFunctionNet](#) (  
 void )

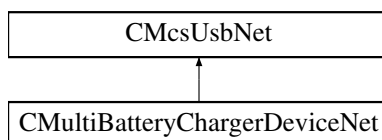
### 11.72.3 Member Function Documentation

**11.72.3.1 SetTransformer()** `void SetTransformer (`  
     `unsigned int index,`  
     `bool onoff )`

## 11.73 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

### 11.73.1 Detailed Description

[CMultiBatteryChargerDeviceNet](#) is the class to access the MBC-08 device.

### 11.73.2 Constructor & Destructor Documentation

#### 11.73.2.1 CMultiBatteryChargerDeviceNet() [CMultiBatteryChargerDeviceNet](#) ( )

Initializes a new instance of the [CMultiBatteryChargerDeviceNet](#) class.

#### 11.73.2.2 ~CMultiBatteryChargerDeviceNet() [virtual](#) [~CMultiBatteryChargerDeviceNet](#) ( ) [\[virtual\]](#)

#### 11.73.2.3 "!CMultiBatteryChargerDeviceNet() [!CMultiBatteryChargerDeviceNet](#) ( )

### 11.73.3 Member Function Documentation

#### 11.73.3.1 CapacityTest() [void](#) [CapacityTest](#) ( [uint32\\_t](#) *NrChannel* )

start capacity test on channel

**Parameters**

|                  |                    |
|------------------|--------------------|
| <i>NrChannel</i> | the channel number |
|------------------|--------------------|

**11.73.3.2 ChannelReset()** `void ChannelReset (`  
    `uint32_t NrChannel )`

cancel charging and capacity test functions; check if battery is connected

**Parameters**

|                  |                    |
|------------------|--------------------|
| <i>NrChannel</i> | the channel number |
|------------------|--------------------|

**11.73.3.3 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

**11.73.3.4 GetChannels()** `uint32_t GetChannels ( )`

gets number of channels

**Returns**

number of channels

**11.73.3.5 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

**11.73.3.6 GetChargeCapacity()** `uint32_t GetChargeCapacity (`  
`uint32_t NrChannel )`

gets the charge capacity; unit:  $\mu\text{Ah}$

## Parameters

|                  |                    |
|------------------|--------------------|
| <i>NrChannel</i> | the channel number |
|------------------|--------------------|

## Returns

the capacity in  $\mu\text{Ah}$

**11.73.3.7 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

**11.73.3.8 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

**11.73.3.9 GetChargingPCoefficient()** `uint32_t GetChargingPCoefficient ( )`

gets the p-coefficient for charging in mA/V / nominal charging current

**Returns**

the p-coefficient

**11.73.3.10 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

**11.73.3.11 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

**11.73.3.12 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

**11.73.3.13 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

**11.73.3.14 GetRatedCapacity()** `MbcRatedCapacityEnumNet GetRatedCapacity (uint32_t NrChannel )`

gets the rated capacity

## Parameters

|                  |                    |
|------------------|--------------------|
| <i>NrChannel</i> | the channel number |
|------------------|--------------------|

## Returns

the capacity

**11.73.3.15 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  |

**11.73.3.16 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 |
|---------------------|-------------------|

**11.73.3.17 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 |

**11.73.3.18 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 |

**11.73.3.19 SetRatedCapacity()** `void SetRatedCapacity (`  
    `uint32_t NrChannel,`  
    `MbcRatedCapacityEnumNet NewRatedCapacity )`

sets the rated capacity

## Parameters

|                         |                    |
|-------------------------|--------------------|
| <i>NrChannel</i>        | the channel number |
| <i>NewRatedCapacity</i> | the capacity       |

**11.73.3.20 SetRatedCapacityVolatile()** `void SetRatedCapacityVolatile (`  
     `uint32_t NrChannel,`  
     `MbcRatedCapacityEnumNet NewRatedCapacity )`

sets the rated capacity (i.e. charge current) without storing it persistently

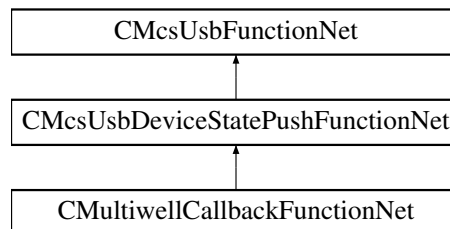
## Parameters

|                         |                    |
|-------------------------|--------------------|
| <i>NrChannel</i>        | the channel number |
| <i>NewRatedCapacity</i> | the capacity       |

## 11.74 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

### 11.74.1 Detailed Description

[CMultiwellCallbackFunctionNet](#) is the class to access the Multiwell-Mini-Stimulator

### 11.74.2 Constructor & Destructor Documentation

**11.74.2.1 CMultiwellCallbackFunctionNet() [1/2]** [CMultiwellCallbackFunctionNet](#) (   
 [CMcsUsbNet](#)<sup>^</sup> *mcsusb*,   
 [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> *pMultiwellCallbackFunctionPointerContainer* )

Initializes a new instance of the [CMultiwellCallbackFunctionNet](#) class.

**11.74.2.2 CMultiwellCallbackFunctionNet() [2/2]** [CMultiwellCallbackFunctionNet](#) (   
 [CMcsUsbNet](#)<sup>^</sup> *mcsusb* )

**11.74.2.3 ~CMultiwellCallbackFunctionNet()** [virtual ~CMultiwellCallbackFunctionNet](#) ( ) [virtual]

**11.74.2.4 ~!CMultiwellCallbackFunctionNet()** [!CMultiwellCallbackFunctionNet](#) ( )

### 11.74.3 Member Function Documentation

**11.74.3.1 GetPlateClampStateByHeadstage()** [PlateClampEnumNet](#) [GetPlateClampStateByHeadstage](#) (   
 [uint32\\_t](#) *Headstage* )

Gets the state of the plate

#### Parameters

|                  |                      |
|------------------|----------------------|
| <i>Headstage</i> | The headstage number |
|------------------|----------------------|

#### Returns

The plate state

**11.74.3.2 OnGetPlateClampStateByHeadstage()** `delegate void OnGetPlateClampStateByHeadstage (`  
`uint32_t Headstage,`  
`PlateClampEnumNet plateState )`

#### 11.74.4 Event Documentation

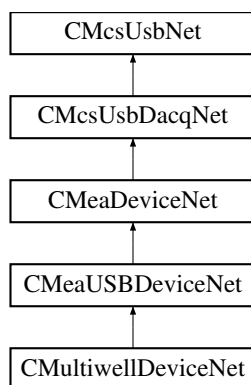
**11.74.4.1 GetPlateClampStateByHeadstageEvent** `OnGetPlateClampStateByHeadstage^ GetPlateClamp↔`  
`StateByHeadstageEvent [add], [remove], [raise]`

Event fires when the plate state for the headstage number has changed

### 11.75 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

### 11.75.1 Detailed Description

[CMultiwellDeviceNet](#) is the class to access the Multiwell device.

### 11.75.2 Constructor & Destructor Documentation

**11.75.2.1 CMultiwellDeviceNet()** `CMultiwellDeviceNet ( )`

Initializes a new instance of the `CMultiwellDeviceNet` class.

**11.75.2.2 ~CMultiwellDeviceNet()** `virtual ~CMultiwellDeviceNet ( ) [virtual]`**11.75.2.3 !CMultiwellDeviceNet()** `!CMultiwellDeviceNet ( )`**11.75.3 Member Function Documentation****11.75.3.1 ClosePlateClamp()** `void ClosePlateClamp ( )`

Closes the plate clamp.

**11.75.3.2 GetPlateClampLockState()** `uint32_t GetPlateClampLockState ( )`

Gets the state of the plate clamp lock.

**Returns**

the state of the plate lock (unlocked/locked)

**11.75.3.3 GetPlateClampState()** [1/2] `PlateClampEnumNet GetPlateClampState ( )`

Gets the state of the Multiwell plate clamp.

**Returns**

the state of the plate clamp (open/closed)

**11.75.3.4 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

**11.75.3.5 GetPlateMux()** [1/2] `uint32_t GetPlateMux ( )`

Gets the selected quarter of the electrodes on a high density Multiwell plate.

## Returns

the selected quarter

**11.75.3.6 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

**11.75.3.7 GetPlateType()** [1/2] `MultiwellPlateTypeEnumNet GetPlateType ( )`

Gets the plate type.

## Returns

the plate type

**11.75.3.8 GetPlateType()** [2/2] `MultiwellPlateTypeEnumNet GetPlateType (   
uint32_t Headstage )`

Gets the plate type.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <i>Headstage</i> | The headstage to query. |
|------------------|-------------------------|

**Returns**

the plate type

**11.75.3.9 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

**11.75.3.10 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".

**11.75.3.11 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.

**11.75.3.12 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".

**11.75.3.13 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".

**11.75.3.14 LockPlateClamp()** `void LockPlateClamp ( )`

Locks the plate clamp.

**11.75.3.15 OpenPlateClamp()** `void OpenPlateClamp ( )`

Opens the plate clamp.

**11.75.3.16 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    |

**11.75.3.17 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 |
|---------------------|----------------------|

**11.75.3.18 SetPlateType()** [1/2] `void SetPlateType (`  
`MultiwellPlateTypeEnumNet plateType )`

Sets the plate type.

## Parameters

|                  |                |
|------------------|----------------|
| <i>plateType</i> | the plate type |
|------------------|----------------|

**11.75.3.19 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          |

**11.75.3.20 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 |

**11.75.3.21 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". |

**11.75.3.22 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 |

**11.75.3.23 StopPlateClamp()** `void StopPlateClamp ( )`

Stops the plate clamp movement.

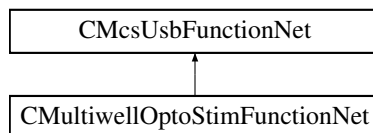
**11.75.3.24 UnlockPlateClamp()** `void UnlockPlateClamp ( )`

Unlocks the plate clamp.

## 11.76 CMultiwellOptoStimFunctionNet Class Reference

[CMultiwellOptoStimFunctionNet](#) is the class to access the optical properties of the Multiwell Optostim device

Inheritance diagram for [CMultiwellOptoStimFunctionNet](#):



### Public Member Functions

- [CMultiwellOptoStimFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> pMultiwellOptoStimFunctionPointerContainer)
- Initializes a new instance of the [CMultiwellOptoStimFunctionNet](#) class.*
- [CMultiwellOptoStimFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- virtual [~CMultiwellOptoStimFunctionNet](#) ()
- [!CMultiwellOptoStimFunctionNet](#) ()
- [uint32\\_t](#) [GetWaveLengthInNanometer](#) ([uint16\\_t](#) channel)
- [uint32\\_t](#) [GetAbsMaxCurrentInMicroAmp](#) ([uint16\\_t](#) channel)
- [uint32\\_t](#) [GetMaxDurationHighCurrentInMicroSec](#) ([uint16\\_t](#) channel)
- [uint32\\_t](#) [GetMaxDutyCycleHighCurrent](#) ([uint16\\_t](#) channel)
- [uint32\\_t](#) [GetPermanentCurrentInMicroAmp](#) ([uint16\\_t](#) channel)
- [uint32\\_t](#) [GetColorRgb](#) ([uint16\\_t](#) channel)
- [String](#)<sup>^</sup> [GetColorStr](#) ([uint16\\_t](#) channel)
- void [SetWaveLengthInNanometer](#) ([uint16\\_t](#) channel, [uint32\\_t](#) WaveLength\_nm)
- void [SetAbsMaxCurrentInMicroAmp](#) ([uint16\\_t](#) channel, [uint32\\_t](#) AbsoluteMaxCurrent\_uA)
- void [SetMaxDurationHighCurrentInMicroSec](#) ([uint16\\_t](#) channel, [uint32\\_t](#) AbsoluteMaxDuration\_us)
- void [SetMaxDutyCycleHighCurrent](#) ([uint16\\_t](#) channel, [uint32\\_t](#) MaxDutyCycleHighCurrent)
- void [SetPermanentCurrentInMicroAmp](#) ([uint16\\_t](#) channel, [uint32\\_t](#) PermanentCurrent\_uA)
- void [SetColorRgb](#) ([uint16\\_t](#) channel, [uint32\\_t](#) ColorRGB)
- void [SetColorStr](#) ([uint16\\_t](#) channel, [String](#)<sup>^</sup> ColorString)

### Additional Inherited Members

#### 11.76.1 Detailed Description

[CMultiwellOptoStimFunctionNet](#) is the class to access the optical properties of the Multiwell Optostim device

#### 11.76.2 Constructor & Destructor Documentation

**11.76.2.1 [CMultiwellOptoStimFunctionNet](#)() [1/2]** [CMultiwellOptoStimFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> pMultiwellOptoStimFunctionPointerContainer )

Initializes a new instance of the [CMultiwellOptoStimFunctionNet](#) class.

**11.76.2.2 CMultiwellOptoStimFunctionNet()** [2/2] `CMultiwellOptoStimFunctionNet ( CMcsUsbNet^ mcsusb )`

**11.76.2.3 ~CMultiwellOptoStimFunctionNet()** `virtual ~CMultiwellOptoStimFunctionNet ( )` [virtual]

**11.76.2.4 ~!CMultiwellOptoStimFunctionNet()** `!CMultiwellOptoStimFunctionNet ( )`

### 11.76.3 Member Function Documentation

**11.76.3.1 GetAbsMaxCurrentInMicroAmp()** `uint32_t GetAbsMaxCurrentInMicroAmp ( uint16_t channel )`

#### Parameters

|                |                             |
|----------------|-----------------------------|
| <i>channel</i> | the (analog) channel number |
|----------------|-----------------------------|

#### Returns

absolute max. current; unit: uA

**11.76.3.2 GetColorRgb()** `uint32_t GetColorRgb ( uint16_t channel )`

#### Parameters

|                |                             |
|----------------|-----------------------------|
| <i>channel</i> | the (analog) channel number |
|----------------|-----------------------------|

#### Returns

RGB-value of LED color

**11.76.3.3 GetColorStr()** `String ^ GetColorStr ( uint16_t channel )`

**Parameters**

|                |                             |
|----------------|-----------------------------|
| <i>channel</i> | the (analog) channel number |
|----------------|-----------------------------|

**Returns**

LED color as string

**11.76.3.4 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

**11.76.3.5 GetMaxDutyCycleHighCurrent()** `uint32_t GetMaxDutyCycleHighCurrent (uint16_t channel )`

**Parameters**

|                |                             |
|----------------|-----------------------------|
| <i>channel</i> | the (analog) channel number |
|----------------|-----------------------------|

**Returns**

max. duty cycle at max. current; unit: 100\*%

**11.76.3.6 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



**11.76.3.7 GetWaveLengthInNanometer()** `uint32_t GetWaveLengthInNanometer ( uint16_t channel )`

#### Parameters

|                |                             |
|----------------|-----------------------------|
| <i>channel</i> | the (analog) channel number |
|----------------|-----------------------------|

#### Returns

wavelength of this channel's LEDs; unit: nm

**11.76.3.8 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 |

**11.76.3.9 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      |

**11.76.3.10 SetColorStr()** `void SetColorStr ( uint16_t channel, String^ ColorString )`

#### Parameters

|                    |                             |
|--------------------|-----------------------------|
| <i>channel</i>     | the (analog) channel number |
| <i>ColorString</i> | LED color as string         |

**11.76.3.11 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 |

**11.76.3.12 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*% |

**11.76.3.13 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 |

**11.76.3.14 SetWaveLengthInNanometer()** `void SetWaveLengthInNanometer (`  

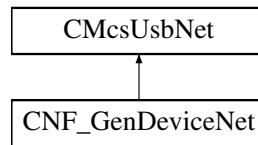
```
uint16_t channel,
uint32_t WaveLength_nm)
```

**Parameters**

|                      |                                             |
|----------------------|---------------------------------------------|
| <i>channel</i>       | the (analog) channel number                 |
| <i>WaveLength_nm</i> | wavelength of this channel's LEDs; unit: nm |

## 11.77 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

#### 11.77.1 Constructor & Destructor Documentation

**11.77.1.1** [CNF\\_GenDeviceNet\(\)](#) [CNF\\_GenDeviceNet](#) (   
 void )

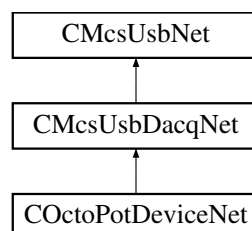
**11.77.1.2** [~CNF\\_GenDeviceNet\(\)](#) [~CNF\\_GenDeviceNet](#) (   
 void )

#### 11.77.2 Member Function Documentation

**11.77.2.1** [Set\\_Values\(\)](#) void [Set\\_Values](#) (   
 unsigned int *frequency*,   
 unsigned int *amplitude* )

## 11.78 COctoPotDeviceNet Class Reference

Inheritance diagram for COctoPotDeviceNet:



## Public Member Functions

- [COctoPotDeviceNet](#) (void)
- [COctoPotDeviceNet](#) ([OnChannelData](#)^ channelData, [OnError](#)^ error)
- [uint32\\_t SetOutputRate](#) ([uint32\\_t](#) rate)
- [uint32\\_t SetBathclamp](#) (unsigned int block, bool enable)
- [uint32\\_t SetDacValue](#) (int channel, int value)
- [uint32\\_t SetDacAutoControl](#) (unsigned int channel)
- [uint32\\_t SetPidParameter](#) (unsigned int channel, int const\_p, int const\_i, int shift\_p, int shift\_i)
- [uint32\\_t SetRampParameter](#) (unsigned int channel, int start, int min, int max, int slope, int slope2, int pause, unsigned int samples)
- [uint32\\_t RampStart](#) (int channelmap)
- [uint32\\_t SetSineParameter](#) (unsigned int channel, int amplitude)
- [uint32\\_t SineStart](#) (int channelmap)
- [uint32\\_t SetPatternListEntry](#) (unsigned int channel, unsigned int position, unsigned int duration, int value)
- [uint32\\_t PatternListStart](#) (int channelmap)
- [uint32\\_t SetAdcOffset](#) (unsigned int channel, int offset)
- [uint32\\_t SetDacOffset](#) (unsigned int channel, int offset)
- [uint32\\_t ResetAdcOffset](#) (unsigned int channel)
- [uint32\\_t ResetDacOffset](#) (unsigned int channel)
- [uint32\\_t BurnAdcOffset](#) ()
- [uint32\\_t BurnDacOffset](#) ()
- [uint32\\_t GetAdcOffset](#) (unsigned int channel, [System::Runtime::InteropServices::Out] int ^ offset)
- [uint32\\_t GetDacOffset](#) (unsigned int channel, [System::Runtime::InteropServices::Out] int ^ offset)
- [uint32\\_t SetAmplificationSwitch](#) (unsigned int channel, unsigned int state)
- [uint32\\_t SetChannelSwitch](#) (unsigned int channel, unsigned int state)
- [uint32\\_t SetNumberOfChannels](#) (unsigned int NumberOfChannels)
- [uint32\\_t EnableDigitalIn](#) (bool enable)
- [uint32\\_t EnableTimestamp](#) (bool enable)
- [uint32\\_t EnableChecksum](#) (bool enable)

## Additional Inherited Members

### 11.78.1 Constructor & Destructor Documentation

**11.78.1.1 COctoPotDeviceNet()** [1/2] [COctoPotDeviceNet](#) (   
 void )

**11.78.1.2 COctoPotDeviceNet()** [2/2] [COctoPotDeviceNet](#) (   
 [OnChannelData](#)^ channelData,   
 [OnError](#)^ error )

### 11.78.2 Member Function Documentation

**11.78.2.1 BurnAdcOffset()** uint32\_t BurnAdcOffset ( )

**11.78.2.2 BurnDacOffset()** uint32\_t BurnDacOffset ( )

**11.78.2.3 EnableChecksum()** uint32\_t EnableChecksum (   
bool *enable* )

**11.78.2.4 EnableDigitalIn()** uint32\_t EnableDigitalIn (   
bool *enable* )

**11.78.2.5 EnableTimestamp()** uint32\_t EnableTimestamp (   
bool *enable* )

**11.78.2.6 GetAdcOffset()** uint32\_t GetAdcOffset (   
unsigned int *channel*,   
[System::Runtime::InteropServices::Out] int ^ *offset* )

**11.78.2.7 GetDacOffset()** uint32\_t GetDacOffset (   
unsigned int *channel*,   
[System::Runtime::InteropServices::Out] int ^ *offset* )

**11.78.2.8 PatternListStart()** uint32\_t PatternListStart (   
int *channelmap* )

**11.78.2.9 RampStart()** uint32\_t RampStart (   
int *channelmap* )

**11.78.2.10 ResetAdcOffset()** uint32\_t ResetAdcOffset (   
unsigned int *channel* )

**11.78.2.11 ResetDacOffset()** uint32\_t ResetDacOffset (   
 unsigned int *channel* )

**11.78.2.12 SetAdcOffset()** uint32\_t SetAdcOffset (   
 unsigned int *channel*,   
 int *offset* )

**11.78.2.13 SetAmplificationSwitch()** uint32\_t SetAmplificationSwitch (   
 unsigned int *channel*,   
 unsigned int *state* )

**11.78.2.14 SetBathclamp()** uint32\_t SetBathclamp (   
 unsigned int *block*,   
 bool *enable* )

**11.78.2.15 SetChannelSwitch()** uint32\_t SetChannelSwitch (   
 unsigned int *channel*,   
 unsigned int *state* )

**11.78.2.16 SetDacAutoControl()** uint32\_t SetDacAutoControl (   
 unsigned int *channel* )

**11.78.2.17 SetDacOffset()** uint32\_t SetDacOffset (   
 unsigned int *channel*,   
 int *offset* )

**11.78.2.18 SetDacValue()** uint32\_t SetDacValue (   
 int *channel*,   
 int *value* )

**11.78.2.19 SetNumberOfChannels()** uint32\_t SetNumberOfChannels (   
 unsigned int *NumberOfChannels* )

**11.78.2.20 SetOutputRate()** uint32\_t SetOutputRate (   
uint32\_t *rate* )

**11.78.2.21 SetPatternListEntry()** uint32\_t SetPatternListEntry (   
unsigned int *channel*,   
unsigned int *position*,   
unsigned int *duration*,   
int *value* )

**11.78.2.22 SetPidParameter()** uint32\_t SetPidParameter (   
unsigned int *channel*,   
int *const\_p*,   
int *const\_i*,   
int *shift\_p*,   
int *shift\_i* )

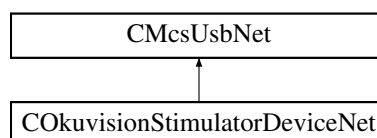
**11.78.2.23 SetRampParameter()** uint32\_t SetRampParameter (   
unsigned int *channel*,   
int *start*,   
int *min*,   
int *max*,   
int *slope*,   
int *slope2*,   
int *pause*,   
unsigned int *samples* )

**11.78.2.24 SetSineParameter()** uint32\_t SetSineParameter (   
unsigned int *channel*,   
int *amplitude* )

**11.78.2.25 SineStart()** uint32\_t SineStart (   
int *channelmap* )

## 11.79 COkuvisionStimulatorDeviceNet Class Reference

Inheritance diagram for COkuvisionStimulatorDeviceNet:



## Public Member Functions

- [COKuvisionStimulatorDeviceNet](#) (void)
- [~COKuvisionStimulatorDeviceNet](#) (void)
- void [SetPulseform](#) (int channel, int current, int pulsewidth, int periode, int duration)
- void [GetPulseform](#) (int channel, [System::Runtime::InteropServices::Out] int% current, [System::Runtime::InteropServices::Out] int% pulsewidth, [System::Runtime::InteropServices::Out] int% periode, [System::Runtime::InteropServices::Out] int% duration)
- void [SetMaxPower](#) (int channel, int power)
- int [GetMaxPower](#) (int channel)
- void [SetMaxVoltage](#) (int channel, int voltage)
- int [GetMaxVoltage](#) (int channel)
- void [SetCheckVoltage](#) (int channel, int voltage)
- int [GetCheckVoltage](#) (int channel)
- int [GetVoltage](#) (int channel)
- void [SetDACOffset](#) (int channel, int part, int offset)
- int [GetDACOffset](#) (int channel, int part)
- void [SetRTC](#) (uint8\_t year, uint8\_t month, uint8\_t day, uint8\_t hour, uint8\_t minute, uint8\_t second)
- void [GetRTC](#) ([System::Runtime::InteropServices::Out] uint8\_t% year, [System::Runtime::InteropServices::Out] uint8\_t% month, [System::Runtime::InteropServices::Out] uint8\_t% day, [System::Runtime::InteropServices::Out] uint8\_t% hour, [System::Runtime::InteropServices::Out] uint8\_t% minute, [System::Runtime::InteropServices::Out] uint8\_t% second)
- void [SetRTC](#) (DateTime timestamp)
- DateTime [GetRTC](#) ()
- void [GetStimulatorStatus](#) ([System::Runtime::InteropServices::Out] int% startstop, [System::Runtime::InteropServices::Out] int% last\_error, [System::Runtime::InteropServices::Out] int% battery\_status)
- void [SetCurrentFactor](#) (int channel, int factor)
- int [GetCurrentFactor](#) (int channel)

## Additional Inherited Members

### 11.79.1 Constructor & Destructor Documentation

**11.79.1.1 COKuvisionStimulatorDeviceNet()** [COKuvisionStimulatorDeviceNet](#) ( void )

**11.79.1.2 ~COKuvisionStimulatorDeviceNet()** [~COKuvisionStimulatorDeviceNet](#) ( void )

### 11.79.2 Member Function Documentation

**11.79.2.1 GetCheckVoltage()** int GetCheckVoltage ( int channel )



**11.79.2.2 GetCurrentFactor()** `int GetCurrentFactor (`  
`int channel )`

**11.79.2.3 GetDACOffset()** `int GetDACOffset (`  
`int channel,`  
`int part )`

**11.79.2.4 GetMaxPower()** `int GetMaxPower (`  
`int channel )`

**11.79.2.5 GetMaxVoltage()** `int GetMaxVoltage (`  
`int channel )`

**11.79.2.6 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 )`

**11.79.2.7 GetRTC()** `[1/2] DateTime GetRTC ( )`

**11.79.2.8 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 )`

**11.79.2.9 GetStimulatorStatus()** `void GetStimulatorStatus (`  
`[System::Runtime::InteropServices::Out] int% startstop,`  
`[System::Runtime::InteropServices::Out] int% last_error,`  
`[System::Runtime::InteropServices::Out] int% battery_status )`

**11.79.2.10 GetVoltage()** `int GetVoltage (`  
    `int channel )`

**11.79.2.11 SetCheckVoltage()** `void SetCheckVoltage (`  
    `int channel,`  
    `int voltage )`

**11.79.2.12 SetCurrentFactor()** `void SetCurrentFactor (`  
    `int channel,`  
    `int factor )`

**11.79.2.13 SetDACOffset()** `void SetDACOffset (`  
    `int channel,`  
    `int part,`  
    `int offset )`

**11.79.2.14 SetMaxPower()** `void SetMaxPower (`  
    `int channel,`  
    `int power )`

**11.79.2.15 SetMaxVoltage()** `void SetMaxVoltage (`  
    `int channel,`  
    `int voltage )`

**11.79.2.16 SetPulseform()** `void SetPulseform (`  
    `int channel,`  
    `int current,`  
    `int pulsewidth,`  
    `int periode,`  
    `int duration )`

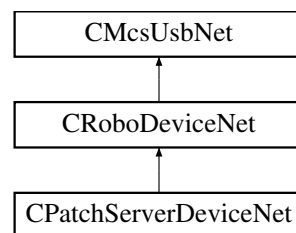
**11.79.2.17 SetRTC()** `[1/2] void SetRTC (`  
    `DateTime timestamp )`

**11.79.2.18 SetRTC()** [2/2] `void SetRTC (`  
`uint8_t year,`  
`uint8_t month,`  
`uint8_t day,`  
`uint8_t hour,`  
`uint8_t minute,`  
`uint8_t second )`

## 11.80 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

#### 11.80.1 Detailed Description

[CPatchServerDeviceNet](#) is the class to control the MCS PatchServer device

#### 11.80.2 Constructor & Destructor Documentation

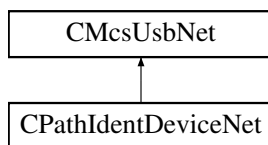
**11.80.2.1 CPatchServerDeviceNet()** [CPatchServerDeviceNet](#) (  
`void` )

#### 11.80.3 Property Documentation

**11.80.3.1 Sensor** `CMcsBus_SensorNet^ Sensor [get]`

## 11.81 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

#### 11.81.1 Constructor & Destructor Documentation

**11.81.1.1 CPathIdentDeviceNet()** `CPathIdentDeviceNet` ( void )

**11.81.1.2 ~CPathIdentDeviceNet()** `~CPathIdentDeviceNet` ( void )

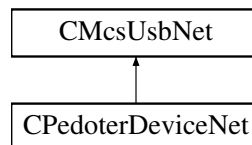
#### 11.81.2 Member Function Documentation

**11.81.2.1 Measure()** void Measure ( [System::Runtime::InteropServices::Out] unsigned int% *phase*, [System::Runtime::InteropServices::Out] unsigned int% *amplitude* )

**11.81.2.2 Set\_Values()** void Set\_Values (   
     unsigned int *frequency*,   
     unsigned int *amplitude* )

## 11.82 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

#### 11.82.1 Detailed Description

#### 11.82.2 Constructor & Destructor Documentation

**11.82.2.1 CPedoterDeviceNet()** [CPedoterDeviceNet](#) ( )

Initializes a new instance of the [CPedoterDeviceNet](#) class.

**11.82.2.2 ~CPedoterDeviceNet()** virtual [~CPedoterDeviceNet](#) ( ) [virtual]

**11.82.2.3 "!CPedoterDeviceNet()** [!CPedoterDeviceNet](#) ( )

#### 11.82.3 Member Function Documentation

**11.82.3.1 GetCommand()** uint32\_t GetCommand (   
     uint16\_t *Argument* )

Get value from the pedoter device

**Parameters**

|                 |          |
|-----------------|----------|
| <i>Argument</i> | argument |
|-----------------|----------|

**Returns**

value

**11.82.3.2 SetCommand()** void SetCommand (   
     uint16\_t *Argument*,   
     uint32\_t *pData* )

Set value on the pedoter device

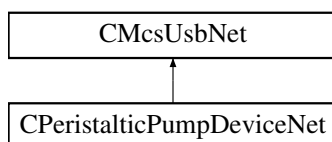
**Parameters**

|                 |          |
|-----------------|----------|
| <i>Argument</i> | argument |
| <i>pData</i>    | value    |

**11.83 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****11.83.1 Detailed Description**

[CPeristalticPumpDeviceNet](#) is the class to control a Persistaltic Pump.

### 11.83.2 Constructor & Destructor Documentation

**11.83.2.1 CPeristalticPumpDeviceNet()** `CPeristalticPumpDeviceNet (void )`

Initialize a new instance of the `CPeristalticPumpDeviceNet` class.

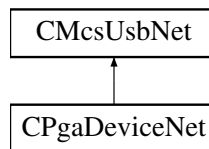
**11.83.2.2 ~CPeristalticPumpDeviceNet()** `~CPeristalticPumpDeviceNet (void )`

### 11.83.3 Property Documentation

**11.83.3.1 McsBus\_MotorControl** `CMcsBus_MotorControlNet^ McsBus_MotorControl [get]`

## 11.84 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

### 11.84.1 Constructor & Destructor Documentation

**11.84.1.1 CPgaDeviceNet()** `CPgaDeviceNet ( )`

**11.84.1.2 ~CPgaDeviceNet()** `~CPgaDeviceNet ( )`

### 11.84.2 Member Function Documentation

**11.84.2.1 ApplyGains()** `uint32_t ApplyGains ( )`

**11.84.2.2 DefineAmplification()** `uint32_t DefineAmplification (`  
    `int index,`  
    `int amplification,`  
    `int poti1,`  
    `int poti2 )`

**11.84.2.3 DefineFrequencyRange()** `uint32_t DefineFrequencyRange (`  
    `int index,`  
    `int low,`  
    `int high,`  
    `int channels,`  
    `int gain )`

**11.84.2.4 DefineNumAmplifications()** `uint32_t DefineNumAmplifications (`  
    `int number )`

**11.84.2.5 DefineNumFrequencyRanges()** `uint32_t DefineNumFrequencyRanges (`  
    `int rnum )`



**11.84.2.6 GetAmplification()** uint32\_t GetAmplification (   
     int *index*,   
     [System::Runtime::InteropServices::Out] int% *amplification*,   
     [System::Runtime::InteropServices::Out] int% *potil*,   
     [System::Runtime::InteropServices::Out] int% *poti2* )

**11.84.2.7 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* )

**11.84.2.8 GetGain()** uint32\_t GetGain (   
     int *channel*,   
     [System::Runtime::InteropServices::Out] int% *Gain*,   
     [System::Runtime::InteropServices::Out] int% *potil*,   
     [System::Runtime::InteropServices::Out] int% *poti2* )

**11.84.2.9 GetNumAmplifications()** uint32\_t GetNumAmplifications (   
     [System::Runtime::InteropServices::Out] int% *number* )

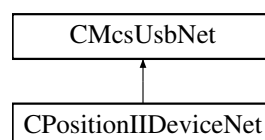
**11.84.2.10 GetNumFrequencyRanges()** uint32\_t GetNumFrequencyRanges (   
     [System::Runtime::InteropServices::Out] int% *numRanges* )

**11.84.2.11 SetGain()** uint32\_t SetGain (   
     int *channel*,   
     int *Gain*,   
     int *potil*,   
     int *poti2* )

## 11.85 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

### 11.85.1 Detailed Description

[CPositionIIDeviceNet](#) is the class to control PositionII devices

### 11.85.2 Constructor & Destructor Documentation

#### 11.85.2.1 CPositionIIDeviceNet() [CPositionIIDeviceNet](#) ( )

Initializes a new instance of the [CPositionIIDeviceNet](#) class.

#### 11.85.2.2 ~CPositionIIDeviceNet() [virtual](#) [~CPositionIIDeviceNet](#) ( ) [virtual]

#### 11.85.2.3 "!CPositionIIDeviceNet() [!CPositionIIDeviceNet](#) ( )

### 11.85.3 Member Function Documentation

#### 11.85.3.1 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

**11.85.3.2 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 |

**11.85.3.3 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         |

**11.85.3.4 GetImplantCurrentSetpoint()** uint32\_t GetImplantCurrentSetpoint (

```

 uint16_t coil)

```

sets the implant current setpoint

**Parameters**

|             |          |
|-------------|----------|
| <i>coil</i> | the coil |
|-------------|----------|

**Returns**

the current

**11.85.3.5 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

**11.85.3.6 GetImplantState()** uint32\_t GetImplantState (  
uint16\_t *coil* )

gets the implantat state

**Parameters**

|             |          |
|-------------|----------|
| <i>coil</i> | the coil |
|-------------|----------|

**Returns**

the state

**11.85.3.7 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

**11.85.3.8 GetPowerStrength()** `uint32_t GetPowerStrength (`  
`uint16_t coil )`

sets the power for the trigger pulses

**Parameters**

|             |          |
|-------------|----------|
| <i>coil</i> | the coil |
|-------------|----------|

**Returns**

the power

**11.85.3.9 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 |

**11.85.3.10 GetStateDebugData()** `uint32_t GetStateDebugData (`  
`uint16_t coil )`

get the debug queue state

**Parameters**

|             |          |
|-------------|----------|
| <i>coil</i> | the coil |
|-------------|----------|

**Returns**

the state

**11.85.3.11 GetStateEventData()** `uint32_t GetStateEventData ( )`

get the event queue state

**Returns**

the state

**11.85.3.12 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 |

**11.85.3.13 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 |

**11.85.3.14 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 |

**11.85.3.15 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 |

**11.85.3.16 SetStateEventData()** `void SetStateEventData (`  
    `uint32_t state )`

clears/starts/stops the event queue for a certain coil

## Parameters

|              |                  |
|--------------|------------------|
| <i>state</i> | clear/start/stop |
|--------------|------------------|

**11.85.3.17 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 |

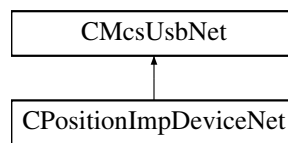
### 11.85.4 Property Documentation

#### 11.85.4.1 RFFunction `CRFFFunctionNet^ RFFunction [get]`

## 11.86 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

#### 11.86.1 Detailed Description

`CPositionImpDeviceNet` is the class to access the Position/Imp devices

## 11.86.2 Constructor & Destructor Documentation

### 11.86.2.1 CPositionImpDeviceNet() `CPositionImpDeviceNet ( )`

Initializes a new instance of the `CPositionImpDeviceNet` class.

### 11.86.2.2 ~CPositionImpDeviceNet() `virtual ~CPositionImpDeviceNet ( ) [virtual]`

### 11.86.2.3 "!CPositionImpDeviceNet() `!CPositionImpDeviceNet ( )`

## 11.86.3 Member Function Documentation

### 11.86.3.1 ConnectedImp() `uint32_t ConnectedImp ( )`

The ID of the connected Imp device

#### Returns

The ID

### 11.86.3.2 ConnectImp() `void ConnectImp ( uint32_t id )`

Connect to a Imp device with a certain ID

#### Parameters

|           |        |
|-----------|--------|
| <i>id</i> | The ID |
|-----------|--------|

### 11.86.3.3 GetDeviceList() `uint32_t GetDeviceList ( int32_t index )`

Gets the device list

**Parameters**

|              |           |
|--------------|-----------|
| <i>index</i> | the index |
|--------------|-----------|

**Returns**

the ID

**11.86.3.4 GetImpId()** `uint32_t GetImpId ( )`

Gets the ID of the impedance measure device

**Returns**

the ID

**11.86.3.5 GetRFFrequency()** `int32_t GetRFFrequency ( )`

Gets currently used RF frequency

**Returns**

The frequency

**11.86.3.6 SetDeviceList()** `void SetDeviceList (`  
    `int32_t index,`  
    `uint32_t id )`

Sets the device list

**Parameters**

|              |           |
|--------------|-----------|
| <i>index</i> | the index |
| <i>id</i>    | the ID    |

**11.86.3.7 SetImpId()** `void SetImpId (`  
    `uint32_t id )`

Sets the ID of the impedance measure device

## Parameters

|           |        |
|-----------|--------|
| <i>id</i> | the ID |
|-----------|--------|

**11.86.3.8 SetRFFrequency()** `void SetRFFrequency (`  
`int32_t frequency )`

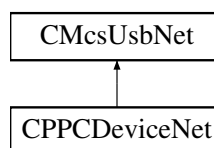
Sets the current RF frequency

## Parameters

|                  |               |
|------------------|---------------|
| <i>frequency</i> | The frequency |
|------------------|---------------|

## 11.87 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

#### 11.87.1 Constructor & Destructor Documentation

**11.87.1.1 CPPCDeviceNet()** `CPPCDeviceNet (`  
`void )`

## 11.87.2 Property Documentation

**11.87.2.1 McsBus** [CMcsBusNet](#)<sup>^</sup> McsBus [get]

**11.87.2.2 McsBus\_MotorControl** [CMcsBus\\_MotorControlNet](#)<sup>^</sup> McsBus\_MotorControl [get]

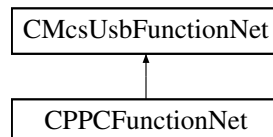
**11.87.2.3 McsBus\_Sensor** [CMcsBus\\_SensorNet](#)<sup>^</sup> McsBus\_Sensor [get]

**11.87.2.4 PPCFunction** [CPPPCFunctionNet](#)<sup>^</sup> PPCFunction [get]

## 11.88 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

### 11.88.1 Detailed Description

[CPPCFunctionNet](#) is the class to access the PPC (high precision Patch Peristaltic patch Pump

### 11.88.2 Constructor & Destructor Documentation

**11.88.2.1 CPPCFunctionNet()** [1/2] [CPPCFunctionNet](#) (  
     CMcsUsbNet^ mcsusb,  
     CMcsUsbFunctionPointerContainer^ pPPCFunctionPointerContainer )

Initializes a new instance of the [CPPCFunctionNet](#) class.

**11.88.2.2 CPPCFunctionNet()** [2/2] `CPPCFunctionNet (`  
`CMcsUsbNet^ mcsusb )`

**11.88.2.3 ~CPPCFunctionNet()** `virtual ~CPPCFunctionNet ( ) [virtual]`

**11.88.2.4 "!CPPCFunctionNet()** `!CPPCFunctionNet ( )`

### 11.88.3 Member Function Documentation

**11.88.3.1 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                |

**11.88.3.2 GetAnalogVoltage()** `uint16_t GetAnalogVoltage (`  
`uint16_t channel )`

Reads the current analog voltage

#### Parameters

|                |                    |
|----------------|--------------------|
| <i>channel</i> | The Channel Number |
|----------------|--------------------|

#### Returns

The Analog Voltage

**11.88.3.3 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 |

**11.88.3.4 GetDigitalIn()** `uint16_t GetDigitalIn (`  
`uint16_t channel )`

Reads the digital input state

**Parameters**

|                |                    |
|----------------|--------------------|
| <i>channel</i> | The Channel Number |
|----------------|--------------------|

**Returns**

The Digital State

**11.88.3.5 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 |

**11.88.3.6 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.

**11.88.3.7 GetPumpSpeedUnit()** `int GetPumpSpeedUnit ( uint16_t channel )`

Reads the Pump Speed Unit

**Parameters**

|                |                    |
|----------------|--------------------|
| <i>channel</i> | The Channel Number |
|----------------|--------------------|

**Returns**

The Speed Unit

**11.88.3.8 GetSupplyVoltage()** `uint16_t GetSupplyVoltage ( )`

Reads the current supply voltage in mV

**Returns**

The supply voltage

**11.88.3.9 GetValveActive()** `int GetValveActive ( uint16_t valve )`

Gets the valve active/inactive state

**Parameters**

|              |                  |
|--------------|------------------|
| <i>valve</i> | The valve number |
|--------------|------------------|

**Returns**

The valve state

**11.88.3.10 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 |

**11.88.3.11 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 |

**11.88.3.12 MeasureReservoir()** `int32_t MeasureReservoir ( )`

Measures the reservoir pressure

## Returns

The pressure

**11.88.3.13 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 |

**11.88.3.14 SetPressureOffset()** `void SetPressureOffset ( )`

Sets the pressure offset

**11.88.3.15 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 |

**11.88.3.16 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. |

**11.88.3.17 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     |

**11.88.3.18 SetValveActive()** `void SetValveActive (`  
    `uint16_t valve,`  
    `int valveActive )`

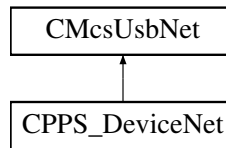
Sets the valve active/inactive state

Parameters

|                    |                  |
|--------------------|------------------|
| <i>valve</i>       | The valve number |
| <i>valveActive</i> | The valve state  |

## 11.89 CPPS\_DeviceNet Class Reference

Inheritance diagram for CPPS\_DeviceNet:



### Public Member Functions

- [CPPS\\_DeviceNet](#) (void)

### Properties

- [CPPS\\_FunctionNet](#)^ [PPS\\_Function](#) [get]
- [CMcsBusNet](#)^ [McsBus](#) [get]
- [CMcsBus\\_MotorControlNet](#)^ [McsBus\\_MotorControl](#) [get]
- [CMcsBus\\_SensorNet](#)^ [McsBus\\_Sensor](#) [get]

### Additional Inherited Members

#### 11.89.1 Constructor & Destructor Documentation

**11.89.1.1** [CPPS\\_DeviceNet\(\)](#) [CPPS\\_DeviceNet](#) (  
void )

#### 11.89.2 Property Documentation

**11.89.2.1** [McsBus](#) [CMcsBusNet](#)^ [McsBus](#) [get]

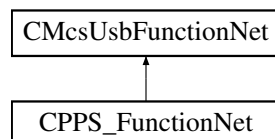
**11.89.2.2** [McsBus\\_MotorControl](#) [CMcsBus\\_MotorControlNet](#)^ [McsBus\\_MotorControl](#) [get]

**11.89.2.3** [McsBus\\_Sensor](#) [CMcsBus\\_SensorNet](#)^ [McsBus\\_Sensor](#) [get]

#### 11.89.2.4 PPS\_Function `CPPS_FunctionNet^ PPS_Function [get]`

### 11.90 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

##### 11.90.1 Constructor & Destructor Documentation



**11.90.1.1 CPPS\_FunctionNet()** [1/2] `CPPS_FunctionNet (`  
    `CMcsUsbNet^ mcsusb,`  
    `CMcsUsbFunctionPointerContainer^ cPPS_FunctionPointerContainer )`

**11.90.1.2 CPPS\_FunctionNet()** [2/2] `CPPS_FunctionNet (`  
    `CMcsUsbNet^ mcsusb )`

## 11.90.2 Member Function Documentation

**11.90.2.1 GetAnalogVoltage()** `unsigned short GetAnalogVoltage (`  
    `unsigned short index )`

**11.90.2.2 GetAnalogVoltages()** `void GetAnalogVoltages (`  
    `unsigned short index,`  
    `[System::Runtime::InteropServices::Out] unsigned short% minvoltage,`  
    `[System::Runtime::InteropServices::Out] unsigned short% maxvoltage )`

**11.90.2.3 GetBubbleState()** `unsigned short GetBubbleState ( )`

**11.90.2.4 GetDigitalIn()** `unsigned short GetDigitalIn (`  
    `unsigned short index )`

**11.90.2.5 GetPumpCouple()** `unsigned int GetPumpCouple ( )`

**11.90.2.6 GetPumpEnableSpeedRatio()** `unsigned int GetPumpEnableSpeedRatio ( )`

**11.90.2.7 GetPumpFastOnOff()** `unsigned int GetPumpFastOnOff (`  
    `unsigned short index )`

**11.90.2.8 GetPumpFastSpeed()** short GetPumpFastSpeed (   
 unsigned short *index* )

**11.90.2.9 GetPumpFunctionSpeeds()** void GetPumpFunctionSpeeds (   
 unsigned short *index*,   
 [System::Runtime::InteropServices::Out] short% *offspeed*,   
 [System::Runtime::InteropServices::Out] short% *onspeed* )

**11.90.2.10 GetPumpManualOnOff()** unsigned int GetPumpManualOnOff (   
 unsigned short *index* )

**11.90.2.11 GetPumpMaxSpeed()** unsigned short GetPumpMaxSpeed (   
 unsigned short *index* )

**11.90.2.12 GetPumpModeType()** [PP\\_Pump\\_Mode\\_Type\\_EnumNet](#) GetPumpModeType (   
 unsigned short *index* )

**11.90.2.13 GetPumpSpeedRatio()** int GetPumpSpeedRatio ( )

**11.90.2.14 GetPumpSpeedUnit()** int GetPumpSpeedUnit (   
 unsigned short *index* )

**11.90.2.15 GetSupplyVoltage()** unsigned short GetSupplyVoltage ( )

**11.90.2.16 GetUseBubble()** unsigned int GetUseBubble (   
 unsigned short *index* )

**11.90.2.17 SetAnalogVoltages()** void SetAnalogVoltages (   
    unsigned short *index*,  
    unsigned short *minvoltage*,  
    unsigned short *maxvoltage* )

**11.90.2.18 SetPumpCouple()** void SetPumpCouple (   
    unsigned int *i* )

**11.90.2.19 SetPumpEnableSpeedRatio()** void SetPumpEnableSpeedRatio (   
    unsigned int *enable* )

**11.90.2.20 SetPumpFastOnOff()** void SetPumpFastOnOff (   
    unsigned short *index*,  
    unsigned int *onoff* )

**11.90.2.21 SetPumpFastSpeed()** void SetPumpFastSpeed (   
    unsigned short *index*,  
    short *fastspeed* )

**11.90.2.22 SetPumpFunctionSpeeds()** void SetPumpFunctionSpeeds (   
    unsigned short *index*,  
    short *offspeed*,  
    short *onspeed* )

**11.90.2.23 SetPumpManualOnOff()** void SetPumpManualOnOff (   
    unsigned short *index*,  
    unsigned int *onoff* )

**11.90.2.24 SetPumpMaxSpeed()** void SetPumpMaxSpeed (   
    unsigned short *index*,  
    unsigned short *maxspeed* )

**11.90.2.25 SetPumpModeType()** `void SetPumpModeType (`  
    `unsigned short index,`  
    `PP_Pump_Mode_Type_EnumNet type )`

**11.90.2.26 SetPumpSpeedRatio()** `void SetPumpSpeedRatio (`  
    `int ratio )`

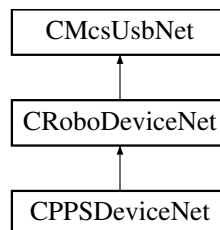
**11.90.2.27 SetPumpSpeedUnit()** `void SetPumpSpeedUnit (`  
    `unsigned short index,`  
    `int speedunit )`

**11.90.2.28 SetUseBubble()** `void SetUseBubble (`  
    `unsigned short index,`  
    `unsigned int usebubble )`

## 11.91 CPPSDeviceNet Class Reference

CPPS4plus1DeviceNet is the to control the MCS HLA device

Inheritance diagram for CPPSDeviceNet:



### Public Member Functions

- [CPPSDeviceNet](#) (void)

### Additional Inherited Members

#### 11.91.1 Detailed Description

CPPS4plus1DeviceNet is the to control the MCS HLA device

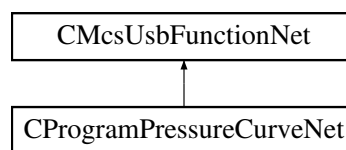
### 11.91.2 Constructor & Destructor Documentation

**11.91.2.1 CPPSDeviceNet()** `CPPSDeviceNet ( void )`

## 11.92 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

### 11.92.1 Detailed Description

`CProgramPressureCurveNet` is the class to program pressure curves

### 11.92.2 Constructor & Destructor Documentation

**11.92.2.1 CProgramPressureCurveNet()** `CProgramPressureCurveNet ( CMcsUsbNet^ mcsusb )`

Initializes a new instance of the `CPulseGeneratorFunctionNet` class.

**11.92.2.2** `~CProgramPressureCurveNet()` `virtual ~CProgramPressureCurveNet (`  
`void ) [virtual]`

**11.92.2.3** `!"CProgramPressureCurveNet()` `!CProgramPressureCurveNet (`  
`void )`

### 11.92.3 Member Function Documentation

**11.92.3.1** `GetRepeats()` `unsigned int GetRepeats (`  
`unsigned char busnumber,`  
`unsigned char busaddress,`  
`int32_t channel )`

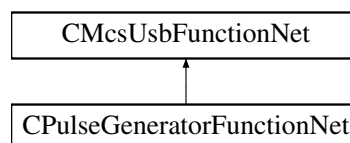
**11.92.3.2** `Program()` `void Program (`  
`unsigned char busnumber,`  
`unsigned char busaddress,`  
`int32_t channel,`  
`array< int32_t >^ pressures,`  
`array< int32_t >^ steps,`  
`array< int16_t >^ durations )`

**11.92.3.3** `SetRepeats()` `void SetRepeats (`  
`unsigned char busnumber,`  
`unsigned char busaddress,`  
`int32_t channel,`  
`uint32_t repeats )`

## 11.93 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

### 11.93.1 Detailed Description

[CPulseGeneratorFunctionNet](#) is the class to control the pulse generator for video tracking

### 11.93.2 Constructor & Destructor Documentation

**11.93.2.1 [CPulseGeneratorFunctionNet](#)() [1/2]** [CPulseGeneratorFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb,  
[CMcsUsbFunctionPointerContainer](#)<sup>^</sup> pPulseGeneratorFunctionPointerContainer )

Initializes a new instance of the [CPulseGeneratorFunctionNet](#) class.

**11.93.2.2 [CPulseGeneratorFunctionNet](#)() [2/2]** [CPulseGeneratorFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb )

**11.93.2.3 [~CPulseGeneratorFunctionNet](#)()** virtual [~CPulseGeneratorFunctionNet](#) ( ) [virtual]

**11.93.2.4** `!CPulseGeneratorFunctionNet()` `!CPulseGeneratorFunctionNet ( )`

### 11.93.3 Member Function Documentation

**11.93.3.1** `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 |

**11.93.3.2** `GetPeriod()` `int32_t GetPeriod (`  
    `int32_t generator_number )`

Reads the generator period

#### Parameters

|                         |                      |
|-------------------------|----------------------|
| <i>generator_number</i> | The generator number |
|-------------------------|----------------------|

#### Returns

The period

**11.93.3.3** `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

**11.93.3.4 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 |

**11.93.3.5 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           |

**11.93.3.6 SetPulseLength()** void SetPulseLength (   
     int32\_t *generator\_number*,   
     int32\_t *pulselength\_in\_10us* )

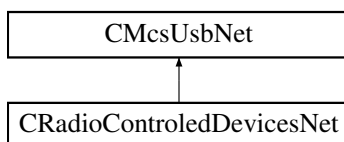
Writes the generator pulse length

**Parameters**

|                            |                      |
|----------------------------|----------------------|
| <i>generator_number</i>    | The generator number |
| <i>pulselength_in_10us</i> | The pulse length     |

**11.94 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

#### 11.94.1 Constructor & Destructor Documentation

**11.94.1.1 CRadioControlledDevicesNet()** [1/2] [CRadioControlledDevicesNet](#) (  
void )

**11.94.1.2 CRadioControlledDevicesNet()** [2/2] [CRadioControlledDevicesNet](#) (  
CRadioControlledDevices \* *pRadioControlled* ) [protected]

#### 11.94.2 Member Function Documentation

**11.94.2.1 ConnectDevice()** void [ConnectDevice](#) (  
unsigned short *sn* )

**11.94.2.2 DisConnectDevice()** void [DisConnectDevice](#) ( )

**11.94.2.3 GetDeviceNames()** `array<unsigned short> ^ GetDeviceNames ( )`

**11.94.2.4 GetFrequency()** `unsigned short GetFrequency ( )`

**11.94.2.5 HasRadioControl()** `bool HasRadioControl ( )`

**11.94.2.6 SetFrequency()** `void SetFrequency (   
 unsigned short frequency )`

**11.94.2.7 StillConnected()** `bool StillConnected ( )`

## 11.95 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](#)

### 11.95.1 Constructor & Destructor Documentation

**11.95.1.1 CRegionOfInterestRect()** `CRegionOfInterestRect (   
 int left,   
 int top,   
 int right,   
 int bottom )`

### 11.95.2 Member Function Documentation

**11.95.2.1 DeepCopy()** [CRegionOfInterestRect](#) ^ DeepCopy ( )

### 11.95.3 Member Data Documentation

**11.95.3.1 m\_Bottom** `int m_Bottom`

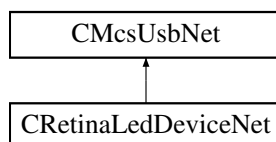
**11.95.3.2 m\_Left** `int m_Left`

**11.95.3.3 m\_Right** `int m_Right`

**11.95.3.4 m\_Top** `int m_Top`

## 11.96 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

### 11.96.1 Constructor & Destructor Documentation

**11.96.1.1 CRetinaLedDeviceNet()** `CRetinaLedDeviceNet ( )`

**11.96.1.2 ~CRetinaLedDeviceNet()** `~CRetinaLedDeviceNet ( )`

### 11.96.2 Member Function Documentation

**11.96.2.1 AddLoopEntry()** `unsigned int AddLoopEntry ( unsigned short repeats, unsigned short steps_back )`

**11.96.2.2 AddTableEntry()** `unsigned int AddTableEntry ( unsigned long long pattern )`

**11.96.2.3 ClearTable()** `unsigned int ClearTable ( )`

**11.96.2.4 GetTablepointer()** `unsigned int GetTablepointer ( int % position )`

**11.96.2.5 SetLED()** `unsigned int SetLED ( unsigned long long pattern )`

**11.96.2.6 SetLumi()** `unsigned int SetLumi ( int lumi )`

**11.96.2.7 SetPersistence()** unsigned int SetPersistence (   
 unsigned int *persistence* )

**11.96.2.8 SetRepeat()** unsigned int SetRepeat (   
 int *repeat* )

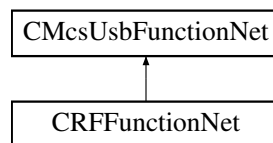
**11.96.2.9 SetTablepointer()** unsigned int SetTablepointer (   
 int *position* )

**11.96.2.10 SetTrigger()** unsigned int SetTrigger (   
 int *enable* )

## 11.97 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

### 11.97.1 Detailed Description

[CRFFunctionNet](#) is the class to control RF devices

### 11.97.2 Constructor & Destructor Documentation

**11.97.2.1 [CRFFunctionNet\(\)](#) [1/2]** [CRFFunctionNet](#) (  
     CMcsUsbNet^ mcsusb,  
     CMcsUsbFunctionPointerContainer^ pRFFunctionPointerContainer )

Initializes a new instance of the [CRFFunctionNet](#) class.

**11.97.2.2 [CRFFunctionNet\(\)](#) [2/2]** [CRFFunctionNet](#) (  
     CMcsUsbNet^ mcsusb )

**11.97.2.3 [~CRFFunctionNet\(\)](#)** virtual [~CRFFunctionNet](#) ( ) [virtual]

**11.97.2.4 ["!CRFFunctionNet\(\)](#)** [!CRFFunctionNet](#) ( )

### 11.97.3 Member Function Documentation

**11.97.3.1 [Connect\(\)](#)** void [Connect](#) (  
     uint32\_t sn )

connect to a RF device, use 0 to disconnect

**Parameters**

|           |                   |
|-----------|-------------------|
| <i>sn</i> | the serial number |
|-----------|-------------------|

**11.97.3.2 GetAvailableDeviceList()** `array<uint32_t> ^ GetAvailableDeviceList ( )`

get a list of available devices

**Returns**

array of devices

**11.97.3.3 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

**11.97.3.4 GetAvailableStateList()** `array<uint32_t> ^ GetAvailableStateList ( )`

get a list of the states of the available devices

**Returns**

array of states

**11.97.3.5 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

**11.97.3.6 GetBaseFrequency()** `uint32_t GetBaseFrequency ( CFirmwareDestinationNet destination )`

gets the base advertise frequency

**Parameters**

|                    |                          |
|--------------------|--------------------------|
| <i>destination</i> | the destination to query |
|--------------------|--------------------------|

**Returns**

the frequency

**11.97.3.7 GetConnectedDevice()** `uint32_t GetConnectedDevice ( )`

get connect RF device, 0 = no device connected

**Returns**

the serial number

**11.97.3.8 GetState()** `uint32_t GetState ( )`

get connection state

**Returns**

the state

**11.97.3.9 GetTestMode()** `uint32_t GetTestMode ( )`

gets test mode

**Returns**

the mode

**11.97.3.10 GetWorkingFrequency()** `uint32_t GetWorkingFrequency ( )`

gets the working frequency

**Returns**

the frequency

**11.97.3.11 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          |

**11.97.3.12 SetTestMode()** `void SetTestMode (`  
`uint32_t mode )`

set test mode

**Parameters**

|             |          |
|-------------|----------|
| <i>mode</i> | the mode |
|-------------|----------|

**11.97.3.13 SetWorkingFrequency()** `void SetWorkingFrequency (`  
`uint32_t frequency )`

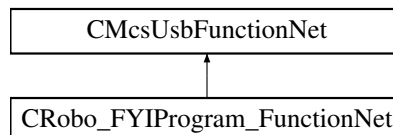
sets the working frequency

## Parameters

|                  |               |
|------------------|---------------|
| <i>frequency</i> | the frequency |
|------------------|---------------|

## 11.98 CRobo\_FYIPProgram\_FunctionNet Class Reference

Inheritance diagram for CRobo\_FYIPProgram\_FunctionNet:



## Public Member Functions

- [CRobo\\_FYIPProgram\\_FunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> robo\_↔ FYIPProgram\_FunctionPointerContainer)
- [CRobo\\_FYIPProgram\\_FunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- void [SetValve1](#) (unsigned char index, unsigned int valve1)
- unsigned int [GetValve1](#) (unsigned char index)
- void [SetValve2](#) (unsigned char index, unsigned int valve2)
- unsigned int [GetValve2](#) (unsigned char index)
- void [SetLength](#) (unsigned char index, int length)
- int [GetLength](#) (unsigned char index)
- void [Start](#) ()
- int [GetState](#) ()

## Additional Inherited Members

## 11.98.1 Constructor &amp; Destructor Documentation

**11.98.1.1 CRobo\_FYIPProgram\_FunctionNet() [1/2]** [CRobo\\_FYIPProgram\\_FunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb,  
[CMcsUsbFunctionPointerContainer](#)<sup>^</sup> robo\_FYIPProgram\_FunctionPointerContainer )

**11.98.1.2 CRobo\_FYIPProgram\_FunctionNet() [2/2]** [CRobo\\_FYIPProgram\\_FunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb )

## 11.98.2 Member Function Documentation

**11.98.2.1 GetLength()** `int GetLength (`  
`unsigned char index )`

**11.98.2.2 GetState()** `int GetState ( )`

**11.98.2.3 GetValve1()** `unsigned int GetValve1 (`  
`unsigned char index )`

**11.98.2.4 GetValve2()** `unsigned int GetValve2 (`  
`unsigned char index )`

**11.98.2.5 SetLength()** `void SetLength (`  
`unsigned char index,`  
`int length )`

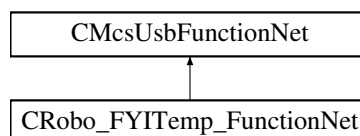
**11.98.2.6 SetValve1()** `void SetValve1 (`  
`unsigned char index,`  
`unsigned int valve1 )`

**11.98.2.7 SetValve2()** `void SetValve2 (`  
`unsigned char index,`  
`unsigned int valve2 )`

**11.98.2.8 Start()** `void Start ( )`

## 11.99 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

### 11.99.1 Constructor & Destructor Documentation

**11.99.1.1 CRobo\_FYITemp\_FunctionNet()** [CRobo\\_FYITemp\\_FunctionNet](#) (  
    [CMcsUsbNet](#)^ mcsusb )

### 11.99.2 Member Function Documentation

**11.99.2.1 GetICoeff()** int GetICoeff (  
    unsigned char index )

**11.99.2.2 GetMaxPower()** int GetMaxPower (  
    unsigned char index )

**11.99.2.3 GetPCoeff()** int GetPCoeff (  
    unsigned char index )

**11.99.2.4 GetRegulatorOnOff()** int GetRegulatorOnOff (  
    unsigned char index )

**11.99.2.5 GetSollTemp()** `int GetSollTemp (`  
    `unsigned char index )`

**11.99.2.6 SetICoeff()** `void SetICoeff (`  
    `unsigned char index,`  
    `int icoeff )`

**11.99.2.7 SetMaxPower()** `void SetMaxPower (`  
    `unsigned char index,`  
    `int power )`

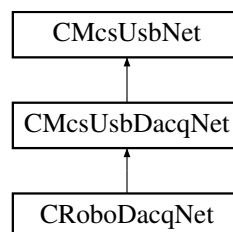
**11.99.2.8 SetPCoeff()** `void SetPCoeff (`  
    `unsigned char index,`  
    `int pcoeff )`

**11.99.2.9 SetRegulatorOnOff()** `void SetRegulatorOnOff (`  
    `unsigned char index,`  
    `int onoff )`

**11.99.2.10 SetSollTemp()** `void SetSollTemp (`  
    `unsigned char index,`  
    `int temp )`

## 11.100 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

#### 11.100.1 Constructor & Destructor Documentation

**11.100.1.1 CRoboDacqNet()** [1/2] [CRoboDacqNet](#) (  
void )

**11.100.1.2 CRoboDacqNet()** [2/2] [CRoboDacqNet](#) (  
[CRoboDeviceNet](#)^ *robodevice* )

#### 11.100.2 Member Function Documentation

**11.100.2.1 CancelTableLoop()** void [CancelTableLoop](#) ( )

**11.100.2.2 CancelTableLoopAndStopTable()** void [CancelTableLoopAndStopTable](#) ( )

**11.100.2.3 ClampAmpRestart()** void [ClampAmpRestart](#) ( )

**11.100.2.4 DoRamp()** void DoRamp (   
    int *startValue*,  
    int *endValue*,  
    int *duration*,  
    int *mode* )

**11.100.2.5 Emu\_GetCellCapacity()** int Emu\_GetCellCapacity ( )

**11.100.2.6 Emu\_GetCellPotential()** int Emu\_GetCellPotential ( )

**11.100.2.7 Emu\_GetCellResists()** int Emu\_GetCellResists ( )

**11.100.2.8 Emu\_GetElectrodeResists()** int Emu\_GetElectrodeResists ( )

**11.100.2.9 Emu\_GetNoise()** int Emu\_GetNoise ( )

**11.100.2.10 Emu\_SetCellCapacity()** void Emu\_SetCellCapacity (   
    int *emuCellCapacity* )

**11.100.2.11 Emu\_SetCellPotential()** void Emu\_SetCellPotential (   
    int *emuCellPotential* )

**11.100.2.12 Emu\_SetCellResists()** void Emu\_SetCellResists (   
    int *emuCellResist* )

**11.100.2.13 Emu\_SetElectrodeResists()** void Emu\_SetElectrodeResists (   
    int *emuElectrodeResist* )

**11.100.2.14 Emu\_SetNoise()** void Emu\_SetNoise (   
int *emuNoise* )

**11.100.2.15 GetAllDigout()** uint32\_t GetAllDigout ( )

**11.100.2.16 GetCapacityC()** int GetCapacityC ( )

**11.100.2.17 GetCapacityV()** int GetCapacityV ( )

**11.100.2.18 GetCapacityX()** int GetCapacityX ( )

**11.100.2.19 GetClampAmpSerialNumber()** unsigned int GetClampAmpSerialNumber ( )

**11.100.2.20 GetCommand()** int GetCommand (   
unsigned char *command* )

**11.100.2.21 GetConfigurationBit()** bool GetConfigurationBit (   
unsigned short *bit* )

**11.100.2.22 GetConfigurationBitAxc()** bool GetConfigurationBitAxc ( )

**11.100.2.23 GetConfigurationBitBlu\_Led()** bool GetConfigurationBitBlu\_Led ( )

**11.100.2.24 GetConfigurationBitBlu\_LedToggleFast()** bool GetConfigurationBitBlu\_LedToggleFast ( )

**11.100.2.25    GetConfigurationBitBlu\_LedToggleSlow()**    `bool GetConfigurationBitBlu_LedToggleSlow ( )`

**11.100.2.26    GetConfigurationBitCC\_Gen()**    `bool GetConfigurationBitCC_Gen ( )`

**11.100.2.27    GetConfigurationBitCV\_Gen()**    `bool GetConfigurationBitCV_Gen ( )`

**11.100.2.28    GetConfigurationBitRC\_Gen()**    `bool GetConfigurationBitRC_Gen ( )`

**11.100.2.29    GetConfigurationBitRed\_Led()**    `bool GetConfigurationBitRed_Led ( )`

**11.100.2.30    GetConfigurationBitRed\_LedSaturation()**    `bool GetConfigurationBitRed_LedSaturation ( )`

**11.100.2.31    GetConfigurationBitRed\_LedToggleFast()**    `bool GetConfigurationBitRed_LedToggleFast ( )`

**11.100.2.32    GetConfigurationBitRed\_LedToggleSlow()**    `bool GetConfigurationBitRed_LedToggleSlow ( )`

**11.100.2.33    GetConfigurationBitRelais()**    `bool GetConfigurationBitRelais ( )`

**11.100.2.34    GetConfigurationBitRV\_Gen()**    `bool GetConfigurationBitRV_Gen ( )`

**11.100.2.35    GetConfigurationBits()**    `unsigned int GetConfigurationBits ( )`

**11.100.2.36 GetConfigurationBitStream()** `bool GetConfigurationBitStream ( )`

**11.100.2.37 GetConfigurationBitSupply()** `bool GetConfigurationBitSupply ( )`

**11.100.2.38 GetCrossTalkOffset()** `int GetCrossTalkOffset ( )`

**11.100.2.39 GetCrossTalkOptimum()** `int GetCrossTalkOptimum ( )`

**11.100.2.40 GetDigout()** `bool GetDigout (`  
`uint16_t index )`

**11.100.2.41 GetDisplayText()** `String ^ GetDisplayText (`  
`int index )`

**11.100.2.42 GetDownsampleFactor()** `int GetDownsampleFactor (`  
`int index )`

**11.100.2.43 GetFilter()** `int GetFilter ( )`

**11.100.2.44 GetFilterCoeffs()** `array<int> ^ GetFilterCoeffs (`  
`int index )`

**11.100.2.45 GetIC()** `int GetIC ( )`

**11.100.2.46 GetIClamp()** `int GetIClamp ( )`

**11.100.2.47 GetICOffset()** `int GetICOffset ( )`

**11.100.2.48 GetIGain()** `int GetIGain ( )`

**11.100.2.49 GetNIC\_MS()** `int GetNIC_MS ( )`

**11.100.2.50 GetNUC\_MS()** `int GetNUC_MS ( )`

**11.100.2.51 GetNUV\_MS()** `int GetNUV_MS ( )`

**11.100.2.52 GetPGain()** `int GetPGain ( )`

**11.100.2.53 GetRecordingNumber()** `unsigned int GetRecordingNumber ( )`

**11.100.2.54 GetResistanceC()** `int GetResistanceC ( )`

**11.100.2.55 GetResistanceV()** `int GetResistanceV ( )`

**11.100.2.56 GetScreen()** `int GetScreen ( )`

**11.100.2.57 GetSimulation()** `unsigned int GetSimulation ( )`

**11.100.2.58 GetUC()** `int GetUC ( )`

**11.100.2.59 GetUClamp()** `int GetUClamp ( )`

**11.100.2.60 GetUCOffset()** `int GetUCOffset ( )`

**11.100.2.61 GetUpdateDisplay()** `int GetUpdateDisplay ( )`

**11.100.2.62 GetUV()** `int GetUV ( )`

**11.100.2.63 GetUVOffset()** `int GetUVOffset ( )`

**11.100.2.64 GetXGain()** `int GetXGain ( )`

**11.100.2.65 RunTable()** [1/2] `void RunTable ( )`

**11.100.2.66 RunTable()** [2/2] `void RunTable (`  
`int timeout )`

**11.100.2.67 SetAllDigout()** `void SetAllDigout (`  
`uint32_t value )`

**11.100.2.68 SetCommand()** `void SetCommand (`  
`unsigned char command,`  
`int value )`

**11.100.2.69 SetConfigurationBit()** void SetConfigurationBit (  
    unsigned short *bit*,  
    bool *value* )

**11.100.2.70 SetConfigurationBitAxc()** void SetConfigurationBitAxc (  
    bool *value* )

**11.100.2.71 SetConfigurationBitBlu\_Led()** void SetConfigurationBitBlu\_Led (  
    bool *value* )

**11.100.2.72 SetConfigurationBitBlu\_LedToggleFast()** void SetConfigurationBitBlu\_LedToggleFast (  
    bool *value* )

**11.100.2.73 SetConfigurationBitBlu\_LedToggleSlow()** void SetConfigurationBitBlu\_LedToggleSlow (  
    bool *value* )

**11.100.2.74 SetConfigurationBitCC\_Gen()** void SetConfigurationBitCC\_Gen (  
    bool *value* )

**11.100.2.75 SetConfigurationBitCV\_Gen()** void SetConfigurationBitCV\_Gen (  
    bool *value* )

**11.100.2.76 SetConfigurationBitRC\_Gen()** void SetConfigurationBitRC\_Gen (  
    bool *value* )

**11.100.2.77 SetConfigurationBitRed\_Led()** void SetConfigurationBitRed\_Led (  
    bool *value* )



**11.100.2.78 SetConfigurationBitRed\_LedSaturation()** void SetConfigurationBitRed\_LedSaturation ( bool *value* )

**11.100.2.79 SetConfigurationBitRed\_LedToggleFast()** void SetConfigurationBitRed\_LedToggleFast ( bool *value* )

**11.100.2.80 SetConfigurationBitRed\_LedToggleSlow()** void SetConfigurationBitRed\_LedToggleSlow ( bool *value* )

**11.100.2.81 SetConfigurationBitRelais()** void SetConfigurationBitRelais ( bool *value* )

**11.100.2.82 SetConfigurationBitRV\_Gen()** void SetConfigurationBitRV\_Gen ( bool *value* )

**11.100.2.83 SetConfigurationBitStream()** void SetConfigurationBitStream ( bool *value* )

**11.100.2.84 SetConfigurationBitSupply()** void SetConfigurationBitSupply ( bool *value* )

**11.100.2.85 SetCrossTalkOffset()** void SetCrossTalkOffset ( int *CrossTalk* )

**11.100.2.86 SetCrossTalkOptimum()** void SetCrossTalkOptimum ( int *cxOptimum* )

**11.100.2.87 SetDigout()** void SetDigout (   
    uint16\_t *index*,  
    bool *enable* )

**11.100.2.88 SetDisplayText()** void SetDisplayText (   
    int *index*,  
    String^ *displayText* )

**11.100.2.89 SetDownsampleFactor()** void SetDownsampleFactor (   
    int *index*,  
    int *downsample\_factor* )

**11.100.2.90 SetFilter()** void SetFilter (   
    int *filter* )

**11.100.2.91 SetFilterCoeffs()** void SetFilterCoeffs (   
    int *index*,  
    array< int >^ *coeffs* )

**11.100.2.92 SetIClamp()** void SetIClamp (   
    int *iClamp* )

**11.100.2.93 SetICOffset()** void SetICOffset (   
    int *ICOffset* )

**11.100.2.94 SetIGain()** void SetIGain (   
    int *iGain* )

**11.100.2.95 SetNoFilterCoeffs()** void SetNoFilterCoeffs (   
    int *index* )

**11.100.2.96 SetPGain()** void SetPGain (  
int *pGain* )

**11.100.2.97 SetRecordingNumber()** void SetRecordingNumber (  
unsigned int *recordingNumber* )

**11.100.2.98 SetScreen()** void SetScreen (  
int *screen* )

**11.100.2.99 SetSimulation()** void SetSimulation (  
unsigned int *enable* )

**11.100.2.100 SetTriggerMaskValue()** void SetTriggerMaskValue (  
unsigned int *mask*,  
unsigned int *value*,  
unsigned int *virtualDevice* )

**11.100.2.101 SetUClamp()** void SetUClamp (  
int *uClamp* )

**11.100.2.102 SetUCOffset()** void SetUCOffset (  
int *UCOffset* )

**11.100.2.103 SetUVOffset()** void SetUVOffset (  
int *UVOffset* )

**11.100.2.104 SetXGain()** void SetXGain (  
int *xGain* )

**11.100.2.105 StopTable()** [1/2] `void StopTable ( )`

**11.100.2.106 StopTable()** [2/2] `void StopTable (`  
`int timeout )`

**11.100.2.107 Table\_Wait()** `void Table_Wait (`  
`unsigned int tableWait )`

**11.100.2.108 TableDefBegin()** `void TableDefBegin ( )`

**11.100.2.109 TableDefEnd()** `void TableDefEnd ( )`

**11.100.2.110 UpdateDisplay()** `void UpdateDisplay ( )`

### 11.100.3 Member Data Documentation

**11.100.3.1 TriggerMask\_Default** `const unsigned int TriggerMask_Default = 0xFF00 [static]`

**11.100.3.2 TriggerValue\_MoveAbs** `const unsigned int TriggerValue_MoveAbs = COMMAND_ROBO_↔`  
`MOVEABS [static]`

**11.100.3.3 TriggerValue\_StartQueue** `const unsigned int TriggerValue_StartQueue = COMMAND_ROBO_↔`  
`QUEUE [static]`

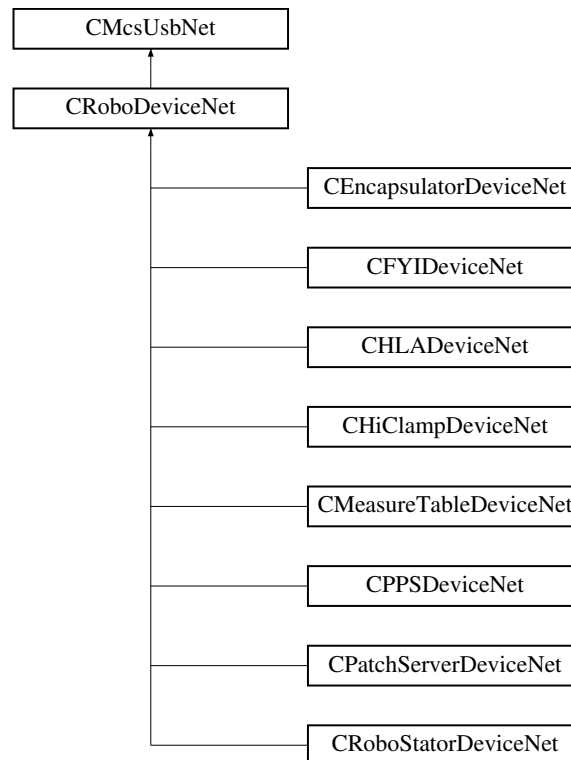
**11.100.3.4 VirtualDevice\_ContinousDacq** `const unsigned int VirtualDevice_ContinousDacq = 0`  
`[static]`

**11.100.3.5 VirtualDevice\_TableRun** `const unsigned int VirtualDevice_TableRun = 1 [static]`

## 11.101 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

### 11.101.1 Detailed Description

[CRoboDeviceNet](#) is the base class for all Robo platform based devices

### 11.101.2 Constructor & Destructor Documentation

**11.101.2.1 [CRoboDeviceNet](#)()** [CRoboDeviceNet](#) (  
void )

**11.101.2.2 [~CRoboDeviceNet](#)()** [~CRoboDeviceNet](#) (  
void )

### 11.101.3 Member Function Documentation

**11.101.3.1 [CancelPoolLoop](#)()** void [CancelPoolLoop](#) ( )



**11.101.3.2 CancelPoolLoopAndStopMovement()** `void CancelPoolLoopAndStopMovement ( )`

**11.101.3.3 EnableQueue()** `void EnableQueue (`  
`bool enable )`

**11.101.3.4 FindReference() [1/2]** `void FindReference (`  
`unsigned char busaddress,`  
`char axes )`

**11.101.3.5 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) |

**11.101.3.6 GetAirpressure()** `unsigned int GetAirpressure ( )`

**11.101.3.7 GetAirpressureLimit()** `void GetAirpressureLimit (`  
`[System::Runtime::InteropServices::Out] unsigned int% lowerpressure,`  
`[System::Runtime::InteropServices::Out] unsigned int% upperpressure )`

**11.101.3.8 GetAirValve()** `unsigned int GetAirValve ( )`

**11.101.3.9 GetCurrentAirvalve()** `unsigned int GetCurrentAirvalve ( )`

**11.101.3.10 GetCurrentAirvalveLimit()** `void GetCurrentAirvalveLimit (`  
    `[System::Runtime::InteropServices::Out] unsigned int% lowercurrent,`  
    `[System::Runtime::InteropServices::Out] unsigned int% uppercurrent )`

**11.101.3.11 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 |

**11.101.3.12 GetErrorAirpressure()** `unsigned int GetErrorAirpressure ( )`

**11.101.3.13 GetErrorCurrentAirvalve()** `unsigned int GetErrorCurrentAirvalve ( )`

**11.101.3.14 GetErrorVoltage12V()** `unsigned int GetErrorVoltage12V ( )`

**11.101.3.15 GetErrorVoltage5V()** `unsigned int GetErrorVoltage5V ( )`

**11.101.3.16 GetErrorVoltageAirvalve()** `unsigned int GetErrorVoltageAirvalve ( )`

**11.101.3.17 GetErrorVoltageRs485A()** `unsigned int GetErrorVoltageRs485A ( )`

**11.101.3.18 GetErrorVoltageRs485B()** unsigned int GetErrorVoltageRs485B ( )

**11.101.3.19 GetErrorVoltageValves()** unsigned int GetErrorVoltageValves ( )

**11.101.3.20 GetInMovement()** bool GetInMovement ( )

Low level command, gets the internal state "In Movement"

**11.101.3.21 GetMinPressure()** int GetMinPressure ( )

**11.101.3.22 GetMovementError()** uint32\_t GetMovementError ( )

Low level command, gets the error of the last movement end

**11.101.3.23 GetVoltage12V()** unsigned int GetVoltage12V ( )

**11.101.3.24 GetVoltage12VLimit()** void GetVoltage12VLimit (   
 [System::Runtime::InteropServices::Out] unsigned int% *lowervoltage*,   
 [System::Runtime::InteropServices::Out] unsigned int% *uppervoltage* )

**11.101.3.25 GetVoltage5V()** unsigned int GetVoltage5V ( )

**11.101.3.26 GetVoltage5VLimit()** void GetVoltage5VLimit (   
 [System::Runtime::InteropServices::Out] unsigned int% *lowervoltage*,   
 [System::Runtime::InteropServices::Out] unsigned int% *uppervoltage* )

**11.101.3.27 GetVoltageAirvalve()** unsigned int GetVoltageAirvalve ( )

**11.101.3.28 GetVoltageAirvalveLimit()** void GetVoltageAirvalveLimit (   
 [System::Runtime::InteropServices::Out] unsigned int% *lowervoltage*,   
 [System::Runtime::InteropServices::Out] unsigned int% *uppervoltage* )

**11.101.3.29 GetVoltageRs485A()** unsigned int GetVoltageRs485A ( )

**11.101.3.30 GetVoltageRs485ALimit()** void GetVoltageRs485ALimit (   
 [System::Runtime::InteropServices::Out] unsigned int% *lowervoltage*,   
 [System::Runtime::InteropServices::Out] unsigned int% *uppervoltage* )

**11.101.3.31 GetVoltageRs485B()** unsigned int GetVoltageRs485B ( )

**11.101.3.32 GetVoltageRs485BLimit()** void GetVoltageRs485BLimit (   
 [System::Runtime::InteropServices::Out] unsigned int% *lowervoltage*,   
 [System::Runtime::InteropServices::Out] unsigned int% *uppervoltage* )

**11.101.3.33 GetVoltageValves()** unsigned int GetVoltageValves ( )

**11.101.3.34 GetVoltageValvesLimit()** void GetVoltageValvesLimit (   
 [System::Runtime::InteropServices::Out] unsigned int% *lowervoltage*,   
 [System::Runtime::InteropServices::Out] unsigned int% *uppervoltage* )

**11.101.3.35 IsQueueEnabled()** bool IsQueueEnabled ( )

**11.101.3.36 IsQueueStarted()** bool IsQueueStarted ( )

**11.101.3.37 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 |

**11.101.3.38 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) |

**11.101.3.39 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 |

**11.101.3.40 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) |

**11.101.3.41 NullCommand()** `void NullCommand (`  
    `unsigned int marker )`

**11.101.3.42 SetAirpressureLimit()** `void SetAirpressureLimit (`  
    `unsigned int lowerpressure,`  
    `unsigned int upperpressure )`

**11.101.3.43 SetAirValve()** `void SetAirValve (`  
    `unsigned int onoff )`

**11.101.3.44 SetCurrentAirvalveLimit()** `void SetCurrentAirvalveLimit (`  
    `unsigned int lowercurrent,`  
    `unsigned int uppercurrent )`

**11.101.3.45 SetCurrentAndAir() [1/2]** `void SetCurrentAndAir (`  
    `unsigned char busaddress,`  
    `char axes,`  
    `unsigned short onoff )`

**11.101.3.46 SetCurrentAndAir() [2/2]** `void SetCurrentAndAir (`  
    `unsigned char busaddress,`  
    `char axes,`  
    `unsigned short onoff,`  
    `int timeout )`

**11.101.3.47 SetInMovement()** `void SetInMovement ( )`

Low level command, sets the internal state to "In Movement"

**11.101.3.48 SetMinPressure()** `void SetMinPressure (`  
`int pressure )`

**11.101.3.49 SetVoltage12VLimit()** `void SetVoltage12VLimit (`  
`unsigned int lowervoltage,`  
`unsigned int uppervoltage )`

**11.101.3.50 SetVoltage5VLimit()** `void SetVoltage5VLimit (`  
`unsigned int lowervoltage,`  
`unsigned int uppervoltage )`

**11.101.3.51 SetVoltageAirvalveLimit()** `void SetVoltageAirvalveLimit (`  
`unsigned int lowervoltage,`  
`unsigned int uppervoltage )`

**11.101.3.52 SetVoltageRs485ALimit()** `void SetVoltageRs485ALimit (`  
`unsigned int lowervoltage,`  
`unsigned int uppervoltage )`

**11.101.3.53 SetVoltageRs485BLimit()** `void SetVoltageRs485BLimit (`  
`unsigned int lowervoltage,`  
`unsigned int uppervoltage )`

**11.101.3.54 SetVoltageValvesLimit()** `void SetVoltageValvesLimit (`  
`unsigned int lowervoltage,`  
`unsigned int uppervoltage )`

**11.101.3.55 StartQueue()** `void StartQueue (`  
    `bool start )`

**11.101.3.56 StopMovement() [1/2]** `void StopMovement (`  
    `unsigned char busaddress,`  
    `char axes )`

**11.101.3.57 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) |

**11.101.3.58 WaitTimer()** `void WaitTimer (`  
    `uint32_t waittime,`  
    `int timeout )`

### 11.101.4 Member Data Documentation

**11.101.4.1 Axes\_I** `const char Axes_I = 2 [static]`

Bit pattern for i axis for axes argument

**11.101.4.2 Axes\_X** `const char Axes_X = 1 [static]`

Bit pattern for x axis for axes argument



**11.101.4.3 Axes\_Y** `const char Axes_Y = 2 [static]`

Bit pattern for y axis for axes argument

**11.101.4.4 Axes\_Z** `const char Axes_Z = 1 [static]`

Bit pattern for z axis for axes argument

**11.101.4.5 Axis\_I** `const byte Axis_I = 1 [static]`

Axis number of i for axis argument

**11.101.4.6 Axis\_X** `const byte Axis_X = 0 [static]`

Axis number of x for axis argument

**11.101.4.7 Axis\_Y** `const byte Axis_Y = 1 [static]`

Axis number of y for axis argument

**11.101.4.8 Axis\_Z** `const byte Axis_Z = 0 [static]`

Axis number of z for axis argument

**11.101.4.9 McsBus\_XY** `const byte McsBus_XY = 1 [static]`

McsBus address for the xy-plane

**11.101.4.10 McsBus\_ZI** `const byte McsBus_ZI = 2 [static]`

McsBus address for the z and i axes

**11.101.4.11 RoboError\_AnotherMaster** `const uint32_t RoboError_AnotherMaster = ( (0xA0110000L) | 0x0005 ) [static]`

**11.101.4.12 RoboError\_Base** `const uint32_t RoboError_Base = (0xA0110000L) [static]`

**11.101.4.13 RoboError\_CannotEscapeEndSwitch** `const uint32_t RoboError_CannotEscapeEndSwitch = ( (0xA0110000L) | 0x0009 ) [static]`

**11.101.4.14 RoboError\_CommandAlreadyInProgress** `const uint32_t RoboError_CommandAlreadyInProgress = ( (0xA0110000L) | 0x000A ) [static]`

**11.101.4.15 RoboError\_CommandNotPossible** `const uint32_t RoboError_CommandNotPossible = ( (0xA0110000L) | 0x0014 ) [static]`

**11.101.4.16 RoboError\_CommunicationTimeout** `const uint32_t RoboError_CommunicationTimeout = ( (0xA0110000L) | 0x0004 ) [static]`

**11.101.4.17 RoboError\_DacqNotReady** `const uint32_t RoboError_DacqNotReady = ( (0xA0110000L) | 0x0015 ) [static]`

**11.101.4.18 RoboError\_DLLMovementTimeout** `const uint32_t RoboError_DLLMovementTimeout = ( (0xA0110000L) | 0x1001 ) [static]`

**11.101.4.19 RoboError\_FindReferenceMethod** `const uint32_t RoboError_FindReferenceMethod = ( (0xA0110000L) | 0x0006 ) [static]`

**11.101.4.20 RoboError\_GilsonCommandPending** `const uint32_t RoboError_GilsonCommandPending = ( (0xA0110000L) | 0x0011 ) [static]`

**11.101.4.21 RoboError\_GilsonTimeout** `const uint32_t RoboError_GilsonTimeout = ( (0xA0110000L) | 0x000F ) [static]`

**11.101.4.22 RoboError\_GilsonWrondID** `const uint32_t RoboError_GilsonWrondID = ( (0xA0110000L) | 0x0010 ) [static]`

**11.101.4.23 RoboError\_McsBus\_UnknownCommand** `const uint32_t RoboError_McsBus_UnknownCommand = ( (0xA0110000L) | 0x003F ) [static]`

**11.101.4.24 RoboError\_NoEndSwitch** `const uint32_t RoboError_NoEndSwitch = ( (0xA0110000L) | 0x0008 ) [static]`

**11.101.4.25 RoboError\_NoMoreData** `const uint32_t RoboError_NoMoreData = ( (0xA0110000L) | 0x0016 ) [static]`

**11.101.4.26 RoboError\_NoReference** `const uint32_t RoboError_NoReference = ( (0xA0110000L) | 0x000B ) [static]`

**11.101.4.27 RoboError\_NoSpeedOrAcceleration** `const uint32_t RoboError_NoSpeedOrAcceleration = ( (0xA0110000L) | 0x0007 ) [static]`

**11.101.4.28 RoboError\_OverPressure** `const uint32_t RoboError_OverPressure = ( (0xA0110000L) | 0x000C ) [static]`

**11.101.4.29 RoboError\_ParameterNotAllowed** `const uint32_t RoboError_ParameterNotAllowed = ( (0xA0110000L) | 0x0012 ) [static]`

**11.101.4.30 RoboError\_PeristalticTimeout** `const uint32_t RoboError_PeristalticTimeout = ( (0xA0110000L) | 0x000E ) [static]`

**11.101.4.31 RoboError\_Phase0OutOfRange** `const uint32_t RoboError_Phase0OutOfRange = ( (0x←A0110000L) | 0x000D ) [static]`

**11.101.4.32 RoboError\_PollLoopCanceled** `const uint32_t RoboError_PollLoopCanceled = ( (0x←A0110000L) | 0x1002) [static]`

**11.101.4.33 RoboError\_PollLoopCanceledAndStopMovement** `const uint32_t RoboError_PollLoop←CanceledAndStopMovement = ( (0xA0110000L) | 0x1003) [static]`

**11.101.4.34 RoboError\_Pressure** `const uint32_t RoboError_Pressure = ( (0xA0110000L) | 0x0002 ) [static]`

**11.101.4.35 RoboError\_RangeExceeded** `const uint32_t RoboError_RangeExceeded = ( (0xA0110000L) | 0x0003 ) [static]`

**11.101.4.36 RoboError\_StateChangeNotPossible** `const uint32_t RoboError_StateChangeNotPossible = ( (0xA0110000L) | 0x0013 ) [static]`

**11.101.4.37 RoboError\_Timeout** `const uint32_t RoboError_Timeout = ( (0xA0110000L) | 0x0001 ) [static]`

**11.101.4.38 RoboError\_UnknownCommand** `const uint32_t RoboError_UnknownCommand = ( (0x←A0110000L) ) [static]`

## 11.101.5 Property Documentation

**11.101.5.1 McsBus** `CMcsBusNet^ McsBus [get]`

**11.101.5.2 McsBus\_MotorControl** [CMcsBus\\_MotorControlNet](#)<sup>^</sup> McsBus\_MotorControl [get]

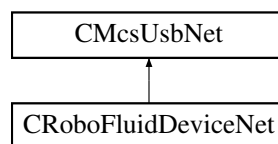
**11.101.5.3 RoboMainLowLevelCommand** [RoboMainLowLevelCommands](#)<sup>^</sup> RoboMainLowLevelCommand [get]

## 11.101.6 Event Documentation

**11.101.6.1 RoboStatusEvent** [RoboStatusEventDelegate](#)<sup>^</sup> RoboStatusEvent

## 11.102 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

### 11.102.1 Constructor & Destructor Documentation

**11.102.1.1 CRoboFluidDeviceNet()** [CRoboFluidDeviceNet](#) (  
void )

**11.102.1.2 ~CRoboFluidDeviceNet()** [~CRoboFluidDeviceNet](#) (  
void )

### 11.102.2 Member Function Documentation

**11.102.2.1 CloseAllValves()** void CloseAllValves ( )

**11.102.2.2 GetPumpSpeed()** short GetPumpSpeed (  
int *index* )

**11.102.2.3 GetSingleValve()** bool GetSingleValve (  
int *valve* )

Query the state of a valve.

#### Parameters

|              |                     |
|--------------|---------------------|
| <i>valve</i> | number of valve /*! |
|--------------|---------------------|

#### Returns

state of the valve

**11.102.2.4 GetValve()** `int GetValve ( )`

Query the state of the valves.

**Returns**

the current state of the valves as a bit pattern.

**11.102.2.5 IsPumpMotorOn()** `bool IsPumpMotorOn (`  
`int index )`**11.102.2.6 PumpOff()** `void PumpOff (`  
`int index )`**11.102.2.7 PumpOn()** `void PumpOn (`  
`int index,`  
`short speed )`**11.102.2.8 SetPumpSpeed()** `void SetPumpSpeed (`  
`int index,`  
`short speed )`**11.102.2.9 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           |

**11.102.2.10 SetValve()** `void SetValve (`  
`int value )`

Open or Close valves.

## Parameters

|              |                                             |
|--------------|---------------------------------------------|
| <i>value</i> | bit pattern of valves which should be open. |
|--------------|---------------------------------------------|

## 11.102.3 Member Data Documentation

**11.102.3.1 m\_pMcsBus\_MotorControlNet** [CMcsBus\\_MotorControlNet](#) ^ m\_pMcsBus\_MotorControlNet  
[protected]

**11.102.3.2 m\_pRoboFluidDevice** [CRoboFluidDevice\\*](#) m\_pRoboFluidDevice [protected]

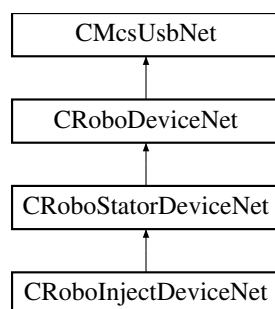
## 11.102.4 Property Documentation

**11.102.4.1 McsBus\_MotorControl** [CMcsBus\\_MotorControlNet](#) ^ McsBus\_MotorControl [get]

## 11.103 CRoboInjectDeviceNet Class Reference

[CRoboInjectDeviceNet](#) is the to control the MCS RoboInject device

Inheritance diagram for CRoboInjectDeviceNet:



## Public Member Functions

- [CRoboInjectDeviceNet](#) (void)



## Additional Inherited Members

### 11.103.1 Detailed Description

[CRobolInjectDeviceNet](#) is the to control the MCS RobolInject device

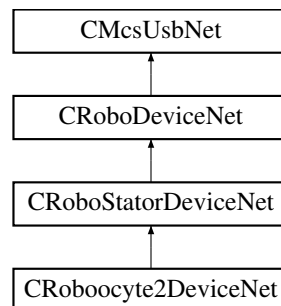
### 11.103.2 Constructor & Destructor Documentation

**11.103.2.1 CRobolInjectDeviceNet()** [CRobolInjectDeviceNet](#) (  
void )

## 11.104 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

### 11.104.1 Detailed Description

[CRoboocyte2DeviceNet](#) is the class to control the MCS Roboocyte2 device

## 11.104.2 Constructor & Destructor Documentation

**11.104.2.1 CRoboocyte2DeviceNet()** `CRoboocyte2DeviceNet ( void )`

## 11.104.3 Member Function Documentation

**11.104.3.1 GetAxisLED()** `bool GetAxisLED ( )`

**11.104.3.2 GetGilsonDevice()** `CGilsonDeviceNet ^ GetGilsonDevice ( )`

**11.104.3.3 GetMcsBus\_Extension()** `CMcsBus_ExtensionNet ^ GetMcsBus_Extension ( )`

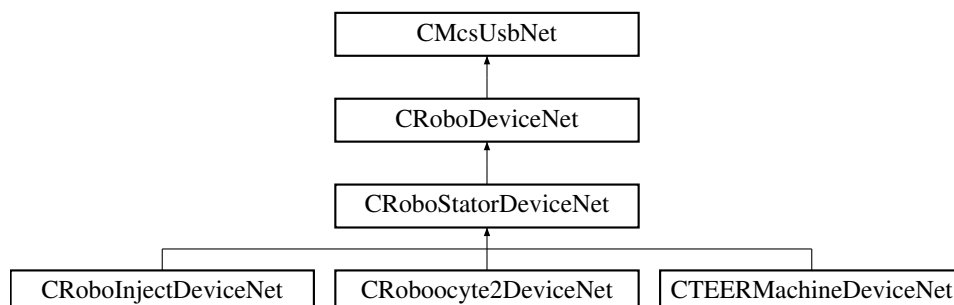
**11.104.3.4 GetRoboDacq()** `CRoboDacqNet ^ GetRoboDacq ( )`

**11.104.3.5 GetRoboFluidDevice()** `CRoboFluidDeviceNet ^ GetRoboFluidDevice ( )`

**11.104.3.6 SetAxisLED()** `void SetAxisLED ( bool onoff )`

## 11.105 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

### 11.105.1 Constructor & Destructor Documentation

**11.105.1.1 CRoboStatorDeviceNet()** `CRoboStatorDeviceNet ( void )`

### 11.105.2 Member Function Documentation

**11.105.2.1 FindReferenceI()** [1/2] `void FindReferenceI ( )`

**11.105.2.2 FindReferenceI()** [2/2] `void FindReferenceI ( int timeout )`

**11.105.2.3 FindReferenceXY()** [1/2] `void FindReferenceXY ( )`

**11.105.2.4 FindReferenceXY()** [2/2] `void FindReferenceXY ( int timeout )`

**11.105.2.5 FindReferenceZ()** [1/2] `void FindReferenceZ ( )`

**11.105.2.6 FindReferenceZ()** [2/2] `void FindReferenceZ ( int timeout )`

**11.105.2.7 GetCurrentPositionI()** `void GetCurrentPositionI ( [System::Runtime::InteropServices::Out] int% i )`

**11.105.2.8 GetCurrentPositionXY()** void GetCurrentPositionXY (   
 [System::Runtime::InteropServices::Out] int% x,   
 [System::Runtime::InteropServices::Out] int% y )

**11.105.2.9 GetCurrentPositionZ()** void GetCurrentPositionZ (   
 [System::Runtime::InteropServices::Out] int% z )

**11.105.2.10 HasRefI()** unsigned char HasRefI ( )

**11.105.2.11 HasRefXY()** unsigned char HasRefXY ( )

**11.105.2.12 HasRefZ()** unsigned char HasRefZ ( )

**11.105.2.13 MoveAbsI()** [1/2] void MoveAbsI (   
 int i )

**11.105.2.14 MoveAbsI()** [2/2] void MoveAbsI (   
 int i,   
 int timeout )

**11.105.2.15 MoveAbsXY()** [1/2] void MoveAbsXY (   
 int x,   
 int y )

**11.105.2.16 MoveAbsXY()** [2/2] void MoveAbsXY (   
 int x,   
 int y,   
 int timeout )

**11.105.2.17 MoveAbsZ()** [1/2] void MoveAbsZ (  
int z )

**11.105.2.18 MoveAbsZ()** [2/2] void MoveAbsZ (  
int z,  
int timeout )

**11.105.2.19 SetAccelerationI()** void SetAccelerationI (  
int a )

**11.105.2.20 SetAccelerationNativeI()** void SetAccelerationNativeI (  
int a )

**11.105.2.21 SetAccelerationNativeXY()** void SetAccelerationNativeXY (  
int a )

**11.105.2.22 SetAccelerationNativeZ()** void SetAccelerationNativeZ (  
int a )

**11.105.2.23 SetAccelerationXY()** void SetAccelerationXY (  
int a )

**11.105.2.24 SetAccelerationZ()** void SetAccelerationZ (  
int a )

**11.105.2.25 SetCurrentAndAirXY()** [1/2] void SetCurrentAndAirXY (  
unsigned short onoff )

**11.105.2.26 SetCurrentAndAirXY()** [2/2] void SetCurrentAndAirXY (  
    unsigned short *onoff*,  
    int *timeout* )

**11.105.2.27 SetSpeedI()** void SetSpeedI (  
    int *v* )

**11.105.2.28 SetSpeedNativeI()** void SetSpeedNativeI (  
    int *v* )

**11.105.2.29 SetSpeedNativeXY()** void SetSpeedNativeXY (  
    int *v* )

**11.105.2.30 SetSpeedNativeZ()** void SetSpeedNativeZ (  
    int *v* )

**11.105.2.31 SetSpeedXY()** void SetSpeedXY (  
    int *v* )

**11.105.2.32 SetSpeedZ()** void SetSpeedZ (  
    int *v* )

**11.105.2.33 SetVelocityI()** void SetVelocityI (  
    int *v* )

**11.105.2.34 SetVelocityXY()** void SetVelocityXY (  
    int *v* )

**11.105.2.35 SetVelocityZ()** `void SetVelocityZ (`  
    `int v )`

**11.105.2.36 StopMovementI()** [1/2] `void StopMovementI ( )`

**11.105.2.37 StopMovementI()** [2/2] `void StopMovementI (`  
    `int timeout )`

**11.105.2.38 StopMovementXY()** [1/2] `void StopMovementXY ( )`

**11.105.2.39 StopMovementXY()** [2/2] `void StopMovementXY (`  
    `int timeout )`

**11.105.2.40 StopMovementZ()** [1/2] `void StopMovementZ ( )`

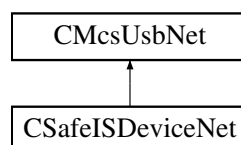
**11.105.2.41 StopMovementZ()** [2/2] `void StopMovementZ (`  
    `int timeout )`

### 11.105.3 Property Documentation

**11.105.3.1 RoboMainStatorLowLevelCommand** `RoboMainStatorLowLevelCommands^ RoboMainStator←`  
`LowLevelCommand [get]`

## 11.106 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

### 11.106.1 Detailed Description

### 11.106.2 Constructor & Destructor Documentation

**11.106.2.1 CSafeISDeviceNet()** [CSafeISDeviceNet](#) (  
void )

Initializes a new instance of the [CSafeISDeviceNet](#) class.

**11.106.2.2 ~CSafeISDeviceNet()** [~CSafeISDeviceNet](#) (  
void )

Releases unmanaged resources and performs other cleanup operations before the [CSafeISDeviceNet](#) is reclaimed by garbage collection.

### 11.106.3 Member Function Documentation

**11.106.3.1 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 |
|-----------------|------------------------------------------------------------------------------|

**11.106.3.2 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. |
|------------------|----------------------------------|

**11.106.3.3 SetDacMode()** `void SetDacMode (`  
`unsigned char mode )`

Sets the DAC mode.

#### Parameters

|             |                                           |
|-------------|-------------------------------------------|
| <i>mode</i> | The mode: 0 = Impedance ; 1 = Amperometry |
|-------------|-------------------------------------------|

**11.106.3.4 SetDacPeriode()** `void SetDacPeriode (`  
`unsigned int periode )`

Sets the DAC periode.

#### Parameters

|                |                               |
|----------------|-------------------------------|
| <i>periode</i> | The periode in units of 10µs. |
|----------------|-------------------------------|

**11.106.3.5 SetDacPulseform()** `void SetDacPulseform ( array< short >^ pulseform )`

Sets the DAC pulseform.

#### Parameters

|                                  |                |
|----------------------------------|----------------|
| <i><a href="#">pulseform</a></i> | The pulseform. |
|----------------------------------|----------------|

**11.106.3.6 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 |
|---------------------------------|----------------------------------------------|

## 11.106.4 Property Documentation

**11.106.4.1 DacqDevice** [CMcsUsbDacqNet](#)^ DacqDevice [get]

Gets the [CMcsUsbDacqNet](#). Use this to control the data aquisition.

**11.106.4.2 FluidControlDevice** [CFluidControlDeviceNet](#)^ FluidControlDevice [get]

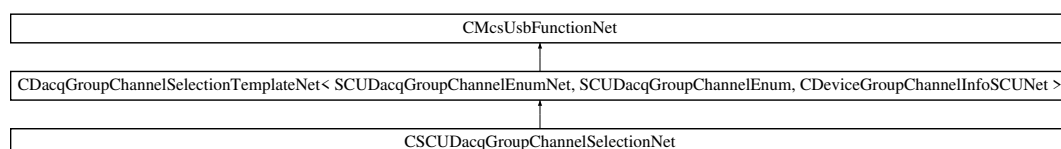
Gets the [CFluidControlDeviceNet](#). Use this to control the valves. Only SetSingleValve is implemented for [CSafeISDeviceNet](#).

**11.106.4.3 RoboDevice** [CRoboDeviceNet](#)^ RoboDevice [get]

Gets the [CRoboDeviceNet](#). Use this to control the syringe.

## 11.107 CSCUDacqGroupChannelSelectionNet Class Reference

Inheritance diagram for CSCUDacqGroupChannelSelectionNet:



## Public Member Functions

- [CSCUDacqGroupChannelSelectionNet](#) ([CMcsUsbNet](#)^ mcsusb)

## Additional Inherited Members

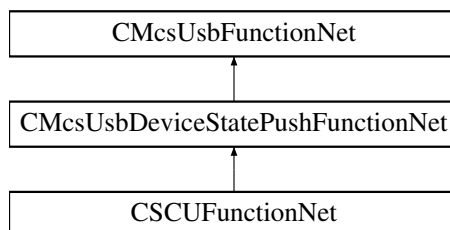
### 11.107.1 Constructor & Destructor Documentation

#### 11.107.1.1 CSCUDacqGroupChannelSelectionNet() [CSCUDacqGroupChannelSelectionNet](#) ([CMcsUsbNet](#)^ mcsusb )

## 11.108 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*

## 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

### 11.108.1 Detailed Description

[CSCUFunctionNet](#) is the class to control the SCU device

## 11.108.2 Constructor & Destructor Documentation

**11.108.2.1 CSCUFunctionNet()** [1/2] `CSCUFunctionNet (`  
`CMcsUsbNet^ mcsusb,`  
`CMcsUsbFunctionPointerContainer^ pSCUFunctionPointerContainer )`

Initializes a new instance of the `CSCUFunctionNet` class.

**11.108.2.2 CSCUFunctionNet()** [2/2] `CSCUFunctionNet (`  
`CMcsUsbNet^ mcsusb )`

**11.108.2.3 ~CSCUFunctionNet()** `virtual ~CSCUFunctionNet ( ) [virtual]`

**11.108.2.4 !CSCUFunctionNet()** `!CSCUFunctionNet ( )`

## 11.108.3 Member Function Documentation

**11.108.3.1 AutomaticAnalogOut()** `void AutomaticAnalogOut (`  
`bool automatic )`

Sets automatic source channel selection

Parameters

|                        |           |
|------------------------|-----------|
| <code>automatic</code> | Automatic |
|------------------------|-----------|

**11.108.3.2 EnableAnalogOut()** `void EnableAnalogOut (`  
`bool enable )`

Enables AnalogOut globally

Parameters

|                     |        |
|---------------------|--------|
| <code>enable</code> | Enable |
|---------------------|--------|

**11.108.3.3 GetAnalogOutADCRange()** `uint32_t GetAnalogOutADCRange ( )`

Gets the analog out ADC range

**Returns**

Range

**11.108.3.4 GetAnalogOutChannels()** `uint32_t GetAnalogOutChannels (   
uint32_t out_channel )`

Get the connected source channel number for a certain output channel

**Parameters**

|                          |                       |
|--------------------------|-----------------------|
| <code>out_channel</code> | Output channel number |
|--------------------------|-----------------------|

**Returns**

Source channel number

**11.108.3.5 GetAnalogOutDACRange()** `AnalogOut_DAC_Range_EnumNet GetAnalogOutDACRange ( )`

Gets the analog out DAC range

**Returns**

Range

**11.108.3.6 GetAvailableHeadstages()** `uint32_t GetAvailableHeadstages ( )`

Gets a bitmap of available headstages.

**Returns**

The bitmap of available headstages.

**11.108.3.7 GetFilterProperties()** `array<CFilterPropertyNet^> ^ GetFilterProperties (   
SCUDacqGroupChannelEnumNet GroupID,   
int filterConfigurations_Length )`

Gets multiple filter properties



## Parameters

|                                    |                                             |
|------------------------------------|---------------------------------------------|
| <i>GroupID</i>                     | The group ID                                |
| <i>filterConfigurations_Length</i> | The maximal length of filterConfigurations. |

## Returns

array of filter properties

**11.108.3.8 GetFilterProperty()** `CFilterPropertyNet ^ GetFilterProperty ( SCUDacqGroupChannelEnumNet GroupID, uint32_t FilterNumber )`

Gets the filter property

## Parameters

|                     |                   |
|---------------------|-------------------|
| <i>GroupID</i>      | The group ID      |
| <i>FilterNumber</i> | The filter number |

## Returns

The filter property

**11.108.3.9 GetHeadstageAdcBits()** `uint32_t GetHeadstageAdcBits ( uint32_t Headstage )`

Gets the Number of ADC bits for a given headstage.

## Parameters

|                  |                         |
|------------------|-------------------------|
| <i>Headstage</i> | The headstage to query. |
|------------------|-------------------------|

## Returns

The number of bits the ADC has for the given headstage.

**11.108.3.10 GetHeadstageAdcRangeInMicroVolt()** `uint32_t GetHeadstageAdcRangeInMicroVolt ( uint32_t Headstage )`

Gets the ADC Range in uV for a given headstage.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <i>Headstage</i> | The headstage to query. |
|------------------|-------------------------|

**Returns**

The ADC Range in uV for the given headstage.

**11.108.3.11 GetHeadstageDacBits()** `uint32_t GetHeadstageDacBits (uint32_t Headstage )`

Gets the Number of DAC bits for a given headstage.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <i>Headstage</i> | The headstage to query. |
|------------------|-------------------------|

**Returns**

The number of bits the DAC has for the given headstage.

**11.108.3.12 GetHeadstageDacCurrentRangeInMicroAmpere()** `uint32_t GetHeadstageDacCurrentRange↔InMicroAmpere (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.

**11.108.3.13 GetHeadstageDacCurrentResolutionInNanoAmpere()** `uint32_t GetHeadstageDacCurrent↔ResolutionInNanoAmpere (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.

**11.108.3.14 GetHeadstageDacVoltageRangeInMilliVolt()** `uint32_t GetHeadstageDacVoltageRangeIn↔  
MilliVolt (`  
`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.

**11.108.3.15 GetHeadstageDacVoltageResolutionInMicroVolt()** `uint32_t GetHeadstageDacVoltage↔  
ResolutionInMicroVolt (`  
`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.

**11.108.3.16 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.

**11.108.3.17 GetHeadstageGainInPer mille()** `uint32_t GetHeadstageGainInPer mille (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.

**11.108.3.18 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.

**11.108.3.19 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.

**11.108.3.20 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.

**11.108.3.21 GetHeadstageNumberOfStimulationChannels()** `uint32_t GetHeadstageNumberOfStimulationChannels (uint32_t Headstage )`

Gets the number of stimulation channels for a given headstage.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <i>Headstage</i> | The headstage to query. |
|------------------|-------------------------|

**Returns**

The number of stimulation channels the headstage has.

**11.108.3.22 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.

**11.108.3.23 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.

**11.108.3.24 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.

**11.108.3.25 GetMaxNumberOfHeadstages()** `uint32_t GetMaxNumberOfHeadstages ( )`

Gets the maximal number of headstages.

**Returns**

The maximal number of headstages.

**11.108.3.26 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.

**11.108.3.27 GetReferenceElectrodeMode()** `ReferenceElectrodeModeEnumNet GetReferenceElectrodeMode ( uint32_t Headstage )`

Gets the mode for the reference electrode

#### Parameters

|                  |                      |
|------------------|----------------------|
| <i>Headstage</i> | The headstage number |
|------------------|----------------------|

#### Returns

The mode

**11.108.3.28 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

**11.108.3.29 HasAnalogOut()** `bool HasAnalogOut ( )`

Has AnalogOut hardware

#### Returns

Enabled

**11.108.3.30 HasGalvanicIsolation()** `bool HasGalvanicIsolation ( )`

Has galvanic isolated hardware

Returns

Enabled

**11.108.3.31 HasHSPowerSwitch()** `bool HasHSPowerSwitch ( )`

Has SCU HS power switch

Returns

Has Switch

**11.108.3.32 IsAnalogOutEnabled()** `bool IsAnalogOutEnabled ( )`

Is AnalogOut enabled

Returns

Enabled

**11.108.3.33 IsAutomaticAnalogOut()** `bool IsAutomaticAnalogOut ( )`

Is Automatic source channel selection selected

Returns

Automatic

**11.108.3.34 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.

**11.108.3.35 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.

**11.108.3.36 IsInDacqLegacyMode()** `bool IsInDacqLegacyMode ( )`

Is the SCU in legacy mode

**Returns**

Is Enabled

**11.108.3.37 OnGetAvailableHeadstages()** `delegate void OnGetAvailableHeadstages (`  
    `uint32_t AvailableHeadstages )`

**11.108.3.38 OnIsHeadstageAvailable()** `delegate void OnIsHeadstageAvailable (`  
    `uint32_t Headstage,`  
    `bool available )`

**11.108.3.39 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. |

**11.108.3.40 SetAnalogOutADCRange()** `void SetAnalogOutADCRange (`  
    `uint32_t range )`

Sets the analog out ADC range

## Parameters

|              |       |
|--------------|-------|
| <i>range</i> | Range |
|--------------|-------|

**11.108.3.41 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 |

**11.108.3.42 SetAnalogOutDACRange()** `void SetAnalogOutDACRange (`  
    `AnalogueOut_DAC_Range_EnumNet range )`

Sets the analog out DAC range

## Parameters

|              |       |
|--------------|-------|
| <i>range</i> | Range |
|--------------|-------|

**11.108.3.43 SetDacqLegacyMode()** `void SetDacqLegacyMode (`  
    `bool enable )`

Enable the SCU legacy mode

## Parameters

|               |        |
|---------------|--------|
| <i>enable</i> | Enable |
|---------------|--------|

**11.108.3.44 SetHeadstageFrameCyclesToComparePermanent()** `void SetHeadstageFrameCyclesToComparePermanent (`  
`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. |

**11.108.3.45 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. |

**11.108.3.46 SetHeadstageNumberOfAnalogChannelsPermanent()** `void SetHeadstageNumberOfAnalogChannelsPermanent (`  
`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 |

**11.108.3.47 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. |

**11.108.3.48 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. |

**11.108.3.49 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             |

**11.108.3.50 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  |

### 11.108.4 Event Documentation

**11.108.4.1 GetAvailableHeadstagesEvent** `OnGetAvailableHeadstages^ GetAvailableHeadstagesEvent [add], [remove], [raise]`

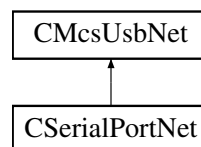
Event fires when the bitmap of available headstages has changed

**11.108.4.2 IsHeadstageAvailableEvent** `OnIsHeadstageAvailable^ IsHeadstageAvailableEvent [add], [remove], [raise]`

Event fires when 'true' if the headstage is connected for the headstage to query has changed

## 11.109 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

### 11.109.1 Constructor & Destructor Documentation

**11.109.1.1 CSerialPortNet()** `CSerialPortNet ( void )`

### 11.109.2 Member Function Documentation

**11.109.2.1 GetBytesAvailable()** `int GetBytesAvailable (`  
`void )`

**11.109.2.2 Receive()** [1/2] `array<byte> ^ Receive (`  
`int length )`

**11.109.2.3 Receive()** [2/2] `array<byte> ^ Receive (`  
`void )`

**11.109.2.4 ReceiveString()** [1/2] `String ^ ReceiveString (`  
`int length )`

**11.109.2.5 ReceiveString()** [2/2] `String ^ ReceiveString (`  
`void )`

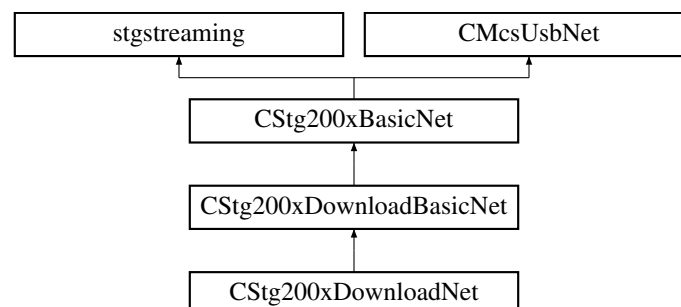
**11.109.2.6 Send()** [1/2] `void Send (`  
`array< byte > ^ buffer )`

**11.109.2.7 Send()** [2/2] `void Send (`  
`String ^ command )`

### 11.110 CStg200xBasicNet Class Reference

Base class for the Stg200x.

Inheritance diagram for CStg200xBasicNet:



## Public Member Functions

- virtual [~CStg200xBasicNet](#) ()  
*The destructor.*
- void [SetOutputRate](#) (uint32\_t rate)  
*Change the output rate of the STG. Valid rates are from 1000 Hz to 50000 Hz.*
- uint32\_t [GetOutputRate](#) ()  
*Queries the output rate of the STG. Valid rates are from 1000 Hz to 50000 Hz.*
- void [SendStart](#) (uint32\_t triggermap)  
*Start (Trigger) the STG. The startup delay is in the range of a few ms.*
- void [SendStop](#) (uint32\_t triggermap)  
*Stop some or all triggers of the STG.*
- void [SendStop](#) (uint32\_t triggermap, int options)  
*Stop some or all triggers of the STG.*
- void [GetStgVersionInfo](#) ([Out]String^% SwVersion, [Out]String^% HwVersion)  
*Queries software and hardware version.*
- void [GetAnalogRanges](#) (int channel, [Out]int% URange, [Out]int% IRange)  
*Gets the range of the analog outputs.*
- void [GetAnalogResolution](#) (int channel, [Out]int% URes, [Out]int% IRes)  
*Gets the resolution of the analog outputs.*
- virtual int32\_t [GetDACResolution](#) ()  
*Gets number of bits of the DAC resolution.*
- virtual int32\_t [GetVoltageRangeInMicroVolt](#) (uint32\_t channel)  
*Gets the Voltage Range of the specified channel in Microvolts.*
- virtual int32\_t [GetVoltageResolutionInMicroVolt](#) (uint32\_t channel)  
*Gets the Voltage Resolution of the specified channel in Microvolts.*
- virtual int32\_t [GetCurrentRangeInNanoAmp](#) (uint32\_t channel)  
*Gets the Current Range of the specified channel in Nanoamps.*
- virtual int32\_t [GetCurrentResolutionInNanoAmp](#) (uint32\_t channel)  
*Gets the Current Resolution of the specified channel in Nanoamps.*
- void [GetStgProgramInfo](#) ([Out]bool% IsProgrammed, [Out]System::Runtime::InteropServices::ComTypes::FILETIME% timestamp, [Out]String^% filename, [Out]Guid% guid)  
*Queries Download information from the STG. If download information was stored by the use of [SetStgProgramInfo](#), this function can be used to retrieve it.*
- void [GetStgProgramInfo](#) ([Out]bool% IsProgrammed, [Out]DateTime% timestamp, [Out]String^% filename, [Out]Guid% guid)  
*Queries Download information from the STG. If download information was stored by the use of [SetStgProgramInfo](#), this function can be used to retrieve it.*
- void [SetStgProgramInfo](#) (DateTime timestamp, String^ filename, Guid guid)  
*Store Download information in the STG. This function can be used to store the filename and timestamp of the last download for later query.*
- uint32\_t [GetAvailableMemory](#) ()  
*Gets the amount of memory available in the currently selected segment of the STG.*
- uint32\_t [GetTotalMemory](#) ()  
*Gets the total amount of memory available on the STG (all segments).*
- virtual uint32\_t [GetNumberOfAnalogChannels](#) ()  
*Gets the Number of available analog channels of the device.*
- virtual uint32\_t [GetNumberOfSyncoutChannels](#) ()  
*Gets the Number of available syncout channels of the device.*
- virtual uint32\_t [GetNumberOfTriggerInputs](#) ()  
*Gets the Number of trigger inputs of the device.*
- virtual uint32\_t [GetNumberOfHWDACPaths](#) ()

- Gets the Number of HW Stimulation DACs of the device.*

  - virtual uint32\_t [GetNumberOfStimulationSourcesPerElectrode](#) ()
- Gets the number of stimulation sources (DACs) per electrode.*

  - virtual void [SetVoltageMode](#) (unsigned int channel)
- Sets a channel to voltage mode (STG3008-FA and STG400x only).*

  - virtual void [SetCurrentMode](#) (unsigned int channel)
- Sets a channel to current mode (STG3008-FA and STG400x only).*

  - virtual void [SetVoltageMode](#) ()
- Sets all channels to voltage mode (STG3008-FA and STG400x only).*

  - virtual void [SetCurrentMode](#) ()
- Sets all channels to current mode (STG3008-FA and STG400x only).*

  - virtual System::Collections::Generic::List< int32\_t > ^ [GetVoltageRangeListInMilliVolt](#) (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 [GetVoltageRangeInMilliVoltByIndex](#) (uint32\_t channel, uint32\_t index)
- Gets the voltage range for the given channel and index (STG5 only).*

  - virtual uint32\_t [GetVoltageResolutionInMicroVoltByIndex](#) (uint32\_t channel, uint32\_t index)
- Gets the voltage resolution 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 > ^ [GetCurrentRangeListInMicroAmp](#) (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 [GetCurrentRangeInMicroAmpByIndex](#) (uint32\_t channel, uint32\_t index)
- Gets the current range for the given channel and index (STG5 only).*

  - virtual uint32\_t [GetCurrentResolutionInNanoAmpByIndex](#) (uint32\_t channel, uint32\_t index)
- Gets the current resolution 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](#) (unsigned int amplification)
- Sets a channel to measurement mode (STG3008-FA).*

  - virtual uint32\_t [GetFAAmplification](#) ()
- Sets a channel to measurement mode (STG3008-FA).*

  - virtual void [SetAutocalibrationDisabled](#) (unsigned int channel, bool disable)
- Sets the autocalibration configuration.*

  - virtual bool [GetAutocalibrationDisabled](#) (unsigned int channel)
- Gets the autocalibration configuration.*

  - virtual void [SetElectrodeMode](#) (uint32\_t electrode, array< [ElectrodeModeEnumNet](#) > ^ mode)
- Puts an electrode in either automatic or manual mode.*

  - virtual void [SetElectrodeMode](#) (uint32\_t electrode, [ElectrodeModeEnumNet](#) mode)
- Puts an electrode in either automatic or manual mode.*

  - virtual void [SetElectrodeMode](#) (uint32\_t Scu\_HS, uint32\_t electrode, array< [ElectrodeModeEnumNet](#) > ^ mode)
- Puts an electrode in either automatic or manual mode.*

  - virtual void [SetElectrodeMode](#) (uint32\_t Scu\_HS, uint32\_t electrode, [ElectrodeModeEnumNet](#) mode)



- Puts an electrode in either automatic or manual mode.*

  - virtual uint32\_t [GetElectrodeMode](#) (uint32\_t electrode)

*Gets the mode an electrode is in.*

  - virtual uint32\_t [GetElectrodeMode](#) (uint32\_t Scu\_HS, uint32\_t electrode)

*Gets the mode an electrode is in.*

  - virtual void [SetElectrodeDacMux](#) (uint32\_t electrode, uint32\_t listmodelIndex, array< [ElectrodeDacMuxEnumNet](#) >^ dacMux)

*Defines the DAC to use for an electrode.*

  - virtual void [SetElectrodeDacMux](#) (uint32\_t electrode, uint32\_t listmodelIndex, [ElectrodeDacMuxEnumNet](#) dacMux)

*Defines the DAC to use for an electrode.*

  - virtual void [SetElectrodeDacMux](#) (uint32\_t Scu\_HS, uint32\_t electrode, uint32\_t listmodelIndex, [ElectrodeDacMuxEnumNet](#) dacMux)

*Defines the DAC to use for an electrode.*

  - virtual void [SetElectrodeDacMux](#) (uint32\_t Scu\_HS, uint32\_t electrode, uint32\_t listmodelIndex, array< [ElectrodeDacMuxEnumNet](#) >^ dacMux)

*Defines the DAC to use for an electrode.*

  - virtual [ElectrodeDacMuxEnumNet](#) [GetElectrodeDacMux](#) (uint32\_t electrode, uint32\_t listmodelIndex)

*Gets the DAC which is used for an electrode.*

  - virtual [ElectrodeDacMuxEnumNet](#) [GetElectrodeDacMux](#) (uint32\_t Scu\_HS, uint32\_t electrode, uint32\_t listmodelIndex)

*Gets the DAC which is used for an electrode.*

  - virtual void [SetElectrodeEnable](#) (uint32\_t electrode, uint32\_t listmodelIndex, array< bool >^ enable)

*Enables or disables the stimulation switch for an electrode.*

  - virtual void [SetElectrodeEnable](#) (uint32\_t electrode, uint32\_t listmodelIndex, bool enable)

*Enables or disables the stimulation switch for an electrode.*

  - virtual void [SetElectrodeEnable](#) (uint32\_t Scu\_HS, uint32\_t electrode, uint32\_t listmodelIndex, bool enable)

*Enables or disables the stimulation switch for an electrode.*

  - virtual void [SetElectrodeEnable](#) (uint32\_t Scu\_HS, uint32\_t electrode, uint32\_t listmodelIndex, array< bool >^ enable)

*Enables or disables the stimulation switch for an electrode.*

  - virtual bool [GetElectrodeEnable](#) (uint32\_t electrode, uint32\_t listmodelIndex)

*Gets weather an electrode is enabled or disabled for stimulation.*

  - virtual bool [GetElectrodeEnable](#) (uint32\_t Scu\_HS, uint32\_t electrode, uint32\_t listmodelIndex)

*Gets weather an electrode is enabled or disabled for stimulation.*

  - virtual void [SetExternalElectrodeEnable](#) (uint32\_t electrode, uint32\_t listmodelIndex, array< bool >^ enable)

*Enables or disables the stimulation switch for an external electrode.*

  - virtual void [SetExternalElectrodeEnable](#) (uint32\_t electrode, uint32\_t listmodelIndex, bool enable)

*Enables or disables the stimulation switch for an external electrode.*

  - virtual bool [GetExternalElectrodeEnable](#) (uint32\_t electrode, uint32\_t listmodelIndex)

*Gets weather 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.*

## Additional Inherited Members

### 11.110.1 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.

### 11.110.2 Constructor & Destructor Documentation

**11.110.2.1** [~CStg200xBasicNet\(\)](#) `virtual ~CStg200xBasicNet ( ) [virtual]`

The destructor.

### 11.110.3 Member Function Documentation

**11.110.3.1** [GetAnalogRanges\(\)](#) `void GetAnalogRanges (`  
`int channel,`  
`[Out] int% URange,`  
`[Out] int% IRange )`

Gets the range of the analog outputs.

**Parameters**

|                |                               |
|----------------|-------------------------------|
| <i>channel</i> | The channel which is queried. |
| <i>URange</i>  | The Voltage range in mV.      |
| <i>IRange</i>  | The Current range in uA.      |

**11.110.3.2 GetAnalogResolution()** `void GetAnalogResolution (`  
    `int channel,`  
    `[Out] int% URes,`  
    `[Out] int% IRes )`

Gets the resolution of the analog outputs.

**Parameters**

|                |                               |
|----------------|-------------------------------|
| <i>channel</i> | The channel which is queried. |
| <i>URes</i>    | The Voltage resolution in mV. |
| <i>IRes</i>    | The Current resolution in uA. |

**11.110.3.3 GetAutocalibrationDisabled()** `virtual bool GetAutocalibrationDisabled (`  
    `unsigned int channel ) [virtual]`

Gets the autocalibration configuration.

**Parameters**

|                |                     |
|----------------|---------------------|
| <i>channel</i> | The channel number. |
|----------------|---------------------|

**Returns**

`true` if autocalibration is disabled.

**11.110.3.4 GetAvailableMemory()** `uint32_t GetAvailableMemory ( )`

Gets the amount of memory available in the currently selected segment of the STG.

**Returns**

The memory available in the currently selected segment in bytes.

**11.110.3.5 GetBlankingEnable()** `[1/2] virtual bool GetBlankingEnable (`  
    `uint32_t electrode ) [virtual]`

Gets whether an electrode should be blanked while stimulation is in progress.

**Parameters**

|                  |                       |
|------------------|-----------------------|
| <i>electrode</i> | The electrode number. |
|------------------|-----------------------|

**Returns**

true if blanking is enabled while stimulation is in progress.

**11.110.3.6 GetBlankingEnable() [2/2]** `virtual bool GetBlankingEnable (`  
     `uint32_t Scu_HS,`  
     `uint32_t electrode ) [virtual]`

Gets whether an electrode should be blanked while stimulation is in progress.

**Parameters**

|               |                           |
|---------------|---------------------------|
| <i>Scu_HS</i> | The SCU headstage number. |
|---------------|---------------------------|

**Parameters**

|                  |                       |
|------------------|-----------------------|
| <i>electrode</i> | The electrode number. |
|------------------|-----------------------|

**Returns**

true if blanking is enabled while stimulation is in progress.

**11.110.3.7 GetCurrentRangeInMicroAmpByIndex()** `virtual uint32_t GetCurrentRangeInMicroAmpBy↔`  
     `Index (`  
         `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.

**11.110.3.8 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.

**11.110.3.9 GetCurrentRangeListInMicroAmp()** `virtual System::Collections::Generic::List<int32_t> ^ GetCurrentRangeListInMicroAmp ( uint32_t channel ) [virtual]`

Gets a list of current ranges supported by the device (STG5 only).

**11.110.3.10 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.

**11.110.3.11 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.

**11.110.3.12 GetCurrentResolutionInNanoAmpByIndex()** `virtual uint32_t GetCurrentResolutionInNanoAmpByIndex (`  
`uint32_t channel,`  
`uint32_t index ) [virtual]`

Gets the current resolution for the given channel and index (STG5 only).

**Parameters**

|                |              |
|----------------|--------------|
| <i>channel</i> | The channel. |
| <i>index</i>   | The channel. |

**Returns**

The current resolution in nA.

**11.110.3.13 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.

**11.110.3.14 GetDACResolution()** `virtual int32_t GetDACResolution ( ) [virtual]`

Gets number of bits of the DAC resolution.

**Returns**

The DAC resolution in bits.

**11.110.3.15 GetDiginValue()** `virtual uint32_t GetDiginValue ( ) [virtual]`

Gets the Value on the digital input port.

**Returns**

The current value on the digital inputs.

**11.110.3.16 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.

**11.110.3.17 GetDigoutValue()** `virtual uint32_t GetDigoutValue ( ) [virtual]`

Gets the Value on the digital output port.

**Returns**

The current value on the digital outputs.

**11.110.3.18 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.



**11.110.3.19 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.

**11.110.3.20 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.

**11.110.3.21 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.

**11.110.3.22 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.

**11.110.3.23 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.

**11.110.3.24 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.

**11.110.3.25 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.

**11.110.3.26 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.

**11.110.3.27 GetFAAmplification()** `virtual uint32_t GetFAAmplification ( ) [virtual]`

**11.110.3.28 GetHeadstage()** `virtual uint32_t GetHeadstage ( ) [virtual]`

**11.110.3.29 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.  |

## Parameters

|             |                                                             |
|-------------|-------------------------------------------------------------|
| <i>mode</i> | 0 for "start with startIndex", 1 for "start with endIndex". |
|-------------|-------------------------------------------------------------|

**11.110.3.30 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.

**11.110.3.31 GetNumberOfAnalogChannels()** `virtual uint32_t GetNumberOfAnalogChannels ( ) [virtual]`

Gets the Number of available analog channels of the device.

## Returns

The number of analog channels.

**11.110.3.32 GetNumberOfCurrentRangeIndexes()** `virtual uint32_t GetNumberOfCurrentRangeIndexes ( ) [virtual]`

Gets the number of current ranges (STG5 only).

## Returns

The number of current ranges available on the device.

**11.110.3.33 GetNumberOfHWDACPaths()** `virtual uint32_t GetNumberOfHWDACPaths ( ) [virtual]`

Gets the Number of HW Stimulation DACs of the device.

**Returns**

The number of independent HW Stimulation outputs.

**11.110.3.34 GetNumberOfStimulationElectrodes()** `virtual uint32_t GetNumberOfStimulationElectrodes ( ) [virtual]`

**11.110.3.35 GetNumberOfStimulationSourcesPerElectrode()** `virtual uint32_t GetNumberOfStimulationSourcesPerElectrode ( ) [virtual]`

Gets the number of stimulation sources (DACs) per electrode.

**Returns**

The number of stimulation sources (DACs) per electrode.

**11.110.3.36 GetNumberOfSyncoutChannels()** `virtual uint32_t GetNumberOfSyncoutChannels ( ) [virtual]`

Gets the Number of available syncout channels of the device.

**Returns**

The number of analog channels.

**11.110.3.37 GetNumberOfTriggerInputs()** `virtual uint32_t GetNumberOfTriggerInputs ( ) [virtual]`

Gets the Number of trigger inputs of the device.

**Returns**

The number of trigger inputs.

**11.110.3.38 GetNumberOfVoltageRangeIndexes()** `virtual uint32_t GetNumberOfVoltageRangeIndexes ( ) [virtual]`

Gets the number of voltage ranges (STG5 only).

#### Returns

The number of voltage ranges available on the device.

**11.110.3.39 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.

**11.110.3.40 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.                                    |

**11.110.3.41 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.   |

**11.110.3.42 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.        |

**11.110.3.43 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.

**11.110.3.44 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.

**11.110.3.45 GetTriggerSource()** `virtual TriggerSourceEnumNet GetTriggerSource (`  
    `unsigned int triggernum ) [virtual]`

**11.110.3.46 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.

**11.110.3.47 GetVoltageRangeInMilliVoltByIndex()** virtual uint32\_t GetVoltageRangeInMilliVoltByIndex (   
     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.

**11.110.3.48 GetVoltageRangeListInMilliVolt()** virtual System::Collections::Generic::List<int32\_t> ^ GetVoltageRangeListInMilliVolt (   
     uint32\_t *channel* ) [virtual]

Gets a list of current ranges supported by the device (STG5 only).

**11.110.3.49 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.

**11.110.3.50 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.

**11.110.3.51 GetVoltageResolutionInMicroVoltByIndex()** `virtual uint32_t GetVoltageResolutionInMicroVoltByIndex ( uint32_t channel, uint32_t index ) [virtual]`

Gets the voltage resolution for the given channel and index (STG5 only).

**Parameters**

|                |              |
|----------------|--------------|
| <i>channel</i> | The channel. |
| <i>index</i>   | The channel. |

**Returns**

The voltage resolution.

**11.110.3.52 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. |
|---------------------------|----------------------------------------------------------------------|

**11.110.3.53 ListModeSendStop()** `virtual void ListModeSendStop ( unsigned int electrodeGroupMask ) [virtual]`

Deactivate the Listmode for the selected electrode groups.

## Parameters

|                           |                                                                       |
|---------------------------|-----------------------------------------------------------------------|
| <i>electrodeGroupMask</i> | The bitmask of electrodegroups for which the listmode is deactivated. |
|---------------------------|-----------------------------------------------------------------------|

**11.110.3.54 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. |
|-------------------|---------------------------------------------|

**11.110.3.55 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. |
|-------------------|---------------------------------------------|

**11.110.3.56 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.. |

**11.110.3.57 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. |

**11.110.3.58 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. |

**11.110.3.59 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. |

**11.110.3.60 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. |

**11.110.3.61 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. |

**11.110.3.62 SetCurrentMode()** [1/2] `virtual void SetCurrentMode ( ) [virtual]`

Sets all channels to current mode (STG3008-FA and STG400x only).

**11.110.3.63 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. |
|----------------|------------------------|

**11.110.3.64 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.   |

**11.110.3.65 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. |

**11.110.3.66 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. |
|-------------------|-------------------------|

**11.110.3.67 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. |
|--------------------|---------------------------------------|

**11.110.3.68 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). |

**11.110.3.69 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). |

**11.110.3.70 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). |

**11.110.3.71 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). |

**11.110.3.72 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. |

**11.110.3.73 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. |

**11.110.3.74 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. |

**11.110.3.75 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. |

**11.110.3.76 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.

**11.110.3.77 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. |
|-------------|----------------------------------------|

**11.110.3.78 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.

**11.110.3.79 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. |
|-------------|----------------------------------------|

**11.110.3.80 SetEnableAmplifierProtectionSwitch()** [1/4] `virtual void SetEnableAmplifierProtectionSwitch (`  
`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. |

**11.110.3.81 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. |

**11.110.3.82 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. |

**11.110.3.83 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. |

**11.110.3.84 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. |

**11.110.3.85 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. |

**11.110.3.86 SetFAAmplification()** virtual void SetFAAmplification (   
 unsigned int *amplification* ) [virtual]

**11.110.3.87 SetHeadstage()** virtual void SetHeadstage (   
 unsigned int *headstage* ) [virtual]

**11.110.3.88 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". |

**11.110.3.89 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. |
|----------------------|-------------------------------------------------|

**11.110.3.90 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 triggersource 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. |

**11.110.3.91 SetMeasurementMode()** `virtual void SetMeasurementMode ( unsigned int channel ) [virtual]`

Sets a channel to measurement mode (STG3008-FA).

## Parameters

|                |                        |
|----------------|------------------------|
| <i>channel</i> | The channel to change. |
|----------------|------------------------|

**11.110.3.92 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. |
|-------------|----------------------------|

**11.110.3.93 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. |

**11.110.3.94 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. |

**11.110.3.95 SetTriggerSource()** `[1/3] virtual void SetTriggerSource (`  
     `unsigned int triggernum,`

```
DigitalSource< digitalsourceenum >^ triggersource,
int bitnum_offset) [virtual]
```

**11.110.3.96 SetTriggerSource() [2/3]** virtual void SetTriggerSource (   
unsigned int *triggernum*,  
*TriggerSourceEnumNet* *triggersource* ) [virtual]

**11.110.3.97 SetTriggerSource() [3/3]** virtual void SetTriggerSource (   
unsigned int *triggernum*,  
*TriggerSourceEnumNet* *triggersource*,  
int *bitnum\_offset* ) [virtual]

**11.110.3.98 SetVoltageMode() [1/2]** virtual void SetVoltageMode ( ) [virtual]

Sets all channels to voltage mode (STG3008-FA and STG400x only).

**11.110.3.99 SetVoltageMode() [2/2]** virtual void SetVoltageMode (   
unsigned int *channel* ) [virtual]

Sets a channel to voltage mode (STG3008-FA and STG400x only).

**Parameters**

|                |                        |
|----------------|------------------------|
| <i>channel</i> | The channel to change. |
|----------------|------------------------|

**11.110.3.100 SetVoltageRangeSelectedIndex()** virtual void SetVoltageRangeSelectedIndex (   
uint32\_t *channel*,  
uint32\_t *rangeIndex* ) [virtual]

Sets the range index for the voltage output (not used yet).

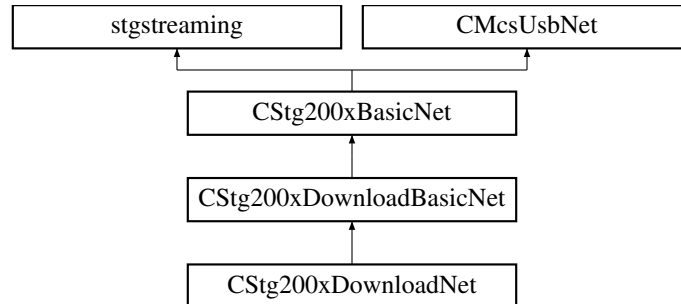
**Parameters**

|                   |                        |
|-------------------|------------------------|
| <i>channel</i>    | The channel to change. |
| <i>rangeIndex</i> | The new range index.   |

## 11.111 CStg200xDownloadBasicNet Class Reference

[CStg200xDownloadBasicNet](#) is the base class to control the download mode of the MCS STG device.

Inheritance diagram for CStg200xDownloadBasicNet:



### Public Member Functions

- virtual void [SetupTrigger](#) (uint32\_t first\_trigger, array< uint32\_t >^ channelmap, array< uint32\_t >^ syncoutmap, array< uint32\_t >^ repeat)  
*Configures the trigger settings for the STG. Note that all memory segments have their own trigger setting.*
- virtual void [SetupTriggerSingle](#) (uint32\_t trigger, uint32\_t channelmap, uint32\_t syncoutmap, uint32\_t repeat)  
*Configures the trigger settings for the STG. Note that all memory segments have their own trigger setting.*
- void [GetTrigger](#) ([Out] array< uint32\_t >^% channelmap, [Out] array< uint32\_t >^% syncoutmap, [Out] array< uint32\_t >^% repeat)  
*Queries the trigger settings for the STG. Note that all memory segments have their own trigger setting.*
- void [GetSweepCount](#) ([Out] array< uint32\_t >^% sweeps, [Out] array< uint32\_t >^% triggers)  
*Get the sweep and trigger count of the STG.*
  - The triggercount tells how many times each trigger was active and is reset to zero on download of new channel data.
  - The sweepcount tells how many times each trigger was already repeated. This count is set to zero on trigger and counts up to repeat in [CStg200xDownloadBasicNet::SetupTrigger](#).
- void [ForceStatusEvent](#) ()  
*Force a status event.*
- void [ResetStatus](#) (uint32\_t triggermap)  
*Reset the status flag.*
- uint32\_t [GetMemoryUsageDAC](#) (uint32\_t Channel)  
*Queries the memory usage of the current segment and selected analog DAC channel.*
- uint32\_t [GetMemoryUsageSyncout](#) (uint32\_t Channel)  
*Queries the memory usage of the current segment and selected syncout channel.*
- virtual void [ClearSyncData](#) (uint32\_t channel)  
*Delete a SyncOut pattern for a channel from STG memory.*
- virtual void [SendSyncData](#) (uint32\_t channel, array< uint16\_t >^ pData, array< uint64\_t >^ tData)  
*Uploads sync output data to the STG.*  
*Sends sync output data to a given channel on the STG. The list of datapoints will be sent to the selected sync output channel. Sync output data previously sent to the channel is overwritten.*  
*Each datapoint is represented by an integer value and can be either 0 or 1.*  
*The duration is given as a list of 64 bit integers. Durations are given in units of  $\mu$ s. The STG has a resolution of 20  $\mu$ s. If your application can not handle 64 bit integers, use the [STG200x\\_SendSyncData32\(\)](#) call instead.*
- virtual void [ClearChannelData](#) (uint32\_t channel)  
*Delete a stimulus pattern for a channel from STG memory*
- virtual void [SendChannelData](#) (uint32\_t channel, array< uint16\_t >^ pData, array< uint64\_t >^ tData)

*Uploads analog data (stimulus patterns) to the STG.*

*Sends datapoints to a given channel on the STG. The list of datapoints will be sent to the selected channel. Data previously sent to the channel is overwritten.*

*Each datapoint is represented by an integer value in the range from 0 to 4095 (bit 0 to 11), its sign is taken from bit 12, 0 is for positive amplitude, and 1 for negative amplitude Bits 13 to 15 have to be zero.*

*The duration is given as a list of 64 bit integers. Durations are given in units of  $\mu\text{s}$ . The STG has a resolution of 20  $\mu\text{s}$ .*

- 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

### 11.111.1 Detailed Description

[CStg200xDownloadBasicNet](#) is the base class to control the download mode of the MCS STG device.

### 11.111.2 Member Function Documentation

**11.111.2.1 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. |
|----------------|---------------------------------|

**11.111.2.2 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. |
|----------------|-----------------------------------------|

**11.111.2.3 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()

**11.111.2.4 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.

**11.111.2.5 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.

**11.111.2.6 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.

**11.111.2.7 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.

**11.111.2.8 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. |

**11.111.2.9 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. |

**11.111.2.10 ResetStatus()** `void ResetStatus (`  
`uint32_t triggermap )`

Reset the status flag.

## Parameters

|                   |                                                  |
|-------------------|--------------------------------------------------|
| <i>triggermap</i> | bitmap of trigger for which to reset the status. |
|-------------------|--------------------------------------------------|

**11.111.2.11 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. |

**11.111.2.12 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  $\mu\text{s}$ . The STG has a resolution of 20  $\mu\text{s}$ . 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 $\mu\text{s}$ . |

**11.111.2.13 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. |

**11.111.2.14 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. |

**11.111.2.15 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. |

**11.111.2.16 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. |

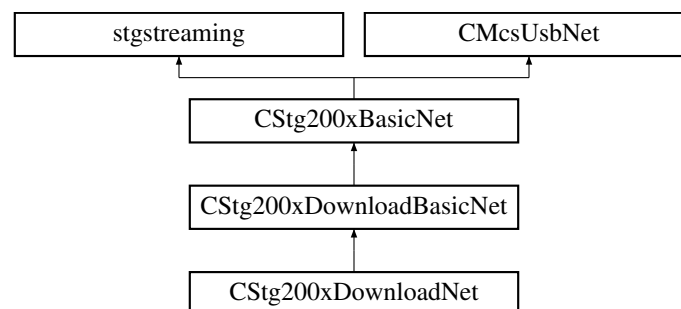
**11.111.3 Property Documentation**

**11.111.3.1 Stimulus** `CStimulusFunctionNet^` Stimulus [get]

**11.112 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 [SendSegmentDefine](#) (array< uint32\_t >^ segment\_list)  
*Defines the segment memory layout of the STG.*
- void [SendSegmentStart](#) (uint32\_t triggermap, uint32\_t segment, [Stg200xSegmentFlagsEnumNet](#) segment-flags)  
*Switchs segment and starts trigger.*
- void [SendSegmentSelect](#) (uint32\_t segment, [Stg200xSegmentFlagsEnumNet](#) segmentflags)  
*Switchs segment.*
- void [EnableMultiFileMode](#) (uint32\_t submode)  
*Enable the Multi-File mode of the STG.*
- void [DisableMultiFileMode](#) ()  
*Disable the Multi-File mode of the STG*
- [StgStatusNet](#) ^ [QueryTriggerstatus](#) ()
- void [SetOutputMap](#) (array< uint32\_t >^ ChannelLayout)
- int32\_t [GetModuleTemp](#) (unsigned int channel)
- uint32\_t [GetModuleCurrent](#) (unsigned int channel)

## Events

- [OnStgPollStatus](#)^ [Stg200xPollStatusEvent](#) [add, remove, raise]
- [OnMwPollStatus](#)^ [MwPollStatusEvent](#) [add, remove, raise]

## Additional Inherited Members

### 11.112.1 Detailed Description

Main class for the STG download mode This class implements the STG download mode interface.

### 11.112.2 Constructor & Destructor Documentation

#### 11.112.2.1 CStg200xDownloadNet() [1/2] [CStg200xDownloadNet](#) ( )

Use this constructor if you do not want to use the status callback.

**11.112.2.2 CStg200xDownloadNet()** [2/2] `CStg200xDownloadNet ( OnStgPollStatus^ pollStatus )`

Use this constructor if you want to use the status callback.

**11.112.2.3 ~CStg200xDownloadNet()** `~CStg200xDownloadNet ( )`

### 11.112.3 Member Function Documentation

**11.112.3.1 ClearChannel\_PrepareAndSendData()** `void ClearChannel_PrepareAndSendData ( uint32_t channel, array< int32_t >^ amplitude, array< uint64_t >^ duration, STG_DestinationEnumNet destType, bool doClear )`

Prepare and append data to a given channel on the STG.

Each datapoint is represented by an signed 32bit integer value. When using voltage stimulation, the values are in multiple of 1  $\mu$ V, thus the possible range is  $\pm 2000$  V. When using current stimulation, the values are in multiple of 1 nA, this the possible range is  $\pm 2000$  mA.

The duration is given as a list of 64 bit integers. Durations are given in units of  $\mu$ s. The STG has a resolution of 20  $\mu$ s.

Blocks of data which should repeat can be defined by prepending such a block with an entry in the arrays where both amplitude and duration is zero. The end of such an block is marked by an entry where the duration is set to zero and the amplitude beeing set to the number of times the block should run. Blocks can be nested.

#### Parameters

|                |                                     |
|----------------|-------------------------------------|
| <i>channel</i> | The channel number to send data to. |
|----------------|-------------------------------------|

#### Parameters

|                  |                                                                                            |
|------------------|--------------------------------------------------------------------------------------------|
| <i>amplitude</i> | A list of amplitudes in units of $\mu$ V and nA in voltage and current mode, respectively. |
|------------------|--------------------------------------------------------------------------------------------|

#### Parameters

|                 |                                                                             |
|-----------------|-----------------------------------------------------------------------------|
| <i>duration</i> | A list of durations in units of $\mu$ s.                                    |
| <i>destType</i> | specifies wheather the data is for syncout, current or voltage stimulation. |

**11.112.3.2 DisableMultiFileMode()** `void DisableMultiFileMode ( )`

Disable the Multi-File mode of the STG

Switch the STG back to normal mode. In this mode, trigger inputs are assigned to channels, not to segments.

**11.112.3.3 EnableMultiFileMode()** `void EnableMultiFileMode (   
uint32_t submode )`

Enable the Multi-File mode of the STG.

In Multi-File mode, the trigger inputs switch between segments. To use this mode, define up to as many segments as trigger inputs are available and fill each segment with a stimulus pattern.

Now a trigger on trigger input 1 switches the STG to the first segment and starts all triggers in this segment. Likewise, a trigger on trigger input 2, 3 and 4 selects the respective segment and start all triggers in this segment. So the Multi-File Mode can be used to predefine up to four different stimuli which can be selected without the need for a computer connection.

**Parameters**

|                |                                                                                                                                                                                                                                                              |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>submode</i> | The submode. Submode 0 is regular Multi-File mode as described above, submode 1 is extended Multi-File mode, where the segment is selected based on the digital pattern on the digital inputs. In this mode, 256 different segments can be defined and used. |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**11.112.3.4 GetModuleCurrent()** `uint32_t GetModuleCurrent (   
unsigned int channel )`**11.112.3.5 GetModuleTemp()** `int32_t GetModuleTemp (   
unsigned int channel )`**11.112.3.6 PrepareAndAppendData()** `void PrepareAndAppendData (   
uint32_t channel,   
array< int32_t >^ amplitude,   
array< uint64_t >^ duration,   
STG_DestinationEnumNet destType )`

Prepare and append data to a given channel on the STG.

Each datapoint is represented by an signed 32bit integer value. When using voltage stimulation, the values are in multiple of 1  $\mu$ V, thus the possible range is  $\pm 2000$  V. When using current stimulation, the values are in multiple of 1 nA, this the possible range is  $\pm 2000$  mA.

The duration is given as a list of 64 bit integers. Durations are given in units of  $\mu$ s. The STG has a resolution of 20  $\mu$ s.

Blocks of data which should repeat can be defined by prepending such a block with an entry in the arrays where both amplitude and duration is zero. The end of such an block is marked by an entry where the duration is set to zero and the amplitude beeing set to the number of times the block should run. Blocks can be nested.

#### Parameters

|                |                                     |
|----------------|-------------------------------------|
| <i>channel</i> | The channel number to send data to. |
|----------------|-------------------------------------|

#### Parameters

|                  |                                                                                            |
|------------------|--------------------------------------------------------------------------------------------|
| <i>amplitude</i> | A list of amplitudes in units of $\mu$ V and nA in voltage and current mode, respectively. |
|------------------|--------------------------------------------------------------------------------------------|

#### Parameters

|                 |                                                                             |
|-----------------|-----------------------------------------------------------------------------|
| <i>duration</i> | A list of durations in units of $\mu$ s.                                    |
| <i>destType</i> | specifies wheather the data is for syncout, current or voltage stimulation. |

**11.112.3.7 PrepareAndSendData()** `void PrepareAndSendData (`  
     `uint32_t channel,`  
     `array< int32_t >^ amplitude,`  
     `array< uint64_t >^ duration,`  
     `STG_DestinationEnumNet destType )`

Prepare and send data to a given channel on the STG. Previous data sent to that channel is erased first.

Each datapoint is represented by an signed 32bit integer value. When using voltage stimulation, the values are in multiple of 1  $\mu$ 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. |

### 11.112.3.8 QueryTriggerstatus() `StgStatusNet ^ QueryTriggerstatus ( )`

### 11.112.3.9 SendSegmentDefine() `void SendSegmentDefine ( array< uint32_t >^ segment_list )`

Defines the segment memory layout of the STG.

On reset, the STG has one segment containing all available memory.

With this command, the STG memory can be devided into several segments. Each segment can be filled with stimulus data.

## Parameters

|                     |                                             |
|---------------------|---------------------------------------------|
| <i>segment_list</i> | The List of memory sizes (one per segment). |
|---------------------|---------------------------------------------|

### 11.112.3.10 SendSegmentSelect() `void SendSegmentSelect ( uint32_t segment, Stg200xSegmentFlagsEnumNet segmentflags )`

Switchs segment.

**Parameters**

|                |                                      |
|----------------|--------------------------------------|
| <i>segment</i> | The number of the segment to select. |
|----------------|--------------------------------------|

**Parameters**

|                     |                                                                                           |
|---------------------|-------------------------------------------------------------------------------------------|
| <i>segmentflags</i> | A bitmap of flags, bit 1: assign all channels to the trigger number equal to the segment. |
|---------------------|-------------------------------------------------------------------------------------------|

**11.112.3.11 SendSegmentStart()** `void SendSegmentStart (`  
    `uint32_t triggermap,`  
    `uint32_t segment,`  
    `Stg200xSegmentFlagsEnumNet segmentflags )`

Switchs segment and starts trigger.

**Parameters**

|                   |                                            |
|-------------------|--------------------------------------------|
| <i>triggermap</i> | A bitmap of triggers that will be started. |
|-------------------|--------------------------------------------|

**Parameters**

|                |                                      |
|----------------|--------------------------------------|
| <i>segment</i> | The number of the segment to select. |
|----------------|--------------------------------------|

**Parameters**

|                     |                                                                                           |
|---------------------|-------------------------------------------------------------------------------------------|
| <i>segmentflags</i> | A bitmap of flags, bit 1: assign all channels to the trigger number equal to the segment. |
|---------------------|-------------------------------------------------------------------------------------------|

**11.112.3.12 SetOutputMap()** `void SetOutputMap (`  
    `array< uint32_t >^ ChannelLayout )`

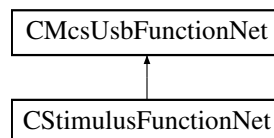
**11.112.4 Event Documentation**

**11.112.4.1 MwPollStatusEvent** [OnMwPollStatus](#)<sup>^</sup> MwPollStatusEvent [add], [remove], [raise]

**11.112.4.2 Stg200xPollStatusEvent** [OnStgPollStatus](#)<sup>^</sup> Stg200xPollStatusEvent [add], [remove], [raise]

## 11.113 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 >^ amplitude, array< uint64\_t >^ duration, [STG\\_DestinationEnumNet](#) destType, bool doClear)
- [StimulusDeviceDataAndUnrolledData](#) ^ [PrepareData](#) (int channel, array< int32\_t >^ amplitude, array< uint64\_t >^ 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.*
- 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

### 11.113.1 Constructor & Destructor Documentation

**11.113.1.1 CStimulusFunctionNet()** [1/2] `CStimulusFunctionNet (`  
`CMcsUsbNet^ mcsusb,`  
`CMcsUsbFunctionPointerContainer^ stimulusFunctionPointerContainer )`

**11.113.1.2 CStimulusFunctionNet()** [2/2] `CStimulusFunctionNet (`  
`CMcsUsbNet^ mcsusb )`

## 11.113.2 Member Function Documentation

**11.113.2.1 ClearChannel\_PrepareAndSendData()** `void ClearChannel_PrepareAndSendData (`  
`uint32_t channel,`  
`array< int32_t >^ amplitude,`  
`array< uint64_t >^ duration,`  
`STG_DestinationEnumNet destType,`  
`bool doClear )`

**11.113.2.2 ClearChannelData()** `void ClearChannelData (`  
`int channel )`

Delete a Stimulus Pattern from STG memory

### Parameters

|                |                                 |
|----------------|---------------------------------|
| <i>channel</i> | specifies the channel to clear. |
|----------------|---------------------------------|

**11.113.2.3 ClearMultiplexedData()** `void ClearMultiplexedData ( )`

Clears the Stimulation Memory in the STG device.

**11.113.2.4 ClearSyncData()** `void ClearSyncData (`  
`int channel )`

Delete a Syncout Pattern from STG memory

### Parameters

|                |                                 |
|----------------|---------------------------------|
| <i>channel</i> | specifies the channel to clear. |
|----------------|---------------------------------|

**11.113.2.5 CreateSideband()** `SidebandData ^ CreateSideband (`  
`array< int32_t >^ StimulusActive,`  
`array< int32_t >^ Syncout,`  
`array< uint64_t >^ Duration,`  
`uint32_t Bit0Time,`  
`uint32_t Bit3Time,`  
`uint32_t Bit4Time )`

Creates the Sideband Channel for the MEA2100 device.

Each datapoint is represented by an signed 32bit integer value. A value 0 means that the stimulation is active during that time. A value 1 means that the stimulation is not active during that time.

The duration is given as a list of 64 bit integers. Durations are given in units of  $\mu\text{s}$ . The STG has a resolution of 20  $\mu\text{s}$ .

#### Parameters

|                       |                                                                                                 |
|-----------------------|-------------------------------------------------------------------------------------------------|
| <i>StimulusActive</i> | A list of datapoints which define weather the Stimulus is active or idle at that time as int32. |
|-----------------------|-------------------------------------------------------------------------------------------------|

#### Parameters

|                 |                                                                              |
|-----------------|------------------------------------------------------------------------------|
| <i>Duration</i> | A list of durations as uint64. The time is given in units of $\mu\text{s}$ . |
| <i>Bit0Time</i> | Time in $\mu\text{s}$ for which Bit 0 (Blanking) is to be extended.          |

#### Parameters

|                 |                                                                            |
|-----------------|----------------------------------------------------------------------------|
| <i>Bit3Time</i> | Time in $\mu\text{s}$ for which Bit 3 (Stimulus Enable) is to be extended. |
|-----------------|----------------------------------------------------------------------------|

#### Parameters

|                 |                                                                              |
|-----------------|------------------------------------------------------------------------------|
| <i>Bit4Time</i> | Time in $\mu\text{s}$ for which Bit 4 (Stimulus Selector) is to be extended. |
|-----------------|------------------------------------------------------------------------------|

#### Returns

Error Status. 0 on success.

**11.113.2.6 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.

**11.113.2.7 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.

**11.113.2.8 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.

**11.113.2.9 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.

**11.113.2.10 GetDACResolution()** `int GetDACResolution ( )`

Gets number of bits of the DAC resolution.

**Returns**

The DAC resolution in bits.

**11.113.2.11 GetMultiplexedDataChannelsInBlock()** `int GetMultiplexedDataChannelsInBlock ( )`

Gets the number of stimulus data channels to send per time slice. Might be greater than the number of configured channels. Fill unused channels with dummy data in `SendMultiplexedData`

**11.113.2.12 GetNumberOfAnalogChannels()** `int GetNumberOfAnalogChannels ( )`

Get the number of STG channels.

**Returns**

The number of STG channels.

**11.113.2.13 GetTotalMemory()** `uint32_t GetTotalMemory ( )`

Get the total amount of memory available on the STG (all segments).

**Returns**

The total memory available on the STG in bytes.

**11.113.2.14 GetVoltageRangeInMicroVolt()** `int GetVoltageRangeInMicroVolt (   
uint32_t channel )`

Gets the Voltage Range of the specified channel in Microvolts.

**Parameters**

|                |                           |
|----------------|---------------------------|
| <i>channel</i> | Channel which is queried. |
|----------------|---------------------------|



**Returns**

The Voltage Range of the specified channel in Microvolts.

**11.113.2.15 GetVoltageResolutionInMicroVolt()** `int GetVoltageResolutionInMicroVolt ( uint32_t channel )`

Gets the Voltage Resolution of the specified channel in Microvolts.

**Parameters**

|                |                           |
|----------------|---------------------------|
| <i>channel</i> | Channel which is queried. |
|----------------|---------------------------|

**Returns**

The Voltage Resolution of the specified channel in Microvolts.

**11.113.2.16 PrepareAndAppendData()** `void PrepareAndAppendData ( uint32_t channel, array< int32_t >^ amplitude, array< uint64_t >^ duration, STG_DestinationEnumNet destType )`

Prepare and append data to a given channel on the STG.

Each datapoint is represented by an signed 32bit integer value. When using voltage stimulation, the values are in multiple of 1  $\mu$ V, thus the possible range is  $\pm 2000$  V. When using current stimulation, the values are in multiple of 1 nA, this the possible range is  $\pm 2000$  mA.

The duration is given as a list of 64 bit integers. Durations are given in units of  $\mu$ s. The STG has a resolution of 20  $\mu$ s.

Blocks of data which should repeat can be defined by prepending such a block with an entry in the arrays where both amplitude and duration is zero. The end of such an block is marked by an entry where the duration is set to zero and the amplitude beeing set to the number of times the block should run. Blocks can be nested.

**Parameters**

|                |                                     |
|----------------|-------------------------------------|
| <i>channel</i> | The channel number to send data to. |
|----------------|-------------------------------------|

**Parameters**

|                  |                                                                                            |
|------------------|--------------------------------------------------------------------------------------------|
| <i>amplitude</i> | A list of amplitudes in units of $\mu$ V and nA in voltage and current mode, respectively. |
|------------------|--------------------------------------------------------------------------------------------|

**Parameters**

|                 |                                                                             |
|-----------------|-----------------------------------------------------------------------------|
| <i>duration</i> | A list of durations in units of $\mu\text{s}$ .                             |
| <i>destType</i> | specifies wheather the data is for syncout, current or voltage stimulation. |

**Returns**

Error Status. 0 on success.

**11.113.2.17 PrepareAndSendData()** `void PrepareAndSendData (`  
    `uint32_t channel,`  
    `array< int32_t >^ amplitude,`  
    `array< uint64_t >^ duration,`  
    `STG_DestinationEnumNet destType )`

Prepare and send data to a given channel on the STG. Previous data sent to that channel is erased first.

Each datapoint is represented by an signed 32bit integer value. When using voltage stimulation, the values are in multiple of 1  $\mu\text{V}$ , 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.

**11.113.2.18 PrepareData()** `StimulusDeviceDataAndUnrolledData ^ PrepareData (`  
`int channel,`  
`array< int32_t >^ amplitude,`  
`array< uint64_t >^ duration,`  
`STG_DestinationEnumNet destType )`

**11.113.2.19 SendMultiplexedData()** `void SendMultiplexedData (`  
`array< uint16_t >^ data )`

Sends stimulus data in multiplexed form. All 16 bits words for the enabled DAC and digital channels are muxed together per time slice.

## Parameters

|             |                           |
|-------------|---------------------------|
| <i>data</i> | Array of data to be sent. |
|-------------|---------------------------|

**11.113.2.20 SendPreparedData()** `void SendPreparedData (`  
`int channel,`  
`StimulusDeviceDataAndUnrolledData^ device_data_and_unrolled,`  
`STG_DestinationEnumNet destType )`

**11.113.2.21 SendStart()** `void SendStart (`  
`uint32_t triggermap )`

Start (Trigger) the STG. The startup delay is in the range of a few ms.

## Parameters

|                   |                                             |
|-------------------|---------------------------------------------|
| <i>triggermap</i> | A bitmap of triggers which will be started. |
|-------------------|---------------------------------------------|

**11.113.2.22 SendStop() [1/2]** `void SendStop (`  
`uint32_t triggermap )`

Stop some or all triggers of the STG.

## Parameters

|                   |                                             |
|-------------------|---------------------------------------------|
| <i>triggermap</i> | A bitmap of triggers which will be stopped. |
|-------------------|---------------------------------------------|

**11.113.2.23 SendStop() [2/2]** `void SendStop (`  
`uint32_t triggermap,`  
`int options )`

Stop some or all triggers of the STG.

## Parameters

|                   |                                                                                                                                                                                                                                                                    |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>triggermap</i> | A bitmap of triggers which will be stopped.                                                                                                                                                                                                                        |
| <i>options</i>    | bitmap of options, currently only STOP_OPTION_SAVESTOP (0x80) is defined, which bypasses the stop commands when a syncout associated with a given sync-out has bit 1 (0x02) set. Can be used e.g. to prevent a stop while a biphasic stimulation pulse is active.. |

**11.113.2.24 SetupTrigger()** `void SetupTrigger (`  
`uint32_t first_trigger,`  
`array< uint32_t >^ channelmap,`  
`array< uint32_t >^ syncoutmap,`  
`array< uint32_t >^ repeat )`

Configures the trigger settings for the STG. Note that all memory segments have their own trigger setting.

## Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <i>first_trigger</i> | The number of the first trigger to change. |
|----------------------|--------------------------------------------|

## Parameters

|                   |                                                                     |
|-------------------|---------------------------------------------------------------------|
| <i>channelmap</i> | For each trigger, a bitmap of channels that belong to this trigger. |
|-------------------|---------------------------------------------------------------------|

## Parameters

|                   |                                                                               |
|-------------------|-------------------------------------------------------------------------------|
| <i>syncoutmap</i> | For each trigger, a bitmap of syncouts that belong to this trigger.           |
| <i>repeat</i>     | For each trigger, define the number of times this trigger should be repeated. |

**11.113.2.25 SetupTriggerSingle()** `void SetupTriggerSingle (`  
     `uint32_t trigger,`  
     `uint32_t channelmap,`  
     `uint32_t syncoutmap,`  
     `uint32_t repeat )`

Configures the trigger settings for the STG. Note that all memory segments have their own trigger setting.

## Parameters

|                |                        |
|----------------|------------------------|
| <i>trigger</i> | The trigger to change. |
|----------------|------------------------|

## Parameters

|                   |                                                   |
|-------------------|---------------------------------------------------|
| <i>channelmap</i> | A bitmap of channels that belong to this trigger. |
|-------------------|---------------------------------------------------|

## Parameters

|                   |                                                      |
|-------------------|------------------------------------------------------|
| <i>syncoutmap</i> | A bitmap of syncouts that belong to this trigger.    |
| <i>repeat</i>     | The number of times this trigger should be repeated. |

**11.113.2.26 StartPoll()** `void StartPoll ( )`

Starts the interrupt fetching thread and delivers events

### 11.113.2.27 StopPoll() `void StopPoll ( )`

Stops the interrupt fetching thread and delivers events

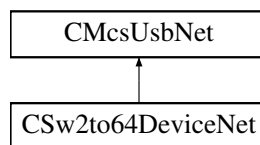
## 11.113.3 Event Documentation

### 11.113.3.1 PollStatusEvent `OnStgPollStatus^ PollStatusEvent`

## 11.114 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

#### 11.114.1 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.

### 11.114.2 Constructor & Destructor Documentation

**11.114.2.1 CSw2to64DeviceNet()** `CSw2to64DeviceNet ( )`

**11.114.2.2 ~CSw2to64DeviceNet()** `~CSw2to64DeviceNet ( )`

### 11.114.3 Member Function Documentation

**11.114.3.1 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

**11.114.3.2 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

**11.114.3.3 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

**11.114.3.4 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                    |

**11.114.3.5 SetChannels()** `void SetChannels (`  
`array< unsigned char >^ pattern )`

Sets the switch positions from a char array.

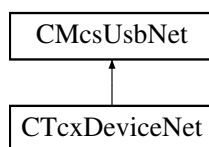
## Parameters

|    |                |                                                                                               |
|----|----------------|-----------------------------------------------------------------------------------------------|
| in | <i>pattern</i> | array of char with the size of the number of channels, each char has the setting of a channel |
|----|----------------|-----------------------------------------------------------------------------------------------|

## 11.115 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

### 11.115.1 Detailed Description

Class to control a Temperature Controller (TCX)

### 11.115.2 Constructor & Destructor Documentation

**11.115.2.1 CTcxDeviceNet()** `CTcxDeviceNet ( )`

Initializes a new instance of `CTcxDeviceNet` class.

**11.115.2.2 ~CTcxDeviceNet()** `~CTcxDeviceNet ( )`**11.115.3 Member Function Documentation****11.115.3.1 CalibrateThermocouple()** `void CalibrateThermocouple ( unsigned int channel )`**11.115.3.2 FactoryReset()** `void FactoryReset ( )`**11.115.3.3 GetBoardTemp()** `unsigned int GetBoardTemp ( )`

Gets the temperate of the mainboard in units of 0.1 °C.

**11.115.3.4 GetCalibration()** `int GetCalibration ( unsigned int channel )`**11.115.3.5 GetCalibrationDecp()** `int GetCalibrationDecp ( unsigned int channel )`**11.115.3.6 GetCalibrationMax()** `int GetCalibrationMax ( unsigned int channel )`**11.115.3.7 GetCalibrationMin()** `int GetCalibrationMin ( unsigned int channel )`

**11.115.3.8 GetCurrent()** `int GetCurrent (`  
`unsigned int channel )`

**11.115.3.9 GetD()** `int GetD (`  
`unsigned int device )`

Gets the D-coefficient of the specified device.

**11.115.3.10 GetDDecp()** `int GetDDecp (`  
`unsigned int device )`

**11.115.3.11 GetDevice()** `int GetDevice (`  
`unsigned int channel )`

**11.115.3.12 GetDeviceType()** `TcxDeviceTypeEnumNet GetDeviceType ( )`

**11.115.3.13 GetDevname()** `String ^ GetDevname (`  
`unsigned int device )`

**11.115.3.14 GetDMax()** `int GetDMax (`  
`unsigned int device )`

**11.115.3.15 GetDMin()** `int GetDMin (`  
`unsigned int device )`

**11.115.3.16 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 %.

**11.115.3.17 GetEnableHeaterLimit()** `bool GetEnableHeaterLimit ( unsigned int device )`

**11.115.3.18 GetEnableThermocouple()** `bool GetEnableThermocouple ( unsigned int device )`

**11.115.3.19 GetHasThermocouple()** `bool GetHasThermocouple ( )`

Gets whether the device supports a thermocouple.

**11.115.3.20 GetHeaterLimit()** `int GetHeaterLimit ( unsigned int device )`

Gets the temperature limit of the specified heater in units of 0.1 °C.

**11.115.3.21 GetHeaterTemp()** `int GetHeaterTemp ( unsigned int channel )`

Gets the temperature of the specified heater in units of 0.1 °C.

**11.115.3.22 GetI()** `int GetI ( unsigned int device )`

Gets the I-coefficient of the specified device.

**11.115.3.23 GetIDecp()** `int GetIDecp (`  
`unsigned int device )`

**11.115.3.24 GetIMax()** `int GetIMax (`  
`unsigned int device )`

**11.115.3.25 GetIMin()** `int GetIMin (`  
`unsigned int device )`

**11.115.3.26 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.

**11.115.3.27 GetMaxHeaterPowerMultiwell()** `double GetMaxHeaterPowerMultiwell ( )`

queries the max. heater power that the Multiwell temperature controller will apply; unit: W; useful range: 5.2W..7.6W

**11.115.3.28 GetMaxP()** `int GetMaxP (`  
`unsigned int device )`

Gets the maximum heater power of the specified device.

**11.115.3.29 GetMaxpDecp()** `int GetMaxpDecp (`  
`unsigned int device )`

**11.115.3.30 GetMaxpMax()** `int GetMaxpMax ( unsigned int device )`

**11.115.3.31 GetMaxpMin()** `int GetMaxpMin ( unsigned int device )`

**11.115.3.32 GetNumControlChannels()** `unsigned int GetNumControlChannels ( )`

Gets the number of channels the device can control/regulate.

**11.115.3.33 GetNumDevices()** `unsigned int GetNumDevices ( )`

**11.115.3.34 GetNumMeasureChannels()** `unsigned int GetNumMeasureChannels ( )`

Gets the number of channels the device can measure.

**11.115.3.35 GetOnOff()** `int GetOnOff ( unsigned int channel )`

Gets if the specified channel is on or off.

**11.115.3.36 GetP()** `int GetP ( unsigned int device )`

Gets the P-coefficient of the specified device.

**11.115.3.37 GetPDecp()** `int GetPDecp ( unsigned int device )`

**11.115.3.38 GetPMax()** `int GetPMax ( unsigned int device )`

**11.115.3.39 GetPMin()** `int GetPMin ( unsigned int device )`

**11.115.3.40 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.

**11.115.3.41 GetPwrOut()** `int GetPwrOut (`  
`unsigned int channel )`

**11.115.3.42 GetPwrSet()** `int GetPwrSet (`  
`unsigned int channel )`

**11.115.3.43 GetRes1()** `int GetRes1 (`  
`unsigned int channel )`

**11.115.3.44 GetRes2()** `int GetRes2 (`  
`unsigned int channel )`

**11.115.3.45 GetResS()** `int GetResS (`  
`unsigned int channel )`

**11.115.3.46 GetResX()** `int GetResX (`  
`unsigned int channel )`

**11.115.3.47 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.

**11.115.3.48 GetSensorType()** `TcxSensorTypeEnumNet` GetSensorType (   
 unsigned int *device* )

**11.115.3.49 GetSetpoint()** `int` GetSetpoint (   
 unsigned int *channel* )

Gets the target temperate of specified channel in units of 0.1 °C.

**11.115.3.50 GetSetpointDecp()** `int` GetSetpointDecp (   
 unsigned int *channel* )

**11.115.3.51 GetSetpointMax()** `int` GetSetpointMax (   
 unsigned int *channel* )

**11.115.3.52 GetSetpointMin()** `int` GetSetpointMin (   
 unsigned int *channel* )

**11.115.3.53 GetThermocoupleCalibration()** `int` GetThermocoupleCalibration (   
 unsigned int *channel* )

**11.115.3.54 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.

**11.115.3.55 GetThermocoupleReferenceTemp()** `int GetThermocoupleReferenceTemp ( unsigned int channel )`

**11.115.3.56 GetThermocoupleTemp()** `int GetThermocoupleTemp ( unsigned int channel )`

**11.115.3.57 GetThermocoupleTempAbs()** `int GetThermocoupleTempAbs ( unsigned int channel )`

**11.115.3.58 GetUnit()** `String ^ GetUnit ( unsigned int channel )`

**11.115.3.59 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.

**11.115.3.60 GetValue()** `int GetValue (`  
`unsigned int channel )`

Gets the temperate of the specified channel in units of 0.1 °C.

**11.115.3.61 GetValueHires()** `int GetValueHires (`  
`unsigned int channel )`

Gets the temperate of the specified channel in units of 0.01 °C.

**11.115.3.62 GetVolti()** `unsigned int GetVolti (`  
`unsigned int channel )`

**11.115.3.63 SetCalibration()** `void SetCalibration (`  
`unsigned int channel,`  
`int calib )`

**11.115.3.64 SetD()** `void SetD (`  
`unsigned int device,`  
`int d_coeff )`

Sets the D-coefficient of the specified device.

**11.115.3.65 SetDevice()** `void SetDevice (`  
`unsigned int channel,`  
`int device )`

**11.115.3.66 SetDeviceType()** `void SetDeviceType (`  
`TcxDeviceTypeEnumNet devicetype )`

**11.115.3.67 SetDevname()** `void SetDevname (`  
`unsigned int device,`  
`String^ Devicename )`

**11.115.3.68 SetEnableHeaterLimit()** `void SetEnableHeaterLimit (`  
    `unsigned int device,`  
    `bool enable )`

**11.115.3.69 SetEnableThermocouple()** `void SetEnableThermocouple (`  
    `unsigned int device,`  
    `bool enable )`

**11.115.3.70 SetHeaterLimit()** `void SetHeaterLimit (`  
    `unsigned int device,`  
    `int heater_limit )`

**11.115.3.71 SetI()** `void SetI (`  
    `unsigned int device,`  
    `int i_coeff )`

Sets the I-coefficient of the specified device.

**11.115.3.72 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

**11.115.3.73 SetMaxP()** `void SetMaxP (`  
    `unsigned int device,`  
    `int maxp )`

Sets the maximum heater power of the specified device.

**11.115.3.74 SetOnOff()** `void SetOnOff (`  
    `unsigned int channel,`  
    `bool on )`

Switches the specified channel on or off.

#### Parameters

|                |                     |
|----------------|---------------------|
| <i>channel</i> | The channel number. |
|----------------|---------------------|

**11.115.3.75 SetP()** `void SetP (`  
     `unsigned int device,`  
     `int p_coeff )`

Sets the P-coefficient of the specified device.

**11.115.3.76 SetSensorType()** `void SetSensorType (`  
     `unsigned int device,`  
     `TcxSensorTypeEnumNet type )`

**11.115.3.77 SetSetpoint()** `void SetSetpoint (`  
     `unsigned int channel,`  
     `int sp )`

Sets the target temperate of specified channel in units of 0.1 °C.

**11.115.3.78 SetThermocoupleNanovoltPerKelvin()** `void SetThermocoupleNanovoltPerKelvin (`  
     `unsigned int channel,`  
     `unsigned int value )`

Sets the proportional constant for the thermocouple.

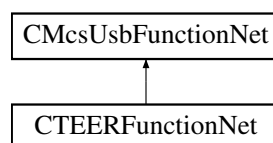
#### Parameters

|                |                                              |
|----------------|----------------------------------------------|
| <i>channel</i> | Thermocouple channel number.                 |
| <i>value</i>   | Proportinal constant in Nanovolt per Kelvin. |

## 11.116 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

### 11.116.1 Detailed Description

[CTEERFunctionNet](#) is the class to control the TEER device

### 11.116.2 Constructor & Destructor Documentation

**11.116.2.1 CTEERFunctionNet()** [1/2] [CTEERFunctionNet](#) (  
     CMcsUsbNet^ mcsusb,  
     CMcsUsbFunctionPointerContainer^ pTEERFunctionPointerContainer )

Initializes a new instance of the [CTEERFunctionNet](#) class.

**11.116.2.2 CTEERFunctionNet()** [2/2] `CTEERFunctionNet ( CMcsUsbNet^ mcsusb )`

**11.116.2.3 ~CTEERFunctionNet()** `virtual ~CTEERFunctionNet ( ) [virtual]`

**11.116.2.4 !CTEERFunctionNet()** `!CTEERFunctionNet ( )`

### 11.116.3 Member Function Documentation

**11.116.3.1 CancelInternalCalibration()** `void CancelInternalCalibration ( )`

in case the internal calibration "hangs", this will cancel it

**11.116.3.2 GetAdapterCode()** `uint32_t GetAdapterCode ( )`

gets the adapter code

#### Returns

the adapter code

**11.116.3.3 GetAdcOffsetU1()** `int GetAdcOffsetU1 ( )`

returns ADC offset of U1 channel (result of internal calibration)

#### Returns

the ADC offset for U1

**11.116.3.4 GetAdcOffsetU2()** `int GetAdcOffsetU2 ( )`

returns ADC offset of U2 channel (result of internal calibration)

#### Returns

the ADC offset for U2



**11.116.3.5 GetAmplitude\_nA()** `uint32_t GetAmplitude_nA ( )`

gets TEER stimulation amplitude in nA

**Returns**

current stimulation amplitude in nA

**11.116.3.6 GetBytesPerSample()** `uint32_t GetBytesPerSample ( )`

private function to be called internally only

**Returns**

not documented

**11.116.3.7 GetClampMode()** `TeerClampModeEnumNet GetClampMode ( )`

gets TEER clamp mode (voltage/current)

**Returns**

current TEER clamp mode

**11.116.3.8 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 |

**11.116.3.9 GetCurrentEnable()** `bool GetCurrentEnable ( )`

when disabled, no current will flow through chamber

**Returns**

false when disabled, true when enabled

**11.116.3.10 GetDacZero()** `int GetDacZero ( )`

returns DAC-offset (result of internal calibration); use to check for plausibility only

**Returns**

the DAC offset

**11.116.3.11 GetFrameErrorCounter()** `int GetFrameErrorCounter ( )`

returns number of times (since bootup) sample memory got overwritten

**Returns**

the number of errors

**11.116.3.12 GetLiquidResistance()** `int32_t GetLiquidResistance ( )`

gets the resistance of the liquid in ohms

**Returns**

the resistance in ohms

**11.116.3.13 GetMaxChunkSize\_Byte()** `uint32_t GetMaxChunkSize_Byte ( )`

private function to be called internally only

**Returns**

not documented

**11.116.3.14 GetNumberOfAvailableSamples()** `uint32_t GetNumberOfAvailableSamples ( )`

private function to be called internally only

**Returns**

not documented

**11.116.3.15 GetPeriod\_us()** `uint32_t GetPeriod_us ( )`

gets the period of TEER stimulation in us

**Returns**

the period in us

**11.116.3.16 GetRotaryPositionCode()** `uint32_t GetRotaryPositionCode ( )`

gets the rotary position code

**Returns**

the rotary position code

**11.116.3.17 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

**11.116.3.18 GetSampleRate()** `int GetSampleRate ( )`

returns sample rate in Hz

**Returns**

the sample rate in Hz

**11.116.3.19 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

**11.116.3.20 GetScaleFactorU1()** `int GetScaleFactorU1 ( )`

returns U1 scale factor times  $10^6$  (result of internal calibration)

**Returns**

the U1 scale factor

**11.116.3.21 GetScaleFactorU2()** `int GetScaleFactorU2 ( )`

returns U2 scale factor times  $10^6$  (result of internal calibration)

**Returns**

the U2 scale factor

**11.116.3.22 GetUptimeSeconds()** `int32_t GetUptimeSeconds ( )`

returns time in seconds since device was powered up

**Returns**

seconds since power-on

**11.116.3.23 GetWaveform()** [TeerWaveformEnumNet](#) GetWaveform ( )

gets TEER stimulation waveform (sine/rect)

**Returns**

waveform enum

**11.116.3.24 IsInternalCalibrationFinished()** [bool](#) IsInternalCalibrationFinished ( )

queries whether internal calibration has finished

**Returns**

true if calibration has finished

**11.116.3.25 IsSamplingFinished()** [uint32\\_t](#) IsSamplingFinished ( )

returns false iff stimulation/sampling is going on, otherwise true

**Returns**

true if sampling is finished

**11.116.3.26 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 |
|---------------------|---------------------------------|

**11.116.3.27 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 |
|-----------------------|---------------------------------------------|

**11.116.3.28 SetClampMode()** `void SetClampMode (`  
    `TeerClampModeEnumNet ClampMode )`

sets TEER clamp mode (voltage/current)

Parameters

|                  |                     |
|------------------|---------------------|
| <i>ClampMode</i> | new TEER clamp mode |
|------------------|---------------------|

**11.116.3.29 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 |

**11.116.3.30 SetCurrentEnable()** `void SetCurrentEnable (`  
    `bool NewCurrentEnable )`

when disabled, no current will flow through chamber

Parameters

|                         |                                        |
|-------------------------|----------------------------------------|
| <i>NewCurrentEnable</i> | disabled when false, enabled when true |
|-------------------------|----------------------------------------|

**11.116.3.31 SetExternalLED()** `void SetExternalLED (`  
    `uint32_t NewState )`

sets the external LED

Parameters

|                 |       |
|-----------------|-------|
| <i>NewState</i> | state |
|-----------------|-------|

**11.116.3.32 SetLiquidResistance()** `void SetLiquidResistance (`  
`int32_t NewLiquidResistance_Ohm )`

sets the resistance of the liquid in ohms

Parameters

|                                |                        |
|--------------------------------|------------------------|
| <i>NewLiquidResistance_Ohm</i> | the resistance in ohms |
|--------------------------------|------------------------|

**11.116.3.33 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 |
|------------------|------------------|

**11.116.3.34 SetWaveform()** `void SetWaveform (`  
`TeerWaveformEnumNet Waveform )`

sets TEER stimulation waveform (sine/rect)

Parameters

|                 |               |
|-----------------|---------------|
| <i>Waveform</i> | waveform enum |
|-----------------|---------------|

**11.116.3.35 StartInternalCalibration()** `void StartInternalCalibration ( )`

starts determination of internal DAC-offset; result is used internally; NON-BLOCKING call

**11.116.3.36 StartSampling()** `void StartSampling (`  
`uint32_t NumberOfCycles )`

starts TEER stimulation (duration: n cycles) and samples during last cycle

## Parameters

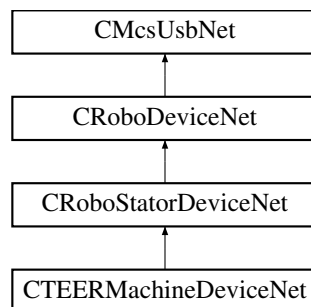
|                       |                                                              |
|-----------------------|--------------------------------------------------------------|
| <i>NumberOfCycles</i> | number of cycles (sine or rect) to output (0 - loop forever) |
|-----------------------|--------------------------------------------------------------|

**11.116.3.37 StopSampling()** `void StopSampling ( )`

stops TEER stimulation and sampling

**11.117 CTEERMachineDeviceNet Class Reference**

Inheritance diagram for CTEERMachineDeviceNet:

**Public Member Functions**

- [CTEERMachineDeviceNet \( \)](#)
- [~CTEERMachineDeviceNet \( \)](#)

**Properties**

- [CTEERFunctionNet^ TEERFunctionNet](#) [get]

**Additional Inherited Members****11.117.1 Constructor & Destructor Documentation****11.117.1.1 CTEERMachineDeviceNet()** [CTEERMachineDeviceNet \( \)](#)**11.117.1.2 ~CTEERMachineDeviceNet()** [~CTEERMachineDeviceNet \( \)](#)



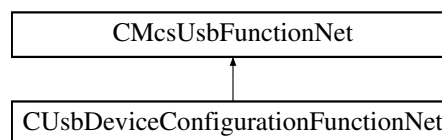
### 11.117.2 Property Documentation

**11.117.2.1 TEERFunctionNet** [CTEERFunctionNet](#)<sup>^</sup> [TEERFunctionNet](#) [get]

## 11.118 CUsbDeviceConfigurationFunctionNet Class Reference

[CUsbDeviceConfigurationFunctionNet](#) is the class to configure the USB firmware

Inheritance diagram for CUsbDeviceConfigurationFunctionNet:



### Public Member Functions

- [CUsbDeviceConfigurationFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb, [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> p↔  
UsbDeviceConfigurationFunctionPointerContainer)  
*Initializes a new instance of the [CUsbDeviceConfigurationFunctionNet](#) class.*
- [CUsbDeviceConfigurationFunctionNet](#) ([CMcsUsbNet](#)<sup>^</sup> mcsusb)
- virtual [~CUsbDeviceConfigurationFunctionNet](#) ()
- [ICUsbDeviceConfigurationFunctionNet](#) ()
- void [SetDeviceName](#) (String<sup>^</sup> name)  
*sets the USB device name for configurable devices*
- void [SetDeviceId](#) ([ProductIdEnumNet](#) id)  
*sets the USB device name for configurable devices*

### Additional Inherited Members

#### 11.118.1 Detailed Description

[CUsbDeviceConfigurationFunctionNet](#) is the class to configure the USB firmware

#### 11.118.2 Constructor & Destructor Documentation

**11.118.2.1 CUsbDeviceConfigurationFunctionNet()** [1/2] [CUsbDeviceConfigurationFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb,  
[CMcsUsbFunctionPointerContainer](#)<sup>^</sup> pUsbDeviceConfigurationFunctionPointerContainer  
)

Initializes a new instance of the [CUsbDeviceConfigurationFunctionNet](#) class.

**11.118.2.2 CUsbDeviceConfigurationFunctionNet()** [2/2] `CUsbDeviceConfigurationFunctionNet ( CMcsUsbNet^ mcsusb )`

**11.118.2.3 ~CUsbDeviceConfigurationFunctionNet()** `virtual ~CUsbDeviceConfigurationFunctionNet ( ) [virtual]`

**11.118.2.4 !CUsbDeviceConfigurationFunctionNet()** `!CUsbDeviceConfigurationFunctionNet ( )`

### 11.118.3 Member Function Documentation

**11.118.3.1 SetDeviceId()** `void SetDeviceId ( ProductIdEnumNet id )`

sets the USB device name for configurable devices

#### Parameters

|           |  |
|-----------|--|
| <i>id</i> |  |
|-----------|--|

**11.118.3.2 SetDeviceName()** `void SetDeviceName ( String^ name )`

sets the USB device name for configurable devices

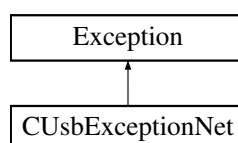
#### Parameters

|             |  |
|-------------|--|
| <i>name</i> |  |
|-------------|--|

## 11.119 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]

### 11.119.1 Detailed Description

Exception class that is thrown in case of an USB error.

### 11.119.2 Constructor & Destructor Documentation

#### 11.119.2.1 CUsbExceptionNet() [1/2] [CUsbExceptionNet](#) ( uint32\_t status )

Constructor of a CUsbException.

#### Parameters

|               |                   |
|---------------|-------------------|
| <i>status</i> | the status number |
|---------------|-------------------|

#### 11.119.2.2 CUsbExceptionNet() [2/2] [CUsbExceptionNet](#) ( uint32\_t status, String^ message )

### 11.119.3 Property Documentation

#### 11.119.3.1 Status uint32\_t Status [get]

## 11.120 CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet Class Reference

## Public Member Functions

- [CVoltageRangeInfoNet](#) (int vr, String^ vrString)

## Public Attributes

- int [VoltageRangeInMicroVolt](#)
- String ^ [VoltageRangeDisplayStringMilliVolt](#)

### 11.120.1 Constructor & Destructor Documentation

**11.120.1.1 CVoltageRangeInfoNet()** [CVoltageRangeInfoNet](#) (  
    int *vr*,  
    String^ *vrString* )

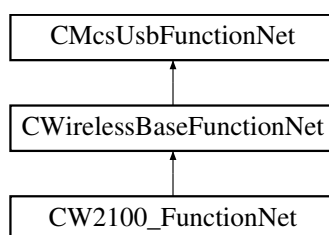
### 11.120.2 Member Data Documentation

**11.120.2.1 VoltageRangeDisplayStringMilliVolt** String ^ VoltageRangeDisplayStringMilliVolt

**11.120.2.2 VoltageRangeInMicroVolt** int VoltageRangeInMicroVolt

## 11.121 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

#### 11.121.1 Constructor & Destructor Documentation

**11.121.1.1 CW2100\_FunctionNet() [1/2]** [CW2100\\_FunctionNet](#) (  
    [CMcsUsbNet](#)<sup>^</sup> *mcsusb*,  
    [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> *w2100\_FunctionPointerContainer* )

**11.121.1.2 CW2100\_FunctionNet() [2/2]** [CW2100\\_FunctionNet](#) (  
    [CMcsUsbNet](#)<sup>^</sup> *mcsusb* )

#### 11.121.2 Member Function Documentation

**11.121.2.1 ClearStimulusParametersCache() [1/2]** static void [ClearStimulusParametersCache](#) ( )  
[static]

**11.121.2.2 ClearStimulusParametersCache() [2/2]** static void [ClearStimulusParametersCache](#) (  
    unsigned short *ID* ) [static]

**11.121.2.3 ClearUserDefinedNameCache() [1/2]** static void [ClearUserDefinedNameCache](#) ( ) [static]

**11.121.2.4 ClearUserDefinedNameCache()** [2/2] static void ClearUserDefinedNameCache ( unsigned short *ID* ) [static]

**11.121.2.5 DeselectAllHeadstages()** void DeselectAllHeadstages ( )

**11.121.2.6 DeselectHeadstage()** void DeselectHeadstage ( int *TimeSlotNr* )

**11.121.2.7 GetAccelGyroCurrentRate()** int GetAccelGyroCurrentRate ( int *TimeSlotNr* )

**11.121.2.8 GetAccelGyroDesiredRate()** int GetAccelGyroDesiredRate ( int *TimeSlotNr* )

**11.121.2.9 GetAccelGyroEnabled()** [W2100\\_Accel\\_Gyro\\_Select\\_EnumNet](#) GetAccelGyroEnabled ( int *TimeSlotNr* )

**11.121.2.10 GetAccelRange()** int GetAccelRange ( int *TimeSlotNr* )

**11.121.2.11 GetAnalogOutChannel()** unsigned int GetAnalogOutChannel ( [System::Runtime::InteropServices::Out] int % *automatic*, unsigned short *index* )

**11.121.2.12 GetAnalogOutFilter()** array<unsigned int> ^ GetAnalogOutFilter ( [System::Runtime::InteropServices::Out] int % *automatic* )

**11.121.2.13 GetAudioChannels()** array<[AudioChannelsNet](#)> ^ GetAudioChannels ( )

**11.121.2.14 GetAvailableHeadstages()** `array<HeadStageIDType^> ^ GetAvailableHeadstages (`  
`unsigned int max_length )`

**11.121.2.15 GetBatteryState()** `BatteryState ^ GetBatteryState (`  
`int TimeSlotNr )`

**11.121.2.16 GetDacRange()** `AnalogOut_DAC_Range_EnumNet GetDacRange ( )`

**11.121.2.17 GetFilterProperties()** `array<CFilterPropertyNet^> ^ GetFilterProperties (`  
`W2100DacqGroupChannelEnumNet GroupID )`

**11.121.2.18 GetFilterProperty()** `CFilterPropertyNet ^ GetFilterProperty (`  
`W2100DacqGroupChannelEnumNet GroupID,`  
`unsigned int index )`

**11.121.2.19 GetFPGA FirmwareType()** `unsigned int GetFPGA FirmwareType (`  
`int TimeSlotNr )`

**11.121.2.20 GetGyroRange()** `int GetGyroRange (`  
`int TimeSlotNr )`

**11.121.2.21 GetHeadstageOnOff()** `unsigned short GetHeadstageOnOff (`  
`int TimeSlotNr )`

**11.121.2.22 GetHeadstageSamplingActive()** `bool GetHeadstageSamplingActive (`  
`int TimeSlotNr )`

**11.121.2.23 GetMultiHeadstageMode()** `bool GetMultiHeadstageMode ( )`



**11.121.2.24 GetPicFirmwareType()** unsigned int GetPicFirmwareType (   
 int *TimeSlotNr* )

**11.121.2.25 GetSelectedChannels()** array<BYTE> ^ GetSelectedChannels (   
 int *TimeSlotNr* )

**11.121.2.26 GetSelectedHeadstageState()** HeadStageIDTypeState ^ GetSelectedHeadstageState (   
 int *TimeSlotNr* )

**11.121.2.27 GetStimulusParametersCache()** [1/2] W2100\_StimulusParametersNet ^ GetStimulus↔   
 ParametersCache (   
 unsigned int *typeValue* )

**11.121.2.28 GetStimulusParametersCache()** [2/2] uint32\_t GetStimulusParametersCache (   
 unsigned int *typeValue*,   
 [System::Runtime::InteropServices::Out] W2100\_StimulusParametersNet^% *Stimulus↔   
 Parameters* )

**11.121.2.29 GetStimulusParametersFromSelectedHS()** W2100\_StimulusParametersNet ^ GetStimulus↔   
 ParametersFromSelectedHS (   
 int *TimeSlotNr* )

**11.121.2.30 GetStiumlusParameters()** W2100\_StimulusParametersNet ^ GetStiumlusParameters (   
 unsigned short *ID* )

**11.121.2.31 GetUserDefinedName()** System::String ^ GetUserDefinedName (   
 unsigned short *ID* )

**11.121.2.32 GetUserDefinedNameCache()** [1/2] System::String ^ GetUserDefinedNameCache (   
 unsigned short *ID* )

**11.121.2.33 GetUserDefinedNameCache()** [2/2] uint32\_t GetUserDefinedNameCache ( unsigned short *ID*, [System::Runtime::InteropServices::Out] System::String^% *Name* )

**11.121.2.34 GetUserDefinedNameFromSelectedHS()** System::String ^ GetUserDefinedNameFromSelectedHS ( int *TimeSlotNr* )

**11.121.2.35 SelectHeadstage()** void SelectHeadstage ( unsigned int *IDorEntry*, int *TimeSlotNr* )

**11.121.2.36 SetAccelGyroDesiredRate()** void SetAccelGyroDesiredRate ( int *rate*, int *TimeSlotNr* )

**11.121.2.37 SetAccelGyroEnabled()** void SetAccelGyroEnabled ( W2100\_Accel\_Gyro\_Select\_EnumNet *enable*, int *TimeSlotNr* )

**11.121.2.38 SetAccelRange()** void SetAccelRange ( int *range*, int *TimeSlotNr* )

**11.121.2.39 SetAnalogOutChannel()** void SetAnalogOutChannel ( int *automatic*, unsigned short *index*, unsigned int *Channel* )

**11.121.2.40 SetAnalogOutFilter()** void SetAnalogOutFilter ( int *automatic*, array< unsigned int >^ *Coeffs* )

**11.121.2.41 SetAudioChannels()** void SetAudioChannels (   
array< AudioChannelsNet^>^ channels )

**11.121.2.42 SetDacRange()** void SetDacRange (   
AnalogOut\_DAC\_Range\_EnumNet range )

**11.121.2.43 SetGyroRange()** void SetGyroRange (   
int range,   
int TimeSlotNr )

**11.121.2.44 SetHeadstageOnOff()** void SetHeadstageOnOff (   
unsigned short On,   
int TimeSlotNr )

**11.121.2.45 SetHeadstageSamplingActive()** void SetHeadstageSamplingActive (   
bool Active,   
int TimeSlotNr )

**11.121.2.46 SetHeadstageToSleep()** void SetHeadstageToSleep (   
unsigned int Sleep16ms,   
int TimeSlotNr )

**11.121.2.47 SetMultiHeadstageMode()** void SetMultiHeadstageMode (   
bool Mode )

**11.121.2.48 SetSelectedChannels()** void SetSelectedChannels (   
array< BYTE >^ channels,   
int TimeSlotNr )

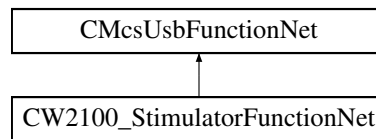
### 11.121.3 Property Documentation

**11.121.3.1 PulseGenerator** `CPulseGeneratorFunctionNet^ PulseGenerator [get]`

**11.121.3.2 Stimulator** `CW2100_StimulatorFunctionNet^ Stimulator [get]`

## 11.122 CW2100\_StimulatorFunctionNet Class Reference

Inheritance diagram for CW2100\_StimulatorFunctionNet:



### Public Member Functions

- `CW2100_StimulatorFunctionNet (CMcsUsbNet^ mcsusb)`
- void `SendStart (uint32_t triggermap)`  
*Start (Trigger) the STG. The startup delay is in the range of a few ms.*
- void `SendStop (uint32_t triggermap)`  
*Stop some or all triggers of the STG.*
- `CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData ^ PrepareData (int channel, array< int32_t >^ amplitude, array< uint64_t >^ duration, STG_DestinationEnumNet destType, uint32_t repeat)`
- `CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData ^ PrepareDataSync (int channel, array< int32_t >^ amplitude, array< uint32_t >^ Sync, array< uint64_t >^ duration, STG_DestinationEnumNet destType, uint32_t repeat)`
- void `SendPreparedData (int channel, CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData^ device_data_and_unrolled, STG_DestinationEnumNet destType)`
- void `ClearChannelData (int channel)`  
*Delete a Stimulus Pattern from STG memory*
- int `GetDACResolution ()`  
*Gets number of bits of the DAC resolution.*
- int `GetTimeResolutionInNanoSeconds ()`  
*Gets number of bits of the DAC resolution.*
- int `GetVoltageRangeInMicroVolt (uint32_t channel)`  
*Gets the Voltage Range of the specified channel in Microvolts.*
- int `GetVoltageResolutionInMicroVolt (uint32_t channel)`  
*Gets the Voltage Resolution of the specified channel in Microvolts.*
- int `GetCurrentRangeInNanoAmp (uint32_t channel)`  
*Gets the Current Range of the specified channel in Nanoamps.*
- int `GetCurrentResolutionInNanoAmp (uint32_t channel)`  
*Gets the Current Resolution of the specified channel in Nanoamps.*
- uint32\_t `GetNumberOfAnalogChannels ()`
- uint32\_t `GetNumberOfSyncoutChannels ()`
- uint32\_t `GetNumberOfTriggerInputs ()`
- void `SelectTimeSlot (int TimeSlotNr)`
- int `GetTimeSlot ()`
- uint32\_t `GetStimulationPatternMemory ()`

- uint32\_t [GetBoostPreTime](#) ()
- uint32\_t [GetBoostAlwaysOnMode](#) ()
- void [SetDigitalStimulatorTrigger](#) (int TimeSlotNr, [DigitalStimulatorTriggerEventEnumNet](#) trigger\_event, int trigger\_number, [W2100DigitalSourceEnumNet](#) digstream\_source, int bitnumber\_offset)
- void [GetDigitalStimulatorTrigger](#) (int TimeSlotNr, [DigitalStimulatorTriggerEventEnumNet](#) trigger\_event, int trigger\_number, [System::Runtime::InteropServices::Out][W2100DigitalSourceEnumNet](#)% digstream\_source, [System::Runtime::InteropServices::Out]int% bitnumber\_offset)
- void [SetDigitalStimulatorTriggerSlope](#) (int TimeSlotNr, [DigitalStimulatorTriggerEventEnumNet](#) trigger\_event, int trigger\_number, [DigitalStimulatorTriggerSlopeEnumNet](#) slope)
- [DigitalStimulatorTriggerSlopeEnumNet](#) [GetDigitalStimulatorTriggerSlope](#) (int TimeSlotNr, [DigitalStimulatorTriggerEventEnumNet](#) trigger\_event, int trigger\_number)
- void [StartPoll](#) ()
- void [StopPoll](#) ()

### Static Public Attributes

- static const uint32\_t [BOOST\\_BIT](#) = (1 << 0)
- static const uint32\_t [GND\\_SWITCH\\_BIT](#) = (1 << 1)
- static const uint32\_t [SYNC\\_BIT0](#) = (1 << 2)
- static const uint32\_t [SYNC\\_BIT1](#) = (1 << 3)

### Events

- [OnStgPollStatus](#)<sup>^</sup> [PollStatusEvent](#)

### Additional Inherited Members

## 11.122.1 Constructor & Destructor Documentation

**11.122.1.1 CW2100\_StimulatorFunctionNet()** [CW2100\\_StimulatorFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> *mcsusb* )

## 11.122.2 Member Function Documentation

**11.122.2.1 ClearChannelData()** void [ClearChannelData](#) (  
int *channel* )

Delete a Stimulus Pattern from STG memory

#### Parameters

|                |                                 |
|----------------|---------------------------------|
| <i>channel</i> | specifies the channel to clear. |
|----------------|---------------------------------|

**11.122.2.2 GetBoostAlwaysOnMode()** `uint32_t GetBoostAlwaysOnMode ( )`

**11.122.2.3 GetBoostPreTime()** `uint32_t GetBoostPreTime ( )`

**11.122.2.4 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.

**11.122.2.5 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.

**11.122.2.6 GetDACResolution()** `int GetDACResolution ( )`

Gets number of bits of the DAC resolution.

**Returns**

The DAC resolution in bits.

**11.122.2.7 GetDigitalStimulatorTrigger()** void GetDigitalStimulatorTrigger (   
     int TimeSlotNr,   
     DigitalStimulatorTriggerEventEnumNet trigger\_event,   
     int trigger\_number,   
     [System::Runtime::InteropServices::Out] W2100DigitalSourceEnumNet% digstream\_↵   
 source,   
     [System::Runtime::InteropServices::Out] int% bitnumber\_offset )

**11.122.2.8 GetDigitalStimulatorTriggerSlope()** DigitalStimulatorTriggerSlopeEnumNet GetDigital↵   
 StimulatorTriggerSlope (   
     int TimeSlotNr,   
     DigitalStimulatorTriggerEventEnumNet trigger\_event,   
     int trigger\_number )

**11.122.2.9 GetNumberOfAnalogChannels()** uint32\_t GetNumberOfAnalogChannels ( )

**11.122.2.10 GetNumberOfSyncoutChannels()** uint32\_t GetNumberOfSyncoutChannels ( )

**11.122.2.11 GetNumberOfTriggerInputs()** uint32\_t GetNumberOfTriggerInputs ( )

**11.122.2.12 GetStimulationPatternMemory()** uint32\_t GetStimulationPatternMemory ( )

**11.122.2.13 GetTimeResolutionInNanoSeconds()** int GetTimeResolutionInNanoSeconds ( )

Gets number of bits of the DAC resolution.

#### Returns

The time resolution in ns.

**11.122.2.14 GetTimeSlot()** int GetTimeSlot ( )

**11.122.2.15 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.

**11.122.2.16 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.

**11.122.2.17 PrepareData()** `CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData ^ PrepareData (`

```
int channel,
array< int32_t >^ amplitude,
array< uint64_t >^ duration,
STG_DestinationEnumNet destType,
uint32_t repeat)
```

**11.122.2.18 PrepareDataSync()** `CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData ^ PrepareDataSync (`

```
int channel,
array< int32_t >^ amplitude,
array< uint32_t >^ Sync,
array< uint64_t >^ duration,
STG_DestinationEnumNet destType,
uint32_t repeat)
```

**11.122.2.19 SelectTimeSlot()** `void SelectTimeSlot ( int TimeSlotNr )`



**11.122.2.20 SendPreparedData()** `void SendPreparedData (`  
     `int channel,`  
     `CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData^ device_data_and_unrolled,`  
     `STG_DestinationEnumNet destType )`

**11.122.2.21 SendStart()** `void SendStart (`  
     `uint32_t triggermap )`

Start (Trigger) the STG. The startup delay is in the range of a few ms.

#### Parameters

|                   |                                             |
|-------------------|---------------------------------------------|
| <i>triggermap</i> | A bitmap of triggers which will be started. |
|-------------------|---------------------------------------------|

**11.122.2.22 SendStop()** `void SendStop (`  
     `uint32_t triggermap )`

Stop some or all triggers of the STG.

#### Parameters

|                   |                                             |
|-------------------|---------------------------------------------|
| <i>triggermap</i> | A bitmap of triggers which will be stopped. |
|-------------------|---------------------------------------------|

**11.122.2.23 SetDigitalStimulatorTrigger()** `void SetDigitalStimulatorTrigger (`  
     `int TimeSlotNr,`  
     `DigitalStimulatorTriggerEventEnumNet trigger_event,`  
     `int trigger_number,`  
     `W2100DigitalSourceEnumNet digstream_source,`  
     `int bitnumber_offset )`

**11.122.2.24 SetDigitalStimulatorTriggerSlope()** `void SetDigitalStimulatorTriggerSlope (`  
     `int TimeSlotNr,`  
     `DigitalStimulatorTriggerEventEnumNet trigger_event,`  
     `int trigger_number,`  
     `DigitalStimulatorTriggerSlopeEnumNet slope )`

**11.122.2.25 StartPoll()** `void StartPoll ( )`

**11.122.2.26 StopPoll()** `void StopPoll ( )`

### 11.122.3 Member Data Documentation

**11.122.3.1 BOOST\_BIT** `const uint32_t BOOST_BIT = (1 << 0) [static]`

**11.122.3.2 GND\_SWITCH\_BIT** `const uint32_t GND_SWITCH_BIT = (1 << 1) [static]`

**11.122.3.3 SYNC\_BIT0** `const uint32_t SYNC_BIT0 = (1 << 2) [static]`

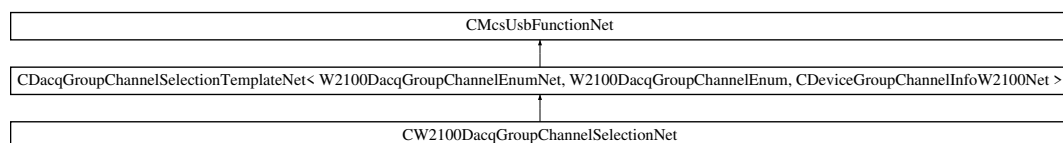
**11.122.3.4 SYNC\_BIT1** `const uint32_t SYNC_BIT1 = (1 << 3) [static]`

### 11.122.4 Event Documentation

**11.122.4.1 PollStatusEvent** `OnStgPollStatus^ PollStatusEvent`

## 11.123 CW2100DacqGroupChannelSelectionNet Class Reference

Inheritance diagram for CW2100DacqGroupChannelSelectionNet:



### Public Member Functions

- `CW2100DacqGroupChannelSelectionNet (CMcsUsbNet^ mcsusb)`

### Additional Inherited Members

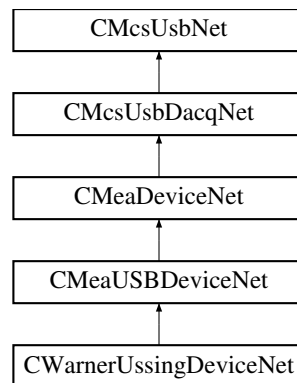
#### 11.123.1 Constructor & Destructor Documentation

**11.123.1.1 CW2100DacqGroupChannelSelectionNet()** `CW2100DacqGroupChannelSelectionNet ( CMcsUsbNet^ mcsusb )`

## 11.124 CWarnerUssingDeviceNet Class Reference

[CWarnerUssingDeviceNet](#) is the class to control the Ussing device

Inheritance diagram for CWarnerUssingDeviceNet:



### Public Member Functions

- [CWarnerUssingDeviceNet \(\)](#)  
*Initializes a new instance of the [CWarnerUssingDeviceNet](#) class.*
- virtual [~CWarnerUssingDeviceNet \(\)](#)
- [ICWarnerUssingDeviceNet \(\)](#)

### Properties

- [CWarnerUssingFunctionNet^ WarnerUssingFunction](#) [get]

### Additional Inherited Members

#### 11.124.1 Detailed Description

[CWarnerUssingDeviceNet](#) is the class to control the Ussing device

#### 11.124.2 Constructor & Destructor Documentation

**11.124.2.1 CWarnerUssingDeviceNet()** `CWarnerUssingDeviceNet ( )`

Initializes a new instance of the [CWarnerUssingDeviceNet](#) class.

**11.124.2.2** `~CWarnerUssingDeviceNet()` `virtual ~CWarnerUssingDeviceNet ( ) [virtual]`

**11.124.2.3** `!CWarnerUssingDeviceNet()` `!CWarnerUssingDeviceNet ( )`

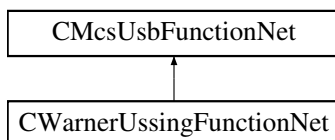
### 11.124.3 Property Documentation

**11.124.3.1 WarnerUssingFunction** `CWarnerUssingFunctionNet^ WarnerUssingFunction [get]`

## 11.125 CWarnerUssingFunctionNet Class Reference

`CWarnerUssingFunctionNet` is the class to control the Ussing device

Inheritance diagram for `CWarnerUssingFunctionNet`:



### Public Member Functions

- `CWarnerUssingFunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ pWarnerUssingFunctionPointerContainer)`  
*Initializes a new instance of the `CWarnerUssingFunctionNet` class.*
- `CWarnerUssingFunctionNet (CMcsUsbNet^ mcsusb)`
- `virtual ~CWarnerUssingFunctionNet ()`
- `!CWarnerUssingFunctionNet ()`
- `int32_t GetChannelsCountOfChamber (int32_t ChamberId)`  
*gets number of channels in datastream from chamber amp with given index*
- `int32_t GetNumberOfHardwareSlotsForChambers ()`  
*gets number of physical hardware slots for chambers amps*
- `int32_t GetNumberOfAvailableChambers ()`  
*gets number of actually connected chamber amps*
- `bool IsChamberAvailable (int32_t ChamberId)`  
*checks whether chamber amp is connected to slot*
- `void SetPulse (int32_t ChamberId, UssingClampModeEnumNet StgMode, int32_t NumberOfRepetitions, array< int >^ Amplitudes, array< int >^ Durations, array< int >^ PulseMarker)`  
*defines stimulation pulse pattern for voltage or current stimulation; CAUTION: zero-length amplitude will be briefly applied -> choose matching to neighbour to avoid spikes*
- `void SetVoltageClampControllerParam_P (int32_t ChamberId, uint32_t P)`  
*sets P value of PID controller;*
- `void SetVoltageClampControllerParam_I (int32_t ChamberId, uint32_t I)`  
*sets I value of PID controller;*

- void [SetVoltageClampControllerParam\\_D](#) (int32\_t ChamberId, uint32\_t D)  
*sets D value of PID controller;*
- uint32\_t [GetVoltageClampControllerParam\\_P](#) (int32\_t ChamberId)  
*gets P value of PID controller;*
- uint32\_t [GetVoltageClampControllerParam\\_I](#) (int32\_t ChamberId)  
*gets I value of PID controller;*
- uint32\_t [GetVoltageClampControllerParam\\_D](#) (int32\_t ChamberId)  
*gets D value of PID controller;*
- void [SetClampMode](#) (int32\_t ChamberId, [UssingClampModeEnumNet](#) NewClampMode)  
*sets clamp mode (voltage, current or open clamp)*
- [UssingClampModeEnumNet](#) [GetClampMode](#) (int32\_t ChamberId)  
*gets clamp mode (voltage, current or open clamp; do not use when device is in internal calibration mode)*
- bool [IsInternalCalibrationFinished](#) (int32\_t ChamberId)  
*when internal calibration is finished, values for U1,2\_offset and U1,2\_reference and DAC\_offset are available*
- int32\_t [GetU1Offset](#) (int32\_t ChamberId)
- int32\_t [GetU2Offset](#) (int32\_t ChamberId)
- int32\_t [GetU1Reference](#) (int32\_t ChamberId)
- int32\_t [GetU2Reference](#) (int32\_t ChamberId)
- int32\_t [GetDacZero](#) (int32\_t ChamberId)
- void [SetHighCurrentMode](#) (int32\_t ChamberId)  
*switch to high-current mode*
- void [SetLowCurrentMode](#) (int32\_t ChamberId)  
*switch to low-current mode*
- bool [IsHighCurrentMode](#) (int32\_t ChamberId)
- uint32\_t [GetLowCurrentRange](#) (int32\_t ChamberId)  
*query the range of the low current mode*
- uint32\_t [GetHighCurrentRange](#) (int32\_t ChamberId)  
*query the range of the high current mode*
- uint32\_t [GetDacPampsPerDigitLowCurrentRange](#) (int32\_t ChamberId)  
*get the resolution of the low current mode*
- uint32\_t [GetDacPampsPerDigitHighCurrentRange](#) (int32\_t ChamberId)  
*get the resolution of the high current mode*
- uint32\_t [GetUnitsPerDigit](#) (int32\_t ChamberId, int32\_t ChannelId)  
*gets amps/volts per digit for specified chamber and channel*
- int32\_t [GetUnitExponent](#) (int32\_t ChamberId, int32\_t ChannelId)  
*gets the unit exponent for specified chamber and channel*
- [UssingUnitEnumNet](#) [GetUnitName](#) (int32\_t ChamberId, int32\_t ChannelId)  
*gets the channel's unit name*
- String ^ [GetUnitDescription](#) (int32\_t ChamberId, int32\_t ChannelId)  
*gets the description for the unit*
- array< int > ^ [GetAvailableChambers](#) ()  
*returns array with (zero-based) ChamberIds of all available chambers*
- int32\_t [GetUptimeSeconds](#) (int32\_t ChamberId)  
*gets the uptime in seconds*
- void [SetIdleModeOffset](#) (int32\_t ChamberId, [UssingClampModeEnumNet](#) ClampMode, int32\_t NewIdleOffset)  
*sets the offset (voltage or current) that will be applied when clamping is DISABLED*
- int32\_t [GetIdleModeOffset](#) (int32\_t ChamberId, [UssingClampModeEnumNet](#) ClampMode)  
*gets the offset (voltage or current) that will be applied when clamping is DISABLED*
- void [SetEnablePulse](#) (int32\_t ChamberId, [UssingClampModeEnumNet](#) ClampMode, bool Enable)  
*enable pulse of given chamber and mode (voltage/current clamp) of this chamber*

- bool [IsPulseEnabled](#) (int32\_t ChamberId, [UssingClampModeEnumNet](#) ClampMode)  
*returns true when pulse of given chamber and current mode (voltage/current clamp) of this chamber is ENABLED*
- void [SetLiquidResistance](#) (int32\_t ChamberId, int32\_t NewLiquidResistance\_Ohm)  
*sets the resistance of the liquid*
- int32\_t [GetLiquidResistance](#) (int32\_t ChamberId)  
*gets the resistance of the liquid*
- int32\_t [GetComplianceVoltageThreshold](#) (int32\_t ChamberId)  
*returns compliance voltage threshold in uV; when 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

### 11.125.1 Detailed Description

[CWarnerUssingFunctionNet](#) is the class to control the Ussing device

### 11.125.2 Constructor & Destructor Documentation

**11.125.2.1 CWarnerUssingFunctionNet()** [1/2] [CWarnerUssingFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb,  
[CMcsUsbFunctionPointerContainer](#)<sup>^</sup> pWarnerUssingFunctionPointerContainer )

Initializes a new instance of the [CWarnerUssingFunctionNet](#) class.

**11.125.2.2 CWarnerUssingFunctionNet()** [2/2] [CWarnerUssingFunctionNet](#) (  
[CMcsUsbNet](#)<sup>^</sup> mcsusb )

**11.125.2.3 ~CWarnerUssingFunctionNet()** virtual [~CWarnerUssingFunctionNet](#) ( ) [virtual]

**11.125.2.4 "!CWarnerUssingFunctionNet()** [!CWarnerUssingFunctionNet](#) ( )

### 11.125.3 Member Function Documentation

**11.125.3.1 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

**11.125.3.2 GetAvailableChambers()** `array<int> ^ GetAvailableChambers ( )`

returns array with (zero-based) ChamberIds of all available chambers

**11.125.3.3 GetChannelsCountOfChamber()** `int32_t GetChannelsCountOfChamber (   
int32_t ChamberId )`

gets number of channels in datastream from chamber amp with given index

**Parameters**

|                        |                                             |
|------------------------|---------------------------------------------|
| <i>Chamber↔<br/>Id</i> | index of hardware chamber slot (zero-based) |
|------------------------|---------------------------------------------|

**Returns**

return value of zero means that amp is not placed

**11.125.3.4 GetClampMode()** `UssingClampModeEnumNet GetClampMode (   
int32_t ChamberId )`

gets clamp mode (voltage, current or open clamp; do not use when device is in internal calibration mode)

**Parameters**

|                        |                                             |
|------------------------|---------------------------------------------|
| <i>Chamber↔<br/>Id</i> | index of hardware chamber slot (zero-based) |
|------------------------|---------------------------------------------|

**Returns**

the current clamp mode

**11.125.3.5 GetComplianceVoltageThreshold()** `int32_t GetComplianceVoltageThreshold (`  
`int32_t ChamberId )`

returns compliance voltage threshold in uV; when  $U_c$  is above, current source is overloaded

**Parameters**

|                  |                                             |
|------------------|---------------------------------------------|
| <i>ChamberId</i> | index of hardware chamber slot (zero-based) |
|------------------|---------------------------------------------|

**Returns**

the compliance voltage threshold in uV

**11.125.3.6 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

**11.125.3.7 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



**11.125.3.8 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

**11.125.3.9 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

**11.125.3.10 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

**11.125.3.11 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

**11.125.3.12 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

**11.125.3.13 GetNumberOfAvailableChambers()** `int32_t GetNumberOfAvailableChambers ( )`

gets number of actually connected chamber amps

**Returns**

the number of actually connected chambers

**11.125.3.14 GetNumberOfHardwareSlotsForChambers()** `int32_t GetNumberOfHardwareSlotsFor↔  
Chambers ( )`

gets number of physical hardware slots for chambers amps

**Returns**

the number of hardware chamber slots on the backplane

**11.125.3.15 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

**11.125.3.16 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

**11.125.3.17 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

**11.125.3.18 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

**11.125.3.19 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

**11.125.3.20 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

**11.125.3.21 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

**11.125.3.22 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

**11.125.3.23 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

**11.125.3.24 GetVoltageClampControllerParam\_D()** `uint32_t GetVoltageClampControllerParam_D (int32_t ChamberId )`

gets D value of PID controller;

#### Parameters

|                  |                                             |
|------------------|---------------------------------------------|
| <i>ChamberId</i> | index of hardware chamber slot (zero-based) |
|------------------|---------------------------------------------|

#### Returns

the D value

**11.125.3.25 GetVoltageClampControllerParam\_I()** `uint32_t GetVoltageClampControllerParam_I (int32_t ChamberId )`

gets I value of PID controller;

#### Parameters

|                  |                                             |
|------------------|---------------------------------------------|
| <i>ChamberId</i> | index of hardware chamber slot (zero-based) |
|------------------|---------------------------------------------|

#### Returns

the I value

**11.125.3.26 GetVoltageClampControllerParam\_P()** `uint32_t GetVoltageClampControllerParam_P (int32_t ChamberId )`

gets P value of PID controller;

#### Parameters

|                  |                                             |
|------------------|---------------------------------------------|
| <i>ChamberId</i> | index of hardware chamber slot (zero-based) |
|------------------|---------------------------------------------|

#### Returns

the P value

**11.125.3.27 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

**11.125.3.28 IsHighCurrentMode()** `bool IsHighCurrentMode (`  
    `int32_t ChamberId )`

**Parameters**

|                  |                                             |
|------------------|---------------------------------------------|
| <i>ChamberId</i> | index of hardware chamber slot (zero-based) |
|------------------|---------------------------------------------|

**Returns**

true if in high current mode

**11.125.3.29 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

**11.125.3.30 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

**11.125.3.31 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                       |

**11.125.3.32 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 |

**11.125.3.33 SetHighCurrentMode()** `void SetHighCurrentMode (`  
    `int32_t ChamberId )`

switch to high-current mode

## Parameters

|                        |                                             |
|------------------------|---------------------------------------------|
| <i>Chamber↵<br/>Id</i> | index of hardware chamber slot (zero-based) |
|------------------------|---------------------------------------------|

**11.125.3.34 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                              |

**11.125.3.35 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                |

**11.125.3.36 SetLowCurrentMode()** `void SetLowCurrentMode (`  
     `int32_t ChamberId )`

switch to low-current mode

## Parameters

|                        |                                             |
|------------------------|---------------------------------------------|
| <i>Chamber↵<br/>Id</i> | index of hardware chamber slot (zero-based) |
|------------------------|---------------------------------------------|

**11.125.3.37 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")        |

**11.125.3.38 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                        |

**11.125.3.39 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                 |

**11.125.3.40 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                        |

**11.125.3.41 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...)

**11.125.3.42 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) |
|------------------|---------------------------------------------|

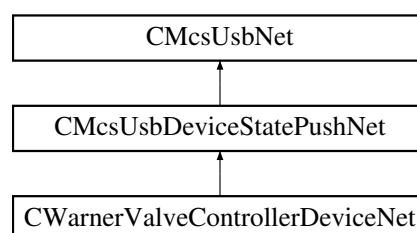
#### Returns

returns false when calibration fails (e.g. timeout...)

## 11.126 CWarnerValveControllerDeviceNet Class Reference

[CWarnerValveControllerDeviceNet](#) is the class to access the Warner Valve Controller

Inheritance diagram for CWarnerValveControllerDeviceNet:



## Public Member Functions

- delegate void [OnGetValveActive](#) (uint16\_t valve, int valveActive)
- delegate void [OnGetValveManualState](#) (uint16\_t valve, int32\_t valveManualState)
- delegate void [OnGetValveManualGroup](#) (uint16\_t valve, int32\_t valveManualGroup)
- delegate void [OnGetValveMode](#) (uint16\_t valve, [WvcValveModeEnumNet](#) ValveMode)
- delegate void [OnGetAnalogThresholdLow](#) (uint16\_t valve, int32\_t threshold)
- delegate void [OnGetAnalogThresholdHigh](#) (uint16\_t valve, int32\_t threshold)
- delegate void [OnGetDigitalPortDirection](#) (uint16\_t port, [PortDirectionEnumNet](#) direction)
- delegate void [OnIsValveDigitalInInverted](#) (uint16\_t valve, bool isInverted)
- delegate void [OnGetValveDigitalInPort](#) (uint16\_t valve, uint32\_t digitalInPort)
- delegate void [OnIsDigitalOutPortInverted](#) (uint16\_t digitalOutPort, bool isInverted)
- delegate void [OnGetDigitalOutPortValve](#) (uint16\_t digitalOutPort, uint32\_t valve)
- delegate void [OnIsValveOpen](#) (uint16\_t valve, bool valveOpen)
- delegate void [OnIsValveOpenInDigitalMode](#) (uint16\_t valve, bool valveOpen)
- delegate void [OnIsValveOpenInAnalogMode](#) (uint16\_t valve, bool valveOpen)
- delegate void [OnGetAnalogVoltage](#) (int32\_t voltage)
- delegate void [OnTableEntryChanged](#) (uint16\_t tableNumber)
- delegate void [OnGetTableNamebyIndex](#) (uint16\_t tableNumber, String^ tableName)
- delegate void [OnGetActiveRunningTableNumber](#) (uint32\_t tableNumber)
- delegate void [OnGetCurrentNumberOfValves](#) (int32\_t numberOfValves)
- delegate void [OnGetValveBoardRevision](#) (uint32\_t revision)
- delegate void [OnGetValveLedOn](#) (bool ledon)
- delegate void [OnGetDisplayMode](#) ([WvcDisplayModeEnumNet](#) DisplayMode)
- [CWarnerValveControllerDeviceNet](#) ()  
*Initializes a new instance of the [CWarnerValveControllerDeviceNet](#) class.*
- virtual [~CWarnerValveControllerDeviceNet](#) ()
- [ICWarnerValveControllerDeviceNet](#) ()
- int [GetValveActive](#) (uint16\_t valve)  
*Gets the valve active/inactive state*
- void [SetValveActive](#) (uint16\_t valve, int valveActive)  
*Sets the valve active/inactive state*
- uint32\_t [GetValvesActiveMap](#) ()  
*Gets the valves active/inactive states*
- void [SetValvesActiveMap](#) (uint32\_t valvesActive)  
*Sets the valve active/inactive state*
- int32\_t [GetValveManualState](#) (uint16\_t valve)  
*Gets the valve manual on/off state*
- void [SetValveManualState](#) (uint16\_t valve, int32\_t valveManualState)  
*Sets the valve manual on/off state*
- uint32\_t [GetValvesManualStateMap](#) ()  
*Gets the valves manual on/off states*
- void [SetValvesManualStateMap](#) (uint32\_t valveaManualState)  
*Sets the valve manual on/off state*
- int32\_t [GetValveManualGroup](#) (uint16\_t valve)  
*Gets the valve manual group*
- void [SetValveManualGroup](#) (uint16\_t valve, int32\_t valveManualGroup)  
*Sets the valve manual group*
- [WvcValveModeEnumNet](#) [GetValveMode](#) (uint16\_t valve)  
*Reads the valve mode*
- void [SetValveMode](#) (uint16\_t valve, [WvcValveModeEnumNet](#) ValveMode)  
*Writes the valve mode*
- int32\_t [GetAnalogThresholdLow](#) (uint16\_t valve)

- Gets the lower threshold for the analog in port per valve*
  - void [SetAnalogThresholdLow](#) (uint16\_t valve, int32\_t threshold)
- Sets the lower threshold for the analog in port per valve*
  - int32\_t [GetAnalogThresholdHigh](#) (uint16\_t valve)
- Gets the upper threshold for the analog in port per valve*
  - void [SetAnalogThresholdHigh](#) (uint16\_t valve, int32\_t threshold)
- Sets the upper threshold for the analog in port per valve*
  - [PortDirectionEnumNet](#) [GetDigitalPortDirection](#) (uint16\_t port)
- Gets the direction of a digital port*
  - void [SetDigitalPortDirection](#) (uint16\_t port, [PortDirectionEnumNet](#) direction)
- Sets the direction of a digital port*
  - bool [IsValveDigitalInInverted](#) (uint16\_t valve)
- Is digital in inverted*
  - void [SetValveDigitalInInvert](#) (uint16\_t valve, bool isInverted)
- Invert digital in*
  - uint32\_t [GetValveDigitalInPort](#) (uint16\_t valve)
- Gets the number of the digital in port which is mapped to a valve*
  - void [SetValveDigitalInPort](#) (uint16\_t valve, uint32\_t digitalInPort)
- Map a digital in port to a valve*
  - bool [IsDigitalOutPortInverted](#) (uint16\_t digitalOutPort)
- Gets the number of the valve which is mapped to a digital out port*
  - void [SetDigitalOutPortInvert](#) (uint16\_t digitalOutPort, bool isInverted)
- Map a valve to a digital out port*
  - uint32\_t [GetDigitalOutPortValve](#) (uint16\_t digitalOutPort)
- Gets the number of the valve which is mapped to a digital out port*
  - void [SetDigitalOutPortValve](#) (uint16\_t digitalOutPort, uint32\_t valve)
- Map a valve to a digital out port*
  - void [SetDefault](#) ()
- Sets the settings of the valve controller to default*
  - bool [IsValveOpen](#) (uint16\_t valve)
- Is valve open*
  - bool [IsValveOpenInDigitalMode](#) (uint16\_t valve)
- True, if the valve would be open when the device is in digital mode*
  - bool [IsValveOpenInAnalogMode](#) (uint16\_t valve)
- True, if the valve would be open when the device is in analog mode*
  - int32\_t [GetAnalogVoltage](#) ()
- Reads the voltage on the analog in port*
  - void [GetValveTableEntry](#) (uint16\_t valve, uint16\_t index, [System::Runtime::InteropServices::Out]uint32\_t% duration, [System::Runtime::InteropServices::Out]bool% state)
- Read an entry from the valve protocol table*
  - void [SetValveTableEntry](#) (uint16\_t valve, uint16\_t index, uint32\_t duration, bool state)
- Write an entry to the valve protocol table*
  - void [ClearValveTable](#) (uint16\_t valve)
- Clear the valve protocol table*
  - void [LoadValveTable](#) ()
- Load the current table from permanent memory*
  - void [StoreValveTable](#) ()
- Store the current table in permanent memory*
  - String ^ [GetTableNamebyIndex](#) (uint16\_t tableNumber)
- Get the name of a protocol table*
  - String ^ [GetTableName](#) ()

- Get the name of the current protocol table*

  - void [SetTableName](#) (String^ tableName)

*Set the name of the current protocol table*
- uint32\_t [GetActiveRunningTableNumber](#) ()

*Gets the number of the table that is active for running*
- void [SetActiveRunningTableNumber](#) (uint32\_t tableNumber)

*Sets the number of the table that is active for running*
- uint32\_t [GetCurrentEditTableNumber](#) ()

*Gets the number of the table that is current for editing*
- void [SetCurrentEditTableNumber](#) (uint32\_t tableNumber)

*Sets the number of the table that is current for editing*
- void [ClearTableName](#) ()

*Clear the name of current protocol table*
- void [SetTableStep](#) (uint16\_t valve, int32\_t steps)

*Skips the table protocol for a valve by steps*
- void [SetTableStepAll](#) (int32\_t steps)

*Skips the table protocol for all valves by steps*
- int32\_t [GetTotalNumberOfValves](#) ()

*Get the total number of valves in the system*
- int32\_t [GetTotalNumberOfDigitalPorts](#) ()

*Get the total number of digital ports in the system*
- int32\_t [GetTotalTableSize](#) ()

*Get the total table size in the system*
- int32\_t [GetTotalNumberOfTables](#) ()

*Get the total number of tables in the system*
- int32\_t [GetCurrentNumberOfValves](#) ()

*Get the current number of valves connected to the system*
- uint32\_t [GetValveBoardRevision](#) ()

*Gets the revision code of the valve board*
- bool [GetValveLedOn](#) ()

*Gets the LED state of the valve board*
- void [SetValveLedOn](#) (bool ledon)

*Sets 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](#)<sup>^</sup> [GetValveActiveEvent](#) [add, remove, raise]  
*Event fires when the valve state for the valve number has changed*
- [OnGetValveManualState](#)<sup>^</sup> [GetValveManualStateEvent](#) [add, remove, raise]  
*Event fires when the manual valve state for the valve number has changed*
- [OnGetValveManualGroup](#)<sup>^</sup> [GetValveManualGroupEvent](#) [add, remove, raise]  
*Event fires when the manual valve group for the valve number has changed*
- [OnGetValveMode](#)<sup>^</sup> [GetValveModeEvent](#) [add, remove, raise]  
*Event fires when the valve mode for the valve number has changed*
- [OnGetAnalogThresholdLow](#)<sup>^</sup> [GetAnalogThresholdLowEvent](#) [add, remove, raise]  
*Event fires when the threshold in mV for the valve number has changed*
- [OnGetAnalogThresholdHigh](#)<sup>^</sup> [GetAnalogThresholdHighEvent](#) [add, remove, raise]  
*Event fires when the threshold in mV for the valve number has changed*
- [OnGetDigitalPortDirection](#)<sup>^</sup> [GetDigitalPortDirectionEvent](#) [add, remove, raise]  
*Event fires when the direction for the port number has changed*
- [OnIsValveDigitalInInverted](#)<sup>^</sup> [IsValveDigitalInInvertedEvent](#) [add, remove, raise]  
*Event fires when is inverted for the valve number has changed*
- [OnGetValveDigitalInPort](#)<sup>^</sup> [GetValveDigitalInPortEvent](#) [add, remove, raise]  
*Event fires when the digital in port for the valve number has changed*
- [OnIsDigitalOutPortInverted](#)<sup>^</sup> [IsDigitalOutPortInvertedEvent](#) [add, remove, raise]  
*Event fires when is inverted for the digital out port has changed*
- [OnGetDigitalOutPortValve](#)<sup>^</sup> [GetDigitalOutPortValveEvent](#) [add, remove, raise]  
*Event fires when the valve number for the digital out port has changed*
- [OnIsValveOpen](#)<sup>^</sup> [IsValveOpenEvent](#) [add, remove, raise]  
*Event fires when is open for the valve number has changed*
- [OnIsValveOpenInDigitalMode](#)<sup>^</sup> [IsValveOpenInDigitalModeEvent](#) [add, remove, raise]  
*Event fires when is open for the valve number has changed*
- [OnIsValveOpenInAnalogMode](#)<sup>^</sup> [IsValveOpenInAnalogModeEvent](#) [add, remove, raise]  
*Event fires when is open for the valve number has changed*
- [OnGetAnalogVoltage](#)<sup>^</sup> [GetAnalogVoltageEvent](#) [add, remove, raise]  
*Event fires when the voltage in mV has changed*
- [OnTableEntryChanged](#)<sup>^</sup> [TableEntryChangedEvent](#) [add, remove, raise]  
*Event fires when an entry of a table changed*
- [OnGetTableNamebyIndex](#)<sup>^</sup> [GetTableNamebyIndexEvent](#) [add, remove, raise]  
*Event fires when the name of the table for the table number has changed*
- [OnGetActiveRunningTableNumber](#)<sup>^</sup> [GetActiveRunningTableNumberEvent](#) [add, remove, raise]  
*Event fires when the table number has changed*
- [OnGetCurrentNumberOfValves](#)<sup>^</sup> [GetCurrentNumberOfValvesEvent](#) [add, remove, raise]  
*Event fires when the number of valves has changed*
- [OnGetValveBoardRevision](#)<sup>^</sup> [GetValveBoardRevisionEvent](#) [add, remove, raise]  
*Event fires when the revision code has changed*
- [OnGetValveLedOn](#)<sup>^</sup> [GetValveLedOnEvent](#) [add, remove, raise]  
*Event fires when the LED state has changed*
- [OnGetDisplayMode](#)<sup>^</sup> [GetDisplayModeEvent](#) [add, remove, raise]  
*Event fires when the display mode has changed*

## Additional Inherited Members

### 11.126.1 Detailed Description

[CWarnerValveControllerDeviceNet](#) is the class to access the Warner Valve Controller



## 11.126.2 Constructor & Destructor Documentation

### 11.126.2.1 CWarnerValveControllerDeviceNet() CWarnerValveControllerDeviceNet ( )

Initializes a new instance of the CWarnerValveControllerDeviceNet class.

### 11.126.2.2 ~CWarnerValveControllerDeviceNet() virtual ~CWarnerValveControllerDeviceNet ( ) [virtual]

### 11.126.2.3 !CWarnerValveControllerDeviceNet() !CWarnerValveControllerDeviceNet ( )

## 11.126.3 Member Function Documentation

### 11.126.3.1 ClearTableName() void ClearTableName ( )

Clear the name of current protocol table

### 11.126.3.2 ClearValveTable() void ClearValveTable ( uint16\_t valve )

Clear the valve protocol table

#### Parameters

|              |                  |
|--------------|------------------|
| <i>valve</i> | The valve number |
|--------------|------------------|

### 11.126.3.3 GetActiveRunningTableNumber() uint32\_t GetActiveRunningTableNumber ( )

Gets the number of the table that is active for running

#### Returns

The table number

**11.126.3.4 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

**11.126.3.5 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

**11.126.3.6 GetAnalogVoltage()** `int32_t GetAnalogVoltage ( )`

Reads the voltage on the analog in port

**Returns**

The voltage in mV

**11.126.3.7 GetCurrentEditTableNumber()** `uint32_t GetCurrentEditTableNumber ( )`

Gets the number of the table that is current for editing

**Returns**

The table number

**11.126.3.8 GetCurrentNumberOfValves()** `int32_t GetCurrentNumberOfValves ( )`

Get the current number of valves connected to the system

**Returns**

The number of valves

**11.126.3.9 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

**11.126.3.10 GetDigitalPortDirection()** `PortDirectionEnumNet GetDigitalPortDirection ( uint16_t port )`

Gets the direction of a digital port

**Parameters**

|             |                 |
|-------------|-----------------|
| <i>port</i> | The port number |
|-------------|-----------------|

**Returns**

the direction

**11.126.3.11 GetDisplayMode()** `WvcDisplayModeEnumNet GetDisplayMode ( )`

Reads the display mode

**Returns**

The display mode

**11.126.3.12 GetTableName()** `String ^ GetTableName ( )`

Get the name of the current protocol table

**Returns**

The name of the table

**11.126.3.13 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

**11.126.3.14 GetTotalNumberOfDigitalPorts()** `int32_t GetTotalNumberOfDigitalPorts ( )`

Get the total number of digital ports in the system

**Returns**

The number of digital ports

**11.126.3.15 GetTotalNumberOfTables()** `int32_t GetTotalNumberOfTables ( )`

Get the total number of tables in the system

**Returns**

The number of tables

**11.126.3.16 GetTotalNumberOfValves()** `int32_t GetTotalNumberOfValves ( )`

Get the total number of valves in the system

**Returns**

The number of valves

**11.126.3.17 GetTotalTableSize()** `int32_t GetTotalTableSize ( )`

Get the total table size in the system

**Returns**

The table size

**11.126.3.18 GetValveActive()** `int GetValveActive (   
 uint16_t valve )`

Gets the valve active/inactive state

**Parameters**

|              |                  |
|--------------|------------------|
| <i>valve</i> | The valve number |
|--------------|------------------|

**Returns**

The valve state

**11.126.3.19 GetValveBoardRevision()** `uint32_t GetValveBoardRevision ( )`

Gets the revision code of the valve board

**Returns**

The revision code

**11.126.3.20 GetValveBoardRevisionString()** `String ^ GetValveBoardRevisionString ( )`

Gets the revision name of the valve board

**Returns**

The revision name

**11.126.3.21 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)   |

**11.126.3.22 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

**11.126.3.23 GetValveLedOn()** `bool GetValveLedOn ( )`

Gets the LED state of the valve board

**Returns**

The LED state

**11.126.3.24 GetValveManualGroup()** `int32_t GetValveManualGroup (`  
    `uint16_t valve )`

Gets the valve manual group

**Parameters**

|              |                  |
|--------------|------------------|
| <i>valve</i> | The valve number |
|--------------|------------------|

**Returns**

The manual valve group

**11.126.3.25 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

**11.126.3.26 GetValveMode()** `WvcValveModeEnumNet GetValveMode ( uint16_t valve )`

Reads the valve mode

**Parameters**

|              |                  |
|--------------|------------------|
| <i>valve</i> | The valve number |
|--------------|------------------|

**Returns**

The valve mode

**11.126.3.27 GetValvesActiveMap()** `uint32_t GetValvesActiveMap ( )`

Gets the valves active/inactive states

**Returns**

The valves states

**11.126.3.28 GetValvesManualStateMap()** `uint32_t GetValvesManualStateMap ( )`

Gets the valves manual on/off states

**Returns**

The manual valves states

**11.126.3.29 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              |

**11.126.3.30 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

**11.126.3.31 IsValveDigitalInInverted()** `bool IsValveDigitalInInverted (   
 uint16_t valve )`

Is digital in inverted

**Parameters**

|              |                  |
|--------------|------------------|
| <i>valve</i> | The valve number |
|--------------|------------------|



**Returns**

is inverted

**11.126.3.32 IsValveOpen()** `bool IsValveOpen (`  
    `uint16_t valve )`

Is valve open

**Parameters**

|              |                  |
|--------------|------------------|
| <i>valve</i> | The valve number |
|--------------|------------------|

**Returns**

is open

**11.126.3.33 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

**11.126.3.34 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

**11.126.3.35 LoadValveTable()** `void LoadValveTable ( )`

Load the current table from permanent memory

**11.126.3.36 OnGetActiveRunningTableNumber()** `delegate void OnGetActiveRunningTableNumber (   
uint32_t tableNumber )`

**11.126.3.37 OnGetAnalogThresholdHigh()** `delegate void OnGetAnalogThresholdHigh (   
uint16_t valve,   
int32_t threshold )`

**11.126.3.38 OnGetAnalogThresholdLow()** `delegate void OnGetAnalogThresholdLow (   
uint16_t valve,   
int32_t threshold )`

**11.126.3.39 OnGetAnalogVoltage()** `delegate void OnGetAnalogVoltage (   
int32_t voltage )`

**11.126.3.40 OnGetCurrentNumberOfValves()** `delegate void OnGetCurrentNumberOfValves (   
int32_t numberOfValves )`

**11.126.3.41 OnGetDigitalOutPortValve()** `delegate void OnGetDigitalOutPortValve (   
uint16_t digitalOutPort,   
uint32_t valve )`

**11.126.3.42 OnGetDigitalPortDirection()** `delegate void OnGetDigitalPortDirection (   
uint16_t port,   
PortDirectionEnumNet direction )`

**11.126.3.43 OnGetDisplayMode()** `delegate void OnGetDisplayMode (   
WvcDisplayModeEnumNet DisplayMode )`

- 11.126.3.44 OnGetTableNamebyIndex()** delegate void OnGetTableNamebyIndex (   
     uint16\_t *tableNumber*,   
     String^ *tableName* )
- 11.126.3.45 OnGetValveActive()** delegate void OnGetValveActive (   
     uint16\_t *valve*,   
     int *valveActive* )
- 11.126.3.46 OnGetValveBoardRevision()** delegate void OnGetValveBoardRevision (   
     uint32\_t *revision* )
- 11.126.3.47 OnGetValveDigitalInPort()** delegate void OnGetValveDigitalInPort (   
     uint16\_t *valve*,   
     uint32\_t *digitalInPort* )
- 11.126.3.48 OnGetValveLedOn()** delegate void OnGetValveLedOn (   
     bool *ledon* )
- 11.126.3.49 OnGetValveManualGroup()** delegate void OnGetValveManualGroup (   
     uint16\_t *valve*,   
     int32\_t *valveManualGroup* )
- 11.126.3.50 OnGetValveManualState()** delegate void OnGetValveManualState (   
     uint16\_t *valve*,   
     int32\_t *valveManualState* )
- 11.126.3.51 OnGetValveMode()** delegate void OnGetValveMode (   
     uint16\_t *valve*,   
     WvcValveModeEnumNet *ValveMode* )

**11.126.3.52 OnIsDigitalOutPortInverted()** `delegate void OnIsDigitalOutPortInverted (`  
    `uint16_t digitalOutPort,`  
    `bool isInverted )`

**11.126.3.53 OnIsValveDigitalInInverted()** `delegate void OnIsValveDigitalInInverted (`  
    `uint16_t valve,`  
    `bool isInverted )`

**11.126.3.54 OnIsValveOpen()** `delegate void OnIsValveOpen (`  
    `uint16_t valve,`  
    `bool valveOpen )`

**11.126.3.55 OnIsValveOpenInAnalogMode()** `delegate void OnIsValveOpenInAnalogMode (`  
    `uint16_t valve,`  
    `bool valveOpen )`

**11.126.3.56 OnIsValveOpenInDigitalMode()** `delegate void OnIsValveOpenInDigitalMode (`  
    `uint16_t valve,`  
    `bool valveOpen )`

**11.126.3.57 OnTableEntryChanged()** `delegate void OnTableEntryChanged (`  
    `uint16_t tableNumber )`

**11.126.3.58 SetActiveRunningTableNumber()** `void SetActiveRunningTableNumber (`  
    `uint32_t tableNumber )`

Sets the number of the tanle that is active for running

#### Parameters

|                    |                  |
|--------------------|------------------|
| <i>tableNumber</i> | The table number |
|--------------------|------------------|

**11.126.3.59 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 |

**11.126.3.60 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 |

**11.126.3.61 SetCurrentEditTableNumber()** `void SetCurrentEditTableNumber (`  
    `uint32_t tableNumber )`

Sets the number of the table that is current for editing

#### Parameters

|                    |                  |
|--------------------|------------------|
| <i>tableNumber</i> | The table number |
|--------------------|------------------|

**11.126.3.62 SetDefault()** `void SetDefault ( )`

Sets the settings of the valve controller to default

**11.126.3.63 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 |

**11.126.3.64 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     |

**11.126.3.65 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   |

**11.126.3.66 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 |

**11.126.3.67 SetTableName()** `void SetTableName (`  
    `String^ tableName )`

Set the name of the current protocol table

Parameters

|                  |                       |
|------------------|-----------------------|
| <i>tableName</i> | The name of the table |
|------------------|-----------------------|

**11.126.3.68 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  |

**11.126.3.69 SetTableStepAll()** void SetTableStepAll (  
    int32\_t steps )

Skips the table protocol for all valves by steps

Parameters

|              |                 |
|--------------|-----------------|
| <i>steps</i> | Number of steps |
|--------------|-----------------|

**11.126.3.70 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  |

**11.126.3.71 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   |

**11.126.3.72 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 |

**11.126.3.73 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 |

**11.126.3.74 SetValveLedOn()** `void SetValveLedOn (`  
    `bool ledon )`

Gets the LED state of the valve board

**Parameters**

|              |               |
|--------------|---------------|
| <i>ledon</i> | The LED state |
|--------------|---------------|

**11.126.3.75 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 |



**11.126.3.76 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 |

**11.126.3.77 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   |

**11.126.3.78 SetValvesActiveMap()** `void SetValvesActiveMap (`  
    `uint32_t valvesActive )`

Sets the valve active/inactive state

Parameters

|                     |                   |
|---------------------|-------------------|
| <i>valvesActive</i> | The valves states |
|---------------------|-------------------|

**11.126.3.79 SetValvesManualStateMap()** `void SetValvesManualStateMap (`  
    `uint32_t valveaManualState )`

Sets the valve manual on/off state

Parameters

|                          |                          |
|--------------------------|--------------------------|
| <i>valveaManualState</i> | The manual valves states |
|--------------------------|--------------------------|

**11.126.3.80 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              |

**11.126.3.81 StoreValveTable()** `void StoreValveTable ( )`

Store the current table in permanent memory

**11.126.4 Event Documentation**

**11.126.4.1 GetActiveRunningTableNumberEvent** `OnGetActiveRunningTableNumber^ GetActiveRunning↔`  
`TableNumberEvent [add], [remove], [raise]`

Event fires when the table number has changed

**11.126.4.2 GetAnalogThresholdHighEvent** `OnGetAnalogThresholdHigh^ GetAnalogThresholdHighEvent`  
`[add], [remove], [raise]`

Event fires when the threshold in mV for the valve number has changed

**11.126.4.3 GetAnalogThresholdLowEvent** `OnGetAnalogThresholdLow^ GetAnalogThresholdLowEvent`  
`[add], [remove], [raise]`

Event fires when the threshold in mV for the valve number has changed

**11.126.4.4 GetAnalogVoltageEvent** `OnGetAnalogVoltage^ GetAnalogVoltageEvent [add], [remove], [raise]`

Event fires when the voltage in mV has changed

**11.126.4.5 GetCurrentNumberOfValvesEvent** `OnGetCurrentNumberOfValves^ GetCurrentNumberOfValvesEvent [add], [remove], [raise]`

Event fires when the number of valves has changed

**11.126.4.6 GetDigitalOutPortValveEvent** `OnGetDigitalOutPortValve^ GetDigitalOutPortValveEvent [add], [remove], [raise]`

Event fires when the valve number for the digital out port has changed

**11.126.4.7 GetDigitalPortDirectionEvent** `OnGetDigitalPortDirection^ GetDigitalPortDirectionEvent [add], [remove], [raise]`

Event fires when the direction for the port number has changed

**11.126.4.8 GetDisplayModeEvent** `OnGetDisplayMode^ GetDisplayModeEvent [add], [remove], [raise]`

Event fires when the display mode has changed

**11.126.4.9 GetTableNamebyIndexEvent** `OnGetTableNamebyIndex^ GetTableNamebyIndexEvent [add], [remove], [raise]`

Event fires when the name of the table for the table number has changed

**11.126.4.10 GetValveActiveEvent** `OnGetValveActive^ GetValveActiveEvent [add], [remove], [raise]`

Event fires when the valve state for the valve number has changed

**11.126.4.11 GetValveBoardRevisionEvent** [OnGetValveBoardRevision](#)<sup>^</sup> GetValveBoardRevisionEvent  
[add], [remove], [raise]

Event fires when the revision code has changed

**11.126.4.12 GetValveDigitalInPortEvent** [OnGetValveDigitalInPort](#)<sup>^</sup> GetValveDigitalInPortEvent [add],  
[remove], [raise]

Event fires when the digital in port for the valve number has changed

**11.126.4.13 GetValveLedOnEvent** [OnGetValveLedOn](#)<sup>^</sup> GetValveLedOnEvent [add], [remove], [raise]

Event fires when the LED state has changed

**11.126.4.14 GetValveManualGroupEvent** [OnGetValveManualGroup](#)<sup>^</sup> GetValveManualGroupEvent [add],  
[remove], [raise]

Event fires when the manual valve group for the valve number has changed

**11.126.4.15 GetValveManualStateEvent** [OnGetValveManualState](#)<sup>^</sup> GetValveManualStateEvent [add],  
[remove], [raise]

Event fires when the manual valve state for the valve number has changed

**11.126.4.16 GetValveModeEvent** [OnGetValveMode](#)<sup>^</sup> GetValveModeEvent [add], [remove], [raise]

Event fires when the valve mode for the valve number has changed

**11.126.4.17 IsDigitalOutPortInvertedEvent** [OnIsDigitalOutPortInverted](#)<sup>^</sup> IsDigitalOutPortInverted↔  
Event [add], [remove], [raise]

Event fires when is inverted for the digital out port has changed

**11.126.4.18 IsValveDigitalInInvertedEvent** [OnIsValveDigitalInInverted](#)<sup>^</sup> IsValveDigitalInInverted↔  
Event [add], [remove], [raise]

Event fires when is inverted for the valve number has changed

**11.126.4.19 IsValveOpenEvent** [OnIsValveOpen](#)<sup>^</sup> IsValveOpenEvent [add], [remove], [raise]

Event fires when is open for the valve number has changed

**11.126.4.20 IsValveOpenInAnalogModeEvent** [OnIsValveOpenInAnalogMode](#)<sup>^</sup> IsValveOpenInAnalogMode↔  
Event [add], [remove], [raise]

Event fires when is open for the valve number has changed

**11.126.4.21 IsValveOpenInDigitalModeEvent** [OnIsValveOpenInDigitalMode](#)<sup>^</sup> IsValveOpenInDigital↔  
ModeEvent [add], [remove], [raise]

Event fires when is open for the valve number has changed

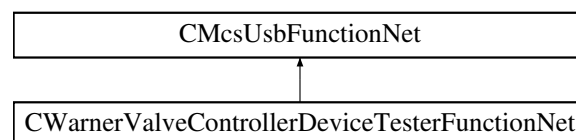
**11.126.4.22 TableEntryChangedEvent** [OnTableEntryChanged](#)<sup>^</sup> TableEntryChangedEvent [add], [remove], [raise]

Event fires when an entry of a table changed

## 11.127 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

### 11.127.1 Detailed Description

[CWarnerValveControllerDeviceTesterFunctionNet](#) is the class to access the functions for the Warner Valve Controller Device Tester

### 11.127.2 Constructor & Destructor Documentation

**11.127.2.1 [CWarnerValveControllerDeviceTesterFunctionNet](#)() [1/2]** [CWarnerValveControllerDeviceTesterFunctionNet](#)  
 (  
     [CMcsUsbNet](#)<sup>^</sup> mcsusb,  
     [CMcsUsbFunctionPointerContainer](#)<sup>^</sup> pWarnerValveControllerDeviceTesterFunctionPointerContainer )

Initializes a new instance of the [CWarnerValveControllerDeviceTesterFunctionNet](#) class.

**11.127.2.2 [CWarnerValveControllerDeviceTesterFunctionNet](#)() [2/2]** [CWarnerValveControllerDeviceTesterFunctionNet](#)  
 (  
     [CMcsUsbNet](#)<sup>^</sup> mcsusb )

**11.127.2.3** `~CWarnerValveControllerDeviceTesterFunctionNet()` `virtual ~CWarnerValveControllerDeviceTesterFunctionNet()` `[virtual]`

**11.127.2.4** `!CWarnerValveControllerDeviceTesterFunctionNet()` `!CWarnerValveControllerDeviceTesterFunctionNet()`

### 11.127.3 Member Function Documentation

**11.127.3.1** `GetIO()` `uint32_t GetIO()`

Gets the output from the io ports

#### Returns

The manual valves states

**11.127.3.2** `GetSync()` `uint32_t GetSync()`

Gets the output from the sync port

#### Returns

The sync state

**11.127.3.3** `SetADC()` `void SetADC (uint32_t onoff)`

Sets the ADC port of the tester

#### Parameters

|              |                |
|--------------|----------------|
| <i>onoff</i> | The port state |
|--------------|----------------|

**11.127.3.4** `SetIO()` `void SetIO (uint32_t io)`

Sets the input to the io ports

**Parameters**

|           |                          |
|-----------|--------------------------|
| <i>io</i> | The manual valves states |
|-----------|--------------------------|

**11.127.3.5 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 |
|------------------|-------------------------------------|

**11.127.3.6 SetTrigger()** `void SetTrigger (`  
`uint32_t trigger )`

Sets the input to the trigger port

**Parameters**

|                |                   |
|----------------|-------------------|
| <i>trigger</i> | The trigger state |
|----------------|-------------------|

**11.127.3.7 SetTriggerSyncDirection()** `void SetTriggerSyncDirection (`  
`uint32_t direction )`

Sets the direction of the trigger/sync test port

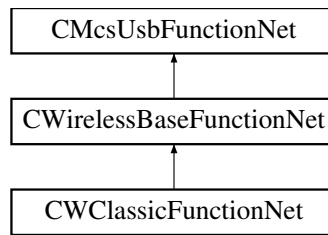
**Parameters**

|                  |                           |
|------------------|---------------------------|
| <i>direction</i> | The direction: 1=IN 0=OUT |
|------------------|---------------------------|

## 11.128 CWClassicFunctionNet Class Reference

Inheritance diagram for CWClassicFunctionNet:





## Public Member Functions

- [CWClassicFunctionNet](#) ([CMcsUsbNet](#)^ mcsusb, [CMcsUsbFunctionPointerContainer](#)^ wClassicFuntionPointerContainer)
- [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

### 11.128.1 Constructor & Destructor Documentation

**11.128.1.1 CWClassicFunctionNet()** [1/2] `CWClassicFunctionNet (`  
    `CMcsUsbNet^ mcsusb,`  
    `CMcsUsbFunctionPointerContainer^ wClassicFuntionPointerContainer )`

**11.128.1.2 CWClassicFunctionNet()** [2/2] `CWClassicFunctionNet (`  
    `CMcsUsbNet^ mcsusb )`

## 11.128.2 Member Function Documentation

**11.128.2.1 GetFilterParametersHeadstage()** `array<int> ^ GetFilterParametersHeadstage (`  
    `unsigned short index )`

**11.128.2.2 GetHasChecksum()** `unsigned int GetHasChecksum (`  
    `unsigned int Device )`

**11.128.2.3 GetHasRedLedHeadstage()** `bool GetHasRedLedHeadstage ( )`

**11.128.2.4 GetHeadstageOnOff()** `USHORT GetHeadstageOnOff ( )`

**11.128.2.5 GetResetFilter()** `unsigned int GetResetFilter (`  
    `unsigned int Device )`

**11.128.2.6 GetRFConnectionStatus()** `unsigned int GetRFConnectionStatus ( )`

**11.128.2.7 GetRFFrequencyHeadstage()** `unsigned short GetRFFrequencyHeadstage (`  
    `uint8_t receiver_nb )`

**11.128.2.8 GetRFFrequencyReceiver()** unsigned short GetRFFrequencyReceiver (   
     uint8\_t *receiver\_nb*,   
     uint8\_t *configuration* )

**11.128.2.9 GetRFPower()** unsigned short GetRFPower ( )

**11.128.2.10 GetScanHeadstagesResult()** uint8\_t GetScanHeadstagesResult (   
     int *max\_wait\_for\_ms* )

**11.128.2.11 GetSelectedHeadstage()** uint8\_t GetSelectedHeadstage ( )

**11.128.2.12 GetSerialNumberHeadstage()** unsigned short GetSerialNumberHeadstage ( )

**11.128.2.13 GetWPADebugMode()** unsigned int GetWPADebugMode (   
     unsigned int *Device* )

**11.128.2.14 GetWPAType()** unsigned short GetWPAType (   
     unsigned int *Device* )

**11.128.2.15 ResetChannelmap()** uint32\_t ResetChannelmap (   
     unsigned int *virtualDevice* )

**11.128.2.16 ScanForHeadstages()** void ScanForHeadstages ( )

**11.128.2.17 SetChannelmap()** uint32\_t SetChannelmap (   
     unsigned char *position*,   
     unsigned char *channel*,   
     unsigned int *Device* )

**11.128.2.18 SetFilterParametersHeadstage()** void SetFilterParametersHeadstage (   
    unsigned short *index*,  
    array< int >^ *buffer* )

**11.128.2.19 SetHasChecksum()** void SetHasChecksum (   
    unsigned int *has*,  
    unsigned int *Device* )

**11.128.2.20 SetHeadstageOnOff()** void SetHeadstageOnOff (   
    uint16\_t *onoff* )

**11.128.2.21 SetHWSelectedChannels()** void SetHWSelectedChannels (   
    array< bool >^ *channels*,  
    unsigned int *Device* )

**11.128.2.22 SetResetFilter()** void SetResetFilter (   
    unsigned int *reset*,  
    unsigned int *Device* )

**11.128.2.23 SetRFFrequencyHeadstage()** void SetRFFrequencyHeadstage (   
    uint8\_t *receiver\_nb*,  
    unsigned short *frequency* )

**11.128.2.24 SetRFFrequencyReceiver()** void SetRFFrequencyReceiver (   
    uint8\_t *receiver\_nb*,  
    uint8\_t *configuration*,  
    unsigned short *frequency* )

**11.128.2.25 SetRFFrequencyReceiverEeprom()** void SetRFFrequencyReceiverEeprom (   
    uint8\_t *receiver\_nb*,  
    uint8\_t *configuration*,  
    unsigned short *frequency* )

**11.128.2.26 SetRFLostBehaviour()** void SetRFLostBehaviour (   
     uint8\_t *stoponfailure*,   
     unsigned int *Device* )

**11.128.2.27 SetRFPower()** void SetRFPower (   
     unsigned short *power* )

**11.128.2.28 SetSelectedHeadstage()** void SetSelectedHeadstage (   
     uint8\_t *number* )

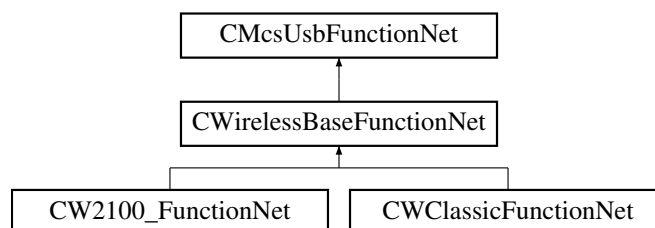
**11.128.2.29 SetSerialNumberHeadstage()** void SetSerialNumberHeadstage (   
     unsigned short *number* )

**11.128.2.30 SetWPADebugMode()** void SetWPADebugMode (   
     unsigned int *mode*,   
     unsigned int *Device* )

**11.128.2.31 SetWPAType()** void SetWPAType (   
     unsigned short *type*,   
     unsigned int *Device* )

## 11.129 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

#### 11.129.1 Constructor & Destructor Documentation

**11.129.1.1 CWirelessBaseFunctionNet()** [CWirelessBaseFunctionNet](#) (   
 [CMcsUsbNet](#)^ *mcsusb*,  
 [CMcsUsbFunctionPointerContainer](#)^ *mcsusbfunction* )

#### 11.129.2 Member Function Documentation

**11.129.2.1 CreateWirelessHeadstageSerialNumberString()** static String ^ [CreateWirelessHeadstageSerialNumberString](#) (   
 unsigned short *ID* ) [static]

### 11.130 DeviceIdNet Struct Reference

Device Id.

#### Public Member Functions

- [DeviceIdNet](#) ()
- [DeviceIdNet](#) ([VendorIdEnumNet](#) vendor, [ProductIdEnumNet](#) product, int bcd, [McsBusTypeEnumNet](#) bustype)
- [DeviceIdNet](#) ([DeviceIdNet](#)% deviceId)
- [DeviceIdNet](#) operator= ([DeviceIdNet](#)% deviceId)

#### Public Attributes

- [VendorIdEnumNet](#) IdVendor
- [ProductIdEnumNet](#) IdProduct
- int [BcdDevice](#)
- [McsBusTypeEnumNet](#) BusType

#### 11.130.1 Detailed Description

Device Id.

## 11.130.2 Constructor & Destructor Documentation

**11.130.2.1 DeviceIdNet()** [1/3] `DeviceIdNet ( )`

**11.130.2.2 DeviceIdNet()** [2/3] `DeviceIdNet (`  
    `VendorIdEnumNet vendor,`  
    `ProductIdEnumNet product,`  
    `int bcd,`  
    `McsBusTypeEnumNet bustype )`

**11.130.2.3 DeviceIdNet()** [3/3] `DeviceIdNet (`  
    `DeviceIdNet% deviceId )`

## 11.130.3 Member Function Documentation

**11.130.3.1 operator=()** `DeviceIdNet operator= (`  
    `DeviceIdNet% deviceId )`

## 11.130.4 Member Data Documentation

**11.130.4.1 BcdDevice** `int BcdDevice`

**11.130.4.2 BusType** `McsBusTypeEnumNet BusType`

**11.130.4.3 IdProduct** `ProductIdEnumNet IdProduct`

**11.130.4.4 IdVendor** `VendorIdEnumNet IdVendor`

## 11.131 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]

### 11.131.1 Constructor & Destructor Documentation

**11.131.1.1 [DigitalSource](#)()** [1/2] [DigitalSource](#) ( )

**11.131.1.2 [DigitalSource](#)()** [2/2] [DigitalSource](#) (  
digitalsourceenum *source* )

### 11.131.2 Member Function Documentation

**11.131.2.1 [MaxBitNumber](#)()** [1/2] int [MaxBitNumber](#) ( )

**11.131.2.2 [MaxBitNumber](#)()** [2/2] int [MaxBitNumber](#) (  
digitalsourceenum *Source* )

**11.131.2.3 [MaxBitNumberStatic](#)()** static int [MaxBitNumberStatic](#) (  
digitalsourceenum *Source* ) [static]



**11.131.2.4 size()** `static int size ( ) [static]`

### 11.131.3 Property Documentation

**11.131.3.1 Source** `digitalsourceenum Source [get], [set]`

## 11.132 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]

### 11.132.1 Constructor & Destructor Documentation

**11.132.1.1 DigitalSourceGeneral()** [1/2] [DigitalSourceGeneral](#) (  
Type^ type )

**11.132.1.2 DigitalSourceGeneral()** [2/2] [DigitalSourceGeneral](#) (  
Type^ type,  
int Source )

### 11.132.2 Member Function Documentation

**11.132.2.1 MaxBitNumber()** [1/3] `int MaxBitNumber ( )`

**11.132.2.2 MaxBitNumber()** [2/3] `int MaxBitNumber (`  
`int Source )`

**11.132.2.3 MaxBitNumber()** [3/3] `static int MaxBitNumber (`  
`Type^ type,`  
`int Source ) [static]`

**11.132.2.4 size()** `static int size (`  
`Type^ type ) [static]`

### 11.132.3 Property Documentation

**11.132.3.1 Source** `int Source [get], [set]`

## 11.133 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)

#### 11.133.1 Detailed Description

Class gives firmware versions of the device's firmware destinations.

#### 11.133.2 Constructor & Destructor Documentation

##### 11.133.2.1 [DriverVersionNet\(\)](#) [DriverVersionNet](#) ( )

Constructor.

##### 11.133.2.2 [~DriverVersionNet\(\)](#) [~DriverVersionNet](#) ( )

Destructor.

#### 11.133.3 Member Function Documentation

**11.133.3.1 DriverVersionNet::FormatVersion()** `static String ^ DriverVersionNet::FormatVersion ( unsigned int v ) [static]`

**11.133.3.2 GetDestinationCode()** `CFirmwareDestinationNet GetDestinationCode ( unsigned int index )`

Get CFirmwareDestinationNet.

## Parameters

|              |                                  |
|--------------|----------------------------------|
| <i>index</i> | by index of firmware destination |
|--------------|----------------------------------|

**11.133.3.3 GetDestinationName()** [1/2] `String ^ GetDestinationName ( CFirmwareDestinationNet dest )`

Get firmware destination name.

## Parameters

|             |                            |
|-------------|----------------------------|
| <i>dest</i> | by CFirmwareDestinationNet |
|-------------|----------------------------|

**11.133.3.4 GetDestinationName()** [2/2] `String ^ GetDestinationName ( unsigned int index )`

Get firmware destination name.

## Parameters

|              |                                  |
|--------------|----------------------------------|
| <i>index</i> | by index of firmware destination |
|--------------|----------------------------------|

**11.133.3.5 GetMajor()** [1/2] `unsigned int GetMajor ( CFirmwareDestinationNet dest )`

Get the major version number of firmware destination.

## Parameters

|             |                            |
|-------------|----------------------------|
| <i>dest</i> | by CFirmwareDestinationNet |
|-------------|----------------------------|

**11.133.3.6 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 |
|--------------|----------------------------------|

**11.133.3.7 GetMinor()** [1/2] unsigned int GetMinor (   
CFirmwareDestinationNet dest )

Get the minor version number of firmware destination.

Parameters

|             |                            |
|-------------|----------------------------|
| <i>dest</i> | by CFirmwareDestinationNet |
|-------------|----------------------------|

**11.133.3.8 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 |
|--------------|----------------------------------|

**11.133.3.9 GetNumEntries()** unsigned int GetNumEntries ( )

Get the number of available firmware destinations.

**11.133.3.10 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 |
|-------------|----------------------------|

**11.133.3.11 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 |
|--------------|----------------------------------|

**11.133.3.12 GetStatus()** [1/2] unsigned int GetStatus (   
CFirmwareDestinationNet *dest* )

Get status of firmware destination.

## Parameters

|             |                            |
|-------------|----------------------------|
| <i>dest</i> | by CFirmwareDestinationNet |
|-------------|----------------------------|

**11.133.3.13 GetStatus()** [2/2] unsigned int GetStatus (   
unsigned int *index* )

Get status of firmware destination.

## Parameters

|              |                                  |
|--------------|----------------------------------|
| <i>index</i> | by index of firmware destination |
|--------------|----------------------------------|

**11.133.3.14 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 |
|-------------|----------------------------|

**11.133.3.15 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 |
|--------------|----------------------------------|

**11.133.3.16 GetVersionString() [1/2]** `String ^ GetVersionString (`  
`CFirmwareDestinationNet dest )`

Get the version as a string in the format Major.Minor.

#### Parameters

|             |                            |
|-------------|----------------------------|
| <i>dest</i> | by CFirmwareDestinationNet |
|-------------|----------------------------|

**11.133.3.17 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 |
|--------------|----------------------|

## 11.134 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" )

#### 11.134.1 Member Data Documentation

**11.134.1.1 Altera** String ^ Altera = gcnew String( "Altera" ) [static]

**11.134.1.2 Bootstrap** String ^ Bootstrap = gcnew String( "Bootstrap" ) [static]

**11.134.1.3 BUS1\_MCSBUS1** String ^ BUS1\_MCSBUS1 = gcnew String( "Bus1McsBus1" ) [static]

**11.134.1.4 BUS1\_MCSBUS2** String ^ BUS1\_MCSBUS2 = gcnew String( "Bus1McsBus2" ) [static]

**11.134.1.5 DSP** String ^ DSP = gcnew String( "DSP" ) [static]

**11.134.1.6 FPGA2** String ^ FPGA2 = gcnew String( "FPGA2" ) [static]

**11.134.1.7 FPGA3** String ^ FPGA3 = gcnew String( "FPGA3" ) [static]

**11.134.1.8 FPGA4** String ^ FPGA4 = gcnew String( "FPGA4" ) [static]

**11.134.1.9 FPGA5** String ^ FPGA5 = gcnew String( "FPGA5" ) [static]

**11.134.1.10 FPGA6** `String ^ FPGA6 = gcnew String( "FPGA6" ) [static]`

**11.134.1.11 MCSBUS1** `String ^ MCSBUS1 = gcnew String( "McsBus1" ) [static]`

**11.134.1.12 MCSBUS10** `String ^ MCSBUS10 = gcnew String( "McsBus10" ) [static]`

**11.134.1.13 MCSBUS11** `String ^ MCSBUS11 = gcnew String( "McsBus11" ) [static]`

**11.134.1.14 MCSBUS12** `String ^ MCSBUS12 = gcnew String( "McsBus12" ) [static]`

**11.134.1.15 MCSBUS13** `String ^ MCSBUS13 = gcnew String( "McsBus13" ) [static]`

**11.134.1.16 MCSBUS2** `String ^ MCSBUS2 = gcnew String( "McsBus2" ) [static]`

**11.134.1.17 MCSBUS3** `String ^ MCSBUS3 = gcnew String( "McsBus3" ) [static]`

**11.134.1.18 MCSBUS4** `String ^ MCSBUS4 = gcnew String( "McsBus4" ) [static]`

**11.134.1.19 MCSBUS5** `String ^ MCSBUS5 = gcnew String( "McsBus5" ) [static]`

**11.134.1.20 MCSBUS6** `String ^ MCSBUS6 = gcnew String( "McsBus6" ) [static]`

**11.134.1.21 MCSBUS7** `String ^ MCSBUS7 = gcnew String( "McsBus7" ) [static]`

**11.134.1.22 MCSBUS8** `String ^ MCSBUS8 = gcnew String( "McsBus8" ) [static]`

**11.134.1.23 MCSBUS9** `String ^ MCSBUS9 = gcnew String( "McsBus9" ) [static]`

**11.134.1.24 MCU1** `String ^ MCU1 = gcnew String( "MCU1" ) [static]`

**11.134.1.25 PIC** `String ^ PIC = gcnew String( "PIC" ) [static]`

**11.134.1.26 PIC2** `String ^ PIC2 = gcnew String( "PIC2" ) [static]`

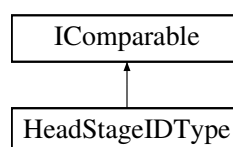
**11.134.1.27 PIC3** `String ^ PIC3 = gcnew String( "PIC3" ) [static]`

**11.134.1.28 PIC4** `String ^ PIC4 = gcnew String( "PIC4" ) [static]`

**11.134.1.29 USB** `String ^ USB = gcnew String( "USB" ) [static]`

## 11.135 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]

## 11.135.1 Member Enumeration Documentation

### 11.135.1.1 HeadstageTypeEnum enum [HeadstageTypeEnum](#) [strong]

#### Enumerator

|                       |  |
|-----------------------|--|
| Unknown               |  |
| MeasuringOnly         |  |
| OpticalStimulation    |  |
| ElectricalStimulation |  |

## 11.135.2 Constructor & Destructor Documentation

**11.135.2.1 HeadStageIDType()** [HeadStageIDType](#) (  
    unsigned int *entry*,  
    [CW2100\\_FunctionNet](#)^ *device* )

### 11.135.3 Member Function Documentation

**11.135.3.1 CompareTo()** virtual Int32 CompareTo (  
    Object^ *obj* ) [virtual]

**11.135.3.2 Equals()** virtual bool Equals (  
    Object^ *obj* ) [override], [virtual]

**11.135.3.3 ToString()** virtual System::String ^ ToString ( ) [override], [virtual]

### 11.135.4 Property Documentation

**11.135.4.1 Entry** unsigned int Entry [get]

**11.135.4.2 HasIMU** bool HasIMU [get]

**11.135.4.3 HasOptoCurrentMeasurement** bool HasOptoCurrentMeasurement [get]

**11.135.4.4 HeadstageType** [HeadstageTypeEnum](#) HeadstageType [get]

**11.135.4.5 ID** unsigned short ID [get]

**11.135.4.6 NumberOfAnalogChannels** `int NumberOfAnalogChannels [get]`

**11.135.4.7 NumberOfStimulationChannels** `int NumberOfStimulationChannels [get]`

**11.135.4.8 SN** `System:: String^ SN [get]`

**11.135.4.9 StimulusParameters** `W2100_StimulusParametersNet^ StimulusParameters [get]`

**11.135.4.10 Type** `System:: String^ Type [get]`

**11.135.4.11 TypeValue** `unsigned int TypeValue [get]`

**11.135.4.12 UserDefinedName** `System:: String^ UserDefinedName [get]`

**11.135.4.13 Valid** `bool Valid [get]`

**11.135.4.14 W16IsW14** `bool W16IsW14 [get]`

## 11.136 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]

## 11.136.1 Constructor & Destructor Documentation

**11.136.1.1 HeadstageIDTypeObject()** [HeadstageIDTypeObject](#) (  
    [HeadStageIDType](#) ^ *idType* )

## 11.136.2 Member Function Documentation

**11.136.2.1 Equals()** virtual bool Equals (  
    Object ^ *obj* ) [override], [virtual]

**11.136.2.2 GetHashCode()** virtual int GetHashCode ( ) [override], [virtual]

**11.136.2.3 ToString()** virtual String ^ ToString ( ) [override], [virtual]

## 11.136.3 Member Data Documentation

**11.136.3.1 \_AdditionalText** String ^ [\\_AdditionalText](#)

**11.136.3.2 \_IdType** [HeadStageIDType](#) ^ [\\_IdType](#)

#### 11.136.4 Property Documentation

**11.136.4.1 AdditionalText** `String^ AdditionalText [get], [set]`

**11.136.4.2 IdType** `HeadStageIDType^ IdType [get]`

### 11.137 HeadStageIDTypeState Class Reference

#### Properties

- unsigned int [State](#) [get]
- [HeadStageIDType^](#) [IdType](#) [get]
- bool [ControlState](#) [get]
- bool [DataState](#) [get]

#### 11.137.1 Property Documentation

**11.137.1.1 ControlState** `bool ControlState [get]`

**11.137.1.2 DataState** `bool DataState [get]`

**11.137.1.3 IdType** `HeadStageIDType^ IdType [get]`

**11.137.1.4 State** `unsigned int State [get]`



## 11.138 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*)

### 11.138.1 Member Function Documentation

**11.138.1.1 [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]

**11.138.1.2 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]

**11.138.1.3 mkfilter\_highpass\_coeff()** static double mkfilter\_highpass\_coeff (   
int *SamplesPerSecond*,  
double *Frequency* ) [static]

**11.138.1.4 mkfilter\_highpass\_frequency\_from\_coeff()** static double mkfilter\_highpass\_frequency\_from↵  
from\_coeff (   
int *SamplesPerSecond*,  
double *coeff* ) [static]

**11.138.1.5 mkfilter\_highpass\_frequency\_from\_k()** static double mkfilter\_highpass\_frequency\_from↵  
\_k (   
int *SamplesPerSecond*,  
double *k* ) [static]

**11.138.1.6 mkfilter\_highpass\_k()** static double mkfilter\_highpass\_k (   
int *SamplesPerSecond*,  
double *Frequency* ) [static]

**11.138.1.7 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]

**11.138.1.8 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]

- 11.138.1.9 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]
- 11.138.1.10 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]
- 11.138.1.11 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]
- 11.138.1.12 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]
- 11.138.1.13 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]
- 11.138.1.14 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]

## 11.139 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)

### 11.139.1 Member Function Documentation

**11.139.1.1 FindReferencePhase0()** [1/2] void FindReferencePhase0 (   
 unsigned char *busaddress*,  
 char *axes* )

**11.139.1.2 FindReferencePhase0()** [2/2] void FindReferencePhase0 (   
 unsigned char *busaddress*,  
 char *axes*,  
 int *timeout* )

**11.139.1.3 GetAxisConfig()** unsigned int GetAxisConfig (   
 unsigned char *busaddress*,  
 char *axes* )

**11.139.1.4 GetHWConfig()** unsigned int GetHWConfig ( )

**11.139.1.5 GetHWRevision()** unsigned int GetHWRevision ( )

**11.139.1.6 GetMaxNoPressure()** unsigned int GetMaxNoPressure ( )

**11.139.1.7 GetMaxNoPressureWaitTime()** unsigned int GetMaxNoPressureWaitTime ( )

**11.139.1.8 GetMaxPressureWaitTime()** unsigned int GetMaxPressureWaitTime ( )

**11.139.1.9 GetMinNoPressureWaitTime()** unsigned int GetMinNoPressureWaitTime ( )

**11.139.1.10 GetMinPressure()** unsigned int GetMinPressure ( )

**11.139.1.11 GetMinPressureWaitTime()** unsigned int GetMinPressureWaitTime ( )

**11.139.1.12 GetParameter() [1/2]** void GetParameter (   
    unsigned short *command*,   
    unsigned short *index*,   
    [System::Runtime::InteropServices::Out] unsigned int% *value* )

**11.139.1.13 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* )

**11.139.1.14 GetPhases()** void GetPhases (   
    unsigned char *busaddress*,   
    char *axes*,   
    [System::Runtime::InteropServices::Out] unsigned short% *phase0*,   
    [System::Runtime::InteropServices::Out] unsigned short% *lastphase* )

**11.139.1.15 GetSearchReferenceFastAccel()** unsigned short GetSearchReferenceFastAccel (   
    unsigned char *busaddress*,   
    char *axes* )

**11.139.1.16 GetSearchReferenceFastSpeed()** unsigned short GetSearchReferenceFastSpeed (   
    unsigned char *busaddress*,   
    char *axes* )

**11.139.1.17 GetSearchReferenceFineAccel()** unsigned short GetSearchReferenceFineAccel (   
    unsigned char *busaddress*,   
    char *axes* )

**11.139.1.18 GetSearchReferenceFineSpeed()** unsigned short GetSearchReferenceFineSpeed (   
     unsigned char *busaddress*,  
     char *axes* )

**11.139.1.19 GetSearchReferenceMethod()** unsigned int GetSearchReferenceMethod (   
     unsigned char *busaddress*,  
     char *axes* )

**11.139.1.20 GetSearchReferenceMoveOut()** int GetSearchReferenceMoveOut (   
     unsigned char *busaddress*,  
     char *axes* )

**11.139.1.21 GetSearchReferenceOffsetPos()** int GetSearchReferenceOffsetPos (   
     unsigned char *busaddress*,  
     char *axes* )

**11.139.1.22 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                               |

**11.139.1.23 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                               |

**11.139.1.24 HasRef()** unsigned char HasRef (  
    unsigned char *busaddress*,  
    char *axes* )

**11.139.1.25 SetAxisConfig()** void SetAxisConfig (  
    unsigned char *busaddress*,  
    char *axes*,  
    unsigned int *config* )

**11.139.1.26 SetHWConfig()** void SetHWConfig (  
    unsigned int *config* )

**11.139.1.27 SetHWRevision()** void SetHWRevision (  
    unsigned int *revision* )

**11.139.1.28 SetMaxNoPressure()** void SetMaxNoPressure (  
    unsigned int *pressure* )

**11.139.1.29 SetMaxNoPressureWaitTime()** void SetMaxNoPressureWaitTime (  
    unsigned int *t* )

**11.139.1.30 SetMaxPressureWaitTime()** void SetMaxPressureWaitTime (  
    unsigned int *t* )

**11.139.1.31 SetMinNoPressureWaitTime()** void SetMinNoPressureWaitTime (  
    unsigned int *t* )

**11.139.1.32 SetMinPressure()** void SetMinPressure (  
    unsigned int *pressure* )



**11.139.1.33 SetMinPressureWaitTime()** void SetMinPressureWaitTime (   
 unsigned int *t* )

**11.139.1.34 SetParameter()** [1/2] void SetParameter (   
 unsigned short *command*,   
 unsigned short *index*,   
 unsigned int *value* )

**11.139.1.35 SetParameter()** [2/2] void SetParameter (   
 unsigned short *command*,   
 unsigned short *index*,   
 unsigned int *value1*,   
 unsigned int *value2* )

**11.139.1.36 SetSearchReferenceFastAccel()** void SetSearchReferenceFastAccel (   
 unsigned char *busaddress*,   
 char *axes*,   
 unsigned short *accel* )

**11.139.1.37 SetSearchReferenceFastSpeed()** void SetSearchReferenceFastSpeed (   
 unsigned char *busaddress*,   
 char *axes*,   
 unsigned short *speed* )

**11.139.1.38 SetSearchReferenceFineAccel()** void SetSearchReferenceFineAccel (   
 unsigned char *busaddress*,   
 char *axes*,   
 unsigned short *accel* )

**11.139.1.39 SetSearchReferenceFineSpeed()** void SetSearchReferenceFineSpeed (   
 unsigned char *busaddress*,   
 char *axes*,   
 unsigned short *speed* )

**11.139.1.40 SetSearchReferenceMethod()** `void SetSearchReferenceMethod (`  
    `unsigned char busaddress,`  
    `char axes,`  
    `unsigned int method )`

**11.139.1.41 SetSearchReferenceMoveOut()** `void SetSearchReferenceMoveOut (`  
    `unsigned char busaddress,`  
    `char axes,`  
    `int move )`

**11.139.1.42 SetSearchReferenceOffsetPos()** `void SetSearchReferenceOffsetPos (`  
    `unsigned char busaddress,`  
    `char axes,`  
    `int offsetpos )`

**11.139.1.43 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                         |

**11.139.1.44 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                         |

## 11.140 CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands Class Reference

### Public Member Functions

- void [FindReferencePhase0XY](#) ()
- void [FindReferencePhase0XY](#) (int timeout)

#### 11.140.1 Member Function Documentation

**11.140.1.1 FindReferencePhase0XY()** [1/2] `void FindReferencePhase0XY ( )`

**11.140.1.2 FindReferencePhase0XY()** [2/2] `void FindReferencePhase0XY (   
int timeout )`

## 11.141 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](#)

#### 11.141.1 Constructor & Destructor Documentation

**11.141.1.1 s\_FilterAttributesNet()** `s_FilterAttributesNet (   
s_FilterAttributes attrib )`

#### 11.141.2 Member Function Documentation

**11.141.2.1 ToCpp()** `s_FilterAttributes ToCpp ( )`

### 11.141.3 Member Data Documentation

**11.141.3.1 CommaPositionA** `uint32_t CommaPositionA`

**11.141.3.2 CommaPositionB** `uint32_t CommaPositionB`

**11.141.3.3 PostCommaA** `uint32_t PostCommaA`

**11.141.3.4 PostCommaB** `uint32_t PostCommaB`

**11.141.3.5 PreCommaA** `uint32_t PreCommaA`

**11.141.3.6 PreCommaB** `uint32_t PreCommaB`

## 11.142 CMeaAudioFunctionNet::s\_setaudionet Struct Reference

### Public Attributes

- int [channel](#)
- int [amplification](#)

### 11.142.1 Member Data Documentation

**11.142.1.1 amplification** `int amplification`

11.142.1.2 **channel** `int channel`

## 11.143 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]

### 11.143.1 Constructor & Destructor Documentation

11.143.1.1 **SidebandData()** [SidebandData](#) ( )

11.143.1.2 **~SidebandData()** [~SidebandData](#) ( )

Destructor: called by Dispose()

11.143.1.3 **!SidebandData()** [!SidebandData](#) ( )

Finalizer: called by GC before collecting

### 11.143.2 Property Documentation

11.143.2.1 **Duration** `array< uint64_t >^` [Duration](#) [get]

11.143.2.2 **Sideband** `array< int32_t >^` [Sideband](#) [get]

## 11.144 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](#)

### 11.144.1 Member Function Documentation

**11.144.1.1 FromIntPtr()** static [StgStatusNet](#) ^ FromIntPtr (   
 IntPtr stgstatus ) [static]

**11.144.1.2 FromPtr()** static [StgStatusNet](#) ^ FromPtr (   
 stgstatus\_t \* stgstatus ) [static]

### 11.144.2 Member Data Documentation

**11.144.2.1 ListOfChangedTriggers** array<uint32\_t> ^ ListOfChangedTriggers

**11.144.2.2 TiggerStatus** array<[Stg200xTriggerStatusEnumNet](#)> ^ TiggerStatus

## 11.145 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]

## 11.145.1 Constructor & Destructor Documentation

**11.145.1.1 StimulusDeviceDataAndUnrolledData()** [StimulusDeviceDataAndUnrolledData](#) ( )

**11.145.1.2 ~StimulusDeviceDataAndUnrolledData()** [~StimulusDeviceDataAndUnrolledData](#) ( )

Destructor: called by Dispose()

**11.145.1.3 ~!StimulusDeviceDataAndUnrolledData()** [!StimulusDeviceDataAndUnrolledData](#) ( )

Finalizer: called by GC before collecting

## 11.145.2 Property Documentation

**11.145.2.1 DeviceData** array< uint8\_t >^ DeviceData [get]

**11.145.2.2 DeviceDataLength** int DeviceDataLength [get]

**11.145.2.3 UnrolledAmplitude** array< int32\_t >^ UnrolledAmplitude [get]

**11.145.2.4 UnrolledDuration** array< uint64\_t >^ UnrolledDuration [get]

**11.145.2.5 UnrolledSync** `array< uint32_t>^ UnrolledSync [get]`

## 11.146 usbSetupPacket\_t Class Reference

### Public Attributes

- `uint8_t` [bmRequestType](#)
- `uint8_t` [bRequest](#)
- `uint16_t` [wValue](#)
- `uint16_t` [wIndex](#)
- `uint16_t` [wLength](#)

### 11.146.1 Member Data Documentation

**11.146.1.1 bmRequestType** `uint8_t bmRequestType`

**11.146.1.2 bRequest** `uint8_t bRequest`

**11.146.1.3 wIndex** `uint16_t wIndex`

**11.146.1.4 wLength** `uint16_t wLength`

**11.146.1.5 wValue** `uint16_t wValue`

## 11.147 W2100\_StimulusParametersNet Struct Reference

### Public Attributes

- `int` [DACResolution](#)
- `int` [TimeResolutionInNanoSeconds](#)
- `int` [VoltageRangeInMicroVolt](#)
- `int` [VoltageResolutionInMicroVolt](#)
- `int` [CurrentRangeInNanoAmp](#)
- `int` [CurrentResolutionInNanoAmp](#)



### 11.147.1 Member Data Documentation

**11.147.1.1 CurrentRangeInNanoAmp** `int CurrentRangeInNanoAmp`

**11.147.1.2 CurrentResolutionInNanoAmp** `int CurrentResolutionInNanoAmp`

**11.147.1.3 DACResolution** `int DACResolution`

**11.147.1.4 TimeResolutionInNanoSeconds** `int TimeResolutionInNanoSeconds`

**11.147.1.5 VoltageRangeInMicroVolt** `int VoltageRangeInMicroVolt`

**11.147.1.6 VoltageResolutionInMicroVolt** `int VoltageResolutionInMicroVolt`



## Index

- !CDacCalibrationFunctionNet
  - CDacCalibrationFunctionNet, [115](#)
- !CDigOutStimulatorFunctionNet
  - CDigOutStimulatorFunctionNet, [125](#)
- !CEXternDTesterDeviceNet
  - CEXternDTesterDeviceNet, [130](#)
- !CGrapheneFunctionNet
  - CGrapheneFunctionNet, [170](#)
- !CInterfaceboard2FunctionNet
  - CInterfaceboard2FunctionNet, [187](#)
- !CInterfaceboardFunctionNet
  - CInterfaceboardFunctionNet, [189](#)
- !CLIH3DeviceNet
  - CLIH3DeviceNet, [192](#)
- !CMEA2100x256FunctionNet
  - CMEA2100x256FunctionNet, [332](#)
- !CMcsUsbFunctionNet
  - CMcsUsbFunctionNet, [301](#)
- !CMcsUsbListNet
  - CMcsUsbListNet, [308](#)
- !CMcsUsbNet
  - CMcsUsbNet, [314](#)
- !CMeFunctionNet
  - CMeFunctionNet, [365](#)
- !CMeaCleanDeviceNet
  - CMeaCleanDeviceNet, [337](#)
- !CMeaCoatDeviceNet
  - CMeaCoatDeviceNet, [341](#)
- !CMultiBatteryChargerDeviceNet
  - CMultiBatteryChargerDeviceNet, [367](#)
- !CMultiwellCallbackFunctionNet
  - CMultiwellCallbackFunctionNet, [375](#)
- !CMultiwellDeviceNet
  - CMultiwellDeviceNet, [378](#)
- !CMultiwellOptoStimFunctionNet
  - CMultiwellOptoStimFunctionNet, [385](#)
- !CPPCFunctionNet
  - CPPCFunctionNet, [418](#)
- !CPedoterDeviceNet
  - CPedoterDeviceNet, [399](#)
- !CPositionIIDeviceNet
  - CPositionIIDeviceNet, [405](#)
- !CPositionImpDeviceNet
  - CPositionImpDeviceNet, [413](#)
- !CProgramPressureCurveNet
  - CProgramPressureCurveNet, [432](#)
- !CPulseGeneratorFunctionNet
  - CPulseGeneratorFunctionNet, [433](#)
- !CRFFFunctionNet
  - CRFFFunctionNet, [441](#)
- !CSCUFunctionNet
  - CSCUFunctionNet, [497](#)
- !CTEERFunctionNet
  - CTEERFunctionNet, [594](#)
- !CUsbDeviceConfigurationFunctionNet
  - CUsbDeviceConfigurationFunctionNet, [604](#)
- !CWarnerUssingDeviceNet
  - CWarnerUssingDeviceNet, [622](#)
- !CWarnerUssingFunctionNet
  - CWarnerUssingFunctionNet, [624](#)
- !CWarnerValveControllerDeviceNet
  - CWarnerValveControllerDeviceNet, [643](#)
- !CWarnerValveControllerDeviceTesterFunctionNet
  - CWarnerValveControllerDeviceTesterFunctionNet, [665](#)
- !SidebandData
  - CStimulusFunctionNet::SidebandData, [703](#)
- !StimulusDeviceDataAndUnrolledData
  - CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, [705](#)
- \_AdditionalText
  - HeadstageIDTypeObject, [689](#)
- \_IdType
  - HeadstageIDTypeObject, [689](#)
- ~CCMOSMeaDeviceNet
  - CCMOSMeaDeviceNet, [110](#)
- ~CChannelTestDeviceNet
  - CChannelTestDeviceNet, [98](#)
- ~CCreateFilterNet
  - CCreateFilterNet, [113](#)
- ~CDacCalibrationFunctionNet
  - CDacCalibrationFunctionNet, [115](#)
- ~CDigOutStimulatorFunctionNet
  - CDigOutStimulatorFunctionNet, [125](#)
- ~CEXternDTesterDeviceNet
  - CEXternDTesterDeviceNet, [130](#)
- ~CFilterCoefficientsNet
  - CFilterCoefficientsNet, [132](#)
- ~CFilterPropertyNet
  - CFilterPropertyNet, [137](#)
- ~CFluidControlDeviceNet
  - CFluidControlDeviceNet, [140](#)
- ~CGenericDevelopDeviceNet
  - CGenericDevelopDeviceNet, [154](#)
- ~CGilsonDeviceNet
  - CGilsonDeviceNet, [166](#)
- ~CGrapheneASICDeviceNet
  - CGrapheneASICDeviceNet, [167](#)
- ~CGrapheneFunctionNet
  - CGrapheneFunctionNet, [170](#)
- ~CInterfaceboard2FunctionNet
  - CInterfaceboard2FunctionNet, [187](#)
- ~CInterfaceboardFunctionNet
  - CInterfaceboardFunctionNet, [189](#)
- ~CLIH3DeviceNet
  - CLIH3DeviceNet, [192](#)
- ~CMEA2100x256FunctionNet
  - CMEA2100x256FunctionNet, [332](#)
- ~CMcsBusNet
  - CMcsBusNet, [238](#)

- ~CMcsBus\_AxisParametersNet
  - CMcsBus\_AxisParametersNet, [200](#)
- ~CMcsBus\_ExtensionNet
  - CMcsBus\_ExtensionNet, [201](#)
- ~CMcsBus\_FYIExtensionNet
  - CMcsBus\_FYIExtensionNet, [202](#)
- ~CMcsBus\_MotorControlNet
  - CMcsBus\_MotorControlNet, [207](#)
- ~CMcsBus\_SensorNet
  - CMcsBus\_SensorNet, [223](#)
- ~CMcsBus\_TempSensorNet
  - CMcsBus\_TempSensorNet, [232](#)
- ~CMcsBus\_VoltageModeNet
  - CMcsBus\_VoltageModeNet, [234](#)
- ~CMcsUsbDacqNet
  - CMcsUsbDacqNet, [248](#)
- ~CMcsUsbFactoryNet
  - CMcsUsbFactoryNet, [293](#)
- ~CMcsUsbFunctionNet
  - CMcsUsbFunctionNet, [301](#)
- ~CMcsUsbListEntryNet
  - CMcsUsbListEntryNet, [302](#)
- ~CMcsUsbListNet
  - CMcsUsbListNet, [308](#)
- ~CMcsUsbNet
  - CMcsUsbNet, [314](#)
- ~CMeFunctionNet
  - CMeFunctionNet, [365](#)
- ~CMeaCleanDeviceNet
  - CMeaCleanDeviceNet, [337](#)
- ~CMeaCoatDeviceNet
  - CMeaCoatDeviceNet, [341](#)
- ~CMeaDeviceNet
  - CMeaDeviceNet, [347](#)
- ~CMealImpedanceDeviceNet
  - CMealImpedanceDeviceNet, [360](#)
- ~CMeaSwitchDeviceNet
  - CMeaSwitchDeviceNet, [362](#)
- ~CMeaUSBDeviceNet
  - CMeaUSBDeviceNet, [364](#)
- ~CMultiBatteryChargerDeviceNet
  - CMultiBatteryChargerDeviceNet, [367](#)
- ~CMultiwellCallbackFunctionNet
  - CMultiwellCallbackFunctionNet, [375](#)
- ~CMultiwellDeviceNet
  - CMultiwellDeviceNet, [378](#)
- ~CMultiwellOptoStimFunctionNet
  - CMultiwellOptoStimFunctionNet, [385](#)
- ~CNF\_GenDeviceNet
  - CNF\_GenDeviceNet, [389](#)
- ~CokuvisionStimulatorDeviceNet
  - CokuvisionStimulatorDeviceNet, [394](#)
- ~CPPCFunctionNet
  - CPPCFunctionNet, [418](#)
- ~CPathIdentDeviceNet
  - CPathIdentDeviceNet, [398](#)
- ~CPedoterDeviceNet
  - CPedoterDeviceNet, [399](#)
- ~CPeristalticPumpDeviceNet
  - CPeristalticPumpDeviceNet, [401](#)
- ~CPgaDeviceNet
  - CPgaDeviceNet, [402](#)
- ~CPositionIIDeviceNet
  - CPositionIIDeviceNet, [405](#)
- ~CPositionImpDeviceNet
  - CPositionImpDeviceNet, [413](#)
- ~CProgramPressureCurveNet
  - CProgramPressureCurveNet, [431](#)
- ~CPulseGeneratorFunctionNet
  - CPulseGeneratorFunctionNet, [433](#)
- ~CRFFFunctionNet
  - CRFFFunctionNet, [441](#)
- ~CRetinaLedDeviceNet
  - CRetinaLedDeviceNet, [439](#)
- ~CRoboDeviceNet
  - CRoboDeviceNet, [466](#)
- ~CRoboFluidDeviceNet
  - CRoboFluidDeviceNet, [480](#)
- ~CSCUFunctionNet
  - CSCUFunctionNet, [497](#)
- ~CSafeISDeviceNet
  - CSafeISDeviceNet, [491](#)
- ~CStg200xBasicNet
  - CStg200xBasicNet, [517](#)
- ~CStg200xDownloadNet
  - CStg200xDownloadNet, [560](#)
- ~CSw2to64DeviceNet
  - CSw2to64DeviceNet, [577](#)
- ~CTEERFunctionNet
  - CTEERFunctionNet, [594](#)
- ~CTEERMachineDeviceNet
  - CTEERMachineDeviceNet, [602](#)
- ~CTcxDeviceNet
  - CTcxDeviceNet, [581](#)
- ~CUsbDeviceConfigurationFunctionNet
  - CUsbDeviceConfigurationFunctionNet, [604](#)
- ~CWarnerUssingDeviceNet
  - CWarnerUssingDeviceNet, [621](#)
- ~CWarnerUssingFunctionNet
  - CWarnerUssingFunctionNet, [624](#)
- ~CWarnerValveControllerDeviceNet
  - CWarnerValveControllerDeviceNet, [643](#)
- ~CWarnerValveControllerDeviceTesterFunctionNet
  - CWarnerValveControllerDeviceTesterFunctionNet, [664](#)
- ~DriverVersionNet
  - DriverVersionNet, [677](#)
- ~SidebandData
  - CStimulusFunctionNet::SidebandData, [703](#)
- ~StimulusDeviceDataAndUnrolledData
  - CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, [705](#)
- A
  - CFilterCoefficientsNet, [133](#)
  - Mcs::Usb, [69](#)
- AccelOnly

- Mcs::Usb, 89
- AdapterTypeEnumNet
  - Mcs::Usb, 51
- AdditionalText
  - HeadstageIDTypeObject, 690
- AddLoopEntry
  - CRetinaLedDeviceNet, 439
- AddSelectedChannelsQueue
  - CMcsUsbDacqNet, 248, 249, 251
- AddSoftwareKey
  - CMcsUsbNet, 314
- AddTableEntry
  - CRetinaLedDeviceNet, 439
- ALA\_VC3
  - Mcs::Usb, 75, 89
- ALA\_VC3\_DEVICE
  - Mcs::Usb, 62
- ALTERA
  - Mcs::Usb, 54
- Altera
  - FirmwareDestinationNames, 683
- ALTERA\_BASE
  - Mcs::Usb, 55
- ALTERA\_BOOTSTRAP
  - Mcs::Usb, 55
- ALTERA\_GOLD
  - Mcs::Usb, 54
- ALTERA\_TARGET1
  - Mcs::Usb, 55
- ALTERA\_TARGET2
  - Mcs::Usb, 55
- ALTERA\_TARGET3
  - Mcs::Usb, 55
- AlwaysOn
  - Mcs::Usb, 78
- Ampere
  - Mcs::Usb, 52, 89
- amplification
  - CMeaAudioFunctionNet::s\_setaudionet, 702
  - CW2100\_FunctionNet::AudioChannelsNet, 94
- AmplifierSettle
  - CIntanMea\_FunctionNet, 185
- AMS\_Dongle
  - Mcs::Usb, 76
- Analog
  - Mcs::Usb, 75, 92
- AnalogGain
  - CMeaDeviceNet, 352
- AnalogGroup
  - Mcs::Usb, 59
- AnalogOut\_DAC\_Range\_EnumNet
  - Mcs::Usb, 51
- AnalogSource\_HS1
  - Mcs::Usb, 52
- AnalogSource\_HS2
  - Mcs::Usb, 52
- AnalogSource\_IF
  - Mcs::Usb, 52
- AnalogSourceEnumNet
  - Mcs::Usb, 52
- AnalogUnitEnumNet
  - Mcs::Usb, 52
- Any
  - Mcs::Usb, 75, 89
- ApplyGains
  - CPgaDeviceNet, 402
- AreTransistorVoltagesSet
  - CCMOSMea\_FunctionNet, 101
- Armed
  - Mcs::Usb, 83
- ASMedia
  - Mcs::Usb, 88
- AssociateToThis
  - CMcsUsbNet, 314
- AudioTestChannelGroup
  - Mcs::Usb, 59, 71, 81, 91
- AutomaticAnalogOut
  - CSCUFunctionNet, 497
- Aux
  - Mcs::Usb, 62
- AuxIn
  - Mcs::Usb, 63, 71, 81, 84, 91
- AuxPort
  - Mcs::Usb, 57
- Axes\_I
  - CRoboDeviceNet, 474
- Axes\_X
  - CRoboDeviceNet, 474
- Axes\_Y
  - CRoboDeviceNet, 474
- Axes\_Z
  - CRoboDeviceNet, 475
- Axis\_I
  - CRoboDeviceNet, 475
- Axis\_X
  - CRoboDeviceNet, 475
- Axis\_Y
  - CRoboDeviceNet, 475
- Axis\_Z
  - CRoboDeviceNet, 475
- B
  - CFilterCoefficientsNet, 133
  - Mcs::Usb, 69
- BatteryState, 94
  - Charge, 94
  - ChargeRegionString, 94
  - ChargeString, 94
  - Voltage, 95
  - VoltageString, 95
- BcdDevice
  - DeviceldNet, 673
- BeginImpedanceCheck
  - CIntanMea\_FunctionNet, 185
- Bessel
  - Mcs::Usb, 66
- BesselFilterHighPassNet, 95

- BesselFilterHighPassNet, [95](#)
- BesselFilterLowPassNet, [95](#)
  - BesselFilterLowPassNet, [96](#)
- BMI
  - Mcs::Usb, [85](#)
- bmRequestType
  - usbSetupPacket\_t, [706](#)
- BOOST\_BIT
  - CW2100\_StimulatorFunctionNet, [620](#)
- Bootstrap
  - FirmwareDestinationNames, [683](#)
  - Mcs::Usb, [54](#)
- BootstrapOtherCypress
  - Mcs::Usb, [54](#)
- Both
  - Mcs::Usb, [89](#)
- Break
  - Mcs::Usb, [79](#)
- bRequest
  - usbSetupPacket\_t, [706](#)
- BurnAdcOffset
  - COctoPotDeviceNet, [390](#)
- BurnDacOffset
  - CDacCalibrationFunctionNet, [115](#)
  - COctoPotDeviceNet, [391](#)
- BUS0MCSBUS0
  - Mcs::Usb, [53](#)
- BUS0MCSBUS1
  - Mcs::Usb, [53](#)
- BUS0MCSBUS10
  - Mcs::Usb, [53](#)
- BUS0MCSBUS11
  - Mcs::Usb, [53](#)
- BUS0MCSBUS12
  - Mcs::Usb, [53](#)
- BUS0MCSBUS13
  - Mcs::Usb, [53](#)
- BUS0MCSBUS14
  - Mcs::Usb, [53](#)
- BUS0MCSBUS15
  - Mcs::Usb, [53](#)
- BUS0MCSBUS2
  - Mcs::Usb, [53](#)
- BUS0MCSBUS3
  - Mcs::Usb, [53](#)
- BUS0MCSBUS4
  - Mcs::Usb, [53](#)
- BUS0MCSBUS5
  - Mcs::Usb, [53](#)
- BUS0MCSBUS6
  - Mcs::Usb, [53](#)
- BUS0MCSBUS7
  - Mcs::Usb, [53](#)
- BUS0MCSBUS8
  - Mcs::Usb, [53](#)
- BUS0MCSBUS9
  - Mcs::Usb, [53](#)
- BUS1\_MCSBUS1
  - FirmwareDestinationNames, [683](#)
- BUS1\_MCSBUS2
  - FirmwareDestinationNames, [683](#)
- BUS1MCSBUS0
  - Mcs::Usb, [53](#)
- BUS1MCSBUS1
  - Mcs::Usb, [53](#)
- BUS1MCSBUS10
  - Mcs::Usb, [53](#)
- BUS1MCSBUS11
  - Mcs::Usb, [53](#)
- BUS1MCSBUS12
  - Mcs::Usb, [53](#)
- BUS1MCSBUS13
  - Mcs::Usb, [53](#)
- BUS1MCSBUS14
  - Mcs::Usb, [53](#)
- BUS1MCSBUS15
  - Mcs::Usb, [53](#)
- BUS1MCSBUS2
  - Mcs::Usb, [53](#)
- BUS1MCSBUS3
  - Mcs::Usb, [53](#)
- BUS1MCSBUS4
  - Mcs::Usb, [53](#)
- BUS1MCSBUS5
  - Mcs::Usb, [53](#)
- BUS1MCSBUS6
  - Mcs::Usb, [53](#)
- BUS1MCSBUS7
  - Mcs::Usb, [53](#)
- BUS1MCSBUS8
  - Mcs::Usb, [53](#)
- BUS1MCSBUS9
  - Mcs::Usb, [53](#)
- BUS2MCSBUS0
  - Mcs::Usb, [54](#)
- BUS2MCSBUS1
  - Mcs::Usb, [53](#)
- BUS2MCSBUS10
  - Mcs::Usb, [54](#)
- BUS2MCSBUS11
  - Mcs::Usb, [54](#)
- BUS2MCSBUS12
  - Mcs::Usb, [54](#)
- BUS2MCSBUS13
  - Mcs::Usb, [54](#)
- BUS2MCSBUS14
  - Mcs::Usb, [54](#)
- BUS2MCSBUS15
  - Mcs::Usb, [54](#)
- BUS2MCSBUS2
  - Mcs::Usb, [53](#)
- BUS2MCSBUS3
  - Mcs::Usb, [53](#)
- BUS2MCSBUS4
  - Mcs::Usb, [53](#)
- BUS2MCSBUS5
  - Mcs::Usb, [53](#)

- Mcs::Usb, [53](#)
- BUS2MCSBUS6
  - Mcs::Usb, [53](#)
- BUS2MCSBUS7
  - Mcs::Usb, [54](#)
- BUS2MCSBUS8
  - Mcs::Usb, [54](#)
- BUS2MCSBUS9
  - Mcs::Usb, [54](#)
- BUSNUMBER0
  - Mcs::Usb, [53](#)
- BUSNUMBER1
  - Mcs::Usb, [53](#)
- BUSNUMBER2
  - Mcs::Usb, [53](#)
- BusType
  - DeviceIdNet, [673](#)
- Butterworth
  - Mcs::Usb, [66](#)
- ButterworthFilterHighPassNet, [96](#)
  - ButterworthFilterHighPassNet, [96](#)
- ButterworthFilterLowPassNet, [97](#)
  - ButterworthFilterLowPassNet, [97](#)
- CalibrateThermocouple
  - CFluidControlDeviceNet, [140](#)
  - CTcxDeviceNet, [581](#)
- Campden\_Ci4600EphysVideoDataIntegrator
  - Mcs::Usb, [75](#)
- CancelInternalCalibration
  - CTEERFunctionNet, [594](#)
- CancelPoolLoop
  - CRoboDeviceNet, [466](#)
- CancelPoolLoopAndStopMovement
  - CRoboDeviceNet, [466](#)
- CancelTableLoop
  - CRoboDacqNet, [451](#)
- CancelTableLoopAndStopTable
  - CRoboDacqNet, [451](#)
- CapacityTest
  - CMultiBatteryChargerDeviceNet, [367](#)
- CatchAmp
  - Mcs::Usb, [74](#)
- CatchAmpGetAdcMean
  - CMcsBus\_SensorNet, [223](#)
- CatchAmpGetAdcValue
  - CMcsBus\_SensorNet, [223](#)
- CatchAmpGetAdcValueH
  - CMcsBus\_SensorNet, [223](#)
- CatchAmpGetAdcValueL
  - CMcsBus\_SensorNet, [223](#)
- CatchAmpGetDacAmplitude
  - CMcsBus\_SensorNet, [224](#)
- CatchAmpGetDacEnable
  - CMcsBus\_SensorNet, [224](#)
- CatchAmpGetDacOffset
  - CMcsBus\_SensorNet, [224](#)
- CatchAmpGetPwmEnable
  - CMcsBus\_SensorNet, [224](#)
- CatchAmpSetDacAmplitude
  - CMcsBus\_SensorNet, [224](#)
- CatchAmpSetDacEnable
  - CMcsBus\_SensorNet, [224](#)
- CatchAmpSetDacOffset
  - CMcsBus\_SensorNet, [224](#)
- CatchAmpSetPwmEnable
  - CMcsBus\_SensorNet, [224](#)
- CChannelTestDeviceNet, [97](#)
  - ~CChannelTestDeviceNet, [98](#)
  - CChannelTestDeviceNet, [98](#)
  - SetAmplitude, [98](#)
  - SetAttenuation, [98](#)
  - SetFrequency, [98](#)
  - SetWaveform, [98](#)
- CCMOSMea\_FunctionNet, [98](#)
  - AreTransistorVoltagesSet, [101](#)
  - CCMOSMea\_FunctionNet, [100](#), [101](#)
  - ClearSTGOutput, [101](#)
  - DetectChipType, [101](#)
  - EnableChannelsInGroup, [101](#)
  - GetADCInputOffset, [101](#)
  - GetBath, [101](#)
  - GetBathMode, [101](#)
  - GetEnabledChannelsInGroup, [102](#)
  - GetGate, [102](#)
  - GetGNDI, [102](#)
  - GetGroupADCBits, [102](#)
  - GetGroupChannelBitmaskBySelect, [102](#)
  - GetGroupChannelBitmaskHS1NCBathCurrent, [102](#), [103](#)
  - GetGroupChannelBitmaskHS1NCCol2Current, [103](#)
  - GetGroupChannelBitmaskHS1NChipTemp, [103](#)
  - GetGroupChannelBitmaskHS1Sidebands, [103](#)
  - GetGroupChannelBitmaskHS1TriggerStatus, [103](#), [104](#)
  - GetGroupChannelBitmaskIFDigChannels, [104](#)
  - GetGroupChannelBitmaskInterfaceADC, [104](#)
  - GetGroupChannelBitmaskPacketFrameContext, [104](#)
  - GetGroupChannelBitmaskSTG1DACSignal, [104](#), [105](#)
  - GetGroupDCOffset, [105](#)
  - GetGroupID, [105](#)
  - GetGroupNumberOfChannels, [105](#)
  - GetGroupResolutionPerDigit, [105](#)
  - GetGroupSampleSize, [106](#)
  - GetGroupType, [106](#)
  - GetGroupUnit, [106](#)
  - GetMaxNumOfColumns, [106](#)
  - GetNeurochipMemoryData, [106](#)
  - GetNeurochipMemorySize, [107](#)
  - GetNumberOfSupportedGroups, [107](#)
  - GetSourceBulk, [107](#)
  - GetSourceDrain, [107](#)
  - GetSourceGate, [107](#)
  - GetStimulusSites, [107](#)

- GetVDD3I, 107
- GetVDDI, 107
- IsChipPowered, 107
- IsGateFloating, 107
- PowerChip, 108
- SetADCInputOffset, 108
- SetBath, 108
- SetBathMode, 108
- SetGate, 108
- SetGateFloating, 108
- SetGateToVOP, 108
- SetNeurochipMemoryData, 108
- SetSourceBulk, 108
- SetSourceDrain, 109
- SetSourceGate, 109
- SetStimulusSites, 109
- UpdateTransistorVoltages, 109
- VOPSTimerSetResetTimes, 109
- CCMOSMeaDeviceNet, 109
  - ~CCMOSMeaDeviceNet, 110
  - CCMOSMeaDeviceNet, 110
  - CMosMea, 112
  - GetAvailableBaseSamplerates, 110
  - GetBaseSamplerate, 111
  - GetChannelDataI16, 111
  - GetChannelDataI32, 111
  - GetChannelDataUI16, 111
  - GetChannelDataUI32, 111
  - GetCMOSDataDictionary, 111
  - GetMaxReadableColumns, 111
  - SetBaseSamplerate, 112
  - SetRegionOfInterests, 112
  - Stimulus, 112
  - UpdateChannelBlock, 112
- CCMOSMeaDeviceNet::CRegionOfInterestRect, 437
  - CRegionOfInterestRect, 437
  - DeepCopy, 437
  - m\_Bottom, 438
  - m\_Left, 438
  - m\_Right, 438
  - m\_Top, 438
- CCreateFilterNet, 112
  - ~CCreateFilterNet, 113
  - CCreateFilterNet, 113
  - CutoffFrequency, 114
  - FindFilter, 113
  - GetBiQuad, 114
  - GetBiQuads, 114
  - NumCoefSets, 114
  - Order, 114
  - SampleRate, 114
  - Scale, 114
- CDacCalibrationFunctionNet, 114
  - !CDacCalibrationFunctionNet, 115
  - ~CDacCalibrationFunctionNet, 115
  - BurnDacOffset, 115
  - CDacCalibrationFunctionNet, 115
  - GetDacOffset, 116
  - SetDacOffset, 116
- CDacqGroupChannelGenericSelectionNet, 116
  - CDacqGroupChannelGenericSelectionNet, 117
- CDacqGroupChannelSelectionNet, 117
  - CDacqGroupChannelSelectionNet, 117
- CDacqGroupChannelSelectionTemplateNet
  - CDacqGroupChannelSelectionTemplateNet< Dac-  
qGroupChannelEnumTemplateNet, Dac-  
qGroupChannelEnumTemplate, CDevice-  
GroupChannelInfoTemplateNet >, 118
- CDacqGroupChannelSelectionTemplateNet< Dac-  
qGroupChannelEnumTemplateNet, Dac-  
qGroupChannelEnumTemplate, CDevice-  
GroupChannelInfoTemplateNet >, 117
- CDacqGroupChannelSelectionTemplateNet, 118
- EnableChannelsInGroup, 118
- GetDeviceGroupChannelInfos, 118, 119
- GetEnabledChannelsInGroup, 119
- GetGroupID, 119
- GetGroupNumberOfChannels, 119
- GetGroupSampleSize, 119
- GetGroupType, 120
- GetNumberOfSupportedGroups, 120
- CDeviceGroupChannelInfoGenericNet, 120
  - CDeviceGroupChannelInfoGenericNet, 120
- CDeviceGroupChannelInfoMEA2100\_256Net, 121
  - CDeviceGroupChannelInfoMEA2100\_256Net, 121
- CDeviceGroupChannelInfoNet, 121
  - CDeviceGroupChannelInfoNet, 122
- CDeviceGroupChannelInfoSCUNet, 122
  - CDeviceGroupChannelInfoSCUNet, 122
- CDeviceGroupChannelInfoTemplateNet
  - CDeviceGroupChannelInfoTemplateNet< Dacq-  
GroupChannelEnumTemplateNet >, 123
- CDeviceGroupChannelInfoTemplateNet< Dacq-  
GroupChannelEnumTemplateNet >, 122
- CDeviceGroupChannelInfoTemplateNet, 123
- GroupID, 123
- GroupType, 123
- NumberOfChannels, 123
- CDeviceGroupChannelInfoW2100Net, 123
  - CDeviceGroupChannelInfoW2100Net, 124
- CDigOutStimulatorFunctionNet, 124
  - !CDigOutStimulatorFunctionNet, 125
  - ~CDigOutStimulatorFunctionNet, 125
  - CDigOutStimulatorFunctionNet, 125
  - ClearChannel, 125
  - GetGlobalRepeat, 125
  - GetNumberOfChannels, 126
  - GetStartTriggerSlope, 126
  - GetStopTriggerSlope, 126
  - PrepareChannelData, 127
  - SendChannelData, 127
  - SetGlobalRepeat, 127
  - SetStartTriggerSlope, 128
  - SetStopTriggerSlope, 128
- CEncapsulatorDeviceNet, 128
  - CEncapsulatorDeviceNet, 129



- GetRoboFluidDevice, 129
- CExternDTesterDeviceNet, 129
  - !CExternDTesterDeviceNet, 130
  - ~CExternDTesterDeviceNet, 130
  - CExternDTesterDeviceNet, 130
  - Read, 130
  - Read2, 130
  - Write, 130
  - Write2, 131
- CFilterCoefficientsNet, 131
  - ~CFilterCoefficientsNet, 132
  - A, 133
  - B, 133
  - CFilterCoefficientsNet, 131, 132
  - GetUintA, 132
  - GetUintB, 132
  - IsEqual, 132
- CFilterCoefficientsNet::s\_FilterAttributesNet, 701
  - CommaPositionA, 702
  - CommaPositionB, 702
  - PostCommaA, 702
  - PostCommaB, 702
  - PreCommaA, 702
  - PreCommaB, 702
  - s\_FilterAttributesNet, 701
  - ToCpp, 701
- CFilterConfigurationNet, 133
  - CFilterConfigurationNet, 133
  - EraseFilterParameterPermanent, 134
  - GetFilterAttributes, 134
  - GetHighpassFilterEnable, 134
  - ResetHighpassFilter, 134
  - SetFilterParameter, 134
  - SetFilterParameterPermanent, 134
  - SetHighpassFilterEnable, 135
- CFilterConfigurationRegisterNet, 135
  - CFilterConfigurationRegisterNet, 135
  - EraseFilterParameterPermanent, 135, 136
  - SetFilterParameter, 136
  - SetFilterParameterPermanent, 136
- CFilterPropertyNet, 137
  - ~CFilterPropertyNet, 137
  - CFilterPropertyNet, 137
  - CornerFrequency, 137
  - CornerFrequencymHz, 138
  - FilterActive, 138
  - FilterBand, 138
  - FilterFamily, 138
  - FilterType, 138
  - Order, 138
  - ToString, 137
- CFirmwareDestinationNet
  - Mcs::Usb, 52
- CFluidControlDeviceNet, 138
  - ~CFluidControlDeviceNet, 140
  - CalibrateThermocouple, 140
  - CFluidControlDeviceNet, 140
  - GetAdc, 140
  - GetDigin, 140
  - GetDigout, 141
  - GetPWM, 141
  - GetReferenceTemperature, 141
  - GetSingleValve, 141
  - GetThermocoupleCalibration, 142
  - GetThermocoupleNanovoltPerKelvin, 142
  - GetThermocoupleTemperature, 142
  - GetValve, 143
  - McsBus\_VoltageMode, 145
  - SetDigout, 143
  - SetPWM, 143
  - SetSingleValve, 143
  - SetThermocoupleNanovoltPerKelvin, 145
  - SetValve, 145
- CFYIDeviceNet, 145
  - CFYIDeviceNet, 146
  - FYIProgram, 146
  - FYITemp, 146
  - Sensor, 146
- CGenericDevelopDeviceNet, 147
  - ~CGenericDevelopDeviceNet, 154
  - CGenericDevelopDeviceNet, 154
  - ClosePipe, 154
  - FindEndpoints, 154
  - GetBuffer, 154
  - GetByteBuffer, 155
  - GetIntBuffer, 155
  - GetShortBuffer, 156
  - GetUByteBuffer, 157
  - GetUIntBuffer, 157
  - GetUShortBuffer, 158
  - OpenPipe, 159
  - ReadPipe, 159
  - ResetPipe, 159
  - SetBuffer, 160
  - SetByteBuffer, 160
  - SetIntBuffer, 160
  - SetShortBuffer, 161
  - SetUByteBuffer, 162
  - SetUIntBuffer, 162
  - SetUShortBuffer, 163
  - SetValue, 163
  - VendorInRequest, 164
  - VendorOutRequest, 164
  - WritePipe, 164
- CGilsonDeviceNet, 165
  - ~CGilsonDeviceNet, 166
  - CGilsonDeviceNet, 165
  - ConnectSlave, 166
  - GetLastAnswer, 166
  - m\_pGilsonDevice, 166
  - SendBuffered, 166
  - SendImmediate, 166
  - SendImmediateGetResponse, 166
- CGrapheneASICDeviceNet, 167
  - ~CGrapheneASICDeviceNet, 167
  - CGrapheneASICDeviceNet, 167

- GetAvailableBaseSamplerates, 167
- SetBaseSamplerate, 168
- SetRegionOfInterests, 168
- CGrapheneFunctionNet, 168
  - !CGrapheneFunctionNet, 170
  - ~CGrapheneFunctionNet, 170
  - CGrapheneFunctionNet, 170
  - GetCur2VolResistance, 170
  - GetDACOffset, 171
  - GetVdsVgs, 171, 172
  - GetVdVs, 172
  - GetVdVsDAC, 172, 174
  - GetVoltageRange, 174
  - GetVoltageReached, 174, 175
  - GetVoltageResolution, 175
  - SetDACOffset, 175, 176
  - SetVdsVgs, 176
  - SetVdVs, 177
  - SetVdVsDAC, 177
  - SetVoltageRange, 178
  - SetVoltageResolution, 178
- ChangeSerialNumber
  - CMcsUsbFactoryNet, 293
- channel
  - CMeaAudioFunctionNet::s\_setaudionet, 702
  - CW2100\_FunctionNet::AudioChannelsNet, 94
- ChannelBlock\_AvailFrames
  - CMcsUsbDacqNet, 251
- ChannelBlock\_ReadAsFrameArrayI16
  - CMcsUsbDacqNet, 251, 252
- ChannelBlock\_ReadAsFrameArrayI32
  - CMcsUsbDacqNet, 253
- ChannelBlock\_ReadAsFrameArrayUI16
  - CMcsUsbDacqNet, 254, 255
- ChannelBlock\_ReadAsFrameArrayUI32
  - CMcsUsbDacqNet, 255, 256
- ChannelBlock\_ReadFramesDictI16
  - CMcsUsbDacqNet, 257
- ChannelBlock\_ReadFramesDictI32
  - CMcsUsbDacqNet, 257
- ChannelBlock\_ReadFramesDictUI16
  - CMcsUsbDacqNet, 258
- ChannelBlock\_ReadFramesDictUI32
  - CMcsUsbDacqNet, 259
- ChannelBlock\_ReadFramesI16
  - CMcsUsbDacqNet, 259, 260
- ChannelBlock\_ReadFramesI32
  - CMcsUsbDacqNet, 260, 261
- ChannelBlock\_ReadFramesUI16
  - CMcsUsbDacqNet, 262
- ChannelBlock\_ReadFramesUI32
  - CMcsUsbDacqNet, 263, 264
- channeldata\_current
  - Mcs::Usb, 84
- channeldata\_current\_always\_boost
  - Mcs::Usb, 84
- channeldata\_current\_always\_boost\_own\_sync
  - Mcs::Usb, 84
- channeldata\_current\_own\_boost\_gnd\_sync
  - Mcs::Usb, 84
- channeldata\_current\_own\_sync
  - Mcs::Usb, 84
- channeldata\_positive\_current
  - Mcs::Usb, 84
- channeldata\_positive\_current\_own\_boost\_gnd\_sync
  - Mcs::Usb, 84
- channeldata\_positive\_current\_own\_sync
  - Mcs::Usb, 84
- channeldata\_positive\_voltage
  - Mcs::Usb, 84
- channeldata\_voltage
  - Mcs::Usb, 84
- ChannelDataEvent
  - CMcsUsbDacqNet, 289
- ChannelPIC
  - Mcs::Usb, 54
- ChannelReset
  - CMultiBatteryChargerDeviceNet, 368
- ChannelTest
  - Mcs::Usb, 76
- Charge
  - BatteryState, 94
- ChargeRegionString
  - BatteryState, 94
- ChargeString
  - BatteryState, 94
- ChecksumAndPacketCounter
  - Mcs::Usb, 58
- CHiClampDeviceNet, 179
  - CHiClampDeviceNet, 179
  - RoboDacq, 180
- CHLADacqNet, 180
  - CHLADacqNet, 180
- CHLADeviceNet, 180
  - CHLADeviceNet, 181
  - HLADacq, 181
  - SerialPort, 181
- CHWInfo
  - CMcsUsbDacqNet::CHWInfo, 182
- Ci4600Intan
  - Mcs::Usb, 51
- CIntanMea\_FunctionNet, 184
  - AmplifierSettle, 185
  - BeginImpedanceCheck, 185
  - CIntanMea\_FunctionNet, 185
  - GetDSPHighPassByIndex, 185
  - GetImpedanceResult, 185
  - GetIntanRegister, 185
  - GetLowerFrequencyByIndex, 185
  - GetUpperFrequencyByIndex, 186
  - SetBandwidthByIndex, 186
  - SetDiagnosticMode, 186
  - SetDSPHighPassByIndex, 186
  - SetIntanRegister, 186
- CInterfaceboard2FunctionNet, 186
  - !CInterfaceboard2FunctionNet, 187

- ~CInterfaceboard2FunctionNet, 187
- CInterfaceboard2FunctionNet, 187
- GetIoVoltage, 187
- SetIoVoltage, 188
- CInterfaceboardFunctionNet, 188
  - !CInterfaceboardFunctionNet, 189
  - ~CInterfaceboardFunctionNet, 189
  - CInterfaceboardFunctionNet, 189
  - GetCardinalDacqSamplerate, 189
  - GetCardinalStgOutputrate, 189
  - SetCardinalDacqSamplerate, 189
  - SetCardinalStgOutputrate, 190
- ClampAmpRestart
  - CRoboDacqNet, 451
- ClampModeCurrent
  - Mcs::Usb, 85
- ClampModeInternalCalibration
  - Mcs::Usb, 85
- ClampModeOpen
  - Mcs::Usb, 85
- ClampModeVoltage
  - Mcs::Usb, 85
- ClearBuffers
  - CMcsUsbDacqNet, 264
- ClearChannel
  - CDigOutStimulatorFunctionNet, 125
- ClearChannel\_PrepAndSendData
  - CStg200xDownloadNet, 560
  - CStimulusFunctionNet, 567
- ClearChannelData
  - CStg200xDownloadBasicNet, 552
  - CStimulusFunctionNet, 567
  - CW2100\_StimulatorFunctionNet, 615
- ClearMultiplexedData
  - CStimulusFunctionNet, 567
- ClearSTGOutput
  - CCMOSMea\_FunctionNet, 101
- ClearStimulusParametersCache
  - CW2100\_FunctionNet, 608
- ClearSyncData
  - CStg200xDownloadBasicNet, 553
  - CStimulusFunctionNet, 567
- ClearTable
  - CRetinalLedDeviceNet, 439
- ClearTableName
  - CWarnerValveControllerDeviceNet, 643
- ClearUserDefinedNameCache
  - CW2100\_FunctionNet, 608
- ClearValveTable
  - CWarnerValveControllerDeviceNet, 643
- CLIH3DeviceNet, 190
  - !CLIH3DeviceNet, 192
  - ~CLIH3DeviceNet, 192
  - CLIH3DeviceNet, 192
  - DummyCommand, 192
  - EnableUserTrigger, 192
  - ErasePermanentAdcOffset, 193
  - ErasePermanentDacOffset, 193
  - GetAdcOffset, 193
  - GetAudioOutDacParameter, 193
  - GetDacIdleValue, 194
  - GetDacOffset, 194
  - GetDacqRunStatus, 194
  - GetDacUseldleValue, 194
  - GetDigInState, 195
  - GetEEPromPage, 195
  - GetSampleInterval, 195
  - IsUserTriggerEnabled, 195
  - ReadClipping, 196
  - ReadUARTData, 196
  - SendCommand, 196
  - SetAdcOffset, 196
  - SetAdcOffsetPermanent, 197
  - SetAudioOutDacParameter, 197
  - SetDacIdleValue, 197
  - SetDacOffset, 197
  - SetDacOffsetPermanent, 198
  - SetDacUseldleValue, 198
  - SetDigOutState, 198
  - SetEEPromPage, 198
  - SetSampleInterval, 199
  - StimulusFunction, 199
  - WriteUARTData, 199
- Close
  - Mcs::Usb, 74
- CloseAllValves
  - CRoboFluidDeviceNet, 480
- ClosePipe
  - CGenericDevelopDeviceNet, 154
- ClosePlateClamp
  - CMultiwellDeviceNet, 378
- CMcsBus\_AxisParametersNet, 199
  - ~CMcsBus\_AxisParametersNet, 200
  - CMcsBus\_AxisParametersNet, 200
  - GetAxisParametersSignedEeprom, 200
  - GetAxisParametersUnsignedEeprom, 200
  - SetAxisParametersEeprom, 200, 201
- CMcsBus\_ExtensionNet, 201
  - ~CMcsBus\_ExtensionNet, 201
  - CMcsBus\_ExtensionNet, 201
  - GetLEDSwitch, 202
  - SetLEDSwitch, 202
- CMcsBus\_FYIExtensionNet, 202
  - ~CMcsBus\_FYIExtensionNet, 202
  - CMcsBus\_FYIExtensionNet, 202
  - GetDIO, 203
  - GetSingleHeater, 203
  - GetValves, 203
  - SetDIO, 203
  - SetSingleHeater, 203
  - SetValves, 203
- CMcsBus\_MotorControlNet, 204
  - ~CMcsBus\_MotorControlNet, 207
  - CMcsBus\_MotorControlNet, 207
  - GetMCAcceleration, 207
  - GetMCAccelerationEeprom, 207

[GetMCAccelerationShortCommand, 207](#)  
[GetMCAXisRevisionEeprom, 207](#)  
[GetMCBreakCurrent, 207](#)  
[GetMCBreakCurrentEeprom, 208](#)  
[GetMCConfig, 208](#)  
[GetMCConfigEeprom, 208](#)  
[GetMCCurrent, 208](#)  
[GetMCCurrentEeprom, 208](#)  
[GetMCCurrentMode, 208](#)  
[GetMCCurrentModeEeprom, 208](#)  
[GetMCCurrentModeShortCommand, 209](#)  
[GetMCCurrentPosition, 209](#)  
[GetMCCurrentShortCommand, 209](#)  
[GetMCCurrentSpeed, 209](#)  
[GetMCMaxAcceleration, 209](#)  
[GetMCMaxAccelerationEeprom, 209](#)  
[GetMCMaxCurrent, 209](#)  
[GetMCMaxCurrentEeprom, 210](#)  
[GetMCMaxSpeed, 210](#)  
[GetMCMaxSpeedEeprom, 210](#)  
[GetMCMaxTravel, 210](#)  
[GetMCMaxTravelEeprom, 210](#)  
[GetMCMaxTravelShortCommand, 210](#)  
[GetMCMovement, 210](#)  
[GetMCNewPosition, 211](#)  
[GetMCOutputOnOff, 211](#)  
[GetMCPhase, 211](#)  
[GetMCPhaseOffset, 211](#)  
[GetMCReference, 211](#)  
[GetMCReferenceCurrent, 211](#)  
[GetMCReferenceCurrentEeprom, 211](#)  
[GetMCRegulatorGain, 212](#)  
[GetMCRegulatorGainEeprom, 212](#)  
[GetMCScalingFactor, 212](#)  
[GetMCScalingFactorEeprom, 212](#)  
[GetMCSpeed, 212](#)  
[GetMCSpeedEeprom, 212](#)  
[GetMCSpeedShortCommand, 212](#)  
[GetMCSpeedUnitEeprom, 213](#)  
[GetMCStandbyCurrent, 213](#)  
[GetMCStandbyCurrentEeprom, 213](#)  
[GetMCStandbyTime, 213](#)  
[GetMCStandbyTimeEeprom, 213](#)  
[GetSubChannel, 213](#)  
[SetMCAcceleration, 213](#)  
[SetMCAccelerationEeprom, 214](#)  
[SetMCAccelerationShortCommand, 214](#)  
[SetMCAXisRevisionEeprom, 214](#)  
[SetMCBreakCurrent, 214](#)  
[SetMCBreakCurrentEeprom, 214](#)  
[SetMCConfig, 214](#)  
[SetMCConfigEeprom, 215](#)  
[SetMCCurrent, 215](#)  
[SetMCCurrentEeprom, 215](#)  
[SetMCCurrentMode, 215](#)  
[SetMCCurrentModeEeprom, 215](#)  
[SetMCCurrentModeShortCommand, 215](#)  
[SetMCCurrentPosition, 216](#)  
[SetMCCurrentShortCommand, 216](#)  
[SetMCMaxAcceleration, 216](#)  
[SetMCMaxAccelerationEeprom, 216](#)  
[SetMCMaxCurrent, 216](#)  
[SetMCMaxCurrentEeprom, 216](#)  
[SetMCMaxSpeed, 217](#)  
[SetMCMaxSpeedEeprom, 217](#)  
[SetMCMaxTravel, 217](#)  
[SetMCMaxTravelEeprom, 217](#)  
[SetMCMaxTravelShortCommand, 217](#)  
[SetMCNewPosition, 217](#)  
[SetMCOutputOnOff, 218](#)  
[SetMCReference, 218](#)  
[SetMCReferenceCurrent, 218](#)  
[SetMCReferenceCurrentEeprom, 218](#)  
[SetMCRegulatorGain, 218](#)  
[SetMCRegulatorGainEeprom, 218](#)  
[SetMCRotation, 219](#)  
[SetMCScalingFactor, 219](#)  
[SetMCScalingFactorEeprom, 219](#)  
[SetMCSpeed, 219](#)  
[SetMCSpeedEeprom, 219](#)  
[SetMCSpeedShortCommand, 219](#)  
[SetMCSpeedUnitEeprom, 220](#)  
[SetMCStandbyCurrent, 220](#)  
[SetMCStandbyCurrentEeprom, 220](#)  
[SetMCStandbyTime, 220](#)  
[SetMCStandbyTimeEeprom, 220](#)  
[SetSubChannel, 220](#)  
[StartMCMovement, 221](#)  
[StopMCMovement, 221](#)  
[CMcsBus\\_SensorNet, 221](#)  
[~CMcsBus\\_SensorNet, 223](#)  
[CatchAmpGetAdcMean, 223](#)  
[CatchAmpGetAdcValue, 223](#)  
[CatchAmpGetAdcValueH, 223](#)  
[CatchAmpGetAdcValueL, 223](#)  
[CatchAmpGetDacAmplitude, 224](#)  
[CatchAmpGetDacEnable, 224](#)  
[CatchAmpGetDacOffset, 224](#)  
[CatchAmpGetPwmEnable, 224](#)  
[CatchAmpSetDacAmplitude, 224](#)  
[CatchAmpSetDacEnable, 224](#)  
[CatchAmpSetDacOffset, 224](#)  
[CatchAmpSetPwmEnable, 224](#)  
[CMcsBus\\_SensorNet, 223](#)  
[Get2AnalogInput, 225](#)  
[Get2DigitalInput, 225](#)  
[Get4ADC, 225](#)  
[Get4ADCAverage, 225](#)  
[Get4ADCCatchampAverageShift, 225](#)  
[Get4ADCMode, 225](#)  
[Get4DAC, 225](#)  
[GetADCs, 225](#)  
[GetADCsLoop, 226](#)  
[GetBubbleStatus, 226](#)  
[GetDACs, 226](#)  
[GetDetectionThreshold, 226](#)

- GetDetectorValue, [226](#)
- GetLatency, [226](#)
- GetLatencyCounter, [226](#)
- GetMinimalThreshold, [226](#)
- GetMovePump, [227](#)
- GetPiezoState, [227](#)
- GetPressure, [227](#)
- GetPressureOffset, [227](#)
- GetRegulationTimeouts, [227](#)
- GetRegulatorFactor, [228](#)
- GetRegulatorOnOff, [228](#)
- GetRegulatorStatus, [228](#)
- GetRotatePump, [228](#)
- GetSamplePeriode, [228](#)
- GetSollPressure, [228](#)
- GetSyncState, [228](#)
- Set4ADCCatchampAverageShift, [229](#)
- Set4ADCMode, [229](#)
- Set4DAC, [229](#)
- SetDACs, [229](#)
- SetDetectionThreshold, [229](#)
- SetLatency, [229](#)
- SetMinimalThreshold, [229](#)
- SetMovePump, [230](#)
- SetPiezoState, [230](#)
- SetPressureOffset, [230](#)
- SetRegulationTimeouts, [230](#)
- SetRegulatorFactor, [230](#)
- SetRegulatorOnOff, [230](#)
- SetRotatePump, [230](#)
- SetSamplePeriode, [231](#)
- SetSollPressure, [231](#)
- StartSync, [231](#)
- TactSwitchGetState, [231](#)
- TactSwitchSetDisplay, [231](#)
- CMcsBus\_TempSensorNet, [231](#)
  - ~CMcsBus\_TempSensorNet, [232](#)
  - CMcsBus\_TempSensorNet, [232](#)
  - GetNanoVoltsPerKelvin, [232](#)
  - GetTemperatur, [232](#)
  - GetThermoOffset, [232](#)
  - GetThermoTemp, [233](#)
  - GetThermoVoltage, [233](#)
  - SetNanoVoltsPerKelvin, [233](#)
  - SetThermoOffset, [233](#)
- CMcsBus\_VoltageModeNet, [233](#)
  - ~CMcsBus\_VoltageModeNet, [234](#)
  - CMcsBus\_VoltageModeNet, [234](#)
  - GetVMMMaxNegativeCurrent, [235](#)
  - GetVMMMaxNegativeCurrentEeprom, [235](#)
  - GetVMMMaxNegativeVoltage, [235](#)
  - GetVMMMaxNegativeVoltageEeprom, [235](#)
  - GetVMMMaxPositiveCurrent, [235](#)
  - GetVMMMaxPositiveCurrentEeprom, [235](#)
  - GetVMMMaxPositiveVoltage, [235](#)
  - GetVMMMaxPositiveVoltageEeprom, [236](#)
  - GetVMOOutputOnOff, [236](#)
  - GetVMVoltage, [236](#)
  - SetVMMMaxNegativeCurrent, [236](#)
  - SetVMMMaxNegativeCurrentEeprom, [236](#)
  - SetVMMMaxNegativeVoltage, [236](#)
  - SetVMMMaxNegativeVoltageEeprom, [236](#)
  - SetVMMMaxPositiveCurrent, [237](#)
  - SetVMMMaxPositiveCurrentEeprom, [237](#)
  - SetVMMMaxPositiveVoltage, [237](#)
  - SetVMMMaxPositiveVoltageEeprom, [237](#)
  - SetVMOOutputOnOff, [237](#)
  - SetVMVoltage, [237](#)
- CMcsBusNet, [238](#)
  - ~CMcsBusNet, [238](#)
  - CMcsBusNet, [238](#)
  - CMcsBusNet::GetMode, [239](#)
  - CMcsBusNet::GetModeEeprom, [239](#)
  - CMcsBusNet::SetMode, [239](#)
  - CMcsBusNet::SetModeEeprom, [239](#)
  - GetBusAddress, [239](#)
  - GetBusAddressEeprom, [239](#)
  - GetCommand, [239](#), [240](#)
  - GetHWRRevisionEeprom, [240](#)
  - SetBusAddress, [240](#)
  - SetBusAddressEeprom, [240](#)
  - SetCommand, [241](#)
  - SetHWRRevisionEeprom, [241](#)
- CMcsBusNet::GetMode
  - CMcsBusNet, [239](#)
- CMcsBusNet::GetModeEeprom
  - CMcsBusNet, [239](#)
- CMcsBusNet::SetMode
  - CMcsBusNet, [239](#)
- CMcsBusNet::SetModeEeprom
  - CMcsBusNet, [239](#)
- CMcsUsbDacqNet, [242](#)
  - ~CMcsUsbDacqNet, [248](#)
  - AddSelectedChannelsQueue, [248](#), [249](#), [251](#)
  - ChannelBlock\_AvailFrames, [251](#)
  - ChannelBlock\_ReadAsFrameArrayI16, [251](#), [252](#)
  - ChannelBlock\_ReadAsFrameArrayI32, [253](#)
  - ChannelBlock\_ReadAsFrameArrayUI16, [254](#), [255](#)
  - ChannelBlock\_ReadAsFrameArrayUI32, [255](#), [256](#)
  - ChannelBlock\_ReadFramesDictI16, [257](#)
  - ChannelBlock\_ReadFramesDictI32, [257](#)
  - ChannelBlock\_ReadFramesDictUI16, [258](#)
  - ChannelBlock\_ReadFramesDictUI32, [259](#)
  - ChannelBlock\_ReadFramesI16, [259](#), [260](#)
  - ChannelBlock\_ReadFramesI32, [260](#), [261](#)
  - ChannelBlock\_ReadFramesUI16, [262](#)
  - ChannelBlock\_ReadFramesUI32, [263](#), [264](#)
  - ChannelDataEvent, [289](#)
  - ClearBuffers, [264](#)
  - CMcsUsbDacqNet, [248](#)
  - CMcsUsbDacqNet::GetFilterProperties, [264](#)
  - Error\_Callback\_Aquisition\_Stopped, [288](#)
  - Error\_Callback\_Data\_lost, [288](#)
  - Error\_Callback\_Frames\_Lost, [288](#)
  - Error\_Callback\_Packet\_Error, [288](#)
  - Error\_Callback\_Queue\_Full, [289](#)

- Error\_Callback\_RingQueue\_Full, 289
- ErrorEvent, 289
- GetAdapterType, 264
- GetAdcDataFormat, 264
- GetAdcZero, 265
- GetAnalogValueUnit, 265
- GetChannelDataFillSize, 265
- GetChannelLayout, 265
- GetChannelsInBlock, 265
- GetDataFormat, 265
- GetDataMode, 265
- GetDigitalSource, 266, 267
- GetErrorMessage, 268
- GetFilterProperty, 268
- GetGroupChannelDataI16, 268
- GetGroupChannelDataI32, 268
- GetGroupChannelDataUI16, 269
- GetGroupChannelDataUI32, 269
- GetHardwareMaxRange, 270
- GetHardwareMinRange, 270
- GetMaxSamplingFrequency, 270
- GetMeaLayout, 270
- GetMinSamplingFrequencyStepsize, 271
- GetNumberOfDataBits, 271
- GetPoti, 271
- GetResolutionPerDigit, 271
- GetSamplerate, 271
- GetVoltageRangeIndex, 271
- GetVoltageRangeInMicroVolt, 272
- GetVoltageRangeInMilliVolt, 272
- HWInfo, 272
- Samplerate, 289
- SendStartDacq, 272
- SendStartStgAndDacq, 272
- SendStopDacq, 273
- SendStopStgAndDacq, 273
- SendStopStgAndDacqWithOptions, 273
- SetDataMode, 274
- SetDigitalSource, 274–276
- SetPoti, 276
- SetSamplerate, 276
- SetSelectedChannels, 277–279
- SetSelectedChannelsQueue, 279–281
- SetSelectedData, 281–283
- SetupGroupDacqQueue, 283
- SetVoltageRangeByIndex, 283
- SetVoltageRangeInMicroVolt, 284
- StartDacq, 284, 285
- StartLoop, 286, 287
- StopDacq, 288
- StopLoop, 288
- CMcsUsbDacqNet::CHWInfo, 181
  - CHWInfo, 182
  - GetAvailableSampleRates, 182
  - GetAvailableVoltageRangesInMicroVolt, 182
  - GetAvailableVoltageRangesInMicroVoltAndStringsInMilliVolt, 183
  - GetNumberOfHWADCCChannels, 183
  - GetNumberOfHWDigitalChannels, 183
  - IsDigitalChannelDedicated, 184
- CMcsUsbDacqNet::CVoltageRangeInfoNet, 605
  - CVoltageRangeInfoNet, 606
  - VoltageRangeDisplayStringMilliVolt, 606
  - VoltageRangeInMicroVolt, 606
- CMcsUsbDacqNet::GetFilterProperties
  - CMcsUsbDacqNet, 264
- CMcsUsbDeviceStatePushFunctionNet, 289
  - CMcsUsbDeviceStatePushFunctionNet, 290
  - McsUsbDeviceStateEvent, 290
  - TriggerStatus, 290
- CMcsUsbDeviceStatePushNet, 290
  - CMcsUsbDeviceStatePushNet, 291
  - McsUsbDeviceStateEvent, 291
  - TriggerStatus, 291
- CMcsUsbFactoryNet, 291
  - ~CMcsUsbFactoryNet, 293
  - ChangeSerialNumber, 293
  - CMcsUsbFactoryNet, 293
  - Coldstart, 293
  - CompareFirmware, 294
  - DownloadFirmware, 294
  - FindFirmwareVersionMagicInBuffer, 294
  - FX3MCSDDataAddress, 299
  - FX3MCSDDataDeviceldOffset, 299
  - FX3MCSDDataIFB1ImageOffset, 299
  - FX3MCSDDataIFB2ImageOffset, 299
  - FX3MCSDDataVersionOffset, 299
  - GetChecksumFromFX3Image, 294
  - GetDestination, 294
  - GetDestinationDisplayLabel, 294
  - GetDestinationName, 294, 295
  - GetDestinationSerialNumber, 295
  - GetDestinationTargetAddress, 295
  - GetFirmwareVersionFromFile, 295
  - GetFirmwareVersionFromHexFile, 295
  - GetNumDestinations, 295
  - GetUSBDeviceIDFromFX3Image, 296
  - GetUserCodeFromBitFile, 296
  - GetUserCodeFromFlash, 296
  - GetXilinxFlashOffset, 296
  - GetXilinxFlashReadCommand, 296
  - LoadUserFirmware, 296, 297
  - ReadBlockFromFlash, 297
  - ReadBlockFromIFBGlobalEEPROM, 297
  - ReadBlockFromNVMEM, 297
  - SetDestinationSerialNumber, 297
  - UpdateFirmware, 297–299
- CMcsUsbFunctionNet, 300
  - !CMcsUsbFunctionNet, 301
  - ~CMcsUsbFunctionNet, 301
  - CMcsUsbFunctionNet, 300, 301
  - m\_pMcsUsb, 301
  - m\_pMcsUsbFunction, 301
  - ThrowCUsbExceptionNetOnError, 301
- CMcsUsbFunctionPointerContainer, 301



- CMcsUsbListEntryNet, 301
  - ~CMcsUsbListEntryNet, 302
  - DeviceId, 306
  - DeviceName, 306
  - Equals, 303
  - GetEntry, 304
  - GetEntryCount, 305
  - HwVersion, 306
  - Manufacturer, 306
  - Product, 306
  - SerialNumber, 306
  - SetStringFormat, 305
  - ToString, 306
- CMcsUsbListNet, 307
  - !CMcsUsbListNet, 308
  - ~CMcsUsbListNet, 308
  - CMcsUsbListNet, 307
  - Count, 309
  - DeviceArrival, 309
  - DeviceRemoval, 309
  - GetNumberOfDevices, 308
  - GetUsbListEntries, 308
  - GetUsbListEntry, 308
  - IsDeviceTypeOf, 309
  - SetStringFormat, 309
- CMcsUsbNet, 310
  - !CMcsUsbNet, 314
  - ~CMcsUsbNet, 314
  - AddSoftwareKey, 314
  - AssociateToThis, 314
  - CMcsUsbNet, 314
  - Connect, 314, 315
  - CyclePort, 316
  - Disconnect, 316
  - EmptyKey, 316
  - EnableExceptions, 316
  - EraseEepromRegisterPreconfig, 316
  - GetConfiguration, 317
  - GetDeviceCannotStallOutRequests, 317
  - GetDeviceCapableSpeed, 317
  - GetDeviceEnum, 317
  - GetDeviceId, 317
  - GetDeviceRootHubVendorEnum, 317
  - GetDeviceRootHubVendorID, 317
  - GetDeviceRootHubVendorName, 317
  - GetDeviceSpeed, 318
  - GetErrorText, 318
  - GetFirmwareVersion, 318
  - GetHardwareRevision, 318
  - GetHeadstageActive, 319
  - GetHeadstageID, 319
  - GetHeadstagePresent, 319
  - GetLastUSBError, 319
  - GetMea21UsbPort, 320
  - GetNumConfigurations, 320
  - GetSerialNumber, 320
  - GetSoftwareKey, 320
  - GetSoftwareKeyString, 320
  - GetStatus, 320
  - GetStatusOfLastCommand, 320
  - GetUsbListEntry, 321
  - GetVersion, 321
  - HasSoftwareKey, 321
  - IsConnected, 321
  - IsDeviceHighSpeed, 321
  - IsDeviceHighSpeedCapable, 321
  - IsExceptionsEnabled, 322
  - MultibootGetCypressImageId, 322
  - MultibootGetImageId, 322
  - MultibootGetSelectedImage, 322
  - MultibootSelectImage, 322
  - ReadEepromRegisterPreconfig, 322, 323
  - ReadRegister, 323
  - ReadRegister32, 323
  - ReadRegisterTimeSlot, 323
  - RemoveSoftwareKey, 323
  - RescanHeadstage, 323
  - SerialNumber, 330
  - SetConfiguration, 324
  - SetSoftwareKey, 324
  - Status\_AlreadyConfigured, 326
  - Status\_BadStartFrame, 326
  - Status\_Btstuff, 326
  - Status\_BufferOverrun, 326
  - Status\_BufferUnderrun, 326
  - Status\_Canceled, 326
  - Status\_Canceling, 327
  - Status\_ConnectedPipes, 327
  - Status\_ControlNotOwned, 327
  - Status\_Crc, 327
  - Status\_DataOverrun, 327
  - Status\_DataToggleMismatch, 327
  - Status\_DataUnderrun, 327
  - Status\_DeviceLocked, 327
  - Status\_DeviceNotFound, 327
  - Status\_DeviceRemoved, 327
  - Status\_DevNotResponding, 327
  - Status\_EndpointHalted, 328
  - Status\_ErrorBusy, 328
  - Status\_ErrorShortTransfer, 328
  - Status\_Fifo, 328
  - Status\_FrameControlOwned, 328
  - Status\_InternalHcError, 328
  - Status\_InvalidDeviceHandle, 328
  - Status\_InvalidHandle, 328
  - Status\_InvalidParameter, 328
  - Status\_InvalidPipeHandle, 328
  - Status\_InvalidUrbFunction, 329
  - Status\_IoPending, 329
  - Status\_IoTimeout, 329
  - Status\_IsochRequestFailed, 329
  - Status\_LastUsbErrorMismatch, 329
  - Status\_NoBandwidth, 329
  - Status\_NoMemory, 329
  - Status\_NoSuchDevice, 329
  - Status\_NotAccessed, 329

- Status\_NotSupported, [329](#)
- Status\_PidCheckFailure, [329](#)
- Status\_PipeNotLinked, [330](#)
- Status\_RequestFailed, [330](#)
- Status\_RequestMutexFailed, [330](#)
- Status\_RequestMutexTimeout, [330](#)
- Status\_Stall, [330](#)
- Status\_Unconfigured, [330](#)
- Status\_UnexpectedPid, [330](#)
- ThrowCUsbExceptionNetOnError, [324](#)
- TxnGetSerialNumber, [324](#)
- TxnSetSerialNumber, [324](#)
- TxnTestMemoryReadAndCheck, [324](#)
- TxnTestMemoryWrite, [324](#)
- ValidKey, [324](#)
- WPAError\_ScanningsIsPending, [330](#)
- WriteEepromRegisterPreconfig, [325](#)
- WriteRegister, [325](#)
- WriteRegister32, [325](#)
- WriteRegisterArray, [325](#)
- WriteRegisterTimeSlot, [326](#)
- WriteRegisterValue, [326](#)
- CMcsUsbPointerContainer, [331](#)
- CMEA2100\_256DacqGroupChannelSelectionNet, [331](#)
  - CMEA2100\_256DacqGroupChannelSelectionNet, [331](#)
- CMEA2100x256FunctionNet, [331](#)
  - !CMEA2100x256FunctionNet, [332](#)
  - ~CMEA2100x256FunctionNet, [332](#)
  - CMEA2100x256FunctionNet, [332](#)
  - GetLayoutConfiguration, [332](#)
  - SetLayoutConfiguration, [332](#)
- CMeaAudioFunctionNet, [333](#)
  - CMeaAudioFunctionNet, [333](#)
  - GetAudioChannels, [334](#)
  - GetNumberOfAudioChannels, [334](#)
  - SetAudioChannels, [335](#)
- CMeaAudioFunctionNet::s\_setaudionet, [702](#)
  - amplification, [702](#)
  - channel, [702](#)
- CMeaCleanDeviceNet, [336](#)
  - !CMeaCleanDeviceNet, [337](#)
  - ~CMeaCleanDeviceNet, [337](#)
  - CMeaCleanDeviceNet, [337](#)
  - GetCycle, [337](#)
  - GetCycles, [337](#)
  - GetMaxVoltage, [337](#)
  - GetMinVoltage, [338](#)
  - GetOutputVoltage, [338](#)
  - GetSlope, [338](#)
  - IsRunning, [338](#)
  - SetCycles, [338](#)
  - SetMaxVoltage, [339](#)
  - SetMinVoltage, [339](#)
  - SetSlope, [339](#)
  - Start, [339](#)
  - Stop, [339](#)
- CMeaCoatDeviceNet, [340](#)
- !CMeaCoatDeviceNet, [341](#)
- ~CMeaCoatDeviceNet, [341](#)
- CMeaCoatDeviceNet, [341](#)
- GetCurrentCycle, [341](#)
- GetCycles, [341](#)
- GetDuration, [342](#)
- GetMaxCurrent, [342](#)
- GetOffsetCurrent, [342](#)
- GetOutputCurrent, [342](#)
- GetPauseDuration, [342](#)
- GetSlope, [343](#)
- GetTimeInPause, [343](#)
- GetTimeInPlateau, [343](#)
- IsRunning, [343](#)
- SetCycles, [343](#)
- SetDuration, [344](#)
- SetMaxCurrent, [344](#)
- SetOffsetCurrent, [344](#)
- SetPauseDuration, [344](#)
- SetSlope, [345](#)
- Start, [345](#)
- Stop, [345](#)
- CMeaDeviceNet, [345](#)
  - ~CMeaDeviceNet, [347](#)
  - AnalogGain, [352](#)
  - CMeaDeviceNet, [347](#)
  - EnableChecksum, [347](#)
  - EnableDigitalIn, [348](#)
  - EnableTimestamp, [349](#)
  - Gain, [352](#)
  - GetAnalogGain, [349](#)
  - GetEnumerationSpeed, [349](#)
  - GetGain, [349](#)
  - MeaAudioFunctionNet, [353](#)
  - MeaDigitalDataFunctionNet, [353](#)
  - MeaFeedbackFunctionNet, [353](#)
  - MeFunctionNet, [353](#)
  - SetDigitalOut, [349](#)
  - SetNumberOfAnalogChannels, [350](#)
  - SetNumberOfChannels, [351](#)
  - SetTriggerMaskValue, [351](#)
  - SetTriggerPeriod, [352](#)
  - W2100\_FunctionNet, [353](#)
  - WClassicFunctionNet, [353](#)
- CMeaDigitalDataFunctionNet, [353](#)
  - CMeaDigitalDataFunctionNet, [354](#)
  - GetDigitalData, [354](#)
  - SetDigitalData, [354](#), [355](#)
- CMeaFeedbackFunctionNet, [355](#)
  - CMeaFeedbackFunctionNet, [356](#)
  - FeedbackGetSampleTimerCount, [356](#)
  - FeedbackSetAnalogSource, [356](#)
  - FeedbackSetChannelFilter, [356](#)
  - FeedbackSetDigitalMapping, [357](#)
  - FeedbackSetFeedback, [357](#)
  - FeedbackSetFilterOff, [357](#)
  - FeedbackSetFilterParameter, [357](#)
  - FeedbackSetFilterParameter32, [357](#)



- FeedbackSetGlobalChannelFilter, [357](#)
- FeedbackSetIIRFilterParameter, [357](#)
- FeedbackSetLogic, [357](#)
- FeedbackSetMkFilter, [358](#)
- FeedbackSetNumberOfLogics, [358](#)
- FeedbackSetNumberOfRateCounter, [358](#)
- FeedbackSetNumberOfRateDetectors, [358](#)
- FeedbackSetNumberOfSpikeDetectors, [358](#)
- FeedbackSetNumberOfTriggers, [358](#)
- FeedbackSetRateCounter, [358](#)
- FeedbackSetRateDetector, [358](#)
- FeedbackSetSpikeDetectorThreshold, [359](#)
- FeedbackSetTrigger, [359](#)
- CMealImpedanceDeviceNet, [359](#)
  - ~CMealImpedanceDeviceNet, [360](#)
- CMealImpedanceDeviceNet, [360](#)
  - GetAdapterCode, [360](#)
  - GetArraySize, [360](#)
  - GetImpedanceTestFrequency, [360](#)
  - GetReady, [360](#)
  - GetResult, [360](#)
  - SetImpedanceTestFrequency, [360](#)
  - StartMeasurement, [360](#)
- CMeasureTableDeviceNet, [361](#)
  - CMeasureTableDeviceNet, [361](#)
  - Sensor, [361](#)
- CMeaSwitchDeviceNet, [362](#)
  - ~CMeaSwitchDeviceNet, [362](#)
- CMeaSwitchDeviceNet, [362](#)
  - GetNumber, [363](#)
  - GetPattern, [363](#)
  - GetPatternBool, [363](#)
  - SetPattern, [363](#)
  - SetPatternBool, [363](#)
- CMeaUSBDeviceNet, [363](#)
  - ~CMeaUSBDeviceNet, [364](#)
- CMeaUSBDeviceNet, [364](#)
- CMeFunctionNet, [365](#)
  - !CMeFunctionNet, [365](#)
  - ~CMeFunctionNet, [365](#)
- CMeFunctionNet, [365](#)
  - SetTransformer, [366](#)
- CMosMea
  - CCMOSMeaDeviceNet, [112](#)
- CmosMea
  - Mcs::Usb, [68](#)
- CMOSMeaBathModeEnumNet
  - Mcs::Usb, [56](#)
- CmosMeaHeadstage
  - Mcs::Usb, [67](#)
- CMOSMeaHeadstage1NCBathCurrentEnumNet
  - Mcs::Usb, [56](#)
- CMOSMeaHeadstage1NCCol2CurrentEnumNet
  - Mcs::Usb, [56](#)
- CMOSMeaHeadstage1NChipTempEnumNet
  - Mcs::Usb, [56](#)
- CMOSMeaHS1SidebandEnumNet
  - Mcs::Usb, [57](#)
- CMOSMeaHS1TriggerStatusEnumNet
  - Mcs::Usb, [57](#)
- CmosmealFB2
  - Mcs::Usb, [67](#)
- CMOSMeaFDigChannelEnumNet
  - Mcs::Usb, [57](#)
- CMOSMeaInterfaceADCEnumNet
  - Mcs::Usb, [57](#)
- CmosMeaInterfaceboard
  - Mcs::Usb, [67](#)
- CMOSMeaPacketFrameContextGroupEnumNet
  - Mcs::Usb, [58](#)
- CMOSMeaSTG1DACSignalEnumNet
  - Mcs::Usb, [58](#)
- CMOSMeaValueUnitEnumNet
  - Mcs::Usb, [58](#)
- CMultiBatteryChargerDeviceNet, [366](#)
  - !CMultiBatteryChargerDeviceNet, [367](#)
  - ~CMultiBatteryChargerDeviceNet, [367](#)
- CapacityTest, [367](#)
- ChannelReset, [368](#)
- CMultiBatteryChargerDeviceNet, [367](#)
  - GetBatteryVoltage, [368](#)
  - GetChannels, [368](#)
  - GetChannelState, [368](#)
  - GetChargeCapacity, [369](#)
  - GetChargeCurrent, [369](#)
  - GetChargingMode, [369](#)
  - GetChargingPCoefficient, [370](#)
  - GetDischargeCapacity, [370](#)
  - GetDischargeCurrent, [370](#)
  - GetDischargeCurrentSetPoint, [370](#)
  - GetFinalDischargeVoltage, [371](#)
  - GetRatedCapacity, [371](#)
  - SetChargingMode, [371](#)
  - SetChargingPCoefficient, [372](#)
  - SetDischargeCurrentSetPoint, [372](#)
  - SetFinalDischargeVoltage, [372](#)
  - SetRatedCapacity, [372](#)
  - SetRatedCapacityVolatile, [374](#)
- CMultiwellCallbackFunctionNet, [374](#)
  - !CMultiwellCallbackFunctionNet, [375](#)
  - ~CMultiwellCallbackFunctionNet, [375](#)
- CMultiwellCallbackFunctionNet, [375](#)
  - GetPlateClampStateByHeadstage, [375](#)
  - GetPlateClampStateByHeadstageEvent, [376](#)
  - OnGetPlateClampStateByHeadstage, [376](#)
- CMultiwellDeviceNet, [376](#)
  - !CMultiwellDeviceNet, [378](#)
  - ~CMultiwellDeviceNet, [378](#)
- ClosePlateClamp, [378](#)
- CMultiwellDeviceNet, [377](#)
  - GetPlateClampLockState, [378](#)
  - GetPlateClampState, [378](#)
  - GetPlateMux, [379](#)
  - GetPlateType, [379](#)
  - GetPowerMuxPlate, [380](#)
  - GetTouchPadEnable, [380](#)

- GetVolatileClampOffset, [380](#)
- IsPlateTypeValid, [381](#)
- LockPlateClamp, [381](#)
- OpenPlateClamp, [381](#)
- SetPlateMux, [381](#), [382](#)
- SetPlateType, [382](#)
- SetPowerMuxPlate, [382](#)
- SetTouchPadEnable, [383](#)
- SetVolatileClampOffset, [383](#)
- StopPlateClamp, [383](#)
- UnlockPlateClamp, [383](#)
- CMultiwellOptoStimFunctionNet, [384](#)
  - !CMultiwellOptoStimFunctionNet, [385](#)
  - ~CMultiwellOptoStimFunctionNet, [385](#)
  - CMultiwellOptoStimFunctionNet, [384](#)
  - GetAbsMaxCurrentInMicroAmp, [385](#)
  - GetColorRgb, [385](#)
  - GetColorStr, [385](#)
  - GetMaxDurationHighCurrentInMicroSec, [386](#)
  - GetMaxDutyCycleHighCurrent, [386](#)
  - GetPermanentCurrentInMicroAmp, [386](#)
  - GetWaveLengthInNanometer, [386](#)
  - SetAbsMaxCurrentInMicroAmp, [387](#)
  - SetColorRgb, [387](#)
  - SetColorStr, [387](#)
  - SetMaxDurationHighCurrentInMicroSec, [387](#)
  - SetMaxDutyCycleHighCurrent, [388](#)
  - SetPermanentCurrentInMicroAmp, [388](#)
  - SetWaveLengthInNanometer, [388](#)
- CNF\_GenDeviceNet, [388](#)
  - ~CNF\_GenDeviceNet, [389](#)
  - CNF\_GenDeviceNet, [389](#)
  - Set\_Values, [389](#)
- COctoPotDeviceNet, [389](#)
  - BurnAdcOffset, [390](#)
  - BurnDacOffset, [391](#)
  - COctoPotDeviceNet, [390](#)
  - EnableChecksum, [391](#)
  - EnableDigitalIn, [391](#)
  - EnableTimestamp, [391](#)
  - GetAdcOffset, [391](#)
  - GetDacOffset, [391](#)
  - PatternListStart, [391](#)
  - RampStart, [391](#)
  - ResetAdcOffset, [391](#)
  - ResetDacOffset, [391](#)
  - SetAdcOffset, [392](#)
  - SetAmplificationSwitch, [392](#)
  - SetBathclamp, [392](#)
  - SetChannelSwitch, [392](#)
  - SetDacAutoControl, [392](#)
  - SetDacOffset, [392](#)
  - SetDacValue, [392](#)
  - SetNumberOfChannels, [392](#)
  - SetOutputRate, [392](#)
  - SetPatternListEntry, [393](#)
  - SetPidParameter, [393](#)
  - SetRampParameter, [393](#)
  - SetSineParameter, [393](#)
  - SineStart, [393](#)
- CokuvisionStimulatorDeviceNet, [393](#)
  - ~CokuvisionStimulatorDeviceNet, [394](#)
  - CokuvisionStimulatorDeviceNet, [394](#)
  - GetCheckVoltage, [394](#)
  - GetCurrentFactor, [394](#)
  - GetDACOffset, [395](#)
  - GetMaxPower, [395](#)
  - GetMaxVoltage, [395](#)
  - GetPulseform, [395](#)
  - GetRTC, [395](#)
  - GetStimulatorStatus, [395](#)
  - GetVoltage, [395](#)
  - SetCheckVoltage, [396](#)
  - SetCurrentFactor, [396](#)
  - SetDACOffset, [396](#)
  - SetMaxPower, [396](#)
  - SetMaxVoltage, [396](#)
  - SetPulseform, [396](#)
  - SetRTC, [396](#)
- Coldstart
  - CMcsUsbFactoryNet, [293](#)
- CommaPositionA
  - CFilterCoefficientsNet::s\_FilterAttributesNet, [702](#)
  - Mcs::Usb, [66](#)
- CommaPositionB
  - CFilterCoefficientsNet::s\_FilterAttributesNet, [702](#)
  - Mcs::Usb, [66](#)
- CompareFirmware
  - CMcsUsbFactoryNet, [294](#)
- CompareTo
  - HeadStageIDType, [687](#)
- CompensateElectrodeOffset
  - CWarnerUssingFunctionNet, [624](#)
- Connect
  - CMcsUsbNet, [314](#), [315](#)
  - CRFFunctionNet, [441](#)
- ConnectDevice
  - CRadioControlledDevicesNet, [436](#)
- ConnectedImp
  - CPositionImpDeviceNet, [413](#)
- ConnectImp
  - CPositionImpDeviceNet, [413](#)
- ConnectSlave
  - CGilsonDeviceNet, [166](#)
- ControlState
  - HeadStageIDTypeState, [690](#)
- CornerFrequency
  - CFilterPropertyNet, [137](#)
- CornerFrequencymHz
  - CFilterPropertyNet, [138](#)
- Count
  - CMcsUsbListNet, [309](#)
- CPatchServerDeviceNet, [397](#)
  - CPatchServerDeviceNet, [397](#)
  - Sensor, [397](#)
- CPathIdentDeviceNet, [398](#)

- ~CPathIdentDeviceNet, 398
- CPathIdentDeviceNet, 398
- Measure, 398
- Set\_Values, 398
- CPedoterDeviceNet, 399
  - !CPedoterDeviceNet, 399
  - ~CPedoterDeviceNet, 399
  - CPedoterDeviceNet, 399
  - GetCommand, 399
  - SetCommand, 400
- CPeristalticPumpDeviceNet, 400
  - ~CPeristalticPumpDeviceNet, 401
  - CPeristalticPumpDeviceNet, 401
  - McsBus\_MotorControl, 401
- CPgaDeviceNet, 401
  - ~CPgaDeviceNet, 402
  - ApplyGains, 402
  - CPgaDeviceNet, 402
  - DefineAmplification, 402
  - DefineFrequencyRange, 402
  - DefineNumAmplifications, 402
  - DefineNumFrequencyRanges, 402
  - GetAmplification, 402
  - GetFrequencyRange, 403
  - GetGain, 403
  - GetNumAmplifications, 403
  - GetNumFrequencyRanges, 403
  - SetGain, 403
- CPositionIIDeviceNet, 403
  - !CPositionIIDeviceNet, 405
  - ~CPositionIIDeviceNet, 405
  - CPositionIIDeviceNet, 405
  - GetCoilCommunication, 405
  - GetDebugData, 405
  - GetEventData, 406
  - GetImplantCurrentSetpoint, 406
  - GetImplantResult, 407
  - GetImplantState, 407
  - GetOnOff, 407
  - GetPowerStrength, 408
  - GetRTC, 408
  - GetStateDebugData, 408
  - GetStateEventData, 409
  - RFFunction, 412
  - SetImplantCurrentSetpoint, 409
  - SetPowerStrength, 409
  - SetRTC, 409
  - SetStateDebugData, 411
  - SetStateEventData, 411
  - SwitchOnOff, 411
- CPositionImpDeviceNet, 412
  - !CPositionImpDeviceNet, 413
  - ~CPositionImpDeviceNet, 413
  - ConnectedImp, 413
  - ConnectImp, 413
  - CPositionImpDeviceNet, 413
  - GetDeviceList, 413
  - GetImpld, 414
  - GetRFFrequency, 414
  - SetDeviceList, 414
  - SetImpld, 414
  - SetRFFrequency, 415
- CPPCDeviceNet, 415
  - CPPCDeviceNet, 415
  - McsBus, 416
  - McsBus\_MotorControl, 416
  - McsBus\_Sensor, 416
  - PPCFunction, 416
- CPPCFunctionNet, 416
  - !CPPCFunctionNet, 418
  - ~CPPCFunctionNet, 418
  - CPPCFunctionNet, 417
  - FirePressurePulse, 418
  - GetAnalogVoltage, 418
  - GetAnalogVoltageRange, 418
  - GetDigitalIn, 420
  - GetPressureRange, 420
  - GetPumpModeType, 420
  - GetPumpSpeedUnit, 421
  - GetSupplyVoltage, 421
  - GetValveActive, 421
  - IsBusy, 421
  - LoadPressure, 423
  - MeasureReservoir, 423
  - SetAnalogVoltageRange, 423
  - SetPressureOffset, 423
  - SetPressureRange, 423
  - SetPumpModeType, 424
  - SetPumpSpeedUnit, 424
  - SetValveActive, 424
- CPPS\_DeviceNet, 425
  - CPPS\_DeviceNet, 425
  - McsBus, 425
  - McsBus\_MotorControl, 425
  - McsBus\_Sensor, 425
  - PPS\_Function, 425
- CPPS\_FunctionNet, 426
  - CPPS\_FunctionNet, 426, 427
  - GetAnalogVoltage, 427
  - GetAnalogVoltages, 427
  - GetBubbleState, 427
  - GetDigitalIn, 427
  - GetPumpCouple, 427
  - GetPumpEnableSpeedRatio, 427
  - GetPumpFastOnOff, 427
  - GetPumpFastSpeed, 427
  - GetPumpFunctionSpeeds, 428
  - GetPumpManualOnOff, 428
  - GetPumpMaxSpeed, 428
  - GetPumpModeType, 428
  - GetPumpSpeedRatio, 428
  - GetPumpSpeedUnit, 428
  - GetSupplyVoltage, 428
  - GetUseBubble, 428
  - SetAnalogVoltages, 428
  - SetPumpCouple, 429

- SetPumpEnableSpeedRatio, [429](#)
- SetPumpFastOnOff, [429](#)
- SetPumpFastSpeed, [429](#)
- SetPumpFunctionSpeeds, [429](#)
- SetPumpManualOnOff, [429](#)
- SetPumpMaxSpeed, [429](#)
- SetPumpModeType, [429](#)
- SetPumpSpeedRatio, [430](#)
- SetPumpSpeedUnit, [430](#)
- SetUseBubble, [430](#)
- CPPSDeviceNet, [430](#)
  - CPPSDeviceNet, [431](#)
- CProgramPressureCurveNet, [431](#)
  - ICProgramPressureCurveNet, [432](#)
  - ~CProgramPressureCurveNet, [431](#)
  - CProgramPressureCurveNet, [431](#)
  - GetRepeats, [432](#)
  - Program, [432](#)
  - SetRepeats, [432](#)
- CPulseGeneratorFunctionNet, [432](#)
  - ICPulseGeneratorFunctionNet, [433](#)
  - ~CPulseGeneratorFunctionNet, [433](#)
  - CPulseGeneratorFunctionNet, [433](#)
  - GetModeSelect, [434](#)
  - GetPeriod, [434](#)
  - GetPulseLength, [434](#)
  - SetModeSelect, [435](#)
  - SetPeriod, [435](#)
  - SetPulseLength, [435](#)
- CRadioControlledDevicesNet, [435](#)
  - ConnectDevice, [436](#)
  - CRadioControlledDevicesNet, [436](#)
  - DisconnectDevice, [436](#)
  - GetDeviceNames, [436](#)
  - GetFrequency, [437](#)
  - HasRadioControl, [437](#)
  - SetFrequency, [437](#)
  - StillConnected, [437](#)
- CreateSideband
  - CStimulusFunctionNet, [568](#)
- CreateWirelessHeadstageSerialNumberString
  - CWirelessBaseFunctionNet, [672](#)
- CRegionOfInterestRect
  - CCMOSMeaDeviceNet::CRegionOfInterestRect, [437](#)
- CRetinaLedDeviceNet, [438](#)
  - ~CRetinaLedDeviceNet, [439](#)
  - AddLoopEntry, [439](#)
  - AddTableEntry, [439](#)
  - ClearTable, [439](#)
  - CRetinaLedDeviceNet, [439](#)
  - GetTablepointer, [439](#)
  - SetLED, [439](#)
  - SetLumi, [439](#)
  - SetPersistency, [439](#)
  - SetRepeat, [440](#)
  - SetTablepointer, [440](#)
  - SetTrigger, [440](#)
- CRFFFunctionNet, [440](#)
  - ICRFFFunctionNet, [441](#)
  - ~CRFFFunctionNet, [441](#)
  - Connect, [441](#)
  - CRFFFunctionNet, [441](#)
  - GetAvailableDeviceList, [442](#)
  - GetAvailableDeviceListEx, [442](#)
  - GetAvailableStateList, [442](#)
  - GetAvailableStateListEx, [442](#)
  - GetBaseFrequency, [443](#)
  - GetConnectedDevice, [443](#)
  - GetState, [443](#)
  - GetTestMode, [443](#)
  - GetWorkingFrequency, [444](#)
  - SetBaseFrequency, [444](#)
  - SetTestMode, [444](#)
  - SetWorkingFrequency, [444](#)
- CRobo\_FYIPProgram\_FunctionNet, [445](#)
  - CRobo\_FYIPProgram\_FunctionNet, [445](#)
  - GetLength, [445](#)
  - GetState, [446](#)
  - GetValve1, [446](#)
  - GetValve2, [446](#)
  - SetLength, [446](#)
  - SetValve1, [446](#)
  - SetValve2, [446](#)
  - Start, [446](#)
- CRobo\_FYITemp\_FunctionNet, [446](#)
  - CRobo\_FYITemp\_FunctionNet, [447](#)
  - GetICoeff, [447](#)
  - GetMaxPower, [447](#)
  - GetPCoeff, [447](#)
  - GetRegulatorOnOff, [447](#)
  - GetSollTemp, [447](#)
  - SetICoeff, [448](#)
  - SetMaxPower, [448](#)
  - SetPCoeff, [448](#)
  - SetRegulatorOnOff, [448](#)
  - SetSollTemp, [448](#)
- CRoboDacqNet, [448](#)
  - CancelTableLoop, [451](#)
  - CancelTableLoopAndStopTable, [451](#)
  - ClampAmpRestart, [451](#)
  - CRoboDacqNet, [451](#)
  - DoRamp, [451](#)
  - Emu\_GetCellCapacity, [452](#)
  - Emu\_GetCellPotential, [452](#)
  - Emu\_GetCellResists, [452](#)
  - Emu\_GetElectrodeResists, [452](#)
  - Emu\_GetNoise, [452](#)
  - Emu\_SetCellCapacity, [452](#)
  - Emu\_SetCellPotential, [452](#)
  - Emu\_SetCellResists, [452](#)
  - Emu\_SetElectrodeResists, [452](#)
  - Emu\_SetNoise, [452](#)
  - GetAllDigout, [453](#)
  - GetCapacityC, [453](#)
  - GetCapacityV, [453](#)

GetCapacityX, [453](#)  
GetClampAmpSerialNumber, [453](#)  
GetCommand, [453](#)  
GetConfigurationBit, [453](#)  
GetConfigurationBitAxc, [453](#)  
GetConfigurationBitBlu\_Led, [453](#)  
GetConfigurationBitBlu\_LedToggleFast, [453](#)  
GetConfigurationBitBlu\_LedToggleSlow, [453](#)  
GetConfigurationBitCC\_Gen, [454](#)  
GetConfigurationBitCV\_Gen, [454](#)  
GetConfigurationBitRC\_Gen, [454](#)  
GetConfigurationBitRed\_Led, [454](#)  
GetConfigurationBitRed\_LedSaturation, [454](#)  
GetConfigurationBitRed\_LedToggleFast, [454](#)  
GetConfigurationBitRed\_LedToggleSlow, [454](#)  
GetConfigurationBitRelais, [454](#)  
GetConfigurationBitRV\_Gen, [454](#)  
GetConfigurationBits, [454](#)  
GetConfigurationBitStream, [454](#)  
GetConfigurationBitSupply, [455](#)  
GetCrossTalkOffset, [455](#)  
GetCrossTalkOptimum, [455](#)  
GetDigout, [455](#)  
GetDisplayText, [455](#)  
GetDownsampleFactor, [455](#)  
GetFilter, [455](#)  
GetFilterCoeffs, [455](#)  
GetIC, [455](#)  
GetIClamp, [455](#)  
GetICOffset, [455](#)  
GetIGain, [456](#)  
GetNIC\_MS, [456](#)  
GetNUC\_MS, [456](#)  
GetNUV\_MS, [456](#)  
GetPGain, [456](#)  
GetRecordingNumber, [456](#)  
GetResistanceC, [456](#)  
GetResistanceV, [456](#)  
GetScreen, [456](#)  
GetSimulation, [456](#)  
GetUC, [456](#)  
GetUClamp, [457](#)  
GetUCOffset, [457](#)  
GetUpdateDisplay, [457](#)  
GetUV, [457](#)  
GetUVOffset, [457](#)  
GetXGain, [457](#)  
RunTable, [457](#)  
SetAllDigout, [457](#)  
SetCommand, [457](#)  
SetConfigurationBit, [457](#)  
SetConfigurationBitAxc, [458](#)  
SetConfigurationBitBlu\_Led, [458](#)  
SetConfigurationBitBlu\_LedToggleFast, [458](#)  
SetConfigurationBitBlu\_LedToggleSlow, [458](#)  
SetConfigurationBitCC\_Gen, [458](#)  
SetConfigurationBitCV\_Gen, [458](#)  
SetConfigurationBitRC\_Gen, [458](#)  
SetConfigurationBitRed\_Led, [458](#)  
SetConfigurationBitRed\_LedSaturation, [458](#)  
SetConfigurationBitRed\_LedToggleFast, [459](#)  
SetConfigurationBitRed\_LedToggleSlow, [459](#)  
SetConfigurationBitRelais, [459](#)  
SetConfigurationBitRV\_Gen, [459](#)  
SetConfigurationBitStream, [459](#)  
SetConfigurationBitSupply, [459](#)  
SetCrossTalkOffset, [459](#)  
SetCrossTalkOptimum, [459](#)  
SetDigout, [459](#)  
SetDisplayText, [460](#)  
SetDownsampleFactor, [460](#)  
SetFilter, [460](#)  
SetFilterCoeffs, [460](#)  
SetIClamp, [460](#)  
SetICOffset, [460](#)  
SetIGain, [460](#)  
SetNoFilterCoeffs, [460](#)  
SetPGain, [460](#)  
SetRecordingNumber, [461](#)  
SetScreen, [461](#)  
SetSimulation, [461](#)  
SetTriggerMaskValue, [461](#)  
SetUClamp, [461](#)  
SetUCOffset, [461](#)  
SetUVOffset, [461](#)  
SetXGain, [461](#)  
StopTable, [461](#), [462](#)  
Table\_Wait, [462](#)  
TableDefBegin, [462](#)  
TableDefEnd, [462](#)  
TriggerMask\_Default, [462](#)  
TriggerValue\_MoveAbs, [462](#)  
TriggerValue\_StartQueue, [462](#)  
UpdateDisplay, [462](#)  
VirtualDevice\_ContinuousDacq, [462](#)  
VirtualDevice\_TableRun, [462](#)  
CRoboDeviceNet, [463](#)  
    ~CRoboDeviceNet, [466](#)  
    Axes\_I, [474](#)  
    Axes\_X, [474](#)  
    Axes\_Y, [474](#)  
    Axes\_Z, [475](#)  
    Axis\_I, [475](#)  
    Axis\_X, [475](#)  
    Axis\_Y, [475](#)  
    Axis\_Z, [475](#)  
    CancelPoolLoop, [466](#)  
    CancelPoolLoopAndStopMovement, [466](#)  
    CRoboDeviceNet, [466](#)  
    EnableQueue, [467](#)  
    FindReference, [467](#)  
    GetAirpressure, [467](#)  
    GetAirpressureLimit, [467](#)  
    GetAirValve, [467](#)  
    GetCurrentAirvalve, [467](#)  
    GetCurrentAirvalveLimit, [467](#)

- GetCurrentPosition, [468](#)
- GetErrorAirpressure, [468](#)
- GetErrorCurrentAirvalve, [468](#)
- GetErrorVoltage12V, [468](#)
- GetErrorVoltage5V, [468](#)
- GetErrorVoltageAirvalve, [468](#)
- GetErrorVoltageRs485A, [468](#)
- GetErrorVoltageRs485B, [468](#)
- GetErrorVoltageValves, [469](#)
- GetInMovement, [469](#)
- GetMinPressure, [469](#)
- GetMovementError, [469](#)
- GetVoltage12V, [469](#)
- GetVoltage12VLimit, [469](#)
- GetVoltage5V, [469](#)
- GetVoltage5VLimit, [469](#)
- GetVoltageAirvalve, [469](#)
- GetVoltageAirvalveLimit, [469](#)
- GetVoltageRs485A, [470](#)
- GetVoltageRs485ALimit, [470](#)
- GetVoltageRs485B, [470](#)
- GetVoltageRs485BLimit, [470](#)
- GetVoltageValves, [470](#)
- GetVoltageValvesLimit, [470](#)
- IsQueueEnabled, [470](#)
- IsQueueStarted, [470](#)
- McsBus, [478](#)
- McsBus\_MotorControl, [478](#)
- McsBus\_XY, [475](#)
- McsBus\_ZI, [475](#)
- MoveAbs, [470](#), [471](#)
- NullCommand, [472](#)
- RoboError\_AnotherMaster, [475](#)
- RoboError\_Base, [476](#)
- RoboError\_CannotEscapeEndSwitch, [476](#)
- RoboError\_CommandAlreadyInProgress, [476](#)
- RoboError\_CommandNotPossible, [476](#)
- RoboError\_CommunicationTimeout, [476](#)
- RoboError\_DacqNotReady, [476](#)
- RoboError\_DLLMovementTimeout, [476](#)
- RoboError\_FindReferenceMethod, [476](#)
- RoboError\_GilsonCommandPending, [476](#)
- RoboError\_GilsonTimeout, [476](#)
- RoboError\_GilsonWronID, [477](#)
- RoboError\_McsBus\_UnknownCommand, [477](#)
- RoboError\_NoEndSwitch, [477](#)
- RoboError\_NoMoreData, [477](#)
- RoboError\_NoReference, [477](#)
- RoboError\_NoSpeedOrAcceleration, [477](#)
- RoboError\_OverPressure, [477](#)
- RoboError\_ParameterNotAllowed, [477](#)
- RoboError\_PeristalticTimeout, [477](#)
- RoboError\_Phase0OutOfRange, [477](#)
- RoboError\_PollLoopCanceled, [478](#)
- RoboError\_PollLoopCanceledAndStopMovement, [478](#)
- RoboError\_Pressure, [478](#)
- RoboError\_RangeExceeded, [478](#)
- RoboError\_StateChangeNotPossible, [478](#)
- RoboError\_Timeout, [478](#)
- RoboError\_UnknownCommand, [478](#)
- RoboMainLowLevelCommand, [479](#)
- RoboStatusEvent, [479](#)
- SetAirpressureLimit, [472](#)
- SetAirValve, [472](#)
- SetCurrentAirvalveLimit, [472](#)
- SetCurrentAndAir, [472](#)
- SetInMovement, [472](#)
- SetMinPressure, [473](#)
- SetVoltage12VLimit, [473](#)
- SetVoltage5VLimit, [473](#)
- SetVoltageAirvalveLimit, [473](#)
- SetVoltageRs485ALimit, [473](#)
- SetVoltageRs485BLimit, [473](#)
- SetVoltageValvesLimit, [473](#)
- StartQueue, [473](#)
- StopMovement, [474](#)
- WaitTimer, [474](#)
- CRoboDeviceNet::RoboMainLowLevelCommands, [694](#)
  - FindReferencePhase0, [695](#)
  - GetAxisConfig, [695](#)
  - GetHWConfig, [695](#)
  - GetHWRevision, [695](#)
  - GetMaxNoPressure, [695](#)
  - GetMaxNoPressureWaitTime, [695](#)
  - GetMaxPressureWaitTime, [695](#)
  - GetMinNoPressureWaitTime, [695](#)
  - GetMinPressure, [695](#)
  - GetMinPressureWaitTime, [696](#)
  - GetParameter, [696](#)
  - GetPhases, [696](#)
  - GetSearchReferenceFastAccel, [696](#)
  - GetSearchReferenceFastSpeed, [696](#)
  - GetSearchReferenceFineAccel, [696](#)
  - GetSearchReferenceFineSpeed, [696](#)
  - GetSearchReferenceMethod, [697](#)
  - GetSearchReferenceMoveOut, [697](#)
  - GetSearchReferenceOffsetPos, [697](#)
  - GetUserParameter, [697](#)
  - HasRef, [698](#)
  - SetAxisConfig, [698](#)
  - SetHWConfig, [698](#)
  - SetHWRevision, [698](#)
  - SetMaxNoPressure, [698](#)
  - SetMaxNoPressureWaitTime, [698](#)
  - SetMaxPressureWaitTime, [698](#)
  - SetMinNoPressureWaitTime, [698](#)
  - SetMinPressure, [698](#)
  - SetMinPressureWaitTime, [698](#)
  - SetParameter, [699](#)
  - SetSearchReferenceFastAccel, [699](#)
  - SetSearchReferenceFastSpeed, [699](#)
  - SetSearchReferenceFineAccel, [699](#)
  - SetSearchReferenceFineSpeed, [699](#)
  - SetSearchReferenceMethod, [699](#)
  - SetSearchReferenceMoveOut, [700](#)



- SetSearchReferenceOffsetPos, 700
- SetUserParameter, 700
- CRoboFluidDeviceNet, 479
  - ~CRoboFluidDeviceNet, 480
  - CloseAllValves, 480
  - CRoboFluidDeviceNet, 480
  - GetPumpSpeed, 480
  - GetSingleValve, 480
  - GetValve, 480
  - IsPumpMotorOn, 481
  - m\_pMcsBus\_MotorControlNet, 482
  - m\_pRoboFluidDevice, 482
  - McsBus\_MotorControl, 482
  - PumpOff, 481
  - PumpOn, 481
  - SetPumpSpeed, 481
  - SetSingleValve, 481
  - SetValve, 481
- CRoboInjectDeviceNet, 482
  - CRoboInjectDeviceNet, 483
- CRoboocyte2DeviceNet, 483
  - CRoboocyte2DeviceNet, 484
  - GetAxisLED, 484
  - GetGilsonDevice, 484
  - GetMcsBus\_Extension, 484
  - GetRoboDacq, 484
  - GetRoboFluidDevice, 484
  - SetAxisLED, 484
- CRoboStatorDeviceNet, 484
  - CRoboStatorDeviceNet, 486
  - FindReferenceI, 486
  - FindReferenceXY, 486
  - FindReferenceZ, 486
  - GetCurrentPositionI, 486
  - GetCurrentPositionXY, 486
  - GetCurrentPositionZ, 487
  - HasRefI, 487
  - HasRefXY, 487
  - HasRefZ, 487
  - MoveAbsI, 487
  - MoveAbsXY, 487
  - MoveAbsZ, 487, 488
  - RoboMainStatorLowLevelCommand, 490
  - SetAccelerationI, 488
  - SetAccelerationNativeI, 488
  - SetAccelerationNativeXY, 488
  - SetAccelerationNativeZ, 488
  - SetAccelerationXY, 488
  - SetAccelerationZ, 488
  - SetCurrentAndAirXY, 488
  - SetSpeedI, 489
  - SetSpeedNativeI, 489
  - SetSpeedNativeXY, 489
  - SetSpeedNativeZ, 489
  - SetSpeedXY, 489
  - SetSpeedZ, 489
  - SetVelocityI, 489
  - SetVelocityXY, 489
  - SetVelocityZ, 489
  - StopMovementI, 490
  - StopMovementXY, 490
  - StopMovementZ, 490
- CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands, 701
  - FindReferencePhase0XY, 701
- CSafeISDeviceNet, 490
  - ~CSafeISDeviceNet, 491
  - CSafeISDeviceNet, 491
  - DacqDevice, 493
  - FluidControlDevice, 493
  - RoboDevice, 493
  - SetAdcChannels, 492
  - SetAdcSamplePos, 492
  - SetDacMode, 492
  - SetDacPeriode, 492
  - SetDacPulseform, 492
  - SetSwitches, 493
- csCapacityTestDischarge
  - Mcs::Usb, 70
- csCapacityTestPrecharge
  - Mcs::Usb, 70
- csCharge
  - Mcs::Usb, 70
- CSCUDacqGroupChannelSelectionNet, 493
  - CSCUDacqGroupChannelSelectionNet, 494
- CSCUFunctionNet, 494
  - !CSCUFunctionNet, 497
  - ~CSCUFunctionNet, 497
  - AutomaticAnalogOut, 497
  - CSCUFunctionNet, 497
  - EnableAnalogOut, 497
  - GetAnalogOutADCRange, 498
  - GetAnalogOutChannels, 498
  - GetAnalogOutDACRange, 498
  - GetAvailableHeadstages, 498
  - GetAvailableHeadstagesEvent, 511
  - GetFilterProperties, 498
  - GetFilterProperty, 499
  - GetHeadstageAdcBits, 499
  - GetHeadstageAdcRangeInMicroVolt, 499
  - GetHeadstageDacBits, 500
  - GetHeadstageDacCurrentRangeInMicroAmpere, 500
  - GetHeadstageDacCurrentResolutionInNanoAmpere, 500
  - GetHeadstageDacVoltageRangeInMilliVolt, 501
  - GetHeadstageDacVoltageResolutionInMicroVolt, 501
  - GetHeadstageFrameCyclesToCompare, 501
  - GetHeadstageGainInPermille, 502
  - GetHeadstageID, 502
  - GetHeadstageLinkSpeed, 502
  - GetHeadstageNumberOfAnalogChannels, 503
  - GetHeadstageNumberOfStimulationChannels, 503
  - GetHeadstagePowerStateAtStart, 503
  - GetHeadstageSamplerate, 504

- GetHeadstageSerialNumber, 504
- GetMaxNumberOfHeadstages, 504
- GetMaxStimulusChannelsPerHeadstage, 504
- GetReferenceElectrodeMode, 505
- GetReferenceElectrodeSwitchState, 505
- HasAnalogOut, 505
- HasGalvanicIsolation, 505
- HasHSPowerSwitch, 506
- IsAnalogOutEnabled, 506
- IsAutomaticAnalogOut, 506
- IsHeadstageAvailable, 506
- IsHeadstageAvailableEvent, 511
- IsHSPowered, 507
- IsInDacqLegacyMode, 507
- OnGetAvailableHeadstages, 507
- OnIsHeadstageAvailable, 507
- PowerHS, 507
- SetAnalogOutADCRange, 508
- SetAnalogOutChannels, 508
- SetAnalogOutDACRange, 508
- SetDacqLegacyMode, 508
- SetHeadstageFrameCyclesToComparePermanent, 509
- SetHeadstageLinkSpeedPermanent, 509
- SetHeadstageNumberOfAnalogChannelsPermanent, 509
- SetHeadstagePowerStateAtStart, 509
- SetHeadstageSampleratePermanent, 510
- SetReferenceElectrodeMode, 510
- SetReferenceElectrodeSwitchState, 510
- csDischarge
  - Mcs::Usb, 70
- CSerialPortNet, 511
  - CSerialPortNet, 511
  - GetBytesAvailable, 512
  - Receive, 512
  - ReceiveString, 512
  - Send, 512
- csError
  - Mcs::Usb, 70
- csIdleChargeFinished
  - Mcs::Usb, 70
- csIdleNoBattery
  - Mcs::Usb, 70
- csRefreshBattery
  - Mcs::Usb, 70
- CStg200xBasicNet, 512
  - ~CStg200xBasicNet, 517
  - GetAnalogRanges, 517
  - GetAnalogResolution, 518
  - GetAutocalibrationDisabled, 518
  - GetAvailableMemory, 518
  - GetBlankingEnable, 518, 519
  - GetCurrentRangeInMicroAmpByIndex, 519
  - GetCurrentRangeInNanoAmp, 519
  - GetCurrentRangeListInMicroAmp, 520
  - GetCurrentRangeSelectedIndex, 520
  - GetCurrentResolutionInNanoAmp, 520
  - GetCurrentResolutionInNanoAmpByIndex, 521
  - GetDacAmplificationFactor, 521
  - GetDACResolution, 521
  - GetDiginValue, 521
  - GetDigoutMode, 522
  - GetDigoutValue, 522
  - GetElectrodeDacMux, 522
  - GetElectrodeEnable, 523
  - GetElectrodeMode, 524
  - GetEnableAmplifierProtectionSwitch, 525
  - GetExternalElectrodeEnable, 526
  - GetFAAmplification, 526
  - GetHeadstage, 526
  - GetListModelIndexRange, 526
  - GetListModelTriggerSource, 527
  - GetNumberOfAnalogChannels, 527
  - GetNumberOfCurrentRangeIndexes, 527
  - GetNumberOfHWDACPaths, 527
  - GetNumberOfStimulationElectrodes, 528
  - GetNumberOfStimulationSourcesPerElectrode, 528
  - GetNumberOfSyncoutChannels, 528
  - GetNumberOfTriggerInputs, 528
  - GetNumberOfVoltageRangeIndexes, 528
  - GetOutputRate, 529
  - GetStgProgramInfo, 529
  - GetStgVersionInfo, 530
  - GetSyncoutMap, 530
  - GetTotalMemory, 530
  - GetTriggerSource, 530
  - GetVoltageRangeInMicroVolt, 530
  - GetVoltageRangeInMilliVoltByIndex, 531
  - GetVoltageRangeListInMilliVolt, 531
  - GetVoltageRangeSelectedIndex, 531
  - GetVoltageResolutionInMicroVolt, 531
  - GetVoltageResolutionInMicroVoltByIndex, 532
  - ListModeSendStart, 532
  - ListModeSendStop, 532
  - SendStart, 533
  - SendStop, 533
  - SetAutocalibrationDisabled, 533
  - SetBlankingEnable, 535, 536
  - SetCurrentMode, 536
  - SetCurrentRangeSelectedIndex, 536
  - SetDacAmplificationFactor, 537
  - SetDigoutMode, 537
  - SetDigoutValue, 537
  - SetElectrodeDacMux, 537, 538, 540
  - SetElectrodeEnable, 541, 542
  - SetElectrodeMode, 543, 544
  - SetEnableAmplifierProtectionSwitch, 545, 546
  - SetExternalElectrodeEnable, 546
  - SetFAAmplification, 547
  - SetHeadstage, 547
  - SetListModelIndexRange, 547
  - SetListModelTriggerSource, 547, 548
  - SetMeasurementMode, 548
  - SetOutputRate, 549



- SetStgProgramInfo, [549](#)
- SetSyncoutMap, [549](#)
- SetTriggerSource, [549](#), [550](#)
- SetVoltageMode, [550](#)
- SetVoltageRangeSelectedIndex, [550](#)
- CStg200xDownloadBasicNet, [551](#)
  - ClearChannelData, [552](#)
  - ClearSyncData, [553](#)
  - DisableAutoReset, [553](#)
  - EnableAutoReset, [553](#)
  - ForceStatusEvent, [553](#)
  - GetMemoryUsageDAC, [553](#)
  - GetMemoryUsageSyncout, [553](#)
  - GetSweepCount, [554](#)
  - GetTrigger, [554](#)
  - ResetStatus, [555](#)
  - SendChannelData, [555](#)
  - SendSyncData, [555](#)
  - SetupRetriggerMode, [556](#)
  - SetupTrigger, [557](#)
  - SetupTriggerSingle, [557](#)
  - Stimulus, [558](#)
- CStg200xDownloadNet, [558](#)
  - ~CStg200xDownloadNet, [560](#)
  - ClearChannel\_PrepareAndSendData, [560](#)
  - CStg200xDownloadNet, [559](#)
  - DisableMultiFileMode, [561](#)
  - EnableMultiFileMode, [561](#)
  - GetModuleCurrent, [561](#)
  - GetModuleTemp, [561](#)
  - MwPollStatusEvent, [564](#)
  - PrepareAndAppendData, [561](#)
  - PrepareAndSendData, [562](#)
  - QueryTriggerstatus, [563](#)
  - SendSegmentDefine, [563](#)
  - SendSegmentSelect, [563](#)
  - SendSegmentStart, [564](#)
  - SetOutputMap, [564](#)
  - Stg200xPollStatusEvent, [565](#)
- CStimulusFunctionNet, [565](#)
  - ClearChannel\_PrepareAndSendData, [567](#)
  - ClearChannelData, [567](#)
  - ClearMultiplexedData, [567](#)
  - ClearSyncData, [567](#)
  - CreateSideband, [568](#)
  - CStimulusFunctionNet, [566](#), [567](#)
  - ForceStatusEvent, [568](#)
  - GetAvailableMemory, [569](#)
  - GetCurrentRangeInNanoAmp, [569](#)
  - GetCurrentResolutionInNanoAmp, [569](#)
  - GetDACResolution, [569](#)
  - GetMultiplexedDataChannelsInBlock, [570](#)
  - GetNumberOfAnalogChannels, [570](#)
  - GetTotalMemory, [570](#)
  - GetVoltageRangeInMicroVolt, [570](#)
  - GetVoltageResolutionInMicroVolt, [571](#)
  - PollStatusEvent, [576](#)
  - PrepareAndAppendData, [571](#)
  - PrepareAndSendData, [572](#)
  - PrepareData, [573](#)
  - SendMultiplexedData, [573](#)
  - SendPreparedData, [573](#)
  - SendStart, [573](#)
  - SendStop, [574](#)
  - SetupTrigger, [574](#)
  - SetupTriggerSingle, [575](#)
  - StartPoll, [575](#)
  - StopPoll, [575](#)
- CStimulusFunctionNet::SidebandData, [703](#)
  - !SidebandData, [703](#)
  - ~SidebandData, [703](#)
  - Duration, [703](#)
  - Sideband, [703](#)
  - SidebandData, [703](#)
- CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, [704](#)
  - !StimulusDeviceDataAndUnrolledData, [705](#)
  - ~StimulusDeviceDataAndUnrolledData, [705](#)
  - DeviceData, [705](#)
  - DeviceDataLength, [705](#)
  - StimulusDeviceDataAndUnrolledData, [705](#)
  - UnrolledAmplitude, [705](#)
  - UnrolledDuration, [705](#)
  - UnrolledSync, [705](#)
- Csw2to64DeviceNet, [576](#)
  - ~Csw2to64DeviceNet, [577](#)
  - Csw2to64DeviceNet, [577](#)
  - GetChannel, [577](#)
  - GetChannels, [577](#)
  - GetNumber, [577](#)
  - SetChannel, [577](#)
  - SetChannels, [578](#)
- CTcxDeviceNet, [578](#)
  - ~CTcxDeviceNet, [581](#)
  - CalibrateThermocouple, [581](#)
  - CTcxDeviceNet, [580](#)
  - FactoryReset, [581](#)
  - GetBoardTemp, [581](#)
  - GetCalibration, [581](#)
  - GetCalibrationDecp, [581](#)
  - GetCalibrationMax, [581](#)
  - GetCalibrationMin, [581](#)
  - GetCurrent, [581](#)
  - GetD, [582](#)
  - GetDDecp, [582](#)
  - GetDevice, [582](#)
  - GetDeviceType, [582](#)
  - GetDevname, [582](#)
  - GetDMax, [582](#)
  - GetDMin, [582](#)
  - GetDuty, [582](#)
  - GetEnableHeaterLimit, [583](#)
  - GetEnableThermocouple, [583](#)
  - GetHasThermocouple, [583](#)
  - GetHeaterLimit, [583](#)
  - GetHeaterTemp, [583](#)

- GetI, [583](#)
- GetIDecp, [583](#)
- GetIMax, [584](#)
- GetIMin, [584](#)
- GetIOut, [584](#)
- GetMaxHeaterPowerMultiwell, [584](#)
- GetMaxP, [584](#)
- GetMaxpDecp, [584](#)
- GetMaxpMax, [584](#)
- GetMaxpMin, [585](#)
- GetNumControlChannels, [585](#)
- GetNumDevices, [585](#)
- GetNumMeasureChannels, [585](#)
- GetOnOff, [585](#)
- GetP, [585](#)
- GetPDecp, [585](#)
- GetPMax, [585](#)
- GetPMin, [585](#)
- GetPOut, [585](#)
- GetPwrOut, [586](#)
- GetPwrSet, [586](#)
- GetRes1, [586](#)
- GetRes2, [586](#)
- GetResS, [586](#)
- GetResX, [586](#)
- GetROut, [586](#)
- GetSensorType, [587](#)
- GetSetpoint, [587](#)
- GetSetpointDecp, [587](#)
- GetSetpointMax, [587](#)
- GetSetpointMin, [587](#)
- GetThermocoupleCalibration, [587](#)
- GetThermocoupleNanovoltPerKelvin, [587](#)
- GetThermocoupleReferenceTemp, [588](#)
- GetThermocoupleTemp, [588](#)
- GetThermocoupleTempAbs, [588](#)
- GetUnit, [588](#)
- GetUOut, [588](#)
- GetValue, [588](#)
- GetValueHires, [589](#)
- GetVolti, [589](#)
- SetCalibration, [589](#)
- SetD, [589](#)
- SetDevice, [589](#)
- SetDeviceType, [589](#)
- SetDevname, [589](#)
- SetEnableHeaterLimit, [589](#)
- SetEnableThermocouple, [590](#)
- SetHeaterLimit, [590](#)
- SetI, [590](#)
- SetMaxHeaterPowerMultiwell, [590](#)
- SetMaxP, [590](#)
- SetOnOff, [590](#)
- SetP, [591](#)
- SetSensorType, [591](#)
- SetSetpoint, [591](#)
- SetThermocoupleNanovoltPerKelvin, [591](#)
- CTEERFunctionNet, [591](#)
- !CTEERFunctionNet, [594](#)
- ~CTEERFunctionNet, [594](#)
- CancelInternalCalibration, [594](#)
- CTEERFunctionNet, [593](#)
- GetAdapterCode, [594](#)
- GetAdcOffsetU1, [594](#)
- GetAdcOffsetU2, [594](#)
- GetAmplitude\_nA, [594](#)
- GetBytesPerSample, [595](#)
- GetClampMode, [595](#)
- GetControllerParams, [595](#)
- GetCurrentEnable, [595](#)
- GetDacZero, [596](#)
- GetFrameErrorCounter, [596](#)
- GetLiquidResistance, [596](#)
- GetMaxChunkSize\_Byte, [596](#)
- GetNumberOfAvailableSamples, [596](#)
- GetPeriod\_us, [597](#)
- GetRotaryPositionCode, [597](#)
- GetSampleBufferChunk, [597](#)
- GetSampleRate, [597](#)
- GetSampleVoltageBuffer\_uV, [598](#)
- GetScaleFactorU1, [598](#)
- GetScaleFactorU2, [598](#)
- GetUptimeSeconds, [598](#)
- GetWaveform, [598](#)
- IsInternalCalibrationFinished, [599](#)
- IsSamplingFinished, [599](#)
- SetAmplitude\_nA, [599](#)
- SetBufferIndex, [599](#)
- SetClampMode, [600](#)
- SetControllerParams, [600](#)
- SetCurrentEnable, [600](#)
- SetExternalLED, [600](#)
- SetLiquidResistance, [601](#)
- SetPeriod\_us, [601](#)
- SetWaveform, [601](#)
- StartInternalCalibration, [601](#)
- StartSampling, [601](#)
- StopSampling, [602](#)
- CTEERMachineDeviceNet, [602](#)
- ~CTEERMachineDeviceNet, [602](#)
- CTEERMachineDeviceNet, [602](#)
- TEERFunctionNet, [603](#)
- CurrentClamp
  - Mcs::Usb, [88](#)
- CurrentMeasure
  - Mcs::Usb, [56](#)
- CurrentRangeInNanoAmp
  - W2100\_StimulusParametersNet, [707](#)
- CurrentResolutionInNanoAmp
  - W2100\_StimulusParametersNet, [707](#)
- CUsbDeviceConfigurationFunctionNet, [603](#)
- !CUsbDeviceConfigurationFunctionNet, [604](#)
- ~CUsbDeviceConfigurationFunctionNet, [604](#)
- CUsbDeviceConfigurationFunctionNet, [603](#)
- SetDeviceId, [604](#)
- SetDeviceName, [604](#)

- CUsbExceptionNet, 604
  - CUsbExceptionNet, 605
  - Status, 605
- CutoffFrequency
  - CCreateFilterNet, 114
- CVoltageRangeInfoNet
  - CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet, 606
- CW2100\_FunctionNet, 606
  - ClearStimulusParametersCache, 608
  - ClearUserDefinedNameCache, 608
  - CW2100\_FunctionNet, 608
  - DeselectAllHeadstages, 609
  - DeselectHeadstage, 609
  - GetAccelGyroCurrentRate, 609
  - GetAccelGyroDesiredRate, 609
  - GetAccelGyroEnabled, 609
  - GetAccelRange, 609
  - GetAnalogOutChannel, 609
  - GetAnalogOutFilter, 609
  - GetAudioChannels, 609
  - GetAvailableHeadstages, 609
  - GetBatteryState, 610
  - GetDacRange, 610
  - GetFilterProperties, 610
  - GetFilterProperty, 610
  - GetFPGA FirmwareType, 610
  - GetGyroRange, 610
  - GetHeadstageOnOff, 610
  - GetHeadstageSamplingActive, 610
  - GetMultiHeadstageMode, 610
  - GetPicFirmwareType, 610
  - GetSelectedChannels, 611
  - GetSelectedHeadstageState, 611
  - GetStimulusParametersCache, 611
  - GetStimulusParametersFromSelectedHS, 611
  - GetStimulusParameters, 611
  - GetUserDefinedName, 611
  - GetUserDefinedNameCache, 611
  - GetUserDefinedNameFromSelectedHS, 612
  - PulseGenerator, 613
  - SelectHeadstage, 612
  - SetAccelGyroDesiredRate, 612
  - SetAccelGyroEnabled, 612
  - SetAccelRange, 612
  - SetAnalogOutChannel, 612
  - SetAnalogOutFilter, 612
  - SetAudioChannels, 612
  - SetDacRange, 613
  - SetGyroRange, 613
  - SetHeadstageOnOff, 613
  - SetHeadstageSamplingActive, 613
  - SetHeadstageToSleep, 613
  - SetMultiHeadstageMode, 613
  - SetSelectedChannels, 613
  - Stimulator, 614
- CW2100\_FunctionNet::AudioChannelsNet, 94
  - amplification, 94
  - channel, 94
  - dacqgroup, 94
- CW2100\_StimulatorFunctionNet, 614
  - BOOST\_BIT, 620
  - ClearChannelData, 615
  - CW2100\_StimulatorFunctionNet, 615
  - GetBoostAlwaysOnMode, 616
  - GetBoostPreTime, 616
  - GetCurrentRangeInNanoAmp, 616
  - GetCurrentResolutionInNanoAmp, 616
  - GetDACResolution, 616
  - GetDigitalStimulatorTrigger, 616
  - GetDigitalStimulatorTriggerSlope, 617
  - GetNumberOfAnalogChannels, 617
  - GetNumberOfSyncoutChannels, 617
  - GetNumberOfTriggerInputs, 617
  - GetStimulationPatternMemory, 617
  - GetTimeResolutionInNanoSeconds, 617
  - GetTimeSlot, 617
  - GetVoltageRangeInMicroVolt, 617
  - GetVoltageResolutionInMicroVolt, 618
  - GND\_SWITCH\_BIT, 620
  - PollStatusEvent, 620
  - PrepareData, 618
  - PrepareDataSync, 618
  - SelectTimeSlot, 618
  - SendPreparedData, 618
  - SendStart, 619
  - SendStop, 619
  - SetDigitalStimulatorTrigger, 619
  - SetDigitalStimulatorTriggerSlope, 619
  - StartPoll, 619
  - StopPoll, 619
  - SYNC\_BIT0, 620
  - SYNC\_BIT1, 620
- CW2100DacqGroupChannelSelectionNet, 620
  - CW2100DacqGroupChannelSelectionNet, 620
- CWarnerUssingDeviceNet, 621
  - !CWarnerUssingDeviceNet, 622
  - ~CWarnerUssingDeviceNet, 621
  - CWarnerUssingDeviceNet, 621
  - WarnerUssingFunction, 622
- CWarnerUssingFunctionNet, 622
  - !CWarnerUssingFunctionNet, 624
  - ~CWarnerUssingFunctionNet, 624
  - CompensateElectrodeOffset, 624
  - CWarnerUssingFunctionNet, 624
  - GetAvailableChambers, 625
  - GetChannelsCountOfChamber, 625
  - GetClampMode, 625
  - GetComplianceVoltageThreshold, 625
  - GetDacPampsPerDigitHighCurrentRange, 626
  - GetDacPampsPerDigitLowCurrentRange, 626
  - GetDacZero, 626
  - GetHighCurrentRange, 627
  - GetIdleModeOffset, 627
  - GetLiquidResistance, 627
  - GetLowCurrentRange, 628

- GetNumberOfAvailableChambers, 628
- GetNumberOfHardwareSlotsForChambers, 628
- GetU1Offset, 628
- GetU1Reference, 630
- GetU2Offset, 630
- GetU2Reference, 630
- GetUnitDescription, 631
- GetUnitExponent, 631
- GetUnitName, 631
- GetUnitsPerDigit, 632
- GetUptimeSeconds, 632
- GetVoltageClampControllerParam\_D, 632
- GetVoltageClampControllerParam\_I, 633
- GetVoltageClampControllerParam\_P, 633
- IsChamberAvailable, 633
- IsHighCurrentMode, 634
- IsInternalCalibrationFinished, 634
- IsPulseEnabled, 634
- SetClampMode, 635
- SetEnablePulse, 635
- SetHighCurrentMode, 635
- SetIdleModeOffset, 636
- SetLiquidResistance, 636
- SetLowCurrentMode, 636
- SetPulse, 636
- SetVoltageClampControllerParam\_D, 637
- SetVoltageClampControllerParam\_I, 637
- SetVoltageClampControllerParam\_P, 637
- WaitForAllChambers, 638
- WaitForChamber, 638
- CWarnerValveControllerDeviceNet, 638
  - !CWarnerValveControllerDeviceNet, 643
  - ~CWarnerValveControllerDeviceNet, 643
- ClearTableName, 643
- ClearValveTable, 643
- CWarnerValveControllerDeviceNet, 643
- GetActiveRunningTableNumber, 643
- GetActiveRunningTableNumberEvent, 660
- GetAnalogThresholdHigh, 643
- GetAnalogThresholdHighEvent, 660
- GetAnalogThresholdLow, 644
- GetAnalogThresholdLowEvent, 660
- GetAnalogVoltage, 644
- GetAnalogVoltageEvent, 660
- GetCurrentEditTableNumber, 644
- GetCurrentNumberOfValves, 644
- GetCurrentNumberOfValvesEvent, 661
- GetDigitalOutPortValve, 645
- GetDigitalOutPortValveEvent, 661
- GetDigitalPortDirection, 645
- GetDigitalPortDirectionEvent, 661
- GetDisplayMode, 645
- GetDisplayModeEvent, 661
- GetTableName, 645
- GetTableNamebyIndex, 646
- GetTableNamebyIndexEvent, 661
- GetTotalNumberOfDigitalPorts, 646
- GetTotalNumberOfTables, 646
- GetTotalNumberOfValves, 646
- GetTotalTableSize, 647
- GetValveActive, 647
- GetValveActiveEvent, 661
- GetValveBoardRevision, 647
- GetValveBoardRevisionEvent, 661
- GetValveBoardRevisionString, 647
- GetValveCurrent, 647
- GetValveDigitalInPort, 648
- GetValveDigitalInPortEvent, 662
- GetValveLedOn, 648
- GetValveLedOnEvent, 662
- GetValveManualGroup, 648
- GetValveManualGroupEvent, 662
- GetValveManualState, 649
- GetValveManualStateEvent, 662
- GetValveMode, 649
- GetValveModeEvent, 662
- GetValvesActiveMap, 649
- GetValvesManualStateMap, 649
- GetValveTableEntry, 650
- IsDigitalOutPortInverted, 650
- IsDigitalOutPortInvertedEvent, 662
- IsValveDigitalInInverted, 650
- IsValveDigitalInInvertedEvent, 662
- IsValveOpen, 651
- IsValveOpenEvent, 663
- IsValveOpenInAnalogMode, 651
- IsValveOpenInAnalogModeEvent, 663
- IsValveOpenInDigitalMode, 651
- IsValveOpenInDigitalModeEvent, 663
- LoadValveTable, 651
- OnGetActiveRunningTableNumber, 652
- OnGetAnalogThresholdHigh, 652
- OnGetAnalogThresholdLow, 652
- OnGetAnalogVoltage, 652
- OnGetCurrentNumberOfValves, 652
- OnGetDigitalOutPortValve, 652
- OnGetDigitalPortDirection, 652
- OnGetDisplayMode, 652
- OnGetTableNamebyIndex, 652
- OnGetValveActive, 653
- OnGetValveBoardRevision, 653
- OnGetValveDigitalInPort, 653
- OnGetValveLedOn, 653
- OnGetValveManualGroup, 653
- OnGetValveManualState, 653
- OnGetValveMode, 653
- OnIsDigitalOutPortInverted, 653
- OnIsValveDigitalInInverted, 654
- OnIsValveOpen, 654
- OnIsValveOpenInAnalogMode, 654
- OnIsValveOpenInDigitalMode, 654
- OnTableEntryChanged, 654
- SetActiveRunningTableNumber, 654
- SetAnalogThresholdHigh, 654
- SetAnalogThresholdLow, 655
- SetCurrentEditTableNumber, 655

- SetDefault, 655
- SetDigitalOutPortInvert, 655
- SetDigitalOutPortValve, 656
- SetDigitalPortDirection, 656
- SetDisplayMode, 656
- SetTableName, 656
- SetTableStep, 657
- SetTableStepAll, 657
- SetValveActive, 657
- SetValveCurrent, 657
- SetValveDigitalInInvert, 658
- SetValveDigitalInPort, 658
- SetValveLedOn, 658
- SetValveManualGroup, 658
- SetValveManualState, 659
- SetValveMode, 659
- SetValvesActiveMap, 659
- SetValvesManualStateMap, 659
- SetValveTableEntry, 659
- StoreValveTable, 660
- TableEntryChangedEvent, 663
- CWarnerValveControllerDeviceTesterFunctionNet, 663
  - !CWarnerValveControllerDeviceTesterFunctionNet, 665
  - ~CWarnerValveControllerDeviceTesterFunctionNet, 664
- CWarnerValveControllerDeviceTesterFunctionNet, 664
- GetIO, 665
- GetSync, 665
- SetADC, 665
- SetIO, 665
- SetIODirection, 666
- SetTrigger, 666
- SetTriggerSyncDirection, 666
- CWClassicFunctionNet, 666
  - CWClassicFunctionNet, 667, 668
  - GetFilterParametersHeadstage, 668
  - GetHasChecksum, 668
  - GetHasRedLedHeadstage, 668
  - GetHeadstageOnOff, 668
  - GetResetFilter, 668
  - GetRFFConnectionStatus, 668
  - GetRFFFrequencyHeadstage, 668
  - GetRFFFrequencyReceiver, 668
  - GetRFFPower, 669
  - GetScanHeadstagesResult, 669
  - GetSelectedHeadstage, 669
  - GetSerialNumberHeadstage, 669
  - GetWPADebugMode, 669
  - GetWPAType, 669
  - ResetChannelmap, 669
  - ScanForHeadstages, 669
  - SetChannelmap, 669
  - SetFilterParametersHeadstage, 669
  - SetHasChecksum, 670
  - SetHeadstageOnOff, 670
  - SetHWSelectedChannels, 670
  - SetResetFilter, 670
  - SetRFFFrequencyHeadstage, 670
  - SetRFFFrequencyReceiver, 670
  - SetRFFFrequencyReceiverEeprom, 670
  - SetRFLostBehaviour, 670
  - SetRFFPower, 671
  - SetSelectedHeadstage, 671
  - SetSerialNumberHeadstage, 671
  - SetWPADebugMode, 671
  - SetWPAType, 671
- CWirelessBaseFunctionNet, 671
  - CreateWirelessHeadstageSerialNumberString, 672
  - CWirelessBaseFunctionNet, 672
- CyclePort
  - CMcsUsbNet, 316
- Cypress
  - Mcs::Usb, 89
- Cypress\_FX1
  - Mcs::Usb, 75
- Cypress\_FX2
  - Mcs::Usb, 75
- Cypress\_FX3
  - Mcs::Usb, 75
- DAC1Channel
  - Mcs::Usb, 58
- DAC2Channel
  - Mcs::Usb, 58
- DAC3Channel
  - Mcs::Usb, 58
- DAC4Channel
  - Mcs::Usb, 58
- DACQ1DigitalGroup
  - Mcs::Usb, 59
- DacqDevice
  - CSafeISDeviceNet, 493
- dacqgroup
  - CW2100\_FunctionNet::AudioChannelsNet, 94
- DacqGroupChannelEnumNet
  - Mcs::Usb, 59
- DacqMeaGroupTypeEnumNet
  - Mcs::Usb, 59
- DacqTrigger
  - Mcs::Usb, 64
- DACResolution
  - W2100\_StimulusParametersNet, 707
- DataModeEnumNet
  - Mcs::Usb, 59
- DataState
  - HeadStageIDTypeState, 690
- DeepCopy
  - CCMOSMeaDeviceNet::CRegionOfInterestRect, 437
- DefineAmplification
  - CPgaDeviceNet, 402
- DefineFrequencyRange
  - CPgaDeviceNet, 402
- DefineNumAmplifications

- CPgaDeviceNet, 402
- DefineNumFrequencyRanges
  - CPgaDeviceNet, 402
- DeselectAllHeadstages
  - CW2100\_FunctionNet, 609
- DeselectHeadstage
  - CW2100\_FunctionNet, 609
- DEST\_FX3\_TARGET\_MASK
  - Mcs::Usb, 55
- DEST\_TARGET1
  - Mcs::Usb, 55
- DEST\_TARGET10
  - Mcs::Usb, 55
- DEST\_TARGET11
  - Mcs::Usb, 55
- DEST\_TARGET12
  - Mcs::Usb, 55
- DEST\_TARGET13
  - Mcs::Usb, 55
- DEST\_TARGET14
  - Mcs::Usb, 55
- DEST\_TARGET15
  - Mcs::Usb, 55
- DEST\_TARGET2
  - Mcs::Usb, 55
- DEST\_TARGET3
  - Mcs::Usb, 55
- DEST\_TARGET4
  - Mcs::Usb, 55
- DEST\_TARGET5
  - Mcs::Usb, 55
- DEST\_TARGET6
  - Mcs::Usb, 55
- DEST\_TARGET7
  - Mcs::Usb, 55
- DEST\_TARGET8
  - Mcs::Usb, 55
- DEST\_TARGET9
  - Mcs::Usb, 55
- DEST\_TARGET\_MASK
  - Mcs::Usb, 55
- DetectChipType
  - CCMOSMea\_FunctionNet, 101
- DEVICE\_NOT\_FOUND
  - Mcs::Usb, 65
- DeviceArrival
  - CMcsUsbListNet, 309
- DeviceData
  - CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, 705
- DeviceDataLength
  - CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, 705
- DeviceEnumNet
  - Mcs::Usb, 61
- DeviceHasNoHeadstage
  - Mcs::Usb, 67, 68, 80
- DeviceId
  - CMcsUsbListEntryNet, 306
- DeviceIdNet, 672
  - BcdDevice, 673
  - BusType, 673
  - DeviceIdNet, 673
  - IdProduct, 673
  - IdVendor, 673
  - operator=, 673
- DeviceName
  - CMcsUsbListEntryNet, 306
- DeviceNotConnected
  - Mcs::Usb, 67, 68, 80
- DeviceRemoval
  - CMcsUsbListNet, 309
- DeviceRunStatus
  - Mcs::Usb, 63, 72, 81, 84, 91
- DigDataFromReceiver
  - Mcs::Usb, 91
- Digital
  - Mcs::Usb, 75, 92
- DigitalData
  - Mcs::Usb, 63, 72, 81, 84, 91
- DigitalDatastreamEnableEnumNet
  - Mcs::Usb, 62
- DigitalGroup
  - Mcs::Usb, 59
- DigitalIn
  - Mcs::Usb, 62, 63, 71, 81, 84, 91
- DigitalInOfOutPort
  - Mcs::Usb, 63, 71, 81, 84, 91
- DigitalInPort
  - Mcs::Usb, 57
- DigitalInReserverd
  - Mcs::Usb, 62
- DigitalMux
  - Mcs::Usb, 57
- DigitalOut
  - Mcs::Usb, 62
- DigitalOutReg
  - Mcs::Usb, 57
- DigitalOutReserved
  - Mcs::Usb, 62
- DigitalOutStimulator
  - Mcs::Usb, 63, 72, 81, 84, 91
- DigitalPulse
  - Mcs::Usb, 63, 71, 81, 84, 91
- DigitalReg
  - Mcs::Usb, 57
- DigitalSource
  - DigitalSource< digitalsourceenum >, 674
  - DigitalSource< digitalsourceenum >, 674
  - DigitalSource, 674
  - MaxBitNumber, 674
  - MaxBitNumberStatic, 674
  - size, 674
  - Source, 675
- DigitalSourceEnumNet
  - Mcs::Usb, 63



- DigitalSourceGeneral, [675](#)
  - DigitalSourceGeneral, [675](#)
  - MaxBitNumber, [675](#), [676](#)
  - size, [676](#)
  - Source, [676](#)
- DigitalStimulatorTriggerEventEnumNet
  - Mcs::Usb, [64](#)
- DigitalStimulatorTriggerSlopeEnumNet
  - Mcs::Usb, [64](#)
- DigitalTargetEnumNet
  - Mcs::Usb, [64](#)
- Digout
  - Mcs::Usb, [64](#)
- DigOutStim
  - Mcs::Usb, [62](#)
- DigOutStimulatorStartTrigger
  - Mcs::Usb, [64](#)
- DigOutStimulatorStopTrigger
  - Mcs::Usb, [64](#)
- Digstream
  - Mcs::Usb, [64](#)
- DigStreamFromReceiver
  - Mcs::Usb, [91](#)
- DigStreamToReceiver
  - Mcs::Usb, [64](#)
- Dilutor
  - Mcs::Usb, [78](#)
- DisableAutoReset
  - CStg200xDownloadBasicNet, [553](#)
- DisableMultiFileMode
  - CStg200xDownloadNet, [561](#)
- Disconnect
  - CMcsUsbNet, [316](#)
- DisConnectDevice
  - CRadioControlledDevicesNet, [436](#)
- DongleS
  - Mcs::Usb, [76](#)
- DoRamp
  - CRoboDacqNet, [451](#)
- Dotriapot
  - Mcs::Usb, [76](#)
- DoubleToInt
  - Mcs::Usb, [66](#)
- DownloadFirmware
  - CMcsUsbFactoryNet, [294](#)
- DownloadOnly
  - Mcs::Usb, [83](#)
- DriverVersionNet, [676](#)
  - ~DriverVersionNet, [677](#)
  - DriverVersionNet, [677](#)
  - DriverVersionNet::FormatVersion, [677](#)
  - GetDestinationCode, [678](#)
  - GetDestinationName, [679](#)
  - GetMajor, [679](#)
  - GetMinor, [680](#)
  - GetNumEntries, [680](#)
  - GetSerialNumber, [680](#)
  - GetStatus, [681](#)
  - GetVersionInt, [681](#)
  - GetVersionString, [682](#)
- DriverVersionNet::FormatVersion
  - DriverVersionNet, [677](#)
- DSP
  - FirmwareDestinationNames, [683](#)
  - Mcs::Usb, [52](#)
- DSPAnalogGroup
  - Mcs::Usb, [71](#), [81](#)
- DSPDataGroup
  - Mcs::Usb, [59](#), [90](#)
- DSPDigitalGroup
  - Mcs::Usb, [71](#), [81](#)
- DummyCommand
  - CLIH3DeviceNet, [192](#)
- Duration
  - CStimulusFunctionNet::SidebandData, [703](#)
- eCube
  - Mcs::Usb, [77](#), [80](#)
- eCubeHeadstage
  - Mcs::Usb, [67](#)
- ElectricalStimulation
  - HeadStageIDType, [686](#)
- ElectrodeDacMuxEnumNet
  - Mcs::Usb, [64](#)
- ElectrodeModeEnumNet
  - Mcs::Usb, [65](#)
- ElectrodeOffset
  - Mcs::Usb, [88](#)
- emAutomatic
  - Mcs::Usb, [65](#)
- emManual
  - Mcs::Usb, [65](#)
- EmptyKey
  - CMcsUsbNet, [316](#)
- Emu\_GetCellCapacity
  - CRoboDacqNet, [452](#)
- Emu\_GetCellPotential
  - CRoboDacqNet, [452](#)
- Emu\_GetCellResists
  - CRoboDacqNet, [452](#)
- Emu\_GetElectrodeResists
  - CRoboDacqNet, [452](#)
- Emu\_GetNoise
  - CRoboDacqNet, [452](#)
- Emu\_SetCellCapacity
  - CRoboDacqNet, [452](#)
- Emu\_SetCellPotential
  - CRoboDacqNet, [452](#)
- Emu\_SetCellResists
  - CRoboDacqNet, [452](#)
- Emu\_SetElectrodeResists
  - CRoboDacqNet, [452](#)
- Emu\_SetNoise
  - CRoboDacqNet, [452](#)
- EnableAnalogOut
  - CSCUFunctionNet, [497](#)
- EnableAutoReset

- CStg200xDownloadBasicNet, 553
- EnableChannelsInGroup
  - CCMOSMea\_FunctionNet, 101
  - CDacqGroupChannelSelectionTemplateNet< Dac-  
qGroupChannelEnumTemplateNet, Dac-  
qGroupChannelEnumTemplate, CDevice-  
GroupChannelInfoTemplateNet >, 118
- EnableChecksum
  - CMeaDeviceNet, 347
  - COctoPotDeviceNet, 391
- EnableDigitalIn
  - CMeaDeviceNet, 348
  - COctoPotDeviceNet, 391
- EnableExceptions
  - CMcsUsbNet, 316
- EnableMultiFileMode
  - CStg200xDownloadNet, 561
- EnableQueue
  - CRoboDeviceNet, 467
- EnableTimestamp
  - CMeaDeviceNet, 349
  - COctoPotDeviceNet, 391
- EnableUserTrigger
  - CLIH3DeviceNet, 192
- Encapsulator
  - Mcs::Usb, 77
- enCMosMeaChipType
  - Mcs::Usb, 65
- EnSTG200x\_STATUS
  - Mcs::Usb, 65
- Entry
  - HeadStageIDType, 687
- EOFAAndCRC
  - Mcs::Usb, 58
- Equals
  - CMcsUsbListEntryNet, 303
  - HeadStageIDType, 687
  - HeadstageIDTypeObject, 689
- EraseEepromRegisterPreconfig
  - CMcsUsbNet, 316
- EraseFilterParameterPermanent
  - CFilterConfigurationNet, 134
  - CFilterConfigurationRegisterNet, 135, 136
- ErasePermanentAdcOffset
  - CLIH3DeviceNet, 193
- ErasePermanentDacOffset
  - CLIH3DeviceNet, 193
- Error\_Callback\_Aquisition\_Stopped
  - CMcsUsbDacqNet, 288
- Error\_Callback\_Data\_lost
  - CMcsUsbDacqNet, 288
- Error\_Callback\_Frames\_Lost
  - CMcsUsbDacqNet, 288
- Error\_Callback\_Packet\_Error
  - CMcsUsbDacqNet, 288
- Error\_Callback\_Queue\_Full
  - CMcsUsbDacqNet, 289
- Error\_Callback\_RingQueue\_Full
  - CMcsUsbDacqNet, 289
- ErrorEvent
  - CMcsUsbDacqNet, 289
- ExternBCTester
  - Mcs::Usb, 76
- ExternDTester
  - Mcs::Usb, 76
- ExternSTester
  - Mcs::Usb, 76
- FactoryReset
  - CTcxDeviceNet, 581
- Falling
  - Mcs::Usb, 64
- FCB
  - Mcs::Usb, 76
- FCX
  - Mcs::Usb, 76
- Feedback
  - Mcs::Usb, 63, 71, 81, 84, 91
- FeedbackGetSampleTimerCount
  - CMeaFeedbackFunctionNet, 356
- FeedbackHigh
  - Mcs::Usb, 62
- FeedbackLow
  - Mcs::Usb, 62
- FeedbackReg
  - Mcs::Usb, 57
- FeedbackSetAnalogSource
  - CMeaFeedbackFunctionNet, 356
- FeedbackSetChannelFilter
  - CMeaFeedbackFunctionNet, 356
- FeedbackSetDigitalMapping
  - CMeaFeedbackFunctionNet, 357
- FeedbackSetFeedback
  - CMeaFeedbackFunctionNet, 357
- FeedbackSetFilterOff
  - CMeaFeedbackFunctionNet, 357
- FeedbackSetFilterParameter
  - CMeaFeedbackFunctionNet, 357
- FeedbackSetFilterParameter32
  - CMeaFeedbackFunctionNet, 357
- FeedbackSetGlobalChannelFilter
  - CMeaFeedbackFunctionNet, 357
- FeedbackSetIIRFilterParameter
  - CMeaFeedbackFunctionNet, 357
- FeedbackSetLogic
  - CMeaFeedbackFunctionNet, 357
- FeedbackSetMkFilter
  - CMeaFeedbackFunctionNet, 358
- FeedbackSetNumberOfLogics
  - CMeaFeedbackFunctionNet, 358
- FeedbackSetNumberOfRateCounter
  - CMeaFeedbackFunctionNet, 358
- FeedbackSetNumberOfRateDetectors
  - CMeaFeedbackFunctionNet, 358
- FeedbackSetNumberOfSpikeDetectors
  - CMeaFeedbackFunctionNet, 358
- FeedbackSetNumberOfTriggers



- CMeaFeedbackFunctionNet, [358](#)
- FeedbackSetRateCounter
  - CMeaFeedbackFunctionNet, [358](#)
- FeedbackSetRateDetector
  - CMeaFeedbackFunctionNet, [358](#)
- FeedbackSetSpikeDetectorThreshold
  - CMeaFeedbackFunctionNet, [359](#)
- FeedbackSetTrigger
  - CMeaFeedbackFunctionNet, [359](#)
- FilterActive
  - CFilterPropertyNet, [138](#)
- FilterAttributeEnumNet
  - Mcs::Usb, [65](#)
- FilterBand
  - CFilterPropertyNet, [138](#)
- FilterBandEnumNet
  - Mcs::Usb, [66](#)
- FilterCalculationDirectionEnumNet
  - Mcs::Usb, [66](#)
- FilterFamily
  - CFilterPropertyNet, [138](#)
- FilterFamilyEnumNet
  - Mcs::Usb, [66](#)
- FilterType
  - CFilterPropertyNet, [138](#)
- FilterTypeEnumNet
  - Mcs::Usb, [66](#)
- FindEndpoints
  - CGenericDevelopDeviceNet, [154](#)
- FindFilter
  - CCreateFilterNet, [113](#)
- FindFirmwareVersionMagicInBuffer
  - CMcsUsbFactoryNet, [294](#)
- FindReference
  - CRoboDeviceNet, [467](#)
- FindReferencel
  - CRoboStatorDeviceNet, [486](#)
- FindReferencePhase0
  - CRoboDeviceNet::RoboMainLowLevelCommands, [695](#)
- FindReferencePhase0XY
  - CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands, [701](#)
- FindReferenceXY
  - CRoboStatorDeviceNet, [486](#)
- FindReferenceZ
  - CRoboStatorDeviceNet, [486](#)
- Finished
  - Mcs::Usb, [83](#)
- FirePressurePulse
  - CPPCFunctionNet, [418](#)
- FirmwareDestinationNames, [682](#)
  - Altera, [683](#)
  - Bootstrap, [683](#)
  - BUS1\_MCSBUS1, [683](#)
  - BUS1\_MCSBUS2, [683](#)
  - DSP, [683](#)
  - FPGA2, [683](#)
  - FPGA3, [683](#)
  - FPGA4, [683](#)
  - FPGA5, [683](#)
  - FPGA6, [683](#)
  - MCSBUS1, [684](#)
  - MCSBUS10, [684](#)
  - MCSBUS11, [684](#)
  - MCSBUS12, [684](#)
  - MCSBUS13, [684](#)
  - MCSBUS2, [684](#)
  - MCSBUS3, [684](#)
  - MCSBUS4, [684](#)
  - MCSBUS5, [684](#)
  - MCSBUS6, [684](#)
  - MCSBUS7, [684](#)
  - MCSBUS8, [685](#)
  - MCSBUS9, [685](#)
  - MCU1, [685](#)
  - PIC, [685](#)
  - PIC2, [685](#)
  - PIC3, [685](#)
  - PIC4, [685](#)
  - USB, [685](#)
- FluidControlDevice
  - CSafeISDeviceNet, [493](#)
- ForceStatusEvent
  - CStg200xDownloadBasicNet, [553](#)
  - CStimulusFunctionNet, [568](#)
- FPGA10
  - Mcs::Usb, [54](#)
- FPGA10\_BASE
  - Mcs::Usb, [55](#)
- FPGA10\_GOLD
  - Mcs::Usb, [55](#)
- FPGA11
  - Mcs::Usb, [54](#)
- FPGA11\_BASE
  - Mcs::Usb, [55](#)
- FPGA11\_GOLD
  - Mcs::Usb, [55](#)
- FPGA12
  - Mcs::Usb, [54](#)
- FPGA12\_BASE
  - Mcs::Usb, [55](#)
- FPGA12\_GOLD
  - Mcs::Usb, [55](#)
- FPGA13
  - Mcs::Usb, [54](#)
- FPGA13\_BASE
  - Mcs::Usb, [55](#)
- FPGA13\_GOLD
  - Mcs::Usb, [55](#)
- FPGA14
  - Mcs::Usb, [54](#)
- FPGA14\_BASE
  - Mcs::Usb, [55](#)
- FPGA14\_GOLD
  - Mcs::Usb, [55](#)

FPGA15  
     Mcs::Usb, [54](#)  
 FPGA15\_BASE  
     Mcs::Usb, [55](#)  
 FPGA15\_GOLD  
     Mcs::Usb, [55](#)  
 FPGA16  
     Mcs::Usb, [54](#)  
 FPGA16\_BASE  
     Mcs::Usb, [55](#)  
 FPGA16\_GOLD  
     Mcs::Usb, [55](#)  
 FPGA2  
     FirmwareDestinationNames, [683](#)  
     Mcs::Usb, [54](#)  
 FPGA2\_BASE  
     Mcs::Usb, [55](#)  
 FPGA2\_GOLD  
     Mcs::Usb, [54](#)  
 FPGA3  
     FirmwareDestinationNames, [683](#)  
     Mcs::Usb, [54](#)  
 FPGA3\_BASE  
     Mcs::Usb, [55](#)  
 FPGA3\_GOLD  
     Mcs::Usb, [54](#)  
 FPGA4  
     FirmwareDestinationNames, [683](#)  
     Mcs::Usb, [54](#)  
 FPGA4\_BASE  
     Mcs::Usb, [55](#)  
 FPGA4\_GOLD  
     Mcs::Usb, [54](#)  
 FPGA5  
     FirmwareDestinationNames, [683](#)  
     Mcs::Usb, [54](#)  
 FPGA5\_BASE  
     Mcs::Usb, [55](#)  
 FPGA5\_GOLD  
     Mcs::Usb, [54](#)  
 FPGA6  
     FirmwareDestinationNames, [683](#)  
     Mcs::Usb, [54](#)  
 FPGA6\_BASE  
     Mcs::Usb, [55](#)  
 FPGA6\_GOLD  
     Mcs::Usb, [54](#)  
 FPGA7  
     Mcs::Usb, [54](#)  
 FPGA7\_BASE  
     Mcs::Usb, [55](#)  
 FPGA7\_GOLD  
     Mcs::Usb, [54](#)  
 FPGA8  
     Mcs::Usb, [54](#)  
 FPGA8\_BASE  
     Mcs::Usb, [55](#)  
 FPGA8\_GOLD  
     Mcs::Usb, [54](#)  
 FPGA9  
     Mcs::Usb, [54](#)  
 FPGA9\_BASE  
     Mcs::Usb, [55](#)  
 FPGA9\_GOLD  
     Mcs::Usb, [55](#)  
 FPGA\_BASE  
     Mcs::Usb, [55](#)  
 FPGA\_BOOTSTRAP  
     Mcs::Usb, [55](#)  
 FPGA\_GOLD  
     Mcs::Usb, [54](#)  
 FPGA\_NORMAL  
     Mcs::Usb, [52](#)  
 FpgalEnumNet  
     Mcs::Usb, [67](#)  
 FrameContextGroup  
     Mcs::Usb, [59](#)  
 FromIntPtr  
     StgStatusNet, [704](#)  
 FromPtr  
     StgStatusNet, [704](#)  
 FullCharge  
     Mcs::Usb, [70](#)  
 FullSpeed  
     Mcs::Usb, [71](#)  
 FunkDongleS  
     Mcs::Usb, [76](#)  
 FX3MCSDDataAddress  
     CMcsUsbFactoryNet, [299](#)  
 FX3MCSDDataDeviceIdOffset  
     CMcsUsbFactoryNet, [299](#)  
 FX3MCSDDataIb1ImageOffset  
     CMcsUsbFactoryNet, [299](#)  
 FX3MCSDDataIb2ImageOffset  
     CMcsUsbFactoryNet, [299](#)  
 FX3MCSDDataVersionOffset  
     CMcsUsbFactoryNet, [299](#)  
 FYIPProgram  
     CFYIDeviceNet, [146](#)  
 FYITemp  
     CFYIDeviceNet, [146](#)  
 Gain  
     CMeaDeviceNet, [352](#)  
 Gated\_High\_Active  
     Mcs::Usb, [78](#)  
 Gated\_Low\_Active  
     Mcs::Usb, [78](#)  
 GE2100  
     Mcs::Usb, [77](#)  
 Generic  
     Mcs::Usb, [76](#)  
 Get2AnalogInput  
     CMcsBus\_SensorNet, [225](#)  
 Get2DigitalInput  
     CMcsBus\_SensorNet, [225](#)  
 Get4ADC

- CMcsBus\_SensorNet, [225](#)
- Get4ADCAverage
  - CMcsBus\_SensorNet, [225](#)
- Get4ADCCatchampAverageShift
  - CMcsBus\_SensorNet, [225](#)
- Get4ADCMode
  - CMcsBus\_SensorNet, [225](#)
- Get4DAC
  - CMcsBus\_SensorNet, [225](#)
- GetAbsMaxCurrentInMicroAmp
  - CMultiwellOptoStimFunctionNet, [385](#)
- GetAccelGyroCurrentRate
  - CW2100\_FunctionNet, [609](#)
- GetAccelGyroDesiredRate
  - CW2100\_FunctionNet, [609](#)
- GetAccelGyroEnabled
  - CW2100\_FunctionNet, [609](#)
- GetAccelRange
  - CW2100\_FunctionNet, [609](#)
- GetActiveRunningTableNumber
  - CWarnerValveControllerDeviceNet, [643](#)
- GetActiveRunningTableNumberEvent
  - CWarnerValveControllerDeviceNet, [660](#)
- GetAdapterCode
  - CMealImpedanceDeviceNet, [360](#)
  - CTEERFunctionNet, [594](#)
- GetAdapterType
  - CMcsUsbDacqNet, [264](#)
- GetAdc
  - CFluidControlDeviceNet, [140](#)
- GetAdcDataFormat
  - CMcsUsbDacqNet, [264](#)
- GetADCInputOffset
  - CCMOSMea\_FunctionNet, [101](#)
- GetAdcOffset
  - CLIH3DeviceNet, [193](#)
  - COctoPotDeviceNet, [391](#)
- GetAdcOffsetU1
  - CTEERFunctionNet, [594](#)
- GetAdcOffsetU2
  - CTEERFunctionNet, [594](#)
- GetADCs
  - CMcsBus\_SensorNet, [225](#)
- GetADCsLoop
  - CMcsBus\_SensorNet, [226](#)
- GetAdcZero
  - CMcsUsbDacqNet, [265](#)
- GetAirpressure
  - CRoboDeviceNet, [467](#)
- GetAirpressureLimit
  - CRoboDeviceNet, [467](#)
- GetAirValve
  - CRoboDeviceNet, [467](#)
- GetAllDigout
  - CRoboDacqNet, [453](#)
- GetAmplification
  - CPgaDeviceNet, [402](#)
- GetAmplitude\_nA
  - CTEERFunctionNet, [594](#)
- GetAnalogGain
  - CMeaDeviceNet, [349](#)
- GetAnalogOutADCRange
  - CSCUFunctionNet, [498](#)
- GetAnalogOutChannel
  - CW2100\_FunctionNet, [609](#)
- GetAnalogOutChannels
  - CSCUFunctionNet, [498](#)
- GetAnalogOutDACRange
  - CSCUFunctionNet, [498](#)
- GetAnalogOutFilter
  - CW2100\_FunctionNet, [609](#)
- GetAnalogRanges
  - CStg200xBasicNet, [517](#)
- GetAnalogResolution
  - CStg200xBasicNet, [518](#)
- GetAnalogThresholdHigh
  - CWarnerValveControllerDeviceNet, [643](#)
- GetAnalogThresholdHighEvent
  - CWarnerValveControllerDeviceNet, [660](#)
- GetAnalogThresholdLow
  - CWarnerValveControllerDeviceNet, [644](#)
- GetAnalogThresholdLowEvent
  - CWarnerValveControllerDeviceNet, [660](#)
- GetAnalogValueUnit
  - CMcsUsbDacqNet, [265](#)
- GetAnalogVoltage
  - CPPCFunctionNet, [418](#)
  - CPPS\_FunctionNet, [427](#)
  - CWarnerValveControllerDeviceNet, [644](#)
- GetAnalogVoltageEvent
  - CWarnerValveControllerDeviceNet, [660](#)
- GetAnalogVoltageRange
  - CPPCFunctionNet, [418](#)
- GetAnalogVoltages
  - CPPS\_FunctionNet, [427](#)
- GetArraySize
  - CMealImpedanceDeviceNet, [360](#)
- GetAudioChannels
  - CMeaAudioFunctionNet, [334](#)
  - CW2100\_FunctionNet, [609](#)
- GetAudioOutDacParameter
  - CLIH3DeviceNet, [193](#)
- GetAutocalibrationDisabled
  - CStg200xBasicNet, [518](#)
- GetAvailableBaseSamplerates
  - CCMOSMeaDeviceNet, [110](#)
  - CGrapheneASICDeviceNet, [167](#)
- GetAvailableChambers
  - CWarnerUssingFunctionNet, [625](#)
- GetAvailableDeviceList
  - CRFFFunctionNet, [442](#)
- GetAvailableDeviceListEx
  - CRFFFunctionNet, [442](#)
- GetAvailableHeadstages
  - CSCUFunctionNet, [498](#)
  - CW2100\_FunctionNet, [609](#)

- GetAvailableHeadstagesEvent
  - CSCUFunctionNet, [511](#)
- GetAvailableMemory
  - CStg200xBasicNet, [518](#)
  - CStimulusFunctionNet, [569](#)
- GetAvailableSampleRates
  - CMcsUsbDacqNet::CHWInfo, [182](#)
- GetAvailableStateList
  - CRFFunctionNet, [442](#)
- GetAvailableStateListEx
  - CRFFunctionNet, [442](#)
- GetAvailableVoltageRangesInMicroVolt
  - CMcsUsbDacqNet::CHWInfo, [182](#)
- GetAvailableVoltageRangesInMicroVoltAndStringsInMilliVolt
  - CMcsUsbDacqNet::CHWInfo, [183](#)
- GetAxisConfig
  - CRoboDeviceNet::RoboMainLowLevelCommands, [695](#)
- GetAxisLED
  - CRoboocyte2DeviceNet, [484](#)
- GetAxisParametersSignedEeprom
  - CMcsBus\_AxisParametersNet, [200](#)
- GetAxisParametersUnsignedEeprom
  - CMcsBus\_AxisParametersNet, [200](#)
- GetBaseFrequency
  - CRFFunctionNet, [443](#)
- GetBaseSamplerate
  - CCMOSMeaDeviceNet, [111](#)
- GetBath
  - CCMOSMea\_FunctionNet, [101](#)
- GetBathMode
  - CCMOSMea\_FunctionNet, [101](#)
- GetBatteryState
  - CW2100\_FunctionNet, [610](#)
- GetBatteryVoltage
  - CMultiBatteryChargerDeviceNet, [368](#)
- GetBiQuad
  - CCreateFilterNet, [114](#)
- GetBiQuads
  - CCreateFilterNet, [114](#)
- GetBlankingEnable
  - CStg200xBasicNet, [518](#), [519](#)
- GetBoardTemp
  - CTcxDeviceNet, [581](#)
- GetBoostAlwaysOnMode
  - CW2100\_StimulatorFunctionNet, [616](#)
- GetBoostPreTime
  - CW2100\_StimulatorFunctionNet, [616](#)
- GetBubbleState
  - CPPS\_FunctionNet, [427](#)
- GetBubbleStatus
  - CMcsBus\_SensorNet, [226](#)
- GetBuffer
  - CGenericDevelopDeviceNet, [154](#)
- GetBusAddress
  - CMcsBusNet, [239](#)
- GetBusAddressEeprom
  - CMcsBusNet, [239](#)
- GetByteBuffer
  - CGenericDevelopDeviceNet, [155](#)
- GetBytesAvailable
  - CSerialPortNet, [512](#)
- GetBytesPerSample
  - CTEERFunctionNet, [595](#)
- GetCalibration
  - CTcxDeviceNet, [581](#)
- GetCalibrationDecp
  - CTcxDeviceNet, [581](#)
- GetCalibrationMax
  - CTcxDeviceNet, [581](#)
- GetCalibrationMin
  - CTcxDeviceNet, [581](#)
- GetCapacityC
  - CRoboDacqNet, [453](#)
- GetCapacityV
  - CRoboDacqNet, [453](#)
- GetCapacityX
  - CRoboDacqNet, [453](#)
- GetCardinalDacqSamplerate
  - CInterfaceboardFunctionNet, [189](#)
- GetCardinalStgOutputrate
  - CInterfaceboardFunctionNet, [189](#)
- GetChannel
  - CSw2to64DeviceNet, [577](#)
- GetChannelDataFillSize
  - CMcsUsbDacqNet, [265](#)
- GetChannelData16
  - CCMOSMeaDeviceNet, [111](#)
- GetChannelData132
  - CCMOSMeaDeviceNet, [111](#)
- GetChannelDataUI16
  - CCMOSMeaDeviceNet, [111](#)
- GetChannelDataUI32
  - CCMOSMeaDeviceNet, [111](#)
- GetChannelLayout
  - CMcsUsbDacqNet, [265](#)
- GetChannels
  - CMultiBatteryChargerDeviceNet, [368](#)
  - CSw2to64DeviceNet, [577](#)
- GetChannelsCountOfChamber
  - CWarnerUssingFunctionNet, [625](#)
- GetChannelsInBlock
  - CMcsUsbDacqNet, [265](#)
- GetChannelState
  - CMultiBatteryChargerDeviceNet, [368](#)
- GetChargeCapacity
  - CMultiBatteryChargerDeviceNet, [369](#)
- GetChargeCurrent
  - CMultiBatteryChargerDeviceNet, [369](#)
- GetChargingMode
  - CMultiBatteryChargerDeviceNet, [369](#)
- GetChargingPCoefficient
  - CMultiBatteryChargerDeviceNet, [370](#)
- GetChecksumFromFX3Image
  - CMcsUsbFactoryNet, [294](#)
- GetCheckVoltage

COkuvisionStimulatorDeviceNet, 394  
GetClampAmpSerialNumber  
  CRoboDacqNet, 453  
GetClampMode  
  CTEERFunctionNet, 595  
  CWarnerUssingFunctionNet, 625  
GetCMOSDataDictionary  
  CCMOSMeaDeviceNet, 111  
GetCoilCommunication  
  CPositionIIDeviceNet, 405  
GetColorRgb  
  CMultiwellOptoStimFunctionNet, 385  
GetColorStr  
  CMultiwellOptoStimFunctionNet, 385  
GetCommand  
  CMcsBusNet, 239, 240  
  CPedoterDeviceNet, 399  
  CRoboDacqNet, 453  
GetComplianceVoltageThreshold  
  CWarnerUssingFunctionNet, 625  
GetConfiguration  
  CMcsUsbNet, 317  
GetConfigurationBit  
  CRoboDacqNet, 453  
GetConfigurationBitAxc  
  CRoboDacqNet, 453  
GetConfigurationBitBlu\_Led  
  CRoboDacqNet, 453  
GetConfigurationBitBlu\_LedToggleFast  
  CRoboDacqNet, 453  
GetConfigurationBitBlu\_LedToggleSlow  
  CRoboDacqNet, 453  
GetConfigurationBitCC\_Gen  
  CRoboDacqNet, 454  
GetConfigurationBitCV\_Gen  
  CRoboDacqNet, 454  
GetConfigurationBitRC\_Gen  
  CRoboDacqNet, 454  
GetConfigurationBitRed\_Led  
  CRoboDacqNet, 454  
GetConfigurationBitRed\_LedSaturation  
  CRoboDacqNet, 454  
GetConfigurationBitRed\_LedToggleFast  
  CRoboDacqNet, 454  
GetConfigurationBitRed\_LedToggleSlow  
  CRoboDacqNet, 454  
GetConfigurationBitRelais  
  CRoboDacqNet, 454  
GetConfigurationBitRV\_Gen  
  CRoboDacqNet, 454  
GetConfigurationBits  
  CRoboDacqNet, 454  
GetConfigurationBitStream  
  CRoboDacqNet, 454  
GetConfigurationBitSupply  
  CRoboDacqNet, 455  
GetConnectedDevice  
  CRFFFunctionNet, 443  
GetControllerParams  
  CTEERFunctionNet, 595  
GetCrossTalkOffset  
  CRoboDacqNet, 455  
GetCrossTalkOptimum  
  CRoboDacqNet, 455  
GetCur2VolResistance  
  CGrapheneFunctionNet, 170  
GetCurrent  
  CTcxDeviceNet, 581  
GetCurrentAirvalve  
  CRoboDeviceNet, 467  
GetCurrentAirvalveLimit  
  CRoboDeviceNet, 467  
GetCurrentCycle  
  CMeaCoatDeviceNet, 341  
GetCurrentEditTableNumber  
  CWarnerValveControllerDeviceNet, 644  
GetCurrentEnable  
  CTEERFunctionNet, 595  
GetCurrentFactor  
  COkuvisionStimulatorDeviceNet, 394  
GetCurrentNumberOfValves  
  CWarnerValveControllerDeviceNet, 644  
GetCurrentNumberOfValvesEvent  
  CWarnerValveControllerDeviceNet, 661  
GetCurrentPosition  
  CRoboDeviceNet, 468  
GetCurrentPositionI  
  CRoboStatorDeviceNet, 486  
GetCurrentPositionXY  
  CRoboStatorDeviceNet, 486  
GetCurrentPositionZ  
  CRoboStatorDeviceNet, 487  
GetCurrentRangeInMicroAmpByIndex  
  CStg200xBasicNet, 519  
GetCurrentRangeInNanoAmp  
  CStg200xBasicNet, 519  
  CStimulusFunctionNet, 569  
  CW2100\_StimulatorFunctionNet, 616  
GetCurrentRangeListInMicroAmp  
  CStg200xBasicNet, 520  
GetCurrentRangeSelectedIndex  
  CStg200xBasicNet, 520  
GetCurrentResolutionInNanoAmp  
  CStg200xBasicNet, 520  
  CStimulusFunctionNet, 569  
  CW2100\_StimulatorFunctionNet, 616  
GetCurrentResolutionInNanoAmpByIndex  
  CStg200xBasicNet, 521  
GetCycle  
  CMeaCleanDeviceNet, 337  
GetCycles  
  CMeaCleanDeviceNet, 337  
  CMeaCoatDeviceNet, 341  
GetD  
  CTcxDeviceNet, 582  
GetDacAmplificationFactor

- CStg200xBasicNet, 521
- GetDacIdleValue
  - CLIH3DeviceNet, 194
- GetDACOffset
  - CGrapheneFunctionNet, 171
  - COkuvisionStimulatorDeviceNet, 395
- GetDacOffset
  - CDacCalibrationFunctionNet, 116
  - CLIH3DeviceNet, 194
  - COctoPotDeviceNet, 391
- GetDacPampsPerDigitHighCurrentRange
  - CWarnerUssingFunctionNet, 626
- GetDacPampsPerDigitLowCurrentRange
  - CWarnerUssingFunctionNet, 626
- GetDacqRunStatus
  - CLIH3DeviceNet, 194
- GetDacRange
  - CW2100\_FunctionNet, 610
- GetDACResolution
  - CStg200xBasicNet, 521
  - CStimulusFunctionNet, 569
  - CW2100\_StimulatorFunctionNet, 616
- GetDACs
  - CMcsBus\_SensorNet, 226
- GetDacUsIdleValue
  - CLIH3DeviceNet, 194
- GetDacZero
  - CTEERFunctionNet, 596
  - CWarnerUssingFunctionNet, 626
- GetDataFormat
  - CMcsUsbDacqNet, 265
- GetDataMode
  - CMcsUsbDacqNet, 265
- GetDDecp
  - CTcxDeviceNet, 582
- GetDebugData
  - CPositionIIDeviceNet, 405
- GetDestination
  - CMcsUsbFactoryNet, 294
- GetDestinationCode
  - DriverVersionNet, 678
- GetDestinationDisplayLabel
  - CMcsUsbFactoryNet, 294
- GetDestinationName
  - CMcsUsbFactoryNet, 294, 295
  - DriverVersionNet, 679
- GetDestinationSerialNumber
  - CMcsUsbFactoryNet, 295
- GetDestinationTargetAddress
  - CMcsUsbFactoryNet, 295
- GetDetectionThreshold
  - CMcsBus\_SensorNet, 226
- GetDetectorValue
  - CMcsBus\_SensorNet, 226
- GetDevice
  - CTcxDeviceNet, 582
- GetDeviceCannotStallOutRequests
  - CMcsUsbNet, 317
- GetDeviceCapableSpeed
  - CMcsUsbNet, 317
- GetDeviceEnum
  - CMcsUsbNet, 317
- GetDeviceGroupChannelInfos
  - CDacqGroupChannelSelectionTemplateNet< Dac-  
qGroupChannelEnumTemplateNet, Dac-  
qGroupChannelEnumTemplate, CDevice-  
GroupChannelInfoTemplateNet >, 118, 119
- GetDeviceId
  - CMcsUsbNet, 317
- GetDeviceList
  - CPositionImpDeviceNet, 413
- GetDeviceNames
  - CRadioControlledDevicesNet, 436
- GetDeviceRootHubVendorEnum
  - CMcsUsbNet, 317
- GetDeviceRootHubVendorID
  - CMcsUsbNet, 317
- GetDeviceRootHubVendorName
  - CMcsUsbNet, 317
- GetDeviceSpeed
  - CMcsUsbNet, 318
- GetDeviceType
  - CTcxDeviceNet, 582
- GetDevname
  - CTcxDeviceNet, 582
- GetDigin
  - CFluidControlDeviceNet, 140
- GetDiginState
  - CLIH3DeviceNet, 195
- GetDiginValue
  - CStg200xBasicNet, 521
- GetDigitalData
  - CMeaDigitalDataFunctionNet, 354
- GetDigitalIn
  - CPPCFunctionNet, 420
  - CPPS\_FunctionNet, 427
- GetDigitalOutPortValve
  - CWarnerValveControllerDeviceNet, 645
- GetDigitalOutPortValveEvent
  - CWarnerValveControllerDeviceNet, 661
- GetDigitalPortDirection
  - CWarnerValveControllerDeviceNet, 645
- GetDigitalPortDirectionEvent
  - CWarnerValveControllerDeviceNet, 661
- GetDigitalSource
  - CMcsUsbDacqNet, 266, 267
- GetDigitalStimulatorTrigger
  - CW2100\_StimulatorFunctionNet, 616
- GetDigitalStimulatorTriggerSlope
  - CW2100\_StimulatorFunctionNet, 617
- GetDigout
  - CFluidControlDeviceNet, 141
  - CRoboDacqNet, 455
- GetDigoutMode
  - CStg200xBasicNet, 522
- GetDigoutValue



- CStg200xBasicNet, 522
- GetDIO
  - CMcsBus\_FYIExtensionNet, 203
- GetDischargeCapacity
  - CMultiBatteryChargerDeviceNet, 370
- GetDischargeCurrent
  - CMultiBatteryChargerDeviceNet, 370
- GetDischargeCurrentSetPoint
  - CMultiBatteryChargerDeviceNet, 370
- GetDisplayMode
  - CWarnerValveControllerDeviceNet, 645
- GetDisplayModeEvent
  - CWarnerValveControllerDeviceNet, 661
- GetDisplayText
  - CRoboDacqNet, 455
- GetDMax
  - CTcxDeviceNet, 582
- GetDMin
  - CTcxDeviceNet, 582
- GetDownsampleFactor
  - CRoboDacqNet, 455
- GetDSPHighPassByIndex
  - CIntanMea\_FunctionNet, 185
- GetDuration
  - CMeaCoatDeviceNet, 342
- GetDuty
  - CTcxDeviceNet, 582
- GetEEPromPage
  - CLIH3DeviceNet, 195
- GetElectrodeDacMux
  - CStg200xBasicNet, 522
- GetElectrodeEnable
  - CStg200xBasicNet, 523
- GetElectrodeMode
  - CStg200xBasicNet, 524
- GetEnableAmplifierProtectionSwitch
  - CStg200xBasicNet, 525
- GetEnabledChannelsInGroup
  - CCMOSMea\_FunctionNet, 102
  - CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDevice-GroupChannelInfoTemplateNet >, 119
- GetEnableHeaterLimit
  - CTcxDeviceNet, 583
- GetEnableThermocouple
  - CTcxDeviceNet, 583
- GetEntry
  - CMcsUsbListEntryNet, 304
- GetEntryCount
  - CMcsUsbListEntryNet, 305
- GetEnumerationSpeed
  - CMeaDeviceNet, 349
- GetErrorAirpressure
  - CRoboDeviceNet, 468
- GetErrorCurrentAirvalve
  - CRoboDeviceNet, 468
- GetErrorMessage
  - CMcsUsbDacqNet, 268
- GetErrorText
  - CMcsUsbNet, 318
- GetErrorVoltage12V
  - CRoboDeviceNet, 468
- GetErrorVoltage5V
  - CRoboDeviceNet, 468
- GetErrorVoltageAirvalve
  - CRoboDeviceNet, 468
- GetErrorVoltageRs485A
  - CRoboDeviceNet, 468
- GetErrorVoltageRs485B
  - CRoboDeviceNet, 468
- GetErrorVoltageValves
  - CRoboDeviceNet, 469
- GetEventData
  - CPositionIIDeviceNet, 406
- GetExternalElectrodeEnable
  - CStg200xBasicNet, 526
- GetFAAmplification
  - CStg200xBasicNet, 526
- GetFilter
  - CRoboDacqNet, 455
- GetFilterAttributes
  - CFilterConfigurationNet, 134
- GetFilterCoeffs
  - CRoboDacqNet, 455
- GetFilterParametersHeadstage
  - CWClassicFunctionNet, 668
- GetFilterProperties
  - CSCUFunctionNet, 498
  - CW2100\_FunctionNet, 610
- GetFilterProperty
  - CMcsUsbDacqNet, 268
  - CSCUFunctionNet, 499
  - CW2100\_FunctionNet, 610
- GetFinalDischargeVoltage
  - CMultiBatteryChargerDeviceNet, 371
- GetFirmwareVersion
  - CMcsUsbNet, 318
- GetFirmwareVersionFromFile
  - CMcsUsbFactoryNet, 295
- GetFirmwareVersionFromHexFile
  - CMcsUsbFactoryNet, 295
- GetFPGAFirmwareType
  - CW2100\_FunctionNet, 610
- GetFrameErrorCounter
  - CTEERFunctionNet, 596
- GetFrequency
  - CRadioControlledDevicesNet, 437
- GetFrequencyRange
  - CPgaDeviceNet, 403
- GetGain
  - CMeaDeviceNet, 349
  - CPgaDeviceNet, 403
- GetGate
  - CCMOSMea\_FunctionNet, 102
- GetGilsonDevice

- CRobocyte2DeviceNet, [484](#)
- GetGlobalRepeat
  - CDigOutStimulatorFunctionNet, [125](#)
- GetGNDI
  - CCMOSMea\_FunctionNet, [102](#)
- GetGroupADCBits
  - CCMOSMea\_FunctionNet, [102](#)
- GetGroupChannelBitmaskBySelect
  - CCMOSMea\_FunctionNet, [102](#)
- GetGroupChannelBitmaskHS1NCBathCurrent
  - CCMOSMea\_FunctionNet, [102](#), [103](#)
- GetGroupChannelBitmaskHS1NCCol2Current
  - CCMOSMea\_FunctionNet, [103](#)
- GetGroupChannelBitmaskHS1NChipTemp
  - CCMOSMea\_FunctionNet, [103](#)
- GetGroupChannelBitmaskHS1Sidebands
  - CCMOSMea\_FunctionNet, [103](#)
- GetGroupChannelBitmaskHS1TriggerStatus
  - CCMOSMea\_FunctionNet, [103](#), [104](#)
- GetGroupChannelBitmaskIFDigChannels
  - CCMOSMea\_FunctionNet, [104](#)
- GetGroupChannelBitmaskInterfaceADC
  - CCMOSMea\_FunctionNet, [104](#)
- GetGroupChannelBitmaskPacketFrameContext
  - CCMOSMea\_FunctionNet, [104](#)
- GetGroupChannelBitmaskSTG1DACSignal
  - CCMOSMea\_FunctionNet, [104](#), [105](#)
- GetGroupChannelDataI16
  - CMcsUsbDacqNet, [268](#)
- GetGroupChannelDataI32
  - CMcsUsbDacqNet, [268](#)
- GetGroupChannelDataUI16
  - CMcsUsbDacqNet, [269](#)
- GetGroupChannelDataUI32
  - CMcsUsbDacqNet, [269](#)
- GetGroupDCOffset
  - CCMOSMea\_FunctionNet, [105](#)
- GetGroupID
  - CCMOSMea\_FunctionNet, [105](#)
  - CDacqGroupChannelSelectionTemplateNet< Dac-  
qGroupChannelEnumTemplateNet, Dac-  
qGroupChannelEnumTemplate, CDevice-  
GroupChannelInfoTemplateNet >, [119](#)
- GetGroupNumberOfChannels
  - CCMOSMea\_FunctionNet, [105](#)
  - CDacqGroupChannelSelectionTemplateNet< Dac-  
qGroupChannelEnumTemplateNet, Dac-  
qGroupChannelEnumTemplate, CDevice-  
GroupChannelInfoTemplateNet >, [119](#)
- GetGroupResolutionPerDigit
  - CCMOSMea\_FunctionNet, [105](#)
- GetGroupSampleSize
  - CCMOSMea\_FunctionNet, [106](#)
  - CDacqGroupChannelSelectionTemplateNet< Dac-  
qGroupChannelEnumTemplateNet, Dac-  
qGroupChannelEnumTemplate, CDevice-  
GroupChannelInfoTemplateNet >, [119](#)
- GetGroupType
  - CCMOSMea\_FunctionNet, [106](#)
- CDacqGroupChannelSelectionTemplateNet< Dac-  
qGroupChannelEnumTemplateNet, Dac-  
qGroupChannelEnumTemplate, CDevice-  
GroupChannelInfoTemplateNet >, [120](#)
- GetGroupUnit
  - CCMOSMea\_FunctionNet, [106](#)
- GetGyroRange
  - CW2100\_FunctionNet, [610](#)
- GetHardwareMaxRange
  - CMcsUsbDacqNet, [270](#)
- GetHardwareMinRange
  - CMcsUsbDacqNet, [270](#)
- GetHardwareRevision
  - CMcsUsbNet, [318](#)
- GetHasChecksum
  - CWClassicFunctionNet, [668](#)
- GetHashCode
  - HeadstageIDTypeObject, [689](#)
- GetHasRedLedHeadstage
  - CWClassicFunctionNet, [668](#)
- GetHasThermocouple
  - CTcxDeviceNet, [583](#)
- GetHeadstage
  - CStg200xBasicNet, [526](#)
- GetHeadstageActive
  - CMcsUsbNet, [319](#)
- GetHeadstageAdcBits
  - CSCUFunctionNet, [499](#)
- GetHeadstageAdcRangeInMicroVolt
  - CSCUFunctionNet, [499](#)
- GetHeadstageDacBits
  - CSCUFunctionNet, [500](#)
- GetHeadstageDacCurrentRangeInMicroAmpere
  - CSCUFunctionNet, [500](#)
- GetHeadstageDacCurrentResolutionInNanoAmpere
  - CSCUFunctionNet, [500](#)
- GetHeadstageDacVoltageRangeInMilliVolt
  - CSCUFunctionNet, [501](#)
- GetHeadstageDacVoltageResolutionInMicroVolt
  - CSCUFunctionNet, [501](#)
- GetHeadstageFrameCyclesToCompare
  - CSCUFunctionNet, [501](#)
- GetHeadstageGainInPer mille
  - CSCUFunctionNet, [502](#)
- GetHeadstageID
  - CMcsUsbNet, [319](#)
  - CSCUFunctionNet, [502](#)
- GetHeadstageLinkSpeed
  - CSCUFunctionNet, [502](#)
- GetHeadstageNumberOfAnalogChannels
  - CSCUFunctionNet, [503](#)
- GetHeadstageNumberOfStimulationChannels
  - CSCUFunctionNet, [503](#)
- GetHeadstageOnOff
  - CW2100\_FunctionNet, [610](#)
  - CWClassicFunctionNet, [668](#)
- GetHeadstagePowerStateAtStart



- CSCUFunctionNet, [503](#)
- GetHeadstagePresent
  - CMcsUsbNet, [319](#)
- GetHeadstageSamplerate
  - CSCUFunctionNet, [504](#)
- GetHeadstageSamplingActive
  - CW2100\_FunctionNet, [610](#)
- GetHeadstageSerialNumber
  - CSCUFunctionNet, [504](#)
- GetHeaterLimit
  - CTcxDeviceNet, [583](#)
- GetHeaterTemp
  - CTcxDeviceNet, [583](#)
- GetHighCurrentRange
  - CWarnerUssingFunctionNet, [627](#)
- GetHighpassFilterEnable
  - CFilterConfigurationNet, [134](#)
- GetHWConfig
  - CRoboDeviceNet::RoboMainLowLevelCommands, [695](#)
- GetHWRRevision
  - CRoboDeviceNet::RoboMainLowLevelCommands, [695](#)
- GetHWRRevisionEeprom
  - CMcsBusNet, [240](#)
- GetI
  - CTcxDeviceNet, [583](#)
- GetIC
  - CRoboDacqNet, [455](#)
- GetIClamp
  - CRoboDacqNet, [455](#)
- GetICoeff
  - CRobo\_FYITemp\_FunctionNet, [447](#)
- GetICOffset
  - CRoboDacqNet, [455](#)
- GetIDecp
  - CTcxDeviceNet, [583](#)
- GetIdleModeOffset
  - CWarnerUssingFunctionNet, [627](#)
- GetIGain
  - CRoboDacqNet, [456](#)
- GetIMax
  - CTcxDeviceNet, [584](#)
- GetIMin
  - CTcxDeviceNet, [584](#)
- GetImpedanceResult
  - CIntanMea\_FunctionNet, [185](#)
- GetImpedanceTestFrequency
  - CMealImpedanceDeviceNet, [360](#)
- GetImpId
  - CPositionImpDeviceNet, [414](#)
- GetImplantCurrentSetpoint
  - CPositionIIDeviceNet, [406](#)
- GetImplantResult
  - CPositionIIDeviceNet, [407](#)
- GetImplantState
  - CPositionIIDeviceNet, [407](#)
- GetInMovement
  - CRoboDeviceNet, [469](#)
- GetIntanRegister
  - CIntanMea\_FunctionNet, [185](#)
- GetIntBuffer
  - CGenericDevelopDeviceNet, [155](#)
- GetIO
  - CWarnerValveControllerDeviceTesterFunctionNet, [665](#)
- GetIOut
  - CTcxDeviceNet, [584](#)
- GetIoVoltage
  - CInterfaceboard2FunctionNet, [187](#)
- GetLastAnswer
  - CGilsonDeviceNet, [166](#)
- GetLastUSBError
  - CMcsUsbNet, [319](#)
- GetLatency
  - CMcsBus\_SensorNet, [226](#)
- GetLatencyCounter
  - CMcsBus\_SensorNet, [226](#)
- GetLayoutConfiguration
  - CMEA2100x256FunctionNet, [332](#)
- GetLEDSwitch
  - CMcsBus\_ExtensionNet, [202](#)
- GetLength
  - CRobo\_FYIProgram\_FunctionNet, [445](#)
- GetLiquidResistance
  - CTEERFunctionNet, [596](#)
  - CWarnerUssingFunctionNet, [627](#)
- GetListModelIndexRange
  - CStg200xBasicNet, [526](#)
- GetListModelTriggerSource
  - CStg200xBasicNet, [527](#)
- GetLowCurrentRange
  - CWarnerUssingFunctionNet, [628](#)
- GetLowerFrequencyByIndex
  - CIntanMea\_FunctionNet, [185](#)
- GetMajor
  - DriverVersionNet, [679](#)
- GetMaxChunkSize\_Byte
  - CTEERFunctionNet, [596](#)
- GetMaxCurrent
  - CMeaCoatDeviceNet, [342](#)
- GetMaxDurationHighCurrentInMicroSec
  - CMultiwellOptoStimFunctionNet, [386](#)
- GetMaxDutyCycleHighCurrent
  - CMultiwellOptoStimFunctionNet, [386](#)
- GetMaxHeaterPowerMultiwell
  - CTcxDeviceNet, [584](#)
- GetMaxNoPressure
  - CRoboDeviceNet::RoboMainLowLevelCommands, [695](#)
- GetMaxNoPressureWaitTime
  - CRoboDeviceNet::RoboMainLowLevelCommands, [695](#)
- GetMaxNumberOfHeadstages
  - CSCUFunctionNet, [504](#)
- GetMaxNumOfColumns

- CCMOSMea\_FunctionNet, 106
- GetMaxP
  - CTcxDeviceNet, 584
- GetMaxpDecp
  - CTcxDeviceNet, 584
- GetMaxpMax
  - CTcxDeviceNet, 584
- GetMaxpMin
  - CTcxDeviceNet, 585
- GetMaxPower
  - COkuvisionStimulatorDeviceNet, 395
  - CRobo\_FYITemp\_FunctionNet, 447
- GetMaxPressureWaitTime
  - CRoboDeviceNet::RoboMainLowLevelCommands, 695
- GetMaxReadableColumns
  - CCMOSMeaDeviceNet, 111
- GetMaxSamplingFrequency
  - CMcsUsbDacqNet, 270
- GetMaxStimulusChannelsPerHeadstage
  - CSCUFunctionNet, 504
- GetMaxVoltage
  - CMeaCleanDeviceNet, 337
  - COkuvisionStimulatorDeviceNet, 395
- GetMCAcceleration
  - CMcsBus\_MotorControlNet, 207
- GetMCAccelerationEeprom
  - CMcsBus\_MotorControlNet, 207
- GetMCAccelerationShortCommand
  - CMcsBus\_MotorControlNet, 207
- GetMCAXisRevisionEeprom
  - CMcsBus\_MotorControlNet, 207
- GetMCBreakCurrent
  - CMcsBus\_MotorControlNet, 207
- GetMCBreakCurrentEeprom
  - CMcsBus\_MotorControlNet, 208
- GetMCConfig
  - CMcsBus\_MotorControlNet, 208
- GetMCConfigEeprom
  - CMcsBus\_MotorControlNet, 208
- GetMCCurrent
  - CMcsBus\_MotorControlNet, 208
- GetMCCurrentEeprom
  - CMcsBus\_MotorControlNet, 208
- GetMCCurrentMode
  - CMcsBus\_MotorControlNet, 208
- GetMCCurrentModeEeprom
  - CMcsBus\_MotorControlNet, 208
- GetMCCurrentModeShortCommand
  - CMcsBus\_MotorControlNet, 209
- GetMCCurrentPosition
  - CMcsBus\_MotorControlNet, 209
- GetMCCurrentShortCommand
  - CMcsBus\_MotorControlNet, 209
- GetMCCurrentSpeed
  - CMcsBus\_MotorControlNet, 209
- GetMCMaxAcceleration
  - CMcsBus\_MotorControlNet, 209
- GetMCMaxAccelerationEeprom
  - CMcsBus\_MotorControlNet, 209
- GetMCMaxCurrent
  - CMcsBus\_MotorControlNet, 209
- GetMCMaxCurrentEeprom
  - CMcsBus\_MotorControlNet, 210
- GetMCMaxSpeed
  - CMcsBus\_MotorControlNet, 210
- GetMCMaxSpeedEeprom
  - CMcsBus\_MotorControlNet, 210
- GetMCMaxTravel
  - CMcsBus\_MotorControlNet, 210
- GetMCMaxTravelEeprom
  - CMcsBus\_MotorControlNet, 210
- GetMCMaxTravelShortCommand
  - CMcsBus\_MotorControlNet, 210
- GetMCMovement
  - CMcsBus\_MotorControlNet, 210
- GetMCNewPosition
  - CMcsBus\_MotorControlNet, 211
- GetMCOutputOnOff
  - CMcsBus\_MotorControlNet, 211
- GetMCPhase
  - CMcsBus\_MotorControlNet, 211
- GetMCPhaseOffset
  - CMcsBus\_MotorControlNet, 211
- GetMCReference
  - CMcsBus\_MotorControlNet, 211
- GetMCReferenceCurrent
  - CMcsBus\_MotorControlNet, 211
- GetMCReferenceCurrentEeprom
  - CMcsBus\_MotorControlNet, 211
- GetMCRegulatorGain
  - CMcsBus\_MotorControlNet, 212
- GetMCRegulatorGainEeprom
  - CMcsBus\_MotorControlNet, 212
- GetMcsBus\_Extension
  - CRoboocyte2DeviceNet, 484
- GetMCScalingFactor
  - CMcsBus\_MotorControlNet, 212
- GetMCScalingFactorEeprom
  - CMcsBus\_MotorControlNet, 212
- GetMCSpeed
  - CMcsBus\_MotorControlNet, 212
- GetMCSpeedEeprom
  - CMcsBus\_MotorControlNet, 212
- GetMCSpeedShortCommand
  - CMcsBus\_MotorControlNet, 212
- GetMCSpeedUnitEeprom
  - CMcsBus\_MotorControlNet, 213
- GetMCStandbyCurrent
  - CMcsBus\_MotorControlNet, 213
- GetMCStandbyCurrentEeprom
  - CMcsBus\_MotorControlNet, 213
- GetMCStandbyTime
  - CMcsBus\_MotorControlNet, 213
- GetMCStandbyTimeEeprom
  - CMcsBus\_MotorControlNet, 213

- GetMea21UsbPort
  - CMcsUsbNet, [320](#)
- GetMeaLayout
  - CMcsUsbDacqNet, [270](#)
- GetMemoryUsageDAC
  - CStg200xDownloadBasicNet, [553](#)
- GetMemoryUsageSyncout
  - CStg200xDownloadBasicNet, [553](#)
- GetMinimalThreshold
  - CMcsBus\_SensorNet, [226](#)
- GetMinNoPressureWaitTime
  - CRoboDeviceNet::RoboMainLowLevelCommands, [695](#)
- GetMinor
  - DriverVersionNet, [680](#)
- GetMinPressure
  - CRoboDeviceNet, [469](#)
  - CRoboDeviceNet::RoboMainLowLevelCommands, [695](#)
- GetMinPressureWaitTime
  - CRoboDeviceNet::RoboMainLowLevelCommands, [696](#)
- GetMinSamplingFrequencyStepsize
  - CMcsUsbDacqNet, [271](#)
- GetMinVoltage
  - CMeaCleanDeviceNet, [338](#)
- GetModeSelect
  - CPulseGeneratorFunctionNet, [434](#)
- GetModuleCurrent
  - CStg200xDownloadNet, [561](#)
- GetModuleTemp
  - CStg200xDownloadNet, [561](#)
- GetMovementError
  - CRoboDeviceNet, [469](#)
- GetMovePump
  - CMcsBus\_SensorNet, [227](#)
- GetMultiHeadstageMode
  - CW2100\_FunctionNet, [610](#)
- GetMultiplexedDataChannelsInBlock
  - CStimulusFunctionNet, [570](#)
- GetNanoVoltsPerKelvin
  - CMcsBus\_TempSensorNet, [232](#)
- GetNeurochipMemoryData
  - CCMOSMea\_FunctionNet, [106](#)
- GetNeurochipMemorySize
  - CCMOSMea\_FunctionNet, [107](#)
- GetNIC\_MS
  - CRoboDacqNet, [456](#)
- GetNUC\_MS
  - CRoboDacqNet, [456](#)
- GetNumAmplifications
  - CPgaDeviceNet, [403](#)
- GetNumber
  - CMeaSwitchDeviceNet, [363](#)
  - CSw2to64DeviceNet, [577](#)
- GetNumberOfAnalogChannels
  - CStg200xBasicNet, [527](#)
  - CStimulusFunctionNet, [570](#)
  - CW2100\_StimulatorFunctionNet, [617](#)
- GetNumberOfAudioChannels
  - CMeaAudioFunctionNet, [334](#)
- GetNumberOfAvailableChambers
  - CWarnerUssingFunctionNet, [628](#)
- GetNumberOfAvailableSamples
  - CTEERFunctionNet, [596](#)
- GetNumberOfChannels
  - CDigOutStimulatorFunctionNet, [126](#)
- GetNumberOfCurrentRangeIndexes
  - CStg200xBasicNet, [527](#)
- GetNumberOfDataBits
  - CMcsUsbDacqNet, [271](#)
- GetNumberOfDevices
  - CMcsUsbListNet, [308](#)
- GetNumberOfHardwareSlotsForChambers
  - CWarnerUssingFunctionNet, [628](#)
- GetNumberOfHWADCCChannels
  - CMcsUsbDacqNet::CHWInfo, [183](#)
- GetNumberOfHWDACPaths
  - CStg200xBasicNet, [527](#)
- GetNumberOfHWDigitalChannels
  - CMcsUsbDacqNet::CHWInfo, [183](#)
- GetNumberOfStimulationElectrodes
  - CStg200xBasicNet, [528](#)
- GetNumberOfStimulationSourcesPerElectrode
  - CStg200xBasicNet, [528](#)
- GetNumberOfSupportedGroups
  - CCMOSMea\_FunctionNet, [107](#)
  - CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet >, [120](#)
- GetNumberOfSyncoutChannels
  - CStg200xBasicNet, [528](#)
  - CW2100\_StimulatorFunctionNet, [617](#)
- GetNumberOfTriggerInputs
  - CStg200xBasicNet, [528](#)
  - CW2100\_StimulatorFunctionNet, [617](#)
- GetNumberOfVoltageRangeIndexes
  - CStg200xBasicNet, [528](#)
- GetNumConfigurations
  - CMcsUsbNet, [320](#)
- GetNumControlChannels
  - CTcxDeviceNet, [585](#)
- GetNumDestinations
  - CMcsUsbFactoryNet, [295](#)
- GetNumDevices
  - CTcxDeviceNet, [585](#)
- GetNumEntries
  - DriverVersionNet, [680](#)
- GetNumFrequencyRanges
  - CPgaDeviceNet, [403](#)
- GetNumMeasureChannels
  - CTcxDeviceNet, [585](#)
- GetNUV\_MS
  - CRoboDacqNet, [456](#)
- GetOffsetCurrent

- CMeaCoatDeviceNet, [342](#)
- GetOnOff
  - CPositionIIDeviceNet, [407](#)
  - CTcxDeviceNet, [585](#)
- GetOutputCurrent
  - CMeaCoatDeviceNet, [342](#)
- GetOutputRate
  - CStg200xBasicNet, [529](#)
- GetOutputVoltage
  - CMeaCleanDeviceNet, [338](#)
- GetP
  - CTcxDeviceNet, [585](#)
- GetParameter
  - CRoboDeviceNet::RoboMainLowLevelCommands, [696](#)
- GetPattern
  - CMeaSwitchDeviceNet, [363](#)
- GetPatternBool
  - CMeaSwitchDeviceNet, [363](#)
- GetPauseDuration
  - CMeaCoatDeviceNet, [342](#)
- GetPCoeff
  - CRobo\_FYITemp\_FunctionNet, [447](#)
- GetPDecp
  - CTcxDeviceNet, [585](#)
- GetPeriod
  - CPulseGeneratorFunctionNet, [434](#)
- GetPeriod\_us
  - CTEERFunctionNet, [597](#)
- GetPermanentCurrentInMicroAmp
  - CMultiwellOptoStimFunctionNet, [386](#)
- GetPGain
  - CRoboDacqNet, [456](#)
- GetPhases
  - CRoboDeviceNet::RoboMainLowLevelCommands, [696](#)
- GetPicFirmwareType
  - CW2100\_FunctionNet, [610](#)
- GetPiezoState
  - CMcsBus\_SensorNet, [227](#)
- GetPlateClampLockState
  - CMultiwellDeviceNet, [378](#)
- GetPlateClampState
  - CMultiwellDeviceNet, [378](#)
- GetPlateClampStateByHeadstage
  - CMultiwellCallbackFunctionNet, [375](#)
- GetPlateClampStateByHeadstageEvent
  - CMultiwellCallbackFunctionNet, [376](#)
- GetPlateMux
  - CMultiwellDeviceNet, [379](#)
- GetPlateType
  - CMultiwellDeviceNet, [379](#)
- GetPMax
  - CTcxDeviceNet, [585](#)
- GetPMin
  - CTcxDeviceNet, [585](#)
- GetPoti
  - CMcsUsbDacqNet, [271](#)
- GetPOut
  - CTcxDeviceNet, [585](#)
- GetPowerMuxPlate
  - CMultiwellDeviceNet, [380](#)
- GetPowerStrength
  - CPositionIIDeviceNet, [408](#)
- GetPressure
  - CMcsBus\_SensorNet, [227](#)
- GetPressureOffset
  - CMcsBus\_SensorNet, [227](#)
- GetPressureRange
  - CPPCFunctionNet, [420](#)
- GetPulseform
  - COkuvisionStimulatorDeviceNet, [395](#)
- GetPulseLength
  - CPulseGeneratorFunctionNet, [434](#)
- GetPumpCouple
  - CPPS\_FunctionNet, [427](#)
- GetPumpEnableSpeedRatio
  - CPPS\_FunctionNet, [427](#)
- GetPumpFastOnOff
  - CPPS\_FunctionNet, [427](#)
- GetPumpFastSpeed
  - CPPS\_FunctionNet, [427](#)
- GetPumpFunctionSpeeds
  - CPPS\_FunctionNet, [428](#)
- GetPumpManualOnOff
  - CPPS\_FunctionNet, [428](#)
- GetPumpMaxSpeed
  - CPPS\_FunctionNet, [428](#)
- GetPumpModeType
  - CPPCFunctionNet, [420](#)
  - CPPS\_FunctionNet, [428](#)
- GetPumpSpeed
  - CRoboFluidDeviceNet, [480](#)
- GetPumpSpeedRatio
  - CPPS\_FunctionNet, [428](#)
- GetPumpSpeedUnit
  - CPPCFunctionNet, [421](#)
  - CPPS\_FunctionNet, [428](#)
- GetPWM
  - CFluidControlDeviceNet, [141](#)
- GetPwrOut
  - CTcxDeviceNet, [586](#)
- GetPwrSet
  - CTcxDeviceNet, [586](#)
- GetRatedCapacity
  - CMultiBatteryChargerDeviceNet, [371](#)
- GetReady
  - CMealImpedanceDeviceNet, [360](#)
- GetRecordingNumber
  - CRoboDacqNet, [456](#)
- GetReferenceElectrodeMode
  - CSCUFunctionNet, [505](#)
- GetReferenceElectrodeSwitchState
  - CSCUFunctionNet, [505](#)
- GetReferenceTemperature
  - CFluidControlDeviceNet, [141](#)

- GetRegulationTimeouts
  - CMcsBus\_SensorNet, [227](#)
- GetRegulatorFactor
  - CMcsBus\_SensorNet, [228](#)
- GetRegulatorOnOff
  - CMcsBus\_SensorNet, [228](#)
  - CRobo\_FYITemp\_FunctionNet, [447](#)
- GetRegulatorStatus
  - CMcsBus\_SensorNet, [228](#)
- GetRepeats
  - CProgramPressureCurveNet, [432](#)
- GetRes1
  - CTcxDeviceNet, [586](#)
- GetRes2
  - CTcxDeviceNet, [586](#)
- GetResetFilter
  - CWClassicFunctionNet, [668](#)
- GetResistanceC
  - CRoboDacqNet, [456](#)
- GetResistanceV
  - CRoboDacqNet, [456](#)
- GetResolutionPerDigit
  - CMcsUsbDacqNet, [271](#)
- GetResS
  - CTcxDeviceNet, [586](#)
- GetResult
  - CMealImpedanceDeviceNet, [360](#)
- GetResX
  - CTcxDeviceNet, [586](#)
- GetRFConnectionStatus
  - CWClassicFunctionNet, [668](#)
- GetRFFrequency
  - CPositionImpDeviceNet, [414](#)
- GetRFFrequencyHeadstage
  - CWClassicFunctionNet, [668](#)
- GetRFFrequencyReceiver
  - CWClassicFunctionNet, [668](#)
- GetRFPower
  - CWClassicFunctionNet, [669](#)
- GetRoboDacq
  - CRoboocyte2DeviceNet, [484](#)
- GetRoboFluidDevice
  - CEncapsulatorDeviceNet, [129](#)
  - CRoboocyte2DeviceNet, [484](#)
- GetRotaryPositionCode
  - CTEERFunctionNet, [597](#)
- GetRotatePump
  - CMcsBus\_SensorNet, [228](#)
- GetROut
  - CTcxDeviceNet, [586](#)
- GetRTC
  - CokuvisionStimulatorDeviceNet, [395](#)
  - CPositionIIDeviceNet, [408](#)
- GetSampleBufferChunk
  - CTEERFunctionNet, [597](#)
- GetSampleInterval
  - CLIH3DeviceNet, [195](#)
- GetSamplePeriode
  - CMcsBus\_SensorNet, [228](#)
- GetSampleRate
  - CTEERFunctionNet, [597](#)
- GetSamplerate
  - CMcsUsbDacqNet, [271](#)
- GetSampleVoltageBuffer\_uV
  - CTEERFunctionNet, [598](#)
- GetScaleFactorU1
  - CTEERFunctionNet, [598](#)
- GetScaleFactorU2
  - CTEERFunctionNet, [598](#)
- GetScanHeadstagesResult
  - CWClassicFunctionNet, [669](#)
- GetScreen
  - CRoboDacqNet, [456](#)
- GetSearchReferenceFastAccel
  - CRoboDeviceNet::RoboMainLowLevelCommands, [696](#)
- GetSearchReferenceFastSpeed
  - CRoboDeviceNet::RoboMainLowLevelCommands, [696](#)
- GetSearchReferenceFineAccel
  - CRoboDeviceNet::RoboMainLowLevelCommands, [696](#)
- GetSearchReferenceFineSpeed
  - CRoboDeviceNet::RoboMainLowLevelCommands, [696](#)
- GetSearchReferenceMethod
  - CRoboDeviceNet::RoboMainLowLevelCommands, [697](#)
- GetSearchReferenceMoveOut
  - CRoboDeviceNet::RoboMainLowLevelCommands, [697](#)
- GetSearchReferenceOffsetPos
  - CRoboDeviceNet::RoboMainLowLevelCommands, [697](#)
- GetSelectedChannels
  - CW2100\_FunctionNet, [611](#)
- GetSelectedHeadstage
  - CWClassicFunctionNet, [669](#)
- GetSelectedHeadstageState
  - CW2100\_FunctionNet, [611](#)
- GetSensorType
  - CTcxDeviceNet, [587](#)
- GetSerialNumber
  - CMcsUsbNet, [320](#)
  - DriverVersionNet, [680](#)
- GetSerialNumberHeadstage
  - CWClassicFunctionNet, [669](#)
- GetSetpoint
  - CTcxDeviceNet, [587](#)
- GetSetpointDecp
  - CTcxDeviceNet, [587](#)
- GetSetpointMax
  - CTcxDeviceNet, [587](#)
- GetSetpointMin
  - CTcxDeviceNet, [587](#)
- GetShortBuffer

- CGenericDevelopDeviceNet, [156](#)
- GetSimulation
  - CRoboDacqNet, [456](#)
- GetSingleHeater
  - CMcsBus\_FYIExtensionNet, [203](#)
- GetSingleValve
  - CFluidControlDeviceNet, [141](#)
  - CRoboFluidDeviceNet, [480](#)
- GetSlope
  - CMeaCleanDeviceNet, [338](#)
  - CMeaCoatDeviceNet, [343](#)
- GetSoftwareKey
  - CMcsUsbNet, [320](#)
- GetSoftwareKeyString
  - CMcsUsbNet, [320](#)
- GetSollPressure
  - CMcsBus\_SensorNet, [228](#)
- GetSollTemp
  - CRobo\_FYITemp\_FunctionNet, [447](#)
- GetSourceBulk
  - CCMOSMea\_FunctionNet, [107](#)
- GetSourceDrain
  - CCMOSMea\_FunctionNet, [107](#)
- GetSourceGate
  - CCMOSMea\_FunctionNet, [107](#)
- GetStartTriggerSlope
  - CDigOutStimulatorFunctionNet, [126](#)
- GetState
  - CRFFFunctionNet, [443](#)
  - CRobo\_FYIProgram\_FunctionNet, [446](#)
- GetStateDebugData
  - CPositionIIDeviceNet, [408](#)
- GetStateEventData
  - CPositionIIDeviceNet, [409](#)
- GetStatus
  - CMcsUsbNet, [320](#)
  - DriverVersionNet, [681](#)
- GetStatusOfLastCommand
  - CMcsUsbNet, [320](#)
- GetStgProgramInfo
  - CStg200xBasicNet, [529](#)
- GetStgVersionInfo
  - CStg200xBasicNet, [530](#)
- GetStimulationPatternMemory
  - CW2100\_StimulatorFunctionNet, [617](#)
- GetStimulatorStatus
  - CokuvisionStimulatorDeviceNet, [395](#)
- GetStimulusParametersCache
  - CW2100\_FunctionNet, [611](#)
- GetStimulusParametersFromSelectedHS
  - CW2100\_FunctionNet, [611](#)
- GetStimulusSites
  - CCMOSMea\_FunctionNet, [107](#)
- GetStimulusParameters
  - CW2100\_FunctionNet, [611](#)
- GetStopTriggerSlope
  - CDigOutStimulatorFunctionNet, [126](#)
- GetSubChannel
  - CMcsBus\_MotorControlNet, [213](#)
- GetSupplyVoltage
  - CPPCFunctionNet, [421](#)
  - CPPS\_FunctionNet, [428](#)
- GetSweepCount
  - CStg200xDownloadBasicNet, [554](#)
- GetSync
  - CWarnerValveControllerDeviceTesterFunctionNet, [665](#)
- GetSyncoutMap
  - CStg200xBasicNet, [530](#)
- GetSyncState
  - CMcsBus\_SensorNet, [228](#)
- GetTableName
  - CWarnerValveControllerDeviceNet, [645](#)
- GetTableNamebyIndex
  - CWarnerValveControllerDeviceNet, [646](#)
- GetTableNamebyIndexEvent
  - CWarnerValveControllerDeviceNet, [661](#)
- GetTablepointer
  - CRetinaLedDeviceNet, [439](#)
- GetTemperatur
  - CMcsBus\_TempSensorNet, [232](#)
- GetTestMode
  - CRFFFunctionNet, [443](#)
- GetThermocoupleCalibration
  - CFluidControlDeviceNet, [142](#)
  - CTcxDeviceNet, [587](#)
- GetThermocoupleNanovoltPerKelvin
  - CFluidControlDeviceNet, [142](#)
  - CTcxDeviceNet, [587](#)
- GetThermocoupleReferenceTemp
  - CTcxDeviceNet, [588](#)
- GetThermocoupleTemp
  - CTcxDeviceNet, [588](#)
- GetThermocoupleTempAbs
  - CTcxDeviceNet, [588](#)
- GetThermocoupleTemperature
  - CFluidControlDeviceNet, [142](#)
- GetThermoOffset
  - CMcsBus\_TempSensorNet, [232](#)
- GetThermoTemp
  - CMcsBus\_TempSensorNet, [233](#)
- GetThermoVoltage
  - CMcsBus\_TempSensorNet, [233](#)
- GetTimeInPause
  - CMeaCoatDeviceNet, [343](#)
- GetTimeInPlateau
  - CMeaCoatDeviceNet, [343](#)
- GetTimeResolutionInNanoSeconds
  - CW2100\_StimulatorFunctionNet, [617](#)
- GetTimeSlot
  - CW2100\_StimulatorFunctionNet, [617](#)
- GetTotalMemory
  - CStg200xBasicNet, [530](#)
  - CStimulusFunctionNet, [570](#)
- GetTotalNumberOfDigitalPorts
  - CWarnerValveControllerDeviceNet, [646](#)



- GetTotalNumberOfTables
  - CWarnerValveControllerDeviceNet, 646
- GetTotalNumberOfValves
  - CWarnerValveControllerDeviceNet, 646
- GetTotalTableSize
  - CWarnerValveControllerDeviceNet, 647
- GetTouchPadEnable
  - CMultiwellDeviceNet, 380
- GetTrigger
  - CStg200xDownloadBasicNet, 554
- GetTriggerSource
  - CStg200xBasicNet, 530
- GetU1Offset
  - CWarnerUssingFunctionNet, 628
- GetU1Reference
  - CWarnerUssingFunctionNet, 630
- GetU2Offset
  - CWarnerUssingFunctionNet, 630
- GetU2Reference
  - CWarnerUssingFunctionNet, 630
- GetUByteBuffer
  - CGenericDevelopDeviceNet, 157
- GetUC
  - CRoboDacqNet, 456
- GetUClamp
  - CRoboDacqNet, 457
- GetUCOffset
  - CRoboDacqNet, 457
- GetUIntA
  - CFilterCoefficientsNet, 132
- GetUIntB
  - CFilterCoefficientsNet, 132
- GetUIntBuffer
  - CGenericDevelopDeviceNet, 157
- GetUnit
  - CTcxDeviceNet, 588
- GetUnitDescription
  - CWarnerUssingFunctionNet, 631
- GetUnitExponent
  - CWarnerUssingFunctionNet, 631
- GetUnitName
  - CWarnerUssingFunctionNet, 631
- GetUnitsPerDigit
  - CWarnerUssingFunctionNet, 632
- GetUOut
  - CTcxDeviceNet, 588
- GetUpdateDisplay
  - CRoboDacqNet, 457
- GetUpperFrequencyByIndex
  - CIntanMea\_FunctionNet, 186
- GetUptimeSeconds
  - CTEERFunctionNet, 598
  - CWarnerUssingFunctionNet, 632
- GetUSBDeviceIDFromFX3Image
  - CMcsUsbFactoryNet, 296
- GetUsbListEntries
  - CMcsUsbListNet, 308
- GetUsbListEntry
  - CMcsUsbListNet, 308
  - CMcsUsbNet, 321
- GetUseBubble
  - CPPS\_FunctionNet, 428
- GetUserCodeFromBitFile
  - CMcsUsbFactoryNet, 296
- GetUserCodeFromFlash
  - CMcsUsbFactoryNet, 296
- GetUserDefinedName
  - CW2100\_FunctionNet, 611
- GetUserDefinedNameCache
  - CW2100\_FunctionNet, 611
- GetUserDefinedNameFromSelectedHS
  - CW2100\_FunctionNet, 612
- GetUserParameter
  - CRoboDeviceNet::RoboMainLowLevelCommands, 697
- GetUShortBuffer
  - CGenericDevelopDeviceNet, 158
- GetUV
  - CRoboDacqNet, 457
- GetUVOffset
  - CRoboDacqNet, 457
- GetValue
  - CTcxDeviceNet, 588
- GetValueHires
  - CTcxDeviceNet, 589
- GetValve
  - CFluidControlDeviceNet, 143
  - CRoboFluidDeviceNet, 480
- GetValve1
  - CRobo\_FYIProgram\_FunctionNet, 446
- GetValve2
  - CRobo\_FYIProgram\_FunctionNet, 446
- GetValveActive
  - CPPCFunctionNet, 421
  - CWarnerValveControllerDeviceNet, 647
- GetValveActiveEvent
  - CWarnerValveControllerDeviceNet, 661
- GetValveBoardRevision
  - CWarnerValveControllerDeviceNet, 647
- GetValveBoardRevisionEvent
  - CWarnerValveControllerDeviceNet, 661
- GetValveBoardRevisionString
  - CWarnerValveControllerDeviceNet, 647
- GetValveCurrent
  - CWarnerValveControllerDeviceNet, 647
- GetValveDigitalInPort
  - CWarnerValveControllerDeviceNet, 648
- GetValveDigitalInPortEvent
  - CWarnerValveControllerDeviceNet, 662
- GetValveLedOn
  - CWarnerValveControllerDeviceNet, 648
- GetValveLedOnEvent
  - CWarnerValveControllerDeviceNet, 662
- GetValveManualGroup
  - CWarnerValveControllerDeviceNet, 648
- GetValveManualGroupEvent

- CWarnerValveControllerDeviceNet, 662
- GetValveManualState
  - CWarnerValveControllerDeviceNet, 649
- GetValveManualStateEvent
  - CWarnerValveControllerDeviceNet, 662
- GetValveMode
  - CWarnerValveControllerDeviceNet, 649
- GetValveModeEvent
  - CWarnerValveControllerDeviceNet, 662
- GetValves
  - CMcsBus\_FYIExtensionNet, 203
- GetValvesActiveMap
  - CWarnerValveControllerDeviceNet, 649
- GetValvesManualStateMap
  - CWarnerValveControllerDeviceNet, 649
- GetValveTableEntry
  - CWarnerValveControllerDeviceNet, 650
- GetVDD3I
  - CCMOSMea\_FunctionNet, 107
- GetVDDI
  - CCMOSMea\_FunctionNet, 107
- GetVdsVgs
  - CGrapheneFunctionNet, 171, 172
- GetVdVs
  - CGrapheneFunctionNet, 172
- GetVdVsDAC
  - CGrapheneFunctionNet, 172, 174
- GetVersion
  - CMcsUsbNet, 321
- GetVersionInt
  - DriverVersionNet, 681
- GetVersionString
  - DriverVersionNet, 682
- GetVMMaxNegativeCurrent
  - CMcsBus\_VoltageModeNet, 235
- GetVMMaxNegativeCurrentEeprom
  - CMcsBus\_VoltageModeNet, 235
- GetVMMaxNegativeVoltage
  - CMcsBus\_VoltageModeNet, 235
- GetVMMaxNegativeVoltageEeprom
  - CMcsBus\_VoltageModeNet, 235
- GetVMMaxPositiveCurrent
  - CMcsBus\_VoltageModeNet, 235
- GetVMMaxPositiveCurrentEeprom
  - CMcsBus\_VoltageModeNet, 235
- GetVMMaxPositiveVoltage
  - CMcsBus\_VoltageModeNet, 235
- GetVMMaxPositiveVoltageEeprom
  - CMcsBus\_VoltageModeNet, 236
- GetVMOutputOnOff
  - CMcsBus\_VoltageModeNet, 236
- GetVMVoltage
  - CMcsBus\_VoltageModeNet, 236
- GetVolatileClampOffset
  - CMultiwellDeviceNet, 380
- GetVoltage
  - CokuvisionStimulatorDeviceNet, 395
- GetVoltage12V
  - CRoboDeviceNet, 469
- GetVoltage12VLimit
  - CRoboDeviceNet, 469
- GetVoltage5V
  - CRoboDeviceNet, 469
- GetVoltage5VLimit
  - CRoboDeviceNet, 469
- GetVoltageAirvalve
  - CRoboDeviceNet, 469
- GetVoltageAirvalveLimit
  - CRoboDeviceNet, 469
- GetVoltageClampControllerParam\_D
  - CWarnerUssingFunctionNet, 632
- GetVoltageClampControllerParam\_I
  - CWarnerUssingFunctionNet, 633
- GetVoltageClampControllerParam\_P
  - CWarnerUssingFunctionNet, 633
- GetVoltageRange
  - CGrapheneFunctionNet, 174
- GetVoltageRangeIndex
  - CMcsUsbDacqNet, 271
- GetVoltageRangeInMicroVolt
  - CMcsUsbDacqNet, 272
- GetVoltageRangeInMicroVolt
  - CStg200xBasicNet, 530
- GetVoltageRangeInMicroVolt
  - CStimulusFunctionNet, 570
- GetVoltageRangeInMicroVolt
  - CW2100\_StimulatorFunctionNet, 617
- GetVoltageRangeInMilliVolt
  - CMcsUsbDacqNet, 272
- GetVoltageRangeInMilliVoltByIndex
  - CStg200xBasicNet, 531
- GetVoltageRangeListInMilliVolt
  - CStg200xBasicNet, 531
- GetVoltageRangeSelectedIndex
  - CStg200xBasicNet, 531
- GetVoltageReached
  - CGrapheneFunctionNet, 174, 175
- GetVoltageResolution
  - CGrapheneFunctionNet, 175
- GetVoltageResolutionInMicroVolt
  - CStg200xBasicNet, 531
- GetVoltageResolutionInMicroVolt
  - CStimulusFunctionNet, 571
- GetVoltageResolutionInMicroVolt
  - CW2100\_StimulatorFunctionNet, 618
- GetVoltageResolutionInMicroVoltByIndex
  - CStg200xBasicNet, 532
- GetVoltageRs485A
  - CRoboDeviceNet, 470
- GetVoltageRs485ALimit
  - CRoboDeviceNet, 470
- GetVoltageRs485B
  - CRoboDeviceNet, 470
- GetVoltageRs485BLimit
  - CRoboDeviceNet, 470
- GetVoltageValves
  - CRoboDeviceNet, 470
- GetVoltageValvesLimit
  - CRoboDeviceNet, 470
- GetVolti
  - CTcxDeviceNet, 589



- GetWaveform
  - CTEERFunctionNet, [598](#)
- GetWaveLengthInNanometer
  - CMultiwellOptoStimFunctionNet, [386](#)
- GetWorkingFrequency
  - CRFFFunctionNet, [444](#)
- GetWPADebugMode
  - CWClassicFunctionNet, [669](#)
- GetWPAType
  - CWClassicFunctionNet, [669](#)
- GetXGain
  - CRoboDacqNet, [457](#)
- GetXilinxFlashOffset
  - CMcsUsbFactoryNet, [296](#)
- GetXilinxFlashReadCommand
  - CMcsUsbFactoryNet, [296](#)
- GND\_SWITCH\_BIT
  - CW2100\_StimulatorFunctionNet, [620](#)
- Graphene\_ASIC
  - Mcs::Usb, [77](#)
- GrapheneASIC
  - Mcs::Usb, [80](#)
- GrapheneASICHeadstage
  - Mcs::Usb, [67](#)
- GrapheneProjectTestDevice
  - Mcs::Usb, [78](#)
- Ground
  - Mcs::Usb, [56](#), [65](#)
- GroupID
  - CDeviceGroupChannelInfoTemplateNet< Dacq-GroupChannelEnumTemplateNet >, [123](#)
- GroupType
  - CDeviceGroupChannelInfoTemplateNet< Dacq-GroupChannelEnumTemplateNet >, [123](#)
- GyroOnly
  - Mcs::Usb, [89](#)
- Hardware
  - Mcs::Usb, [66](#)
- HasAnalogOut
  - CSCUFunctionNet, [505](#)
- HasGalvanicIsolation
  - CSCUFunctionNet, [505](#)
- HasHSPowerSwitch
  - CSCUFunctionNet, [506](#)
- HasIMU
  - HeadStageIDType, [687](#)
- HasOptoCurrentMessurement
  - HeadStageIDType, [687](#)
- HasRadioControl
  - CRadioControlledDevicesNet, [437](#)
- HasRef
  - CRoboDeviceNet::RoboMainLowLevelCommands, [698](#)
- HasRefI
  - CRoboStatorDeviceNet, [487](#)
- HasRefXY
  - CRoboStatorDeviceNet, [487](#)
- HasRefZ
  - CRoboStatorDeviceNet, [487](#)
- HasSoftwareKey
  - CMcsUsbNet, [321](#)
- Headstage1NCBathCurrentGroup
  - Mcs::Usb, [59](#)
- Headstage1NCCol2CurrentGroup
  - Mcs::Usb, [59](#)
- Headstage1NChipTempGroup
  - Mcs::Usb, [59](#)
- HeadstageElectrodeGroup
  - Mcs::Usb, [59](#)
- HeadstageIdEnumNet
  - Mcs::Usb, [68](#)
- HeadStageIDType, [685](#)
  - CompareTo, [687](#)
  - ElectricalStimulation, [686](#)
  - Entry, [687](#)
  - Equals, [687](#)
  - HasIMU, [687](#)
  - HasOptoCurrentMessurement, [687](#)
  - HeadStageIDType, [686](#)
  - HeadstageType, [687](#)
  - HeadstageTypeEnum, [686](#)
  - ID, [687](#)
  - MeasuringOnly, [686](#)
  - NumberOfAnalogChannels, [687](#)
  - NumberOfStimulationChannels, [688](#)
  - OpticalStimulation, [686](#)
  - SN, [688](#)
  - StimulusParameters, [688](#)
  - ToString, [687](#)
  - Type, [688](#)
  - TypeValue, [688](#)
  - Unknown, [686](#)
  - UserDefinedName, [688](#)
  - Valid, [688](#)
  - W16IsW14, [688](#)
- HeadstageIDTypeObject, [688](#)
  - \_AdditionalText, [689](#)
  - \_IdType, [689](#)
  - AdditionalText, [690](#)
  - Equals, [689](#)
  - GetHashCode, [689](#)
  - HeadstageIDTypeObject, [689](#)
  - IdType, [690](#)
  - ToString, [689](#)
- HeadStageIDTypeState, [690](#)
  - ControlState, [690](#)
  - DataState, [690](#)
  - IdType, [690](#)
  - State, [690](#)
- HeadstageType
  - HeadStageIDType, [687](#)
- HeadstageTypeEnum
  - HeadStageIDType, [686](#)
- HEKA\_LIH3\_DEVICE
  - Mcs::Usb, [62](#)
- HekaEPC10Double

Mcs::Usb, [75](#)  
 HekaEPC10Quadro  
   Mcs::Usb, [75](#)  
 HekaEPC10Single  
   Mcs::Usb, [75](#)  
 HekaEPC10Triple  
   Mcs::Usb, [75](#)  
 HekaEPCLite  
   Mcs::Usb, [76](#)  
 HekaITEV100  
   Mcs::Usb, [76](#)  
 HekaLIH30  
   Mcs::Usb, [75](#)  
 HekaLIH406  
   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, [71](#)  
 HLA  
   Mcs::Usb, [76](#)  
 HLADacq  
   CHLADeviceNet, [181](#)  
 Hs1Digital  
   Mcs::Usb, [62](#)  
 HS1DigitalData1  
   Mcs::Usb, [84](#)  
 HS1ElectrodeGroup  
   Mcs::Usb, [71](#)  
 HS1Sideband1  
   Mcs::Usb, [63](#), [72](#), [82](#)  
 HS1Sideband10  
   Mcs::Usb, [72](#), [82](#)  
 HS1Sideband11  
   Mcs::Usb, [72](#), [82](#)  
 HS1Sideband12  
   Mcs::Usb, [72](#), [82](#)  
 HS1Sideband13  
   Mcs::Usb, [72](#)  
 HS1Sideband14  
   Mcs::Usb, [72](#)  
 HS1Sideband15

Mcs::Usb, [72](#)  
 HS1Sideband16  
   Mcs::Usb, [72](#)  
 HS1Sideband17  
   Mcs::Usb, [72](#)  
 HS1Sideband18  
   Mcs::Usb, [72](#)  
 HS1Sideband2  
   Mcs::Usb, [63](#), [72](#), [82](#)  
 HS1Sideband3  
   Mcs::Usb, [63](#), [72](#), [82](#)  
 HS1Sideband4  
   Mcs::Usb, [63](#), [72](#), [82](#)  
 HS1Sideband5  
   Mcs::Usb, [63](#), [72](#), [82](#)  
 HS1Sideband6  
   Mcs::Usb, [63](#), [72](#), [82](#)  
 HS1Sideband7  
   Mcs::Usb, [72](#), [82](#)  
 HS1Sideband8  
   Mcs::Usb, [72](#), [82](#)  
 HS1Sideband9  
   Mcs::Usb, [72](#), [82](#)  
 Hs1SidebandHigh  
   Mcs::Usb, [62](#)  
 Hs1SidebandLow  
   Mcs::Usb, [62](#)  
 Hs1Trigger  
   Mcs::Usb, [62](#)  
 HS1Trigger10Status  
   Mcs::Usb, [72](#), [82](#)  
 HS1Trigger11Status  
   Mcs::Usb, [72](#), [82](#)  
 HS1Trigger12Status  
   Mcs::Usb, [72](#), [82](#)  
 HS1Trigger13Status  
   Mcs::Usb, [72](#)  
 HS1Trigger14Status  
   Mcs::Usb, [72](#)  
 HS1Trigger15Status  
   Mcs::Usb, [72](#)  
 HS1Trigger16Status  
   Mcs::Usb, [72](#)  
 HS1Trigger17Status  
   Mcs::Usb, [72](#)  
 HS1Trigger18Status  
   Mcs::Usb, [72](#)  
 HS1Trigger1Status  
   Mcs::Usb, [63](#), [72](#), [81](#)  
 HS1Trigger2Status  
   Mcs::Usb, [63](#), [72](#), [81](#)  
 HS1Trigger3Status  
   Mcs::Usb, [63](#), [72](#), [81](#)  
 HS1Trigger4Status  
   Mcs::Usb, [63](#), [72](#), [81](#)  
 HS1Trigger5Status  
   Mcs::Usb, [63](#), [72](#), [81](#)  
 HS1Trigger6Status

- Mcs::Usb, [63](#), [72](#), [81](#)
- HS1Trigger7Status
  - Mcs::Usb, [72](#), [81](#)
- HS1Trigger8Status
  - Mcs::Usb, [72](#), [82](#)
- HS1Trigger9Status
  - Mcs::Usb, [72](#), [82](#)
- Hs2Digital
  - Mcs::Usb, [63](#)
- HS2DigitalData1
  - Mcs::Usb, [84](#)
- HS2ElectrodeGroup
  - Mcs::Usb, [71](#)
- HS2Sideband1
  - Mcs::Usb, [63](#), [73](#), [82](#)
- HS2Sideband10
  - Mcs::Usb, [73](#), [82](#)
- HS2Sideband11
  - Mcs::Usb, [73](#), [82](#)
- HS2Sideband12
  - Mcs::Usb, [73](#), [82](#)
- HS2Sideband13
  - Mcs::Usb, [73](#)
- HS2Sideband14
  - Mcs::Usb, [73](#)
- HS2Sideband15
  - Mcs::Usb, [73](#)
- HS2Sideband16
  - Mcs::Usb, [73](#)
- HS2Sideband17
  - Mcs::Usb, [73](#)
- HS2Sideband18
  - Mcs::Usb, [73](#)
- HS2Sideband2
  - Mcs::Usb, [63](#), [73](#), [82](#)
- HS2Sideband3
  - Mcs::Usb, [63](#), [73](#), [82](#)
- HS2Sideband4
  - Mcs::Usb, [63](#), [73](#), [82](#)
- HS2Sideband5
  - Mcs::Usb, [63](#), [73](#), [82](#)
- HS2Sideband6
  - Mcs::Usb, [63](#), [73](#), [82](#)
- HS2Sideband7
  - Mcs::Usb, [73](#), [82](#)
- HS2Sideband8
  - Mcs::Usb, [73](#), [82](#)
- HS2Sideband9
  - Mcs::Usb, [73](#), [82](#)
- Hs2SidebandHigh
  - Mcs::Usb, [63](#)
- Hs2SidebandLow
  - Mcs::Usb, [63](#)
- Hs2Trigger
  - Mcs::Usb, [63](#)
- HS2Trigger10Status
  - Mcs::Usb, [73](#), [82](#)
- HS2Trigger11Status
  - Mcs::Usb, [73](#), [82](#)
- HS2Trigger12Status
  - Mcs::Usb, [73](#), [82](#)
- HS2Trigger13Status
  - Mcs::Usb, [73](#)
- HS2Trigger14Status
  - Mcs::Usb, [73](#)
- HS2Trigger15Status
  - Mcs::Usb, [73](#)
- HS2Trigger16Status
  - Mcs::Usb, [73](#)
- HS2Trigger17Status
  - Mcs::Usb, [73](#)
- HS2Trigger18Status
  - Mcs::Usb, [73](#)
- HS2Trigger1Status
  - Mcs::Usb, [63](#), [72](#), [82](#)
- HS2Trigger2Status
  - Mcs::Usb, [63](#), [72](#), [82](#)
- HS2Trigger3Status
  - Mcs::Usb, [63](#), [72](#), [82](#)
- HS2Trigger4Status
  - Mcs::Usb, [63](#), [72](#), [82](#)
- HS2Trigger5Status
  - Mcs::Usb, [63](#), [72](#), [82](#)
- HS2Trigger6Status
  - Mcs::Usb, [63](#), [72](#), [82](#)
- HS2Trigger7Status
  - Mcs::Usb, [72](#), [82](#)
- HS2Trigger8Status
  - Mcs::Usb, [73](#), [82](#)
- HS2Trigger9Status
  - Mcs::Usb, [73](#), [82](#)
- HWInfo
  - CMcsUsbDacqNet, [272](#)
- HwVersion
  - CMcsUsbListEntryNet, [306](#)
- ID
  - HeadStageIDType, [687](#)
- Idle
  - Mcs::Usb, [83](#)
- IdProduct
  - DeviceldNet, [673](#)
- IdType
  - HeadstageIDTypeObject, [690](#)
  - HeadStageIDTypeState, [690](#)
- IdVendor
  - DeviceldNet, [673](#)
- IFB2GoldenInterfaceboard
  - Mcs::Usb, [67](#)
- IFB30GoldenInterfaceboard
  - Mcs::Usb, [67](#)
- IFChannel1
  - Mcs::Usb, [58](#)
- IFChannel2
  - Mcs::Usb, [58](#)
- IFChannel3
  - Mcs::Usb, [58](#)

- IFChannel4
  - Mcs::Usb, [58](#)
- IFChannel5
  - Mcs::Usb, [58](#)
- IFChannel6
  - Mcs::Usb, [58](#)
- IFChannel7
  - Mcs::Usb, [58](#)
- IFChannel8
  - Mcs::Usb, [58](#)
- IFDigChannelsGroup
  - Mcs::Usb, [59](#), [71](#), [81](#), [91](#)
- IM16KRC
  - Mcs::Usb, [78](#)
- IM16S16KRA
  - Mcs::Usb, [78](#)
- IM16S8KRA
  - Mcs::Usb, [78](#)
- IM64KRB
  - Mcs::Usb, [78](#)
- IM64KRC
  - Mcs::Usb, [78](#)
- Input
  - Mcs::Usb, [75](#)
- Intel
  - Mcs::Usb, [88](#)
- InterfaceADCGroup
  - Mcs::Usb, [59](#), [71](#), [81](#), [89](#)
- 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, [506](#)
- IsAutomaticAnalogOut
  - CSCUFunctionNet, [506](#)
- IsBusy
  - CPPCFunctionNet, [421](#)
- IsChamberAvailable
  - CWarnerUssingFunctionNet, [633](#)
- IsChipPowered
  - CCMOSMea\_FunctionNet, [107](#)
- IsConnected
  - CMcsUsbNet, [321](#)
- IsDeviceHighSpeed
  - CMcsUsbNet, [321](#)
- IsDeviceHighSpeedCapable
  - CMcsUsbNet, [321](#)
- IsDeviceTypeOf
  - CMcsUsbListNet, [309](#)
- IsDigitalChannelDedicated
  - CMcsUsbDacqNet::CHWInfo, [184](#)
- IsDigitalOutPortInverted
  - CWarnerValveControllerDeviceNet, [650](#)
- IsDigitalOutPortInvertedEvent
  - CWarnerValveControllerDeviceNet, [662](#)
- IsEqual
  - CFilterCoefficientsNet, [132](#)
- IsExceptionsEnabled
  - CMcsUsbNet, [322](#)
- IsGateFloating
  - CCMOSMea\_FunctionNet, [107](#)
- IsHeadstageAvailable
  - CSCUFunctionNet, [506](#)
- IsHeadstageAvailableEvent
  - CSCUFunctionNet, [511](#)
- IsHighCurrentMode
  - CWarnerUssingFunctionNet, [634](#)
- IsHSPowered
  - CSCUFunctionNet, [507](#)
- IsInDacqLegacyMode
  - CSCUFunctionNet, [507](#)
- IsInternalCalibrationFinished
  - CTEERFunctionNet, [599](#)
  - CWarnerUssingFunctionNet, [634](#)
- IsPlateTypeValid
  - CMultiwellDeviceNet, [381](#)
- IsPulseEnabled
  - CWarnerUssingFunctionNet, [634](#)
- IsPumpMotorOn
  - CRoboFluidDeviceNet, [481](#)
- IsQueueEnabled
  - CRoboDeviceNet, [470](#)
- IsQueueStarted
  - CRoboDeviceNet, [470](#)
- IsRunning
  - CMeaCleanDeviceNet, [338](#)
  - CMeaCoatDeviceNet, [343](#)
- IsSamplingFinished
  - CTEERFunctionNet, [599](#)
- IsUserTriggerEnabled
  - CLIH3DeviceNet, [195](#)
- IsValveDigitalInInverted
  - CWarnerValveControllerDeviceNet, [650](#)
- IsValveDigitalInInvertedEvent
  - CWarnerValveControllerDeviceNet, [662](#)
- IsValveOpen
  - CWarnerValveControllerDeviceNet, [651](#)
- IsValveOpenEvent
  - CWarnerValveControllerDeviceNet, [663](#)
- IsValveOpenInAnalogMode
  - CWarnerValveControllerDeviceNet, [651](#)
- IsValveOpenInAnalogModeEvent
  - CWarnerValveControllerDeviceNet, [663](#)
- IsValveOpenInDigitalMode
  - CWarnerValveControllerDeviceNet, [651](#)
- IsValveOpenInDigitalModeEvent
  - CWarnerValveControllerDeviceNet, [663](#)

- Kelvin
  - Mcs::Usb, [52](#)
- LastPosition
  - Mcs::Usb, [63](#), [73](#), [82](#), [84](#), [91](#)
- LegacyMeaUsb
  - Mcs::Usb, [75](#)
- LIH30\_ADC\_Channel\_EnumNet
  - Mcs::Usb, [68](#)
- LIH30\_DAC\_Channel\_EnumNet
  - Mcs::Usb, [69](#)
- LIH30\_EPC10\_Bus\_EnumNet
  - Mcs::Usb, [69](#)
- LIH30ADCCtrl
  - Mcs::Usb, [67](#), [68](#)
- LIH30ADCModulesGroup
  - Mcs::Usb, [59](#)
- LIH30Interfaceboard
  - Mcs::Usb, [67](#)
- LIH30TestADCGroup
  - Mcs::Usb, [59](#)
- LIH30UserADCGroup
  - Mcs::Usb, [59](#)
- ListModeSendStart
  - CStg200xBasicNet, [532](#)
- ListModeSendStop
  - CStg200xBasicNet, [532](#)
- ListOfChangedTriggers
  - StgStatusNet, [704](#)
- LoadPressure
  - CPPCFunctionNet, [423](#)
- LoadUserFirmware
  - CMcsUsbFactoryNet, [296](#), [297](#)
- LoadValveTable
  - CWarnerValveControllerDeviceNet, [651](#)
- Lock
  - Mcs::Usb, [75](#)
- LockPlateClamp
  - CMultiwellDeviceNet, [381](#)
- Lowpass
  - Mcs::Usb, [66](#)
- LowSpeed
  - Mcs::Usb, [71](#)
- m\_Bottom
  - CCMOSMeaDeviceNet::CRegionOfInterestRect, [438](#)
- m\_Left
  - CCMOSMeaDeviceNet::CRegionOfInterestRect, [438](#)
- m\_pGilsonDevice
  - CGilsonDeviceNet, [166](#)
- m\_pMcsBus\_MotorControlNet
  - CRoboFluidDeviceNet, [482](#)
- m\_pMcsUsb
  - CMcsUsbFunctionNet, [301](#)
- m\_pMcsUsbFunction
  - CMcsUsbFunctionNet, [301](#)
- m\_pRoboFluidDevice
  - CRoboFluidDeviceNet, [482](#)
- m\_Right
  - CCMOSMeaDeviceNet::CRegionOfInterestRect, [438](#)
- m\_Top
  - CCMOSMeaDeviceNet::CRegionOfInterestRect, [438](#)
- Manual
  - Mcs::Usb, [75](#), [83](#), [92](#)
- Manufacturer
  - CMcsUsbListEntryNet, [306](#)
- MaxBitNumber
  - DigitalSource< digitalsourceenum >, [674](#)
  - DigitalSourceGeneral, [675](#), [676](#)
- MaxBitNumberStatic
  - DigitalSource< digitalsourceenum >, [674](#)
- MBC08
  - Mcs::Usb, [77](#)
- MbcChannelStateEnumNet
  - Mcs::Usb, [69](#)
- MbcChargingModeEnumNet
  - Mcs::Usb, [70](#)
- MbcRatedCapacityEnumNet
  - Mcs::Usb, [70](#)
- MC\_Card
  - Mcs::Usb, [75](#)
- MCS
  - Mcs::Usb, [89](#)
- Mcs, [22](#)
- Mcs::Usb, [22](#)
  - A, [69](#)
  - AccelOnly, [89](#)
  - AdapterTypeEnumNet, [51](#)
  - ALA\_VC3, [75](#), [89](#)
  - ALA\_VC3\_DEVICE, [62](#)
  - ALTERA, [54](#)
  - ALTERA\_BASE, [55](#)
  - ALTERA\_BOOTSTRAP, [55](#)
  - ALTERA\_GOLD, [54](#)
  - ALTERA\_TARGET1, [55](#)
  - ALTERA\_TARGET2, [55](#)
  - ALTERA\_TARGET3, [55](#)
  - AlwaysOn, [78](#)
  - Ampere, [52](#), [89](#)
  - AMS\_Dongle, [76](#)
  - Analog, [75](#), [92](#)
  - AnalogGroup, [59](#)
  - AnalogOut\_DAC\_Range\_EnumNet, [51](#)
  - AnalogSource\_HS1, [52](#)
  - AnalogSource\_HS2, [52](#)
  - AnalogSource\_IF, [52](#)
  - AnalogSourceEnumNet, [52](#)
  - AnalogUnitEnumNet, [52](#)
  - Any, [75](#), [89](#)
  - Armed, [83](#)
  - ASMedia, [88](#)
  - AudioTestChannelGroup, [59](#), [71](#), [81](#), [91](#)
  - Aux, [62](#)

- AuxIn, [63](#), [71](#), [81](#), [84](#), [91](#)
- AuxPort, [57](#)
- B, [69](#)
- Bessel, [66](#)
- BMI, [85](#)
- Bootstrap, [54](#)
- BootstrapOtherCypress, [54](#)
- Both, [89](#)
- Break, [79](#)
- BUS0MCSBUS0, [53](#)
- BUS0MCSBUS1, [53](#)
- BUS0MCSBUS10, [53](#)
- BUS0MCSBUS11, [53](#)
- BUS0MCSBUS12, [53](#)
- BUS0MCSBUS13, [53](#)
- BUS0MCSBUS14, [53](#)
- BUS0MCSBUS15, [53](#)
- BUS0MCSBUS2, [53](#)
- BUS0MCSBUS3, [53](#)
- BUS0MCSBUS4, [53](#)
- BUS0MCSBUS5, [53](#)
- BUS0MCSBUS6, [53](#)
- BUS0MCSBUS7, [53](#)
- BUS0MCSBUS8, [53](#)
- BUS0MCSBUS9, [53](#)
- BUS1MCSBUS0, [53](#)
- BUS1MCSBUS1, [53](#)
- BUS1MCSBUS10, [53](#)
- BUS1MCSBUS11, [53](#)
- BUS1MCSBUS12, [53](#)
- BUS1MCSBUS13, [53](#)
- BUS1MCSBUS14, [53](#)
- BUS1MCSBUS15, [53](#)
- BUS1MCSBUS2, [53](#)
- BUS1MCSBUS3, [53](#)
- BUS1MCSBUS4, [53](#)
- BUS1MCSBUS5, [53](#)
- BUS1MCSBUS6, [53](#)
- BUS1MCSBUS7, [53](#)
- BUS1MCSBUS8, [53](#)
- BUS1MCSBUS9, [53](#)
- BUS2MCSBUS0, [54](#)
- BUS2MCSBUS1, [53](#)
- BUS2MCSBUS10, [54](#)
- BUS2MCSBUS11, [54](#)
- BUS2MCSBUS12, [54](#)
- BUS2MCSBUS13, [54](#)
- BUS2MCSBUS14, [54](#)
- BUS2MCSBUS15, [54](#)
- BUS2MCSBUS2, [53](#)
- BUS2MCSBUS3, [53](#)
- BUS2MCSBUS4, [53](#)
- BUS2MCSBUS5, [53](#)
- BUS2MCSBUS6, [53](#)
- BUS2MCSBUS7, [54](#)
- BUS2MCSBUS8, [54](#)
- BUS2MCSBUS9, [54](#)
- BUSNUMBER0, [53](#)
- BUSNUMBER1, [53](#)
- BUSNUMBER2, [53](#)
- Butterworth, [66](#)
- Campden\_Ci4600EphysVideoDataIntegrator, [75](#)
- CatchAmp, [74](#)
- CFirmwareDestinationNet, [52](#)
- channeldata\_current, [84](#)
- channeldata\_current\_always\_boost, [84](#)
- channeldata\_current\_always\_boost\_own\_sync, [84](#)
- channeldata\_current\_own\_boost\_gnd\_sync, [84](#)
- channeldata\_current\_own\_sync, [84](#)
- channeldata\_positive\_current, [84](#)
- channeldata\_positive\_current\_own\_boost\_gnd\_sync, [84](#)
- channeldata\_positive\_current\_own\_sync, [84](#)
- channeldata\_positive\_voltage, [84](#)
- channeldata\_voltage, [84](#)
- ChannelPIC, [54](#)
- ChannelTest, [76](#)
- ChecksumAndPacketCounter, [58](#)
- Ci4600Intan, [51](#)
- ClampModeCurrent, [85](#)
- ClampModeInternalCalibration, [85](#)
- ClampModeOpen, [85](#)
- ClampModeVoltage, [85](#)
- Close, [74](#)
- CmosMea, [68](#)
- CMOSMeaBathModeEnumNet, [56](#)
- CmosMeaHeadstage, [67](#)
- CMOSMeaHeadstage1NCBathCurrentEnumNet, [56](#)
- CMOSMeaHeadstage1NCCol2CurrentEnumNet, [56](#)
- CMOSMeaHeadstage1NChipTempEnumNet, [56](#)
- CMOSMeaHS1SidebandEnumNet, [57](#)
- CMOSMeaHS1TriggerStatusEnumNet, [57](#)
- CmosmealFB2, [67](#)
- CMOSMealFDigChannelEnumNet, [57](#)
- CMOSMealInterfaceADCEnumNet, [57](#)
- CmosMealInterfaceboard, [67](#)
- CMOSMeaPacketFrameContextGroupEnumNet, [58](#)
- CMOSMeaSTG1DACSignalEnumNet, [58](#)
- CMOSMeaValueUnitEnumNet, [58](#)
- CommaPositionA, [66](#)
- CommaPositionB, [66](#)
- csCapacityTestDischarge, [70](#)
- csCapacityTestPrecharge, [70](#)
- csCharge, [70](#)
- csDischarge, [70](#)
- csError, [70](#)
- csIdleChargeFinished, [70](#)
- csIdleNoBattery, [70](#)
- csRefreshBattery, [70](#)
- CurrentClamp, [88](#)
- CurrentMeasure, [56](#)
- Cypress, [89](#)
- Cypress\_FX1, [75](#)

Cypress\_FX2, 75  
Cypress\_FX3, 75  
DAC1Channel, 58  
DAC2Channel, 58  
DAC3Channel, 58  
DAC4Channel, 58  
DACQ1DigitalGroup, 59  
DacqGroupChannelEnumNet, 59  
DacqMeaGroupTypeEnumNet, 59  
DacqTrigger, 64  
DataModeEnumNet, 59  
DEST\_FX3\_TARGET\_MASK, 55  
DEST\_TARGET1, 55  
DEST\_TARGET10, 55  
DEST\_TARGET11, 55  
DEST\_TARGET12, 55  
DEST\_TARGET13, 55  
DEST\_TARGET14, 55  
DEST\_TARGET15, 55  
DEST\_TARGET2, 55  
DEST\_TARGET3, 55  
DEST\_TARGET4, 55  
DEST\_TARGET5, 55  
DEST\_TARGET6, 55  
DEST\_TARGET7, 55  
DEST\_TARGET8, 55  
DEST\_TARGET9, 55  
DEST\_TARGET\_MASK, 55  
DEVICE\_NOT\_FOUND, 65  
DeviceEnumNet, 61  
DeviceHasNoHeadstage, 67, 68, 80  
DeviceNotConnected, 67, 68, 80  
DeviceRunStatus, 63, 72, 81, 84, 91  
DigDataFromReceiver, 91  
Digital, 75, 92  
DigitalData, 63, 72, 81, 84, 91  
DigitalDatastreamEnableEnumNet, 62  
DigitalGroup, 59  
DigitalIn, 62, 63, 71, 81, 84, 91  
DigitalInOfOutPort, 63, 71, 81, 84, 91  
DigitalInPort, 57  
DigitalInReserverd, 62  
DigitalMux, 57  
DigitalOut, 62  
DigitalOutReg, 57  
DigitalOutReserved, 62  
DigitalOutStimulator, 63, 72, 81, 84, 91  
DigitalPulse, 63, 71, 81, 84, 91  
DigitalReg, 57  
DigitalSourceEnumNet, 63  
DigitalStimulatorTriggerEventEnumNet, 64  
DigitalStimulatorTriggerSlopeEnumNet, 64  
DigitalTargetEnumNet, 64  
Digout, 64  
DigOutStim, 62  
DigOutStimulatorStartTrigger, 64  
DigOutStimulatorStopTrigger, 64  
Digstream, 64  
DigStreamFromReceiver, 91  
DigStreamToReceiver, 64  
Dilutor, 78  
DongleS, 76  
Dotriapot, 76  
DoubleToInt, 66  
DownloadOnly, 83  
DSP, 52  
DSPAnalogGroup, 71, 81  
DSPDataGroup, 59, 90  
DSPDigitalGroup, 71, 81  
eCube, 77, 80  
eCubeHeadstage, 67  
ElectrodeDacMuxEnumNet, 64  
ElectrodeModeEnumNet, 65  
ElectrodeOffset, 88  
emAutomatic, 65  
emManual, 65  
Encapsulator, 77  
enCMosMeaChipType, 65  
EnSTG200x\_STATUS, 65  
EOAndCRC, 58  
ExternBCTester, 76  
ExternDTester, 76  
ExternSTester, 76  
Falling, 64  
FCB, 76  
FCX, 76  
Feedback, 63, 71, 81, 84, 91  
FeedbackHigh, 62  
FeedbackLow, 62  
FeedbackReg, 57  
FilterAttributeEnumNet, 65  
FilterBandEnumNet, 66  
FilterCalculationDirectionEnumNet, 66  
FilterFamilyEnumNet, 66  
FilterTypeEnumNet, 66  
Finished, 83  
FPGA10, 54  
FPGA10\_BASE, 55  
FPGA10\_GOLD, 55  
FPGA11, 54  
FPGA11\_BASE, 55  
FPGA11\_GOLD, 55  
FPGA12, 54  
FPGA12\_BASE, 55  
FPGA12\_GOLD, 55  
FPGA13, 54  
FPGA13\_BASE, 55  
FPGA13\_GOLD, 55  
FPGA14, 54  
FPGA14\_BASE, 55  
FPGA14\_GOLD, 55  
FPGA15, 54  
FPGA15\_BASE, 55  
FPGA15\_GOLD, 55  
FPGA16, 54  
FPGA16\_BASE, 55



FPGA16\_GOLD, 55  
FPGA2, 54  
FPGA2\_BASE, 55  
FPGA2\_GOLD, 54  
FPGA3, 54  
FPGA3\_BASE, 55  
FPGA3\_GOLD, 54  
FPGA4, 54  
FPGA4\_BASE, 55  
FPGA4\_GOLD, 54  
FPGA5, 54  
FPGA5\_BASE, 55  
FPGA5\_GOLD, 54  
FPGA6, 54  
FPGA6\_BASE, 55  
FPGA6\_GOLD, 54  
FPGA7, 54  
FPGA7\_BASE, 55  
FPGA7\_GOLD, 54  
FPGA8, 54  
FPGA8\_BASE, 55  
FPGA8\_GOLD, 54  
FPGA9, 54  
FPGA9\_BASE, 55  
FPGA9\_GOLD, 55  
FPGA\_BASE, 55  
FPGA\_BOOTSTRAP, 55  
FPGA\_GOLD, 54  
FPGA\_NORMAL, 52  
FpgaldEnumNet, 67  
FrameContextGroup, 59  
FullCharge, 70  
FullSpeed, 71  
FunkDongleS, 76  
Gated\_High\_Active, 78  
Gated\_Low\_Active, 78  
GE2100, 77  
Generic, 76  
Graphene\_ASIC, 77  
GrapheneASIC, 80  
GrapheneASICHeadstage, 67  
GrapheneProjectTestDevice, 78  
Ground, 56, 65  
GyroOnly, 89  
Hardware, 66  
Headstage1NCBathCurrentGroup, 59  
Headstage1NCCol2CurrentGroup, 59  
Headstage1NChipTempGroup, 59  
HeadstageElectrodeGroup, 59  
HeadstageIdEnumNet, 68  
HEKA\_LIH3\_DEVICE, 62  
HekaEPC10Double, 75  
HekaEPC10Quadro, 75  
HekaEPC10Single, 75  
HekaEPC10Triple, 75  
HekaEPCLite, 76  
HekaITEV100, 76  
HekaLIH30, 75  
HekaLIH406, 76  
HekaLIH816, 76  
HekaPG610, 76  
HekaPG611, 76  
HekaPG612, 76  
HekaPG618, 76  
HekaPG690, 76  
HiClamp, 78  
HiClamp4Uart, 78  
Highpass, 66  
HighSpeed, 71  
HLA, 76  
Hs1Digital, 62  
HS1DigitalData1, 84  
HS1ElectrodeGroup, 71  
HS1Sideband1, 63, 72, 82  
HS1Sideband10, 72, 82  
HS1Sideband11, 72, 82  
HS1Sideband12, 72, 82  
HS1Sideband13, 72  
HS1Sideband14, 72  
HS1Sideband15, 72  
HS1Sideband16, 72  
HS1Sideband17, 72  
HS1Sideband18, 72  
HS1Sideband2, 63, 72, 82  
HS1Sideband3, 63, 72, 82  
HS1Sideband4, 63, 72, 82  
HS1Sideband5, 63, 72, 82  
HS1Sideband6, 63, 72, 82  
HS1Sideband7, 72, 82  
HS1Sideband8, 72, 82  
HS1Sideband9, 72, 82  
Hs1SidebandHigh, 62  
Hs1SidebandLow, 62  
Hs1Trigger, 62  
HS1Trigger10Status, 72, 82  
HS1Trigger11Status, 72, 82  
HS1Trigger12Status, 72, 82  
HS1Trigger13Status, 72  
HS1Trigger14Status, 72  
HS1Trigger15Status, 72  
HS1Trigger16Status, 72  
HS1Trigger17Status, 72  
HS1Trigger18Status, 72  
HS1Trigger1Status, 63, 72, 81  
HS1Trigger2Status, 63, 72, 81  
HS1Trigger3Status, 63, 72, 81  
HS1Trigger4Status, 63, 72, 81  
HS1Trigger5Status, 63, 72, 81  
HS1Trigger6Status, 63, 72, 81  
HS1Trigger7Status, 72, 81  
HS1Trigger8Status, 72, 82  
HS1Trigger9Status, 72, 82  
Hs2Digital, 63  
HS2DigitalData1, 84  
HS2ElectrodeGroup, 71  
HS2Sideband1, 63, 73, 82



- HS2Sideband10, [73](#), [82](#)
- HS2Sideband11, [73](#), [82](#)
- HS2Sideband12, [73](#), [82](#)
- HS2Sideband13, [73](#)
- HS2Sideband14, [73](#)
- HS2Sideband15, [73](#)
- HS2Sideband16, [73](#)
- HS2Sideband17, [73](#)
- HS2Sideband18, [73](#)
- HS2Sideband2, [63](#), [73](#), [82](#)
- HS2Sideband3, [63](#), [73](#), [82](#)
- HS2Sideband4, [63](#), [73](#), [82](#)
- HS2Sideband5, [63](#), [73](#), [82](#)
- HS2Sideband6, [63](#), [73](#), [82](#)
- HS2Sideband7, [73](#), [82](#)
- HS2Sideband8, [73](#), [82](#)
- HS2Sideband9, [73](#), [82](#)
- Hs2SidebandHigh, [63](#)
- Hs2SidebandLow, [63](#)
- Hs2Trigger, [63](#)
- HS2Trigger10Status, [73](#), [82](#)
- HS2Trigger11Status, [73](#), [82](#)
- HS2Trigger12Status, [73](#), [82](#)
- HS2Trigger13Status, [73](#)
- HS2Trigger14Status, [73](#)
- HS2Trigger15Status, [73](#)
- HS2Trigger16Status, [73](#)
- HS2Trigger17Status, [73](#)
- HS2Trigger18Status, [73](#)
- HS2Trigger1Status, [63](#), [72](#), [82](#)
- HS2Trigger2Status, [63](#), [72](#), [82](#)
- HS2Trigger3Status, [63](#), [72](#), [82](#)
- HS2Trigger4Status, [63](#), [72](#), [82](#)
- HS2Trigger5Status, [63](#), [72](#), [82](#)
- HS2Trigger6Status, [63](#), [72](#), [82](#)
- HS2Trigger7Status, [72](#), [82](#)
- HS2Trigger8Status, [73](#), [82](#)
- HS2Trigger9Status, [73](#), [82](#)
- Idle, [83](#)
- IFB2GoldenInterfaceboard, [67](#)
- IFB30GoldenInterfaceboard, [67](#)
- IFChannel1, [58](#)
- IFChannel2, [58](#)
- IFChannel3, [58](#)
- IFChannel4, [58](#)
- IFChannel5, [58](#)
- IFChannel6, [58](#)
- IFChannel7, [58](#)
- IFChannel8, [58](#)
- IFDigChannelsGroup, [59](#), [71](#), [81](#), [91](#)
- IM16KRC, [78](#)
- IM16S16KRA, [78](#)
- IM16S8KRA, [78](#)
- IM64KRB, [78](#)
- IM64KRC, [78](#)
- Input, [75](#)
- Intel, [88](#)
- InterfaceADCGroup, [59](#), [71](#), [81](#), [89](#)
- InterfaceBoard2, [67](#)
- IntToDouble, [66](#)
- InvitroSignalCollectorUnit, [68](#)
- InvivoSignalCollectorUnit, [68](#)
- IoVoltageEnumNet, [68](#)
- IS32KRA, [78](#)
- Kelvin, [52](#)
- LastPosition, [63](#), [73](#), [82](#), [84](#), [91](#)
- LegacyMeaUsb, [75](#)
- LIH30\_ADC\_Channel\_EnumNet, [68](#)
- LIH30\_DAC\_Channel\_EnumNet, [69](#)
- LIH30\_EPC10\_Bus\_EnumNet, [69](#)
- LIH30ADCCtrl, [67](#), [68](#)
- LIH30ADCModulesGroup, [59](#)
- LIH30Interfaceboard, [67](#)
- LIH30TestADCGroup, [59](#)
- LIH30UserADCGroup, [59](#)
- Lock, [75](#)
- Lowpass, [66](#)
- LowSpeed, [71](#)
- Manual, [75](#), [83](#), [92](#)
- MBC08, [77](#)
- MbcChannelStateEnumNet, [69](#)
- MbcChargingModeEnumNet, [70](#)
- MbcRatedCapacityEnumNet, [70](#)
- MC\_Card, [75](#)
- MCS, [89](#)
- MCS\_ANY\_BUS, [70](#)
- MCS\_CHANNELTEST\_DEVICE, [61](#)
- MCS\_DEVICE\_ANY, [61](#)
- MCS\_DEVICE\_USB, [61](#)
- MCS\_DEVICE\_USB\_CYPRESS, [62](#)
- MCS\_ENCAPSULATOR\_DEVICE, [61](#)
- MCS\_EXTERN\_BC\_TESTER\_DEVICE, [62](#)
- MCS\_EXTERN\_D\_TESTER\_DEVICE, [62](#)
- MCS\_FCX\_DEVICE, [61](#)
- MCS\_FYI\_DEVICE, [61](#)
- MCS\_GENERIC\_DEVELOPMENT\_DEVICE, [61](#)
- MCS\_HICLAMP\_DEVICE, [61](#)
- MCS\_HLA\_DEVICE, [61](#)
- MCS\_MBC08\_DEVICE, [62](#)
- MCS\_MC\_STIMULUS\_DEVICE, [61](#)
- MCS\_MCCARD\_DEVICE, [61](#)
- MCS\_MEA\_CLEAN\_DEVICE, [62](#)
- MCS\_MEA\_COAT\_DEVICE, [62](#)
- MCS\_MEA\_DEVICE, [61](#)
- MCS\_MEA\_IMPEDANCE\_DEVICE, [61](#)
- MCS\_MEA\_SWITCH\_DEVICE, [61](#)
- MCS\_MEASURETABLE\_DEVICE, [61](#)
- MCS\_MEASUSB\_DEVICE, [61](#)
- MCS\_NF\_GEN\_DEVICE, [62](#)
- MCS\_OCTOPOT\_DEVICE, [61](#)
- MCS\_OKUVISION\_STIMULATOR\_DEVICE, [62](#)
- MCS\_PATCHSERVER\_DEVICE, [61](#)
- MCS\_PATHIDENT\_DEVICE, [61](#)
- MCS\_PCI\_BUS, [70](#)
- MCS\_PCX\_DEVICE, [61](#)
- MCS\_PEDOTER\_DEVICE, [62](#)

MCS\_PERISTALTIC\_PUMP\_DEVICE, 62  
MCS\_PGA\_DEVICE, 61  
MCS\_PPC\_DEVICE, 62  
MCS\_PPS5\_DEVICE, 61  
MCS\_PPS\_DEVICE, 61  
MCS\_RETINA\_AMS\_DONGLE, 61  
MCS\_RETINA\_LED\_DEVICE, 61  
MCS\_ROBO\_DEVICE, 61  
MCS\_ROBOINJECT\_DEVICE, 61  
MCS\_ROBOOCYTE2\_DEVICE, 61  
MCS\_SAFEIS\_DEVICE, 62  
MCS\_SMARTIMPLANT\_DEVICE, 62  
MCS\_SOFTWARE\_DONGLE\_DEVICE, 62  
MCS\_STG\_DEVICE, 61  
MCS\_SW2TO64\_DEVICE, 61  
MCS\_TCX\_DEVICE, 61  
MCS\_TERSENS\_DEVICE, 61  
MCS\_UNDEFINED\_BUS, 70  
MCS\_USB\_BUS, 70  
MCSBUS0, 53  
MCSBUS1, 52  
MCSBUS10, 53  
MCSBUS11, 53  
MCSBUS12, 53  
MCSBUS13, 53  
MCSBUS14, 53  
MCSBUS15, 53  
MCSBUS2, 52  
MCSBUS3, 52  
MCSBUS4, 52  
MCSBUS5, 52  
MCSBUS6, 52  
MCSBUS7, 52  
MCSBUS8, 53  
MCSBUS9, 53  
McsBusTypeEnumNet, 70  
McsUsbSpeedEnumNet, 70  
MCU1, 52  
ME128, 77  
ME16, 77  
ME2100, 77  
Me2100\_32PICiCE40, 80  
Me2100\_32PICiCE40Headstage, 67  
Me2100\_32Xilinx, 80  
Me2100\_32XilinxHeadstage, 67  
Me2100Graphene16\_32, 80  
Me2100Graphene16\_32Headstage, 67  
Me2100Interfaceboard, 67  
Me2100InvitroSignalCollectorUnit, 67  
Me2100InvivoSignalCollectorUnit, 67  
Me2100UPA32, 80  
Me2100UPA32Headstage, 67  
ME256, 77  
ME32, 77  
ME64, 77  
MEA1060, 77  
MEA120, 51  
MEA2100, 77  
Mea2100, 68  
MEA2100\_256, 77  
Mea2100\_256, 68  
MEA2100\_256DacqGroupChannelEnumNet, 71  
MEA2100\_256DigitalSourceEnumNet, 71  
Mea2100\_256Headstage, 67  
Mea2100\_256Interfaceboard, 67  
MEA2100\_32, 77  
MEA2100\_Lite, 77  
Mea2100\_Lite, 68  
MEA2100\_Mini, 77  
MEA2100\_Mini\_Usb\_develop, 77  
MEA2100BetaScreen, 77  
Mea2100BetaScreen, 80  
Mea2100BetaScreenHeadstage, 67  
Mea2100Headstage, 67  
Mea2100Interfaceboard, 67  
Mea2100LiteHeadstage, 67  
Mea2100Mini120, 80  
Mea2100Mini120Headstage, 67  
Mea2100Mini60ECP5, 80  
Mea2100Mini60ECP5Headstage, 67  
Mea2100Mini60PICiCE40, 80  
Mea2100Mini60PICiCE40Headstage, 67  
Mea2100MultiwellIFB2, 67  
Mea2100STG, 67  
MEA252, 51  
MEA256, 77  
MEA2x32, 51  
MEA2x60, 51  
MEA32, 51  
MEA60, 51  
MEA\_2\_252\_2, 51  
MEA\_2\_252\_2\_6Well, 51  
MEA\_2\_252\_2\_9Well, 51  
MEA\_2\_252\_2\_Test, 51  
MEA\_Clean, 77  
MEA\_Coat, 77  
MEA\_Impedance, 76  
MEA\_Sanofi, 77  
MEA\_Switch, 76  
MEA\_Switch\_2\_1, 77  
MEA\_Switch\_4\_2, 77  
MeaLayoutEnumNet, 73  
MeasureTable, 77  
MicroAmpere, 59  
MilliDegreeCelsius, 59  
mlMEA60, 73  
mlUnknown, 73  
ModulA\_ADC0, 68  
ModulA\_ADC1, 68  
ModulA\_ADC2, 68  
ModulA\_ADC3, 68  
ModulA\_DAC0, 69  
ModulA\_DAC1, 69  
ModulB\_ADC0, 68  
ModulB\_ADC1, 69  
ModulB\_ADC2, 69

ModulB\_ADC3, [69](#)  
ModulB\_DAC0, [69](#)  
ModulB\_DAC1, [69](#)  
ModulC\_ADC0, [69](#)  
ModulC\_ADC1, [69](#)  
ModulC\_ADC2, [69](#)  
ModulC\_ADC3, [69](#)  
ModulC\_DAC0, [69](#)  
ModulC\_DAC1, [69](#)  
ModulD\_ADC0, [69](#)  
ModulD\_ADC1, [69](#)  
ModulD\_ADC2, [69](#)  
ModulD\_ADC3, [69](#)  
ModulD\_DAC0, [69](#)  
ModulD\_DAC1, [69](#)  
Monitor, [83](#)  
Movement, [79](#)  
Multiboot, [77](#)  
Multiwell, [68](#), [77](#)  
Multiwell96, [51](#)  
Multiwell\_ICC, [77](#)  
Multiwell\_MEA\_Mini, [77](#)  
MultiwellHeadstage, [67](#)  
MultiwellInterfaceboard, [67](#)  
MultiwellMini, [80](#)  
MultiwellMiniHeadstage, [67](#)  
MultiwellOptoStim, [76](#)  
MultiwellPlateTypeEnumNet, [73](#)  
Mux, [62](#)  
MuxOtherDevice, [62](#)  
Nanion, [85](#)  
NanoAmpere, [59](#)  
NanoVolt, [58](#)  
NCBathCurrent, [56](#)  
NCCol2Current, [56](#)  
NChipTemperature, [56](#)  
Neptun, [78](#)  
NeuroChip, [77](#)  
NeurochipConfig, [77](#)  
NF\_Gen, [77](#)  
NineWell, [84](#)  
nMos16LV, [65](#)  
nMos32LV, [65](#)  
nMos36LN, [65](#)  
nMos64LN, [65](#)  
No\_Plate, [74](#)  
None, [51](#), [62](#), [75](#), [83](#), [88](#), [89](#)  
Normal, [74](#)  
NOT\_CONNECTED, [65](#)  
NotApplicable, [51](#)  
NoUnit, [58](#)  
NTC10K, [85](#)  
Octopot, [76](#)  
Off, [78](#), [79](#), [89](#)  
off, [79](#)  
OK, [65](#)  
Okuvision\_Stimulator, [76](#)  
OnChannelData, [92](#)  
OnDeviceArrivalRemoval, [92](#)  
One, [63](#), [72](#), [81](#), [84](#), [91](#)  
OnError, [92](#)  
OnMcsUsbDeviceState, [92](#)  
OnMcsUsbDeviceStateCallback, [93](#)  
OnMwPollStatus, [93](#)  
OnStg200xDataHandler, [93](#)  
OnStg200xErrorHandler, [93](#)  
OnStgPollStatus, [93](#)  
OnUpdateFirmwareProgress, [93](#)  
OnUpdateFirmwareStatusChange, [93](#)  
Open, [74](#)  
OpenClamp, [88](#)  
Output, [75](#)  
PacketFrameContextGroup, [59](#), [71](#), [81](#), [91](#)  
PatchServAdcModeEnumNet, [74](#)  
PatchServer, [78](#)  
PathIdent, [77](#)  
PC, [92](#)  
PCI, [89](#)  
PCX, [76](#)  
PeriodicPulse, [62](#)  
PeristalticPump, [77](#)  
PGA, [76](#)  
PIC, [54](#)  
PIC10, [54](#)  
PIC11, [54](#)  
PIC12, [54](#)  
PIC2, [54](#)  
PIC3, [54](#)  
PIC4, [54](#)  
PIC5, [54](#)  
PIC6, [54](#)  
PIC7, [54](#)  
PIC8, [54](#)  
PIC9, [54](#)  
PicoAmpere, [58](#)  
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\_100PBA, [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\_Dummy, [74](#)

Plate\_Dummy\_126, [74](#)  
 PlateClampEnumNet, [74](#)  
 PlateClampLockEnumNet, [74](#)  
 PlusMinus10Volts, [52](#)  
 PlusMinus2Comma5Volts, [52](#)  
 PlusMinus5Volts, [52](#)  
 PortDirectionEnumNet, [75](#)  
 Pos900, [78](#)  
 PositionBase, [78](#)  
 PositionIIBase, [78](#)  
 PositionIICentralUnit, [78](#)  
 PositionImp, [78](#)  
 PostCommaA, [66](#)  
 PostCommaB, [66](#)  
 PP\_Pump\_Mode\_Type\_EnumNet, [75](#)  
 PPC, [77](#)  
 PPS2, [77](#)  
 PPS4plus1, [77](#)  
 PPS5, [77](#)  
 PPS5\_DIG, [77](#)  
 PreCommaA, [66](#)  
 PreCommaB, [66](#)  
 ProductIdEnumNet, [75](#)  
 PT100, [85](#)  
 PT1000, [85](#)  
 PulseGenerator, [63](#), [72](#), [81](#), [84](#), [91](#)  
 PulseGenerator\_Mode\_EnumNet, [78](#)  
 raGate, [79](#)  
 ralgnore, [79](#)  
 raRestart, [79](#)  
 raSingle, [79](#)  
 raStop, [79](#)  
 rawdata, [84](#)  
 RC, [66](#)  
 rc100mAh, [70](#)  
 rc200mAh, [70](#)  
 rc300mAh, [70](#)  
 rc30mAh, [70](#)  
 rcGreater300mAh, [70](#)  
 Rectangle, [86](#)  
 Ref16, [79](#)  
 Ref24, [79](#)  
 Ref32, [79](#)  
 Ref8, [79](#)  
 Reference, [79](#)  
 ReferenceElectrodeModeEnumNet, [78](#)  
 ReferenceElectrodeSwitchPositionEnumNet, [79](#)  
 RegisterHigh, [62](#)  
 RegisterLow, [62](#)  
 Regular, [85](#)  
 Renesas, [88](#)  
 Reserved1, [85](#)  
 Reserved2, [85](#)  
 Reserved3, [85](#)  
 Reserved4, [85](#)  
 Reserved5, [85](#)  
 Retina\_LED, [76](#)  
 RetriggerActionEnumNet, [79](#)  
 Rising, [64](#)  
 RoboCurrentModeEnumNet, [79](#)  
 RoboInject, [78](#)  
 Roboocyte2, [78](#)  
 RoboStatusEventDelegate, [93](#)  
 Running, [83](#)  
 SafeIS, [77](#)  
 SampleDstSize16, [80](#)  
 SampleDstSize32, [80](#)  
 SampleDstSizeNet, [79](#)  
 SampleSize16Signed, [80](#)  
 SampleSize16Unsigned, [80](#)  
 SampleSize24Signed, [80](#)  
 SampleSize24Unsigned, [80](#)  
 SampleSize32Signed, [80](#)  
 SampleSize32Unsigned, [80](#)  
 SampleSize64Signed, [80](#)  
 SampleSize64Unsigned, [80](#)  
 SampleSizeNet, [80](#)  
 SBSVector1, [57](#)  
 SBSVector2, [57](#)  
 SBSVector3, [57](#)  
 SBSVector4, [57](#)  
 SCU1ElectrodeGroupHS1, [81](#)  
 SCU1ElectrodeGroupHS2, [81](#)  
 SCU1ElectrodeGroupHS3, [81](#)  
 SCU1ElectrodeGroupHS4, [81](#)  
 SCU2ElectrodeGroupHS1, [81](#)  
 SCU2ElectrodeGroupHS2, [81](#)  
 SCU2ElectrodeGroupHS3, [81](#)  
 SCU2ElectrodeGroupHS4, [81](#)  
 SCU\_HeadstageIdEnumNet, [80](#)  
 SCUDacqGroupChannelEnumNet, [80](#)  
 SCUDigitalSourceEnumNet, [81](#)  
 Settings, [92](#)  
 Signed\_16bit, [61](#)  
 Signed\_24bit, [61](#)  
 Signed\_32bit, [61](#)  
 Sine, [86](#)  
 SingleWell, [84](#)  
 SixWell, [84](#)  
 SmartImplant, [78](#)  
 SOFAndCTRLword, [68](#)  
 Software, [66](#)  
 SoftwareDongle, [77](#)  
 Standby, [79](#), [88](#)  
 Start, [64](#)  
 State, [89](#)  
 STG, [76](#)  
 Stg1, [65](#)  
 STG1DACSignalGroup, [59](#), [71](#), [81](#)  
 STG1SidebandsGroup, [59](#), [71](#), [81](#)  
 STG1TriggerStatusGroup, [59](#), [71](#), [81](#)  
 Stg2, [65](#)  
 Stg200xDigoutModeEnumNet, [82](#)  
 Stg200xSegmentFlagsEnumNet, [83](#)  
 Stg200xTriggerStatusEnumNet, [83](#)  
 STG2DACSignalGroup, [71](#), [81](#)

STG2SidebandsGroup, [71](#), [81](#)  
STG2TriggerStatusGroup, [71](#), [81](#)  
Stg3, [65](#)  
STG3008\_FA, [76](#)  
STG4002, [76](#)  
STG4002\_opto, [76](#)  
STG4004, [76](#)  
STG4004\_opto, [76](#)  
STG4008, [76](#)  
STG4008\_opto, [76](#)  
STG400x, [76](#)  
STG400x\_opto, [76](#)  
STG5, [76](#)  
STG\_DestinationEnumNet, [83](#)  
StgListModeTrigger, [64](#)  
StgTrigger, [64](#)  
Stimulation, [56](#)  
StimulationLayoutConfigurationEnumNet, [84](#)  
Stop, [64](#), [74](#)  
StorageCharge, [70](#)  
SubtractFromAll, [78](#)  
SubtractFromAllOther, [78](#)  
SubtractFromReferenceElectrodeOnly, [78](#)  
SubtractionOff, [78](#)  
SuperSpeed, [71](#)  
Sw2to64, [76](#)  
SYNCOOUT1, [83](#)  
SYNCOOUT2, [83](#)  
SYNCOOUT3, [83](#)  
SYNCOOUT4, [83](#)  
SYNCOOUT5, [83](#)  
SYNCOOUT6, [83](#)  
SYNCOOUT7, [83](#)  
SYNCOOUT8, [83](#)  
syncoutdata, [84](#)  
SyncStart, [83](#)  
Table, [92](#)  
TBSI\_127, [51](#)  
TBSI\_15, [51](#)  
TBSI\_31, [51](#)  
TBSI\_5, [51](#)  
TBSI\_63, [51](#)  
TBSI\_Dacq, [77](#)  
TBSI\_DACQDigitalSourceEnumNet, [84](#)  
TBSI\_Reserved, [51](#)  
TbsiDacq, [68](#)  
TbsiDacqHeadstage, [67](#)  
TbsiDacqInterfaceboard, [67](#)  
TC01, [76](#)  
TC02, [76](#)  
TCX, [76](#)  
TcxDeviceTypeEnumNet, [85](#)  
TcxSensorTypeEnumNet, [85](#)  
TeerClampModeEnumNet, [85](#)  
TeerWaveformEnumNet, [85](#)  
Tersens, [76](#)  
Test\_ADC\_EPC10, [68](#)  
Test\_DAC\_EPC10, [69](#)  
Timestamp, [58](#)  
TouchTest, [92](#)  
Triggerbox\_AMS, [76](#)  
Triggerbox\_AMS3, [76](#)  
Triggerbox\_IMS, [76](#)  
Triggerbox\_R5, [76](#)  
TriggerOnly, [83](#)  
TriggerSourceEnumNet, [86](#)  
TriggerStatus1, [57](#)  
TriggerStatus2, [57](#)  
TriggerStatus3, [57](#)  
TriggerStatus4, [57](#)  
tsAuxIn1, [87](#)  
tsAuxIn2, [87](#)  
tsDACQCy1Dev1Runs, [88](#)  
tsDACQCy1Dev2Runs, [88](#)  
tsDACQCy2Dev1Runs, [88](#)  
tsDACQCy2Dev2Runs, [88](#)  
tsDigitalIn1, [86](#)  
tsDigitalIn10, [86](#)  
tsDigitalIn11, [86](#)  
tsDigitalIn12, [86](#)  
tsDigitalIn13, [86](#)  
tsDigitalIn14, [86](#)  
tsDigitalIn15, [86](#)  
tsDigitalIn16, [86](#)  
tsDigitalIn17, [86](#)  
tsDigitalIn18, [86](#)  
tsDigitalIn19, [86](#)  
tsDigitalIn2, [86](#)  
tsDigitalIn20, [86](#)  
tsDigitalIn21, [86](#)  
tsDigitalIn22, [86](#)  
tsDigitalIn23, [86](#)  
tsDigitalIn24, [86](#)  
tsDigitalIn25, [86](#)  
tsDigitalIn26, [86](#)  
tsDigitalIn27, [86](#)  
tsDigitalIn28, [86](#)  
tsDigitalIn29, [86](#)  
tsDigitalIn3, [86](#)  
tsDigitalIn30, [86](#)  
tsDigitalIn31, [86](#)  
tsDigitalIn32, [86](#)  
tsDigitalIn4, [86](#)  
tsDigitalIn5, [86](#)  
tsDigitalIn6, [86](#)  
tsDigitalIn7, [86](#)  
tsDigitalIn8, [86](#)  
tsDigitalIn9, [86](#)  
tsDigitalPuse0, [87](#)  
tsDigitalPuse1, [87](#)  
tsDigitalPuse10, [87](#)  
tsDigitalPuse11, [87](#)  
tsDigitalPuse12, [87](#)  
tsDigitalPuse13, [87](#)  
tsDigitalPuse14, [87](#)  
tsDigitalPuse15, [87](#)

tsDigitalPuse16, [87](#)  
tsDigitalPuse17, [87](#)  
tsDigitalPuse18, [88](#)  
tsDigitalPuse19, [88](#)  
tsDigitalPuse2, [87](#)  
tsDigitalPuse20, [88](#)  
tsDigitalPuse21, [88](#)  
tsDigitalPuse22, [88](#)  
tsDigitalPuse23, [88](#)  
tsDigitalPuse24, [88](#)  
tsDigitalPuse25, [88](#)  
tsDigitalPuse26, [88](#)  
tsDigitalPuse27, [88](#)  
tsDigitalPuse28, [88](#)  
tsDigitalPuse29, [88](#)  
tsDigitalPuse3, [87](#)  
tsDigitalPuse30, [88](#)  
tsDigitalPuse31, [88](#)  
tsDigitalPuse4, [87](#)  
tsDigitalPuse5, [87](#)  
tsDigitalPuse6, [87](#)  
tsDigitalPuse7, [87](#)  
tsDigitalPuse8, [87](#)  
tsDigitalPuse9, [87](#)  
tsFeedback1, [86](#)  
tsFeedback10, [87](#)  
tsFeedback11, [87](#)  
tsFeedback12, [87](#)  
tsFeedback13, [87](#)  
tsFeedback14, [87](#)  
tsFeedback15, [87](#)  
tsFeedback16, [87](#)  
tsFeedback17, [87](#)  
tsFeedback18, [87](#)  
tsFeedback19, [87](#)  
tsFeedback2, [86](#)  
tsFeedback20, [87](#)  
tsFeedback21, [87](#)  
tsFeedback22, [87](#)  
tsFeedback23, [87](#)  
tsFeedback24, [87](#)  
tsFeedback25, [87](#)  
tsFeedback26, [87](#)  
tsFeedback27, [87](#)  
tsFeedback28, [87](#)  
tsFeedback29, [87](#)  
tsFeedback3, [86](#)  
tsFeedback30, [87](#)  
tsFeedback31, [87](#)  
tsFeedback32, [87](#)  
tsFeedback4, [87](#)  
tsFeedback5, [87](#)  
tsFeedback6, [87](#)  
tsFeedback7, [87](#)  
tsFeedback8, [87](#)  
tsFeedback9, [87](#)  
tsNone, [86](#)  
tsSidebandBit8, [88](#)  
tsTriggered, [88](#)  
Unknown, [51](#), [52](#), [66](#), [85](#), [88](#)  
unknown, [65](#)  
UnknownDest, [56](#)  
UnknownSpeed, [71](#)  
Unlock, [75](#)  
Unsigned\_16bit, [61](#)  
Unsigned\_24bit, [61](#)  
Unsigned\_32bit, [61](#)  
UpdateTrigger, [83](#)  
USB, [52](#)  
USB\_TARGET1, [55](#)  
USB\_TARGET2, [55](#)  
USB\_TARGET3, [56](#)  
UsbTest, [77](#)  
UsbVendorIdEnumNet, [88](#)  
User\_ADC\_0, [68](#)  
User\_ADC\_1, [68](#)  
User\_ADC\_2, [68](#)  
User\_ADC\_3, [68](#)  
User\_ADC\_4, [68](#)  
User\_DAC\_0, [69](#)  
User\_DAC\_1, [69](#)  
User\_DAC\_2, [69](#)  
UssingChamber, [67](#)  
UssingClampModeEnumNet, [88](#)  
UssingRail, [67](#)  
UssingUnitEnumNet, [89](#)  
VendorIdEnumNet, [89](#)  
Volt, [52](#), [89](#)  
Voltage\_3V3, [68](#)  
Voltage\_5V0, [68](#)  
VoltageClamp, [88](#)  
W2100, [77](#)  
W2100\_Accel\_Gyro\_Select\_EnumNet, [89](#)  
W2100DacqGroupChannelEnumNet, [89](#)  
W2100DigitalSourceEnumNet, [91](#)  
W2100IFB2, [67](#)  
W2100Interfaceboard, [67](#)  
W2100WirelessReceiver, [67](#), [68](#)  
W2100WirelessReceiverAnalog, [67](#), [68](#)  
Warner, [85](#)  
Warner\_TEER\_Machine, [78](#)  
Warner\_Ussing, [78](#)  
WARNER\_USSING\_DEVICE, [62](#)  
Warner\_Valve\_Control, [78](#)  
WARNER\_VALVE\_CONTROL\_DEVICE, [62](#)  
Whole\_Cell\_Patch, [77](#)  
WholeCellPatch, [80](#)  
WholeCellPatchHeadstage, [67](#)  
WirelessHeadStageAccDataRE1HS1, [90](#)  
WirelessHeadStageAccDataRE1HS2, [90](#)  
WirelessHeadStageAccDataRE1HS3, [90](#)  
WirelessHeadStageAccDataRE1HS4, [90](#)  
WirelessHeadStageAccDataRE2HS1, [90](#)  
WirelessHeadStageAccDataRE2HS2, [90](#)  
WirelessHeadStageAccDataRE2HS3, [90](#)  
WirelessHeadStageAccDataRE2HS4, [90](#)



- WirelessHeadStageAnalogRE1HS1, [90](#)
- WirelessHeadStageAnalogRE1HS2, [90](#)
- WirelessHeadStageAnalogRE1HS3, [90](#)
- WirelessHeadStageAnalogRE1HS4, [90](#)
- WirelessHeadStageAnalogRE2HS1, [90](#)
- WirelessHeadStageAnalogRE2HS2, [90](#)
- WirelessHeadStageAnalogRE2HS3, [90](#)
- WirelessHeadStageAnalogRE2HS4, [90](#)
- WirelessHeadStageGyroDataRE1HS1, [90](#)
- WirelessHeadStageGyroDataRE1HS2, [90](#)
- WirelessHeadStageGyroDataRE1HS3, [90](#)
- WirelessHeadStageGyroDataRE1HS4, [90](#)
- WirelessHeadStageGyroDataRE2HS1, [90](#)
- WirelessHeadStageGyroDataRE2HS2, [90](#)
- WirelessHeadStageGyroDataRE2HS3, [90](#)
- WirelessHeadStageGyroDataRE2HS4, [90](#)
- WirelessHeadStageOptoStimCurrentRE1HS1, [90](#)
- WirelessHeadStageOptoStimCurrentRE1HS2, [90](#)
- WirelessHeadStageOptoStimCurrentRE1HS3, [90](#)
- WirelessHeadStageOptoStimCurrentRE1HS4, [90](#)
- WirelessHeadStageOptoStimCurrentRE2HS1, [90](#)
- WirelessHeadStageOptoStimCurrentRE2HS2, [90](#)
- WirelessHeadStageOptoStimCurrentRE2HS3, [90](#)
- WirelessHeadStageOptoStimCurrentRE2HS4, [91](#)
- WirelessHeadStageReservedARE1HS1, [90](#)
- WirelessHeadStageReservedARE1HS2, [90](#)
- WirelessHeadStageReservedARE1HS3, [90](#)
- WirelessHeadStageReservedARE1HS4, [90](#)
- WirelessHeadStageReservedARE2HS1, [90](#)
- WirelessHeadStageReservedARE2HS2, [90](#)
- WirelessHeadStageReservedARE2HS3, [91](#)
- WirelessHeadStageReservedARE2HS4, [91](#)
- WirelessHeadStageReservedBRE1HS1, [91](#)
- WirelessHeadStageReservedBRE1HS2, [91](#)
- WirelessHeadStageReservedBRE1HS3, [91](#)
- WirelessHeadStageReservedBRE1HS4, [91](#)
- WirelessHeadStageReservedBRE2HS1, [91](#)
- WirelessHeadStageReservedBRE2HS2, [91](#)
- WirelessHeadStageReservedBRE2HS3, [91](#)
- WirelessHeadStageReservedBRE2HS4, [91](#)
- WirelessHeadStageReservedCRE1HS1, [91](#)
- WirelessHeadStageReservedCRE1HS2, [91](#)
- WirelessHeadStageReservedCRE1HS3, [91](#)
- WirelessHeadStageReservedCRE1HS4, [91](#)
- WirelessHeadStageReservedCRE2HS1, [91](#)
- WirelessHeadStageReservedCRE2HS2, [91](#)
- WirelessHeadStageReservedCRE2HS3, [91](#)
- WirelessHeadStageReservedCRE2HS4, [91](#)
- WirelessHeadStageStatusRE1HS1, [90](#)
- WirelessHeadStageStatusRE1HS2, [90](#)
- WirelessHeadStageStatusRE1HS3, [90](#)
- WirelessHeadStageStatusRE1HS4, [90](#)
- WirelessHeadStageStatusRE2HS1, [90](#)
- WirelessHeadStageStatusRE2HS2, [90](#)
- WirelessHeadStageStatusRE2HS3, [90](#)
- WirelessHeadStageStatusRE2HS4, [90](#)
- WirelessTestAdapter, [51](#)
- Work, [92](#)
- WPA16, [77](#)
- WPA32, [77](#)
- WPA4, [77](#)
- WPA8, [77](#)
- WvcDisplayModeEnumNet, [92](#)
- WvcValveModeEnumNet, [92](#)
- Zero, [63](#), [71](#), [81](#), [84](#), [91](#)
- MCS\_ANY\_BUS
  - Mcs::Usb, [70](#)
- MCS\_CHANNELTEST\_DEVICE
  - Mcs::Usb, [61](#)
- MCS\_DEVICE\_ANY
  - Mcs::Usb, [61](#)
- MCS\_DEVICE\_USB
  - Mcs::Usb, [61](#)
- MCS\_DEVICE\_USB\_CYPRESS
  - Mcs::Usb, [62](#)
- MCS\_ENCAPSULATOR\_DEVICE
  - Mcs::Usb, [61](#)
- MCS\_EXTERN\_BC\_TESTER\_DEVICE
  - Mcs::Usb, [62](#)
- MCS\_EXTERN\_D\_TESTER\_DEVICE
  - Mcs::Usb, [62](#)
- MCS\_FCX\_DEVICE
  - Mcs::Usb, [61](#)
- MCS\_FYI\_DEVICE
  - Mcs::Usb, [61](#)
- MCS\_GENERIC\_DEVELOPMENT\_DEVICE
  - Mcs::Usb, [61](#)
- MCS\_HICLAMP\_DEVICE
  - Mcs::Usb, [61](#)
- MCS\_HLA\_DEVICE
  - Mcs::Usb, [61](#)
- MCS\_MBC08\_DEVICE
  - Mcs::Usb, [62](#)
- MCS\_MC\_STIMULUS\_DEVICE
  - Mcs::Usb, [61](#)
- MCS\_MCCARD\_DEVICE
  - Mcs::Usb, [61](#)
- MCS\_MEA\_CLEAN\_DEVICE
  - Mcs::Usb, [62](#)
- MCS\_MEA\_COAT\_DEVICE
  - Mcs::Usb, [62](#)
- MCS\_MEA\_DEVICE
  - Mcs::Usb, [61](#)
- MCS\_MEA\_IMPEDANCE\_DEVICE
  - Mcs::Usb, [61](#)
- MCS\_MEA\_SWITCH\_DEVICE
  - Mcs::Usb, [61](#)
- MCS\_MEASURETABLE\_DEVICE
  - Mcs::Usb, [61](#)
- MCS\_MEASUSB\_DEVICE
  - Mcs::Usb, [61](#)
- MCS\_NF\_GEN\_DEVICE
  - Mcs::Usb, [62](#)
- MCS\_OCTOPOT\_DEVICE
  - Mcs::Usb, [61](#)
- MCS\_OKUVISION\_STIMULATOR\_DEVICE

Mcs::Usb, [62](#)  
 MCS\_PATCHSERVER\_DEVICE  
   Mcs::Usb, [61](#)  
 MCS\_PATHIDENT\_DEVICE  
   Mcs::Usb, [61](#)  
 MCS\_PCI\_BUS  
   Mcs::Usb, [70](#)  
 MCS\_PCX\_DEVICE  
   Mcs::Usb, [61](#)  
 MCS\_PEDOTER\_DEVICE  
   Mcs::Usb, [62](#)  
 MCS\_PERISTALTIC\_PUMP\_DEVICE  
   Mcs::Usb, [62](#)  
 MCS\_PGA\_DEVICE  
   Mcs::Usb, [61](#)  
 MCS\_PPC\_DEVICE  
   Mcs::Usb, [62](#)  
 MCS\_PPS5\_DEVICE  
   Mcs::Usb, [61](#)  
 MCS\_PPS\_DEVICE  
   Mcs::Usb, [61](#)  
 MCS\_RETINA\_AMS\_DONGLE  
   Mcs::Usb, [61](#)  
 MCS\_RETINA\_LED\_DEVICE  
   Mcs::Usb, [61](#)  
 MCS\_ROBO\_DEVICE  
   Mcs::Usb, [61](#)  
 MCS\_ROBOINJECT\_DEVICE  
   Mcs::Usb, [61](#)  
 MCS\_ROBOOCYTE2\_DEVICE  
   Mcs::Usb, [61](#)  
 MCS\_SAFEIS\_DEVICE  
   Mcs::Usb, [62](#)  
 MCS\_SMARTIMPLANT\_DEVICE  
   Mcs::Usb, [62](#)  
 MCS\_SOFTWARE\_DONGLE\_DEVICE  
   Mcs::Usb, [62](#)  
 MCS\_STG\_DEVICE  
   Mcs::Usb, [61](#)  
 MCS\_SW2TO64\_DEVICE  
   Mcs::Usb, [61](#)  
 MCS\_TCX\_DEVICE  
   Mcs::Usb, [61](#)  
 MCS\_TERSENS\_DEVICE  
   Mcs::Usb, [61](#)  
 MCS\_UNDEFINED\_BUS  
   Mcs::Usb, [70](#)  
 MCS\_USB\_BUS  
   Mcs::Usb, [70](#)  
 McsBus  
   CPPCDeviceNet, [416](#)  
   CPPS\_DeviceNet, [425](#)  
   CRoboDeviceNet, [478](#)  
 MCSBUS0  
   Mcs::Usb, [53](#)  
 MCSBUS1  
   FirmwareDestinationNames, [684](#)  
   Mcs::Usb, [52](#)

MCSBUS10  
   FirmwareDestinationNames, [684](#)  
   Mcs::Usb, [53](#)  
 MCSBUS11  
   FirmwareDestinationNames, [684](#)  
   Mcs::Usb, [53](#)  
 MCSBUS12  
   FirmwareDestinationNames, [684](#)  
   Mcs::Usb, [53](#)  
 MCSBUS13  
   FirmwareDestinationNames, [684](#)  
   Mcs::Usb, [53](#)  
 MCSBUS14  
   Mcs::Usb, [53](#)  
 MCSBUS15  
   Mcs::Usb, [53](#)  
 MCSBUS2  
   FirmwareDestinationNames, [684](#)  
   Mcs::Usb, [52](#)  
 MCSBUS3  
   FirmwareDestinationNames, [684](#)  
   Mcs::Usb, [52](#)  
 MCSBUS4  
   FirmwareDestinationNames, [684](#)  
   Mcs::Usb, [52](#)  
 MCSBUS5  
   FirmwareDestinationNames, [684](#)  
   Mcs::Usb, [52](#)  
 MCSBUS6  
   FirmwareDestinationNames, [684](#)  
   Mcs::Usb, [52](#)  
 MCSBUS7  
   FirmwareDestinationNames, [684](#)  
   Mcs::Usb, [52](#)  
 MCSBUS8  
   FirmwareDestinationNames, [685](#)  
   Mcs::Usb, [53](#)  
 MCSBUS9  
   FirmwareDestinationNames, [685](#)  
   Mcs::Usb, [53](#)  
 McsBus\_MotorControl  
   CPeristalticPumpDeviceNet, [401](#)  
   CPPCDeviceNet, [416](#)  
   CPPS\_DeviceNet, [425](#)  
   CRoboDeviceNet, [478](#)  
   CRoboFluidDeviceNet, [482](#)  
 McsBus\_Sensor  
   CPPCDeviceNet, [416](#)  
   CPPS\_DeviceNet, [425](#)  
 McsBus\_VoltageMode  
   CFluidControlDeviceNet, [145](#)  
 McsBus\_XY  
   CRoboDeviceNet, [475](#)  
 McsBus\_ZI  
   CRoboDeviceNet, [475](#)  
 McsBusTypeEnumNet  
   Mcs::Usb, [70](#)  
 McsUsbDeviceStateEvent



- CMcsUsbDeviceStatePushFunctionNet, 290
- CMcsUsbDeviceStatePushNet, 291
- McsUsbSpeedEnumNet
  - Mcs::Usb, 70
- MCU1
  - FirmwareDestinationNames, 685
  - Mcs::Usb, 52
- ME128
  - Mcs::Usb, 77
- ME16
  - Mcs::Usb, 77
- ME2100
  - Mcs::Usb, 77
- Me2100\_32PICiCE40
  - Mcs::Usb, 80
- Me2100\_32PICiCE40Headstage
  - Mcs::Usb, 67
- Me2100\_32Xilinx
  - Mcs::Usb, 80
- Me2100\_32XilinxHeadstage
  - Mcs::Usb, 67
- Me2100Graphene16\_32
  - Mcs::Usb, 80
- Me2100Graphene16\_32Headstage
  - Mcs::Usb, 67
- Me2100Interfaceboard
  - Mcs::Usb, 67
- Me2100InvitroSignalCollectorUnit
  - Mcs::Usb, 67
- Me2100InvivoSignalCollectorUnit
  - Mcs::Usb, 67
- Me2100UPA32
  - Mcs::Usb, 80
- Me2100UPA32Headstage
  - Mcs::Usb, 67
- ME256
  - Mcs::Usb, 77
- ME32
  - Mcs::Usb, 77
- ME64
  - Mcs::Usb, 77
- MEA1060
  - Mcs::Usb, 77
- MEA120
  - Mcs::Usb, 51
- MEA2100
  - Mcs::Usb, 77
- Mea2100
  - Mcs::Usb, 68
- MEA2100\_256
  - Mcs::Usb, 77
- Mea2100\_256
  - Mcs::Usb, 68
- MEA2100\_256DacqGroupChannelEnumNet
  - Mcs::Usb, 71
- MEA2100\_256DigitalSourceEnumNet
  - Mcs::Usb, 71
- Mea2100\_256Headstage
  - Mcs::Usb, 67
- Mea2100\_256Interfaceboard
  - Mcs::Usb, 67
- MEA2100\_32
  - Mcs::Usb, 77
- MEA2100\_Lite
  - Mcs::Usb, 77
- Mea2100\_Lite
  - Mcs::Usb, 68
- MEA2100\_Mini
  - Mcs::Usb, 77
- MEA2100\_Mini\_Usb\_develop
  - Mcs::Usb, 77
- MEA2100BetaScreen
  - Mcs::Usb, 77
- Mea2100BetaScreen
  - Mcs::Usb, 80
- Mea2100BetaScreenHeadstage
  - Mcs::Usb, 67
- Mea2100Headstage
  - Mcs::Usb, 67
- Mea2100Interfaceboard
  - Mcs::Usb, 67
- Mea2100LiteHeadstage
  - Mcs::Usb, 67
- Mea2100Mini120
  - Mcs::Usb, 80
- Mea2100Mini120Headstage
  - Mcs::Usb, 67
- Mea2100Mini60ECP5
  - Mcs::Usb, 80
- Mea2100Mini60ECP5Headstage
  - Mcs::Usb, 67
- Mea2100Mini60PICiCE40
  - Mcs::Usb, 80
- Mea2100Mini60PICiCE40Headstage
  - Mcs::Usb, 67
- Mea2100MultiwellIFB2
  - Mcs::Usb, 67
- Mea2100STG
  - Mcs::Usb, 67
- MEA252
  - Mcs::Usb, 51
- MEA256
  - Mcs::Usb, 77
- MEA2x32
  - Mcs::Usb, 51
- MEA2x60
  - Mcs::Usb, 51
- MEA32
  - Mcs::Usb, 51
- MEA60
  - Mcs::Usb, 51
- MEA\_2\_252\_2
  - Mcs::Usb, 51
- MEA\_2\_252\_2\_6Well
  - Mcs::Usb, 51
- MEA\_2\_252\_2\_9Well

Mcs::Usb, [51](#)  
 MEA\_2\_252\_2\_Test  
     Mcs::Usb, [51](#)  
 MEA\_Clean  
     Mcs::Usb, [77](#)  
 MEA\_Coat  
     Mcs::Usb, [77](#)  
 MEA\_Impedance  
     Mcs::Usb, [76](#)  
 MEA\_Sanofi  
     Mcs::Usb, [77](#)  
 MEA\_Switch  
     Mcs::Usb, [76](#)  
 MEA\_Switch\_2\_1  
     Mcs::Usb, [77](#)  
 MEA\_Switch\_4\_2  
     Mcs::Usb, [77](#)  
 MeaAudioFunctionNet  
     CMeaDeviceNet, [353](#)  
 MeaDigitalDataFunctionNet  
     CMeaDeviceNet, [353](#)  
 MeaFeedbackFunctionNet  
     CMeaDeviceNet, [353](#)  
 MeaLayoutEnumNet  
     Mcs::Usb, [73](#)  
 Measure  
     CPathIdentDeviceNet, [398](#)  
 MeasureReservoir  
     CPPCFunctionNet, [423](#)  
 MeasureTable  
     Mcs::Usb, [77](#)  
 MeasuringOnly  
     HeadStageIDType, [686](#)  
 MeFunctionNet  
     CMeaDeviceNet, [353](#)  
 MicroAmpere  
     Mcs::Usb, [59](#)  
 MilliDegreeCelsius  
     Mcs::Usb, [59](#)  
 mkfilter  
     mkfilterNet, [691](#)  
 mkfilter\_coef\_in\_one\_set  
     mkfilterNet, [691](#)  
 mkfilter\_highpass\_coef  
     mkfilterNet, [692](#)  
 mkfilter\_highpass\_frequency\_from\_coef  
     mkfilterNet, [692](#)  
 mkfilter\_highpass\_frequency\_from\_k  
     mkfilterNet, [692](#)  
 mkfilter\_highpass\_k  
     mkfilterNet, [692](#)  
 mkfilter\_MCS  
     mkfilterNet, [692](#)  
 mkfilter\_MCS\_k  
     mkfilterNet, [692](#), [693](#)  
 mkfilter\_normalize\_coeffs\_int  
     mkfilterNet, [693](#)  
 mkfilter\_normalize\_coeffs\_short  
     mkfilterNet, [693](#)  
 mkfilter\_scale\_coef\_in\_one\_set  
     mkfilterNet, [693](#)  
 mkfilterNet, [691](#)  
     mkfilter, [691](#)  
     mkfilter\_coef\_in\_one\_set, [691](#)  
     mkfilter\_highpass\_coef, [692](#)  
     mkfilter\_highpass\_frequency\_from\_coef, [692](#)  
     mkfilter\_highpass\_frequency\_from\_k, [692](#)  
     mkfilter\_highpass\_k, [692](#)  
     mkfilter\_MCS, [692](#)  
     mkfilter\_MCS\_k, [692](#), [693](#)  
     mkfilter\_normalize\_coeffs\_int, [693](#)  
     mkfilter\_normalize\_coeffs\_short, [693](#)  
     mkfilter\_normalize\_scale\_coeffs\_int, [693](#)  
     mkfilter\_scale\_coef\_in\_one\_set, [693](#)  
 mIMEA60  
     Mcs::Usb, [73](#)  
 mlUnknown  
     Mcs::Usb, [73](#)  
 ModuIA\_ADC0  
     Mcs::Usb, [68](#)  
 ModuIA\_ADC1  
     Mcs::Usb, [68](#)  
 ModuIA\_ADC2  
     Mcs::Usb, [68](#)  
 ModuIA\_ADC3  
     Mcs::Usb, [68](#)  
 ModuIA\_DAC0  
     Mcs::Usb, [69](#)  
 ModuIA\_DAC1  
     Mcs::Usb, [69](#)  
 ModuB\_ADC0  
     Mcs::Usb, [68](#)  
 ModuB\_ADC1  
     Mcs::Usb, [69](#)  
 ModuB\_ADC2  
     Mcs::Usb, [69](#)  
 ModuB\_ADC3  
     Mcs::Usb, [69](#)  
 ModuB\_DAC0  
     Mcs::Usb, [69](#)  
 ModuB\_DAC1  
     Mcs::Usb, [69](#)  
 ModuC\_ADC0  
     Mcs::Usb, [69](#)  
 ModuC\_ADC1  
     Mcs::Usb, [69](#)  
 ModuC\_ADC2  
     Mcs::Usb, [69](#)  
 ModuC\_ADC3  
     Mcs::Usb, [69](#)  
 ModuC\_DAC0  
     Mcs::Usb, [69](#)  
 ModuC\_DAC1  
     Mcs::Usb, [69](#)

- ModulD\_ADC0
  - Mcs::Usb, [69](#)
- ModulD\_ADC1
  - Mcs::Usb, [69](#)
- ModulD\_ADC2
  - Mcs::Usb, [69](#)
- ModulD\_ADC3
  - Mcs::Usb, [69](#)
- ModulD\_DAC0
  - Mcs::Usb, [69](#)
- ModulD\_DAC1
  - Mcs::Usb, [69](#)
- Monitor
  - Mcs::Usb, [83](#)
- MoveAbs
  - CRoboDeviceNet, [470](#), [471](#)
- MoveAbsI
  - CRoboStatorDeviceNet, [487](#)
- MoveAbsXY
  - CRoboStatorDeviceNet, [487](#)
- MoveAbsZ
  - CRoboStatorDeviceNet, [487](#), [488](#)
- Movement
  - Mcs::Usb, [79](#)
- Multiboot
  - Mcs::Usb, [77](#)
- MultibootGetCypressImageld
  - CMcsUsbNet, [322](#)
- MultibootGetImageld
  - CMcsUsbNet, [322](#)
- MultibootGetSelectedImage
  - CMcsUsbNet, [322](#)
- MultibootSelectImage
  - CMcsUsbNet, [322](#)
- Multiwell
  - Mcs::Usb, [68](#), [77](#)
- MultiwellI96
  - Mcs::Usb, [51](#)
- Multiwell\_ICC
  - Mcs::Usb, [77](#)
- Multiwell\_MEA\_Mini
  - Mcs::Usb, [77](#)
- MultiwellHeadstage
  - Mcs::Usb, [67](#)
- MultiwellInterfaceboard
  - Mcs::Usb, [67](#)
- MultiwellMini
  - Mcs::Usb, [80](#)
- MultiwellMiniHeadstage
  - Mcs::Usb, [67](#)
- MultiwellOptoStim
  - Mcs::Usb, [76](#)
- MultiwellPlateTypeEnumNet
  - Mcs::Usb, [73](#)
- Mux
  - Mcs::Usb, [62](#)
- MuxOtherDevice
  - Mcs::Usb, [62](#)
- MwPollStatusEvent
  - CStg200xDownloadNet, [564](#)
- Nanion
  - Mcs::Usb, [85](#)
- NanoAmpere
  - Mcs::Usb, [59](#)
- NanoVolt
  - Mcs::Usb, [58](#)
- NCBathCurrent
  - Mcs::Usb, [56](#)
- NCCol2Current
  - Mcs::Usb, [56](#)
- NChipTemperature
  - Mcs::Usb, [56](#)
- Neptun
  - Mcs::Usb, [78](#)
- NeuroChip
  - Mcs::Usb, [77](#)
- NeurochipConfig
  - Mcs::Usb, [77](#)
- NF\_Gen
  - Mcs::Usb, [77](#)
- NineWell
  - Mcs::Usb, [84](#)
- nMos16LV
  - Mcs::Usb, [65](#)
- nMos32LV
  - Mcs::Usb, [65](#)
- nMos36LN
  - Mcs::Usb, [65](#)
- nMos64LN
  - Mcs::Usb, [65](#)
- No\_Plate
  - Mcs::Usb, [74](#)
- None
  - Mcs::Usb, [51](#), [62](#), [75](#), [83](#), [88](#), [89](#)
- Normal
  - Mcs::Usb, [74](#)
- NOT\_CONNECTED
  - Mcs::Usb, [65](#)
- NotApplicable
  - Mcs::Usb, [51](#)
- NoUnit
  - Mcs::Usb, [58](#)
- NTC10K
  - Mcs::Usb, [85](#)
- NullCommand
  - CRoboDeviceNet, [472](#)
- NumberOfAnalogChannels
  - HeadStageIDType, [687](#)
- NumberOfChannels
  - CDeviceGroupChannelInfoTemplateNet < Dacq-GroupChannelEnumTemplateNet >, [123](#)
- NumberOfStimulationChannels
  - HeadStageIDType, [688](#)
- NumCoefSets
  - CCreateFilterNet, [114](#)

- Octopot
  - Mcs::Usb, [76](#)
- Off
  - Mcs::Usb, [78](#), [79](#), [89](#)
- off
  - Mcs::Usb, [79](#)
- OK
  - Mcs::Usb, [65](#)
- Okuvision\_Stimulator
  - Mcs::Usb, [76](#)
- OnChannelData
  - Mcs::Usb, [92](#)
- OnDeviceArrivalRemoval
  - Mcs::Usb, [92](#)
- One
  - Mcs::Usb, [63](#), [72](#), [81](#), [84](#), [91](#)
- OnError
  - Mcs::Usb, [92](#)
- OnGetActiveRunningTableNumber
  - CWarnerValveControllerDeviceNet, [652](#)
- OnGetAnalogThresholdHigh
  - CWarnerValveControllerDeviceNet, [652](#)
- OnGetAnalogThresholdLow
  - CWarnerValveControllerDeviceNet, [652](#)
- OnGetAnalogVoltage
  - CWarnerValveControllerDeviceNet, [652](#)
- OnGetAvailableHeadstages
  - CSCUFunctionNet, [507](#)
- OnGetCurrentNumberOfValves
  - CWarnerValveControllerDeviceNet, [652](#)
- OnGetDigitalOutPortValve
  - CWarnerValveControllerDeviceNet, [652](#)
- OnGetDigitalPortDirection
  - CWarnerValveControllerDeviceNet, [652](#)
- OnGetDisplayMode
  - CWarnerValveControllerDeviceNet, [652](#)
- OnGetPlateClampStateByHeadstage
  - CMultiwellCallbackFunctionNet, [376](#)
- OnGetTableNamebyIndex
  - CWarnerValveControllerDeviceNet, [652](#)
- OnGetValveActive
  - CWarnerValveControllerDeviceNet, [653](#)
- OnGetValveBoardRevision
  - CWarnerValveControllerDeviceNet, [653](#)
- OnGetValveDigitalInPort
  - CWarnerValveControllerDeviceNet, [653](#)
- OnGetValveLedOn
  - CWarnerValveControllerDeviceNet, [653](#)
- OnGetValveManualGroup
  - CWarnerValveControllerDeviceNet, [653](#)
- OnGetValveManualState
  - CWarnerValveControllerDeviceNet, [653](#)
- OnGetValveMode
  - CWarnerValveControllerDeviceNet, [653](#)
- OnIsDigitalOutPortInverted
  - CWarnerValveControllerDeviceNet, [653](#)
- OnIsHeadstageAvailable
  - CSCUFunctionNet, [507](#)
- OnIsValveDigitalInInverted
  - CWarnerValveControllerDeviceNet, [654](#)
- OnIsValveOpen
  - CWarnerValveControllerDeviceNet, [654](#)
- OnIsValveOpenInAnalogMode
  - CWarnerValveControllerDeviceNet, [654](#)
- OnIsValveOpenInDigitalMode
  - CWarnerValveControllerDeviceNet, [654](#)
- OnMcsUsbDeviceState
  - Mcs::Usb, [92](#)
- OnMcsUsbDeviceStateCallback
  - Mcs::Usb, [93](#)
- OnMwPollStatus
  - Mcs::Usb, [93](#)
- OnStg200xDataHandler
  - Mcs::Usb, [93](#)
- OnStg200xErrorHandler
  - Mcs::Usb, [93](#)
- OnStgPollStatus
  - Mcs::Usb, [93](#)
- OnTableEntryChanged
  - CWarnerValveControllerDeviceNet, [654](#)
- OnUpdateFirmwareProgress
  - Mcs::Usb, [93](#)
- OnUpdateFirmwareStatusChange
  - Mcs::Usb, [93](#)
- Open
  - Mcs::Usb, [74](#)
- OpenClamp
  - Mcs::Usb, [88](#)
- OpenPipe
  - CGenericDevelopDeviceNet, [159](#)
- OpenPlateClamp
  - CMultiwellDeviceNet, [381](#)
- operator=
  - DeviceldNet, [673](#)
- OpticalStimulation
  - HeadStageIDType, [686](#)
- Order
  - CCreateFilterNet, [114](#)
  - CFilterPropertyNet, [138](#)
- Output
  - Mcs::Usb, [75](#)
- PacketFrameContextGroup
  - Mcs::Usb, [59](#), [71](#), [81](#), [91](#)
- PatchServAdcModeEnumNet
  - Mcs::Usb, [74](#)
- PatchServer
  - Mcs::Usb, [78](#)
- PathIdent
  - Mcs::Usb, [77](#)
- PatternListStart
  - COctoPotDeviceNet, [391](#)
- PC
  - Mcs::Usb, [92](#)
- PCI
  - Mcs::Usb, [89](#)
- PCX

- Mcs::Usb, [76](#)
- PeriodicPulse
  - Mcs::Usb, [62](#)
- PeristalticPump
  - Mcs::Usb, [77](#)
- PGA
  - Mcs::Usb, [76](#)
- PIC
  - FirmwareDestinationNames, [685](#)
  - Mcs::Usb, [54](#)
- PIC10
  - Mcs::Usb, [54](#)
- PIC11
  - Mcs::Usb, [54](#)
- PIC12
  - Mcs::Usb, [54](#)
- PIC2
  - FirmwareDestinationNames, [685](#)
  - Mcs::Usb, [54](#)
- PIC3
  - FirmwareDestinationNames, [685](#)
  - Mcs::Usb, [54](#)
- PIC4
  - FirmwareDestinationNames, [685](#)
  - Mcs::Usb, [54](#)
- PIC5
  - Mcs::Usb, [54](#)
- PIC6
  - Mcs::Usb, [54](#)
- PIC7
  - Mcs::Usb, [54](#)
- PIC8
  - Mcs::Usb, [54](#)
- PIC9
  - Mcs::Usb, [54](#)
- PicoAmpere
  - Mcs::Usb, [58](#)
- 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\_100PBA
  - 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\_Dummy
  - Mcs::Usb, [74](#)
- Plate\_Dummy\_126
  - Mcs::Usb, [74](#)
- PlateClampEnumNet
  - Mcs::Usb, [74](#)
- PlateClampLockEnumNet
  - Mcs::Usb, [74](#)
- PlusMinus10Volts
  - Mcs::Usb, [52](#)
- PlusMinus2Comma5Volts
  - Mcs::Usb, [52](#)
- PlusMinus5Volts
  - Mcs::Usb, [52](#)
- PollStatusEvent
  - CStimulusFunctionNet, [576](#)
  - CW2100\_StimulatorFunctionNet, [620](#)
- 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, [702](#)
  - Mcs::Usb, [66](#)
- PostCommaB
  - CFilterCoefficientsNet::s\_FilterAttributesNet, [702](#)
  - Mcs::Usb, [66](#)
- PowerChip
  - CCMOSMea\_FunctionNet, [108](#)
- PowerHS
  - CSCUFunctionNet, [507](#)

PP\_Pump\_Mode\_Type\_EnumNet  
     Mcs::Usb, [75](#)  
 PPC  
     Mcs::Usb, [77](#)  
 PPCFunction  
     CPPCDeviceNet, [416](#)  
 PPS2  
     Mcs::Usb, [77](#)  
 PPS4plus1  
     Mcs::Usb, [77](#)  
 PPS5  
     Mcs::Usb, [77](#)  
 PPS5\_DIG  
     Mcs::Usb, [77](#)  
 PPS\_Function  
     CPPS\_DeviceNet, [425](#)  
 PreCommaA  
     CFilterCoefficientsNet::s\_FilterAttributesNet, [702](#)  
     Mcs::Usb, [66](#)  
 PreCommaB  
     CFilterCoefficientsNet::s\_FilterAttributesNet, [702](#)  
     Mcs::Usb, [66](#)  
 PrepareAndAppendData  
     CStg200xDownloadNet, [561](#)  
     CStimulusFunctionNet, [571](#)  
 PrepareAndSendData  
     CStg200xDownloadNet, [562](#)  
     CStimulusFunctionNet, [572](#)  
 PrepareChannelData  
     CDigOutStimulatorFunctionNet, [127](#)  
 PrepareData  
     CStimulusFunctionNet, [573](#)  
     CW2100\_StimulatorFunctionNet, [618](#)  
 PrepareDataSync  
     CW2100\_StimulatorFunctionNet, [618](#)  
 Product  
     CMcsUsbListEntryNet, [306](#)  
 ProductIdEnumNet  
     Mcs::Usb, [75](#)  
 Program  
     CProgramPressureCurveNet, [432](#)  
 PT100  
     Mcs::Usb, [85](#)  
 PT1000  
     Mcs::Usb, [85](#)  
 PulseGenerator  
     CW2100\_FunctionNet, [613](#)  
     Mcs::Usb, [63](#), [72](#), [81](#), [84](#), [91](#)  
 PulseGenerator\_Mode\_EnumNet  
     Mcs::Usb, [78](#)  
 PumpOff  
     CRoboFluidDeviceNet, [481](#)  
 PumpOn  
     CRoboFluidDeviceNet, [481](#)  
 QueryTriggerstatus  
     CStg200xDownloadNet, [563](#)  
 raGate  
     Mcs::Usb, [79](#)  
 raIgnore  
     Mcs::Usb, [79](#)  
 RampStart  
     COctoPotDeviceNet, [391](#)  
 raRestart  
     Mcs::Usb, [79](#)  
 raSingle  
     Mcs::Usb, [79](#)  
 raStop  
     Mcs::Usb, [79](#)  
 rawdata  
     Mcs::Usb, [84](#)  
 RC  
     Mcs::Usb, [66](#)  
 rc100mAh  
     Mcs::Usb, [70](#)  
 rc200mAh  
     Mcs::Usb, [70](#)  
 rc300mAh  
     Mcs::Usb, [70](#)  
 rc30mAh  
     Mcs::Usb, [70](#)  
 rcGreater300mAh  
     Mcs::Usb, [70](#)  
 Read  
     CExternDTesterDeviceNet, [130](#)  
 Read2  
     CExternDTesterDeviceNet, [130](#)  
 ReadBlockFromFlash  
     CMcsUsbFactoryNet, [297](#)  
 ReadBlockFromIFBGlobalEEPROM  
     CMcsUsbFactoryNet, [297](#)  
 ReadBlockFromNVMEM  
     CMcsUsbFactoryNet, [297](#)  
 ReadClipping  
     CLIH3DeviceNet, [196](#)  
 ReadEepromRegisterPreconfig  
     CMcsUsbNet, [322](#), [323](#)  
 ReadPipe  
     CGenericDevelopDeviceNet, [159](#)  
 ReadRegister  
     CMcsUsbNet, [323](#)  
 ReadRegister32  
     CMcsUsbNet, [323](#)  
 ReadRegisterTimeSlot  
     CMcsUsbNet, [323](#)  
 ReadUARTData  
     CLIH3DeviceNet, [196](#)  
 Receive  
     CSerialPortNet, [512](#)  
 ReceiveString  
     CSerialPortNet, [512](#)  
 Rectangle  
     Mcs::Usb, [86](#)  
 Ref16  
     Mcs::Usb, [79](#)  
 Ref24

- Mcs::Usb, [79](#)
- Ref32
  - Mcs::Usb, [79](#)
- Ref8
  - Mcs::Usb, [79](#)
- Reference
  - Mcs::Usb, [79](#)
- ReferenceElectrodeModeEnumNet
  - Mcs::Usb, [78](#)
- ReferenceElectrodeSwitchPositionEnumNet
  - Mcs::Usb, [79](#)
- RegisterHigh
  - Mcs::Usb, [62](#)
- RegisterLow
  - Mcs::Usb, [62](#)
- Regular
  - Mcs::Usb, [85](#)
- RemoveSoftwareKey
  - CMcsUsbNet, [323](#)
- Renasas
  - Mcs::Usb, [88](#)
- RescanHeadstage
  - CMcsUsbNet, [323](#)
- Reserved1
  - Mcs::Usb, [85](#)
- Reserved2
  - Mcs::Usb, [85](#)
- Reserved3
  - Mcs::Usb, [85](#)
- Reserved4
  - Mcs::Usb, [85](#)
- Reserved5
  - Mcs::Usb, [85](#)
- ResetAdcOffset
  - COctoPotDeviceNet, [391](#)
- ResetChannelmap
  - CWClassicFunctionNet, [669](#)
- ResetDacOffset
  - COctoPotDeviceNet, [391](#)
- ResetHighpassFilter
  - CFilterConfigurationNet, [134](#)
- ResetPipe
  - CGenericDevelopDeviceNet, [159](#)
- ResetStatus
  - CStg200xDownloadBasicNet, [555](#)
- Retina\_LED
  - Mcs::Usb, [76](#)
- RetriggerActionEnumNet
  - Mcs::Usb, [79](#)
- RFFunction
  - CPositionIIDeviceNet, [412](#)
- Rising
  - Mcs::Usb, [64](#)
- RoboCurrentModeEnumNet
  - Mcs::Usb, [79](#)
- RoboDacq
  - CHiClampDeviceNet, [180](#)
- RoboDevice
  - CSafeISDeviceNet, [493](#)
  - RoboError\_AnotherMaster
    - CRoboDeviceNet, [475](#)
  - RoboError\_Base
    - CRoboDeviceNet, [476](#)
  - RoboError\_CannotEscapeEndSwitch
    - CRoboDeviceNet, [476](#)
  - RoboError\_CommandAlreadyInProgress
    - CRoboDeviceNet, [476](#)
  - RoboError\_CommandNotPossible
    - CRoboDeviceNet, [476](#)
  - RoboError\_CommunicationTimeout
    - CRoboDeviceNet, [476](#)
  - RoboError\_DacqNotReady
    - CRoboDeviceNet, [476](#)
  - RoboError\_DLLMovementTimeout
    - CRoboDeviceNet, [476](#)
  - RoboError\_FindReferenceMethod
    - CRoboDeviceNet, [476](#)
  - RoboError\_GilsonCommandPending
    - CRoboDeviceNet, [476](#)
  - RoboError\_GilsonTimeout
    - CRoboDeviceNet, [476](#)
  - RoboError\_GilsonWrondID
    - CRoboDeviceNet, [477](#)
  - RoboError\_McsBus\_UnknownCommand
    - CRoboDeviceNet, [477](#)
  - RoboError\_NoEndSwitch
    - CRoboDeviceNet, [477](#)
  - RoboError\_NoMoreData
    - CRoboDeviceNet, [477](#)
  - RoboError\_NoReference
    - CRoboDeviceNet, [477](#)
  - RoboError\_NoSpeedOrAcceleration
    - CRoboDeviceNet, [477](#)
  - RoboError\_OverPressure
    - CRoboDeviceNet, [477](#)
  - RoboError\_ParameterNotAllowed
    - CRoboDeviceNet, [477](#)
  - RoboError\_PeristalticTimeout
    - CRoboDeviceNet, [477](#)
  - RoboError\_Phase0OutOfRange
    - CRoboDeviceNet, [477](#)
  - RoboError\_PollLoopCanceled
    - CRoboDeviceNet, [478](#)
  - RoboError\_PollLoopCanceledAndStopMovement
    - CRoboDeviceNet, [478](#)
  - RoboError\_Pressure
    - CRoboDeviceNet, [478](#)
  - RoboError\_RangeExceeded
    - CRoboDeviceNet, [478](#)
  - RoboError\_StateChangeNotPossible
    - CRoboDeviceNet, [478](#)
  - RoboError\_Timeout
    - CRoboDeviceNet, [478](#)
  - RoboError\_UnknownCommand
    - CRoboDeviceNet, [478](#)
  - RoboInject



- Mcs::Usb, [78](#)
- RoboMainLowLevelCommand
  - CRoboDeviceNet, [479](#)
- RoboMainStatorLowLevelCommand
  - CRoboStatorDeviceNet, [490](#)
- Roboocyte2
  - Mcs::Usb, [78](#)
- RoboStatusEvent
  - CRoboDeviceNet, [479](#)
- RoboStatusEventDelegate
  - Mcs::Usb, [93](#)
- Running
  - Mcs::Usb, [83](#)
- RunTable
  - CRoboDacqNet, [457](#)
- s\_FilterAttributesNet
  - CFilterCoefficientsNet::s\_FilterAttributesNet, [701](#)
- SafeIS
  - Mcs::Usb, [77](#)
- SampleDstSize16
  - Mcs::Usb, [80](#)
- SampleDstSize32
  - Mcs::Usb, [80](#)
- SampleDstSizeNet
  - Mcs::Usb, [79](#)
- SampleRate
  - CCreateFilterNet, [114](#)
- Samplerate
  - CMcsUsbDacqNet, [289](#)
- 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, [57](#)
- SBSVector2
  - Mcs::Usb, [57](#)
- SBSVector3
  - Mcs::Usb, [57](#)
- SBSVector4
  - Mcs::Usb, [57](#)
- Scale
  - CCreateFilterNet, [114](#)
- ScanForHeadstages
  - CWClassicFunctionNet, [669](#)
- SCU1ElectrodeGroupHS1
  - Mcs::Usb, [81](#)
- SCU1ElectrodeGroupHS2
  - Mcs::Usb, [81](#)
- SCU1ElectrodeGroupHS3
  - Mcs::Usb, [81](#)
- SCU1ElectrodeGroupHS4
  - Mcs::Usb, [81](#)
- SCU2ElectrodeGroupHS1
  - Mcs::Usb, [81](#)
- SCU2ElectrodeGroupHS2
  - Mcs::Usb, [81](#)
- SCU2ElectrodeGroupHS3
  - Mcs::Usb, [81](#)
- SCU2ElectrodeGroupHS4
  - Mcs::Usb, [81](#)
- SCU\_HeadstageIdEnumNet
  - Mcs::Usb, [80](#)
- SCUDacqGroupChannelEnumNet
  - Mcs::Usb, [80](#)
- SCUDigitalSourceEnumNet
  - Mcs::Usb, [81](#)
- SelectHeadstage
  - CW2100\_FunctionNet, [612](#)
- SelectTimeSlot
  - CW2100\_StimulatorFunctionNet, [618](#)
- Send
  - CSerialPortNet, [512](#)
- SendBuffered
  - CGilsonDeviceNet, [166](#)
- SendChannelData
  - CDigOutStimulatorFunctionNet, [127](#)
  - CStg200xDownloadBasicNet, [555](#)
- SendCommand
  - CLIH3DeviceNet, [196](#)
- SendImmediate
  - CGilsonDeviceNet, [166](#)
- SendImmediateGetResponse
  - CGilsonDeviceNet, [166](#)
- SendMultiplexedData
  - CStimulusFunctionNet, [573](#)
- SendPreparedData
  - CStimulusFunctionNet, [573](#)
  - CW2100\_StimulatorFunctionNet, [618](#)
- SendSegmentDefine
  - CStg200xDownloadNet, [563](#)
- SendSegmentSelect
  - CStg200xDownloadNet, [563](#)
- SendSegmentStart
  - CStg200xDownloadNet, [564](#)
- SendStart
  - CStg200xBasicNet, [533](#)
  - CStimulusFunctionNet, [573](#)
  - CW2100\_StimulatorFunctionNet, [619](#)
- SendStartDacq
  - CMcsUsbDacqNet, [272](#)
- SendStartStgAndDacq



- CMcsUsbDacqNet, 272
- SendStop
  - CStg200xBasicNet, 533
  - CStimulusFunctionNet, 574
  - CW2100\_StimulatorFunctionNet, 619
- SendStopDacq
  - CMcsUsbDacqNet, 273
- SendStopStgAndDacq
  - CMcsUsbDacqNet, 273
- SendStopStgAndDacqWithOptions
  - CMcsUsbDacqNet, 273
- SendSyncData
  - CStg200xDownloadBasicNet, 555
- Sensor
  - CFYIDeviceNet, 146
  - CMeasureTableDeviceNet, 361
  - CPatchServerDeviceNet, 397
- SerialNumber
  - CMcsUsbListEntryNet, 306
  - CMcsUsbNet, 330
- SerialPort
  - CHLADeviceNet, 181
- Set4ADCCatchampAverageShift
  - CMcsBus\_SensorNet, 229
- Set4ADCMode
  - CMcsBus\_SensorNet, 229
- Set4DAC
  - CMcsBus\_SensorNet, 229
- Set\_Values
  - CNF\_GenDeviceNet, 389
  - CPathIdentDeviceNet, 398
- SetAbsMaxCurrentInMicroAmp
  - CMultiwellOptoStimFunctionNet, 387
- SetAccelerationI
  - CRoboStatorDeviceNet, 488
- SetAccelerationNativeI
  - CRoboStatorDeviceNet, 488
- SetAccelerationNativeXY
  - CRoboStatorDeviceNet, 488
- SetAccelerationNativeZ
  - CRoboStatorDeviceNet, 488
- SetAccelerationXY
  - CRoboStatorDeviceNet, 488
- SetAccelerationZ
  - CRoboStatorDeviceNet, 488
- SetAccelGyroDesiredRate
  - CW2100\_FunctionNet, 612
- SetAccelGyroEnabled
  - CW2100\_FunctionNet, 612
- SetAccelRange
  - CW2100\_FunctionNet, 612
- SetActiveRunningTableNumber
  - CWarnerValveControllerDeviceNet, 654
- SetADC
  - CWarnerValveControllerDeviceTesterFunctionNet, 665
- SetAdcChannels
  - CSafeISDeviceNet, 492
- SetADCInputOffset
  - CCMOSMea\_FunctionNet, 108
- SetAdcOffset
  - CLIH3DeviceNet, 196
  - COctoPotDeviceNet, 392
- SetAdcOffsetPermanent
  - CLIH3DeviceNet, 197
- SetAdcSamplePos
  - CSafeISDeviceNet, 492
- SetAirpressureLimit
  - CRoboDeviceNet, 472
- SetAirValve
  - CRoboDeviceNet, 472
- SetAllDigout
  - CRoboDacqNet, 457
- SetAmplificationSwitch
  - COctoPotDeviceNet, 392
- SetAmplitude
  - CChannelTestDeviceNet, 98
- SetAmplitude\_nA
  - CTEERFunctionNet, 599
- SetAnalogOutADCRange
  - CSCUFunctionNet, 508
- SetAnalogOutChannel
  - CW2100\_FunctionNet, 612
- SetAnalogOutChannels
  - CSCUFunctionNet, 508
- SetAnalogOutDACRange
  - CSCUFunctionNet, 508
- SetAnalogOutFilter
  - CW2100\_FunctionNet, 612
- SetAnalogThresholdHigh
  - CWarnerValveControllerDeviceNet, 654
- SetAnalogThresholdLow
  - CWarnerValveControllerDeviceNet, 655
- SetAnalogVoltageRange
  - CPPCFunctionNet, 423
- SetAnalogVoltages
  - CPPS\_FunctionNet, 428
- SetAttenuation
  - CChannelTestDeviceNet, 98
- SetAudioChannels
  - CMeaAudioFunctionNet, 335
  - CW2100\_FunctionNet, 612
- SetAudioOutDacParameter
  - CLIH3DeviceNet, 197
- SetAutocalibrationDisabled
  - CStg200xBasicNet, 533
- SetAxisConfig
  - CRoboDeviceNet::RoboMainLowLevelCommands, 698
- SetAxisLED
  - CRoboocyte2DeviceNet, 484
- SetAxisParametersEeprom
  - CMcsBus\_AxisParametersNet, 200, 201
- SetBandwidthByIndex
  - CLintanMea\_FunctionNet, 186
- SetBaseFrequency

- CRFFunctionNet, 444
- SetBaseSamplerate
  - CCMOSMeaDeviceNet, 112
  - CGrapheneASICDeviceNet, 168
- SetBath
  - CCMOSMea\_FunctionNet, 108
- SetBathclamp
  - COctoPotDeviceNet, 392
- SetBathMode
  - CCMOSMea\_FunctionNet, 108
- SetBlankingEnable
  - CStg200xBasicNet, 535, 536
- SetBuffer
  - CGenericDevelopDeviceNet, 160
- SetBufferIndex
  - CTEERFunctionNet, 599
- SetBusAddress
  - CMcsBusNet, 240
- SetBusAddressEeprom
  - CMcsBusNet, 240
- SetByteBuffer
  - CGenericDevelopDeviceNet, 160
- SetCalibration
  - CTcxDeviceNet, 589
- SetCardinalDacqSamplerate
  - CInterfaceboardFunctionNet, 189
- SetCardinalStgOutputrate
  - CInterfaceboardFunctionNet, 190
- SetChannel
  - CSw2to64DeviceNet, 577
- SetChannelmap
  - CWClassicFunctionNet, 669
- SetChannels
  - CSw2to64DeviceNet, 578
- SetChannelSwitch
  - COctoPotDeviceNet, 392
- SetChargingMode
  - CMultiBatteryChargerDeviceNet, 371
- SetChargingPCoefficient
  - CMultiBatteryChargerDeviceNet, 372
- SetCheckVoltage
  - CokuvisionStimulatorDeviceNet, 396
- SetClampMode
  - CTEERFunctionNet, 600
  - CWarnerUssingFunctionNet, 635
- SetColorRgb
  - CMultiwellOptoStimFunctionNet, 387
- SetColorStr
  - CMultiwellOptoStimFunctionNet, 387
- SetCommand
  - CMcsBusNet, 241
  - CPedoterDeviceNet, 400
  - CRoboDacqNet, 457
- SetConfiguration
  - CMcsUsbNet, 324
- SetConfigurationBit
  - CRoboDacqNet, 457
- SetConfigurationBitAxc
  - CRoboDacqNet, 458
- SetConfigurationBitBlu\_Led
  - CRoboDacqNet, 458
- SetConfigurationBitBlu\_LedToggleFast
  - CRoboDacqNet, 458
- SetConfigurationBitBlu\_LedToggleSlow
  - CRoboDacqNet, 458
- SetConfigurationBitCC\_Gen
  - CRoboDacqNet, 458
- SetConfigurationBitCV\_Gen
  - CRoboDacqNet, 458
- SetConfigurationBitRC\_Gen
  - CRoboDacqNet, 458
- SetConfigurationBitRed\_Led
  - CRoboDacqNet, 458
- SetConfigurationBitRed\_LedSaturation
  - CRoboDacqNet, 458
- SetConfigurationBitRed\_LedToggleFast
  - CRoboDacqNet, 459
- SetConfigurationBitRed\_LedToggleSlow
  - CRoboDacqNet, 459
- SetConfigurationBitRelais
  - CRoboDacqNet, 459
- SetConfigurationBitRV\_Gen
  - CRoboDacqNet, 459
- SetConfigurationBitStream
  - CRoboDacqNet, 459
- SetConfigurationBitSupply
  - CRoboDacqNet, 459
- SetControllerParams
  - CTEERFunctionNet, 600
- SetCrossTalkOffset
  - CRoboDacqNet, 459
- SetCrossTalkOptimum
  - CRoboDacqNet, 459
- SetCurrentAirvalveLimit
  - CRoboDeviceNet, 472
- SetCurrentAndAir
  - CRoboDeviceNet, 472
- SetCurrentAndAirXY
  - CRoboStatorDeviceNet, 488
- SetCurrentEditTableNumber
  - CWarnerValveControllerDeviceNet, 655
- SetCurrentEnable
  - CTEERFunctionNet, 600
- SetCurrentFactor
  - CokuvisionStimulatorDeviceNet, 396
- SetCurrentMode
  - CStg200xBasicNet, 536
- SetCurrentRangeSelectedIndex
  - CStg200xBasicNet, 536
- SetCycles
  - CMeaCleanDeviceNet, 338
  - CMeaCoatDeviceNet, 343
- SetD
  - CTcxDeviceNet, 589
- SetDacAmplificationFactor
  - CStg200xBasicNet, 537

- SetDacAutoControl
  - COctoPotDeviceNet, [392](#)
- SetDacIdleValue
  - CLIH3DeviceNet, [197](#)
- SetDacMode
  - CSafeISDeviceNet, [492](#)
- SetDACOffset
  - CGrapheneFunctionNet, [175](#), [176](#)
  - COkuvisionStimulatorDeviceNet, [396](#)
- SetDacOffset
  - CDacCalibrationFunctionNet, [116](#)
  - CLIH3DeviceNet, [197](#)
  - COctoPotDeviceNet, [392](#)
- SetDacOffsetPermanent
  - CLIH3DeviceNet, [198](#)
- SetDacPeriode
  - CSafeISDeviceNet, [492](#)
- SetDacPulseform
  - CSafeISDeviceNet, [492](#)
- SetDacqLegacyMode
  - CSCUFunctionNet, [508](#)
- SetDacRange
  - CW2100\_FunctionNet, [613](#)
- SetDACs
  - CMcsBus\_SensorNet, [229](#)
- SetDacUseldleValue
  - CLIH3DeviceNet, [198](#)
- SetDacValue
  - COctoPotDeviceNet, [392](#)
- SetDataMode
  - CMcsUsbDacqNet, [274](#)
- SetDefault
  - CWarnerValveControllerDeviceNet, [655](#)
- SetDestinationSerialNumber
  - CMcsUsbFactoryNet, [297](#)
- SetDetectionThreshold
  - CMcsBus\_SensorNet, [229](#)
- SetDevice
  - CTcxDeviceNet, [589](#)
- SetDeviceld
  - CUsbDeviceConfigurationFunctionNet, [604](#)
- SetDeviceList
  - CPositionImpDeviceNet, [414](#)
- SetDeviceName
  - CUsbDeviceConfigurationFunctionNet, [604](#)
- SetDeviceType
  - CTcxDeviceNet, [589](#)
- SetDevname
  - CTcxDeviceNet, [589](#)
- SetDiagnosticMode
  - CIntanMea\_FunctionNet, [186](#)
- SetDigitalData
  - CMeaDigitalDataFunctionNet, [354](#), [355](#)
- SetDigitalOut
  - CMeaDeviceNet, [349](#)
- SetDigitalOutPortInvert
  - CWarnerValveControllerDeviceNet, [655](#)
- SetDigitalOutPortValve
  - CWarnerValveControllerDeviceNet, [656](#)
- SetDigitalPortDirection
  - CWarnerValveControllerDeviceNet, [656](#)
- SetDigitalSource
  - CMcsUsbDacqNet, [274–276](#)
- SetDigitalStimulatorTrigger
  - CW2100\_StimulatorFunctionNet, [619](#)
- SetDigitalStimulatorTriggerSlope
  - CW2100\_StimulatorFunctionNet, [619](#)
- SetDigout
  - CFluidControlDeviceNet, [143](#)
  - CRoboDacqNet, [459](#)
- SetDigoutMode
  - CStg200xBasicNet, [537](#)
- SetDigOutState
  - CLIH3DeviceNet, [198](#)
- SetDigoutValue
  - CStg200xBasicNet, [537](#)
- SetDIO
  - CMcsBus\_FYIExtensionNet, [203](#)
- SetDischargeCurrentSetPoint
  - CMultiBatteryChargerDeviceNet, [372](#)
- SetDisplayMode
  - CWarnerValveControllerDeviceNet, [656](#)
- SetDisplayText
  - CRoboDacqNet, [460](#)
- SetDownsampleFactor
  - CRoboDacqNet, [460](#)
- SetDSPHighPassByIndex
  - CIntanMea\_FunctionNet, [186](#)
- SetDuration
  - CMeaCoatDeviceNet, [344](#)
- SetEepromPage
  - CLIH3DeviceNet, [198](#)
- SetElectrodeDacMux
  - CStg200xBasicNet, [537](#), [538](#), [540](#)
- SetElectrodeEnable
  - CStg200xBasicNet, [541](#), [542](#)
- SetElectrodeMode
  - CStg200xBasicNet, [543](#), [544](#)
- SetEnableAmplifierProtectionSwitch
  - CStg200xBasicNet, [545](#), [546](#)
- SetEnableHeaterLimit
  - CTcxDeviceNet, [589](#)
- SetEnablePulse
  - CWarnerUssingFunctionNet, [635](#)
- SetEnableThermocouple
  - CTcxDeviceNet, [590](#)
- SetExternalElectrodeEnable
  - CStg200xBasicNet, [546](#)
- SetExternalLED
  - CTEERFunctionNet, [600](#)
- SetFAAmplification
  - CStg200xBasicNet, [547](#)
- SetFilter
  - CRoboDacqNet, [460](#)
- SetFilterCoeffs
  - CRoboDacqNet, [460](#)

- SetFilterParameter
  - CFilterConfigurationNet, [134](#)
  - CFilterConfigurationRegisterNet, [136](#)
- SetFilterParameterPermanent
  - CFilterConfigurationNet, [134](#)
  - CFilterConfigurationRegisterNet, [136](#)
- SetFilterParametersHeadstage
  - CWClassicFunctionNet, [669](#)
- SetFinalDischargeVoltage
  - CMultiBatteryChargerDeviceNet, [372](#)
- SetFrequency
  - CChannelTestDeviceNet, [98](#)
  - CRadioControlledDevicesNet, [437](#)
- SetGain
  - CPgaDeviceNet, [403](#)
- SetGate
  - CCMOSMea\_FunctionNet, [108](#)
- SetGateFloating
  - CCMOSMea\_FunctionNet, [108](#)
- SetGateToVOP
  - CCMOSMea\_FunctionNet, [108](#)
- SetGlobalRepeat
  - CDigOutStimulatorFunctionNet, [127](#)
- SetGyroRange
  - CW2100\_FunctionNet, [613](#)
- SetHasChecksum
  - CWClassicFunctionNet, [670](#)
- SetHeadstage
  - CStg200xBasicNet, [547](#)
- SetHeadstageFrameCyclesToComparePermanent
  - CSCUFunctionNet, [509](#)
- SetHeadstageLinkSpeedPermanent
  - CSCUFunctionNet, [509](#)
- SetHeadstageNumberOfAnalogChannelsPermanent
  - CSCUFunctionNet, [509](#)
- SetHeadstageOnOff
  - CW2100\_FunctionNet, [613](#)
  - CWClassicFunctionNet, [670](#)
- SetHeadstagePowerStateAtStart
  - CSCUFunctionNet, [509](#)
- SetHeadstageSampleratePermanent
  - CSCUFunctionNet, [510](#)
- SetHeadstageSamplingActive
  - CW2100\_FunctionNet, [613](#)
- SetHeadstageToSleep
  - CW2100\_FunctionNet, [613](#)
- SetHeaterLimit
  - CTcxDeviceNet, [590](#)
- SetHighCurrentMode
  - CWarnerUssingFunctionNet, [635](#)
- SetHighpassFilterEnable
  - CFilterConfigurationNet, [135](#)
- SetHWConfig
  - CRoboDeviceNet::RoboMainLowLevelCommands, [698](#)
- SetHWRevision
  - CRoboDeviceNet::RoboMainLowLevelCommands, [698](#)
- SetHWRevisionEeprom
  - CMcsBusNet, [241](#)
- SetHWSelectedChannels
  - CWClassicFunctionNet, [670](#)
- SetI
  - CTcxDeviceNet, [590](#)
- SetIClamp
  - CRoboDacqNet, [460](#)
- SetICoeff
  - CRobo\_FYITemp\_FunctionNet, [448](#)
- SetICOffset
  - CRoboDacqNet, [460](#)
- SetIdleModeOffset
  - CWarnerUssingFunctionNet, [636](#)
- SetIGain
  - CRoboDacqNet, [460](#)
- SetImpedanceTestFrequency
  - CMealImpedanceDeviceNet, [360](#)
- SetImpld
  - CPositionImpDeviceNet, [414](#)
- SetImplantCurrentSetpoint
  - CPositionIIDeviceNet, [409](#)
- SetInMovement
  - CRoboDeviceNet, [472](#)
- SetIntanRegister
  - CIntanMea\_FunctionNet, [186](#)
- SetIntBuffer
  - CGenericDevelopDeviceNet, [160](#)
- SetIO
  - CWarnerValveControllerDeviceTesterFunctionNet, [665](#)
- SetIODirection
  - CWarnerValveControllerDeviceTesterFunctionNet, [666](#)
- SetIoVoltage
  - CInterfaceboard2FunctionNet, [188](#)
- SetLatency
  - CMcsBus\_SensorNet, [229](#)
- SetLayoutConfiguration
  - CMEA2100x256FunctionNet, [332](#)
- SetLED
  - CRetinaLedDeviceNet, [439](#)
- SetLEDSwitch
  - CMcsBus\_ExtensionNet, [202](#)
- SetLength
  - CRobo\_FYIProgram\_FunctionNet, [446](#)
- SetLiquidResistance
  - CTEERFunctionNet, [601](#)
  - CWarnerUssingFunctionNet, [636](#)
- SetListmodeIndexRange
  - CStg200xBasicNet, [547](#)
- SetListmodeTriggerSource
  - CStg200xBasicNet, [547](#), [548](#)
- SetLowCurrentMode
  - CWarnerUssingFunctionNet, [636](#)
- SetLumi
  - CRetinaLedDeviceNet, [439](#)
- SetMaxCurrent

- CMeaCoatDeviceNet, [344](#)
- SetMaxDurationHighCurrentInMicroSec
  - CMultiwellOptoStimFunctionNet, [387](#)
- SetMaxDutyCycleHighCurrent
  - CMultiwellOptoStimFunctionNet, [388](#)
- SetMaxHeaterPowerMultiwell
  - CTcxDeviceNet, [590](#)
- SetMaxNoPressure
  - CRoboDeviceNet::RoboMainLowLevelCommands, [698](#)
- SetMaxNoPressureWaitTime
  - CRoboDeviceNet::RoboMainLowLevelCommands, [698](#)
- SetMaxP
  - CTcxDeviceNet, [590](#)
- SetMaxPower
  - CokuvisionStimulatorDeviceNet, [396](#)
  - CRobo\_FYITemp\_FunctionNet, [448](#)
- SetMaxPressureWaitTime
  - CRoboDeviceNet::RoboMainLowLevelCommands, [698](#)
- SetMaxVoltage
  - CMeaCleanDeviceNet, [339](#)
  - CokuvisionStimulatorDeviceNet, [396](#)
- SetMCAcceleration
  - CMcsBus\_MotorControlNet, [213](#)
- SetMCAccelerationEeprom
  - CMcsBus\_MotorControlNet, [214](#)
- SetMCAccelerationShortCommand
  - CMcsBus\_MotorControlNet, [214](#)
- SetMCAxisRevisionEeprom
  - CMcsBus\_MotorControlNet, [214](#)
- SetMCBreakCurrent
  - CMcsBus\_MotorControlNet, [214](#)
- SetMCBreakCurrentEeprom
  - CMcsBus\_MotorControlNet, [214](#)
- SetMCConfig
  - CMcsBus\_MotorControlNet, [214](#)
- SetMCConfigEeprom
  - CMcsBus\_MotorControlNet, [215](#)
- SetMCCurrent
  - CMcsBus\_MotorControlNet, [215](#)
- SetMCCurrentEeprom
  - CMcsBus\_MotorControlNet, [215](#)
- SetMCCurrentMode
  - CMcsBus\_MotorControlNet, [215](#)
- SetMCCurrentModeEeprom
  - CMcsBus\_MotorControlNet, [215](#)
- SetMCCurrentModeShortCommand
  - CMcsBus\_MotorControlNet, [215](#)
- SetMCCurrentPosition
  - CMcsBus\_MotorControlNet, [216](#)
- SetMCCurrentShortCommand
  - CMcsBus\_MotorControlNet, [216](#)
- SetMCMaxAcceleration
  - CMcsBus\_MotorControlNet, [216](#)
- SetMCMaxAccelerationEeprom
  - CMcsBus\_MotorControlNet, [216](#)
- SetMCMaxCurrent
  - CMcsBus\_MotorControlNet, [216](#)
- SetMCMaxCurrentEeprom
  - CMcsBus\_MotorControlNet, [216](#)
- SetMCMaxSpeed
  - CMcsBus\_MotorControlNet, [217](#)
- SetMCMaxSpeedEeprom
  - CMcsBus\_MotorControlNet, [217](#)
- SetMCMaxTravel
  - CMcsBus\_MotorControlNet, [217](#)
- SetMCMaxTravelEeprom
  - CMcsBus\_MotorControlNet, [217](#)
- SetMCMaxTravelShortCommand
  - CMcsBus\_MotorControlNet, [217](#)
- SetMCNewPosition
  - CMcsBus\_MotorControlNet, [217](#)
- SetMCOutputOnOff
  - CMcsBus\_MotorControlNet, [218](#)
- SetMCReference
  - CMcsBus\_MotorControlNet, [218](#)
- SetMCReferenceCurrent
  - CMcsBus\_MotorControlNet, [218](#)
- SetMCReferenceCurrentEeprom
  - CMcsBus\_MotorControlNet, [218](#)
- SetMCRegulatorGain
  - CMcsBus\_MotorControlNet, [218](#)
- SetMCRegulatorGainEeprom
  - CMcsBus\_MotorControlNet, [218](#)
- SetMCRotation
  - CMcsBus\_MotorControlNet, [219](#)
- SetMCScalingFactor
  - CMcsBus\_MotorControlNet, [219](#)
- SetMCScalingFactorEeprom
  - CMcsBus\_MotorControlNet, [219](#)
- SetMCSpeed
  - CMcsBus\_MotorControlNet, [219](#)
- SetMCSpeedEeprom
  - CMcsBus\_MotorControlNet, [219](#)
- SetMCSpeedShortCommand
  - CMcsBus\_MotorControlNet, [219](#)
- SetMCSpeedUnitEeprom
  - CMcsBus\_MotorControlNet, [220](#)
- SetMCStandbyCurrent
  - CMcsBus\_MotorControlNet, [220](#)
- SetMCStandbyCurrentEeprom
  - CMcsBus\_MotorControlNet, [220](#)
- SetMCStandbyTime
  - CMcsBus\_MotorControlNet, [220](#)
- SetMCStandbyTimeEeprom
  - CMcsBus\_MotorControlNet, [220](#)
- SetMeasurementMode
  - CStg200xBasicNet, [548](#)
- SetMinimalThreshold
  - CMcsBus\_SensorNet, [229](#)
- SetMinNoPressureWaitTime
  - CRoboDeviceNet::RoboMainLowLevelCommands, [698](#)
- SetMinPressure

- CRoboDeviceNet, [473](#)
- CRoboDeviceNet::RoboMainLowLevelCommands, [698](#)
- SetMinPressureWaitTime
  - CRoboDeviceNet::RoboMainLowLevelCommands, [698](#)
- SetMinVoltage
  - CMeaCleanDeviceNet, [339](#)
- SetModeSelect
  - CPulseGeneratorFunctionNet, [435](#)
- SetMovePump
  - CMcsBus\_SensorNet, [230](#)
- SetMultiHeadstageMode
  - CW2100\_FunctionNet, [613](#)
- SetNanoVoltsPerKelvin
  - CMcsBus\_TempSensorNet, [233](#)
- SetNeurochipMemoryData
  - CCMOSMea\_FunctionNet, [108](#)
- SetNoFilterCoeffs
  - CRoboDacqNet, [460](#)
- SetNumberOfAnalogChannels
  - CMeaDeviceNet, [350](#)
- SetNumberOfChannels
  - CMeaDeviceNet, [351](#)
  - COctoPotDeviceNet, [392](#)
- SetOffsetCurrent
  - CMeaCoatDeviceNet, [344](#)
- SetOnOff
  - CTcxDeviceNet, [590](#)
- SetOutputMap
  - CStg200xDownloadNet, [564](#)
- SetOutputRate
  - COctoPotDeviceNet, [392](#)
  - CStg200xBasicNet, [549](#)
- SetP
  - CTcxDeviceNet, [591](#)
- SetParameter
  - CRoboDeviceNet::RoboMainLowLevelCommands, [699](#)
- SetPattern
  - CMeaSwitchDeviceNet, [363](#)
- SetPatternBool
  - CMeaSwitchDeviceNet, [363](#)
- SetPatternListEntry
  - COctoPotDeviceNet, [393](#)
- SetPauseDuration
  - CMeaCoatDeviceNet, [344](#)
- SetPCoeff
  - CRobo\_FYITemp\_FunctionNet, [448](#)
- SetPeriod
  - CPulseGeneratorFunctionNet, [435](#)
- SetPeriod\_us
  - CTEERFunctionNet, [601](#)
- SetPermanentCurrentInMicroAmp
  - CMultiwellOptoStimFunctionNet, [388](#)
- SetPersistency
  - CRetinaLedDeviceNet, [439](#)
- SetPGain
  - CRoboDacqNet, [460](#)
- SetPidParameter
  - COctoPotDeviceNet, [393](#)
- SetPiezoState
  - CMcsBus\_SensorNet, [230](#)
- SetPlateMux
  - CMultiwellDeviceNet, [381](#), [382](#)
- SetPlateType
  - CMultiwellDeviceNet, [382](#)
- SetPoti
  - CMcsUsbDacqNet, [276](#)
- SetPowerMuxPlate
  - CMultiwellDeviceNet, [382](#)
- SetPowerStrength
  - CPositionIIDeviceNet, [409](#)
- SetPressureOffset
  - CMcsBus\_SensorNet, [230](#)
  - CPPCFunctionNet, [423](#)
- SetPressureRange
  - CPPCFunctionNet, [423](#)
- SetPulse
  - CWarnerUssingFunctionNet, [636](#)
- SetPulseform
  - COkuvisionStimulatorDeviceNet, [396](#)
- SetPulseLength
  - CPulseGeneratorFunctionNet, [435](#)
- SetPumpCouple
  - CPPS\_FunctionNet, [429](#)
- SetPumpEnableSpeedRatio
  - CPPS\_FunctionNet, [429](#)
- SetPumpFastOnOff
  - CPPS\_FunctionNet, [429](#)
- SetPumpFastSpeed
  - CPPS\_FunctionNet, [429](#)
- SetPumpFunctionSpeeds
  - CPPS\_FunctionNet, [429](#)
- SetPumpManualOnOff
  - CPPS\_FunctionNet, [429](#)
- SetPumpMaxSpeed
  - CPPS\_FunctionNet, [429](#)
- SetPumpModeType
  - CPPCFunctionNet, [424](#)
  - CPPS\_FunctionNet, [429](#)
- SetPumpSpeed
  - CRoboFluidDeviceNet, [481](#)
- SetPumpSpeedRatio
  - CPPS\_FunctionNet, [430](#)
- SetPumpSpeedUnit
  - CPPCFunctionNet, [424](#)
  - CPPS\_FunctionNet, [430](#)
- SetPWM
  - CFluidControlDeviceNet, [143](#)
- SetRampParameter
  - COctoPotDeviceNet, [393](#)
- SetRatedCapacity
  - CMultiBatteryChargerDeviceNet, [372](#)
- SetRatedCapacityVolatile
  - CMultiBatteryChargerDeviceNet, [374](#)



- SetRecordingNumber
  - CRoboDacqNet, [461](#)
- SetReferenceElectrodeMode
  - CSCUFunctionNet, [510](#)
- SetReferenceElectrodeSwitchState
  - CSCUFunctionNet, [510](#)
- SetRegionOfInterests
  - CCMOSMeaDeviceNet, [112](#)
  - CGrapheneASICDeviceNet, [168](#)
- SetRegulationTimeouts
  - CMcsBus\_SensorNet, [230](#)
- SetRegulatorFactor
  - CMcsBus\_SensorNet, [230](#)
- SetRegulatorOnOff
  - CMcsBus\_SensorNet, [230](#)
  - CRobo\_FYITemp\_FunctionNet, [448](#)
- SetRepeat
  - CRetinaLedDeviceNet, [440](#)
- SetRepeats
  - CProgramPressureCurveNet, [432](#)
- SetResetFilter
  - CWClassicFunctionNet, [670](#)
- SetRFFrequency
  - CPositionImpDeviceNet, [415](#)
- SetRFFrequencyHeadstage
  - CWClassicFunctionNet, [670](#)
- SetRFFrequencyReceiver
  - CWClassicFunctionNet, [670](#)
- SetRFFrequencyReceiverEeprom
  - CWClassicFunctionNet, [670](#)
- SetRFLostBehaviour
  - CWClassicFunctionNet, [670](#)
- SetRFPower
  - CWClassicFunctionNet, [671](#)
- SetRotatePump
  - CMcsBus\_SensorNet, [230](#)
- SetRTC
  - CokuvisionStimulatorDeviceNet, [396](#)
  - CPositionIIDeviceNet, [409](#)
- SetSampleInterval
  - CLIH3DeviceNet, [199](#)
- SetSamplePeriode
  - CMcsBus\_SensorNet, [231](#)
- SetSamplerate
  - CMcsUsbDacqNet, [276](#)
- SetScreen
  - CRoboDacqNet, [461](#)
- SetSearchReferenceFastAccel
  - CRoboDeviceNet::RoboMainLowLevelCommands, [699](#)
- SetSearchReferenceFastSpeed
  - CRoboDeviceNet::RoboMainLowLevelCommands, [699](#)
- SetSearchReferenceFineAccel
  - CRoboDeviceNet::RoboMainLowLevelCommands, [699](#)
- SetSearchReferenceFineSpeed
  - CRoboDeviceNet::RoboMainLowLevelCommands, [699](#)
- SetSearchReferenceMethod
  - CRoboDeviceNet::RoboMainLowLevelCommands, [699](#)
- SetSearchReferenceMoveOut
  - CRoboDeviceNet::RoboMainLowLevelCommands, [700](#)
- SetSearchReferenceOffsetPos
  - CRoboDeviceNet::RoboMainLowLevelCommands, [700](#)
- SetSelectedChannels
  - CMcsUsbDacqNet, [277–279](#)
  - CW2100\_FunctionNet, [613](#)
- SetSelectedChannelsQueue
  - CMcsUsbDacqNet, [279–281](#)
- SetSelectedData
  - CMcsUsbDacqNet, [281–283](#)
- SetSelectedHeadstage
  - CWClassicFunctionNet, [671](#)
- SetSensorType
  - CTcxDeviceNet, [591](#)
- SetSerialNumberHeadstage
  - CWClassicFunctionNet, [671](#)
- SetSetpoint
  - CTcxDeviceNet, [591](#)
- SetShortBuffer
  - CGenericDevelopDeviceNet, [161](#)
- SetSimulation
  - CRoboDacqNet, [461](#)
- SetSineParameter
  - COctoPotDeviceNet, [393](#)
- SetSingleHeater
  - CMcsBus\_FYIExtensionNet, [203](#)
- SetSingleValve
  - CFluidControlDeviceNet, [143](#)
  - CRoboFluidDeviceNet, [481](#)
- SetSlope
  - CMeaCleanDeviceNet, [339](#)
  - CMeaCoatDeviceNet, [345](#)
- SetSoftwareKey
  - CMcsUsbNet, [324](#)
- SetSollPressure
  - CMcsBus\_SensorNet, [231](#)
- SetSollTemp
  - CRobo\_FYITemp\_FunctionNet, [448](#)
- SetSourceBulk
  - CCMOSMea\_FunctionNet, [108](#)
- SetSourceDrain
  - CCMOSMea\_FunctionNet, [109](#)
- SetSourceGate
  - CCMOSMea\_FunctionNet, [109](#)
- SetSpeedI
  - CRoboStatorDeviceNet, [489](#)
- SetSpeedNative
  - CRoboStatorDeviceNet, [489](#)
- SetSpeedNativeXY
  - CRoboStatorDeviceNet, [489](#)

- SetSpeedNativeZ
  - CRoboStatorDeviceNet, [489](#)
- SetSpeedXY
  - CRoboStatorDeviceNet, [489](#)
- SetSpeedZ
  - CRoboStatorDeviceNet, [489](#)
- SetStartTriggerSlope
  - CDigOutStimulatorFunctionNet, [128](#)
- SetStateDebugData
  - CPositionIIDeviceNet, [411](#)
- SetStateEventData
  - CPositionIIDeviceNet, [411](#)
- SetStgProgramInfo
  - CStg200xBasicNet, [549](#)
- SetStimulusSites
  - CCMOSMea\_FunctionNet, [109](#)
- SetStopTriggerSlope
  - CDigOutStimulatorFunctionNet, [128](#)
- SetStringFormat
  - CMcsUsbListEntryNet, [305](#)
  - CMcsUsbListNet, [309](#)
- SetSubChannel
  - CMcsBus\_MotorControlNet, [220](#)
- SetSwitches
  - CSafeISDeviceNet, [493](#)
- SetSyncoutMap
  - CStg200xBasicNet, [549](#)
- SetTableName
  - CWarnerValveControllerDeviceNet, [656](#)
- SetTablepointer
  - CRetinaLedDeviceNet, [440](#)
- SetTableStep
  - CWarnerValveControllerDeviceNet, [657](#)
- SetTableStepAll
  - CWarnerValveControllerDeviceNet, [657](#)
- SetTestMode
  - CRFFFunctionNet, [444](#)
- SetThermocoupleNanovoltPerKelvin
  - CFluidControlDeviceNet, [145](#)
  - CTcxDeviceNet, [591](#)
- SetThermoOffset
  - CMcsBus\_TempSensorNet, [233](#)
- Settings
  - Mcs::Usb, [92](#)
- SetTouchPadEnable
  - CMultiwellDeviceNet, [383](#)
- SetTransformer
  - CMeFunctionNet, [366](#)
- SetTrigger
  - CRetinaLedDeviceNet, [440](#)
  - CWarnerValveControllerDeviceTesterFunctionNet, [666](#)
- SetTriggerMaskValue
  - CMeaDeviceNet, [351](#)
  - CRoboDacqNet, [461](#)
- SetTriggerPeriod
  - CMeaDeviceNet, [352](#)
- SetTriggerSource
  - CStg200xBasicNet, [549](#), [550](#)
- SetTriggerSyncDirection
  - CWarnerValveControllerDeviceTesterFunctionNet, [666](#)
- SetUByteBuffer
  - CGenericDevelopDeviceNet, [162](#)
- SetUClamp
  - CRoboDacqNet, [461](#)
- SetUCOffset
  - CRoboDacqNet, [461](#)
- SetUIntBuffer
  - CGenericDevelopDeviceNet, [162](#)
- SetupGroupDacqQueue
  - CMcsUsbDacqNet, [283](#)
- SetupRetriggerMode
  - CStg200xDownloadBasicNet, [556](#)
- SetupTrigger
  - CStg200xDownloadBasicNet, [557](#)
  - CStimulusFunctionNet, [574](#)
- SetupTriggerSingle
  - CStg200xDownloadBasicNet, [557](#)
  - CStimulusFunctionNet, [575](#)
- SetUseBubble
  - CPPS\_FunctionNet, [430](#)
- SetUserParameter
  - CRoboDeviceNet::RoboMainLowLevelCommands, [700](#)
- SetUShortBuffer
  - CGenericDevelopDeviceNet, [163](#)
- SetUVOffset
  - CRoboDacqNet, [461](#)
- SetValue
  - CGenericDevelopDeviceNet, [163](#)
- SetValve
  - CFluidControlDeviceNet, [145](#)
  - CRoboFluidDeviceNet, [481](#)
- SetValve1
  - CRobo\_FYIProgram\_FunctionNet, [446](#)
- SetValve2
  - CRobo\_FYIProgram\_FunctionNet, [446](#)
- SetValveActive
  - CPPCFunctionNet, [424](#)
  - CWarnerValveControllerDeviceNet, [657](#)
- SetValveCurrent
  - CWarnerValveControllerDeviceNet, [657](#)
- SetValveDigitalInInvert
  - CWarnerValveControllerDeviceNet, [658](#)
- SetValveDigitalInPort
  - CWarnerValveControllerDeviceNet, [658](#)
- SetValveLedOn
  - CWarnerValveControllerDeviceNet, [658](#)
- SetValveManualGroup
  - CWarnerValveControllerDeviceNet, [658](#)
- SetValveManualState
  - CWarnerValveControllerDeviceNet, [659](#)
- SetValveMode
  - CWarnerValveControllerDeviceNet, [659](#)
- SetValves



- CMcsBus\_FYIExtensionNet, [203](#)
- SetValvesActiveMap
  - CWarnerValveControllerDeviceNet, [659](#)
- SetValvesManualStateMap
  - CWarnerValveControllerDeviceNet, [659](#)
- SetValveTableEntry
  - CWarnerValveControllerDeviceNet, [659](#)
- SetVdsVgs
  - CGrapheneFunctionNet, [176](#)
- SetVdVs
  - CGrapheneFunctionNet, [177](#)
- SetVdVsDAC
  - CGrapheneFunctionNet, [177](#)
- SetVelocityI
  - CRoboStatorDeviceNet, [489](#)
- SetVelocityXY
  - CRoboStatorDeviceNet, [489](#)
- SetVelocityZ
  - CRoboStatorDeviceNet, [489](#)
- SetVMMaxNegativeCurrent
  - CMcsBus\_VoltageModeNet, [236](#)
- SetVMMaxNegativeCurrentEeprom
  - CMcsBus\_VoltageModeNet, [236](#)
- SetVMMaxNegativeVoltage
  - CMcsBus\_VoltageModeNet, [236](#)
- SetVMMaxNegativeVoltageEeprom
  - CMcsBus\_VoltageModeNet, [236](#)
- SetVMMaxPositiveCurrent
  - CMcsBus\_VoltageModeNet, [237](#)
- SetVMMaxPositiveCurrentEeprom
  - CMcsBus\_VoltageModeNet, [237](#)
- SetVMMaxPositiveVoltage
  - CMcsBus\_VoltageModeNet, [237](#)
- SetVMMaxPositiveVoltageEeprom
  - CMcsBus\_VoltageModeNet, [237](#)
- SetVMOutputOnOff
  - CMcsBus\_VoltageModeNet, [237](#)
- SetVMVoltage
  - CMcsBus\_VoltageModeNet, [237](#)
- SetVolatileClampOffset
  - CMultiwellDeviceNet, [383](#)
- SetVoltage12VLimit
  - CRoboDeviceNet, [473](#)
- SetVoltage5VLimit
  - CRoboDeviceNet, [473](#)
- SetVoltageAirvalveLimit
  - CRoboDeviceNet, [473](#)
- SetVoltageClampControllerParam\_D
  - CWarnerUssingFunctionNet, [637](#)
- SetVoltageClampControllerParam\_I
  - CWarnerUssingFunctionNet, [637](#)
- SetVoltageClampControllerParam\_P
  - CWarnerUssingFunctionNet, [637](#)
- SetVoltageMode
  - CStg200xBasicNet, [550](#)
- SetVoltageRange
  - CGrapheneFunctionNet, [178](#)
- SetVoltageRangeByIndex
  - CMcsUsbDacqNet, [283](#)
- SetVoltageRangeInMicroVolt
  - CMcsUsbDacqNet, [284](#)
- SetVoltageRangeSelectedIndex
  - CStg200xBasicNet, [550](#)
- SetVoltageResolution
  - CGrapheneFunctionNet, [178](#)
- SetVoltageRs485ALimit
  - CRoboDeviceNet, [473](#)
- SetVoltageRs485BLimit
  - CRoboDeviceNet, [473](#)
- SetVoltageValvesLimit
  - CRoboDeviceNet, [473](#)
- SetWaveform
  - CChannelTestDeviceNet, [98](#)
  - CTEERFunctionNet, [601](#)
- SetWaveLengthInNanometer
  - CMultiwellOptoStimFunctionNet, [388](#)
- SetWorkingFrequency
  - CRFFFunctionNet, [444](#)
- SetWPADebugMode
  - CWClassicFunctionNet, [671](#)
- SetWPAType
  - CWClassicFunctionNet, [671](#)
- SetXGain
  - CRoboDacqNet, [461](#)
- Sideband
  - CStimulusFunctionNet::SidebandData, [703](#)
- SidebandData
  - CStimulusFunctionNet::SidebandData, [703](#)
- Signed\_16bit
  - Mcs::Usb, [61](#)
- Signed\_24bit
  - Mcs::Usb, [61](#)
- Signed\_32bit
  - Mcs::Usb, [61](#)
- Sine
  - Mcs::Usb, [86](#)
- SineStart
  - COctoPotDeviceNet, [393](#)
- SingleWell
  - Mcs::Usb, [84](#)
- SixWell
  - Mcs::Usb, [84](#)
- size
  - DigitalSource< digitalsourceenum >, [674](#)
  - DigitalSourceGeneral, [676](#)
- SmartImplant
  - Mcs::Usb, [78](#)
- SN
  - HeadStageIDType, [688](#)
- SOFAndCTRLword
  - Mcs::Usb, [58](#)
- Software
  - Mcs::Usb, [66](#)
- SoftwareDongle
  - Mcs::Usb, [77](#)
- Source

- DigitalSource< digitalsourceenum >, 675
- DigitalSourceGeneral, 676
- Standby
  - Mcs::Usb, 79, 88
- Start
  - CMeaCleanDeviceNet, 339
  - CMeaCoatDeviceNet, 345
  - CRobo\_FYIPProgram\_FunctionNet, 446
  - Mcs::Usb, 64
- StartDacq
  - CMcsUsbDacqNet, 284, 285
- StartInternalCalibration
  - CTEERFunctionNet, 601
- StartLoop
  - CMcsUsbDacqNet, 286, 287
- StartMCMovement
  - CMcsBus\_MotorControlNet, 221
- StartMeasurement
  - CMealImpedanceDeviceNet, 360
- StartPoll
  - CStimulusFunctionNet, 575
  - CW2100\_StimulatorFunctionNet, 619
- StartQueue
  - CRoboDeviceNet, 473
- StartSampling
  - CTEERFunctionNet, 601
- StartSync
  - CMcsBus\_SensorNet, 231
- State
  - HeadStageIDTypeState, 690
  - Mcs::Usb, 89
- Status
  - CUsbExceptionNet, 605
- Status\_AlreadyConfigured
  - CMcsUsbNet, 326
- Status\_BadStartFrame
  - CMcsUsbNet, 326
- Status\_Btstuff
  - CMcsUsbNet, 326
- Status\_BufferOverrun
  - CMcsUsbNet, 326
- Status\_BufferUnderrun
  - CMcsUsbNet, 326
- Status\_Canceled
  - CMcsUsbNet, 326
- Status\_Canceling
  - CMcsUsbNet, 327
- Status\_ConnectedPipes
  - CMcsUsbNet, 327
- Status\_ControlNotOwned
  - CMcsUsbNet, 327
- Status\_Crc
  - CMcsUsbNet, 327
- Status\_DataOverrun
  - CMcsUsbNet, 327
- Status\_DataToggleMismatch
  - CMcsUsbNet, 327
- Status\_DataUnderrun
  - CMcsUsbNet, 327
- Status\_DeviceLocked
  - CMcsUsbNet, 327
- Status\_DeviceNotFound
  - CMcsUsbNet, 327
- Status\_DeviceRemoved
  - CMcsUsbNet, 327
- Status\_DevNotResponding
  - CMcsUsbNet, 327
- Status\_EndpointHalted
  - CMcsUsbNet, 328
- Status\_ErrorBusy
  - CMcsUsbNet, 328
- Status\_ErrorShortTransfer
  - CMcsUsbNet, 328
- Status\_Fifo
  - CMcsUsbNet, 328
- Status\_FrameControlOwned
  - CMcsUsbNet, 328
- Status\_InternalHcError
  - CMcsUsbNet, 328
- Status\_InvalidDeviceHandle
  - CMcsUsbNet, 328
- Status\_InvalidHandle
  - CMcsUsbNet, 328
- Status\_InvalidParameter
  - CMcsUsbNet, 328
- Status\_InvalidPipeHandle
  - CMcsUsbNet, 328
- Status\_InvalidUrbFunction
  - CMcsUsbNet, 329
- Status\_IoPending
  - CMcsUsbNet, 329
- Status\_IoTimeout
  - CMcsUsbNet, 329
- Status\_IsochRequestFailed
  - CMcsUsbNet, 329
- Status\_LastUsbErrorMismatch
  - CMcsUsbNet, 329
- Status\_NoBandwidth
  - CMcsUsbNet, 329
- Status\_NoMemory
  - CMcsUsbNet, 329
- Status\_NoSuchDevice
  - CMcsUsbNet, 329
- Status\_NotAccessed
  - CMcsUsbNet, 329
- Status\_NotSupported
  - CMcsUsbNet, 329
- Status\_PidCheckFailure
  - CMcsUsbNet, 329
- Status\_PipeNotLinked
  - CMcsUsbNet, 330
- Status\_RequestFailed
  - CMcsUsbNet, 330
- Status\_RequestMutexFailed
  - CMcsUsbNet, 330
- Status\_RequestMutexTimeout
  - CMcsUsbNet, 330

- CMcsUsbNet, 330
- Status\_Stall
  - CMcsUsbNet, 330
- Status\_Unconfigured
  - CMcsUsbNet, 330
- Status\_UnexpectedPid
  - CMcsUsbNet, 330
- STG
  - Mcs::Usb, 76
- Stg1
  - Mcs::Usb, 65
- STG1DACSignalGroup
  - Mcs::Usb, 59, 71, 81
- STG1SidebandsGroup
  - Mcs::Usb, 59, 71, 81
- STG1TriggerStatusGroup
  - Mcs::Usb, 59, 71, 81
- Stg2
  - Mcs::Usb, 65
- Stg200xDigoutModeEnumNet
  - Mcs::Usb, 82
- Stg200xPollStatusEvent
  - CStg200xDownloadNet, 565
- Stg200xSegmentFlagsEnumNet
  - Mcs::Usb, 83
- Stg200xTriggerStatusEnumNet
  - Mcs::Usb, 83
- STG2DACSignalGroup
  - Mcs::Usb, 71, 81
- STG2SidebandsGroup
  - Mcs::Usb, 71, 81
- STG2TriggerStatusGroup
  - Mcs::Usb, 71, 81
- Stg3
  - Mcs::Usb, 65
- STG3008\_FA
  - Mcs::Usb, 76
- 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
- STG400x
  - Mcs::Usb, 76
- STG400x\_opto
  - Mcs::Usb, 76
- STG5
  - Mcs::Usb, 76
- STG\_DestinationEnumNet
  - Mcs::Usb, 83
- StgListModeTrigger
  - Mcs::Usb, 64
- StgStatusNet, 704
  - FromIntPtr, 704
  - FromPtr, 704
  - ListOfChangedTriggers, 704
  - TiggerStatus, 704
- StgTrigger
  - Mcs::Usb, 64
- StillConnected
  - CRadioControlledDevicesNet, 437
- Stimulation
  - Mcs::Usb, 56
- StimulationLayoutConfigurationEnumNet
  - Mcs::Usb, 84
- Stimulator
  - CW2100\_FunctionNet, 614
- Stimulus
  - CCMOSMeaDeviceNet, 112
  - CStg200xDownloadBasicNet, 558
- StimulusDeviceDataAndUnrolledData
  - CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, 705
- StimulusFunction
  - CLIH3DeviceNet, 199
- StimulusParameters
  - HeadStageIDType, 688
- Stop
  - CMeaCleanDeviceNet, 339
  - CMeaCoatDeviceNet, 345
  - Mcs::Usb, 64, 74
- StopDacq
  - CMcsUsbDacqNet, 288
- StopLoop
  - CMcsUsbDacqNet, 288
- StopMCMovement
  - CMcsBus\_MotorControlNet, 221
- StopMovement
  - CRoboDeviceNet, 474
- StopMovementI
  - CRoboStatorDeviceNet, 490
- StopMovementXY
  - CRoboStatorDeviceNet, 490
- StopMovementZ
  - CRoboStatorDeviceNet, 490
- StopPlateClamp
  - CMultiwellDeviceNet, 383
- StopPoll
  - CStimulusFunctionNet, 575
  - CW2100\_StimulatorFunctionNet, 619
- StopSampling
  - CTEERFunctionNet, 602
- StopTable
  - CRoboDacqNet, 461, 462
- StorageCharge
  - Mcs::Usb, 70
- StoreValveTable
  - CWarnerValveControllerDeviceNet, 660
- SubtractFromAll

- Mcs::Usb, [78](#)
- SubtractFromAllOther
  - Mcs::Usb, [78](#)
- SubtractFromReferenceElectrodeOnly
  - Mcs::Usb, [78](#)
- SubtractionOff
  - Mcs::Usb, [78](#)
- SuperSpeed
  - Mcs::Usb, [71](#)
- Sw2to64
  - Mcs::Usb, [76](#)
- SwitchOnOff
  - CPositionIIDeviceNet, [411](#)
- SYNC\_BIT0
  - CW2100\_StimulatorFunctionNet, [620](#)
- SYNC\_BIT1
  - CW2100\_StimulatorFunctionNet, [620](#)
- 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, [84](#)
- SyncStart
  - Mcs::Usb, [83](#)
- Table
  - Mcs::Usb, [92](#)
- Table\_Wait
  - CRoboDacqNet, [462](#)
- TableDefBegin
  - CRoboDacqNet, [462](#)
- TableDefEnd
  - CRoboDacqNet, [462](#)
- TableEntryChangedEvent
  - CWarnerValveControllerDeviceNet, [663](#)
- TactSwitchGetState
  - CMcsBus\_SensorNet, [231](#)
- TactSwitchSetDisplay
  - CMcsBus\_SensorNet, [231](#)
- TBSI\_127
  - Mcs::Usb, [51](#)
- TBSI\_15
  - Mcs::Usb, [51](#)
- TBSI\_31
  - Mcs::Usb, [51](#)
- TBSI\_5
  - Mcs::Usb, [51](#)
- TBSI\_63
  - Mcs::Usb, [51](#)
- TBSI\_Dacq
  - Mcs::Usb, [77](#)
- TBSI\_DACQDigitalSourceEnumNet
  - Mcs::Usb, [84](#)
- TBSI\_Reserved
  - Mcs::Usb, [51](#)
- TbsiDacq
  - Mcs::Usb, [68](#)
- TbsiDacqHeadstage
  - Mcs::Usb, [67](#)
- TbsiDacqInterfaceboard
  - Mcs::Usb, [67](#)
- TC01
  - Mcs::Usb, [76](#)
- TC02
  - Mcs::Usb, [76](#)
- TCX
  - Mcs::Usb, [76](#)
- TcxDeviceTypeEnumNet
  - Mcs::Usb, [85](#)
- TcxSensorTypeEnumNet
  - Mcs::Usb, [85](#)
- TeerClampModeEnumNet
  - Mcs::Usb, [85](#)
- TEERFunctionNet
  - CTEERMachineDeviceNet, [603](#)
- TeerWaveformEnumNet
  - Mcs::Usb, [85](#)
- Tersens
  - Mcs::Usb, [76](#)
- Test\_ADC\_EPC10
  - Mcs::Usb, [68](#)
- Test\_DAC\_EPC10
  - Mcs::Usb, [69](#)
- ThrowCUsbExceptionNetOnError
  - CMcsUsbFunctionNet, [301](#)
  - CMcsUsbNet, [324](#)
- TiggerStatus
  - StgStatusNet, [704](#)
- TimeResolutionInNanoSeconds
  - W2100\_StimulusParametersNet, [707](#)
- Timestamp
  - Mcs::Usb, [58](#)
- ToCpp
  - CFilterCoefficientsNet::s\_FilterAttributesNet, [701](#)
- ToString
  - CFilterPropertyNet, [137](#)
  - CMcsUsbListEntryNet, [306](#)
  - HeadstageIDType, [687](#)
  - HeadstageIDTypeObject, [689](#)
- TouchTest
  - Mcs::Usb, [92](#)
- Triggerbox\_AMS
  - Mcs::Usb, [76](#)
- Triggerbox\_AMS3

Mcs::Usb, [76](#)  
Triggerbox\_IMS  
Mcs::Usb, [76](#)  
Triggerbox\_R5  
Mcs::Usb, [76](#)  
TriggerMask\_Default  
CRoboDacqNet, [462](#)  
TriggerOnly  
Mcs::Usb, [83](#)  
TriggerSourceEnumNet  
Mcs::Usb, [86](#)  
TriggerStatus  
CMcsUsbDeviceStatePushFunctionNet, [290](#)  
CMcsUsbDeviceStatePushNet, [291](#)  
TriggerStatus1  
Mcs::Usb, [57](#)  
TriggerStatus2  
Mcs::Usb, [57](#)  
TriggerStatus3  
Mcs::Usb, [57](#)  
TriggerStatus4  
Mcs::Usb, [57](#)  
TriggerValue\_MoveAbs  
CRoboDacqNet, [462](#)  
TriggerValue\_StartQueue  
CRoboDacqNet, [462](#)  
tsAuxIn1  
Mcs::Usb, [87](#)  
tsAuxIn2  
Mcs::Usb, [87](#)  
tsDACQCy1Dev1Runs  
Mcs::Usb, [88](#)  
tsDACQCy1Dev2Runs  
Mcs::Usb, [88](#)  
tsDACQCy2Dev1Runs  
Mcs::Usb, [88](#)  
tsDACQCy2Dev2Runs  
Mcs::Usb, [88](#)  
tsDigitalIn1  
Mcs::Usb, [86](#)  
tsDigitalIn10  
Mcs::Usb, [86](#)  
tsDigitalIn11  
Mcs::Usb, [86](#)  
tsDigitalIn12  
Mcs::Usb, [86](#)  
tsDigitalIn13  
Mcs::Usb, [86](#)  
tsDigitalIn14  
Mcs::Usb, [86](#)  
tsDigitalIn15  
Mcs::Usb, [86](#)  
tsDigitalIn16  
Mcs::Usb, [86](#)  
tsDigitalIn17  
Mcs::Usb, [86](#)  
tsDigitalIn18  
Mcs::Usb, [86](#)  
tsDigitalIn19  
Mcs::Usb, [86](#)  
tsDigitalIn2  
Mcs::Usb, [86](#)  
tsDigitalIn20  
Mcs::Usb, [86](#)  
tsDigitalIn21  
Mcs::Usb, [86](#)  
tsDigitalIn22  
Mcs::Usb, [86](#)  
tsDigitalIn23  
Mcs::Usb, [86](#)  
tsDigitalIn24  
Mcs::Usb, [86](#)  
tsDigitalIn25  
Mcs::Usb, [86](#)  
tsDigitalIn26  
Mcs::Usb, [86](#)  
tsDigitalIn27  
Mcs::Usb, [86](#)  
tsDigitalIn28  
Mcs::Usb, [86](#)  
tsDigitalIn29  
Mcs::Usb, [86](#)  
tsDigitalIn3  
Mcs::Usb, [86](#)  
tsDigitalIn30  
Mcs::Usb, [86](#)  
tsDigitalIn31  
Mcs::Usb, [86](#)  
tsDigitalIn32  
Mcs::Usb, [86](#)  
tsDigitalIn4  
Mcs::Usb, [86](#)  
tsDigitalIn5  
Mcs::Usb, [86](#)  
tsDigitalIn6  
Mcs::Usb, [86](#)  
tsDigitalIn7  
Mcs::Usb, [86](#)  
tsDigitalIn8  
Mcs::Usb, [86](#)  
tsDigitalIn9  
Mcs::Usb, [86](#)  
tsDigitalPuse0  
Mcs::Usb, [87](#)  
tsDigitalPuse1  
Mcs::Usb, [87](#)  
tsDigitalPuse10  
Mcs::Usb, [87](#)  
tsDigitalPuse11  
Mcs::Usb, [87](#)  
tsDigitalPuse12  
Mcs::Usb, [87](#)  
tsDigitalPuse13  
Mcs::Usb, [87](#)  
tsDigitalPuse14  
Mcs::Usb, [87](#)

tsDigitalPuse15  
Mcs::Usb, [87](#)  
tsDigitalPuse16  
Mcs::Usb, [87](#)  
tsDigitalPuse17  
Mcs::Usb, [87](#)  
tsDigitalPuse18  
Mcs::Usb, [88](#)  
tsDigitalPuse19  
Mcs::Usb, [88](#)  
tsDigitalPuse2  
Mcs::Usb, [87](#)  
tsDigitalPuse20  
Mcs::Usb, [88](#)  
tsDigitalPuse21  
Mcs::Usb, [88](#)  
tsDigitalPuse22  
Mcs::Usb, [88](#)  
tsDigitalPuse23  
Mcs::Usb, [88](#)  
tsDigitalPuse24  
Mcs::Usb, [88](#)  
tsDigitalPuse25  
Mcs::Usb, [88](#)  
tsDigitalPuse26  
Mcs::Usb, [88](#)  
tsDigitalPuse27  
Mcs::Usb, [88](#)  
tsDigitalPuse28  
Mcs::Usb, [88](#)  
tsDigitalPuse29  
Mcs::Usb, [88](#)  
tsDigitalPuse3  
Mcs::Usb, [87](#)  
tsDigitalPuse30  
Mcs::Usb, [88](#)  
tsDigitalPuse31  
Mcs::Usb, [88](#)  
tsDigitalPuse4  
Mcs::Usb, [87](#)  
tsDigitalPuse5  
Mcs::Usb, [87](#)  
tsDigitalPuse6  
Mcs::Usb, [87](#)  
tsDigitalPuse7  
Mcs::Usb, [87](#)  
tsDigitalPuse8  
Mcs::Usb, [87](#)  
tsDigitalPuse9  
Mcs::Usb, [87](#)  
tsFeedback1  
Mcs::Usb, [86](#)  
tsFeedback10  
Mcs::Usb, [87](#)  
tsFeedback11  
Mcs::Usb, [87](#)  
tsFeedback12  
Mcs::Usb, [87](#)  
tsFeedback13  
Mcs::Usb, [87](#)  
tsFeedback14  
Mcs::Usb, [87](#)  
tsFeedback15  
Mcs::Usb, [87](#)  
tsFeedback16  
Mcs::Usb, [87](#)  
tsFeedback17  
Mcs::Usb, [87](#)  
tsFeedback18  
Mcs::Usb, [87](#)  
tsFeedback19  
Mcs::Usb, [87](#)  
tsFeedback2  
Mcs::Usb, [86](#)  
tsFeedback20  
Mcs::Usb, [87](#)  
tsFeedback21  
Mcs::Usb, [87](#)  
tsFeedback22  
Mcs::Usb, [87](#)  
tsFeedback23  
Mcs::Usb, [87](#)  
tsFeedback24  
Mcs::Usb, [87](#)  
tsFeedback25  
Mcs::Usb, [87](#)  
tsFeedback26  
Mcs::Usb, [87](#)  
tsFeedback27  
Mcs::Usb, [87](#)  
tsFeedback28  
Mcs::Usb, [87](#)  
tsFeedback29  
Mcs::Usb, [87](#)  
tsFeedback3  
Mcs::Usb, [86](#)  
tsFeedback30  
Mcs::Usb, [87](#)  
tsFeedback31  
Mcs::Usb, [87](#)  
tsFeedback32  
Mcs::Usb, [87](#)  
tsFeedback4  
Mcs::Usb, [87](#)  
tsFeedback5  
Mcs::Usb, [87](#)  
tsFeedback6  
Mcs::Usb, [87](#)  
tsFeedback7  
Mcs::Usb, [87](#)  
tsFeedback8  
Mcs::Usb, [87](#)  
tsFeedback9  
Mcs::Usb, [87](#)  
tsNone  
Mcs::Usb, [86](#)

- tsSidebandBit8
  - Mcs::Usb, [88](#)
- tsTriggered
  - Mcs::Usb, [88](#)
- TxnGetSerialNumber
  - CMcsUsbNet, [324](#)
- TxnSetSerialNumber
  - CMcsUsbNet, [324](#)
- TxnTestMemoryReadAndCheck
  - CMcsUsbNet, [324](#)
- TxnTestMemoryWrite
  - CMcsUsbNet, [324](#)
- Type
  - HeadStageIDType, [688](#)
- TypeValue
  - HeadStageIDType, [688](#)
- Unknown
  - HeadStageIDType, [686](#)
  - Mcs::Usb, [51](#), [52](#), [66](#), [85](#), [88](#)
- unknown
  - Mcs::Usb, [65](#)
- UnknownDest
  - Mcs::Usb, [56](#)
- UnknownSpeed
  - Mcs::Usb, [71](#)
- Unlock
  - Mcs::Usb, [75](#)
- UnlockPlateClamp
  - CMultiwellDeviceNet, [383](#)
- UnrolledAmplitude
  - CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, [705](#)
- UnrolledDuration
  - CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, [705](#)
- UnrolledSync
  - CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, [705](#)
- Unsigned\_16bit
  - Mcs::Usb, [61](#)
- Unsigned\_24bit
  - Mcs::Usb, [61](#)
- Unsigned\_32bit
  - Mcs::Usb, [61](#)
- UpdateChannelBlock
  - CCMOSMeaDeviceNet, [112](#)
- UpdateDisplay
  - CRoboDacqNet, [462](#)
- UpdateFirmware
  - CMcsUsbFactoryNet, [297–299](#)
- UpdateTransistorVoltages
  - CCMOSMea\_FunctionNet, [109](#)
- UpdateTrigger
  - Mcs::Usb, [83](#)
- USB
  - FirmwareDestinationNames, [685](#)
  - Mcs::Usb, [52](#)
- USB\_TARGET1
  - Mcs::Usb, [55](#)
- USB\_TARGET2
  - Mcs::Usb, [55](#)
- USB\_TARGET3
  - Mcs::Usb, [56](#)
- usbSetupPacket\_t, [706](#)
  - bmRequestType, [706](#)
  - bRequest, [706](#)
  - wIndex, [706](#)
  - wLength, [706](#)
  - wValue, [706](#)
- UsbTest
  - Mcs::Usb, [77](#)
- UsbVendorIdEnumNet
  - Mcs::Usb, [88](#)
- User\_ADC\_0
  - Mcs::Usb, [68](#)
- User\_ADC\_1
  - Mcs::Usb, [68](#)
- User\_ADC\_2
  - Mcs::Usb, [68](#)
- User\_ADC\_3
  - Mcs::Usb, [68](#)
- User\_ADC\_4
  - Mcs::Usb, [68](#)
- User\_DAC\_0
  - Mcs::Usb, [69](#)
- User\_DAC\_1
  - Mcs::Usb, [69](#)
- User\_DAC\_2
  - Mcs::Usb, [69](#)
- UserDefinedName
  - HeadStageIDType, [688](#)
- UssingChamber
  - Mcs::Usb, [67](#)
- UssingClampModeEnumNet
  - Mcs::Usb, [88](#)
- UssingRail
  - Mcs::Usb, [67](#)
- UssingUnitEnumNet
  - Mcs::Usb, [89](#)
- Valid
  - HeadStageIDType, [688](#)
- ValidKey
  - CMcsUsbNet, [324](#)
- VendorIdEnumNet
  - Mcs::Usb, [89](#)
- VendorInRequest
  - CGenericDevelopDeviceNet, [164](#)
- VendorOutRequest
  - CGenericDevelopDeviceNet, [164](#)
- VirtualDevice\_ContinuousDacq
  - CRoboDacqNet, [462](#)
- VirtualDevice\_TableRun
  - CRoboDacqNet, [462](#)
- Volt
  - Mcs::Usb, [52](#), [89](#)
- Voltage



BatteryState, [95](#)  
 Voltage\_3V3  
     Mcs::Usb, [68](#)  
 Voltage\_5V0  
     Mcs::Usb, [68](#)  
 VoltageClamp  
     Mcs::Usb, [88](#)  
 VoltageRangeDisplayStringMilliVolt  
     CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet,  
         [606](#)  
 VoltageRangeInMicroVolt  
     CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet,Whole\_Cell\_Patch  
         [606](#)  
         W2100\_StimulusParametersNet, [707](#)  
 VoltageResolutionInMicroVolt  
     W2100\_StimulusParametersNet, [707](#)  
 VoltageString  
     BatteryState, [95](#)  
 VOPSTimerSetResetTimes  
     CCMOSMea\_FunctionNet, [109](#)  
  
 W16IsW14  
     HeadStageIDType, [688](#)  
 W2100  
     Mcs::Usb, [77](#)  
 W2100\_Accel\_Gyro\_Select\_EnumNet  
     Mcs::Usb, [89](#)  
 W2100\_FunctionNet  
     CMeaDeviceNet, [353](#)  
 W2100\_StimulusParametersNet, [706](#)  
     CurrentRangeInNanoAmp, [707](#)  
     CurrentResolutionInNanoAmp, [707](#)  
     DACResolution, [707](#)  
     TimeResolutionInNanoSeconds, [707](#)  
     VoltageRangeInMicroVolt, [707](#)  
     VoltageResolutionInMicroVolt, [707](#)  
 W2100DacqGroupChannelEnumNet  
     Mcs::Usb, [89](#)  
 W2100DigitalSourceEnumNet  
     Mcs::Usb, [91](#)  
 W2100IFB2  
     Mcs::Usb, [67](#)  
 W2100Interfaceboard  
     Mcs::Usb, [67](#)  
 W2100WirelessReceiver  
     Mcs::Usb, [67](#), [68](#)  
 W2100WirelessReceiverAnalog  
     Mcs::Usb, [67](#), [68](#)  
 WaitForAllChambers  
     CWarnerUssingFunctionNet, [638](#)  
 WaitForChamber  
     CWarnerUssingFunctionNet, [638](#)  
 WaitTimer  
     CRoboDeviceNet, [474](#)  
 Warner  
     Mcs::Usb, [85](#)  
 Warner\_TEER\_Machine  
     Mcs::Usb, [78](#)  
 Warner\_Ussing  
     Mcs::Usb, [78](#)  
 WARNER\_USSING\_DEVICE  
     Mcs::Usb, [62](#)  
 Warner\_Valve\_Control  
     Mcs::Usb, [78](#)  
 WARNER\_VALVE\_CONTROL\_DEVICE  
     Mcs::Usb, [62](#)  
 WarnerUssingFunction  
     CWarnerUssingDeviceNet, [622](#)  
 WClassicFunctionNet  
     CMeaDeviceNet, [353](#)  
 Whole\_Cell\_Patch  
     Mcs::Usb, [77](#)  
 WholeCellPatch  
     Mcs::Usb, [80](#)  
 WholeCellPatchHeadstage  
     Mcs::Usb, [67](#)  
 wIndex  
     usbSetupPacket\_t, [706](#)  
 WirelessHeadStageAccDataRE1HS1  
     Mcs::Usb, [90](#)  
 WirelessHeadStageAccDataRE1HS2  
     Mcs::Usb, [90](#)  
 WirelessHeadStageAccDataRE1HS3  
     Mcs::Usb, [90](#)  
 WirelessHeadStageAccDataRE1HS4  
     Mcs::Usb, [90](#)  
 WirelessHeadStageAccDataRE2HS1  
     Mcs::Usb, [90](#)  
 WirelessHeadStageAccDataRE2HS2  
     Mcs::Usb, [90](#)  
 WirelessHeadStageAccDataRE2HS3  
     Mcs::Usb, [90](#)  
 WirelessHeadStageAccDataRE2HS4  
     Mcs::Usb, [90](#)  
 WirelessHeadStageAnalogRE1HS1  
     Mcs::Usb, [90](#)  
 WirelessHeadStageAnalogRE1HS2  
     Mcs::Usb, [90](#)  
 WirelessHeadStageAnalogRE1HS3  
     Mcs::Usb, [90](#)  
 WirelessHeadStageAnalogRE1HS4  
     Mcs::Usb, [90](#)  
 WirelessHeadStageAnalogRE2HS1  
     Mcs::Usb, [90](#)  
 WirelessHeadStageAnalogRE2HS2  
     Mcs::Usb, [90](#)  
 WirelessHeadStageAnalogRE2HS3  
     Mcs::Usb, [90](#)  
 WirelessHeadStageAnalogRE2HS4  
     Mcs::Usb, [90](#)  
 WirelessHeadStageGyroDataRE1HS1  
     Mcs::Usb, [90](#)  
 WirelessHeadStageGyroDataRE1HS2  
     Mcs::Usb, [90](#)  
 WirelessHeadStageGyroDataRE1HS3  
     Mcs::Usb, [90](#)  
 WirelessHeadStageGyroDataRE1HS4



- Mcs::Usb, [90](#)
- WirelessHeadStageGyroDataRE2HS1
  - Mcs::Usb, [90](#)
- WirelessHeadStageGyroDataRE2HS2
  - Mcs::Usb, [90](#)
- WirelessHeadStageGyroDataRE2HS3
  - Mcs::Usb, [90](#)
- WirelessHeadStageGyroDataRE2HS4
  - Mcs::Usb, [90](#)
- WirelessHeadStageOptoStimCurrentRE1HS1
  - Mcs::Usb, [90](#)
- WirelessHeadStageOptoStimCurrentRE1HS2
  - Mcs::Usb, [90](#)
- WirelessHeadStageOptoStimCurrentRE1HS3
  - Mcs::Usb, [90](#)
- WirelessHeadStageOptoStimCurrentRE1HS4
  - Mcs::Usb, [90](#)
- WirelessHeadStageOptoStimCurrentRE2HS1
  - Mcs::Usb, [90](#)
- WirelessHeadStageOptoStimCurrentRE2HS2
  - Mcs::Usb, [90](#)
- WirelessHeadStageOptoStimCurrentRE2HS3
  - Mcs::Usb, [90](#)
- WirelessHeadStageOptoStimCurrentRE2HS4
  - Mcs::Usb, [91](#)
- WirelessHeadStageReservedARE1HS1
  - Mcs::Usb, [90](#)
- WirelessHeadStageReservedARE1HS2
  - Mcs::Usb, [90](#)
- WirelessHeadStageReservedARE1HS3
  - Mcs::Usb, [90](#)
- WirelessHeadStageReservedARE1HS4
  - Mcs::Usb, [90](#)
- WirelessHeadStageReservedARE2HS1
  - Mcs::Usb, [90](#)
- WirelessHeadStageReservedARE2HS2
  - Mcs::Usb, [90](#)
- WirelessHeadStageReservedARE2HS3
  - Mcs::Usb, [91](#)
- WirelessHeadStageReservedARE2HS4
  - Mcs::Usb, [91](#)
- WirelessHeadStageReservedBRE1HS1
  - Mcs::Usb, [91](#)
- WirelessHeadStageReservedBRE1HS2
  - Mcs::Usb, [91](#)
- WirelessHeadStageReservedBRE1HS3
  - Mcs::Usb, [91](#)
- WirelessHeadStageReservedBRE1HS4
  - Mcs::Usb, [91](#)
- WirelessHeadStageReservedBRE2HS1
  - Mcs::Usb, [91](#)
- WirelessHeadStageReservedBRE2HS2
  - Mcs::Usb, [91](#)
- WirelessHeadStageReservedBRE2HS3
  - Mcs::Usb, [91](#)
- WirelessHeadStageReservedBRE2HS4
  - Mcs::Usb, [91](#)
- WirelessHeadStageReservedCRE1HS1
  - Mcs::Usb, [91](#)
- WirelessHeadStageReservedCRE1HS2
  - Mcs::Usb, [91](#)
- WirelessHeadStageReservedCRE1HS3
  - Mcs::Usb, [91](#)
- WirelessHeadStageReservedCRE1HS4
  - Mcs::Usb, [91](#)
- WirelessHeadStageReservedCRE2HS1
  - Mcs::Usb, [91](#)
- WirelessHeadStageReservedCRE2HS2
  - Mcs::Usb, [91](#)
- WirelessHeadStageReservedCRE2HS3
  - Mcs::Usb, [91](#)
- WirelessHeadStageReservedCRE2HS4
  - Mcs::Usb, [91](#)
- WirelessHeadStageStatusRE1HS1
  - Mcs::Usb, [90](#)
- WirelessHeadStageStatusRE1HS2
  - Mcs::Usb, [90](#)
- WirelessHeadStageStatusRE1HS3
  - Mcs::Usb, [90](#)
- WirelessHeadStageStatusRE1HS4
  - Mcs::Usb, [90](#)
- WirelessHeadStageStatusRE2HS1
  - Mcs::Usb, [90](#)
- WirelessHeadStageStatusRE2HS2
  - Mcs::Usb, [90](#)
- WirelessHeadStageStatusRE2HS3
  - Mcs::Usb, [90](#)
- WirelessHeadStageStatusRE2HS4
  - Mcs::Usb, [90](#)
- WirelessTestAdapter
  - Mcs::Usb, [51](#)
- wLength
  - usbSetupPacket\_t, [706](#)
- Work
  - Mcs::Usb, [92](#)
- WPA16
  - Mcs::Usb, [77](#)
- WPA32
  - Mcs::Usb, [77](#)
- WPA4
  - Mcs::Usb, [77](#)
- WPA8
  - Mcs::Usb, [77](#)
- WPAError\_ScanningIsPending
  - CMcsUsbNet, [330](#)
- Write
  - CExternDTesterDeviceNet, [130](#)
- Write2
  - CExternDTesterDeviceNet, [131](#)
- WriteEepromRegisterPreconfig
  - CMcsUsbNet, [325](#)
- WritePipe
  - CGenericDevelopDeviceNet, [164](#)
- WriteRegister
  - CMcsUsbNet, [325](#)
- WriteRegister32

- CMcsUsbNet, [325](#)
- WriteRegisterArray
  - CMcsUsbNet, [325](#)
- WriteRegisterTimeSlot
  - CMcsUsbNet, [326](#)
- WriteRegisterValue
  - CMcsUsbNet, [326](#)
- WriteUARTData
  - CLIH3DeviceNet, [199](#)
- wValue
  - usbSetupPacket\_t, [706](#)
- WvcDisplayModeEnumNet
  - Mcs::Usb, [92](#)
- WvcValveModeEnumNet
  - Mcs::Usb, [92](#)
- Zero
  - Mcs::Usb, [63](#), [71](#), [81](#), [84](#), [91](#)