multichannel systems

McsUsbNet.dll

Version 5.1.32

Multi Channel Systems MCS GmbH
Aspenhaustrasse 21
72770 Reutlingen
Germany
Fon +49-71 21-90 92 5 - 0
Fax +49-71 21-90 92 5 -11
info@multichannelsystems.com
www.multichannelsystems.com

Generated by Doxygen 1.9.1

1 McsUsbNet.dll for MCS USB devices 1
1.1 Introduction . . . 1
1.2 System requirements . . . 2
1.3 Connecting to an MCS device . . . 2

2 Device Classes 2
2.1 The MCS FluidControl Device . . . 2
2.1.1 Introduction . . . 2
2.1.2 Access to the FluidControl device . . . 3
2.2 MCS-USB-Sw2to64 device . . . 3

3 Function Classes 4

4 Data ACQuisition (DACQ) Devices 5

5 The MCS Robo Device 5
5.1 Introduction . . . 5

6 STG200x & STG400x STimulus Generator 6
6.1 Introduction . . . 6
6.2 Download mode . . . 6
6.2.1 Memory Layout and Trigger Setup . . . 6
6.3 Streaming mode . . . 8
6.3.1 Memory Layout and Trigger Setup . . . 9

7 Namespace Index 11
7.1 Namespace List . . . 11

8 Hierarchical Index 11
8.1 Class Hierarchy . . . 11

9 Class Index 17
9.1 Class List . . . 17

10 Namespace Documentation 23
10.1 Mcs Namespace Reference . . . 23
10.2 Mcs::Usb Namespace Reference . . . 23
10.2.1 Enumeration Type Documentation . . . 52
10.2.2 Function Documentation . . . 95

11 Class Documentation 97
11.1 CW2100_FunctionNet::AudioChannelsNet Struct Reference . . . 97
11.1.1 Member Data Documentation . . . 97
11.2 BatteryState Class Reference . . . 98
11.2.1 Property Documentation . . . 98
11.3 Bessel3dBFilterBandPassNet Class Reference . . . 98

11.3.1 Constructor & Destructor Documentation . . . 99
11.4 Bessel3dBFilterHighPassNet Class Reference . . . 99
11.4.1 Constructor & Destructor Documentation . . . 99
11.5 Bessel3dBFilterLowPassNet Class Reference . . . 99
11.5.1 Constructor & Destructor Documentation . . . 100
11.6 BesselConstDelayFilterBandPassNet Class Reference . . . 100
11.6.1 Constructor & Destructor Documentation . . . 100
11.7 BesselConstDelayFilterHighPassNet Class Reference . . . 101
11.7.1 Constructor & Destructor Documentation . . . 101
11.8 BesselConstDelayFilterLowPassNet Class Reference . . . 101
11.8.1 Constructor & Destructor Documentation . . . 101
11.9 BesselLegacyFilterHighPassNet Class Reference . . . 102
11.9.1 Constructor & Destructor Documentation . . . 102
11.10 BesselLegacyFilterLowPassNet Class Reference . . . 102
11.10.1 Constructor & Destructor Documentation . . . 103
11.11 ButterworthFilterBandPassNet Class Reference . . . 103
11.11.1 Constructor & Destructor Documentation . . . 103
11.12 ButterworthFilterHighPassNet Class Reference . . . 103
11.12.1 Constructor & Destructor Documentation . . . 104
11.13 ButterworthFilterLowPassNet Class Reference . . . 104
11.13.1 Constructor & Destructor Documentation . . . 104
11.14 CChannelTestDeviceNet Class Reference . . . 105
11.14.1 Constructor & Destructor Documentation . . . 105
11.14.2 Member Function Documentation . . . 105
11.15 CCMOSMea_FunctionNet Class Reference . . . 106
11.15.1 Constructor & Destructor Documentation . . . 108
11.15.2 Member Function Documentation . . . 108
11.16 CCMOSMeaDeviceNet Class Reference . . . 117
11.16.1 Constructor & Destructor Documentation . . . 118
11.16.2 Member Function Documentation . . . 118
11.16.3 Property Documentation . . . 120
11.17 CCreateFilterNet Class Reference . . . 120
11.17.1 Constructor & Destructor Documentation . . . 121
11.17.2 Member Function Documentation . . . 122
11.17.3 Property Documentation . . . 122
11.18 CDacCalibrationFunctionNet Class Reference . . . 123
11.18.1 Detailed Description . . . 123
11.18.2 Constructor & Destructor Documentation . . . 123
11.18.3 Member Function Documentation . . . 124
11.19 CDacqGroupChannelGenericSelectionNet Class Reference . . . 125
11.19.1 Constructor & Destructor Documentation . . . 125
11.20 CDacqGroupChannelSelectionNet Class Reference . . . 125

11.20.1 Constructor & Destructor Documentation . . . 126
11.21 CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet > Class Template Reference 126
11.21.1 Constructor & Destructor Documentation . . . 126
11.21.2 Member Function Documentation . . . 127
11.22 CDeviceGroupChannelInfoGenericNet Class Reference . . . 129
11.22.1 Constructor & Destructor Documentation . . . 129
11.23 CDeviceGroupChannelInfoMEA2100_256Net Class Reference . . . 129
11.23.1 Constructor & Destructor Documentation . . . 129
11.24 CDeviceGroupChannelInfoNet Class Reference . . . 130
11.24.1 Constructor & Destructor Documentation . . . 130
11.25 CDeviceGroupChannelInfoSCUNet Class Reference . . . 130
11.25.1 Constructor & Destructor Documentation . . . 131
11.26 CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumTemplateNet > Class Template Reference . . . 131
11.26.1 Constructor & Destructor Documentation . . . 131
11.26.2 Member Data Documentation . . . 131
11.27 CDeviceGroupChannelInfoW2100Net Class Reference . . . 132
11.27.1 Constructor & Destructor Documentation . . . 132
11.28 CDigOutStimulatorFunctionNet Class Reference . . . 132
11.28.1 Detailed Description . . . 133
11.28.2 Constructor & Destructor Documentation . . . 133
11.28.3 Member Function Documentation . . . 134
11.29 CEncapsulatorDeviceNet Class Reference . . . 137
11.29.1 Detailed Description . . . 137
11.29.2 Constructor & Destructor Documentation . . . 137
11.29.3 Member Function Documentation . . . 137
11.30 CExternDTesterDeviceNet Class Reference . . . 137
11.30.1 Detailed Description . . . 138
11.30.2 Constructor & Destructor Documentation . . . 138
11.30.3 Member Function Documentation . . . 138
11.31 CFilterCoefficientsNet Class Reference . . . 139
11.31.1 Constructor & Destructor Documentation . . . 140
11.31.2 Member Function Documentation . . . 141
11.31.3 Property Documentation . . . 141
11.32 CFilterConfigurationNet Class Reference . . . 141
11.32.1 Constructor & Destructor Documentation . . . 142
11.32.2 Member Function Documentation . . . 142
11.33 CFilterConfigurationRegisterNet Class Reference . . . 144
11.33.1 Constructor & Destructor Documentation . . . 144
11.33.2 Member Function Documentation . . . 144
11.34 CFilterPropertyNet Class Reference . . . 145
11.34.1 Constructor & Destructor Documentation . . . 146

11.34.2 Member Function Documentation . . . 146
11.34.3 Property Documentation . . . 146
11.35 CFluidControlDeviceNet Class Reference . . . 147
11.35.1 Detailed Description . . . 148
11.35.2 Constructor & Destructor Documentation . . . 148
11.35.3 Member Function Documentation . . . 149
11.35.4 Property Documentation . . . 154
11.36 CFYIDeviceNet Class Reference . . . 154
11.36.1 Detailed Description . . . 154
11.36.2 Constructor & Destructor Documentation . . . 154
11.36.3 Property Documentation . . . 155
11.37 CGenericDevelopDeviceNet Class Reference . . . 155
11.37.1 Detailed Description . . . 162
11.37.2 Constructor & Destructor Documentation . . . 162
11.37.3 Member Function Documentation . . . 162
11.38 CGilsonDeviceNet Class Reference . . . 173
11.38.1 Detailed Description . . . 173
11.38.2 Constructor & Destructor Documentation . . . 174
11.38.3 Member Function Documentation . . . 174
11.38.4 Member Data Documentation . . . 174
11.39 CGrapheneASICDeviceNet Class Reference . . . 175
11.39.1 Constructor & Destructor Documentation . . . 175
11.39.2 Member Function Documentation . . . 175
11.40 CGrapheneFunctionNet Class Reference . . . 176
11.40.1 Detailed Description . . . 178
11.40.2 Constructor & Destructor Documentation . . . 178
11.40.3 Member Function Documentation . . . 179
11.41 CHiClampDeviceNet Class Reference . . . 189
11.41.1 Detailed Description . . . 189
11.41.2 Constructor & Destructor Documentation . . . 189
11.41.3 Property Documentation . . . 189
11.42 CHLADacqNet Class Reference . . . 190
11.42.1 Constructor & Destructor Documentation . . . 190
11.43 CHLADeviceNet Class Reference . . . 190
11.43.1 Detailed Description . . . 191
11.43.2 Constructor & Destructor Documentation . . . 191
11.43.3 Property Documentation . . . 191
11.44 CMcsUsbDacqNet::CHWInfo Class Reference . . . 191
11.44.1 Detailed Description . . . 192
11.44.2 Constructor & Destructor Documentation . . . 192
11.44.3 Member Function Documentation . . . 192
11.45 CIntanMea_FunctionNet Class Reference . . . 194

11.45.1 Constructor & Destructor Documentation . . . 195
11.45.2 Member Function Documentation . . . 195
11.46 CInterfaceboard2FunctionNet Class Reference . . . 196
11.46.1 Detailed Description . . . 197
11.46.2 Constructor & Destructor Documentation . . . 197
11.46.3 Member Function Documentation . . . 197
11.47 CInterfaceboardFunctionNet Class Reference . . . 198
11.47.1 Detailed Description . . . 199
11.47.2 Constructor & Destructor Documentation . . . 199
11.47.3 Member Function Documentation . . . 199
11.48 CLIH3DeviceNet Class Reference . . . 200
11.48.1 Detailed Description . . . 202
11.48.2 Constructor & Destructor Documentation . . . 202
11.48.3 Member Function Documentation . . . 202
11.48.4 Property Documentation . . . 209
11.49 CMcsBus_AxisParametersNet Class Reference . . . 209
11.49.1 Constructor & Destructor Documentation . . . 210
11.49.2 Member Function Documentation . . . 210
11.50 CMcsBus_ExtensionNet Class Reference . . . 211
11.50.1 Constructor & Destructor Documentation . . . 211
11.50.2 Member Function Documentation . . . 212
11.51 CMcsBus_FYIExtensionNet Class Reference . . . 212
11.51.1 Constructor & Destructor Documentation . . . 212
11.51.2 Member Function Documentation . . . 213
11.52 CMcsBus_MotorControlNet Class Reference . . . 214
11.52.1 Constructor & Destructor Documentation . . . 217
11.52.2 Member Function Documentation . . . 217
11.53 CMcsBus_SensorNet Class Reference . . . 231
11.53.1 Constructor & Destructor Documentation . . . 233
11.53.2 Member Function Documentation . . . 233
11.54 CMcsBus_TempSensorNet Class Reference . . . 241
11.54.1 Constructor & Destructor Documentation . . . 242
11.54.2 Member Function Documentation . . . 242
11.55 CMcsBus_VoltageModeNet Class Reference . . . 243
11.55.1 Constructor & Destructor Documentation . . . 244
11.55.2 Member Function Documentation . . . 245
11.56 CMcsBusNet Class Reference . . . 248
11.56.1 Constructor & Destructor Documentation . . . 248
11.56.2 Member Function Documentation . . . 249
11.57 CMcsUsbDacqNet Class Reference . . . 252
11.57.1 Detailed Description . . . 258
11.57.2 Constructor & Destructor Documentation . . . 258

11.57.3 Member Function Documentation . . . 258
11.57.4 Member Data Documentation . . . 301
11.57.5 Property Documentation . . . 302
11.57.6 Event Documentation . . . 302
11.58 CMcsUsbDeviceStatePushFunctionNet Class Reference . . . 302
11.58.1 Constructor & Destructor Documentation . . . 303
11.58.2 Member Function Documentation . . . 303
11.58.3 Event Documentation . . . 303
11.59 CMcsUsbDeviceStatePushNet Class Reference . . . 303
11.59.1 Constructor & Destructor Documentation . . . 304
11.59.2 Member Function Documentation . . . 304
11.59.3 Event Documentation . . . 304
11.60 CMcsUsbFactoryNet Class Reference . . . 304
11.60.1 Constructor & Destructor Documentation . . . 306
11.60.2 Member Function Documentation . . . 306
11.60.3 Member Data Documentation . . . 312
11.61 CMcsUsbFunctionNet Class Reference . . . 313
11.61.1 Constructor & Destructor Documentation . . . 313
11.61.2 Member Function Documentation . . . 314
11.61.3 Member Data Documentation . . . 314
11.62 CMcsUsbFunctionPointerContainer Class Reference . . . 314
11.63 CMcsUsbListEntryNet Class Reference . . . 314
11.63.1 Detailed Description . . . 315
11.63.2 Constructor & Destructor Documentation . . . 315
11.63.3 Member Function Documentation . . . 316
11.63.4 Property Documentation . . . 319
11.64 CMcsUsbListNet Class Reference . . . 320
11.64.1 Detailed Description . . . 320
11.64.2 Constructor & Destructor Documentation . . . 320
11.64.3 Member Function Documentation . . . 321
11.64.4 Property Documentation . . . 322
11.64.5 Event Documentation . . . 322
11.65 CMcsUsbNet Class Reference . . . 323
11.65.1 Detailed Description . . . 327
11.65.2 Constructor & Destructor Documentation . . . 327
11.65.3 Member Function Documentation . . . 327
11.65.4 Member Data Documentation . . . 339
11.65.5 Property Documentation . . . 343
11.66 CMcsUsbPointerContainer Class Reference . . . 344
11.67 CMEA2100_256DacqGroupChannelSelectionNet Class Reference . . . 344
11.67.1 Constructor & Destructor Documentation . . . 344
11.68 CMEA2100x256FunctionNet Class Reference . . . 344

11.68.1 Detailed Description . . . 345
11.68.2 Constructor & Destructor Documentation . . . 345
11.68.3 Member Function Documentation . . . 345
11.69 CMeaAudioFunctionNet Class Reference . . . 346
11.69.1 Constructor & Destructor Documentation . . . 346
11.69.2 Member Function Documentation . . . 347
11.70 CMeaCleanDeviceNet Class Reference . . . 349
11.70.1 Detailed Description . . . 350
11.70.2 Constructor & Destructor Documentation . . . 350
11.70.3 Member Function Documentation . . . 350
11.71 CMeaCoatDeviceNet Class Reference . . . 353
11.71.1 Detailed Description . . . 354
11.71.2 Constructor & Destructor Documentation . . . 354
11.71.3 Member Function Documentation . . . 354
11.72 CMeaDeviceNet Class Reference . . . 358
11.72.1 Detailed Description . . . 359
11.72.2 Constructor & Destructor Documentation . . . 360
11.72.3 Member Function Documentation . . . 360
11.72.4 Property Documentation . . . 365
11.73 CMeaDigitalDataFunctionNet Class Reference . . . 366
11.73.1 Constructor & Destructor Documentation . . . 367
11.73.2 Member Function Documentation . . . 367
11.74 CMeaFeedbackFunctionNet Class Reference . . . 368
11.74.1 Constructor & Destructor Documentation . . . 369
11.74.2 Member Function Documentation . . . 369
11.75 CMeaImpedanceDeviceNet Class Reference . . . 373
11.75.1 Constructor & Destructor Documentation . . . 373
11.75.2 Member Function Documentation . . . 373
11.76 CMeasureTableDeviceNet Class Reference . . . 374
11.76.1 Detailed Description . . . 375
11.76.2 Constructor & Destructor Documentation . . . 375
11.76.3 Property Documentation . . . 375
11.77 CMeaSwitchDeviceNet Class Reference . . . 375
11.77.1 Detailed Description . . . 376
11.77.2 Constructor & Destructor Documentation . . . 376
11.77.3 Member Function Documentation . . . 376
11.78 CMeaUSBDeviceNet Class Reference . . . 377
11.78.1 Detailed Description . . . 377
11.78.2 Constructor & Destructor Documentation . . . 377
11.78.3 Member Function Documentation . . . 378
11.79 CMeFunctionNet Class Reference . . . 378
11.79.1 Detailed Description . . . 379

11.79.2 Constructor & Destructor Documentation . . . 379
11.79.3 Member Function Documentation . . . 379
11.80 CMultiBatteryChargerDeviceNet Class Reference . . . 379
11.80.1 Detailed Description . . . 381
11.80.2 Constructor & Destructor Documentation . . . 381
11.80.3 Member Function Documentation . . . 381
11.81 CMultiwellCallbackFunctionNet Class Reference . . . 387
11.81.1 Detailed Description . . . 387
11.81.2 Constructor & Destructor Documentation . . . 387
11.81.3 Member Function Documentation . . . 388
11.81.4 Event Documentation . . . 388
11.82 CMultiwellDeviceNet Class Reference . . . 389
11.82.1 Detailed Description . . . 390
11.82.2 Constructor & Destructor Documentation . . . 390
11.82.3 Member Function Documentation . . . 391
11.83 CMultiwellOptoStimFunctionNet Class Reference . . . 396
11.83.1 Detailed Description . . . 397
11.83.2 Constructor & Destructor Documentation . . . 397
11.83.3 Member Function Documentation . . . 397
11.84 CNF_GenDeviceNet Class Reference . . . 401
11.84.1 Constructor & Destructor Documentation . . . 401
11.84.2 Member Function Documentation . . . 402
11.85 COctoPotDeviceNet Class Reference . . . 402
11.85.1 Constructor & Destructor Documentation . . . 403
11.85.2 Member Function Documentation . . . 403
11.86 COkuvisionStimulatorDeviceNet Class Reference . . . 406
11.86.1 Constructor & Destructor Documentation . . . 407
11.86.2 Member Function Documentation . . . 407
11.87 CPatchServerDeviceNet Class Reference . . . 409
11.87.1 Detailed Description . . . 410
11.87.2 Constructor & Destructor Documentation . . . 410
11.87.3 Property Documentation . . . 410
11.88 CPathIdentDeviceNet Class Reference . . . 410
11.88.1 Constructor & Destructor Documentation . . . 411
11.88.2 Member Function Documentation . . . 411
11.89 CPedoterDeviceNet Class Reference . . . 411
11.89.1 Detailed Description . . . 412
11.89.2 Constructor & Destructor Documentation . . . 412
11.89.3 Member Function Documentation . . . 412
11.90 CPeristalticPumpDeviceNet Class Reference . . . 413
11.90.1 Detailed Description . . . 413
11.90.2 Constructor & Destructor Documentation . . . 413

11.90.3 Property Documentation . . . 414
11.91 CPgaDeviceNet Class Reference . . . 414
11.91.1 Constructor & Destructor Documentation . . . 414
11.91.2 Member Function Documentation . . . 415
11.92 CPositionIIDeviceNet Class Reference . . . 416
11.92.1 Detailed Description . . . 418
11.92.2 Constructor & Destructor Documentation . . . 418
11.92.3 Member Function Documentation . . . 418
11.92.4 Property Documentation . . . 425
11.93 CPositionImpDeviceNet Class Reference . . . 425
11.93.1 Detailed Description . . . 425
11.93.2 Constructor & Destructor Documentation . . . 426
11.93.3 Member Function Documentation . . . 426
11.94 CPPCDeviceNet Class Reference . . . 428
11.94.1 Constructor & Destructor Documentation . . . 428
11.94.2 Property Documentation . . . 429
11.95 CPPCFunctionNet Class Reference . . . 429
11.95.1 Detailed Description . . . 430
11.95.2 Constructor & Destructor Documentation . . . 430
11.95.3 Member Function Documentation . . . 431
11.96 CPPS_DeviceNet Class Reference . . . 438
11.96.1 Constructor & Destructor Documentation . . . 438
11.96.2 Property Documentation . . . 438
11.97 CPPS_FunctionNet Class Reference . . . 439
11.97.1 Constructor & Destructor Documentation . . . 439
11.97.2 Member Function Documentation . . . 440
11.98 CPPSDeviceNet Class Reference . . . 443
11.98.1 Detailed Description . . . 443
11.98.2 Constructor & Destructor Documentation . . . 444
11.99 CProgramPressureCurveNet Class Reference . . . 444
11.99.1 Detailed Description . . . 444
11.99.2 Constructor & Destructor Documentation . . . 444
11.99.3 Member Function Documentation . . . 445
11.100 CPulseGeneratorFunctionNet Class Reference . . . 445
11.100.1 Detailed Description . . . 446
11.100.2 Constructor & Destructor Documentation . . . 446
11.100.3 Member Function Documentation . . . 447
11.101 CRadioControledDevicesNet Class Reference . . . 448
11.101.1 Constructor & Destructor Documentation . . . 449
11.101.2 Member Function Documentation . . . 449
11.102 CCMOSMeaDeviceNet::CRegionOfInterestRect Class Reference . . . 450
11.102.1 Constructor & Destructor Documentation . . . 450

11.102.2 Member Function Documentation . . . 450
11.102.3 Member Data Documentation . . . 451
11.103 CRetinaLedDeviceNet Class Reference . . . 451
11.103.1 Constructor & Destructor Documentation . . . 452
11.103.2 Member Function Documentation . . . 452
11.104 CRFFunctionNet Class Reference . . . 453
11.104.1 Detailed Description . . . 454
11.104.2 Constructor & Destructor Documentation . . . 454
11.104.3 Member Function Documentation . . . 454
11.105 CRobo_FYIProgram_FunctionNet Class Reference . . . 458
11.105.1 Constructor & Destructor Documentation . . . 458
11.105.2 Member Function Documentation . . . 458
11.106 CRobo_FYITemp_FunctionNet Class Reference . . . 459
11.106.1 Constructor & Destructor Documentation . . . 460
11.106.2 Member Function Documentation . . . 460
11.107 CRoboDacqNet Class Reference . . . 461
11.107.1 Constructor & Destructor Documentation . . . 464
11.107.2 Member Function Documentation . . . 464
11.107.3 Member Data Documentation . . . 475
11.108 CRoboDeviceNet Class Reference . . . 476
11.108.1 Detailed Description . . . 479
11.108.2 Constructor & Destructor Documentation . . . 479
11.108.3 Member Function Documentation . . . 479
11.108.4 Member Data Documentation . . . 487
11.108.5 Property Documentation . . . 491
11.108.6 Event Documentation . . . 492
11.109 CRoboFluidDeviceNet Class Reference . . . 492
11.109.1 Constructor & Destructor Documentation . . . 493
11.109.2 Member Function Documentation . . . 493
11.109.3 Member Data Documentation . . . 495
11.109.4 Property Documentation . . . 495
11.110 CRoboInjectDeviceNet Class Reference . . . 495
11.110.1 Detailed Description . . . 496
11.110.2 Constructor & Destructor Documentation . . . 496
11.111 CRoboocyte2DeviceNet Class Reference . . . 496
11.111.1 Detailed Description . . . 496
11.111.2 Constructor & Destructor Documentation . . . 497
11.111.3 Member Function Documentation . . . 497
11.112 CRoboStatorDeviceNet Class Reference . . . 497
11.112.1 Constructor & Destructor Documentation . . . 499
11.112.2 Member Function Documentation . . . 499
11.112.3 Property Documentation . . . 503

11.113 CSafeISDeviceNet Class Reference . . . 503
11.113.1 Detailed Description . . . 504
11.113.2 Constructor & Destructor Documentation . . . 504
11.113.3 Member Function Documentation . . . 505
11.113.4 Property Documentation . . . 506
11.114 CSCUDacqGroupChannelSelectionNet Class Reference . . . 506
11.114.1 Constructor & Destructor Documentation . . . 507
11.115 CSCUFunctionNet Class Reference . . . 507
11.115.1 Detailed Description . . . 509
11.115.2 Constructor & Destructor Documentation . . . 510
11.115.3 Member Function Documentation . . . 510
11.115.4 Event Documentation . . . 524
11.116 CSerialPortNet Class Reference . . . 524
11.116.1 Constructor & Destructor Documentation . . . 524
11.116.2 Member Function Documentation . . . 525
11.117 CStg200xBasicNet Class Reference . . . 525
11.117.1 Detailed Description . . . 530
11.117.2 Constructor & Destructor Documentation . . . 530
11.117.3 Member Function Documentation . . . 530
11.118 CStg200xDownloadBasicNet Class Reference . . . 564
11.118.1 Detailed Description . . . 566
11.118.2 Member Function Documentation . . . 566
11.118.3 Property Documentation . . . 573
11.119 CStg200xDownloadNet Class Reference . . . 573
11.119.1 Detailed Description . . . 574
11.119.2 Constructor & Destructor Documentation . . . 574
11.119.3 Member Function Documentation . . . 574
11.119.4 Event Documentation . . . 579
11.120 CStimulusFunctionNet Class Reference . . . 580
11.120.1 Constructor & Destructor Documentation . . . 581
11.120.2 Member Function Documentation . . . 582
11.120.3 Event Documentation . . . 591
11.121 CSw2to64DeviceNet Class Reference . . . 591
11.121.1 Detailed Description . . . 591
11.121.2 Constructor & Destructor Documentation . . . 592
11.121.3 Member Function Documentation . . . 592
11.122 CTcxDeviceNet Class Reference . . . 593
11.122.1 Detailed Description . . . 595
11.122.2 Constructor & Destructor Documentation . . . 595
11.122.3 Member Function Documentation . . . 596
11.123 CTEERFunctionNet Class Reference . . . 606
11.123.1 Detailed Description . . . 608

11.123.2 Constructor & Destructor Documentation . . . 608
11.123.3 Member Function Documentation . . . 609
11.124 CTEERMachineDeviceNet Class Reference . . . 617
11.124.1 Constructor & Destructor Documentation . . . 617
11.124.2 Property Documentation . . . 618
11.125 CUsbDeviceConfigurationFunctionNet Class Reference . . . 618
11.125.1 Detailed Description . . . 618
11.125.2 Constructor & Destructor Documentation . . . 618
11.125.3 Member Function Documentation . . . 619
11.126 CUsbExceptionNet Class Reference . . . 619
11.126.1 Detailed Description . . . 620
11.126.2 Constructor & Destructor Documentation . . . 620
11.126.3 Property Documentation . . . 620
11.127 CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet Class Reference . . . 620
11.127.1 Constructor & Destructor Documentation . . . 621
11.127.2 Member Data Documentation . . . 621
11.128 CW2100_FunctionNet Class Reference . . . 621
11.128.1 Constructor & Destructor Documentation . . . 623
11.128.2 Member Function Documentation . . . 623
11.128.3 Property Documentation . . . 628
11.129 CW2100_StimulatorFunctionNet Class Reference . . . 629
11.129.1 Constructor & Destructor Documentation . . . 630
11.129.2 Member Function Documentation . . . 630
11.129.3 Member Data Documentation . . . 635
11.129.4 Event Documentation . . . 636
11.130 CW2100DacqGroupChannelSelectionNet Class Reference . . . 636
11.130.1 Constructor & Destructor Documentation . . . 636
11.131 CWarnerUssingDeviceNet Class Reference . . . 636
11.131.1 Detailed Description . . . 637
11.131.2 Constructor & Destructor Documentation . . . 637
11.131.3 Property Documentation . . . 637
11.132 CWarnerUssingFunctionNet Class Reference . . . 638
11.132.1 Detailed Description . . . 640
11.132.2 Constructor & Destructor Documentation . . . 640
11.132.3 Member Function Documentation . . . 640
11.133 CWarnerValveControllerDeviceNet Class Reference . . . 653
11.133.1 Detailed Description . . . 657
11.133.2 Constructor & Destructor Documentation . . . 658
11.133.3 Member Function Documentation . . . 658
11.133.4 Event Documentation . . . 675
11.134 CWarnerValveControllerDeviceTesterFunctionNet Class Reference . . . 678
11.134.1 Detailed Description . . . 679

11.134.2 Constructor & Destructor Documentation . . . 679
11.134.3 Member Function Documentation . . . 680
11.135 CWClassicFunctionNet Class Reference . . . 681
11.135.1 Constructor & Destructor Documentation . . . 682
11.135.2 Member Function Documentation . . . 683
11.136 CWirelessBaseFunctionNet Class Reference . . . 686
11.136.1 Constructor & Destructor Documentation . . . 687
11.136.2 Member Function Documentation . . . 687
11.137 DeviceIdNet Struct Reference . . . 687
11.137.1 Detailed Description . . . 687
11.137.2 Constructor & Destructor Documentation . . . 688
11.137.3 Member Function Documentation . . . 688
11.137.4 Member Data Documentation . . . 688
11.138 DigitalSource< digitalsourceenum > Class Template Reference . . . 689
11.138.1 Constructor & Destructor Documentation . . . 689
11.138.2 Member Function Documentation . . . 689
11.138.3 Property Documentation . . . 690
11.139 DigitalSourceGeneral Class Reference . . . 690
11.139.1 Constructor & Destructor Documentation . . . 690
11.139.2 Member Function Documentation . . . 690
11.139.3 Property Documentation . . . 691
11.140 DriverVersionNet Class Reference . . . 691
11.140.1 Detailed Description . . . 692
11.140.2 Constructor & Destructor Documentation . . . 692
11.140.3 Member Function Documentation . . . 692
11.141 FirmwareDestinationNames Class Reference . . . 697
11.141.1 Member Data Documentation . . . 698
11.142 HeadStageIDType Class Reference . . . 700
11.142.1 Member Enumeration Documentation . . . 701
11.142.2 Constructor & Destructor Documentation . . . 701
11.142.3 Member Function Documentation . . . 702
11.142.4 Property Documentation . . . 702
11.143 HeadstageIDTypeObject Class Reference . . . 703
11.143.1 Constructor & Destructor Documentation . . . 704
11.143.2 Member Function Documentation . . . 704
11.143.3 Member Data Documentation . . . 704
11.143.4 Property Documentation . . . 705
11.144 HeadStageIDTypeState Class Reference . . . 705
11.144.1 Property Documentation . . . 705
11.145 mkfilterNet Class Reference . . . 706
11.145.1 Member Function Documentation . . . 706
11.146 CRoboDeviceNet::RoboMainLowLevelCommands Class Reference . . . 709

11.146.1 Member Function Documentation . . . 710
11.147 CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands Class Reference . . . 716
11.147.1 Member Function Documentation . . . 716
11.148 CFilterCoefficientsNet::s_FilterAttributesNet Struct Reference . . . 716
11.148.1 Constructor & Destructor Documentation . . . 716
11.148.2 Member Function Documentation . . . 716
11.148.3 Member Data Documentation . . . 717
11.149 CMeaAudioFunctionNet::s_setaudionet Struct Reference . . . 717
11.149.1 Member Data Documentation . . . 717
11.150 CStimulusFunctionNet::SidebandData Class Reference . . . 718
11.150.1 Constructor & Destructor Documentation . . . 718
11.150.2 Property Documentation . . . 718
11.151 StgStatusNet Class Reference . . . 719
11.151.1 Member Function Documentation . . . 719
11.151.2 Member Data Documentation . . . 719
11.152 CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData Class Reference . . . 719
11.152.1 Constructor & Destructor Documentation . . . 720
11.152.2 Property Documentation . . . 720
11.153 usbSetupPacket_t Class Reference . . . 721
11.153.1 Member Data Documentation . . . 721
11.154 W2100_StimulusParametersNet Struct Reference . . . 721
11.154.1 Member Data Documentation . . . 722

Index **723**

1 McsUsbNet.dll for MCS USB devices

1.1 Introduction

This DLL provides the .NET interface to MCS devices

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

- STG200x & STG400x STimulus Generator
- Data ACQuisition (DACQ) Devices

See here for a list of our other devices: Device Classes.

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

1.2 System requirements

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

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

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

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

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

All examples assume that the Mcs.Usb namespace is loaded:
```
using namespace Mcs.Usb;
```

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

1.3 Connecting to an MCS device

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

When more than one DAQ device of the specific type is connected, you can use the Mcs.Usb.CMcsUsbListNet class to get a list of available devices:
```
CMcsUsbListNet usblist = new CMcsUsbListNet(DeviceEnumNet.MCS_DEVICE_USB);
var entry = usblist.GetUsbListEntry((uint)0);
CMcsUsbNet device = new CMcsUsbNet();
device.Connect(entry);
```

After you are finished with the device, you can disconnect the device object from the device by:
```
device.Disonnect();
```

2 Device Classes

- For FluidControl device see MCS FluidControl
- For SW2TO64 device see MCS-USB-Sw2to64
- For TCx device see Mcs.Usb.CTcxDeviceNet

2.1 The MCS FluidControl Device

2.1.1 Introduction

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

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

The device has 8 ADC inputs with a rage 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().GetNumberOfHWADCChannels(out hwchannels);
device.SetNumberOfChannels(hwchannels);
int samplingrate = 1000;
device.SetSamplerate(samplingrate, 1, 0);
device.EnableDigitalIn(true, 0);
```

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

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

For the Mcs.Usb.OnChannelData callback function you have to provide a definition of the channels you want to receive.

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

The Mcs.Usb.OnChannelData callback function gets a callback for each channelblock that is defined. In this example a callback for each channel.

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

see MEA_Recording in the Examples directory.

5 The MCS Robo Device

5.1 Introduction

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

- The MCS Roboinject device is controlled by the Mcs.Usb.CRoboInjectDeviceNet class.
- The MCS Roboocyte2 device is controlled by the Mcs.Usb.CRoboocyte2DeviceNet class.

Both classes are derived from Mcs.Usb.CRoboDeviceNet

6 STG200x & STG400x STimulus Generator

6.1 Introduction

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

6.2 Download mode

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

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

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

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

Mcs.Usb.CStg200xDownloadNet is the class for using the STG in download mode.

6.2.1 Memory Layout and Trigger Setup

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

For connecting to an STG see Connecting to an MCS device.

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

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

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

```
uint32_t nchannels = device.GetNumberOfAnalogChannels();
uint32_t nsync = device.GetNumberOfSyncoutChannels();
```

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

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

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

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

```
// Only meaningfull 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 transfered 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 funtion 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 funtion, 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.CStg200xStreaming↩ Net.GetSyncoutQueueSpace.

The values for the analog outputs are 16 bits signed integers. The lower bits are trunctated 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.EnableContinousMode 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.DisableContinousMode is used, the STG stops when it runs out of data. It has to be retriggered to resume the output.

```
device.EnableContinousMode();
```

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:

```
latency = (ringbuffersize in bytes/4) / 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 erlier.

The following example divides the total memory equally amoung 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
uuint[] 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.CStg200xStreaming↩ Net.SetupTrigger call. Its arguments are a list of channelmaps, syncoutmaps, digoutmap, autostart and callback↩ _threshold, with one entry for each of the available triggers. channelmap is a bitmap, each bit representing one of the available channels. To assign channel 1 and 3 and syncout 1 to trigger 1 use:

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

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

Start Trigger by pushing the front button or by Software

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

see the StgStreamingExampleNet in the example directory.

7 Namespace Index

7.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Mcs 23

Mcs::Usb 23

8 Hierarchical Index

8.1 Class Hierarchy

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

CW2100_FunctionNet::AudioChannelsNet 97

BatteryState 98

CCreateFilterNet 120

- **Bessel3dBFilterBandPassNet** 98
- **Bessel3dBFilterHighPassNet** 99
- **Bessel3dBFilterLowPassNet** 99
- **BesselConstDelayFilterBandPassNet** 100
- **BesselConstDelayFilterHighPassNet** 101
- **BesselConstDelayFilterLowPassNet** 101
- **BesselLegacyFilterHighPassNet** 102
- **BesselLegacyFilterLowPassNet** 102
- **ButterworthFilterBandPassNet** 103

ButterworthFilterHighPassNet 103

ButterworthFilterLowPassNet 104

CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumTemplateNet > 131

CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumNet > 131

CDeviceGroupChannelInfoNet 130

CDeviceGroupChannelInfoTemplateNet< int > 131

CDeviceGroupChannelInfoGenericNet 129

CDeviceGroupChannelInfoTemplateNet< MEA2100_256DacqGroupChannelEnumNet > 131

CDeviceGroupChannelInfoMEA2100_256Net 129

CDeviceGroupChannelInfoTemplateNet< SCUDacqGroupChannelEnumNet > 131

CDeviceGroupChannelInfoSCUNet 130

CDeviceGroupChannelInfoTemplateNet< W2100DacqGroupChannelEnumNet > 131

CDeviceGroupChannelInfoW2100Net 132

CFilterCoefficientsNet 139

CFilterPropertyNet 145

CMcsUsbDacqNet::CHWInfo 191

CMcsUsbFunctionNet 313

CDacqGroupChannelSelectionTemplateNet< W2100DacqGroupChannelEnumNet, W2100↩ DacqGroupChannelEnum, CDeviceGroupChannelInfoW2100Net > 126

CW2100DacqGroupChannelSelectionNet 636

CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumNet, DacqGroup↩ ChannelEnum, CDeviceGroupChannelInfoNet > 126

CDacqGroupChannelSelectionNet 125

CDacqGroupChannelSelectionTemplateNet< int, int, CDeviceGroupChannelInfoGenericNet > 126

CDacqGroupChannelGenericSelectionNet 125

CDacqGroupChannelSelectionTemplateNet< SCUDacqGroupChannelEnumNet, SCUDacq↩ GroupChannelEnum, CDeviceGroupChannelInfoSCUNet > 126

CSCUDacqGroupChannelSelectionNet 506

CDacqGroupChannelSelectionTemplateNet< MEA2100_256DacqGroupChannelEnumNet, MEA2100_256DacqGroupChannelEnum, CDeviceGroupChannelInfoMEA2100_256Net > 126

CMEA2100_256DacqGroupChannelSelectionNet 344

CCMOSMea_FunctionNet 106

CDacCalibrationFunctionNet 123

CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet > 126

CDigOutStimulatorFunctionNet 132

CFilterConfigurationNet 141

CFilterConfigurationRegisterNet 144

CGrapheneFunctionNet 176

CIntanMea_FunctionNet 194

CInterfaceboardFunctionNet 198

CInterfaceboard2FunctionNet 196

CMEA2100x256FunctionNet 344

CMcsBusNet 248

CMcsBus_AxisParametersNet 209

CMcsBus_ExtensionNet 211

CMcsBus_FYIExtensionNet 212

CMcsBus_MotorControlNet 214

CMcsBus_SensorNet 231

CMcsBus_TempSensorNet 241

CMcsBus_VoltageModeNet 243

CMcsUsbDeviceStatePushFunctionNet 302

CMultiwellCallbackFunctionNet 387

CSCUFunctionNet 507

CMeFunctionNet 378

CMeaAudioFunctionNet 346

CMeaDigitalDataFunctionNet 366

CMeaFeedbackFunctionNet 368

CMultiwellOptoStimFunctionNet 396

CPPCFunctionNet 429

CPPS_FunctionNet 439

CProgramPressureCurveNet 444

CPulseGeneratorFunctionNet 445

CRFFunctionNet 453

CRobo_FYIProgram_FunctionNet 458

CRobo_FYITemp_FunctionNet 459
CStimulusFunctionNet 580
CTEERFunctionNet 606
CUsbDeviceConfigurationFunctionNet 618
CW2100_StimulatorFunctionNet 629
CWarnerUssingFunctionNet 638
CWarnerValveControllerDeviceTesterFunctionNet 678
CWirelessBaseFunctionNet 686
CW2100_FunctionNet 621
CWClassicFunctionNet 681
CMcsUsbFunctionPointerContainer 314
CMcsUsbListEntryNet 314
CMcsUsbListNet 320
CMcsUsbNet 323
CExternDTesterDeviceNet 137
CFluidControlDeviceNet 147
CGenericDevelopDeviceNet 155
CGilsonDeviceNet 173
CMcsUsbDacqNet 252
CMeaDeviceNet 358
CMeaUSBDeviceNet 377
CCMOSMeaDeviceNet 117
CGrapheneASICDeviceNet 175
CHLADacqNet 190
CLIH3DeviceNet 200
CMultiwellDeviceNet 389
CWarnerUssingDeviceNet 636
COctoPotDeviceNet 402
CRoboDacqNet 461
CMcsUsbDeviceStatePushNet 303
CWarnerValveControllerDeviceNet 653
CMcsUsbFactoryNet 304

CMeaCleanDeviceNet 349

CMeaCoatDeviceNet 353

CMeaImpedanceDeviceNet 373

CMeaSwitchDeviceNet 375

CChannelTestDeviceNet 105

CMultiBatteryChargerDeviceNet 379

CNF_GenDeviceNet 401

COkuvisionStimulatorDeviceNet 406

CPPCDeviceNet 428

CPPS_DeviceNet 438

CPathIdentDeviceNet 410

CPedoterDeviceNet 411

CPeristalticPumpDeviceNet 413

CPgaDeviceNet 414

CPositionIIDeviceNet 416

CPositionImpDeviceNet 425

CRadioControledDevicesNet 448

CRetinaLedDeviceNet 451

CRoboDeviceNet 476

CEncapsulatorDeviceNet 137

CFYIDeviceNet 154

CHLADeviceNet 190

CHiClampDeviceNet 189

CMeasureTableDeviceNet 374

CPPSDeviceNet 443

CPatchServerDeviceNet 409

CRoboStatorDeviceNet 497

CRoboInjectDeviceNet 495

CRoboocyte2DeviceNet 496

CTEERMachineDeviceNet 617

CRoboFluidDeviceNet 492

CSafeISDeviceNet 503

CSerialPortNet 524

CStg200xBasicNet 525

CStg200xDownloadBasicNet 564

CStg200xDownloadNet 573

CSw2to64DeviceNet 591

CTcxDeviceNet 593

CMcsUsbPointerContainer 344

CCMOSMeaDeviceNet::CRegionOfInterestRect 450

CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet 620

DeviceIdNet 687

DigitalSource< digitalsourceenum > 689

DigitalSourceGeneral 690

DriverVersionNet 691
Exception

CUsbExceptionNet 619

FirmwareDestinationNames 697

HeadstageIDTypeObject 703

HeadStageIDTypeState 705
IComparable

HeadStageIDType 700

mkfilterNet 706

CRoboDeviceNet::RoboMainLowLevelCommands 709

CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands 716

CFilterCoefficientsNet::s_FilterAttributesNet 716

CMeaAudioFunctionNet::s_setaudionet 717

CStimulusFunctionNet::SidebandData 718

StgStatusNet 719
stgstreaming

CStg200xBasicNet 525

CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData 719

usbSetupPacket_t 721

W2100_StimulusParametersNet 721

9 Class Index

9.1 Class List

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

CW2100_FunctionNet::AudioChannelsNet 97

BatteryState 98

Bessel3dBFilterBandPassNet 98

Bessel3dBFilterHighPassNet 99

Bessel3dBFilterLowPassNet 99

BesselConstDelayFilterBandPassNet 100

BesselConstDelayFilterHighPassNet 101

BesselConstDelayFilterLowPassNet 101

BesselLegacyFilterHighPassNet 102

BesselLegacyFilterLowPassNet 102

ButterworthFilterBandPassNet 103

ButterworthFilterHighPassNet 103

ButterworthFilterLowPassNet 104

CChannelTestDeviceNet 105

CCMOSMea_FunctionNet 106

CCMOSMeaDeviceNet 117

CCreateFilterNet 120

CDacCalibrationFunctionNet 123

CDacqGroupChannelGenericSelectionNet 125

CDacqGroupChannelSelectionNet 125

CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, 126

CDeviceGroupChannelInfoGenericNet 129

CDeviceGroupChannelInfoMEA2100_256Net 129

CDeviceGroupChannelInfoNet 130

CDeviceGroupChannelInfoSCUNet 130

CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumTemplateNet > 131

CDeviceGroupChannelInfoW2100Net 132

CDigOutStimulatorFunctionNet
CDigOutStimulatorFunctionNet is the class of the DigOut stimulator function class 132

CEncapsulatorDeviceNet
CEncapsulatorDeviceNet is the to control the MCS HiClamp device 137

CExternDTesterDeviceNet
CExternDTesterDeviceNet is the class to access the ExternD Tester (Handheld Device Tester D) 137

CFilterCoefficientsNet 139

CFilterConfigurationNet 141

CFilterConfigurationRegisterNet 144

CFilterPropertyNet 145

CFluidControlDeviceNet
CFluidControlDeviceNet is the class to control MCS FluidControl (FCB and FCX) device 147

CFYIDeviceNet
CFYIDeviceNet is the class to control the MCS FYI device 154

CGenericDevelopDeviceNet
CGenericDevelopDeviceNet is the class to use during development of a new device 155

CGilsonDeviceNet
CGilsonDeviceNet is the class to control a Gilson device 173

CGrapheneASICDeviceNet 175

CGrapheneFunctionNet
CGrapheneFunctionNet is the class to control Graphene device functions 176

CHiClampDeviceNet
CHiClampDeviceNet is the to control the MCS HiClamp device 189

CHLADacqNet 190

CHLADeviceNet
CHLADeviceNet is the to control the MCS HLA device 190

CMcsUsbDacqNet::CHWInfo
Class to provide hardware information about the device 191

CIntanMea_FunctionNet 194

CInterfaceboard2FunctionNet
CInterfaceboard2FunctionNet is the class to control the Interfaceboard 196

CInterfaceboardFunctionNet
CInterfaceboardFunctionNet is the class to control the Interfaceboard 198

CLIH3DeviceNet
CLIH3DeviceNet is the class to access the HEKA LIH3 device 200

CMcsBus_AxisParametersNet 209

CMcsBus_ExtensionNet 211

CMcsBus_FYIExtensionNet 212

CMcsBus_MotorControlNet 214

CMcsBus_SensorNet 231

CMcsBus_TempSensorNet 241

CMcsBus_VoltageModeNet 243

CMcsBusNet 248

CMcsUsbDacqNet
Base class for data acquisition devices 252

CMcsUsbDeviceStatePushFunctionNet 302

CMcsUsbDeviceStatePushNet 303

CMcsUsbFactoryNet 304

CMcsUsbFunctionNet 313

CMcsUsbFunctionPointerContainer 314

CMcsUsbListEntryNet
McsUsbListEntryNet identifies a connected device 314

CMcsUsbListNet
Class to handle a list of connected MCS USB devices 320

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 323

CMcsUsbPointerContainer 344

CMEA2100_256DacqGroupChannelSelectionNet 344

CMEA2100x256FunctionNet
CMEA2100x256FunctionNet is the class to control the MEA2100-256 device needs #include "Stg200xNet.h" to resolve documentation reference 344

CMeaAudioFunctionNet 346

CMeaCleanDeviceNet
CMeaCleanDeviceNet is the class to access the MEA Clean device 349

CMeaCoatDeviceNet
CMeaCoatDeviceNet is the class to access the MEA Coat device 353

CMeaDeviceNet
Base class for MEA data acquisition devices 358

CMeaDigitalDataFunctionNet 366

CMeaFeedbackFunctionNet 368

CMeaImpedanceDeviceNet 373

CMeasureTableDeviceNet
CMeasureTableDeviceNet is the to control the MCS HLA device 374

CMeaSwitchDeviceNet
The class to control the USB-MEA-Switch 375

CMeaUSBDeviceNet
Class for data acquisition via ME and MEA USB amplifiers 377

CMeFunctionNet
378

CMultiBatteryChargerDeviceNet
CMultiBatteryChargerDeviceNet is the class to access the MBC-08 device 379

CMultiwellCallbackFunctionNet
CMultiwellCallbackFunctionNet is the class to access the Multiwell-Mini-Stimulator 387

CMultiwellDeviceNet
CMultiwellDeviceNet is the class to access the Multiwell device 389

CMultiwellOptoStimFunctionNet
CMultiwellOptoStimFunctionNet is the class to access the optical properties of the Multiwell Optostim device 396

CNF_GenDeviceNet 401

COctoPotDeviceNet 402

COkuvisionStimulatorDeviceNet 406

CPatchServerDeviceNet
CPatchServerDeviceNet is the class to control the MCS PatchServer device 409

CPathIdentDeviceNet 410

CPedoterDeviceNet
411

CPeristalticPumpDeviceNet
CPeristalticPumpDeviceNet is the class to control a Persistaltic Pump 413

CPgaDeviceNet 414

CPositionIIDeviceNet
CPositionIIDeviceNet is the class to control PositionII devices 416

CPositionImpDeviceNet
CPositionImpDeviceNet is the class to access the Position/Imp devices 425

CPPCDeviceNet 428

CPPCFunctionNet
CPPCFunctionNet is the class to access the PPC (high precision Patch Peristalic patch Pump 429

CPPS_DeviceNet 438

CPPS_FunctionNet 439

CPPSDeviceNet
CPPS4plus1DeviceNet is the to control the MCS HLA device 443

CProgramPressureCurveNet
CProgramPressureCurveNet is the class to program pressure curves 444

CPulseGeneratorFunctionNet
CPulseGeneratorFunctionNet is the class to control the pulse generator for video tracking 445

CRadioControledDevicesNet 448

CCMOSMeaDeviceNet::CRegionOfInterestRect 450

CRetinaLedDeviceNet 451

CRFFunctionNet
CRFFunctionNet is the class to control RF devices 453

CRobo_FYIProgram_FunctionNet 458

CRobo_FYITemp_FunctionNet 459

CRoboDacqNet 461

CRoboDeviceNet
CRoboDeviceNet is the base class for all Robo platform based devices 476

CRoboFluidDeviceNet 492

CRoboInjectDeviceNet
CRoboInjectDeviceNet is the to control the MCS RoboInject device 495

CRoboocyte2DeviceNet
CRoboocyte2DeviceNet is the class to control the MCS Roboocyte2 device 496

CRoboStatorDeviceNet 497

CSafeISDeviceNet
503

CSCUDacqGroupChannelSelectionNet 506

CSCUFunctionNet
CSCUFunctionNet is the class to control the SCU device 507

CSerialPortNet 524

CStg200xBasicNet
Base class for the Stg200x 525

CStg200xDownloadBasicNet
CStg200xDownloadBasicNet is the base class to control the download mode of the MCS STG device 564

CStg200xDownloadNet
Main class for the STG download mode This class implements the STG download mode interface. 573

CStimulusFunctionNet 580

CSw2to64DeviceNet
The class to control the MCS-USB-Sw2to64 device 591

CTcxDeviceNet
Class to control a Temperature Controller (TCX) 593

CTEERFunctionNet
CTEERFunctionNet is the class to control the TEER device 606

CTEERMachineDeviceNet 617

CUsbDeviceConfigurationFunctionNet
CUsbDeviceConfigurationFunctionNet is the class to configure the USB firmware 618

CUsbExceptionNet
Exception class that is thrown in case of an USB error 619

CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet 620

CW2100_FunctionNet 621

CW2100_StimulatorFunctionNet 629

CW2100DacqGroupChannelSelectionNet 636

CWarnerUssingDeviceNet
CWarnerUssingDeviceNet is the class to control the Ussing device 636

CWarnerUssingFunctionNet
CWarnerUssingFunctionNet is the class to control the Ussing device 638

CWarnerValveControllerDeviceNet
CWarnerValveControllerDeviceNet is the class to access the Warner Valve Controller 653

CWarnerValveControllerDeviceTesterFunctionNet
CWarnerValveControllerDeviceTesterFunctionNet is the class to access the functions for the Warner Valve Controller Device Tester 678

CWClassicFunctionNet 681

CWirelessBaseFunctionNet 686

DeviceIdNet
Device Id 687

DigitalSource< digitalsourceenum > 689

DigitalSourceGeneral 690

DriverVersionNet
Class gives firmware versions of the device's firmware destinations 691

FirmwareDestinationNames 697

HeadStageIDType 700

HeadstageIDTypeObject 703

HeadStageIDTypeState 705

mkfilterNet 706

CRoboDeviceNet::RoboMainLowLevelCommands 709

CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands 716

CFilterCoefficientsNet::s_FilterAttributesNet 716

CMeaAudioFunctionNet::s_setaudionet 717

CStimulusFunctionNet::SidebandData 718

StgStatusNet 719

CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData 719

usbSetupPacket_t 721

W2100_StimulusParametersNet 721

10 Namespace Documentation

10.1 Mcs Namespace Reference

Namespaces

- Usb

10.2 Mcs::Usb Namespace Reference

Classes

- class CChannelTestDeviceNet
- class CCMOSMea_FunctionNet
- class CCMOSMeaDeviceNet
- class CCreateFilterNet
- class ButterworthFilterLowPassNet
- class ButterworthFilterHighPassNet
- class ButterworthFilterBandPassNet
- class BesselLegacyFilterLowPassNet
- class BesselLegacyFilterHighPassNet
- class BesselConstDelayFilterLowPassNet
- class BesselConstDelayFilterHighPassNet
- class BesselConstDelayFilterBandPassNet
- class Bessel3dBFilterLowPassNet
- class Bessel3dBFilterHighPassNet
- class Bessel3dBFilterBandPassNet
- class CDeviceGroupChannelInfoTemplateNet
- class CDeviceGroupChannelInfoGenericNet
- class CDeviceGroupChannelInfoNet
- class CDeviceGroupChannelInfoW2100Net
- class CDeviceGroupChannelInfoSCUNet
- class CDeviceGroupChannelInfoMEA2100_256Net
- class CDacqGroupChannelSelectionTemplateNet
- class CDacqGroupChannelGenericSelectionNet
- class CDacqGroupChannelSelectionNet
- class CW2100DacqGroupChannelSelectionNet
- class CSCUDacqGroupChannelSelectionNet
- class CMEA2100_256DacqGroupChannelSelectionNet
- class CDacCalibrationFunctionNet
- class CDigOutStimulatorFunctionNet

  CDigOutStimulatorFunctionNet is the class of the DigOut stimulator function class.
- class CExternDTesterDeviceNet

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

- class CGrapheneFunctionNet

  CGrapheneFunctionNet is the class to control Graphene device functions
- class CInterfaceboard2FunctionNet

  CInterfaceboard2FunctionNet is the class to control the Interfaceboard
- class CInterfaceboardFunctionNet

  CInterfaceboardFunctionNet is the class to control the Interfaceboard
- class CLIH3DeviceNet

  CLIH3DeviceNet is the class to access the HEKA LIH3 device.
- class CMEA2100x256FunctionNet

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

  CMeaCleanDeviceNet is the class to access the MEA Clean device.
- class CMeaCoatDeviceNet

  CMeaCoatDeviceNet is the class to access the MEA Coat device.
- class CMultiBatteryChargerDeviceNet

  CMultiBatteryChargerDeviceNet is the class to access the MBC-08 device.
- class CMultiwellCallbackFunctionNet

  CMultiwellCallbackFunctionNet is the class to access the Multiwell-Mini-Stimulator
- class CMultiwellDeviceNet

  CMultiwellDeviceNet is the class to access the Multiwell device.
- class CMultiwellOptoStimFunctionNet

  CMultiwellOptoStimFunctionNet is the class to access the optical properties of the Multiwell Optostim device
- class CPedoterDeviceNet
- class CPositionIIDeviceNet

  CPositionIIDeviceNet is the class to control PositionII devices
- class CPositionImpDeviceNet

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

  CPPCFunctionNet is the class to access the PPC (high precision Patch Peristalic 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 CRadioControledDevicesNet
- 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 CTEERMachineDeviceNet
- 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_MEAUSB_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_old = 0xABCD ,
  MCS = MCS_VENDOR_ID ,
  PCI = 0x10E8 ,
  Cypress = CYPRESS_VENDOR_ID ,
  ALA_VC3 = ALA_VC3_VENDOR_ID }

  Enumerates the group of MCS devices to connect to.

- enum class ProductIdEnumNet {
  Any = -1 ,
  None = 0 ,
  LegacyMeaUsb = MCS_PRODUCT_ID_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 ,
Robooycte2 = MCS_PRODUCT_ID_ROBOOCYTE2 ,
RoboInject = MCS_PRODUCT_ID_ROBOINJECT ,
HiClamp = MCS_PRODUCT_ID_HICLAMP ,
PatchServer = MCS_PRODUCT_ID_PATCHSERVER ,
Dilutor = MCS_PRODUCT_ID_DILUTOR ,
HiClamp4Uart = MCS_PRODUCT_ID_HICLAMP4UART ,
IM16S16KRA = MCS_PRODUCT_ID_IM16S16KRA ,
IM64KRB = MCS_PRODUCT_ID_IM64KRB ,
IS32KRA = MCS_PRODUCT_ID_IS32KRA ,
IM64KRC = MCS_PRODUCT_ID_IM64KRC ,
IM16S8KRA = MCS_PRODUCT_ID_IM16S8KRA ,
IM16KRC = MCS_PRODUCT_ID_IM16KRC ,
SmartImplant = MCS_PRODUCT_ID_SMARTIMPLANT ,
PositionImp = MCS_PRODUCT_ID_POSITION_IMP ,
PositionBase = MCS_PRODUCT_ID_POSITION_BASE ,
PositionIICentralUnit = MCS_PRODUCT_ID_POSITIONII_CENTRAL_UNIT ,
PositionIIBase = MCS_PRODUCT_ID_POSITIONII_BASE ,
GrapheneProjectTestDevice = MCS_PRODUCT_ID_GRAPHENE_PROJECT_TEST_DEVICE ,
Pos900 = MCS_PRODUCT_ID_POS900 ,
Neptun = MCS_PRODUCT_ID_NEPTUN ,
Warner_Valve_Control = MCS_PRODUCT_ID_WARNER_VALVE_CONTROL ,
Warner_TEER_Machine = MCS_PRODUCT_ID_WARNER_TEER_MACHINE ,
Warner_Ussing = MCS_PRODUCT_ID_WARNER_USSING }

Enumerates the group of MCS devices to connect to.

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

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

- enum class McsUsbSpeedEnumNet {
  LowSpeed = 0 ,
  FullSpeed = 1 ,
  HighSpeed = 2 ,
  SuperSpeed = 3 ,
  UnknownSpeed = 0xff }

  Enumerates the current connection speed of the device
- enum class CFirmwareDestinationNet {
  FPGA_NORMAL = 0 ,
  DSP = MCSUSB_DEST_DSP ,
  USB = MCSUSB_DEST_USB ,
  MCU1 = MCSUSB_DEST_MCU1 ,
  MCSBUS1 = MCSUSB_DEST_MCSBUS1 ,
  MCSBUS2 = MCSUSB_DEST_MCSBUS2 ,
  MCSBUS3 = MCSUSB_DEST_MCSBUS3 ,
  MCSBUS4 = MCSUSB_DEST_MCSBUS4 ,
  MCSBUS5 = MCSUSB_DEST_MCSBUS5 ,
  MCSBUS6 = MCSUSB_DEST_MCSBUS6 ,
  MCSBUS7 = MCSUSB_DEST_MCSBUS7 ,
  MCSBUS8 = MCSUSB_DEST_MCSBUS8 ,
  MCSBUS9 = MCSUSB_DEST_MCSBUS9 ,
  MCSBUS10 = MCSUSB_DEST_MCSBUS10 ,
  MCSBUS11 = MCSUSB_DEST_MCSBUS11 ,
  MCSBUS12 = MCSUSB_DEST_MCSBUS12 ,
  MCSBUS13 = MCSUSB_DEST_MCSBUS13 ,
  MCSBUS14 = MCSUSB_DEST_MCSBUS14 ,
  MCSBUS15 = MCSUSB_DEST_MCSBUS15 ,
  MCSBUS0 = MCSUSB_DEST_MCSBUS0 ,
  BUSNUMBER0 = MCSUSB_DEST_BUSNUMBER0 ,
  BUS0MCSBUS1 = MCSUSB_DEST_BUS0_MCSBUS1 ,
  BUS0MCSBUS2 = MCSUSB_DEST_BUS0_MCSBUS2 ,
  BUS0MCSBUS3 = MCSUSB_DEST_BUS0_MCSBUS3 ,
  BUS0MCSBUS4 = MCSUSB_DEST_BUS0_MCSBUS4 ,
  BUS0MCSBUS5 = MCSUSB_DEST_BUS0_MCSBUS5 ,
  BUS0MCSBUS6 = MCSUSB_DEST_BUS0_MCSBUS6 ,
  BUS0MCSBUS7 = MCSUSB_DEST_BUS0_MCSBUS7 ,
  BUS0MCSBUS8 = MCSUSB_DEST_BUS0_MCSBUS8 ,
  BUS0MCSBUS9 = MCSUSB_DEST_BUS0_MCSBUS9 ,
  BUS0MCSBUS10 = MCSUSB_DEST_BUS0_MCSBUS10 ,
  BUS0MCSBUS11 = MCSUSB_DEST_BUS0_MCSBUS11 ,
  BUS0MCSBUS12 = MCSUSB_DEST_BUS0_MCSBUS12 ,
  BUS0MCSBUS13 = MCSUSB_DEST_BUS0_MCSBUS13 ,
  BUS0MCSBUS14 = MCSUSB_DEST_BUS0_MCSBUS14 ,
  BUS0MCSBUS15 = MCSUSB_DEST_BUS0_MCSBUS15 ,
  BUS0MCSBUS0 = MCSUSB_DEST_BUS0_MCSBUS0 ,
  BUSNUMBER1 = MCSUSB_DEST_BUSNUMBER1 ,
  BUS1MCSBUS1 = MCSUSB_DEST_BUS1_MCSBUS1 ,
  BUS1MCSBUS2 = MCSUSB_DEST_BUS1_MCSBUS2 ,
  BUS1MCSBUS3 = MCSUSB_DEST_BUS1_MCSBUS3 ,
  BUS1MCSBUS4 = MCSUSB_DEST_BUS1_MCSBUS4 ,
  BUS1MCSBUS5 = MCSUSB_DEST_BUS1_MCSBUS5 ,
  BUS1MCSBUS6 = MCSUSB_DEST_BUS1_MCSBUS6 ,
  BUS1MCSBUS7 = MCSUSB_DEST_BUS1_MCSBUS7 ,
  BUS1MCSBUS8 = MCSUSB_DEST_BUS1_MCSBUS8 ,
  BUS1MCSBUS9 = MCSUSB_DEST_BUS1_MCSBUS9 ,
  BUS1MCSBUS10 = MCSUSB_DEST_BUS1_MCSBUS10 ,
  BUS1MCSBUS11 = MCSUSB_DEST_BUS1_MCSBUS11 ,
  BUS1MCSBUS12 = MCSUSB_DEST_BUS1_MCSBUS12 ,

BUS1MCSBUS13 = MCSUSB_DEST_BUS1_MCSBUS13 ,
BUS1MCSBUS14 = MCSUSB_DEST_BUS1_MCSBUS14 ,
BUS1MCSBUS15 = MCSUSB_DEST_BUS1_MCSBUS15 ,
BUS1MCSBUS0 = MCSUSB_DEST_BUS1_MCSBUS0 ,
BUSNUMBER2 = MCSUSB_DEST_BUSNUMBER2 ,
BUS2MCSBUS1 = MCSUSB_DEST_BUS2_MCSBUS1 ,
BUS2MCSBUS2 = MCSUSB_DEST_BUS2_MCSBUS2 ,
BUS2MCSBUS3 = MCSUSB_DEST_BUS2_MCSBUS3 ,
BUS2MCSBUS4 = MCSUSB_DEST_BUS2_MCSBUS4 ,
BUS2MCSBUS5 = MCSUSB_DEST_BUS2_MCSBUS5 ,
BUS2MCSBUS6 = MCSUSB_DEST_BUS2_MCSBUS6 ,
BUS2MCSBUS7 = MCSUSB_DEST_BUS2_MCSBUS7 ,
BUS2MCSBUS8 = MCSUSB_DEST_BUS2_MCSBUS8 ,
BUS2MCSBUS9 = MCSUSB_DEST_BUS2_MCSBUS9 ,
BUS2MCSBUS10 = MCSUSB_DEST_BUS2_MCSBUS10 ,
BUS2MCSBUS11 = MCSUSB_DEST_BUS2_MCSBUS11 ,
BUS2MCSBUS12 = MCSUSB_DEST_BUS2_MCSBUS12 ,
BUS2MCSBUS13 = MCSUSB_DEST_BUS2_MCSBUS13 ,
BUS2MCSBUS14 = MCSUSB_DEST_BUS2_MCSBUS14 ,
BUS2MCSBUS15 = MCSUSB_DEST_BUS2_MCSBUS15 ,
BUS2MCSBUS0 = MCSUSB_DEST_BUS2_MCSBUS0 ,
PIC = MCSUSB_DEST_PIC ,
PIC2 = MCSUSB_DEST_PIC2 ,
PIC3 = MCSUSB_DEST_PIC3 ,
PIC4 = MCSUSB_DEST_PIC4 ,
PIC5 = MCSUSB_DEST_PIC5 ,
PIC6 = MCSUSB_DEST_PIC6 ,
PIC7 = MCSUSB_DEST_PIC7 ,
PIC8 = MCSUSB_DEST_PIC8 ,
PIC9 = MCSUSB_DEST_PIC9 ,
PIC10 = MCSUSB_DEST_PIC10 ,
PIC11 = MCSUSB_DEST_PIC11 ,
PIC12 = MCSUSB_DEST_PIC12 ,
ChannelPIC = MCSUSB_DEST_CHANNELPIC ,
Bootstrap = MCSUSB_DEST_BOOTSTRAP ,
BootstrapOtherCypress = MCSUSB_DEST_BOOTSTAP_OTHER_CYPRESS ,
ALTERA = MCSUSB_DEST_ALTERA ,
FPGA2 = MCSUSB_DEST_FPGA2 ,
FPGA3 = MCSUSB_DEST_FPGA3 ,
FPGA4 = MCSUSB_DEST_FPGA4 ,
FPGA5 = MCSUSB_DEST_FPGA5 ,
FPGA6 = MCSUSB_DEST_FPGA6 ,
FPGA7 = MCSUSB_DEST_FPGA7 ,
FPGA8 = MCSUSB_DEST_FPGA8 ,
FPGA9 = MCSUSB_DEST_FPGA9 ,
FPGA10 = MCSUSB_DEST_FPGA10 ,
FPGA11 = MCSUSB_DEST_FPGA11 ,
FPGA12 = MCSUSB_DEST_FPGA12 ,
FPGA13 = MCSUSB_DEST_FPGA13 ,
FPGA14 = MCSUSB_DEST_FPGA14 ,
FPGA15 = MCSUSB_DEST_FPGA15 ,
FPGA16 = MCSUSB_DEST_FPGA16 ,
FPGA_GOLD = XILINX_DEST_GOLDEN ,
ALTERA_GOLD = (MCSUSB_DEST_ALTERA | XILINX_DEST_GOLDEN) ,
FPGA2_GOLD = (MCSUSB_DEST_FPGA2 | XILINX_DEST_GOLDEN) ,
FPGA3_GOLD = (MCSUSB_DEST_FPGA3 | XILINX_DEST_GOLDEN) ,
FPGA4_GOLD = (MCSUSB_DEST_FPGA4 | XILINX_DEST_GOLDEN) ,
FPGA5_GOLD = (MCSUSB_DEST_FPGA5 | XILINX_DEST_GOLDEN) ,

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

Enumerates the destination processor for the firmware.

- enum class DigitalTargetEnumNet {
  Digout = (MEA_COMMAND << 16) + MEA_MEA21_DIGOUT_SOURCE ,
  Digstream = (MEA_COMMAND << 16) + MEA_MEA21_DIGSTREAM_SOURCE ,

DacqTrigger = (MEA_COMMAND << 16) + MEA_MEA21_DACQTRIGGER_SOURCE ,
StgTrigger = (STG200x_COMMAND << 16) + STG200x_TRIGGER_SOURCE ,
StgListModeTrigger = (STG200x_COMMAND << 16) + STG200x_MEA21_LISTMODE_TRIGGERSOURCE ,
DigOutStimulatorStartTrigger = (MEA_COMMAND << 16) + MEA_DIGOUT_STG_START_TRIGGER_↩SOURCE ,
DigOutStimulatorStopTrigger = (MEA_COMMAND << 16) + MEA_DIGOUT_STG_STOP_TRIGGER_↩SOURCE ,
DigStreamToReceiver = (MEA_COMMAND << 16) + MEA_DIGSTREAMTORECEIVER_SOURCE }

Enumerates the Digital Targets for Digital Sources

- enum class MEA2100DigitalSourceEnumNet {
DigitalInOfOutPort = 0 ,
DigitalIn = 16 ,
DigitalPulse = 32 ,
Feedback = 64 ,
AuxIn = 96 ,
Zero = 98 ,
One = 99 ,
HS1Trigger1Status = 100 ,
HS1Trigger2Status = 102 ,
HS1Trigger3Status = 104 ,
HS1Trigger4Status = 106 ,
HS1Trigger5Status = 108 ,
HS1Trigger6Status = 110 ,
HS1Sideband1 = 112 ,
HS1Sideband2 = 128 ,
HS1Sideband3 = 144 ,
HS1Sideband4 = 160 ,
HS1Sideband5 = 176 ,
HS1Sideband6 = 192 ,
HS2Trigger1Status = 208 ,
HS2Trigger2Status = 210 ,
HS2Trigger3Status = 212 ,
HS2Trigger4Status = 214 ,
HS2Trigger5Status = 216 ,
HS2Trigger6Status = 218 ,
HS2Sideband1 = 220 ,
HS2Sideband2 = 236 ,
HS2Sideband3 = 252 ,
HS2Sideband4 = 268 ,
HS2Sideband5 = 284 ,
HS2Sideband6 = 300 ,
PulseGenerator = 316 ,
DigitalOutStimulator = 320 ,
DigitalData = 336 ,
DeviceRunStatus = 368 ,
LastPosition = 372 }

Enumerates the digital source of the MEA2100 device.

- enum class MultiwellDigitalSourceEnumNet {
DigitalInOfOutPort = 0 ,
DigitalIn = 16 ,
DigitalPulse = 32 ,
Feedback = 64 ,
AuxIn = 96 ,
Zero = 98 ,
One = 99 ,
HS1Trigger1Status = 100 ,
HS1Trigger2Status = 102 ,

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

Enumerates the digital source of the Multiwell device.

- enum class CMOSMEA5000DigitalSourceEnumNet {
DigitalInOfOutPort = 0 ,
DigitalIn = 16 ,
DigitalPulse = 32 ,
Feedback = 64 ,
AuxIn = 96 ,
Zero = 98 ,
One = 99 ,
HS1Trigger1Status = 100 ,
HS1Trigger2Status = 102 ,
HS1Trigger3Status = 104 ,
HS1Sideband1 = 112 ,
HS1Sideband2 = 128 ,
HS1Sideband3 = 144 ,
PulseGenerator = 316 ,
DigitalOutStimulator = 320 ,
DigitalData = 336 ,
DeviceRunStatus = 368 ,
LastPosition = 372 }

Enumerates the digital source of the CMOSMEA5000 device.

- enum class W2100DigitalSourceEnumNet {
DigitalInOfOutPort = 0 ,
DigitalIn = 16 ,
DigitalPulse = 32 ,
Feedback = 64 ,
AuxIn = 96 ,
Zero = 98 ,
One = 99 ,
PulseGenerator = 100 ,
DigDataFromReceiver = 128 ,
DigitalOutStimulator = 192 ,
DigitalData = 208 ,
DeviceRunStatus = 240 ,
DigStreamFromReceiver = 256 ,
LastPosition = 320 }

Enumerates the digital source of the W2100 device.

- enum class SCUDigitalSourceEnumNet {
DigitalInOfOutPort = (0x00 << 8) + 0 ,
DigitalIn = (0x00 << 8) + 16 ,
DigitalPulse = (0x01 << 8) ,
Feedback = (0x02 << 8) ,
AuxIn = (0x03 << 8) + 0 ,
Zero = (0x03 << 8) + 2 ,
One = (0x03 << 8) + 3 ,
PulseGenerator = (0x03 << 8) + 8 ,
DigitalOutStimulator = (0x03 << 8) + 16 ,

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

Enumerates the digital source of the SCU device.

- enum class MEA2100_256DigitalSourceEnumNet {
DigitalInOfOutPort = (0x00 << 8) + 0 ,
DigitalIn = (0x00 << 8) + 16 ,
DigitalPulse = (0x01 << 8) ,
Feedback = (0x02 << 8) ,
AuxIn = (0x03 << 8) + 0 ,

Zero = (0x03 << 8) + 2 ,
One = (0x03 << 8) + 3 ,
DeviceRunStatus = (0x03 << 8) + 4 ,
PulseGenerator = (0x03 << 8) + 8 ,
DigitalOutStimulator = (0x03 << 8) + 16 ,
DigitalData = (0x04 << 8) ,
HS1Trigger1Status = (0x40 << 8) + 0 ,
HS1Trigger2Status = (0x40 << 8) + 2 ,
HS1Trigger3Status = (0x40 << 8) + 4 ,
HS1Trigger4Status = (0x40 << 8) + 6 ,
HS1Trigger5Status = (0x40 << 8) + 8 ,
HS1Trigger6Status = (0x40 << 8) + 10 ,
HS1Trigger7Status = (0x40 << 8) + 12 ,
HS1Trigger8Status = (0x40 << 8) + 14 ,
HS1Trigger9Status = (0x40 << 8) + 16 ,
HS1Trigger10Status = (0x40 << 8) + 18 ,
HS1Trigger11Status = (0x40 << 8) + 20 ,
HS1Trigger12Status = (0x40 << 8) + 22 ,
HS1Trigger13Status = (0x40 << 8) + 24 ,
HS1Trigger14Status = (0x40 << 8) + 26 ,
HS1Trigger15Status = (0x40 << 8) + 28 ,
HS1Trigger16Status = (0x40 << 8) + 30 ,
HS1Trigger17Status = (0x41 << 8) + 0 ,
HS1Trigger18Status = (0x41 << 8) + 2 ,
HS1Sideband1 = (0x42 << 8) ,
HS1Sideband2 = (0x43 << 8) ,
HS1Sideband3 = (0x44 << 8) ,
HS1Sideband4 = (0x45 << 8) ,
HS1Sideband5 = (0x46 << 8) ,
HS1Sideband6 = (0x47 << 8) ,
HS1Sideband7 = (0x48 << 8) ,
HS1Sideband8 = (0x49 << 8) ,
HS1Sideband9 = (0x4A << 8) ,
HS1Sideband10 = (0x4B << 8) ,
HS1Sideband11 = (0x4C << 8) ,
HS1Sideband12 = (0x4D << 8) ,
HS1Sideband13 = (0x4E << 8) ,
HS1Sideband14 = (0x4F << 8) ,
HS1Sideband15 = (0x50 << 8) ,
HS1Sideband16 = (0x51 << 8) ,
HS1Sideband17 = (0x52 << 8) ,
HS1Sideband18 = (0x53 << 8) ,
HS2Trigger1Status = (0x80 << 8) + 0 ,
HS2Trigger2Status = (0x80 << 8) + 2 ,
HS2Trigger3Status = (0x80 << 8) + 4 ,
HS2Trigger4Status = (0x80 << 8) + 6 ,
HS2Trigger5Status = (0x80 << 8) + 8 ,
HS2Trigger6Status = (0x80 << 8) + 10 ,
HS2Trigger7Status = (0x80 << 8) + 12 ,
HS2Trigger8Status = (0x80 << 8) + 14 ,
HS2Trigger9Status = (0x80 << 8) + 16 ,
HS2Trigger10Status = (0x80 << 8) + 18 ,
HS2Trigger11Status = (0x80 << 8) + 20 ,
HS2Trigger12Status = (0x80 << 8) + 22 ,
HS2Trigger13Status = (0x80 << 8) + 24 ,
HS2Trigger14Status = (0x80 << 8) + 26 ,
HS2Trigger15Status = (0x80 << 8) + 28 ,
HS2Trigger16Status = (0x80 << 8) + 30 ,

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

Enumerates the digital source of the MEA2100-256 device.

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

Enumerates the digital source of the TBSI-DACQ device.

- enum class TriggerSourceEnumNet {
tsNone = 0 ,
tsDigitalIn1 = 1 ,
tsDigitalIn2 = 2 ,
tsDigitalIn3 = 3 ,
tsDigitalIn4 = 4 ,
tsDigitalIn5 = 5 ,
tsDigitalIn6 = 6 ,
tsDigitalIn7 = 7 ,
tsDigitalIn8 = 8 ,
tsDigitalIn9 = 9 ,
tsDigitalIn10 = 10 ,
tsDigitalIn11 = 11 ,
tsDigitalIn12 = 12 ,
tsDigitalIn13 = 13 ,
tsDigitalIn14 = 14 ,
tsDigitalIn15 = 15 ,
tsDigitalIn16 = 16 ,
tsDigitalIn17 = 17 ,
tsDigitalIn18 = 18 ,

tsDigitalIn19 = 19 ,
tsDigitalIn20 = 20 ,
tsDigitalIn21 = 21 ,
tsDigitalIn22 = 22 ,
tsDigitalIn23 = 23 ,
tsDigitalIn24 = 24 ,
tsDigitalIn25 = 25 ,
tsDigitalIn26 = 26 ,
tsDigitalIn27 = 27 ,
tsDigitalIn28 = 28 ,
tsDigitalIn29 = 29 ,
tsDigitalIn30 = 30 ,
tsDigitalIn31 = 31 ,
tsDigitalIn32 = 32 ,
tsFeedback1 = 33 ,
tsFeedback2 = 34 ,
tsFeedback3 = 35 ,
tsFeedback4 = 36 ,
tsFeedback5 = 37 ,
tsFeedback6 = 38 ,
tsFeedback7 = 39 ,
tsFeedback8 = 40 ,
tsFeedback9 = 41 ,
tsFeedback10 = 42 ,
tsFeedback11 = 43 ,
tsFeedback12 = 44 ,
tsFeedback13 = 45 ,
tsFeedback14 = 46 ,
tsFeedback15 = 47 ,
tsFeedback16 = 48 ,
tsFeedback17 = 49 ,
tsFeedback18 = 50 ,
tsFeedback19 = 51 ,
tsFeedback20 = 52 ,
tsFeedback21 = 53 ,
tsFeedback22 = 54 ,
tsFeedback23 = 55 ,
tsFeedback24 = 56 ,
tsFeedback25 = 57 ,
tsFeedback26 = 58 ,
tsFeedback27 = 59 ,
tsFeedback28 = 60 ,
tsFeedback29 = 61 ,
tsFeedback30 = 62 ,
tsFeedback31 = 63 ,
tsFeedback32 = 64 ,
tsAuxIn1 = 65 ,
tsAuxIn2 = 66 ,
tsDigitalPuse0 = 67 ,
tsDigitalPuse1 = 68 ,
tsDigitalPuse2 = 69 ,
tsDigitalPuse3 = 70 ,
tsDigitalPuse4 = 71 ,
tsDigitalPuse5 = 72 ,
tsDigitalPuse6 = 73 ,
tsDigitalPuse7 = 74 ,
tsDigitalPuse8 = 75 ,
tsDigitalPuse9 = 76 ,

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

Enumerates the trigger source of the MEA2100 device.

- enum class AnalogSourceEnumNet {
AnalogSource_HS1 ,
AnalogSource_HS2 ,
AnalogSource_IF }

Enumerates the analog source of the MEA2100 device.

- enum class Stg200xTriggerStatusEnumNet {
Idle = 0 ,
Running = 1 ,
Finished = 2 ,
Armed = 3 }

Enumerates the STG download mode trigger status

- enum class Stg3008FilterAmpAmplificationEnumNet {
Gain1000 = 0 ,
Gain500 = 1 ,
Gain200 = 2 ,
Gain100 = 3 }

Enumerates the STG3008FA filter amplifier gains

- enum class RetriggerActionEnumNet {
Stop = STG200x_RETRIGGER_STOP ,
Restart = STG200x_RETRIGGER_RESTART ,
Ignore = STG200x_RETRIGGER_IGNORE ,
Gate = STG200x_RETRIGGER_GATEMODE ,
Single = STG200x_RETRIGGER_SINGLE }

Enumerates possible retrigger actions for STG200x devices.

- enum class Stg200xSegmentFlagsEnumNet {
None = 0 ,
UpdateTrigger = SEGMENTFLAGS_UPDATETRIGGER ,
DownloadOnly = SEGMENTFLAGS_DOWNLOADONLY ,

TriggerOnly = SEGMENTFLAGS_TRIGGERONLY ,
SyncStart = SEGMENTFLAGS_SYNCSTART }

Enumerates Segmentflag options for STG400x devices.

- enum class Stg200xMultiFileSubmodeEnumNet {
MultiFileMode = MULTIFILESUBMODE_MULTIFILEMODE ,
ExtendedMultiFileMode = MULTIFILESUBMODE_EXTENDEDMULTIFILEMODE }

Enumerates EnableMultifileMode submodes for STG devices.

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

Enumerates the DigoutMode on STG400x devices.

- enum class DigitalStimulatorTriggerSlopeEnumNet {
Falling = 0 ,
Rising = 1 }

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

- enum class DigitalStimulatorTriggerEventEnumNet {
Start = 0 ,
Stop = 1 }

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

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

Enumerates the adapter type of the MEA2100 device.

- enum class MeaLayoutEnumNet {
mlUnknown = 0 ,
mlMEA60 = 1 }

Enumerates the MEA layout of the MEA2100 device.

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

  Enumerates the data mode of the device, either 16, 24 or 32 bit, can be signed or unsigned.
- enum class SampleSizeNet {
  SampleSize16Unsigned = 2 ,
  SampleSize16Signed = 2 + 0x100 ,
  SampleSize24Unsigned = 3 ,
  SampleSize24Signed = 3 + 0x100 ,
  SampleSize32Unsigned = 4 ,
  SampleSize32Signed = 4 + 0x100 ,
  SampleSize64Unsigned = 8 ,
  SampleSize64Signed = 8 + 0x100 }

  Enumerates the data format for ChannelBlock functions.
- enum class SampleDstSizeNet {
  SampleDstSize16 = 2 ,
  SampleDstSize32 = 4 }

  Enumerates the destination data format for ChannelBlock functions.
- enum class TcxDeviceTypeEnumNet {
  Unknown = 0 ,
  Regular = 1 ,
  BMI = 2 ,
  Nanion = 3 ,
  Warner = 4 }

  Enumerates the type of TCX devices.
- enum class TcxSensorTypeEnumNet {
  Reserved5 = 0 ,
  Reserved4 = 1 ,
  Reserved3 = 2 ,
  Reserved2 = 3 ,
  Reserved1 = 4 ,
  NTC10K = 5 ,
  PT1000 = 6 ,
  PT100 = 7 }

  Enumerates the sensor types for TCX devices
- enum class STG_DestinationEnumNet {
  channeldata_voltage ,
  channeldata_current ,
  syncoutdata ,
  channeldata_positive_voltage ,
  channeldata_positive_current ,
  rawdata ,
  channeldata_current_own_sync ,
  channeldata_positive_current_own_sync ,
  channeldata_current_own_boost_gnd_sync ,
  channeldata_positive_current_own_boost_gnd_sync ,
  channeldata_current_always_boost ,
  channeldata_current_always_boost_own_sync }

  Enumerates the destination for STG downloads.
- enum class ElectrodeModeEnumNet {
  emAutomatic = 0 ,
  emManual = 3 }

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

- enum class ElectrodeDacMuxEnumNet {
  Ground = 0 ,
  Stg1 = 1 ,
  Stg2 = 2 ,
  Stg3 = 3 }

  Enumerates the setting of the Stimulation DAC Multiplexer.

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

  Enumerates the Channel Groups of Datastream

- enum class W2100DacqGroupChannelEnumNet {
  InterfaceADCGroup = INTERFACEANALOGCHANNELSGROUP ,
  DSPDataGroup = DSPDATACHANNELSGROUP ,
  WirelessHeadStageAnalogRE1HS1 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 0 + 0 ,
  WirelessHeadStageStatusRE1HS1 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 0 + 1 ,
  WirelessHeadStageAnalogRE1HS2 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 2 + 0 ,
  WirelessHeadStageStatusRE1HS2 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 2 + 1 ,
  WirelessHeadStageAnalogRE1HS3 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 4 + 0 ,
  WirelessHeadStageStatusRE1HS3 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 4 + 1 ,
  WirelessHeadStageAnalogRE1HS4 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 6 + 0 ,
  WirelessHeadStageStatusRE1HS4 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 6 + 1 ,
  WirelessHeadStageAnalogRE2HS1 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 0 + 0 ,
  WirelessHeadStageStatusRE2HS1 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 0 + 1 ,
  WirelessHeadStageAnalogRE2HS2 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 2 + 0 ,
  WirelessHeadStageStatusRE2HS2 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 2 + 1 ,
  WirelessHeadStageAnalogRE2HS3 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 4 + 0 ,
  WirelessHeadStageStatusRE2HS3 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 4 + 1 ,
  WirelessHeadStageAnalogRE2HS4 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 6 + 0 ,
  WirelessHeadStageStatusRE2HS4 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 6 + 1 ,
  WirelessHeadStageGyroDataRE1HS1 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 0 + 0 + 16 ,
  WirelessHeadStageAccDataRE1HS1 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 0 + 1 + 16 ,
  WirelessHeadStageGyroDataRE1HS2 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 2 + 0 + 16 ,
  WirelessHeadStageAccDataRE1HS2 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 2 + 1 + 16 ,
  WirelessHeadStageGyroDataRE1HS3 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 4 + 0 + 16 ,
  WirelessHeadStageAccDataRE1HS3 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 4 + 1 + 16 ,
  WirelessHeadStageGyroDataRE1HS4 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 6 + 0 + 16 ,
  WirelessHeadStageAccDataRE1HS4 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 6 + 1 + 16 ,
  WirelessHeadStageGyroDataRE2HS1 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 0 + 0 + 16 ,
  WirelessHeadStageAccDataRE2HS1 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 0 + 1 + 16 ,
  WirelessHeadStageGyroDataRE2HS2 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 2 + 0 + 16 ,
  WirelessHeadStageAccDataRE2HS2 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 2 + 1 + 16 ,
  WirelessHeadStageGyroDataRE2HS3 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 4 + 0 + 16 ,
  WirelessHeadStageAccDataRE2HS3 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 4 + 1 + 16 ,

WirelessHeadStageGyroDataRE2HS4 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 6 + 0 + 16 ,
WirelessHeadStageAccDataRE2HS4 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 6 + 1 + 16 ,
WirelessHeadStageOptoStimCurrentRE1HS1 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 0 + 0 + 32 ,
WirelessHeadStageReservedARE1HS1 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 0 + 1 + 32 ,
WirelessHeadStageOptoStimCurrentRE1HS2 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 2 + 0 + 32 ,
WirelessHeadStageReservedARE1HS2 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 2 + 1 + 32 ,
WirelessHeadStageOptoStimCurrentRE1HS3 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 4 + 0 + 32 ,
WirelessHeadStageReservedARE1HS3 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 4 + 1 + 32 ,
WirelessHeadStageOptoStimCurrentRE1HS4 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 6 + 0 + 32 ,
WirelessHeadStageReservedARE1HS4 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 6 + 1 + 32 ,
WirelessHeadStageOptoStimCurrentRE2HS1 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 0 + 0 + 32 ,
WirelessHeadStageReservedARE2HS1 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 0 + 1 + 32 ,
WirelessHeadStageOptoStimCurrentRE2HS2 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 2 + 0 + 32 ,
WirelessHeadStageReservedARE2HS2 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 2 + 1 + 32 ,
WirelessHeadStageOptoStimCurrentRE2HS3 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 4 + 0 + 32 ,
WirelessHeadStageReservedARE2HS3 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 4 + 1 + 32 ,
WirelessHeadStageOptoStimCurrentRE2HS4 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 6 + 0 + 32 ,
WirelessHeadStageReservedARE2HS4 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 6 + 1 + 32 ,
WirelessHeadStageReservedBRE1HS1 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 0 + 0 + 48 ,
WirelessHeadStageReservedCRE1HS1 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 0 + 1 + 48 ,
WirelessHeadStageReservedBRE1HS2 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 2 + 0 + 48 ,
WirelessHeadStageReservedCRE1HS2 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 2 + 1 + 48 ,
WirelessHeadStageReservedBRE1HS3 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 4 + 0 + 48 ,
WirelessHeadStageReservedCRE1HS3 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 4 + 1 + 48 ,
WirelessHeadStageReservedBRE1HS4 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 6 + 0 + 48 ,
WirelessHeadStageReservedCRE1HS4 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 6 + 1 + 48 ,
WirelessHeadStageReservedBRE2HS1 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 0 + 0 + 48 ,
WirelessHeadStageReservedCRE2HS1 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 0 + 1 + 48 ,
WirelessHeadStageReservedBRE2HS2 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 2 + 0 + 48 ,
WirelessHeadStageReservedCRE2HS2 = WIRELESSHEADSTAGEANALOGGROUPBASE + 8 + 2 + 1 + 48 ,

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

Enumerates the W2100 Channel Groups of Datastream

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

Enumerates the SCU Channel Groups of Datastream

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

Enumerates the MEA2100-256 Channel Groups of Datastream

- enum class DacqMeaGroupTypeEnumNet {
AnalogGroup = ANALOG_GROUP ,
DigitalGroup = DIGITAL_GROUP ,
FrameContextGroup = FRAME_CONTEXT_GROUP }

Enumerations of CMOS MEA Groups to detect 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_100PBB = HS_PLATETYPE_60 ,
Plate_96W700_100PBB = HS_PLATETYPE_61 ,
Plate_ICB8_24W700_100P8A = HS_PLATETYPE_80 ,
Plate_ICB8_96W700_100P8A = HS_PLATETYPE_81 ,
Plate_Dummy_126 = HS_PLATETYPE_126 ,
Plate_24W300_30GMA = HS_PLATETYPE_193 ,
Plate_96W700_100GMA = 194 ,
Plate_24W300_30GBA = HS_PLATETYPE_195 ,
Plate_96W700_100GBA = HS_PLATETYPE_224 ,
Plate_24W300_30GBB = HS_PLATETYPE_232 ,
Plate_96W700_100GBB = HS_PLATETYPE_244 ,
No_Plate = 255 }

- enum class FpgaIdEnumNet {
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 ,
M4M_Dock = FPGA_ID_M4M_DOCK ,
M4M_ReUse = FPGA_ID_M4M_REUSE ,
Mea2100LiteHeadstage = FPGA_ID_MEA2100_LITE_HS ,
LIH30Interfaceboard = FPGA_ID_LIH30_USB_IF ,
LIH30ADCCtrl = FPGA_ID_LIH30_ADC_CTRL ,
UssingRail = FPGA_ID_USSING_RAIL ,
UssingChamber = FPGA_ID_USSING_CHAMBER ,
IFB2GoldenInterfaceboard = FPGA_ID_IFB2_GOLDEN ,
IFB30GoldenInterfaceboard = FPGA_ID_IFB30_GOLDEN ,
DeviceHasNoHeadstage = FPGA_ID_HAS_NO_HS }
- enum class HeadstageIdEnumNet {

DeviceNotConnected = FPGA_ID_NOT_CONNECTED ,
Mea2100 = FPGA_ID_MEA2100_HS ,
Multiwell = FPGA_ID_HS_MULTIWELL ,
TbsiDacq = FPGA_ID_TBSI_DACQ_HS ,
CmosMea = FPGA_ID_CMOS_MEA_HS ,
InvivoSignalCollectorUnit = FPGA_ID_ME2100_InvivoSCU ,
InvitroSignalCollectorUnit = FPGA_ID_ME2100_InvitroSCU ,
Mea2100_256 = FPGA_ID_MEA2100_256_HS ,
W2100WirelessReceiver = FPGA_ID_W2100_REC ,
W2100WirelessReceiverAnalog = FPGA_ID_W2100_REC_ANALOG ,
Mea2100_Lite = FPGA_ID_MEA2100_LITE_HS ,
LIH30ADCCtrl = FPGA_ID_LIH30_ADC_CTRL ,
DeviceHasNoHeadstage = FPGA_ID_HAS_NO_HS }

- enum class SCU_HeadstageIdEnumNet {
  DeviceNotConnected = FPGA_ID_NOT_CONNECTED ,
  Me2100_32Xilinx = FPGA_ID_ME2100_32_XILINX_HS ,
  Me2100_32PICiCE40 = FPGA_ID_ME2100_32_PIC_ICE40_HS ,
  Mea2100Mini60PICiCE40 = FPGA_ID_MEA2100MINI60_PIC_ICE40_HS ,
  Mea2100BetaScreen = FPGA_ID_MEA2100BETASCREEN_HS ,
  Me2100UPA32 = FPGA_ID_ME2100UPA32_HS ,
  MultiwellMini = FPGA_ID_MULTIWELL_MINI_HS ,
  Mea2100Mini120 = FPGA_ID_MEA2100MINI120_HS ,
  Mea2100Mini60ECP5 = FPGA_ID_MEA2100MINI60_ECP5_HS ,
  eCube = FPGA_ID_ECUBE_HS ,
  Me2100Graphene16_32 = FPGA_ID_ME2100_GRAPHENE_16_32_HS ,
  GrapheneASIC = FPGA_ID_GRAPHENE_ASIC_HS ,
  WholeCellPatch = FPGA_ID_WHOLE_CELL_PATCH_HS ,
  DeviceHasNoHeadstage = FPGA_ID_HAS_NO_HS }
- enum class UsbVendorIdEnumNet {
  Unknown = -1 ,
  None = 0 ,
  Renesas = 0x1912 ,
  ASMedia = 0x1b21 ,
  Intel = 0x8086 }
- enum class FilterCalculationDirectionEnumNet {
  DoubleToInt = 0 ,
  IntToDouble = 1 }
- enum class FilterBandEnumNet {
  Unknown = 0 ,
  Lowpass = 1 ,
  Highpass = 2 }
- enum class FilterFamilyEnumNet {
  Unknown = 0 ,
  Bessel = 1 ,
  Butterworth = 2 ,
  RC = 3 }
- enum class FilterTypeEnumNet {
  Hardware = 0 ,
  Software = 1 }
- enum class FilterAttributeEnumNet {
  PreCommaB = 0 ,
  PostCommaB = 1 ,
  CommaPositionB = 2 ,
  PreCommaA = 3 ,
  PostCommaA = 4 ,
  CommaPositionA = 5 }
- enum class AnalogOut_DAC_Range_EnumNet {
  PlusMinus2Comma5Volts = 0 ,

PlusMinus5Volts = 1 ,
PlusMinus10Volts = 2 }

- enum class PP_Pump_Mode_Type_EnumNet {
  Manual = 1 ,
  Digital = 2 ,
  Analog = 3 }
- enum class MbcChargingModeEnumNet {
  StorageCharge = 0 ,
  FullCharge = 1 }
- enum class MbcRatedCapacityEnumNet {
  rc30mAh = 0 ,
  rc100mAh = 1 ,
  rc200mAh = 2 ,
  rc300mAh = 3 ,
  rcGreater300mAh = 4 }
- enum class MbcChannelStateEnumNet {
  csIdleNoBattery = 0 ,
  csIdleChargeFinished = 1 ,
  csCapacityTestPrecharge = 2 ,
  csCapacityTestDischarge = 3 ,
  csRefreshBattery = 4 ,
  csCharge = 5 ,
  csDischarge = 6 ,
  csError = 7 }
- enum class PulseGenerator_Mode_EnumNet {
  Off = 0 ,
  AlwaysOn = 1 ,
  Gated_Low_Active = 2 ,
  Gated_High_Active = 3 }
- enum class LIH30_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 referemce electrode switch of the ME2100 device
- enum class ReferenceElectrodeModeEnumNet {
  SubtractionOff = 0 ,
  SubtractFromAllOther = 1 ,
  SubtractFromReferenceElectrodeOnly = 2 ,
  SubtractFromAll = 3 }

  enumerates the electrode subtraction modes
- enum class DigitalDatastreamEnableEnumNet {
  None = 0x0000 ,
  Mux = 0x0001 ,
  MuxOtherDevice = 0x0002 ,
  DigitalInReserverd = 0x0004 ,
  DigitalIn = 0x0008 ,

DigitalOut = 0x0010 ,
DigitalOutReserved = 0x0020 ,
RegisterLow = 0x0040 ,
RegisterHigh = 0x0080 ,
FeedbackLow = 0x0100 ,
FeedbackHigh = 0x0200 ,
Aux = 0x0400 ,
PeriodicPulse = 0x0800 ,
DigOutStim = 0x1000 ,
Hs1Digital = 0x00008000 ,
Hs1Trigger = 0x00010000 ,
Hs1SidebandLow = 0x00020000 ,
Hs1SidebandHigh = 0x00040000 ,
Hs2Digital = 0x00800000 ,
Hs2Trigger = 0x01000000 ,
Hs2SidebandLow = 0x02000000 ,
Hs2SidebandHigh = 0x04000000 }

enumerates the streams available as digital datastream

- enum class IoVoltageEnumNet {
  Voltage_3V3 = IFB2_IO_VOLTAGE_3V3 ,
  Voltage_5V0 = IFB2_IO_VOLTAGE_5V0 }

  enumerates the I/O Voltages available on the IFB2
- enum class EnSTG200x_STATUS {
  OK ,
  NOT_CONNECTED ,
  DEVICE_NOT_FOUND }

Functions

- public delegate void OnMcsUsbDeviceState (usbSetupPacket_t^ request)
- private delegate void OnMcsUsbDeviceStateCallback (IntPtr pThis, uint32_t size, IntPtr buffer)
- public delegate void OnUpdateFirmwareStatusChange (String^)
- public delegate void OnUpdateFirmwareProgress (int)
- public delegate void OnDeviceArrivalRemoval (CMcsUsbListEntryNet^ entry)

  Delegate to show a device arrival or removal.
- public delegate void OnStgPollStatus (unsigned int status, StgStatusNet^ stgStatusNet, array< int >^ index_list)
- public delegate void OnMwPollStatus (unsigned int CurrentTemp, unsigned int PlateState, unsigned int SwitchState)
- public delegate void RoboStatusEventDelegate (array< unsigned char >^ buffer)
- public delegate void OnStg200xDataHandler (uint32_t trigger)
- public delegate void OnStg200xErrorHandler ()
- public delegate void OnChannelData (CMcsUsbDacqNet^ dacq, int CbHandle, int numFrames)
- public delegate void OnError (String^ msg, int action)

10.2.1 Enumeration Type Documentation

10.2.1.1 AdapterTypeEnumNet `enum AdapterTypeEnumNet [strong]`

Enumerates the adapter type of the MEA2100 device.

Enumerator

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

10.2.1.2 AnalogOut_DAC_Range_EnumNet `enum AnalogOut_DAC_Range_EnumNet [strong]`

Enumerator

PlusMinus2Comma5Volts	
PlusMinus5Volts	
PlusMinus10Volts	

10.2.1.3 AnalogSourceEnumNet `enum AnalogSourceEnumNet [strong]`

Enumerates the analog source of the MEA2100 device.

Enumerator

AnalogSource_HS1	
AnalogSource_HS2	
AnalogSource_IF	

10.2.1.4 AnalogUnitEnumNet enum AnalogUnitEnumNet [strong]

Enumerator

Unknown	
Volt	
Ampere	
Kelvin	

10.2.1.5 CFirmwareDestinationNet enum CFirmwareDestinationNet [strong]

Enumerates the destination processor for the firmware.

Enumerator

FPGA_NORMAL	
DSP	The DSP.
USB	The USB controller.
MCU1	The DSP on the MEA2100 system.
MCSBUS1	
MCSBUS2	
MCSBUS3	
MCSBUS4	
MCSBUS5	
MCSBUS6	
MCSBUS7	
MCSBUS8	
MCSBUS9	
MCSBUS10	
MCSBUS11	
MCSBUS12	
MCSBUS13	
MCSBUS14	
MCSBUS15	
MCSBUS0	
BUSNUMBER0	
BUS0MCSBUS1	
BUS0MCSBUS2	
BUS0MCSBUS3	
BUS0MCSBUS4	
BUS0MCSBUS5	
BUS0MCSBUS6	
BUS0MCSBUS7	
BUS0MCSBUS8	
BUS0MCSBUS9	
BUS0MCSBUS10	
BUS0MCSBUS11	
BUS0MCSBUS12	
BUS0MCSBUS13	
BUS0MCSBUS14	

Enumerator

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	
BUS2MCSBUS7	
BUS2MCSBUS8	
BUS2MCSBUS9	
BUS2MCSBUS10	
BUS2MCSBUS11	
BUS2MCSBUS12	
BUS2MCSBUS13	
BUS2MCSBUS14	
BUS2MCSBUS15	
BUS2MCSBUS0	
PIC	
PIC2	
PIC3	
PIC4	
PIC5	
PIC6	
PIC7	
PIC8	
PIC9	
PIC10	
PIC11	
PIC12	
ChannelPIC	
Bootstrap	

Enumerator

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	
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	

Enumerator

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	
USB_TARGET3	
UnknownDest	

10.2.1.6 CMOSMEA5000DigitalSourceEnumNet `enum CMOSMEA5000DigitalSourceEnumNet [strong]`

Enumerates the digital source of the CMOSMEA5000 device.

Enumerator

DigitalInOfOutPort	
DigitalIn	
DigitalPulse	
Feedback	
AuxIn	
Zero	
One	
HS1Trigger1Status	
HS1Trigger2Status	
HS1Trigger3Status	
HS1Sideband1	

Enumerator

HS1Sideband2	
HS1Sideband3	
PulseGenerator	
DigitalOutStimulator	
DigitalData	
DeviceRunStatus	
LastPosition	

10.2.1.7 CMOSMeaBathModeEnumNet enum CMOSMeaBathModeEnumNet [strong]

Enumerations of CMOS MEA Bath Mode

Enumerator

Ground	
Stimulation	
CurrentMeasure	

10.2.1.8 CMOSMeaHeadstage1NCBathCurrentEnumNet enum CMOSMeaHeadstage1NCBathCurrentEnumNet [strong]

Enumerations of CMOS MEA HS Current Monitoring Channels Group Bitmask

Enumerator

NCBathCurrent	

10.2.1.9 CMOSMeaHeadstage1NCCol2CurrentEnumNet enum CMOSMeaHeadstage1NCCol2CurrentEnumNet [strong]

Enumerations of CMOS MEA HS Current Monitoring Channels Group Bitmask

Enumerator

NCCol2Current	

10.2.1.10 CMOSMeaHeadstage1NChipTempEnumNet enum CMOSMeaHeadstage1NChipTempEnumNet [strong]

Enumerations of CMOS MEA HS Temperature Monitoring Channels Group Bitmask

Enumerator

NChipTemperature	

10.2.1.11 CMOSMeaHS1SidebandEnumNet `enum CMOSMeaHS1SidebandEnumNet [strong]`

Enumerations of CMOS MEA HS STG Sideband Channels Group Bitmask

Enumerator

SBSVector1	
SBSVector2	
SBSVector3	
SBSVector4	

10.2.1.12 CMOSMeaHS1TriggerStatusEnumNet `enum CMOSMeaHS1TriggerStatusEnumNet [strong]`

Enumerations of CMOS MEA HS STG Trigger Status Channels Group Bitmask

Enumerator

TriggerStatus1	
TriggerStatus2	
TriggerStatus3	
TriggerStatus4	

10.2.1.13 CMOSMeaIFDigChannelEnumNet `enum CMOSMeaIFDigChannelEnumNet [strong]`

Enumerations of CMOS MEA IF Digital Channels Group Bitmask

Enumerator

DigitalMux	
DigitalInPort	
DigitalOutReg	
FeedbackReg	
DigitalReg	
AuxPort	

10.2.1.14 CMOSMeaInterfaceADCEnumNet enum CMOSMeaInterfaceADCEnumNet [strong]

Enumerations of CMOS MEA IF Analog Channels Group Bitmask

Enumerator

IFChannel1	
IFChannel2	
IFChannel3	
IFChannel4	
IFChannel5	
IFChannel6	
IFChannel7	
IFChannel8	

10.2.1.15 CMOSMeaPacketFrameContextGroupEnumNet enum CMOSMeaPacketFrameContextGroupEnumNet [strong]

Enumerations of CMOS MEA HS STG Trigger Status Channels Group Bitmask

Enumerator

SOFAndCTRLword	
ChecksumAndPacketCounter	
Timestamp	
EOFAndCRC	

10.2.1.16 CMOSMeaSTG1DACSignalEnumNet enum CMOSMeaSTG1DACSignalEnumNet [strong]

Enumerations of CMOS MEA DAC Stimulation Channels Group Bitmask

Enumerator

DAC1Channel	
DAC2Channel	
DAC3Channel	
DAC4Channel	

10.2.1.17 CMOSMeaValueUnitEnumNet enum CMOSMeaValueUnitEnumNet [strong]

Enumerations of CMOS MEA Units of Values in Data stream

Enumerator

NoUnit	
NanoVolt	
PicoAmpere	
NanoAmpere	
MicroAmpere	
MilliDegreeCelsius	

10.2.1.18 DacqGroupChannelEnumNet `enum DacqGroupChannelEnumNet [strong]`

Enumerates the Channel Groups of Datastream

Enumerator

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

10.2.1.19 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	

10.2.1.20 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	

10.2.1.21 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_MEAUSB_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.

Enumerator

MCS_PPS5_DEVICE	Connect to an MCS PPS5 device.
MCS_OKUVISION_STIMULATOR_DEVICE	Connect to an Okuvision Stimulator device.
MCS_NF_GEN_DEVICE	Connect to an MCS NF-Gen device.
MCS_SAFEIS_DEVICE	Connect to an MCS SafeIS device.
MCS_PERISTALTIC_PUMP_DEVICE	Connect to an MCS PeristalticPump device.
MCS_EXTERN_BC_TESTER_DEVICE	Connect to an ExternBCTester device.
MCS_EXTERN_D_TESTER_DEVICE	Connect to an ExternDTester device.
MCS_SOFTWARE_DONGLE_DEVICE	Connect to an Software Dongle device.
MCS_MEA_CLEAN_DEVICE	Connect to a MEA Clean device.
MCS_MEA_COAT_DEVICE	Connect to a MEA Clean device.
MCS_SMARTIMPLANT_DEVICE	Connect to a SmartImplant device.
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.

10.2.1.22 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.

10.2.1.23 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 FpgaIdEnumNet `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	
CmosmeaIFB2	
M4M_Dock	
M4M_ReUse	
Mea2100LiteHeadstage	
LIH30Interfaceboard	

Enumerator

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	

Enumerator

User_ADC_2	
User_ADC_3	
User_ADC_4	
Test_ADC_EPC10	
ModulA_ADC0	
ModulA_ADC1	
ModulA_ADC2	
ModulA_ADC3	
ModulB_ADC0	
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	

Enumerator

Feedback	
AuxIn	
Zero	
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	

Enumerator

HS2Trigger5Status	
HS2Trigger6Status	
HS2Trigger7Status	
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 MEA2100DigitalSourceEnumNet `enum MEA2100DigitalSourceEnumNet [strong]`

Enumerates the digital source of the MEA2100 device.

Enumerator

DigitalInOfOutPort	
DigitalIn	
DigitalPulse	
Feedback	
AuxIn	
Zero	

Enumerator

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	

10.2.1.49 MeaLayoutEnumNet `enum MeaLayoutEnumNet [strong]`

Enumerates the MEA layout of the MEA2100 device.

Enumerator

mlUnknown	
mlMEA60	

10.2.1.50 MultiwellDigitalSourceEnumNet `enum MultiwellDigitalSourceEnumNet [strong]`

Enumerates the digital source of the Multiwell device.

Enumerator

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

10.2.1.51 MultiwellPlateTypeEnumNet `enum MultiwellPlateTypeEnumNet [strong]`

Enumerator

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

Enumerator

Plate_24W300_30GBA	
Plate_96W700_100GBA	
Plate_24W300_30GBB	
Plate_96W700_100GBB	
No_Plate	

10.2.1.52 PatchServAdcModeEnumNet `enum PatchServAdcModeEnumNet [strong]`

Enumerator

Normal	
CatchAmp	

10.2.1.53 PlateClampEnumNet `enum PlateClampEnumNet [strong]`

Enumerator

Close	
Open	
Stop	

10.2.1.54 PlateClampLockEnumNet `enum PlateClampLockEnumNet [strong]`

Enumerator

Lock	
Unlock	

10.2.1.55 PortDirectionEnumNet `enum PortDirectionEnumNet [strong]`

enumerates a port direction

Enumerator

Output	
Input	

10.2.1.56 PP_Pump_Mode_Type_EnumNet enum PP_Pump_Mode_Type_EnumNet [strong]

Enumerator

Manual	
Digital	
Analog	

10.2.1.57 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	
HekaLIH406	
HekaLIH816	
HekaITEV100	
HekaPG610	
HekaPG611	
HekaPG612	
HekaPG618	
HekaPG690	
HekaEPCLite	
STG	
Octopot	
Tersens	
Dotriapot	
HLA	
STG400x	
STG4002	
STG4004	
STG4008	

Enumerator

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	
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	

Enumerator

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	
SafeIS	
Encapsulator	
NeurochipConfig	
MeasureTable	
Robooycte2	
RoboInject	
HiClamp	
PatchServer	
Dilutor	
HiClamp4Uart	
IM16S16KRA	
IM64KRB	
IS32KRA	
IM64KRC	
IM16S8KRA	
IM16KRC	
SmartImplant	
PositionImp	
PositionBase	
PositionIICentralUnit	
PositionIIBase	

Enumerator

GrapheneProjectTestDevice	
Pos900	
Neptun	
Warner_Valve_Control	
Warner_TEER_Machine	
Warner_Ussing	

10.2.1.58 PulseGenerator_Mode_EnumNet `enum PulseGenerator_Mode_EnumNet [strong]`

Enumerator

Off	
AlwaysOn	
Gated_Low_Active	
Gated_High_Active	

10.2.1.59 ReferenceElectrodeModeEnumNet `enum ReferenceElectrodeModeEnumNet [strong]`

enumerates the electrode subtraction modes

Enumerator

SubtractionOff	
SubtractFromAllOther	
SubtractFromReferenceElectrodeOnly	
SubtractFromAll	

10.2.1.60 ReferenceElectrodeSwitchPositionEnumNet `enum ReferenceElectrodeSwitchPositionEnumNet [strong]`

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

Enumerator

off	
Ref8	
Ref16	
Ref24	
Ref32	

10.2.1.61 RetriggerActionEnumNet enum RetriggerActionEnumNet [strong]

Enumerates possible retrigger actions for STG200x devices.

Enumerator

Stop	
Restart	
Ignore	
Gate	
Single	

10.2.1.62 RoboCurrentModeEnumNet enum RoboCurrentModeEnumNet [strong]

Enumerator

Off	
Break	
Standby	
Reference	
Movement	

10.2.1.63 SampleDstSizeNet enum SampleDstSizeNet [strong]

Enumerates the destination data format for ChannelBlock functions.

Enumerator

SampleDstSize16	
SampleDstSize32	

10.2.1.64 SampleSizeNet enum SampleSizeNet [strong]

Enumerates the data format for ChannelBlock functions.

Enumerator

SampleSize16Unsigned	
SampleSize16Signed	
SampleSize24Unsigned	

Enumerator

SampleSize24Signed	
SampleSize32Unsigned	
SampleSize32Signed	
SampleSize64Unsigned	
SampleSize64Signed	

10.2.1.65 SCU_HeadstageIdEnumNet `enum SCU_HeadstageIdEnumNet [strong]`

Enumerator

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

10.2.1.66 SCUDacqGroupChannelEnumNet `enum SCUDacqGroupChannelEnumNet [strong]`

Enumerates the SCU Channel Groups of Datastream

Enumerator

SCU1ElectrodeGroupHS1	
SCU1ElectrodeGroupHS2	
SCU1ElectrodeGroupHS3	
SCU1ElectrodeGroupHS4	
SCU2ElectrodeGroupHS1	
SCU2ElectrodeGroupHS2	
SCU2ElectrodeGroupHS3	
SCU2ElectrodeGroupHS4	
InterfaceADCGroup	
STG1DACSignalGroup	
STG2DACSignalGroup	

Enumerator

DSPAnalogGroup	
DSPDigitalGroup	
IFDigChannelsGroup	
STG1TriggerStatusGroup	
STG1SidebandsGroup	
STG2TriggerStatusGroup	
STG2SidebandsGroup	
AudioTestChannelGroup	
PacketFrameContextGroup	

10.2.1.67 SCUDigitalSourceEnumNet `enum SCUDigitalSourceEnumNet [strong]`

Enumerates the digital source of the SCU device.

Enumerator

DigitalInOfOutPort	
DigitalIn	
DigitalPulse	
Feedback	
AuxIn	
Zero	
One	
PulseGenerator	
DigitalOutStimulator	
DigitalData	
DeviceRunStatus	
SCU1HS1Sideband1TriggerStatus	
SCU1HS1Sideband2TriggerStatus	
SCU1HS2Sideband1TriggerStatus	
SCU1HS2Sideband2TriggerStatus	
SCU1HS3Sideband1TriggerStatus	
SCU1HS3Sideband2TriggerStatus	
SCU1HS4Sideband1TriggerStatus	
SCU1HS4Sideband2TriggerStatus	
SCU1LEDStim1TriggerStatus	
SCU1LEDStim2TriggerStatus	
SCU1LEDStim3TriggerStatus	
SCU1LEDStim4TriggerStatus	
SCU1HS1Sideband1	
SCU1HS1Sideband2	
SCU1HS2Sideband1	
SCU1HS2Sideband2	
SCU1HS3Sideband1	
SCU1HS3Sideband2	

Enumerator

SCU1HS4Sideband1	
SCU1HS4Sideband2	
SCU1LEDStim1	
SCU1LEDStim2	
SCU1LEDStim3	
SCU1LEDStim4	
SCU2HS1Sideband1TriggerStatus	
SCU2HS1Sideband2TriggerStatus	
SCU2HS2Sideband1TriggerStatus	
SCU2HS2Sideband2TriggerStatus	
SCU2HS3Sideband1TriggerStatus	
SCU2HS3Sideband2TriggerStatus	
SCU2HS4Sideband1TriggerStatus	
SCU2HS4Sideband2TriggerStatus	
SCU2LEDStim1TriggerStatus	
SCU2LEDStim2TriggerStatus	
SCU2LEDStim3TriggerStatus	
SCU2LEDStim4TriggerStatus	
SCU2HS1Sideband1	
SCU2HS1Sideband2	
SCU2HS2Sideband1	
SCU2HS2Sideband2	
SCU2HS3Sideband1	
SCU2HS3Sideband2	
SCU2HS4Sideband1	
SCU2HS4Sideband2	
SCU2LEDStim1	
SCU2LEDStim2	
SCU2LEDStim3	
SCU2LEDStim4	
LastPosition	

10.2.1.68 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

10.2.1.69 Stg200xMultiFileSubmodeEnumNet enum Stg200xMultiFileSubmodeEnumNet [strong]

Enumerates EnableMultifileMode submodes for STG devices.

Enumerator

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

10.2.1.70 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 downlaod segment.
SyncStart	Delay the start the new segment with SegmentStart() until the next sweep has finished.

10.2.1.71 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	

10.2.1.72 Stg3008FilterAmpAmplificationEnumNet enum Stg3008FilterAmpAmplificationEnumNet [strong]

Enumerates the STG3008FA filter amplifier gains

Enumerator

Gain1000	
Gain500	
Gain200	
Gain100	

10.2.1.73 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	

10.2.1.74 StimulationLayoutConfigurationEnumNet enum StimulationLayoutConfigurationEnumNet [strong]

enumerates the layout configuration for the MEA2100-256 device

Enumerator

SingleWell	
SixWell	
NineWell	

10.2.1.75 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	

10.2.1.76 TcxDeviceTypeEnumNet `enum TcxDeviceTypeEnumNet [strong]`

Enumerates the type of TCX devices.

Enumerator

Unknown	
Regular	
BMI	
Nanion	
Warner	

10.2.1.77 TcxSensorTypeEnumNet `enum TcxSensorTypeEnumNet [strong]`

Enumerates the sensor types for TCX devices

Enumerator

Reserved5	
Reserved4	
Reserved3	
Reserved2	
Reserved1	
NTC10K	
PT1000	
PT100	

10.2.1.78 TeerClampModeEnumNet enum TeerClampModeEnumNet [strong]

Enumerator

ClampModeVoltage	
ClampModeCurrent	
ClampModeOpen	
ClampModeInternalCalibration	

10.2.1.79 TeerWaveformEnumNet enum TeerWaveformEnumNet [strong]

Enumerator

Rectangle	
Sine	

10.2.1.80 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	

Enumerator

tsDigitalIn22	
tsDigitalIn23	
tsDigitalIn24	
tsDigitalIn25	
tsDigitalIn26	
tsDigitalIn27	
tsDigitalIn28	
tsDigitalIn29	
tsDigitalIn30	
tsDigitalIn31	
tsDigitalIn32	
tsFeedback1	
tsFeedback2	
tsFeedback3	
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	

Enumerator

tsDigitalPuse5	
tsDigitalPuse6	
tsDigitalPuse7	
tsDigitalPuse8	
tsDigitalPuse9	
tsDigitalPuse10	
tsDigitalPuse11	
tsDigitalPuse12	
tsDigitalPuse13	
tsDigitalPuse14	
tsDigitalPuse15	
tsDigitalPuse16	
tsDigitalPuse17	
tsDigitalPuse18	
tsDigitalPuse19	
tsDigitalPuse20	
tsDigitalPuse21	
tsDigitalPuse22	
tsDigitalPuse23	
tsDigitalPuse24	
tsDigitalPuse25	
tsDigitalPuse26	
tsDigitalPuse27	
tsDigitalPuse28	
tsDigitalPuse29	
tsDigitalPuse30	
tsDigitalPuse31	
tsTriggered	
tsSidebandBit8	
tsDACQCy1Dev1Runs	
tsDACQCy1Dev2Runs	
tsDACQCy2Dev1Runs	
tsDACQCy2Dev2Runs	

10.2.1.81 UsbVendorIdEnumNet `enum UsbVendorIdEnumNet [strong]`

Enumerator

Unknown	
None	
Renesas	
ASMedia	
Intel	

10.2.1.82 UssingClampModeEnumNet enum UssingClampModeEnumNet [strong]

Enumerator

VoltageClamp	
CurrentClamp	
OpenClamp	
Standby	
ElectrodeOffset	

10.2.1.83 UssingUnitEnumNet enum UssingUnitEnumNet [strong]

Enumerator

Volt	
Ampere	
State	

10.2.1.84 VendorIdEnumNet enum VendorIdEnumNet [strong]

Enumerates the group of MCS devices to connect to.

Enumerator

Any	
None	
MCS_old	
MCS	
PCI	
Cypress	
ALA_VC3	

10.2.1.85 W2100_Accel_Gyro_Select_EnumNet enum W2100_Accel_Gyro_Select_EnumNet [strong]

enumerates the accelerometer configuration on the W2100 device

Enumerator

Off	

Enumerator

GyroOnly	
AccelOnly	
Both	

10.2.1.86 W2100DacqGroupChannelEnumNet `enum W2100DacqGroupChannelEnumNet [strong]`

Enumerates the W2100 Channel Groups of Datastream

Enumerator

InterfaceADCGroup	
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	

Enumerator

WirelessHeadStageOptoStimCurrentRE1HS1	
WirelessHeadStageReservedARE1HS1	
WirelessHeadStageOptoStimCurrentRE1HS2	
WirelessHeadStageReservedARE1HS2	
WirelessHeadStageOptoStimCurrentRE1HS3	
WirelessHeadStageReservedARE1HS3	
WirelessHeadStageOptoStimCurrentRE1HS4	
WirelessHeadStageReservedARE1HS4	
WirelessHeadStageOptoStimCurrentRE2HS1	
WirelessHeadStageReservedARE2HS1	
WirelessHeadStageOptoStimCurrentRE2HS2	
WirelessHeadStageReservedARE2HS2	
WirelessHeadStageOptoStimCurrentRE2HS3	
WirelessHeadStageReservedARE2HS3	
WirelessHeadStageOptoStimCurrentRE2HS4	
WirelessHeadStageReservedARE2HS4	
WirelessHeadStageReservedBRE1HS1	
WirelessHeadStageReservedCRE1HS1	
WirelessHeadStageReservedBRE1HS2	
WirelessHeadStageReservedCRE1HS2	
WirelessHeadStageReservedBRE1HS3	
WirelessHeadStageReservedCRE1HS3	
WirelessHeadStageReservedBRE1HS4	
WirelessHeadStageReservedCRE1HS4	
WirelessHeadStageReservedBRE2HS1	
WirelessHeadStageReservedCRE2HS1	
WirelessHeadStageReservedBRE2HS2	
WirelessHeadStageReservedCRE2HS2	
WirelessHeadStageReservedBRE2HS3	
WirelessHeadStageReservedCRE2HS3	
WirelessHeadStageReservedBRE2HS4	
WirelessHeadStageReservedCRE2HS4	
IFDigChannelsGroup	
AudioTestChannelGroup	
PacketFrameContextGroup	

10.2.1.87 W2100DigitalSourceEnumNet `enum W2100DigitalSourceEnumNet [strong]`

Enumerates the digital source of the W2100 device.

Enumerator

DigitalInOfOutPort	
DigitalIn	

Enumerator

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

10.2.1.88 WvcDisplayModeEnumNet `enum WvcDisplayModeEnumNet [strong]`

enumerates Wvc display mode

Enumerator

Work	
PC	
Settings	
TouchTest	

10.2.1.89 WvcValveModeEnumNet `enum WvcValveModeEnumNet [strong]`

enumerates Wvc valve mode

Enumerator

Manual	
Digital	
Analog	
Table	

10.2.2 Function Documentation

10.2.2.1 OnChannelData() `public delegate void Mcs::Usb::OnChannelData (`

`CMcsUsbDacqNet^ dacq,`

```
int CbHandle,
int numFrames )
```

10.2.2.2 OnDeviceArrivalRemoval()

```
public delegate void Mcs::Usb::OnDeviceArrivalRemoval (
    CMcsUsbListEntryNet^ entry )
```

Delegate to show a device arrival or removal.

10.2.2.3 OnError()

```
public delegate void Mcs::Usb::OnError (
    String^ msg,
    int action )
```

10.2.2.4 OnMcsUsbDeviceState()

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

10.2.2.5 OnMcsUsbDeviceStateCallback()

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

10.2.2.6 OnMwPollStatus()

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

10.2.2.7 OnStg200xDataHandler()

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

10.2.2.8 OnStg200xErrorHandler()

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

10.2.2.9 OnStgPollStatus()

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

10.2.2.10 OnUpdateFirmwareProgress()

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

10.2.2.11 OnUpdateFirmwareStatusChange()

```
public delegate void Mcs::Usb::OnUpdateFirmware↩
StatusChange (
        String^  )
```

10.2.2.12 RoboStatusEventDelegate()

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

11 Class Documentation

11.1 CW2100_FunctionNet::AudioChannelsNet Struct Reference

Public Attributes

- W2100DacqGroupChannelEnumNet dacqgroup
- int channel
- int amplification

11.1.1 Member Data Documentation

11.1.1.1 amplification

```
int amplification
```

11.1.1.2 channel

```
int channel
```

11.1.1.3 dacqgroup

```
W2100DacqGroupChannelEnumNet dacqgroup
```

11.2 BatteryState Class Reference

Properties

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

11.2.1 Property Documentation

11.2.1.1 Charge unsigned int Charge [get]

11.2.1.2 ChargeRegionString System:: String^ ChargeRegionString [get]

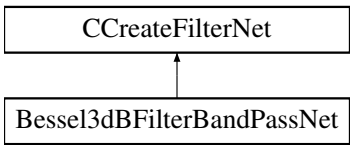
11.2.1.3 ChargeString System:: String^ ChargeString [get]

11.2.1.4 Voltage unsigned int Voltage [get]

11.2.1.5 VoltageString System:: String^ VoltageString [get]

11.3 Bessel3dBFilterBandPassNet Class Reference

Inheritance diagram for Bessel3dBFilterBandPassNet:



Public Member Functions

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

Additional Inherited Members

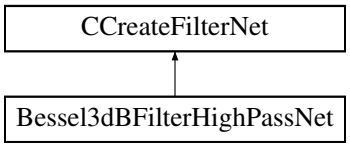
11.3.1 Constructor & Destructor Documentation

11.3.1.1 Bessel3dBFilterBandPassNet() Bessel3dBFilterBandPassNet (

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

11.4 Bessel3dBFilterHighPassNet Class Reference

Inheritance diagram for Bessel3dBFilterHighPassNet:



Public Member Functions

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

Additional Inherited Members

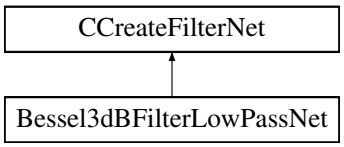
11.4.1 Constructor & Destructor Documentation

11.4.1.1 Bessel3dBFilterHighPassNet() Bessel3dBFilterHighPassNet (

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

11.5 Bessel3dBFilterLowPassNet Class Reference

Inheritance diagram for Bessel3dBFilterLowPassNet:



Public Member Functions

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

Additional Inherited Members

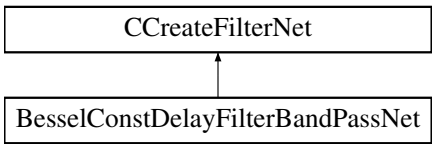
11.5.1 Constructor & Destructor Documentation

11.5.1.1 Bessel3dBFilterLowPassNet() Bessel3dBFilterLowPassNet (

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

11.6 BesselConstDelayFilterBandPassNet Class Reference

Inheritance diagram for BesselConstDelayFilterBandPassNet:



Public Member Functions

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

Additional Inherited Members

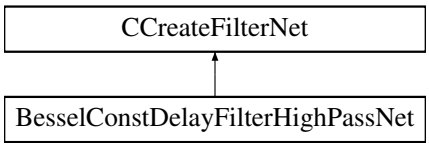
11.6.1 Constructor & Destructor Documentation

11.6.1.1 BesselConstDelayFilterBandPassNet() BesselConstDelayFilterBandPassNet (

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

11.7 BesselConstDelayFilterHighPassNet Class Reference

Inheritance diagram for BesselConstDelayFilterHighPassNet:



Public Member Functions

- BesselConstDelayFilterHighPassNet (int numCoefSets, int order, double sampleRate, double cutoff↩ Frequency, double scale)

Additional Inherited Members

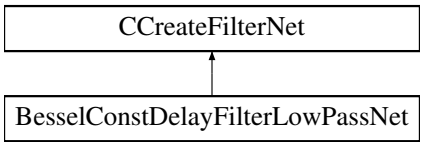
11.7.1 Constructor & Destructor Documentation

11.7.1.1 BesselConstDelayFilterHighPassNet()

```
BesselConstDelayFilterHighPassNet (
        int numCoefSets,
        int order,
        double sampleRate,
        double cutoffFrequency,
        double scale )
```

11.8 BesselConstDelayFilterLowPassNet Class Reference

Inheritance diagram for BesselConstDelayFilterLowPassNet:



Public Member Functions

- BesselConstDelayFilterLowPassNet (int numCoefSets, int order, double sampleRate, double cutoff↩ Frequency, double scale)

Additional Inherited Members

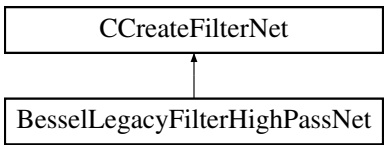
11.8.1 Constructor & Destructor Documentation

11.8.1.1 BesselConstDelayFilterLowPassNet() BesselConstDelayFilterLowPassNet (

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

11.9 BesselLegacyFilterHighPassNet Class Reference

Inheritance diagram for BesselLegacyFilterHighPassNet:



Public Member Functions

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

Additional Inherited Members

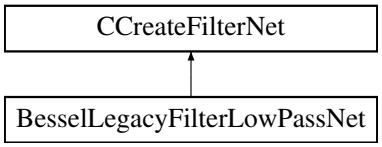
11.9.1 Constructor & Destructor Documentation

11.9.1.1 BesselLegacyFilterHighPassNet() BesselLegacyFilterHighPassNet (

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

11.10 BesselLegacyFilterLowPassNet Class Reference

Inheritance diagram for BesselLegacyFilterLowPassNet:



Public Member Functions

- BesselLegacyFilterLowPassNet (int numCoefSets, int order, double sampleRate, double cutoffFrequency, double scale)

Additional Inherited Members

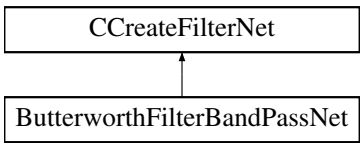
11.10.1 Constructor & Destructor Documentation

11.10.1.1 BesselLegacyFilterLowPassNet() BesselLegacyFilterLowPassNet (

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

11.11 ButterworthFilterBandPassNet Class Reference

Inheritance diagram for ButterworthFilterBandPassNet:



Public Member Functions

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

Additional Inherited Members

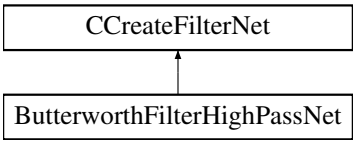
11.11.1 Constructor & Destructor Documentation

11.11.1.1 ButterworthFilterBandPassNet() ButterworthFilterBandPassNet (

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

11.12 ButterworthFilterHighPassNet Class Reference

Inheritance diagram for ButterworthFilterHighPassNet:



Public Member Functions

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

Additional Inherited Members

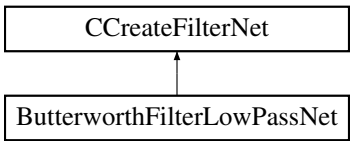
11.12.1 Constructor & Destructor Documentation

11.12.1.1 ButterworthFilterHighPassNet() ButterworthFilterHighPassNet (

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

11.13 ButterworthFilterLowPassNet Class Reference

Inheritance diagram for ButterworthFilterLowPassNet:



Public Member Functions

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

Additional Inherited Members

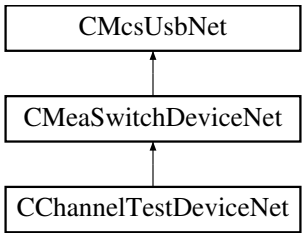
11.13.1 Constructor & Destructor Documentation

11.13.1.1 ButterworthFilterLowPassNet() ButterworthFilterLowPassNet (

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

11.14 CChannelTestDeviceNet Class Reference

Inheritance diagram for CChannelTestDeviceNet:



Public Member Functions

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

Additional Inherited Members

11.14.1 Constructor & Destructor Documentation

11.14.1.1 CChannelTestDeviceNet() `CChannelTestDeviceNet ( )`

11.14.1.2 ~CChannelTestDeviceNet() `~CChannelTestDeviceNet ( )`

11.14.2 Member Function Documentation

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

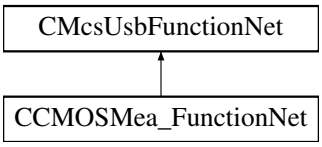
11.14.2.2 SetAttenuation() `void SetAttenuation ( unsigned int Attenuation )`

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

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

11.15 CCMOSMea_FunctionNet Class Reference

Inheritance diagram for CCMOSMea_FunctionNet:



Public Member Functions

- CCMOSMea_FunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ cMOSMea_↩ FunctionPointerContainer)
- CCMOSMea_FunctionNet (CMcsUsbNet^ mcsusb)
- void SetADCInputOffset (int32_t offset)
- int32_t GetADCInputOffset ()
- void SetSourceDrain (int32_t voltage)
- int32_t GetSourceDrain ()
- void SetSourceGate (int32_t voltage)
- int32_t GetSourceGate ()
- void SetSourceBulk (int32_t voltage)
- int32_t GetSourceBulk ()
- void SetGate (int32_t voltage)
- int32_t GetGate ()
- void SetBath (int32_t voltage)
- int32_t GetBath ()
- int32_t GetGNDI ()
- int32_t GetVDDI ()
- int32_t GetVDD3I ()
- void UpdateTransistorVoltages ()
- bool AreTransistorVoltagesSet ()
- void PowerChip (bool on)
- bool IsChipPowered ()
- enCMosMeaChipType DetectChipType ()
- void SetGateToVOP ()
- void SetGateFloating ()
- bool IsGateFloating ()
- void VOPSTimerSetResetTimes (uint32_t ResetTime, uint32_t IntervalTime)
- void VOPSTimerSetResetTimes (uint32_t ResetTime, uint32_t IntervalTime, uint32_t HPFilterResetTime)
- void SetBathMode (CMOSMeaBathModeEnumNet Mode)
- CMOSMeaBathModeEnumNet GetBathMode ()
- void SetNeurochipMemoryData (uint16_t MemAddress, uint32_t MemData)

- void SetNeurochipMemoryData (uint16_t MemAddress, array< uint32_t >^ MemData)
- uint32_t GetNeurochipMemoryData (uint16_t MemAddress)
- array< uint32_t > ^ GetNeurochipMemoryData (uint16_t MemAddress, uint32_t ReqestLength)
- uint32_t GetNeurochipMemorySize ()
- uint32_t GetMaxNumOfColumns (uint32_t Samplerate)
- void SetStimulusSites (List< int16_t >^ SwitchPosition)
- List< int16_t > ^ GetStimulusSites ()
- void ClearSTGOutput (uint32_t Channel)
- uint32_t GetNumberOfSupportedGroups ()
- uint32_t GetNumberOfSupportedGroups (uint32_t virtualDevice)
- DacqGroupChannelEnumNet GetGroupID (uint32_t Index)
- DacqGroupChannelEnumNet GetGroupID (uint32_t Index, uint32_t virtualDevice)
- uint32_t GetGroupNumberOfChannels (DacqGroupChannelEnumNet GroupID)
- uint32_t GetGroupNumberOfChannels (DacqGroupChannelEnumNet GroupID, uint32_t virtualDevice)
- DacqMeaGroupTypeEnumNet GetGroupType (DacqGroupChannelEnumNet GroupID)
- DacqMeaGroupTypeEnumNet GetGroupType (DacqGroupChannelEnumNet GroupID, uint32_t virtual↩ Device)
- void EnableChannelsInGroup (DacqGroupChannelEnumNet GroupID, List< bool >^ EnabledChannelsBit↩ Map)
- void EnableChannelsInGroup (DacqGroupChannelEnumNet GroupID, List< bool >^ EnabledChannelsBit↩ Map, uint32_t virtualDevice)
- List< bool > ^ GetEnabledChannelsInGroup (DacqGroupChannelEnumNet GroupID)
- List< bool > ^ GetEnabledChannelsInGroup (DacqGroupChannelEnumNet GroupID, uint32_t virtualDevice)
- SampleSizeNet GetGroupSampleSize (DacqGroupChannelEnumNet GroupID)
- SampleSizeNet GetGroupSampleSize (DacqGroupChannelEnumNet GroupID, uint32_t virtualDevice)
- uint32_t GetGroupResolutionPerDigit (DacqGroupChannelEnumNet GroupID)
- uint32_t GetGroupResolutionPerDigit (DacqGroupChannelEnumNet GroupID, uint32_t virtualDevice)
- CMOSMeaValueUnitEnumNet GetGroupUnit (DacqGroupChannelEnumNet GroupID)
- CMOSMeaValueUnitEnumNet GetGroupUnit (DacqGroupChannelEnumNet GroupID, uint32_t virtualDevice)
- int32_t GetGroupDCOffset (DacqGroupChannelEnumNet GroupID)
- int32_t GetGroupDCOffset (DacqGroupChannelEnumNet GroupID, uint32_t virtualDevice)
- int32_t GetGroupADCBits (DacqGroupChannelEnumNet GroupID)
- int32_t GetGroupADCBits (DacqGroupChannelEnumNet GroupID, uint32_t virtualDevice)
- uint32_t GetGroupChannelBitmaskBySelect (DacqGroupChannelEnumNet GroupID, uint32_t Channel↩ Number)
- uint32_t GetGroupChannelBitmaskBySelect (DacqGroupChannelEnumNet GroupID, uint32_t Channel↩ Number, uint32_t virtualDevice)
- CMOSMeaInterfaceADCEnumNet GetGroupChannelBitmaskInterfaceADC (uint32_t ChannelNumber)
- CMOSMeaInterfaceADCEnumNet GetGroupChannelBitmaskInterfaceADC (uint32_t ChannelNumber, uint32_t virtualDevice)
- CMOSMeaIFDigChannelEnumNet GetGroupChannelBitmaskIFDigChannels (uint32_t ChannelNumber)
- CMOSMeaIFDigChannelEnumNet GetGroupChannelBitmaskIFDigChannels (uint32_t ChannelNumber, uint32_t virtualDevice)
- CMOSMeaHeadstage1NCBathCurrentEnumNet GetGroupChannelBitmaskHS1NCBathCurrent (uint32_t ChannelNumber)
- CMOSMeaHeadstage1NCBathCurrentEnumNet GetGroupChannelBitmaskHS1NCBathCurrent (uint32_t ChannelNumber, uint32_t virtualDevice)
- CMOSMeaHeadstage1NCCol2CurrentEnumNet GetGroupChannelBitmaskHS1NCCol2Current (uint32_t ChannelNumber)
- CMOSMeaHeadstage1NCCol2CurrentEnumNet GetGroupChannelBitmaskHS1NCCol2Current (uint32_t ChannelNumber, uint32_t virtualDevice)
- CMOSMeaHeadstage1NChipTempEnumNet GetGroupChannelBitmaskHS1NChipTemp (uint32_t Channel↩ Number)
- CMOSMeaHeadstage1NChipTempEnumNet GetGroupChannelBitmaskHS1NChipTemp (uint32_t Channel↩ Number, uint32_t virtualDevice)

- CMOSMeaSTG1DACSignalEnumNet GetGroupChannelBitmaskSTG1DACSignal (uint32_t ChannelNumber)
- CMOSMeaSTG1DACSignalEnumNet GetGroupChannelBitmaskSTG1DACSignal (uint32_t ChannelNumber, uint32_t virtualDevice)
- CMOSMeaHS1SidebandEnumNet GetGroupChannelBitmaskHS1Sidebands (uint32_t ChannelNumber)
- CMOSMeaHS1SidebandEnumNet GetGroupChannelBitmaskHS1Sidebands (uint32_t ChannelNumber, uint32_t virtualDevice)
- CMOSMeaHS1TriggerStatusEnumNet GetGroupChannelBitmaskHS1TriggerStatus (uint32_t ChannelNumber)
- CMOSMeaHS1TriggerStatusEnumNet GetGroupChannelBitmaskHS1TriggerStatus (uint32_t ChannelNumber, uint32_t virtualDevice)
- CMOSMeaPacketFrameContextGroupEnumNet GetGroupChannelBitmaskPacketFrameContext (uint32_t ChannelNumber)
- CMOSMeaPacketFrameContextGroupEnumNet GetGroupChannelBitmaskPacketFrameContext (uint32_t ChannelNumber, uint32_t virtualDevice)

Additional Inherited Members

11.15.1 Constructor & Destructor Documentation

11.15.1.1 CCMOSMea_FunctionNet() [1/2]

```
CCMOSMea_FunctionNet (
    CMcsUsbNet^ mcsusb,
    CMcsUsbFunctionPointerContainer^ cMOSMea_FunctionPointerContainer )
```

11.15.1.2 CCMOSMea_FunctionNet() [2/2]

```
CCMOSMea_FunctionNet (
    CMcsUsbNet^ mcsusb )
```

11.15.2 Member Function Documentation

11.15.2.1 AreTransistorVoltagesSet()

```
bool AreTransistorVoltagesSet ( )
```

11.15.2.2 ClearSTGOutput()

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

11.15.2.3 DetectChipType()

```
enCMosMeaChipType DetectChipType ( )
```

11.15.2.4 EnableChannelsInGroup() [1/2]

```
void EnableChannelsInGroup (
    DacqGroupChannelEnumNet GroupID,
    List< bool >^ EnabledChannelsBitMap )
```

11.15.2.5 EnableChannelsInGroup() [2/2]

```
void EnableChannelsInGroup (
    DacqGroupChannelEnumNet GroupID,
    List< bool >^ EnabledChannelsBitMap,
    uint32_t virtualDevice )
```

11.15.2.6 GetADCInputOffset()

```
int32_t GetADCInputOffset ( )
```

11.15.2.7 GetBath()

```
int32_t GetBath ( )
```

11.15.2.8 GetBathMode()

```
CMOSMeaBathModeEnumNet GetBathMode ( )
```

11.15.2.9 GetEnabledChannelsInGroup() [1/2]

```
List<bool> ^ GetEnabledChannelsInGroup (
    DacqGroupChannelEnumNet GroupID )
```

11.15.2.10 GetEnabledChannelsInGroup() [2/2]

```
List<bool> ^ GetEnabledChannelsInGroup (
    DacqGroupChannelEnumNet GroupID,
    uint32_t virtualDevice )
```

11.15.2.11 GetGate()

```
int32_t GetGate ( )
```

11.15.2.12 GetGNDI()

```
int32_t GetGNDI ( )
```

11.15.2.13 GetGroupADCBits() [1/2]

```
int32_t GetGroupADCBits (
    DacqGroupChannelEnumNet GroupID )
```

11.15.2.14 GetGroupADCBits() [2/2]

```
int32_t GetGroupADCBits (
    DacqGroupChannelEnumNet GroupID,
    uint32_t virtualDevice )
```

11.15.2.15 GetGroupChannelBitmaskBySelect() [1/2]

```
uint32_t GetGroupChannelBitmaskBySelect (
    DacqGroupChannelEnumNet GroupID,
    uint32_t ChannelNumber )
```

11.15.2.16 GetGroupChannelBitmaskBySelect() [2/2]

```
uint32_t GetGroupChannelBitmaskBySelect (
    DacqGroupChannelEnumNet GroupID,
    uint32_t ChannelNumber,
    uint32_t virtualDevice )
```

11.15.2.17 GetGroupChannelBitmaskHS1NCBathCurrent() [1/2]

```
CMOSMeaHeadstage1NCBathCurrentEnumNet
GetGroupChannelBitmaskHS1NCBathCurrent (
    uint32_t ChannelNumber )
```

11.15.2.18 GetGroupChannelBitmaskHS1NCBathCurrent() [2/2]

```
CMOSMeaHeadstage1NCBathCurrentEnumNet
GetGroupChannelBitmaskHS1NCBathCurrent (
    uint32_t ChannelNumber,
    uint32_t virtualDevice )
```

11.15.2.19 GetGroupChannelBitmaskHS1NCCol2Current() [1/2]

```
CMOSMeaHeadstage1NCCol2CurrentEnumNet
GetGroupChannelBitmaskHS1NCCol2Current (
    uint32_t ChannelNumber )
```

11.15.2.20 GetGroupChannelBitmaskHS1NCCol2Current() [2/2]

```
CMOSMeaHeadstage1NCCol2CurrentEnumNet
GetGroupChannelBitmaskHS1NCCol2Current (
    uint32_t ChannelNumber,
    uint32_t virtualDevice )
```

11.15.2.21 GetGroupChannelBitmaskHS1NChipTemp() [1/2]

```
CMOSMeaHeadstage1NChipTempEnumNet
GetGroupChannelBitmaskHS1NChipTemp (
    uint32_t ChannelNumber )
```

11.15.2.22 GetGroupChannelBitmaskHS1NChipTemp() [2/2] CMOSMeaHeadstage1NChipTempEnumNet

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

11.15.2.23 GetGroupChannelBitmaskHS1Sidebands() [1/2] CMOSMeaHS1SidebandEnumNet GetGroup↩

```
ChannelBitmaskHS1Sidebands (
            uint32_t ChannelNumber )
```

11.15.2.24 GetGroupChannelBitmaskHS1Sidebands() [2/2] CMOSMeaHS1SidebandEnumNet GetGroup↩

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

11.15.2.25 GetGroupChannelBitmaskHS1TriggerStatus() [1/2] CMOSMeaHS1TriggerStatusEnumNet Get↩

```
GroupChannelBitmaskHS1TriggerStatus (
            uint32_t ChannelNumber )
```

11.15.2.26 GetGroupChannelBitmaskHS1TriggerStatus() [2/2] CMOSMeaHS1TriggerStatusEnumNet Get↩

```
GroupChannelBitmaskHS1TriggerStatus (
            uint32_t ChannelNumber,
            uint32_t virtualDevice )
```

11.15.2.27 GetGroupChannelBitmaskIFDigChannels() [1/2] CMOSMeaIFDigChannelEnumNet GetGroup↩

```
ChannelBitmaskIFDigChannels (
            uint32_t ChannelNumber )
```

11.15.2.28 GetGroupChannelBitmaskIFDigChannels() [2/2] CMOSMeaIFDigChannelEnumNet GetGroup↩

```
ChannelBitmaskIFDigChannels (
            uint32_t ChannelNumber,
            uint32_t virtualDevice )
```

11.15.2.29 GetGroupChannelBitmaskInterfaceADC() [1/2] CMOSMeaInterfaceADCEnumNet GetGroup↩

```
ChannelBitmaskInterfaceADC (
            uint32_t ChannelNumber )
```

11.15.2.30 GetGroupChannelBitmaskInterfaceADC() [2/2] CMOSMeaInterfaceADCEnumNet GetGroup↩ ChannelBitmaskInterfaceADC (

```
            uint32_t ChannelNumber,
            uint32_t virtualDevice )
```

11.15.2.31 GetGroupChannelBitmaskPacketFrameContext() [1/2] CMOSMeaPacketFrameContextGroupEnumNet GetGroupChannelBitmaskPacketFrameContext (

```
            uint32_t ChannelNumber )
```

11.15.2.32 GetGroupChannelBitmaskPacketFrameContext() [2/2] CMOSMeaPacketFrameContextGroupEnumNet GetGroupChannelBitmaskPacketFrameContext (

```
            uint32_t ChannelNumber,
            uint32_t virtualDevice )
```

11.15.2.33 GetGroupChannelBitmaskSTG1DACSignal() [1/2] CMOSMeaSTG1DACSignalEnumNet Get↩ GroupChannelBitmaskSTG1DACSignal (

```
            uint32_t ChannelNumber )
```

11.15.2.34 GetGroupChannelBitmaskSTG1DACSignal() [2/2] CMOSMeaSTG1DACSignalEnumNet Get↩ GroupChannelBitmaskSTG1DACSignal (

```
            uint32_t ChannelNumber,
            uint32_t virtualDevice )
```

11.15.2.35 GetGroupDCOffset() [1/2] int32_t GetGroupDCOffset (

```
            DacqGroupChannelEnumNet GroupID )
```

11.15.2.36 GetGroupDCOffset() [2/2] int32_t GetGroupDCOffset (

```
            DacqGroupChannelEnumNet GroupID,
            uint32_t virtualDevice )
```

11.15.2.37 GetGroupID() [1/2] DacqGroupChannelEnumNet GetGroupID (

```
            uint32_t Index )
```

11.15.2.38 GetGroupID() [2/2]

```
DacqGroupChannelEnumNet GetGroupID (
    uint32_t Index,
    uint32_t virtualDevice )
```

11.15.2.39 GetGroupNumberOfChannels() [1/2]

```
uint32_t GetGroupNumberOfChannels (
    DacqGroupChannelEnumNet GroupID )
```

11.15.2.40 GetGroupNumberOfChannels() [2/2]

```
uint32_t GetGroupNumberOfChannels (
    DacqGroupChannelEnumNet GroupID,
    uint32_t virtualDevice )
```

11.15.2.41 GetGroupResolutionPerDigit() [1/2]

```
uint32_t GetGroupResolutionPerDigit (
    DacqGroupChannelEnumNet GroupID )
```

11.15.2.42 GetGroupResolutionPerDigit() [2/2]

```
uint32_t GetGroupResolutionPerDigit (
    DacqGroupChannelEnumNet GroupID,
    uint32_t virtualDevice )
```

11.15.2.43 GetGroupSampleSize() [1/2]

```
SampleSizeNet GetGroupSampleSize (
    DacqGroupChannelEnumNet GroupID )
```

11.15.2.44 GetGroupSampleSize() [2/2]

```
SampleSizeNet GetGroupSampleSize (
    DacqGroupChannelEnumNet GroupID,
    uint32_t virtualDevice )
```

11.15.2.45 GetGroupType() [1/2]

```
DacqMeaGroupTypeEnumNet GetGroupType (
    DacqGroupChannelEnumNet GroupID )
```

11.15.2.46 GetGroupType() [2/2]

```
DacqMeaGroupTypeEnumNet GetGroupType (
    DacqGroupChannelEnumNet GroupID,
    uint32_t virtualDevice )
```

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

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

11.15.2.49 GetMaxNumOfColumns() uint32_t GetMaxNumOfColumns (uint32_t *Samplerate*)

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

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

11.15.2.52 GetNeurochipMemorySize() uint32_t GetNeurochipMemorySize ( )

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

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

11.15.2.55 GetSourceBulk() int32_t GetSourceBulk ( )

11.15.2.56 GetSourceDrain() int32_t GetSourceDrain ( )

11.15.2.57 GetSourceGate() `int32_t GetSourceGate ( )`

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

11.15.2.59 GetVDD3I() `int32_t GetVDD3I ( )`

11.15.2.60 GetVDDI() `int32_t GetVDDI ( )`

11.15.2.61 IsChipPowered() `bool IsChipPowered ( )`

11.15.2.62 IsGateFloating() `bool IsGateFloating ( )`

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

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

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

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

11.15.2.67 SetGate() void SetGate (int32_t *voltage*)

11.15.2.68 SetGateFloating() void SetGateFloating ( )

11.15.2.69 SetGateToVOP() void SetGateToVOP ( )

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

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

11.15.2.72 SetSourceBulk() void SetSourceBulk (int32_t *voltage*)

11.15.2.73 SetSourceDrain() void SetSourceDrain (int32_t *voltage*)

11.15.2.74 SetSourceGate() void SetSourceGate (int32_t *voltage*)

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

11.15.2.76 UpdateTransistorVoltages() void UpdateTransistorVoltages ( )

11.15.2.77 VOPSTimerSetResetTimes() [1/2]

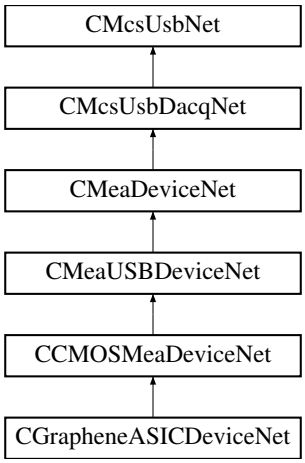
```
void VOPSTimerSetResetTimes (
    uint32_t ResetTime,
    uint32_t IntervalTime )
```

11.15.2.78 VOPSTimerSetResetTimes() [2/2]

```
void VOPSTimerSetResetTimes (
    uint32_t ResetTime,
    uint32_t IntervalTime,
    uint32_t HPFilterResetTime )
```

11.16 CCMOSMeaDeviceNet Class Reference

Inheritance diagram for CCMOSMeaDeviceNet:



Classes

- class CRegionOfInterestRect

Public Member Functions

- CCMOSMeaDeviceNet (void)
- ~CCMOSMeaDeviceNet ()
- virtual void SetBaseSamplerate (int BaseSamplerate)
- virtual int GetBaseSamplerate ()
- virtual array< int > ^ GetAvailableBaseSamplerates ()
- int GetMaxReadableColumns ()
- virtual void SetRegionOfInterests (System::Collections::Generic::Dictionary< int, CRegionOfInterestRect^>^ rois)
- void UpdateChannelBlock (int queuesize, int threshold, int channels_in_block)
- System::Collections::Generic::Dictionary< int, array< array< int16_t >^>^> ^ GetCMOSDataDictionary (int frames, [System::Runtime::InteropServices::Out]int % frames_ret)
- System::Collections::Generic::Dictionary< int, array< uint16_t >^> ^ GetChannelDataUI16 (DacqGroupChannelEnumNet group, int frames, [System::Runtime::InteropServices::Out]int % frames_ret)
- System::Collections::Generic::Dictionary< int, array< int16_t >^> ^ GetChannelDataI16 (DacqGroupChannelEnumNet group, int frames, [System::Runtime::InteropServices::Out]int % frames_ret)
- System::Collections::Generic::Dictionary< int, array< uint32_t >^> ^ GetChannelDataUI32 (DacqGroupChannelEnumNet group, int frames, [System::Runtime::InteropServices::Out]int % frames_ret)
- System::Collections::Generic::Dictionary< int, array< int32_t >^> ^ GetChannelDataI32 (DacqGroupChannelEnumNet group, int frames, [System::Runtime::InteropServices::Out]int % frames_ret)

Properties

- CCMOSMea_FunctionNet^ CMosMea [get]
- CStimulusFunctionNet^ Stimulus [get]

Additional Inherited Members

11.16.1 Constructor & Destructor Documentation

11.16.1.1 CCMOSMeaDeviceNet() CCMOSMeaDeviceNet (void)

11.16.1.2 ~CCMOSMeaDeviceNet() ~CCMOSMeaDeviceNet ( )

11.16.2 Member Function Documentation

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

Reimplemented in CGrapheneASICDeviceNet.

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

Reimplemented in CGrapheneASICDeviceNet.

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

11.16.2.4 GetChannelDataI32()

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

11.16.2.5 GetChannelDataUI16()

```
System::Collections::Generic::Dictionary<int, array<uint16_↩
t>^>^ GetChannelDataUI16 (
            DacqGroupChannelEnumNet group,
            int frames,
            [System::Runtime::InteropServices::Out] int % frames_ret )
```

11.16.2.6 GetChannelDataUI32()

```
System::Collections::Generic::Dictionary<int, array<uint32_↩
t>^>^ GetChannelDataUI32 (
            DacqGroupChannelEnumNet group,
            int frames,
            [System::Runtime::InteropServices::Out] int % frames_ret )
```

11.16.2.7 GetCMOSDataDictionary()

```
System::Collections::Generic::Dictionary<int, array<array<int16↩
_t>^>^> ^ GetCMOSDataDictionary (
            int frames,
            [System::Runtime::InteropServices::Out] int % frames_ret )
```

11.16.2.8 GetMaxReadableColumns()

```
int GetMaxReadableColumns ( )
```

11.16.2.9 SetBaseSamplerate()

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

Reimplemented in CGrapheneASICDeviceNet.

11.16.2.10 SetRegionOfInterests()

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

11.16.2.11 UpdateChannelBlock() `void UpdateChannelBlock (`

```
int queuesize,
int threshold,
int channels_in_block )
```

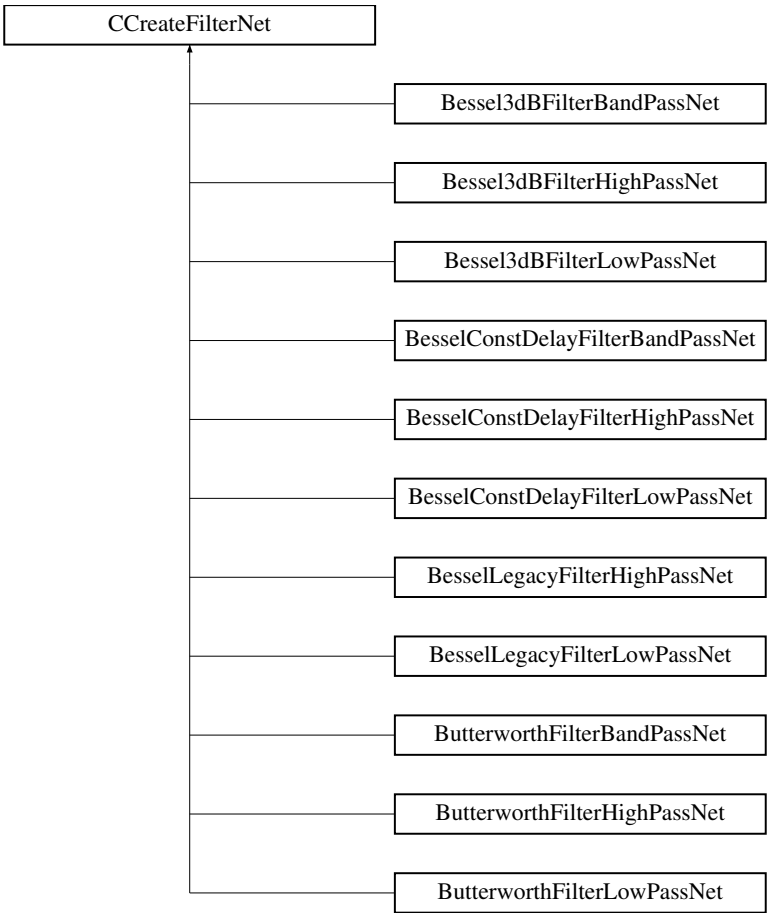
11.16.3 Property Documentation

11.16.3.1 CMosMea `CCMOSMea_FunctionNet^ CMosMea [get]`

11.16.3.2 Stimulus `CStimulusFunctionNet^ Stimulus [get]`

11.17 CCreateFilterNet Class Reference

Inheritance diagram for CCreateFilterNet:



Public Member Functions

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

Static Public Member Functions

- static int FindFilter (array< array< uint64_t >^ >^ coef, array< CCreateFilterNet^ >^ param, CFilterCoefficientsNet::s_FilterAttributesNet^ FiltAttr, bool DoMCSLegacyCompare)

Protected Member Functions

- CCreateFilterNet (int numCoefSets, CCreateFilter *pCreateFilter)

Properties

- int NumCoefSets [get]
- int Order [get]
- double SampleRate [get]
- double CutoffOrCenterFrequency [get]
- double WidthFrequency [get]
- double Scale [get]

11.17.1 Constructor & Destructor Documentation

11.17.1.1 CCreateFilterNet() [1/2]

```
CCreateFilterNet (
    int numCoefSets,
    int order,
    double sampleRate,
    double cutoffFrequency,
    double scale )
```

11.17.1.2 ~CCreateFilterNet()

```
~CCreateFilterNet ( )
```

11.17.1.3 CCreateFilterNet() [2/2]

```
CCreateFilterNet (
    int numCoefSets,
    CCreateFilter * pCreateFilter )  [protected]
```

11.17.2 Member Function Documentation

11.17.2.1 FindFilter()

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

11.17.2.2 GetBiQuad()

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

11.17.2.3 GetBiQuads()

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

11.17.2.4 GetCpp()

```
CCreateFilter* GetCpp ( )
```

11.17.3 Property Documentation

11.17.3.1 CutoffOrCenterFrequency

```
double CutoffOrCenterFrequency  [get]
```

11.17.3.2 NumCoefSets

```
int NumCoefSets  [get]
```

11.17.3.3 Order

```
int Order  [get]
```

11.17.3.4 SampleRate

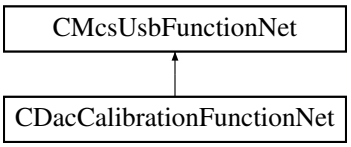
```
double SampleRate  [get]
```

11.17.3.5 Scale `double Scale [get]`

11.17.3.6 WidthFrequency `double WidthFrequency [get]`

11.18 CDacCalibrationFunctionNet Class Reference

Inheritance diagram for CDacCalibrationFunctionNet:



Public Member Functions

- CDacCalibrationFunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ pDac↩ CalibrationFunctionPointerContainer)

  Initializes a new instance of the CDacCalibrationFunctionNet class.
- CDacCalibrationFunctionNet (CMcsUsbNet^ mcsusb)
- virtual ~CDacCalibrationFunctionNet ()
- !CDacCalibrationFunctionNet ()
- void SetDacOffset (uint16_t dacChannel, int32_t offset)

  Sets the offset of a DAC channel.
- int32_t GetDacOffset (uint16_t dacChannel)

  Gets the offset of a DAC channel.
- void BurnDacOffset (uint16_t dacChannel)

  Writes the offset of a DAC channel to permanent memory.

Additional Inherited Members

11.18.1 Detailed Description

11.18.2 Constructor & Destructor Documentation

11.18.2.1 CDacCalibrationFunctionNet() [1/2]

```
CDacCalibrationFunctionNet (
    CMcsUsbNet^ mcsusb,
    CMcsUsbFunctionPointerContainer^ pDacCalibrationFunctionPointerContainer )
```

Initializes a new instance of the CDacCalibrationFunctionNet class.

11.18.2.2 CDacCalibrationFunctionNet() [2/2]

```
CDacCalibrationFunctionNet (
    CMcsUsbNet^ mcsusb )
```

11.18.2.3 ~CDacCalibrationFunctionNet()

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

11.18.2.4 "!CDacCalibrationFunctionNet()

```
!CDacCalibrationFunctionNet ( )
```

11.18.3 Member Function Documentation

11.18.3.1 BurnDacOffset()

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

Writes the offset of a DAC channel to permanent memory.

Parameters

dacChannel	The DAC channel number.

11.18.3.2 GetDacOffset()

```
int32_t GetDacOffset (
    uint16_t dacChannel )
```

Gets the offset of a DAC channel.

Parameters

dacChannel	The DAC channel number.

Returns

The offset in digits.

11.18.3.3 SetDacOffset()

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

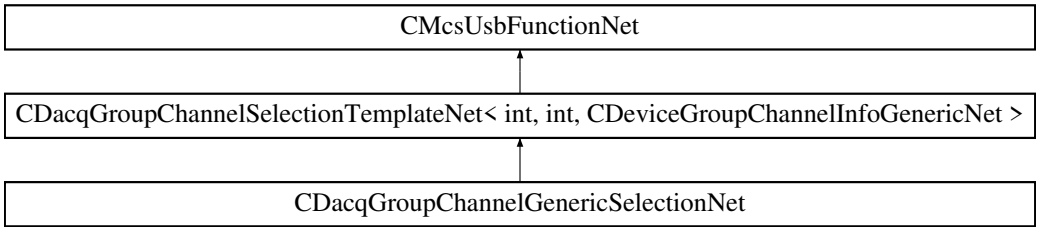
Sets the offset of a DAC channel.

Parameters

dacChannel	The DAC channel number.
offset	The offset in digits.

11.19 CDacqGroupChannelGenericSelectionNet Class Reference

Inheritance diagram for CDacqGroupChannelGenericSelectionNet:



Public Member Functions

- CDacqGroupChannelGenericSelectionNet (CMcsUsbNet^ mcsusb)

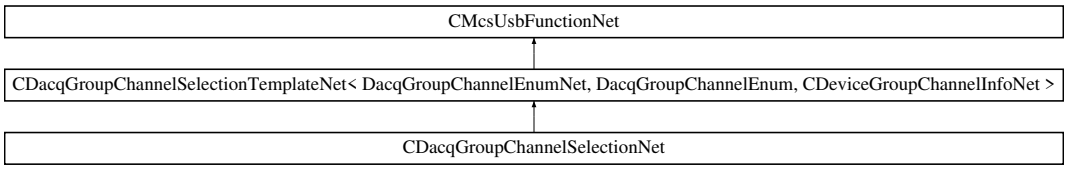
Additional Inherited Members

11.19.1 Constructor & Destructor Documentation

11.19.1.1 CDacqGroupChannelGenericSelectionNet() `CDacqGroupChannelGenericSelectionNet ( CMcsUsbNet^ mcsusb )`

11.20 CDacqGroupChannelSelectionNet Class Reference

Inheritance diagram for CDacqGroupChannelSelectionNet:



Public Member Functions

- CDacqGroupChannelSelectionNet (CMcsUsbNet^ mcsusb)

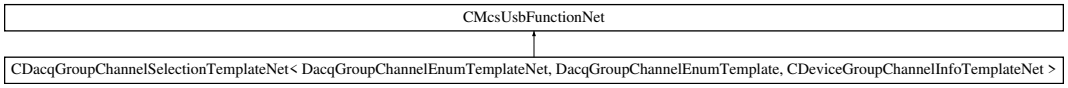
Additional Inherited Members

11.20.1 Constructor & Destructor Documentation

11.20.1.1 CDacqGroupChannelSelectionNet() `CDacqGroupChannelSelectionNet ( CMcsUsbNet^ mcsusb )`

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

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



Public Member Functions

- CDacqGroupChannelSelectionTemplateNet (CMcsUsbNet^ mcsusb)
- uint32_t GetNumberOfSupportedGroups ()
- uint32_t GetNumberOfSupportedGroups (uint32_t virtualDevice)
- DacqGroupChannelEnumTemplateNet GetGroupID (uint32_t Index)
- DacqGroupChannelEnumTemplateNet GetGroupID (uint32_t Index, uint32_t virtualDevice)
- uint32_t GetGroupNumberOfChannels (DacqGroupChannelEnumTemplateNet GroupID)
- uint32_t GetGroupNumberOfChannels (DacqGroupChannelEnumTemplateNet GroupID, uint32_t virtual↩ Device)
- DacqMeaGroupTypeEnumNet GetGroupType (DacqGroupChannelEnumTemplateNet GroupID)
- DacqMeaGroupTypeEnumNet GetGroupType (DacqGroupChannelEnumTemplateNet GroupID, uint32_↩ t virtualDevice)
- void EnableChannelsInGroup (DacqGroupChannelEnumTemplateNet GroupID, List< bool >^ Enabled↩ ChannelsBitMap)
- void EnableChannelsInGroup (DacqGroupChannelEnumTemplateNet GroupID, List< bool >^ Enabled↩ ChannelsBitMap, uint32_t virtualDevice)
- List< bool > ^ GetEnabledChannelsInGroup (DacqGroupChannelEnumTemplateNet GroupID)
- List< bool > ^ GetEnabledChannelsInGroup (DacqGroupChannelEnumTemplateNet GroupID, uint32_↩ t virtualDevice)
- SampleSizeNet GetGroupSampleSize (DacqGroupChannelEnumTemplateNet GroupID)
- SampleSizeNet GetGroupSampleSize (DacqGroupChannelEnumTemplateNet GroupID, uint32_t virtual↩ Device)
- List< CDeviceGroupChannelInfoTemplateNet^ > ^ GetDeviceGroupChannelInfos ()
- List< CDeviceGroupChannelInfoTemplateNet^ > ^ GetDeviceGroupChannelInfos (uint32_t virtualDevice)

Additional Inherited Members

11.21.1 Constructor & Destructor Documentation

11.21.1.1 CDacqGroupChannelSelectionTemplateNet()

```
CDacqGroupChannelSelectionTemplateNet (
    CMcsUsbNet^ mcsusb )
```

11.21.2 Member Function Documentation

11.21.2.1 EnableChannelsInGroup() [1/2]

```
void EnableChannelsInGroup (
    DacqGroupChannelEnumTemplateNet GroupID,
    List< bool >^ EnabledChannelsBitMap )
```

11.21.2.2 EnableChannelsInGroup() [2/2]

```
void EnableChannelsInGroup (
    DacqGroupChannelEnumTemplateNet GroupID,
    List< bool >^ EnabledChannelsBitMap,
    uint32_t virtualDevice )
```

11.21.2.3 GetDeviceGroupChannelInfos() [1/2]

```
List<CDeviceGroupChannelInfoTemplateNet^>^ GetDeviceGroupChannelInfos ( )
```

11.21.2.4 GetDeviceGroupChannelInfos() [2/2]

```
List<CDeviceGroupChannelInfoTemplateNet^>^ GetDeviceGroupChannelInfos (
    uint32_t virtualDevice )
```

11.21.2.5 GetEnabledChannelsInGroup() [1/2]

```
List<bool> ^ GetEnabledChannelsInGroup (
    DacqGroupChannelEnumTemplateNet GroupID )
```

11.21.2.6 GetEnabledChannelsInGroup() [2/2]

```
List<bool> ^ GetEnabledChannelsInGroup (
    DacqGroupChannelEnumTemplateNet GroupID,
    uint32_t virtualDevice )
```

11.21.2.7 GetGroupID() [1/2]

```
DacqGroupChannelEnumTemplateNet GetGroupID (
    uint32_t Index )
```

11.21.2.8 GetGroupID() [2/2]

```
DacqGroupChannelEnumTemplateNet GetGroupID (
    uint32_t Index,
    uint32_t virtualDevice )
```

11.21.2.9 GetGroupNumberOfChannels() [1/2]

```
uint32_t GetGroupNumberOfChannels (
    DacqGroupChannelEnumTemplateNet GroupID )
```

11.21.2.10 GetGroupNumberOfChannels() [2/2]

```
uint32_t GetGroupNumberOfChannels (
    DacqGroupChannelEnumTemplateNet GroupID,
    uint32_t virtualDevice )
```

11.21.2.11 GetGroupSampleSize() [1/2]

```
SampleSizeNet GetGroupSampleSize (
    DacqGroupChannelEnumTemplateNet GroupID )
```

11.21.2.12 GetGroupSampleSize() [2/2]

```
SampleSizeNet GetGroupSampleSize (
    DacqGroupChannelEnumTemplateNet GroupID,
    uint32_t virtualDevice )
```

11.21.2.13 GetGroupType() [1/2]

```
DacqMeaGroupTypeEnumNet GetGroupType (
    DacqGroupChannelEnumTemplateNet GroupID )
```

11.21.2.14 GetGroupType() [2/2]

```
DacqMeaGroupTypeEnumNet GetGroupType (
    DacqGroupChannelEnumTemplateNet GroupID,
    uint32_t virtualDevice )
```

11.21.2.15 GetNumberOfSupportedGroups() [1/2]

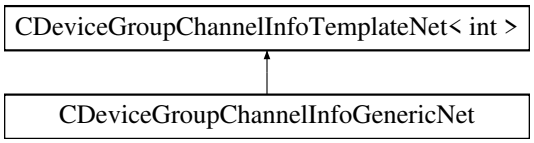
```
uint32_t GetNumberOfSupportedGroups ( )
```

11.21.2.16 GetNumberOfSupportedGroups() [2/2]

```
uint32_t GetNumberOfSupportedGroups (
    uint32_t virtualDevice )
```

11.22 CDeviceGroupChannelInfoGenericNet Class Reference

Inheritance diagram for CDeviceGroupChannelInfoGenericNet:



Public Member Functions

- CDeviceGroupChannelInfoGenericNet (int id, int channels, DacqMeaGroupTypeEnumNet type)

Additional Inherited Members

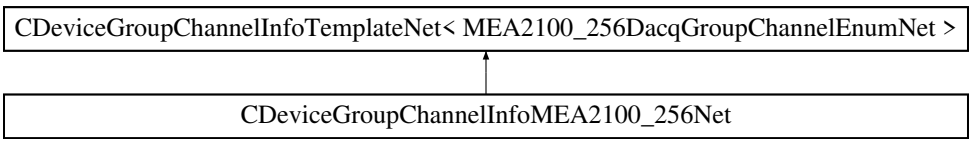
11.22.1 Constructor & Destructor Documentation

11.22.1.1 CDeviceGroupChannelInfoGenericNet()

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

11.23 CDeviceGroupChannelInfoMEA2100_256Net Class Reference

Inheritance diagram for CDeviceGroupChannelInfoMEA2100_256Net:



Public Member Functions

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

Additional Inherited Members

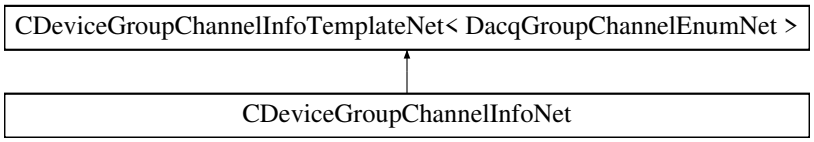
11.23.1 Constructor & Destructor Documentation

11.23.1.1 CDeviceGroupChannelInfoMEA2100_256Net()

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

11.24 CDeviceGroupChannelInfoNet Class Reference

Inheritance diagram for CDeviceGroupChannelInfoNet:



Public Member Functions

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

Additional Inherited Members

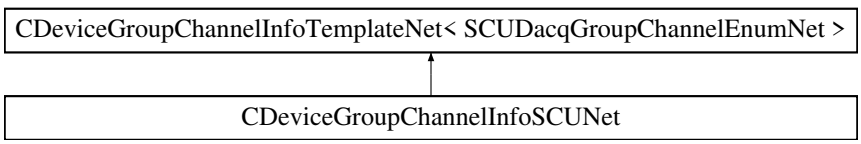
11.24.1 Constructor & Destructor Documentation

11.24.1.1 CDeviceGroupChannelInfoNet()

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

11.25 CDeviceGroupChannelInfoSCUNet Class Reference

Inheritance diagram for CDeviceGroupChannelInfoSCUNet:



Public Member Functions

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

Additional Inherited Members

11.25.1 Constructor & Destructor Documentation

11.25.1.1 CDeviceGroupChannelInfoSCUNet() `CDeviceGroupChannelInfoSCUNet (`

```
SCUDacqGroupChannelEnumNet id,
int channels,
DacqMeaGroupTypeEnumNet type )
```

11.26 CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumTemplateNet > Class Template Reference

Public Member Functions

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

Public Attributes

- DacqGroupChannelEnumTemplateNet GroupID
- int NumberOfChannels
- DacqMeaGroupTypeEnumNet GroupType

11.26.1 Constructor & Destructor Documentation

11.26.1.1 CDeviceGroupChannelInfoTemplateNet() `CDeviceGroupChannelInfoTemplateNet (`

```
DacqGroupChannelEnumTemplateNet id,
int channels,
DacqMeaGroupTypeEnumNet type )
```

11.26.2 Member Data Documentation

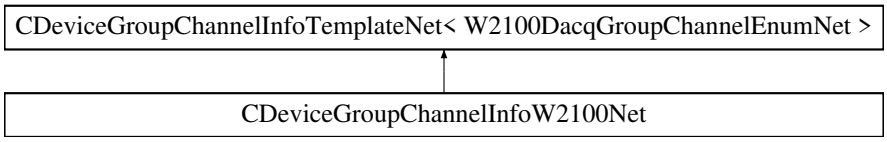
11.26.2.1 GroupID `DacqGroupChannelEnumTemplateNet GroupID`

11.26.2.2 GroupType `DacqMeaGroupTypeEnumNet GroupType`

11.26.2.3 NumberOfChannels `int NumberOfChannels`

11.27 CDeviceGroupChannelInfoW2100Net Class Reference

Inheritance diagram for CDeviceGroupChannelInfoW2100Net:



Public Member Functions

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

Additional Inherited Members

11.27.1 Constructor & Destructor Documentation

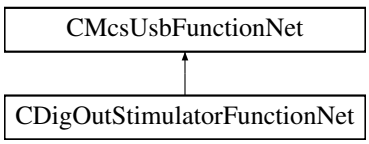
11.27.1.1 CDeviceGroupChannelInfoW2100Net() `CDeviceGroupChannelInfoW2100Net (`

```
W2100DacqGroupChannelEnumNet id,
int channels,
DacqMeaGroupTypeEnumNet type )
```

11.28 CDigOutStimulatorFunctionNet Class Reference

CDigOutStimulatorFunctionNet is the class of the DigOut stimulator function class.

Inheritance diagram for CDigOutStimulatorFunctionNet:



Public Member Functions

- CDigOutStimulatorFunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ pDigOut↩ StimulatorFunctionPointerContainer)

  Initializes a new instance of the CDigOutStimulatorFunctionNet class.
- CDigOutStimulatorFunctionNet (CMcsUsbNet^ mcsusb)
- virtual ~CDigOutStimulatorFunctionNet ()
- !CDigOutStimulatorFunctionNet ()
- void ClearChannel (int32_t NrChannel)

  clear stimulation pattern
- CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData ^ PrepareChannelData (array< int32_t >^ Amplitude, array< uint64_t >^ Duration)

  prepares the channel data for the device and unrolles the data for the GUI
- void SendChannelData (int32_t NrChannel, CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData^ device_data_and_unrolled)

  send or append stimulation pattern
- int32_t GetNumberOfChannels ()

  get the number of channels available on the device
- void SetGlobalRepeat (int32_t NrChannel, bool value)

  set repeat whole stimulation pattern
- bool GetGlobalRepeat (int32_t NrChannel)

  get repeat whole stimulation pattern
- void SetStartTriggerSlope (int32_t NrChannel, DigitalStimulatorTriggerSlopeEnumNet Condition)

  sets start condition of digital out stimulator
- DigitalStimulatorTriggerSlopeEnumNet GetStartTriggerSlope (int32_t NrChannel)

  queries start condition of digital out stimulator
- void SetStopTriggerSlope (int32_t NrChannel, DigitalStimulatorTriggerSlopeEnumNet Condition)

  sets stop condition of digital out stimulator
- DigitalStimulatorTriggerSlopeEnumNet GetStopTriggerSlope (int32_t NrChannel)

  queries stop condition of digital out stimulator

Additional Inherited Members

11.28.1 Detailed Description

CDigOutStimulatorFunctionNet is the class of the DigOut stimulator function class.

11.28.2 Constructor & Destructor Documentation

11.28.2.1 CDigOutStimulatorFunctionNet() [1/2]

```
CDigOutStimulatorFunctionNet (
    CMcsUsbNet^ mcsusb,
    CMcsUsbFunctionPointerContainer^ pDigOutStimulatorFunctionPointerContainer )
```

Initializes a new instance of the CDigOutStimulatorFunctionNet class.

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

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

11.28.2.4 "!CDigOutStimulatorFunctionNet() !CDigOutStimulatorFunctionNet ( )

11.28.3 Member Function Documentation

11.28.3.1 ClearChannel() void ClearChannel (int32_t *NrChannel*)

clear stimulation pattern

Parameters

NrChannel	the channel to clear

11.28.3.2 GetGlobalRepeat() bool GetGlobalRepeat (int32_t *NrChannel*)

get repeat whole stimulation pattern

Parameters

NrChannel	channel number

Returns

current value

11.28.3.3 GetNumberOfChannels() int32_t GetNumberOfChannels ( )

get the number of channels available on the device

Returns

the number of channels

11.28.3.4 GetStartTriggerSlope() DigitalStimulatorTriggerSlopeEnumNet GetStartTriggerSlope (int32_t *NrChannel*)

queries start condition of digital out stimulator

Parameters

NrChannel	channel number

Returns

start condition (rising or falling edge)

11.28.3.5 GetStopTriggerSlope() DigitalStimulatorTriggerSlopeEnumNet GetStopTriggerSlope (int32_t *NrChannel*)

queries stop condition of digital out stimulator

Parameters

NrChannel	channel number

Returns

stop condition (rising or falling edge)

11.28.3.6 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

Amplitude	array of amplitudes
Duration	array of durations

Returns

11.28.3.7 SendChannelData()

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

send or append stimulation pattern

Parameters

NrChannel	the channel to send data to
device_data_and_unrolled	internal, use value obtained from PrepareChannelData

11.28.3.8 SetGlobalRepeat()

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

set repeat whole stimulation pattern

Parameters

NrChannel	channel number
value	new value

11.28.3.9 SetStartTriggerSlope()

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

sets start condition of digital out stimulator

Parameters

NrChannel	channel number
Condition	start condition (rising or falling edge)

11.28.3.10 SetStopTriggerSlope()

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

sets stop condition of digital out stimulator

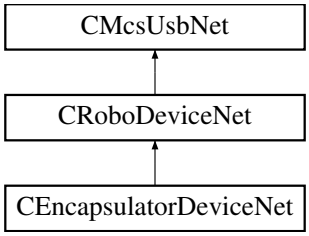
Parameters

NrChannel	channel number
Condition	stop condition (rising or falling edge)

11.29 CEncapsulatorDeviceNet Class Reference

CEncapsulatorDeviceNet is the to control the MCS HiClamp device

Inheritance diagram for CEncapsulatorDeviceNet:



Public Member Functions

- CEncapsulatorDeviceNet (void)
- CRoboFluidDeviceNet ^ GetRoboFluidDevice ()

Additional Inherited Members

11.29.1 Detailed Description

CEncapsulatorDeviceNet is the to control the MCS HiClamp device

11.29.2 Constructor & Destructor Documentation

11.29.2.1 CEncapsulatorDeviceNet() `CEncapsulatorDeviceNet ( void )`

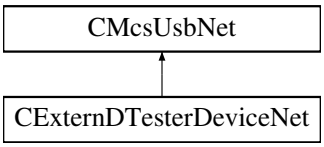
11.29.3 Member Function Documentation

11.29.3.1 GetRoboFluidDevice() `CRoboFluidDeviceNet ^ GetRoboFluidDevice ( )`

11.30 CExternDTesterDeviceNet Class Reference

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

Inheritance diagram for CExternDTesterDeviceNet:



Public Member Functions

- CExternDTesterDeviceNet ()

  Initializes a new instance of the CExternDTesterDeviceNet class.
- virtual ~CExternDTesterDeviceNet ()
- !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

11.30.1 Detailed Description

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

11.30.2 Constructor & Destructor Documentation

11.30.2.1 CExternDTesterDeviceNet() `CExternDTesterDeviceNet ( )`

Initializes a new instance of the CExternDTesterDeviceNet class.

11.30.2.2 ~CExternDTesterDeviceNet() `virtual ~CExternDTesterDeviceNet ( ) [virtual]`

11.30.2.3 "!CExternDTesterDeviceNet() `!CExternDTesterDeviceNet ( )`

11.30.3 Member Function Documentation

11.30.3.1 Read() `array<uint8_t> ^ Read ( int configString_Length )`

Reads the config string from the device.

Parameters

configString_Length	The maximal length of configString.

Returns

The config string.

11.30.3.2 Read2() `String ^ Read2 ( )`

Reads the config string from the device.

Returns

The config string.

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

Reads the config string from the device.

Parameters

configString	The config string.

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

Reads the config string from the device.

Parameters

configString	The config string.

11.31 CFilterCoefficientsNet Class Reference

Classes

- struct s_FilterAttributesNet

Public Member Functions

- CFilterCoefficientsNet ()
- CFilterCoefficientsNet (double b0, double b1, double b2, double a1, double a2)
- CFilterCoefficientsNet (double b0, double b1, double a1)
- CFilterCoefficientsNet (array< double >^ b, array< double >^ a)
- ~CFilterCoefficientsNet ()
- bool IsEqual (array< uint64_t >^ coefficients, s_FilterAttributesNet^ FiltAttr, bool DoMCSLegacyCompare)
- uint64_t GetUintB (int index, s_FilterAttributesNet^ FiltAttr)
- uint64_t GetUintA (int index, s_FilterAttributesNet^ FiltAttr)
- array< uint64_t > ^ GetUints (s_FilterAttributesNet^ FiltAttr)

Properties

- array< double >^ A [get]
- array< double >^ B [get]

11.31.1 Constructor & Destructor Documentation

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

11.31.1.2 CFilterCoefficientsNet() [2/4] CFilterCoefficientsNet (

```
double b0,
double b1,
double b2,
double a1,
double a2 )
```

11.31.1.3 CFilterCoefficientsNet() [3/4] CFilterCoefficientsNet (

```
double b0,
double b1,
double a1 )
```

11.31.1.4 CFilterCoefficientsNet() [4/4] CFilterCoefficientsNet (

```
array< double >^ b,
array< double >^ a )
```

11.31.1.5 ~CFilterCoefficientsNet() ~CFilterCoefficientsNet ( )

11.31.2 Member Function Documentation

11.31.2.1 GetUintA() uint64_t GetUintA (

```
int index,
s_FilterAttributesNet^ FiltAttr )
```

11.31.2.2 GetUintB() uint64_t GetUintB (

```
int index,
s_FilterAttributesNet^ FiltAttr )
```

11.31.2.3 GetUints() array<uint64_t> ^ GetUints (

```
s_FilterAttributesNet^ FiltAttr )
```

11.31.2.4 IsEqual() bool IsEqual (

```
array< uint64_t >^ coefficients,
s_FilterAttributesNet^ FiltAttr,
bool DoMCSLegacyCompare )
```

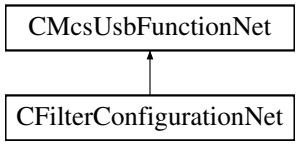
11.31.3 Property Documentation

11.31.3.1 A array< double>^ A [get]

11.31.3.2 B array< double>^ B [get]

11.32 CFilterConfigurationNet Class Reference

Inheritance diagram for CFilterConfigurationNet:



Public Member Functions

- CFilterConfigurationNet (CMcsUsbNet^ mcsusb)
- void SetFilterParameter (DacqGroupChannelEnumNet GroupID, uint32_t FilterNumber, CFilterCoefficientsNet^ Coefficients, CFilterPropertyNet^ FilterProp)
- void SetFilterParameter (DacqGroupChannelEnumNet GroupID, uint32_t FilterNumber, CFilterCoefficientsNet^ CoefficientsSet1, CFilterCoefficientsNet^ CoefficientsSet2, CFilterPropertyNet^ FilterProp)
- void SetFilterParameterPermanent (DacqGroupChannelEnumNet GroupID, uint32_t FilterNumber)
- void EraseFilterParameterPermanent (DacqGroupChannelEnumNet GroupID, uint32_t FilterNumber)
- void SetHighpassFilterEnable (bool enable)
- bool GetHighpassFilterEnable ()
- void ResetHighpassFilter ()
- uint32_t GetFilterAttributes (DacqGroupChannelEnumNet GroupID, uint32_t FilterNumber, FilterAttributeEnumNet index)
- CFilterCoefficientsNet::s_FilterAttributesNet ^ GetFilterAttributes (DacqGroupChannelEnumNet GroupID, uint32_t FilterNumber)

Static Public Member Functions

- static CFilterCoefficientsNet::s_FilterAttributesNet ^ GetDefaultFilterAttributes ()

Additional Inherited Members

11.32.1 Constructor & Destructor Documentation

11.32.1.1 CFilterConfigurationNet() `CFilterConfigurationNet (`

```
CMcsUsbNet^ mcsusb )
```

11.32.2 Member Function Documentation

11.32.2.1 EraseFilterParameterPermanent() `void EraseFilterParameterPermanent (`

```
DacqGroupChannelEnumNet GroupID,
uint32_t FilterNumber )
```

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

11.32.2.3 GetFilterAttributes() [1/2]

```
CFilterCoefficientsNet::s_FilterAttributesNet ^ GetFilterAttributes (
        DacqGroupChannelEnumNet GroupID,
        uint32_t FilterNumber )
```

11.32.2.4 GetFilterAttributes() [2/2]

```
uint32_t GetFilterAttributes (
        DacqGroupChannelEnumNet GroupID,
        uint32_t FilterNumber,
        FilterAttributeEnumNet index )
```

11.32.2.5 GetHighpassFilterEnable()

```
bool GetHighpassFilterEnable ( )
```

11.32.2.6 ResetHighpassFilter()

```
void ResetHighpassFilter ( )
```

11.32.2.7 SetFilterParameter() [1/2]

```
void SetFilterParameter (
        DacqGroupChannelEnumNet GroupID,
        uint32_t FilterNumber,
        CFilterCoefficientsNet^ Coefficients,
        CFilterPropertyNet^ FilterProp )
```

11.32.2.8 SetFilterParameter() [2/2]

```
void SetFilterParameter (
        DacqGroupChannelEnumNet GroupID,
        uint32_t FilterNumber,
        CFilterCoefficientsNet^ CoefficientsSet1,
        CFilterCoefficientsNet^ CoefficientsSet2,
        CFilterPropertyNet^ FilterProp )
```

11.32.2.9 SetFilterParameterPermanent()

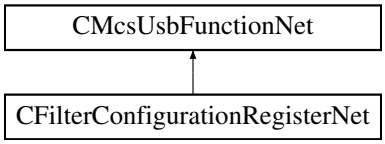
```
void SetFilterParameterPermanent (
        DacqGroupChannelEnumNet GroupID,
        uint32_t FilterNumber )
```

11.32.2.10 SetHighpassFilterEnable()

```
void SetHighpassFilterEnable (
        bool enable )
```

11.33 CFilterConfigurationRegisterNet Class Reference

Inheritance diagram for CFilterConfigurationRegisterNet:



Public Member Functions

- CFilterConfigurationRegisterNet (CMcsUsbNet^ mcsusb)
- void SetFilterParameter (uint32_t FilterCoefRegBase, CFilterCoefficientsNet^ Coefficients, uint32_t FilterInfoRegBase, CFilterPropertyNet^ FilterProp)
- void SetFilterParameter (uint32_t FilterCoefSet1RegBase, CFilterCoefficientsNet^ CoefficientsSet1, uint32_t FilterCoefSet2RegBase, CFilterCoefficientsNet^ CoefficientsSet2, uint32_t FilterInfoRegBase, CFilterPropertyNet^ FilterProp)
- void SetFilterParameterPermanent (uint32_t FilterCoefRegBase, uint32_t FilterCoefDmaReg, uint32_t FilterInfoRegBase, uint32_t FilterInfoDmaReg, uint32_t EEPROMBase, uint32_t EEPROMSize)
- void SetFilterParameterPermanent (uint32_t FilterCoefSet1RegBase, uint32_t FilterCoefSet1DmaReg, uint32_t FilterCoefSet2RegBase, uint32_t FilterCoefSet2DmaReg, uint32_t FilterInfoRegBase, uint32_t FilterInfoDmaReg, uint32_t EEPROMBase, uint32_t EEPROMSize)
- void EraseFilterParameterPermanent (uint32_t FilterCoefDmaReg, uint32_t FilterInfoDmaReg, uint32_t EEPROMBase, uint32_t EEPROMSize)
- void EraseFilterParameterPermanent (uint32_t FilterCoefSet1DmaReg, uint32_t FilterCoefSet2DmaReg, uint32_t FilterInfoDmaReg, uint32_t EEPROMBase, uint32_t EEPROMSize)

Additional Inherited Members

11.33.1 Constructor & Destructor Documentation

11.33.1.1 CFilterConfigurationRegisterNet()

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

11.33.2 Member Function Documentation

11.33.2.1 EraseFilterParameterPermanent() [1/2]

```
void EraseFilterParameterPermanent (
    uint32_t FilterCoefDmaReg,
    uint32_t FilterInfoDmaReg,
    uint32_t EEPROMBase,
    uint32_t EEPROMSize )
```

11.33.2.2 EraseFilterParameterPermanent() [2/2]

```
void EraseFilterParameterPermanent (
    uint32_t FilterCoefSet1DmaReg,
    uint32_t FilterCoefSet2DmaReg,
    uint32_t FilterInfoDmaReg,
    uint32_t EEPROMBase,
    uint32_t EEPROMSize )
```

11.33.2.3 SetFilterParameter() [1/2]

```
void SetFilterParameter (
    uint32_t FilterCoefRegBase,
    CFilterCoefficientsNet^ Coefficients,
    uint32_t FilterInfoRegBase,
    CFilterPropertyNet^ FilterProp )
```

11.33.2.4 SetFilterParameter() [2/2]

```
void SetFilterParameter (
    uint32_t FilterCoefSet1RegBase,
    CFilterCoefficientsNet^ CoefficientsSet1,
    uint32_t FilterCoefSet2RegBase,
    CFilterCoefficientsNet^ CoefficientsSet2,
    uint32_t FilterInfoRegBase,
    CFilterPropertyNet^ FilterProp )
```

11.33.2.5 SetFilterParameterPermanent() [1/2]

```
void SetFilterParameterPermanent (
    uint32_t FilterCoefRegBase,
    uint32_t FilterCoefDmaReg,
    uint32_t FilterInfoRegBase,
    uint32_t FilterInfoDmaReg,
    uint32_t EEPROMBase,
    uint32_t EEPROMSize )
```

11.33.2.6 SetFilterParameterPermanent() [2/2]

```
void SetFilterParameterPermanent (
    uint32_t FilterCoefSet1RegBase,
    uint32_t FilterCoefSet1DmaReg,
    uint32_t FilterCoefSet2RegBase,
    uint32_t FilterCoefSet2DmaReg,
    uint32_t FilterInfoRegBase,
    uint32_t FilterInfoDmaReg,
    uint32_t EEPROMBase,
    uint32_t EEPROMSize )
```

11.34 CFilterPropertyNet Class Reference

Public Member Functions

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

Properties

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

11.34.1 Constructor & Destructor Documentation

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

11.34.1.2 ~CFilterPropertyNet() ~CFilterPropertyNet ( )

11.34.2 Member Function Documentation

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

11.34.3 Property Documentation

11.34.3.1 CornerFrequency double CornerFrequency [get]

11.34.3.2 CornerFrequencymHz uint32_t CornerFrequencymHz [get]

11.34.3.3 FilterActive `bool FilterActive [get]`

11.34.3.4 FilterBand `FilterBandEnumNet FilterBand [get]`

11.34.3.5 FilterFamily `FilterFamilyEnumNet FilterFamily [get]`

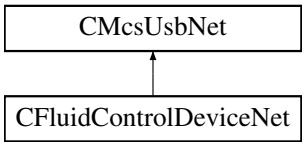
11.34.3.6 FilterType `FilterTypeEnumNet FilterType [get]`

11.34.3.7 Order `uint32_t Order [get]`

11.35 CFluidControlDeviceNet Class Reference

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

Inheritance diagram for CFluidControlDeviceNet:



Public Member Functions

- CFluidControlDeviceNet ()

  Initialize a new instance of the CFluidControlDeviceNet class.
- ~CFluidControlDeviceNet ()

  Default destructor.
- void SetValve (unsigned int value)

  Open or Close valves.
- void SetSingleValve (unsigned short valve, unsigned short onoff)

  Opens or Closes a valve.
- void SetDigout (unsigned int value)

  Define the pattern on the Digital Output.
- void SetPWM (unsigned int channel, unsigned int value)

  Sets the duty cycle of the PWM output.
- void CalibrateThermocouple (unsigned int channel)

  Calibrates the ADC which is used for the Thermocouple. For the calibration, Short circuit the Thermocouple and use this function to correct a possible offset of the ADC which measures the thermocouple.
- void SetThermocoupleNanovoltPerKelvin (unsigned int channel, unsigned int value)

Sets the proportinal constant for the Thermocouple.

- unsigned int GetValve ()

  Gets the state of the valves.
- unsigned short GetSingleValve (unsigned short valve)

  Gets the state of a valve.
- unsigned int GetDigout ()

  Gets the state of the digital output.
- unsigned int GetPWM (unsigned int channel)

  Gets the state of the PWM output.
- unsigned int GetAdc (unsigned int channel)

  Reads an ADC Value.
- unsigned int GetDigin ()

  Reads the digital input.
- int GetThermocoupleTemperature (unsigned int channel)

  Reads the temperature from Thermocouple. The functions gives the temperature difference between both Thermocouple junctions. To get the absolute temperature, add the reference temperature.
- int GetReferenceTemperature (unsigned int channel)

  Reads the reference temperature for the Thermocouple.
- unsigned int GetThermocoupleCalibration (unsigned int channel)

  Gets the calibration constant for the Thermocouple ADC.
- unsigned int GetThermocoupleNanovoltPerKelvin (unsigned int channel)

  Reads the proportional constant for the Thermocouple.

Properties

- CMcsBus_VoltageModeNet^ McsBus_VoltageMode `[get]`

Additional Inherited Members

11.35.1 Detailed Description

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

11.35.2 Constructor & Destructor Documentation

11.35.2.1 CFluidControlDeviceNet() `CFluidControlDeviceNet ( )`

Initialize a new instance of the CFluidControlDeviceNet class.

11.35.2.2 ~CFluidControlDeviceNet() `~CFluidControlDeviceNet ( )`

Default destructor.

11.35.3 Member Function Documentation

11.35.3.1 CalibrateThermocouple() `void CalibrateThermocouple ( unsigned int channel )`

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

Parameters

channel	Thermocouple channel number.

11.35.3.2 GetAdc() `unsigned int GetAdc ( unsigned int channel )`

Reads an ADC Value.

Parameters

channel	The ADC channel number to query.

Returns

The current ADC value.

11.35.3.3 GetDigin() `unsigned int GetDigin ( )`

Reads the digital input.

Returns

The bit pattern of the state of the digital inputs.

11.35.3.4 GetDigout() `unsigned int GetDigout ( )`

Gets the state of the digital output.

Returns

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

11.35.3.5 GetPWM() `unsigned int GetPWM ( unsigned int channel )`

Gets the state of the PWM output.

Returns

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

11.35.3.6 GetReferenceTemperature() `int GetReferenceTemperature ( unsigned int channel )`

Reads the reference temperature for the Thermocouple.

Parameters

channel	Thermocouple channel number.

Returns

The temperature from the Thermocouple in 1/100 ℃.

11.35.3.7 GetSingleValve() `unsigned short GetSingleValve ( unsigned short valve )`

Gets the state of a valve.

Parameters

valve	number of valve

Returns

state of the valve

11.35.3.8 GetThermocoupleCalibration() `unsigned int GetThermocoupleCalibration ( unsigned int channel )`

Gets the calibration constant for the Thermocouple ADC.

Parameters

channel	Thermocouple channel number.

Returns

The calibration constant for the Thermocouple ADC.

11.35.3.9 GetThermocoupleNanovoltPerKelvin() `unsigned int GetThermocoupleNanovoltPerKelvin ( unsigned int channel )`

Reads the proportional constant for the Thermocouple.

Parameters

channel	Thermocouple channel number.

Returns

The proportional constant in Nanovolt per Kelvin.

11.35.3.10 GetThermocoupleTemperature() `int GetThermocoupleTemperature ( unsigned int channel )`

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

Parameters

channel	Thermocouple channel number.

Returns

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

11.35.3.11 GetValve() `unsigned int GetValve ( )`

Gets the state of the valves.

Returns

The current state of the valves as a bit pattern.

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

Define the pattern on the Digital Output.

Parameters

value	bit pattern on the digital output.

11.35.3.13 SetPWM() `void SetPWM ( unsigned int channel, unsigned int value )`

Sets the duty cycle of the PWM output.

Parameters

channel	PWM channel number.
value	duty cycle of the PWM output in permille.

11.35.3.14 SetSingleValve() `void SetSingleValve (`

```
unsigned short valve,
unsigned short onoff )
```

Opens or Closes a valve.

Parameters

valve	number of valve to be changed.

Parameters

onoff	open or close the valve.

11.35.3.15 SetThermocoupleNanovoltPerKelvin() `void SetThermocoupleNanovoltPerKelvin (`

```
unsigned int channel,
unsigned int value )
```

Sets the proportinal constant for the Thermocouple.

Parameters

channel	Thermocouple channel number.
value	proportinal constant for the Thermocouple in Nanovolt per Kelvin.

11.35.3.16 SetValve() `void SetValve (`

```
unsigned int value )
```

Open or Close valves.

Parameters

value	bit pattern of valves which should be open.

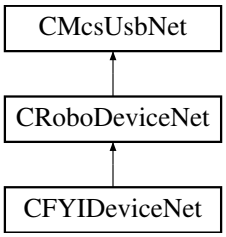
11.35.4 Property Documentation

11.35.4.1 McsBus_VoltageMode `CMcsBus_VoltageModeNet^ McsBus_VoltageMode [get]`

11.36 CFYIDeviceNet Class Reference

CFYIDeviceNet is the class to control the MCS FYI device

Inheritance diagram for CFYIDeviceNet:



Public Member Functions

- CFYIDeviceNet (void)

Properties

- CRobo_FYITemp_FunctionNet^ FYITemp `[get]`
- CRobo_FYIProgram_FunctionNet^ FYIProgram `[get]`
- CMcsBus_SensorNet^ Sensor `[get]`

Additional Inherited Members

11.36.1 Detailed Description

CFYIDeviceNet is the class to control the MCS FYI device

11.36.2 Constructor & Destructor Documentation

11.36.2.1 CFYIDeviceNet() `CFYIDeviceNet (`

```
void )
```

11.36.3 Property Documentation

11.36.3.1 FYIProgram CRobo_FYIProgram_FunctionNet^ FYIProgram [get]

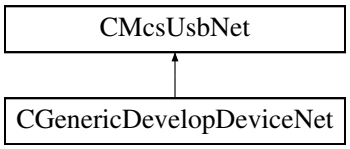
11.36.3.2 FYITemp CRobo_FYITemp_FunctionNet^ FYITemp [get]

11.36.3.3 Sensor CMcsBus_SensorNet^ Sensor [get]

11.37 CGenericDevelopDeviceNet Class Reference

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

Inheritance diagram for CGenericDevelopDeviceNet:



Public Member Functions

- CGenericDevelopDeviceNet (void)

  Initialize a new instance of the CGenericDevelopDeviceNet class.
- ~CGenericDevelopDeviceNet (void)
- void SetValue (uint16_t value, uint16_t index)

  Sets .

Parameters

value	*The value of the request.*

Parameters

index	*The index of the request.*

- 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	*The value of the request.*

Parameters

index	*The index of the request.*

Parameters

buffer	*The buffer to send.*

- void SetByteBuffer (uint16_t value, uint16_t Index, array< char >^ buffer)

  Sends an array of type char to the device.

Parameters

value	*The value of the request.*

Parameters

Index	*The index of the request.*

Parameters

buffer	*The buffer to send.*

- void SetUShortBuffer (uint16_t value, uint16_t index, array< unsigned short >^ buffer)

  Sends an array of type unsigned short to the device.

Parameters

value	*The value of the request.*

Parameters

index	*The index of the request.*

Parameters

buffer	*The buffer to send.*

- void SetShortBuffer (uint16_t value, uint16_t index, array< short >^ buffer)

  Sends an array of type short to the device.

Parameters

value	*The value of the request.*

Parameters

index	*The index of the request.*

Parameters

buffer	*The buffer to send.*

- void SetUIntBuffer (uint16_t value, uint16_t index, array< unsigned int >^ buffer)

  Sends an array of unsigned int to the device.

Parameters

value	*The value of the request.*

Parameters

index	*The index of the request.*

Parameters

buffer	*The buffer to send.*

- void SetIntBuffer (uint16_t value, uint16_t index, array< int >^ buffer)

  Sends an array of type int to the device.

Parameters

value	*The value of the request.*

Parameters

index	*The index of the request.*

Parameters

buffer	*The buffer to send.*

- 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	*The value of the request.*

Parameters

index	*The index of the request.*

Parameters

size	*The size of the array.*

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	*The value of the request.*

Parameters

index	*The index of the request.*

Parameters

size	*The size of the array.*

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	*The value of the request.*

Parameters

index	*The index of the request.*

Parameters

size	*The size of the array.*

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	*The value of the request.*

Parameters

index	*The index of the request.*

Parameters

size	*The size of the array.*

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	*The value of the request.*

Parameters

index	*The index of the request.*

Parameters

size	*The size of the array.*

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	*The value of the request.*

Parameters

index	*The index of the request.*

Parameters

size	*The size of the array.*

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	*The Endpoint Number.*

Returns

A handle to the endpoint.

- void ClosePipe (IntPtr pipeHandle)

  Close a Pipe to an USB Endpoint.

Parameters

pipeHandle	*The endpoint handle.*

- void ResetPipe (IntPtr pipeHandle)

  Reset a Pipe to an USB Endpoint.

Parameters

pipeHandle	*The endpoint handle.*

- template<typename C >
  array< C > ^ ReadPipe (IntPtr pipeHandle, uint32_t size)

  Read data from an USB Endpoint.

Parameters

pipeHandle	*The endpoint handle.*

Parameters

size	*Number of elements to read.*

Returns

An array of data read.

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

Write data to an USB Endpoint.

Parameters

pipeHandle	*The endpoint handle.*

Parameters

buffer	*An array of data to write.*

Additional Inherited Members

11.37.1 Detailed Description

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

11.37.2 Constructor & Destructor Documentation

11.37.2.1 CGenericDevelopDeviceNet() `CGenericDevelopDeviceNet ( void )`

Initialize a new instance of the CGenericDevelopDeviceNet class.

11.37.2.2 ~CGenericDevelopDeviceNet() `~CGenericDevelopDeviceNet ( void )`

11.37.3 Member Function Documentation

11.37.3.1 ClosePipe() `void ClosePipe ( IntPtr pipeHandle )`

Close a Pipe to an USB Endpoint.

Parameters

pipeHandle	The endpoint handle.

11.37.3.2 FindEndpoints()

```
array<uint8_t> ^ FindEndpoints (
    uint8_t type,
    uint8_t direction )
```

11.37.3.3 GetBuffer()

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

11.37.3.4 GetByteBuffer()

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

Gets an array of type char from the device.

Parameters

value	The value of the request.

Parameters

index	The index of the request.

Parameters

size	The size of the array.

Returns

The array of data from the device.

11.37.3.5 GetIntBuffer() array<int> ^ GetIntBuffer (

```
uint16_t value,
uint16_t index,
int size )
```

Gets an array of type int from the device.

Parameters

value	The value of the request.

Parameters

index	The index of the request.

Parameters

size	The size of the array.

Returns

The array of data from the device.

11.37.3.6 GetShortBuffer() array<short> ^ GetShortBuffer (

```
uint16_t value,
uint16_t index,
int size )
```

Gets an array of type short from the device.

Parameters

value	The value of the request.

Parameters

index	The index of the request.

Parameters

size	The size of the array.

Returns

The array of data from the device.

11.37.3.7 GetUByteBuffer()

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

Gets an array of type unsigned char from the device.

Parameters

value	The value of the request.

Parameters

index	The index of the request.

Parameters

size	The size of the array.

Returns

The array of data from the device.

11.37.3.8 GetUIntBuffer() array<unsigned int> ^ GetUIntBuffer (

```
uint16_t value,
uint16_t index,
int size )
```

Gets an array of type unsigned int from the device.

Parameters

value	The value of the request.

Parameters

index	The index of the request.

Parameters

size	The size of the array.

Returns

The array of data from the device.

11.37.3.9 GetUShortBuffer() array<unsigned short> ^ GetUShortBuffer (

```
uint16_t value,
uint16_t index,
int size )
```

Gets an array of type unsigned short from the device.

Parameters

value	The value of the request.

Parameters

index	The index of the request.

Parameters

size	The size of the array.

Returns

The array of data from the device.

11.37.3.10 OpenPipe() `IntPtr OpenPipe ( uint8_t endpointAddress )`

Open a Pipe to an USB Endpoint.

Parameters

endpointAddress	The Endpoint Number.

Returns

A handle to the endpoint.

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

Read data from an USB Endpoint.

Parameters

pipeHandle	The endpoint handle.

Parameters

size	Number of elements to read.

Returns

An array of data read.

11.37.3.12 ResetPipe()

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

Reset a Pipe to an USB Endpoint.

Parameters

pipeHandle	The endpoint handle.

11.37.3.13 SetBuffer()

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

11.37.3.14 SetByteBuffer()

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

Sends an array of type char to the device.

Parameters

value	The value of the request.

Parameters

Index	The index of the request.

Parameters

buffer	The buffer to send.

11.37.3.15 SetIntBuffer()

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

Sends an array of type int to the device.

Parameters

value	The value of the request.

Parameters

index	The index of the request.

Parameters

buffer	The buffer to send.

11.37.3.16 SetShortBuffer()

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

Sends an array of type short to the device.

Parameters

value	The value of the request.

Parameters

index	The index of the request.

Parameters

buffer	The buffer to send.

11.37.3.17 SetUByteBuffer()

```
void SetUByteBuffer (
    uint16_t value,
    uint16_t index,
    array< unsigned char >^ buffer )
```

Sends an array of type unsigned char to the device.

Parameters

value	The value of the request.

Parameters

index	The index of the request.

Parameters

buffer	The buffer to send.

11.37.3.18 SetUIntBuffer()

```
void SetUIntBuffer (
    uint16_t value,
```

```
            uint16_t index,
            array< unsigned int >^ buffer )
```

Sends an array of unsigned int to the device.

Parameters

value	The value of the request.

Parameters

index	The index of the request.

Parameters

buffer	The buffer to send.

11.37.3.19 SetUShortBuffer() `void SetUShortBuffer (`

```
            uint16_t value,
            uint16_t index,
            array< unsigned short >^ buffer )
```

Sends an array of type unsigned short to the device.

Parameters

value	The value of the request.

Parameters

index	The index of the request.

Parameters

buffer	The buffer to send.

11.37.3.20 SetValue() `void SetValue (`

```
uint16_t value,
uint16_t index )
```

Sets .

Parameters

value	The value of the request.

Parameters

index	The index of the request.

11.37.3.21 VendorInRequest() `array<C> ^ VendorInRequest (`

```
uint8_t request,
uint16_t value,
uint16_t index,
int size )
```

11.37.3.22 VendorOutRequest() `void VendorOutRequest (`

```
uint8_t request,
uint16_t value,
uint16_t index,
array< C >^ buffer )
```

11.37.3.23 WritePipe() `void WritePipe (`

```
IntPtr pipeHandle,
array< C >^ buffer )
```

Write data to an USB Endpoint.

Parameters

pipeHandle	The endpoint handle.

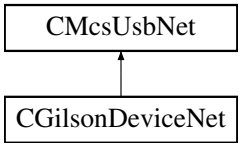
Parameters

buffer	An array of data to write.

11.38 CGilsonDeviceNet Class Reference

CGilsonDeviceNet is the class to control a Gilson device.

Inheritance diagram for CGilsonDeviceNet:



Public Member Functions

- CGilsonDeviceNet (void)

  Initialize a new instance of the CGilsonDeviceNet class.
- ~CGilsonDeviceNet (void)
- void ConnectSlave (byte ID)
- void SendImmediate (wchar_t command)
- String ^ SendImmediateGetResponse (wchar_t command)
- void SendBuffered (String^ command)
- String ^ GetLastAnswer ()

Protected Attributes

- CGilsonDevice * m_pGilsonDevice

Additional Inherited Members

11.38.1 Detailed Description

CGilsonDeviceNet is the class to control a Gilson device.

11.38.2 Constructor & Destructor Documentation

11.38.2.1 CGilsonDeviceNet() `CGilsonDeviceNet (`

`void )`

Initialize a new instance of the CGilsonDeviceNet class.

11.38.2.2 ~CGilsonDeviceNet() `~CGilsonDeviceNet (`

`void )`

11.38.3 Member Function Documentation

11.38.3.1 ConnectSlave() `void ConnectSlave (`

`byte` *`ID`* `)`

11.38.3.2 GetLastAnswer() `String ^ GetLastAnswer ( )`

11.38.3.3 SendBuffered() `void SendBuffered (`

`String^` *`command`* `)`

11.38.3.4 SendImmediate() `void SendImmediate (`

`wchar_t` *`command`* `)`

11.38.3.5 SendImmediateGetResponse() `String ^ SendImmediateGetResponse (`

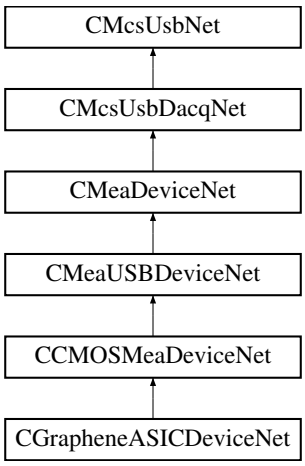
`wchar_t` *`command`* `)`

11.38.4 Member Data Documentation

11.38.4.1 m_pGilsonDevice `CGilsonDevice* m_pGilsonDevice [protected]`

11.39 CGrapheneASICDeviceNet Class Reference

Inheritance diagram for CGrapheneASICDeviceNet:



Public Member Functions

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

Additional Inherited Members

11.39.1 Constructor & Destructor Documentation

11.39.1.1 CGrapheneASICDeviceNet() `CGrapheneASICDeviceNet ( void )`

11.39.1.2 ~CGrapheneASICDeviceNet() `~CGrapheneASICDeviceNet ( )`

11.39.2 Member Function Documentation

11.39.2.1 GetAvailableBaseSamplerates()

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

Reimplemented from CCMOSMeaDeviceNet.

11.39.2.2 GetBaseSamplerate()

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

Reimplemented from CCMOSMeaDeviceNet.

11.39.2.3 SetBaseSamplerate()

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

Reimplemented from CCMOSMeaDeviceNet.

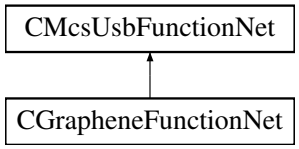
11.39.2.4 SetRegionOfInterests()

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

11.40 CGrapheneFunctionNet Class Reference

CGrapheneFunctionNet is the class to control Graphene device functions

Inheritance diagram for CGrapheneFunctionNet:



Public Member Functions

- CGrapheneFunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ pGraphene↩ FunctionPointerContainer)

  Initializes a new instance of the CGrapheneFunctionNet class.
- CGrapheneFunctionNet (CMcsUsbNet^ mcsusb)
- virtual ~CGrapheneFunctionNet ()
- !CGrapheneFunctionNet ()
- void GetVdVsDAC ([System::Runtime::InteropServices::Out]int16_t% Vd, [System::Runtime::Interop↩ Services::Out]int16_t% Vs)

  Gets Vd and Vs
- void GetVdVsDAC (uint32_t Headstage, [System::Runtime::InteropServices::Out]int16_t% Vd, [System::↩ Runtime::InteropServices::Out]int16_t% Vs)

  Gets Vd and Vs

- void SetVdVsDAC (int16_t Vd, int16_t Vs)

  Sets Vd and Vs
- void SetVdVsDAC (uint32_t Headstage, int16_t Vd, int16_t Vs)

  Sets Vd and VS
- bool GetVoltageReached ()

  Gets the reached voltage
- bool GetVoltageReached (uint32_t Headstage)

  Gets the reached voltage
- int32_t GetVoltageRange ()

  Gets the voltage range
- int32_t GetVoltageRange (uint32_t Headstage)

  Gets the voltage range
- void SetVoltageRange (int32_t range)

  Sets the voltage range
- void SetVoltageRange (uint32_t Headstage, int32_t range)

  Sets the voltage range
- int32_t GetVoltageResolution ()

  Gets the voltage resolution
- int32_t GetVoltageResolution (uint32_t Headstage)

  Gets the voltage resolution
- void SetVoltageResolution (int32_t resolution)

  Sets the voltage resolution
- void SetVoltageResolution (uint32_t Headstage, int32_t resolution)

  Sets the voltage resolution
- void GetDACOffset ([System::Runtime::InteropServices::Out]int16_t% offset_vd, [System::Runtime::↩InteropServices::Out]int16_t% offset_vs)

  Gets the DAC offset
- void GetDACOffset (uint32_t Headstage, [System::Runtime::InteropServices::Out]int16_t% offset_vd, [System::Runtime::InteropServices::Out]int16_t% offset_vs)

  Gets the DAC offset
- void SetDACOffset (int16_t offset_vd, int16_t offset_vs)

  Sets the DAC offset
- void SetDACOffset (uint32_t Headstage, int16_t offset_vd, int16_t offset_vs)

  Set the DAC offset
- void GetVdVs ([System::Runtime::InteropServices::Out]int32_t% Vd, [System::Runtime::InteropServices::↩Out]int32_t% Vs)

  Gets Vd and Vs
- void GetVdVs (uint32_t Headstage, [System::Runtime::InteropServices::Out]int32_t% Vd, [System::↩Runtime::InteropServices::Out]int32_t% Vs)

  Gets Vd and Vs
- void SetVdVs (int32_t Vd, int32_t Vs)

  Sets Vd and Vs
- void SetVdVs (uint32_t Headstage, int32_t Vd, int32_t Vs)

  Sets Vd and Vs
- void SetVdsVgs (int32_t Vds, int32_t Vgs)

  Sets Vds and Vgs
- void SetVdsVgs (uint32_t Headstage, int32_t Vds, int32_t Vgs)

  Sets Vds and Vgs
- int32_t GetCur2VolResistance ()

  Gets the resistance of the current to voltage converter
- int32_t GetCur2VolResistance (uint32_t Headstage)

Gets the resistance of the current to voltage converter

- int32_t GetADCOffset (uint32_t Channel)

  Gets the ADC offset
- int32_t GetADCOffset (uint32_t Headstage, uint32_t Channel)

  Gets the ADC offset
- void SetADCOffset (uint32_t Channel, int32_t offset)

  Sets the ADC offset
- void SetADCOffset (uint32_t Headstage, uint32_t Channel, int32_t offset)

  Sets the ADC offset
- int32_t GetADCGain (uint32_t Channel)

  Gets the ADC offset
- int32_t GetADCGain (uint32_t Headstage, uint32_t Channel)

  Gets the ADC offset
- void SetADCGain (uint32_t Channel, int32_t gain)

  Sets the ADC offset
- void SetADCGain (uint32_t Headstage, uint32_t Channel, int32_t gain)

  Sets the ADC offset
- void GetVdsVgs ([System::Runtime::InteropServices::Out]int32_t% Vds, [System::Runtime::Interop↩ Services::Out]int32_t% Vgs)

  Gets Vds and Vgs
- void GetVdsVgs (uint32_t Headstage, [System::Runtime::InteropServices::Out]int32_t% Vds, [System::↩ Runtime::InteropServices::Out]int32_t% Vgs)

  Gets Vds and Vgs

Additional Inherited Members

11.40.1 Detailed Description

CGrapheneFunctionNet is the class to control Graphene device functions

11.40.2 Constructor & Destructor Documentation

11.40.2.1 CGrapheneFunctionNet() [1/2]

```
CGrapheneFunctionNet (
    CMcsUsbNet^ mcsusb,
    CMcsUsbFunctionPointerContainer^ pGrapheneFunctionPointerContainer )
```

Initializes a new instance of the CGrapheneFunctionNet class.

11.40.2.2 CGrapheneFunctionNet() [2/2]

```
CGrapheneFunctionNet (
    CMcsUsbNet^ mcsusb )
```

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

11.40.2.4 "!CGrapheneFunctionNet() !CGrapheneFunctionNet ( )

11.40.3 Member Function Documentation

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

Gets the ADC offset

Parameters

Channel	The channel to query.

Returns

The gain

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

Gets the ADC offset

Parameters

Headstage	The headstage to query.
Channel	The channel to query.

Returns

The gain

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

Gets the ADC offset

Parameters

Channel	The channel to query.

Returns

The offset

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

Gets the ADC offset

Parameters

Headstage	The headstage to query.
Channel	The channel to query.

Returns

The offset

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

Gets the resistance of the current to voltage converter

Returns

The resistance in Ohm

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

Gets the resistance of the current to voltage converter

Parameters

Headstage	The headstage to query.

Returns

The resistance in Ohm

11.40.3.7 GetDACOffset() [1/2]

```
void GetDACOffset (
    [System::Runtime::InteropServices::Out] int16_t% offset_vd,
    [System::Runtime::InteropServices::Out] int16_t% offset_vs )
```

Gets the DAC offset

Parameters

offset_vd	Vd offset in DAC Units
offset_vs	Vs offset in DAC Units

11.40.3.8 GetDACOffset() [2/2]

```
void GetDACOffset (
    uint32_t Headstage,
    [System::Runtime::InteropServices::Out] int16_t% offset_vd,
    [System::Runtime::InteropServices::Out] int16_t% offset_vs )
```

Gets the DAC offset

Parameters

Headstage	The headstage to query.
offset_vd	Vd offset in DAC Units
offset_vs	Vs offset in DAC Units

11.40.3.9 GetVdsVgs() [1/2]

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

Gets Vds and Vgs

Parameters

Vds	Vds in µV
Vgs	Vgs in µV

11.40.3.10 GetVdsVgs() [2/2]

```
void GetVdsVgs (
    uint32_t Headstage,
```

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

Gets Vds and Vgs

Parameters

Headstage	The headstage to query.
Vds	Vds in µV
Vgs	Vgs in µV

11.40.3.11 GetVdVs() [1/2] void GetVdVs (

```
[System::Runtime::InteropServices::Out] int32_t% Vd,
[System::Runtime::InteropServices::Out] int32_t% Vs )
```

Gets Vd and Vs

Parameters

Vd	Vd in µV
Vs	Vs in µV

11.40.3.12 GetVdVs() [2/2] void GetVdVs (

```
uint32_t Headstage,
[System::Runtime::InteropServices::Out] int32_t% Vd,
[System::Runtime::InteropServices::Out] int32_t% Vs )
```

Gets Vd and Vs

Parameters

Headstage	The headstage to query.
Vd	Vd in µV
Vs	Vs in µV

11.40.3.13 GetVdVsDAC() [1/2] void GetVdVsDAC (

```
[System::Runtime::InteropServices::Out] int16_t% Vd,
[System::Runtime::InteropServices::Out] int16_t% Vs )
```

Gets Vd and Vs

Parameters

Vd	Vd in DAC Units
Vs	Vs in DAC Units

11.40.3.14 GetVdVsDAC() [2/2] void GetVdVsDAC (

```
uint32_t Headstage,
[System::Runtime::InteropServices::Out] int16_t% Vd,
[System::Runtime::InteropServices::Out] int16_t% Vs )
```

Gets Vd and Vs

Parameters

Headstage	The headstage to query.
Vd	Vd in DAC Units
Vs	Vs in DAC Units

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

Gets the voltage range

Returns

The voltage range in mV

11.40.3.16 GetVoltageRange() [2/2] int32_t GetVoltageRange (

```
uint32_t Headstage )
```

Gets the voltage range

Parameters

Headstage	The headstage to query.

Returns

The voltage range in mV

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

Gets the reached voltage

Returns

the reached voltage

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

Gets the reached voltage

Parameters

Headstage	The headstage to query.

Returns

The reached voltage

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

Gets the voltage resolution

Returns

The voltage resolution in µV/digit

11.40.3.20 GetVoltageResolution() [2/2] int32_t GetVoltageResolution (uint32_t *Headstage*)

Gets the voltage resolution

Parameters

Headstage	The headstage to query.

Returns

The voltage resolution in µV/digit

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

Sets the ADC offset

Parameters

Channel	The channel to query.
gain	The gain

11.40.3.22 SetADCGain() [2/2] void SetADCGain (

```
uint32_t Headstage,
uint32_t Channel,
int32_t gain )
```

Sets the ADC offset

Parameters

Headstage	The headstage to query.
Channel	The channel to query.
gain	The gain

11.40.3.23 SetADCOffset() [1/2] void SetADCOffset (

```
uint32_t Channel,
int32_t offset )
```

Sets the ADC offset

Parameters

Channel	The channel to query.
offset	The offset

11.40.3.24 SetADCOffset() [2/2] void SetADCOffset (

```
uint32_t Headstage,
uint32_t Channel,
int32_t offset )
```

Sets the ADC offset

Parameters

Headstage	The headstage to query.
Channel	The channel to query.
offset	The offset

11.40.3.25 SetDACOffset() [1/2] void SetDACOffset (

```
int16_t offset_vd,
int16_t offset_vs )
```

Sets the DAC offset

Parameters

offset_vd	Vd offset in DAC Units
offset_vs	Vs offset in DAC Units

11.40.3.26 SetDACOffset() [2/2]

```
void SetDACOffset (
    uint32_t Headstage,
    int16_t offset_vd,
    int16_t offset_vs )
```

Set the DAC offset

Parameters

Headstage	The headstage to query.
offset_vd	Vd offset in DAC Units
offset_vs	Vs offset in DAC Units

11.40.3.27 SetVdsVgs() [1/2]

```
void SetVdsVgs (
    int32_t Vds,
    int32_t Vgs )
```

Sets Vds and Vgs

Parameters

Vds	Vds in µV
Vgs	Vgs in µV

11.40.3.28 SetVdsVgs() [2/2]

```
void SetVdsVgs (
    uint32_t Headstage,
    int32_t Vds,
    int32_t Vgs )
```

Sets Vds and Vgs

Parameters

Headstage	The headstage to query.
Vds	Vds in µV
Vgs	Vgs in µV

11.40.3.29 SetVdVs() [1/2] void SetVdVs (

```
int32_t Vd,
int32_t Vs )
```

Sets Vd and Vs

Parameters

Vd	Vd in µV
Vs	Vs in µV

11.40.3.30 SetVdVs() [2/2] void SetVdVs (

```
uint32_t Headstage,
int32_t Vd,
int32_t Vs )
```

Sets Vd and Vs

Parameters

Headstage	The headstage to query.
Vd	Vd in µV
Vs	Vs in µV

11.40.3.31 SetVdVsDAC() [1/2] void SetVdVsDAC (

```
int16_t Vd,
int16_t Vs )
```

Sets Vd and Vs

Parameters

Vd	Vd in DAC Units
Vs	Vs in DAC Units

11.40.3.32 SetVdVsDAC() [2/2] void SetVdVsDAC (

```
uint32_t Headstage,
int16_t Vd,
int16_t Vs )
```

Sets Vd and VS

Parameters

Headstage	The headstage to query.
Vd	Vd in DAC Units
Vs	Vs in DAC Units

11.40.3.33 SetVoltageRange() [1/2]

```
void SetVoltageRange (
    int32_t range )
```

Sets the voltage range

Parameters

range	The voltage range in mV

11.40.3.34 SetVoltageRange() [2/2]

```
void SetVoltageRange (
    uint32_t Headstage,
    int32_t range )
```

Sets the voltage range

Parameters

Headstage	The headstage to query.
range	The voltage range in mV

11.40.3.35 SetVoltageResolution() [1/2]

```
void SetVoltageResolution (
    int32_t resolution )
```

Sets the voltage resolution

Parameters

resolution	The voltage resolution in µV/digit

11.40.3.36 SetVoltageResolution() [2/2]

```
void SetVoltageResolution (
    uint32_t Headstage,
    int32_t resolution )
```

Sets the voltage resolution

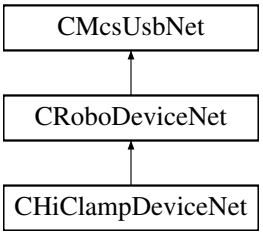
Parameters

Headstage	The headstage to query.
resolution	The voltage resolution in µV/digit

11.41 CHiClampDeviceNet Class Reference

CHiClampDeviceNet is the to control the MCS HiClamp device

Inheritance diagram for CHiClampDeviceNet:



Public Member Functions

- CHiClampDeviceNet (void)

Properties

- CRoboDacqNet^ RoboDacq `[get]`

Additional Inherited Members

11.41.1 Detailed Description

CHiClampDeviceNet is the to control the MCS HiClamp device

11.41.2 Constructor & Destructor Documentation

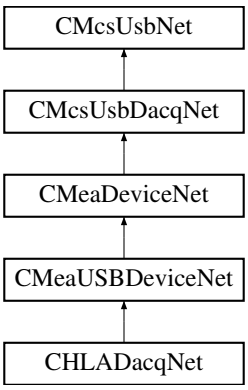
11.41.2.1 CHiClampDeviceNet() `CHiClampDeviceNet (void )`

11.41.3 Property Documentation

11.41.3.1 RoboDacq CRoboDacqNet^ RoboDacq [get]

11.42 CHLADacqNet Class Reference

Inheritance diagram for CHLADacqNet:



Public Member Functions

- CHLADacqNet (void)

Additional Inherited Members

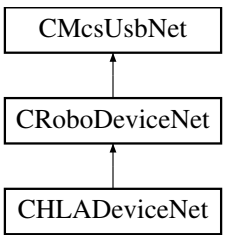
11.42.1 Constructor & Destructor Documentation

11.42.1.1 CHLADacqNet() CHLADacqNet (void)

11.43 CHLADeviceNet Class Reference

CHLADeviceNet is the to control the MCS HLA device

Inheritance diagram for CHLADeviceNet:



Public Member Functions

- CHLADeviceNet (void)

Properties

- CHLADacqNet^ HLADacq [get]
- CSerialPortNet^ SerialPort [get]

Additional Inherited Members

11.43.1 Detailed Description

CHLADeviceNet is the to control the MCS HLA device

11.43.2 Constructor & Destructor Documentation

11.43.2.1 CHLADeviceNet() CHLADeviceNet (void)

11.43.3 Property Documentation

11.43.3.1 HLADacq CHLADacqNet^ HLADacq [get]

11.43.3.2 SerialPort CSerialPortNet^ SerialPort [get]

11.44 CMcsUsbDacqNet::CHWInfo Class Reference

Class to provide hardware information about the device.

Classes

- class CVoltageRangeInfoNet

Public Member Functions

- CHWInfo (CMcsUsbDacqNet^ device)
- virtual uint32_t GetNumberOfHWADCChannels ([System::Runtime::InteropServices::Out]int% numberOf↩ Channels)

  Get the number of analog channels the device supports.
- virtual uint32_t GetNumberOfHWDigitalChannels ([System::Runtime::InteropServices::Out]int% numberOf↩ Channels)

  Get the number of digital channels the device supports.
- virtual bool IsDigitalChannelDedicated ()

  Query if the digital channel replaces an analog channel when enabled (e.g. on MC_Card) or adds a channel link on USB devices.
- virtual uint32_t GetAvailableSampleRates ([System::Runtime::InteropServices::Out]System::Collections::↩ Generic::List< int32_t >^% sampleRates)
- virtual System::Collections::Generic::List< int32_t > ^ GetAvailableVoltageRangesInMicroVolt (int milliGain)

  Gets a List of voltage ranges the device supports.
- virtual System::Collections::Generic::List< CVoltageRangeInfoNet^ > ^ GetAvailableVoltageRangesInMicroVoltAndStringsInMi (int milliGain)

  Gets a List of voltage ranges the device supports.

11.44.1 Detailed Description

Class to provide hardware information about the device.

11.44.2 Constructor & Destructor Documentation

11.44.2.1 CHWInfo() `CHWInfo (`

```
CMcsUsbDacqNet^ device )
```

11.44.3 Member Function Documentation

11.44.3.1 GetAvailableSampleRates() `virtual uint32_t GetAvailableSampleRates (`

```
[System::Runtime::InteropServices::Out] System::Collections::Generic::List< int32↩
_t >^% sampleRates )  [virtual]
```

11.44.3.2 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 Get↩ Gain() 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

milliGain	The gain factor (in milliGain) used to scale the list of voltage ranges.

Returns

List of voltage ranges in µV.

11.44.3.3 GetAvailableVoltageRangesInMicroVoltAndStringsInMilliVolt()

```
virtual System::Collections↩
::Generic::List<CVoltageRangeInfoNet^> ^ GetAvailableVoltageRangesInMicroVoltAndStringsIn↩
MilliVolt (
            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

milliGain	The gain factor (in milliGain) used to scale the list of voltage ranges.

Returns

List of voltage ranges in µV.

11.44.3.4 GetNumberOfHWADCChannels()

```
virtual uint32_t GetNumberOfHWADCChannels (
            [System::Runtime::InteropServices::Out] int% numberOfChannels ) [virtual]
```

Get the number of analog channels the device supports.

Parameters

numberOfChannels	Number of analog channels the device supports.

Returns

Error Status. 0 on success.

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

Get the number of digital channels the device supports.

Parameters

numberOfChannels	Number of digital channels the device supports.

Returns

Error Status. 0 on success.

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

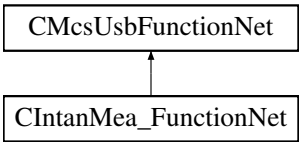
Query if the digital channel replaces an analog channel when enabled (e.g. on MC_Card) or adds a channel link on USB devices.

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.

11.45 CIntanMea_FunctionNet Class Reference

Inheritance diagram for CIntanMea_FunctionNet:



Public Member Functions

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

Additional Inherited Members

11.45.1 Constructor & Destructor Documentation

11.45.1.1 CIntanMea_FunctionNet() [1/2]

```
CIntanMea_FunctionNet (
    CMcsUsbNet^ mcsusb,
    CMcsUsbFunctionPointerContainer^ intalMea_FunctionPointerContainer )
```

11.45.1.2 CIntanMea_FunctionNet() [2/2]

```
CIntanMea_FunctionNet (
    CMcsUsbNet^ mcsusb )
```

11.45.2 Member Function Documentation

11.45.2.1 AmplifierSettle()

```
void AmplifierSettle ( )
```

11.45.2.2 BeginImpedanceCheck()

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

11.45.2.3 GetDSPHighPassByIndex()

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

11.45.2.4 GetImpedanceResult()

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

11.45.2.5 GetIntanRegister()

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

11.45.2.6 GetLowerFrequencyByIndex()

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

11.45.2.7 GetUpperFrequencyByIndex()

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

11.45.2.8 SetBandwidthByIndex()

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

11.45.2.9 SetDiagnosticMode()

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

11.45.2.10 SetDSPHighPassByIndex()

```
void SetDSPHighPassByIndex (
    int index )
```

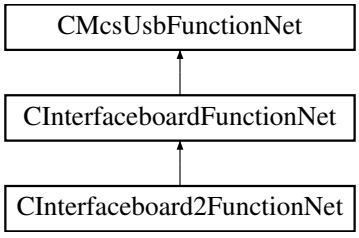
11.45.2.11 SetIntanRegister()

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

11.46 CInterfaceboard2FunctionNet Class Reference

CInterfaceboard2FunctionNet is the class to control the Interfaceboard

Inheritance diagram for CInterfaceboard2FunctionNet:



Public Member Functions

- CInterfaceboard2FunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ pInterfaceboard2↩ FunctionPointerContainer)

  Initializes a new instance of the CInterfaceboard2FunctionNet class.
- CInterfaceboard2FunctionNet (CMcsUsbNet^ mcsusb)
- virtual ~CInterfaceboard2FunctionNet ()
- !CInterfaceboard2FunctionNet ()
- void SetIoVoltage (IoVoltageEnumNet ioVoltage)

  Sets the I/O Voltage level for the IFB2 digital and AUX ports, default is 3.3V.
- IoVoltageEnumNet GetIoVoltage ()

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

Additional Inherited Members

11.46.1 Detailed Description

CInterfaceboard2FunctionNet is the class to control the Interfaceboard

11.46.2 Constructor & Destructor Documentation

11.46.2.1 CInterfaceboard2FunctionNet() [1/2]

```
CInterfaceboard2FunctionNet (
    CMcsUsbNet^ mcsusb,
    CMcsUsbFunctionPointerContainer^ pInterfaceboard2FunctionPointerContainer )
```

Initializes a new instance of the CInterfaceboard2FunctionNet class.

11.46.2.2 CInterfaceboard2FunctionNet() [2/2]

```
CInterfaceboard2FunctionNet (
    CMcsUsbNet^ mcsusb )
```

11.46.2.3 ~CInterfaceboard2FunctionNet()

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

11.46.2.4 "!CInterfaceboard2FunctionNet()

```
!CInterfaceboard2FunctionNet ( )
```

11.46.3 Member Function Documentation

11.46.3.1 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).

11.46.3.2 SetIoVoltage() void SetIoVoltage (IoVoltageEnumNet *ioVoltage*)

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

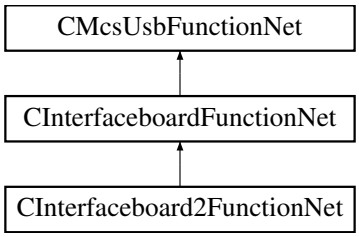
Parameters

ioVoltage	Enum for the I/O Voltage (3.3V or 5.0V).

11.47 CInterfaceboardFunctionNet Class Reference

CInterfaceboardFunctionNet is the class to control the Interfaceboard

Inheritance diagram for CInterfaceboardFunctionNet:



Public Member Functions

- CInterfaceboardFunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ pInterfaceboard↩ FunctionPointerContainer)

  Initializes a new instance of the CInterfaceboardFunctionNet class.
- CInterfaceboardFunctionNet (CMcsUsbNet^ mcsusb)
- virtual ~CInterfaceboardFunctionNet ()
- !CInterfaceboardFunctionNet ()
- void SetCardinalDacqSamplerate (uint32_t samplerate)

  Sets the fundamental/cardinal data aquisition samplerate of the Interfaceboard, default is 50 kHz
- uint32_t GetCardinalDacqSamplerate ()

  Gets the fundamental/cardinal data aquisition samplerate of the Interfaceboard, default is 50 kHz
- void SetCardinalStgOutputrate (uint32_t outputrate)

  Sets the fundamental/cardinal STG output rate of the Interfaceboard, default is 50 kHz
- uint32_t GetCardinalStgOutputrate ()

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

Additional Inherited Members

11.47.1 Detailed Description

CInterfaceboardFunctionNet is the class to control the Interfaceboard

11.47.2 Constructor & Destructor Documentation

11.47.2.1 CInterfaceboardFunctionNet() [1/2]

```
CInterfaceboardFunctionNet (
    CMcsUsbNet^ mcsusb,
    CMcsUsbFunctionPointerContainer^ pInterfaceboardFunctionPointerContainer )
```

Initializes a new instance of the CInterfaceboardFunctionNet class.

11.47.2.2 CInterfaceboardFunctionNet() [2/2]

```
CInterfaceboardFunctionNet (
    CMcsUsbNet^ mcsusb )
```

11.47.2.3 ~CInterfaceboardFunctionNet()

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

11.47.2.4 "!CInterfaceboardFunctionNet()

```
!CInterfaceboardFunctionNet ( )
```

11.47.3 Member Function Documentation

11.47.3.1 GetCardinalDacqSamplerate()

```
uint32_t GetCardinalDacqSamplerate ( )
```

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

Returns

The samplerate in Hz.

11.47.3.2 GetCardinalStgOutputrate()

```
uint32_t GetCardinalStgOutputrate ( )
```

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

Returns

The output rate in Hz.

11.47.3.3 SetCardinalDacqSamplerate()

```
void SetCardinalDacqSamplerate (
    uint32_t samplerate )
```

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

Parameters

samplerate	The samplerate in Hz.

11.47.3.4 SetCardinalStgOutputrate() `void SetCardinalStgOutputrate ( uint32_t outputrate )`

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

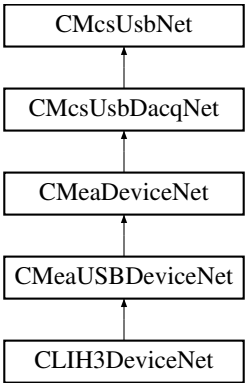
Parameters

outputrate	The output rate in Hz.

11.48 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)

  Delets 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 SetDacUseIdleValue (uint32_t DacChannel, bool UseIdle)

  Sets if the DAC Idle value is used after stimulation
- bool GetDacUseIdleValue (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)

  Delets 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^ StimulusFunction [get]

Additional Inherited Members

11.48.1 Detailed Description

CLIH3DeviceNet is the class to access the HEKA LIH3 device.

11.48.2 Constructor & Destructor Documentation

11.48.2.1 CLIH3DeviceNet() CLIH3DeviceNet ( )

Initializes a new instance of the CLIH3DeviceNet class.

11.48.2.2 ~CLIH3DeviceNet() virtual ~CLIH3DeviceNet ( ) [virtual]

11.48.2.3 "!CLIH3DeviceNet() !CLIH3DeviceNet ( )

11.48.3 Member Function Documentation

11.48.3.1 DummyCommand() void DummyCommand (uint32_t *dummyParameter*)

Dummy command to show how to use the DLL.

Parameters

dummyParameter	parameter to send to the device

11.48.3.2 EnableUserTrigger() void EnableUserTrigger (bool *enable*)

Enables the User Trigger

Parameters

enable	Enable

11.48.3.3 ErasePermanentAdcOffset() `void ErasePermanentAdcOffset (`

`LIH30_ADC_Channel_EnumNet` *AdcChannel*)

Delets the ADC offset of the DACQ for a single channel in permanent EEProm memory

Parameters

AdcChannel	The ADC channel

11.48.3.4 ErasePermanentDacOffset() `void ErasePermanentDacOffset (`

`LIH30_DAC_Channel_EnumNet` *DacChannel*)

Delets the DAC offset of the STG for a single channel in permanent EEProm memory

Parameters

DacChannel	The DAC channel

11.48.3.5 GetAdcOffset() `int32_t GetAdcOffset (`

`LIH30_ADC_Channel_EnumNet` *AdcChannel*)

Gets the ADC offset of the DACQ for a single channel

Parameters

AdcChannel	The ADC channel

Returns

The offset for the given channel number

11.48.3.6 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

Frequency	Frequency(1 - 25000 Hz)
Amplification	Amplification(0 - 0xFFFF)

11.48.3.7 GetDacIdleValue() `int32_t GetDacIdleValue ( uint32_t DacChannel )`

Gets the DAC Idle value

Parameters

DacChannel	The DAC channel

Returns

The idle value

11.48.3.8 GetDacOffset() `int32_t GetDacOffset ( LIH30_DAC_Channel_EnumNet DacChannel )`

Gets the offset of a DAC channel.

Parameters

DacChannel	The DAC channel

Returns

The offset for the given channel number

11.48.3.9 GetDacqRunStatus() `uint16_t GetDacqRunStatus ( )`

Gets the data acquisition running status

Returns

The status (1: running / 0: stopped)

11.48.3.10 GetDacUseIdleValue() `bool GetDacUseIdleValue ( uint32_t DacChannel )`

Gets if the DAC Idle value is used after stimulation

Parameters

DacChannel	The DAC channel

Returns

Use idle value

11.48.3.11 GetDigInState() `uint16_t GetDigInState ( )`

Reads from the LIH30 digital input

Returns

The bit mask defining the digital input state

11.48.3.12 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

EEpromStartAddress	start address of memory area to read from
EEpromData_Length	The maximal length of EEpromData.
epc10bus	The EPC10 bus

Returns

pointer to internal memory for the requested amount of data

11.48.3.13 GetSampleInterval() `uint32_t GetSampleInterval ( )`

Gets the Sample Interval for the DACQ and Stimulation

Returns

Sample Interval configured on the device

11.48.3.14 IsUserTriggerEnabled() `bool IsUserTriggerEnabled ( )`

Is the User Trigger enabled

Returns

Enabled

11.48.3.15 ReadClipping() `uint32_t ReadClipping ( LIH30_EPC10_Bus_EnumNet epc10bus )`

Gets the clipping information

Parameters

epc10bus	The EPC10 bus

Returns

The clipping value

11.48.3.16 ReadUARTData() `String ^ ReadUARTData ( )`

Reads the config string from the device.

Returns

The config string.

11.48.3.17 SendCommand() `void SendCommand ( LIH30_EPC10_Bus_EnumNet epc10bus, uint16_t Command )`

Send command to the EPC10

Parameters

epc10bus	The EPC10 bus
Command	The command

11.48.3.18 SetAdcOffset() void SetAdcOffset (LIH30_ADC_Channel_EnumNet *AdcChannel*, int32_t *Offset*)

Sets the ADC offset of the DACQ for a single channel

Parameters

AdcChannel	The ADC channel
Offset	The offset for the given channel number

11.48.3.19 SetAdcOffsetPermanent() void SetAdcOffsetPermanent (LIH30_ADC_Channel_EnumNet *AdcChannel*)

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

Parameters

AdcChannel	The ADC channel

11.48.3.20 SetAudioOutDacParameter() void SetAudioOutDacParameter (uint32_t *Frequency*, uint32_t *Amplification*)

Sets the parameter of the audio DAC output.

Parameters

Frequency	Frequency(1 - 25000 Hz)
Amplification	Amplification(0 - 0xFFFF)

11.48.3.21 SetDacIdleValue() void SetDacIdleValue (uint32_t *DacChannel*, int32_t *IdleValue*)

Sets the DAC Idle value

Parameters

DacChannel	The DAC channel
IdleValue	The idle value

11.48.3.22 SetDacOffset() void SetDacOffset (LIH30_DAC_Channel_EnumNet *DacChannel*, int32_t *Offset*)

Sets the offset of a DAC channel.

Parameters

DacChannel	The DAC channel
Offset	The offset for the given channel number

11.48.3.23 SetDacOffsetPermanent() void SetDacOffsetPermanent (LIH30_DAC_Channel_EnumNet *DacChannel*)

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

Parameters

DacChannel	The DAC channel

11.48.3.24 SetDacUseIdleValue() void SetDacUseIdleValue (uint32_t *DacChannel*, bool *UseIdle*)

Sets if the DAC Idle value is used after stimulation

Parameters

DacChannel	The DAC channel
UseIdle	Use idle value

11.48.3.25 SetDigOutState() void SetDigOutState (uint16_t *DigOutState*)

Writes to the LIH30 digital output

Parameters

DigOutState	The bit mask defining the digital output state

11.48.3.26 SetEEpromPage() void SetEEpromPage (

```
uint32_t EEpromStartAddress,
array< int8_t >^ EEpromData,
LIH30_EPC10_Bus_EnumNet epc10bus )
```

Writes into EEprom on the EPC10 EEPROM

Parameters

EEpromStartAddress	start address of memory area to write to
EEpromData	pointer to internal memory for the supported amount of data
epc10bus	The EPC10 bus

11.48.3.27 SetSampleInterval() `void SetSampleInterval (`

```
uint32_t SampleInterval )
```

Sets the Sample Interval for the DACQ and Stimulation

Parameters

SampleInterval	between the samples, Sample interval is available from 1 to 4194303

11.48.3.28 WriteUARTData() `void WriteUARTData (`

```
String^ commandString )
```

Write the command string to the device.

Parameters

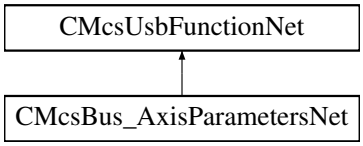
commandString	The config string.

11.48.4 Property Documentation

11.48.4.1 StimulusFunction `CStimulusFunctionNet^ StimulusFunction [get]`

11.49 CMcsBus_AxisParametersNet Class Reference

Inheritance diagram for CMcsBus_AxisParametersNet:



Public Member Functions

- CMcsBus_AxisParametersNet (CMcsUsbNet^ device)
- ~CMcsBus_AxisParametersNet (void)
- void SetAxisParametersEeprom (unsigned char busnumber, unsigned char busaddress, unsigned char axis, unsigned short index, unsigned int parameter)
- void SetAxisParametersEeprom (unsigned char busnumber, unsigned char busaddress, unsigned char axis, unsigned short index, int parameter)
- unsigned int GetAxisParametersUnsignedEeprom (unsigned char busnumber, unsigned char busaddress, unsigned char axis, unsigned short index)
- int GetAxisParametersSignedEeprom (unsigned char busnumber, unsigned char busaddress, unsigned char axis, unsigned short index)

Additional Inherited Members

11.49.1 Constructor & Destructor Documentation

11.49.1.1 CMcsBus_AxisParametersNet()

```
CMcsBus_AxisParametersNet (
    CMcsUsbNet^ device )
```

11.49.1.2 ~CMcsBus_AxisParametersNet()

```
~CMcsBus_AxisParametersNet (
    void  )
```

11.49.2 Member Function Documentation

11.49.2.1 GetAxisParametersSignedEeprom()

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

11.49.2.2 GetAxisParametersUnsignedEeprom()

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

11.49.2.3 SetAxisParametersEeprom() [1/2]

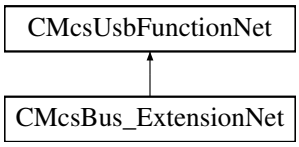
```
void SetAxisParametersEeprom (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis,
    unsigned short index,
    int parameter )
```

11.49.2.4 SetAxisParametersEeprom() [2/2]

```
void SetAxisParametersEeprom (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis,
    unsigned short index,
    unsigned int parameter )
```

11.50 CMcsBus_ExtensionNet Class Reference

Inheritance diagram for CMcsBus_ExtensionNet:



Public Member Functions

- CMcsBus_ExtensionNet (CMcsUsbNet^ device)
- ~CMcsBus_ExtensionNet (void)
- void SetLEDSwitch (unsigned char busnumber, unsigned char busaddress, unsigned short LEDSwitch)
- unsigned short GetLEDSwitch (unsigned char busnumber, unsigned char busaddress)

Additional Inherited Members

11.50.1 Constructor & Destructor Documentation

11.50.1.1 CMcsBus_ExtensionNet()

```
CMcsBus_ExtensionNet (
    CMcsUsbNet^ device )
```

11.50.1.2 ~CMcsBus_ExtensionNet()

```
~CMcsBus_ExtensionNet (
    void  )
```

11.50.2 Member Function Documentation

11.50.2.1 GetLEDSwitch() `unsigned short GetLEDSwitch (`

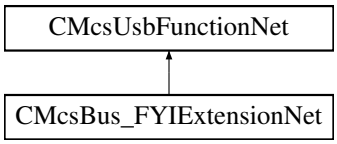
```
unsigned char busnumber,
unsigned char busaddress )
```

11.50.2.2 SetLEDSwitch() `void SetLEDSwitch (`

```
unsigned char busnumber,
unsigned char busaddress,
unsigned short LEDSwitch )
```

11.51 CMcsBus_FYIExtensionNet Class Reference

Inheritance diagram for CMcsBus_FYIExtensionNet:



Public Member Functions

- CMcsBus_FYIExtensionNet (CMcsUsbNet^ device)
- ~CMcsBus_FYIExtensionNet (void)
- void SetValves (unsigned char busnumber, unsigned char busaddress, unsigned int states)
- unsigned int GetValves (unsigned char busnumber, unsigned char busaddress)
- void SetDIO (unsigned char busnumber, unsigned char busaddress, unsigned short io)
- unsigned short GetDIO (unsigned char busnumber, unsigned char busaddress)
- void SetSingleHeater (unsigned char busnumber, unsigned char busaddress, short index, unsigned short power)
- unsigned short GetSingleHeater (unsigned char busnumber, unsigned char busaddress, short index)

Additional Inherited Members

11.51.1 Constructor & Destructor Documentation

11.51.1.1 CMcsBus_FYIExtensionNet() `CMcsBus_FYIExtensionNet (`

```
CMcsUsbNet^ device )
```

11.51.1.2 ~CMcsBus_FYIExtensionNet() ~CMcsBus_FYIExtensionNet (void)

11.51.2 Member Function Documentation

11.51.2.1 GetDIO() unsigned short GetDIO (unsigned char *busnumber*, unsigned char *busaddress*)

11.51.2.2 GetSingleHeater() unsigned short GetSingleHeater (unsigned char *busnumber*, unsigned char *busaddress*, short *index*)

11.51.2.3 GetValves() unsigned int GetValves (unsigned char *busnumber*, unsigned char *busaddress*)

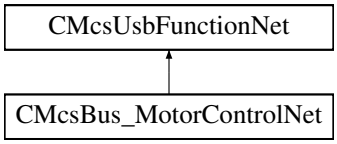
11.51.2.4 SetDIO() void SetDIO (unsigned char *busnumber*, unsigned char *busaddress*, unsigned short *io*)

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

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

11.52 CMcsBus_MotorControlNet Class Reference

Inheritance diagram for CMcsBus_MotorControlNet:



Public Member Functions

- CMcsBus_MotorControlNet (CMcsUsbNet^ 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

11.52.1 Constructor & Destructor Documentation

11.52.1.1 CMcsBus_MotorControlNet()

```
CMcsBus_MotorControlNet (
    CMcsUsbNet^ device )
```

11.52.1.2 ~CMcsBus_MotorControlNet()

```
~CMcsBus_MotorControlNet (
    void  )
```

11.52.2 Member Function Documentation

11.52.2.1 GetMCAcceleration()

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

11.52.2.2 GetMCAccelerationEeprom()

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

11.52.2.3 GetMCAccelerationShortCommand()

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

11.52.2.4 GetMCAxisRevisionEeprom()

```
unsigned short GetMCAxisRevisionEeprom (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.5 GetMCBreakCurrent()

```
short GetMCBreakCurrent (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.6 GetMCBreakCurrentEeprom()

```
short GetMCBreakCurrentEeprom (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.7 GetMCConfig()

```
unsigned short GetMCConfig (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.8 GetMCConfigEeprom()

```
unsigned short GetMCConfigEeprom (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.9 GetMCCurrent()

```
short GetMCCurrent (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.10 GetMCCurrentEeprom()

```
short GetMCCurrentEeprom (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.11 GetMCCurrentMode()

```
RoboCurrentModeEnumNet GetMCCurrentMode (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.12 GetMCCurrentModeEeprom()

```
RoboCurrentModeEnumNet GetMCCurrentModeEeprom (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.13 GetMCCurrentModeShortCommand()

```
RoboCurrentModeEnumNet GetMCCurrentModeShortCommand (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.14 GetMCCurrentPosition()

```
int GetMCCurrentPosition (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.15 GetMCCurrentShortCommand()

```
short GetMCCurrentShortCommand (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.16 GetMCCurrentSpeed()

```
short GetMCCurrentSpeed (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.17 GetMCMaxAcceleration()

```
unsigned short GetMCMaxAcceleration (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.18 GetMCMaxAccelerationEeprom()

```
unsigned short GetMCMaxAccelerationEeprom (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.19 GetMCMaxCurrent()

```
short GetMCMaxCurrent (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.20 GetMCMaxCurrentEeprom()

```
short GetMCMaxCurrentEeprom (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.21 GetMCMaxSpeed()

```
unsigned short GetMCMaxSpeed (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.22 GetMCMaxSpeedEeprom()

```
unsigned short GetMCMaxSpeedEeprom (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.23 GetMCMaxTravel()

```
int GetMCMaxTravel (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.24 GetMCMaxTravelEeprom()

```
int GetMCMaxTravelEeprom (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.25 GetMCMaxTravelShortCommand()

```
int GetMCMaxTravelShortCommand (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.26 GetMCMovement()

```
unsigned short GetMCMovement (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.27 GetMCNewPosition()

```
int GetMCNewPosition (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.28 GetMCOutputOnOff()

```
bool GetMCOutputOnOff (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.29 GetMCPhase()

```
unsigned short GetMCPhase (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.30 GetMCPhaseOffset()

```
unsigned short GetMCPhaseOffset (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.31 GetMCReference()

```
unsigned char GetMCReference (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis,
    [System::Runtime::InteropServices::Out] unsigned char% switch_port )
```

11.52.2.32 GetMCReferenceCurrent()

```
short GetMCReferenceCurrent (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.33 GetMCReferenceCurrentEeprom()

```
short GetMCReferenceCurrentEeprom (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.34 GetMCRegulatorGain()

```
short GetMCRegulatorGain (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.35 GetMCRegulatorGainEeprom()

```
short GetMCRegulatorGainEeprom (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.36 GetMCScalingFactor()

```
int GetMCScalingFactor (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.37 GetMCScalingFactorEeprom()

```
int GetMCScalingFactorEeprom (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.38 GetMCSpeed()

```
short GetMCSpeed (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.39 GetMCSpeedEeprom()

```
unsigned short GetMCSpeedEeprom (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.40 GetMCSpeedShortCommand()

```
short GetMCSpeedShortCommand (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.41 GetMCSpeedUnitEeprom()

```
int32_t GetMCSpeedUnitEeprom (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.42 GetMCStandbyCurrent()

```
short GetMCStandbyCurrent (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.43 GetMCStandbyCurrentEeprom()

```
short GetMCStandbyCurrentEeprom (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.44 GetMCStandbyTime()

```
short GetMCStandbyTime (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.45 GetMCStandbyTimeEeprom()

```
short GetMCStandbyTimeEeprom (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.46 GetSubChannel()

```
unsigned short GetSubChannel (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.47 SetMCAcceleration()

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

11.52.2.48 SetMCAccelerationEeprom()

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

11.52.2.49 SetMCAccelerationShortCommand()

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

11.52.2.50 SetMCAxisRevisionEeprom()

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

11.52.2.51 SetMCBreakCurrent()

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

11.52.2.52 SetMCBreakCurrentEeprom()

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

11.52.2.53 SetMCConfig()

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

11.52.2.54 SetMCConfigEeprom()

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

11.52.2.55 SetMCCurrent()

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

11.52.2.56 SetMCCurrentEeprom()

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

11.52.2.57 SetMCCurrentMode()

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

11.52.2.58 SetMCCurrentModeEeprom()

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

11.52.2.59 SetMCCurrentModeShortCommand()

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

11.52.2.60 SetMCCurrentPosition()

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

11.52.2.61 SetMCCurrentShortCommand()

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

11.52.2.62 SetMCMaxAcceleration()

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

11.52.2.63 SetMCMaxAccelerationEeprom()

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

11.52.2.64 SetMCMaxCurrent()

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

11.52.2.65 SetMCMaxCurrentEeprom()

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

11.52.2.66 SetMCMaxSpeed()

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

11.52.2.67 SetMCMaxSpeedEeprom()

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

11.52.2.68 SetMCMaxTravel()

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

11.52.2.69 SetMCMaxTravelEeprom()

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

11.52.2.70 SetMCMaxTravelShortCommand()

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

11.52.2.71 SetMCNewPosition()

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

11.52.2.72 SetMCOutputOnOff()

```
void SetMCOutputOnOff (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis,
    bool OnOff_status )
```

11.52.2.73 SetMCReference()

```
void SetMCReference (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis,
    unsigned char switch_enable,
    unsigned char switch_polarity )
```

11.52.2.74 SetMCReferenceCurrent()

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

11.52.2.75 SetMCReferenceCurrentEeprom()

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

11.52.2.76 SetMCRegulatorGain()

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

11.52.2.77 SetMCRegulatorGainEeprom()

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

11.52.2.78 SetMCRotation()

```
void SetMCRotation (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis,
    unsigned char onoff )
```

11.52.2.79 SetMCScalingFactor()

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

11.52.2.80 SetMCScalingFactorEeprom()

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

11.52.2.81 SetMCSpeed()

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

11.52.2.82 SetMCSpeedEeprom()

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

11.52.2.83 SetMCSpeedShortCommand()

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

11.52.2.84 SetMCSpeedUnitEeprom()

```
void SetMCSpeedUnitEeprom (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis,
    int32_t speedunit )
```

11.52.2.85 SetMCStandbyCurrent()

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

11.52.2.86 SetMCStandbyCurrentEeprom()

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

11.52.2.87 SetMCStandbyTime()

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

11.52.2.88 SetMCStandbyTimeEeprom()

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

11.52.2.89 SetSubChannel()

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

11.52.2.90 StartMCMovement()

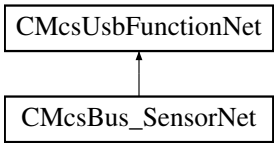
```
void StartMCMovement (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.52.2.91 StopMCMovement()

```
void StopMCMovement (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.53 CMcsBus_SensorNet Class Reference

Inheritance diagram for CMcsBus_SensorNet:



Public Member Functions

- CMcsBus_SensorNet (CMcsUsbNet^ device)
- ~CMcsBus_SensorNet (void)
- void SetMinimalThreshold (unsigned char busnumber, unsigned char busaddress, unsigned short threshold)
- unsigned short GetMinimalThreshold (unsigned char busnumber, unsigned char busaddress)
- void SetDetectionThreshold (unsigned char busnumber, unsigned char busaddress, unsigned short threshold)
- unsigned short GetDetectionThreshold (unsigned char busnumber, unsigned char busaddress)
- void SetLatency (unsigned char busnumber, unsigned char busaddress, unsigned short latency)
- unsigned short GetLatency (unsigned char busnumber, unsigned char busaddress)
- unsigned short GetBubbleStatus (unsigned char busnumber, unsigned char busaddress)
- unsigned short GetLatencyCounter (unsigned char busnumber, unsigned char busaddress)
- unsigned short GetDetectorValue (unsigned char busnumber, unsigned char busaddress)
- array< int > ^ GetPressure (unsigned char busnumber, unsigned char busaddress, int n)
- int GetPressure (unsigned char busnumber, unsigned char busaddress, unsigned short index)
- void SetRegulatorOnOff (unsigned char busnumber, unsigned char busaddress, unsigned short index, unsigned char onoff)
- unsigned char GetRegulatorOnOff (unsigned char busnumber, unsigned char busaddress, unsigned short index)

- void SetSollPressure (unsigned char busnumber, unsigned char busaddress, unsigned short index, int pressure)
- int GetSollPressure (unsigned char busnumber, unsigned char busaddress, unsigned short index)
- void SetRegulatorFactor (unsigned char busnumber, unsigned char busaddress, unsigned short index, int factor)
- int GetRegulatorFactor (unsigned char busnumber, unsigned char busaddress, unsigned short index)
- void SetPressureOffset (unsigned char busnumber, unsigned char busaddress, unsigned short index)
- array< unsigned short > ^ GetPressureOffset (unsigned char busnumber, unsigned char busaddress)
- int GetPressureOffset (unsigned char busnumber, unsigned char busaddress, unsigned short index)
- unsigned int GetRegulatorStatus (unsigned char busnumber, unsigned char busaddress, unsigned short index)
- void SetRotatePump (unsigned char busnumber, unsigned char busaddress, unsigned short index, short speed)
- short GetRotatePump (unsigned char busnumber, unsigned char busaddress, unsigned short index)
- void SetMovePump (unsigned char busnumber, unsigned char busaddress, unsigned short index, unsigned short speed, int position)
- void GetMovePump (unsigned char busnumber, unsigned char busaddress, unsigned short index, [System::Runtime::InteropServices::Out]unsigned short% speed, [System::Runtime::InteropServices::Out]int% position)
- void SetRegulationTimeouts (unsigned char busnumber, unsigned char busaddress, unsigned short MaxSpeedWait, unsigned short MaxSignChange)
- void GetRegulationTimeouts (unsigned char busnumber, unsigned char busaddress, [System::Runtime::InteropServices::Out]unsigned short% MaxSpeedWait, [System::Runtime::InteropServices::Out]unsigned short% MaxSignChange)
- array< int > ^ Get4ADC (unsigned char busnumber, unsigned char busaddress)
- array< int > ^ Get4ADCAverage (unsigned char busnumber, unsigned char busaddress)
- void Set4DAC (unsigned char busnumber, unsigned char busaddress, array< unsigned short >^ dac)
- array< unsigned short > ^ Get4DAC (unsigned char busnumber, unsigned char busaddress)
- void Set4ADCMode (unsigned char busnumber, unsigned char busaddress, PatchServAdcModeEnumNet mode)
- PatchServAdcModeEnumNet Get4ADCMode (unsigned char busnumber, unsigned char busaddress)
- void Set4ADCCatchampAverageShift (unsigned char busnumber, unsigned char busaddress, unsigned int shift)
- unsigned int Get4ADCCatchampAverageShift (unsigned char busnumber, unsigned char busaddress)
- array< unsigned short > ^ Get2AnalogInput (unsigned char busnumber, unsigned char busaddress)
- unsigned short Get2DigitalInput (unsigned char busnumber, unsigned char busaddress)
- array< unsigned short > ^ GetADCs (unsigned char busnumber, unsigned char busaddress, int n)
- array< unsigned short > ^ GetADCsLoop (unsigned char busnumber, unsigned char busaddress, int n)
- void SetPiezoState (unsigned char busnumber, unsigned char busaddress, int state)
- void GetPiezoState (unsigned char busnumber, unsigned char busaddress, [System::Runtime::InteropServices::Out]int% state, [System::Runtime::InteropServices::Out]int% reason)
- void SetDACs (unsigned char busnumber, unsigned char busaddress, unsigned short index, array< unsigned short >^ dac_times_voltages)
- array< unsigned short > ^ GetDACs (unsigned char busnumber, unsigned char busaddress, unsigned short index)
- void SetSamplePeriode (unsigned char busnumber, unsigned char busaddress, unsigned short periode)
- unsigned short GetSamplePeriode (unsigned char busnumber, unsigned char busaddress)
- void StartSync (unsigned char busnumber, unsigned char busaddress)
- unsigned short GetSyncState (unsigned char busnumber, unsigned char busaddress)
- void CatchAmpSetDacAmplitude (unsigned char busnumber, unsigned char busaddress, unsigned short dacAmplitude)
- unsigned short CatchAmpGetDacAmplitude (unsigned char busnumber, unsigned char busaddress)
- void CatchAmpSetDacOffset (unsigned char busnumber, unsigned char busaddress, short dacOffset)
- short CatchAmpGetDacOffset (unsigned char busnumber, unsigned char busaddress)
- int CatchAmpGetAdcMean (unsigned char busnumber, unsigned char busaddress)
- int CatchAmpGetAdcValue (unsigned char busnumber, unsigned char busaddress)

- int CatchAmpGetAdcValueH (unsigned char busnumber, unsigned char busaddress)
- int CatchAmpGetAdcValueL (unsigned char busnumber, unsigned char busaddress)
- void CatchAmpSetPwmEnable (unsigned char busnumber, unsigned char busaddress, bool pwmEnable)
- bool CatchAmpGetPwmEnable (unsigned char busnumber, unsigned char busaddress)
- void CatchAmpSetDacEnable (unsigned char busnumber, unsigned char busaddress, bool dacEnable)
- bool CatchAmpGetDacEnable (unsigned char busnumber, unsigned char busaddress)
- int TactSwitchGetState (unsigned char busnumber, unsigned char busaddress)
- void TactSwitchSetDisplay (unsigned char busnumber, unsigned char busaddress, int Melody)

Additional Inherited Members

11.53.1 Constructor & Destructor Documentation

11.53.1.1 CMcsBus_SensorNet() CMcsBus_SensorNet (CMcsUsbNet^ *device*)

11.53.1.2 ~CMcsBus_SensorNet() ~CMcsBus_SensorNet (void)

11.53.2 Member Function Documentation

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

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

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

11.53.2.4 CatchAmpGetAdcValueL()

```
int CatchAmpGetAdcValueL (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.53.2.5 CatchAmpGetDacAmplitude()

```
unsigned short CatchAmpGetDacAmplitude (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.53.2.6 CatchAmpGetDacEnable()

```
bool CatchAmpGetDacEnable (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.53.2.7 CatchAmpGetDacOffset()

```
short CatchAmpGetDacOffset (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.53.2.8 CatchAmpGetPwmEnable()

```
bool CatchAmpGetPwmEnable (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.53.2.9 CatchAmpSetDacAmplitude()

```
void CatchAmpSetDacAmplitude (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short dacAmplitude )
```

11.53.2.10 CatchAmpSetDacEnable()

```
void CatchAmpSetDacEnable (
    unsigned char busnumber,
    unsigned char busaddress,
    bool dacEnable )
```

11.53.2.11 CatchAmpSetDacOffset()

```
void CatchAmpSetDacOffset (
    unsigned char busnumber,
    unsigned char busaddress,
    short dacOffset )
```

11.53.2.12 CatchAmpSetPwmEnable()

```
void CatchAmpSetPwmEnable (
    unsigned char busnumber,
    unsigned char busaddress,
    bool pwmEnable )
```

11.53.2.13 Get2AnalogInput()

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

11.53.2.14 Get2DigitalInput()

```
unsigned short Get2DigitalInput (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.53.2.15 Get4ADC()

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

11.53.2.16 Get4ADCAverage()

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

11.53.2.17 Get4ADCCatchampAverageShift()

```
unsigned int Get4ADCCatchampAverageShift (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.53.2.18 Get4ADCMode()

```
PatchServAdcModeEnumNet Get4ADCMode (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.53.2.19 Get4DAC()

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

11.53.2.20 GetADCs()

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

11.53.2.21 GetADCsLoop()

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

11.53.2.22 GetBubbleStatus()

```
unsigned short GetBubbleStatus (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.53.2.23 GetDACs()

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

11.53.2.24 GetDetectionThreshold()

```
unsigned short GetDetectionThreshold (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.53.2.25 GetDetectorValue()

```
unsigned short GetDetectorValue (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.53.2.26 GetLatency()

```
unsigned short GetLatency (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.53.2.27 GetLatencyCounter()

```
unsigned short GetLatencyCounter (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.53.2.28 GetMinimalThreshold()

```
unsigned short GetMinimalThreshold (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.53.2.29 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 )
```

11.53.2.30 GetPiezoState()

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

11.53.2.31 GetPressure() [1/2]

```
array<int> ^ GetPressure (
    unsigned char busnumber,
    unsigned char busaddress,
    int n )
```

11.53.2.32 GetPressure() [2/2]

```
int GetPressure (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short index )
```

11.53.2.33 GetPressureOffset() [1/2]

```
array<unsigned short> ^ GetPressureOffset (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.53.2.34 GetPressureOffset() [2/2]

```
int GetPressureOffset (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short index )
```

11.53.2.35 GetRegulationTimeouts()

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

11.53.2.36 GetRegulatorFactor()

```
int GetRegulatorFactor (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short index )
```

11.53.2.37 GetRegulatorOnOff()

```
unsigned char GetRegulatorOnOff (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short index )
```

11.53.2.38 GetRegulatorStatus()

```
unsigned int GetRegulatorStatus (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short index )
```

11.53.2.39 GetRotatePump()

```
short GetRotatePump (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short index )
```

11.53.2.40 GetSamplePeriode()

```
unsigned short GetSamplePeriode (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.53.2.41 GetSollPressure()

```
int GetSollPressure (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short index )
```

11.53.2.42 GetSyncState()

```
unsigned short GetSyncState (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.53.2.43 Set4ADCCatchampAverageShift()

```
void Set4ADCCatchampAverageShift (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned int shift )
```

11.53.2.44 Set4ADCMode()

```
void Set4ADCMode (
    unsigned char busnumber,
    unsigned char busaddress,
    PatchServAdcModeEnumNet mode )
```

11.53.2.45 Set4DAC()

```
void Set4DAC (
    unsigned char busnumber,
    unsigned char busaddress,
    array< unsigned short >^ dac )
```

11.53.2.46 SetDACs()

```
void SetDACs (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short index,
    array< unsigned short >^ dac_times_voltages )
```

11.53.2.47 SetDetectionThreshold()

```
void SetDetectionThreshold (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short threshold )
```

11.53.2.48 SetLatency()

```
void SetLatency (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short latency )
```

11.53.2.49 SetMinimalThreshold()

```
void SetMinimalThreshold (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short threshold )
```

11.53.2.50 SetMovePump()

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

11.53.2.51 SetPiezoState()

```
void SetPiezoState (
    unsigned char busnumber,
    unsigned char busaddress,
    int state )
```

11.53.2.52 SetPressureOffset()

```
void SetPressureOffset (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short index )
```

11.53.2.53 SetRegulationTimeouts()

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

11.53.2.54 SetRegulatorFactor()

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

11.53.2.55 SetRegulatorOnOff()

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

11.53.2.56 SetRotatePump()

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

11.53.2.57 SetSamplePeriode()

```
void SetSamplePeriode (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short periode )
```

11.53.2.58 SetSollPressure()

```
void SetSollPressure (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short index,
    int pressure )
```

11.53.2.59 StartSync()

```
void StartSync (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.53.2.60 TactSwitchGetState()

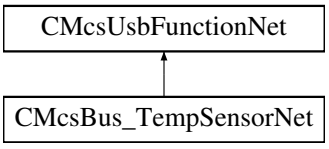
```
int TactSwitchGetState (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.53.2.61 TactSwitchSetDisplay()

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

11.54 CMcsBus_TempSensorNet Class Reference

Inheritance diagram for CMcsBus_TempSensorNet:



Public Member Functions

- CMcsBus_TempSensorNet (CMcsUsbNet^ 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

11.54.1 Constructor & Destructor Documentation

11.54.1.1 CMcsBus_TempSensorNet()

```
CMcsBus_TempSensorNet (
    CMcsUsbNet^ device )
```

11.54.1.2 ~CMcsBus_TempSensorNet()

```
~CMcsBus_TempSensorNet (
    void  )
```

11.54.2 Member Function Documentation

11.54.2.1 GetNanoVoltsPerKelvin()

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

11.54.2.2 GetTemperatur() [1/2]

```
short GetTemperatur (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.54.2.3 GetTemperatur() [2/2]

```
short GetTemperatur (
    unsigned char busnumber,
    unsigned char busaddress,
    short index )
```

11.54.2.4 GetThermoOffset()

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

11.54.2.5 GetThermoTemp()

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

11.54.2.6 GetThermoVoltage()

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

11.54.2.7 SetNanoVoltsPerKelvin()

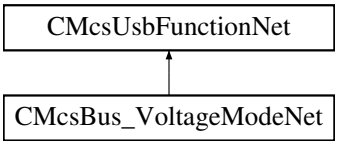
```
void SetNanoVoltsPerKelvin (
    unsigned char busnumber,
    unsigned char busaddress,
    int nanovoltsperkelvin )
```

11.54.2.8 SetThermoOffset()

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

11.55 CMcsBus_VoltageModeNet Class Reference

Inheritance diagram for CMcsBus_VoltageModeNet:



Public Member Functions

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

Additional Inherited Members

11.55.1 Constructor & Destructor Documentation

11.55.1.1 CMcsBus_VoltageModeNet() `CMcsBus_VoltageModeNet ( CMcsUsbNet^ device )`

11.55.1.2 ~CMcsBus_VoltageModeNet() ~CMcsBus_VoltageModeNet (void)

11.55.2 Member Function Documentation

11.55.2.1 GetVMMaxNegativeCurrent() short GetVMMaxNegativeCurrent (unsigned char *busnumber*, unsigned char *busaddress*, unsigned char *channel*)

11.55.2.2 GetVMMaxNegativeCurrentEeprom() short GetVMMaxNegativeCurrentEeprom (unsigned char *busnumber*, unsigned char *busaddress*, unsigned char *channel*)

11.55.2.3 GetVMMaxNegativeVoltage() short GetVMMaxNegativeVoltage (unsigned char *busnumber*, unsigned char *busaddress*, unsigned char *channel*)

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

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

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

11.55.2.7 GetVMMaxPositiveVoltage()

```
short GetVMMaxPositiveVoltage (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char channel )
```

11.55.2.8 GetVMMaxPositiveVoltageEeprom()

```
short GetVMMaxPositiveVoltageEeprom (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char channel )
```

11.55.2.9 GetVMOutputOnOff()

```
unsigned short GetVMOutputOnOff (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char channel )
```

11.55.2.10 GetVMVoltage()

```
short GetVMVoltage (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char channel )
```

11.55.2.11 SetVMMaxNegativeCurrent()

```
void SetVMMaxNegativeCurrent (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char channel,
    short current )
```

11.55.2.12 SetVMMaxNegativeCurrentEeprom()

```
void SetVMMaxNegativeCurrentEeprom (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char channel,
    short current )
```

11.55.2.13 SetVMMaxNegativeVoltage()

```
void SetVMMaxNegativeVoltage (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char channel,
    short voltage )
```

11.55.2.14 SetVMMaxNegativeVoltageEeprom()

```
void SetVMMaxNegativeVoltageEeprom (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char channel,
    short voltage )
```

11.55.2.15 SetVMMaxPositiveCurrent()

```
void SetVMMaxPositiveCurrent (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char channel,
    short current )
```

11.55.2.16 SetVMMaxPositiveCurrentEeprom()

```
void SetVMMaxPositiveCurrentEeprom (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char channel,
    short current )
```

11.55.2.17 SetVMMaxPositiveVoltage()

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

11.55.2.18 SetVMMaxPositiveVoltageEeprom()

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

11.55.2.19 SetVMOutputOnOff()

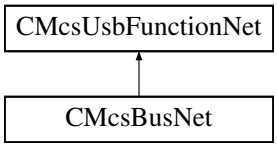
```
void SetVMOutputOnOff (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char channel,
    unsigned short status )
```

11.55.2.20 SetVMVoltage()

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

11.56 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

11.56.1 Constructor & Destructor Documentation

11.56.1.1 CMcsBusNet() `CMcsBusNet (CMcsUsbNet^ device )`

11.56.1.2 ~CMcsBusNet() virtual ~CMcsBusNet (void) [virtual]

11.56.2 Member Function Documentation

11.56.2.1 CMcsBusNet::GetMode() unsigned short CMcsBusNet::GetMode (unsigned char *busnumber*, unsigned char *busaddress*)

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

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

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

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

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

11.56.2.7 GetCommand() [1/4]

```
void GetCommand (
    unsigned char command,
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis,
    [System::Runtime::InteropServices::Out] int% value )
```

11.56.2.8 GetCommand() [2/4]

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

11.56.2.9 GetCommand() [3/4]

```
void GetCommand (
    unsigned char command,
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis,
    [System::Runtime::InteropServices::Out] unsigned int% value )
```

11.56.2.10 GetCommand() [4/4]

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

11.56.2.11 GetHWRevisionEeprom()

```
unsigned short GetHWRevisionEeprom (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.56.2.12 SetBusAddress()

```
void SetBusAddress (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short newaddress )
```

11.56.2.13 SetBusAddressEeprom()

```
void SetBusAddressEeprom (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short newaddress )
```

11.56.2.14 SetCommand() [1/4]

```
void SetCommand (
    unsigned char command,
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis,
    int value )
```

11.56.2.15 SetCommand() [2/4]

```
void SetCommand (
    unsigned char command,
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis,
    short value )
```

11.56.2.16 SetCommand() [3/4]

```
void SetCommand (
    unsigned char command,
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis,
    unsigned int value )
```

11.56.2.17 SetCommand() [4/4]

```
void SetCommand (
    unsigned char command,
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis,
    unsigned short value )
```

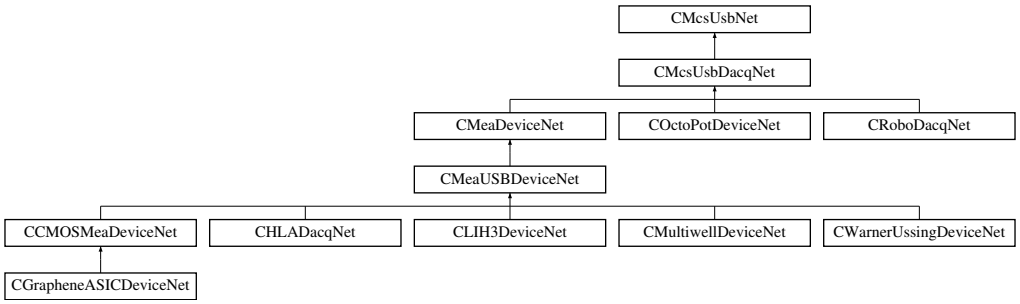
11.56.2.18 SetHWRevisionEeprom()

```
void SetHWRevisionEeprom (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short revision )
```

11.57 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, MEA2100DigitalSourceEnumNet source, int bitnumber_offset)

  Sets the function/source of an digital output bit.
- void SetDigitalSource (DigitalTargetEnumNet digitaltarget, int32_t NrChannel, MultiwellDigitalSourceEnumNet source, int bitnumber_offset)

  Sets the function/source of an digital output bit.
- void SetDigitalSource (DigitalTargetEnumNet digitaltarget, int32_t NrChannel, CMOSMEA5000DigitalSourceEnumNet source, int bitnumber_offset)

  Sets the function/source of an digital output bit.
- void SetDigitalSource (DigitalTargetEnumNet digitaltarget, int32_t NrChannel, W2100DigitalSourceEnumNet source, int bitnumber_offset)

Sets the function/source of an digital output bit.

- void SetDigitalSource (DigitalTargetEnumNet digitaltarget, int32_t NrChannel, SCUDigitalSourceEnumNet source, int bitnumber_offset)

  Sets the function/source of an digital output bit.
- void SetDigitalSource (DigitalTargetEnumNet digitaltarget, int32_t NrChannel, MEA2100_256DigitalSourceEnumNet source, int bitnumber_offset)

  Sets the function/source of an digital output bit.
- template<typename digitalsourceenum >
  void SetDigitalSource (DigitalTargetEnumNet digitaltarget, int32_t NrChannel, DigitalSource< digitalsourceenum >^ source, int bitnumber_offset)

  Sets the function/source of an digital output bit.
- void GetDigitalSource (DigitalTargetEnumNet digitaltarget, int32_t NrChannel, [System::Runtime::InteropServices::Out]MEA2100DigitalSourceEnumNet% source, [System::Runtime::InteropServices::Out]int% bitnumber_offset)

  Gets the function/source of an digital output bit.
- void GetDigitalSource (DigitalTargetEnumNet digitaltarget, int32_t NrChannel, [System::Runtime::InteropServices::Out]MultiwellDigitalSourceEnumNet% source, [System::Runtime::InteropServices::Out]int% bitnumber_offset)

  Gets the function/source of an digital output bit.
- void GetDigitalSource (DigitalTargetEnumNet digitaltarget, int32_t NrChannel, [System::Runtime::InteropServices::Out]CMOSMEA5000DigitalSourceEnumNet% source, [System::Runtime::InteropServices::Out]int% bitnumber_offset)

  Gets the function/source of an digital output bit.
- void GetDigitalSource (DigitalTargetEnumNet digitaltarget, int32_t NrChannel, [System::Runtime::InteropServices::Out]W2100DigitalSourceEnumNet% source, [System::Runtime::InteropServices::Out]int% bitnumber_offset)

  Gets the function/source of an digital output bit.
- void GetDigitalSource (DigitalTargetEnumNet digitaltarget, int32_t NrChannel, [System::Runtime::InteropServices::Out]SCUDigitalSourceEnumNet% source, [System::Runtime::InteropServices::Out]int% bitnumber_offset)

  Gets the function/source of an digital output bit.
- void GetDigitalSource (DigitalTargetEnumNet digitaltarget, int32_t NrChannel, [System::Runtime::InteropServices::Out]MEA2100_256DigitalSourceEnumNet% source, [System::Runtime::InteropServices::Out]int% bitnumber_offset)

  Gets the function/source of an digital output bit.
- template<typename digitalsourceenum >
  void GetDigitalSource (DigitalTargetEnumNet digitaltarget, int32_t NrChannel, [System::Runtime::InteropServices::Out]DigitalSource< digitalsourceenum >^% source, [System::Runtime::InteropServices::Out]int% bitnumber_offset)

  Gets the function/source of an digital output bit.
- virtual AdapterTypeEnumNet GetAdapterType ()

  Gets the adapter which is connected to the MEA2100 device.
- virtual MeaLayoutEnumNet GetMeaLayout ()

  Gets the MEA layout which is connected to the MEA2100 device.
- virtual uint32_t GetAdcDataFormat (uint32_t virtualDevice)

  Gets the ADC data format, 16 means 16 bits, 24 means 24 bits, 32 means 32 bits.
- virtual uint32_t GetAnalogValueUnit (uint32_t virtualDevice, DacqGroupChannelEnumNet group, [System::Runtime::InteropServices::Out] AnalogUnitEnumNet% unit)
- virtual uint32_t GetResolutionPerDigit (uint32_t virtualDevice, DacqGroupChannelEnumNet group, [System::Runtime::InteropServices::Out] int% res, [System::Runtime::InteropServices::Out] int% resUnit)
- virtual uint32_t GetAdcZero (uint32_t virtualDevice, DacqGroupChannelEnumNet group, [System::Runtime::InteropServices::Out] int% adcz)
- virtual uint32_t GetHardwareMinRange (uint32_t virtualDevice, DacqGroupChannelEnumNet group, [System::Runtime::InteropServices::Out] int% r, [System::Runtime::InteropServices::Out] int% rUnit)

- virtual uint32_t GetHardwareMaxRange (uint32_t virtualDevice, DacqGroupChannelEnumNet group, [System::Runtime::InteropServices::Out] int% r, [System::Runtime::InteropServices::Out] int% rUnit)
- virtual uint32_t GetDataFormat (uint32_t virtualDevice, DacqGroupChannelEnumNet group, [System::Runtime::InteropServices::Out] int% numberOfBits)
- virtual uint32_t GetNumberOfDataBits (uint32_t virtualDevice, DacqGroupChannelEnumNet group, [System::Runtime::InteropServices::Out] int% numberOfBits)

  Get the real number of data bits.
- virtual void SetSamplerate (int32_t rate, unsigned int oversample, unsigned int virtualDevice)

  Sets the sampling frequency of the device.
- virtual int32_t GetSamplerate (unsigned int virtualDevice)

  Gets the sampling frequency of the device.
- virtual 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 uint32_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 uint32_t data format
- virtual array< array< uint16_t >^> ^ ChannelBlock_ReadAsFrameArrayUI16 (int handle, int frames, [System::Runtime::InteropServices::Out]int % frames_ret)

  Read data from a FIFO queue as array of uint16_t data frame arrays
- virtual array< array< uint16_t >^> ^ ChannelBlock_ReadAsFrameArrayUI16 (int handle, int queue, int frames, [System::Runtime::InteropServices::Out]int % frames_ret)

  Read data from a FIFO queue as array of uint16_t data frame arrays
- virtual array< array< int16_t >^> ^ ChannelBlock_ReadAsFrameArrayI16 (int handle, int frames, [System↩ ::Runtime::InteropServices::Out]int % frames_ret)

  Read data from a FIFO queue as array of uint16_t data frame arrays
- virtual array< array< int16_t >^> ^ ChannelBlock_ReadAsFrameArrayI16 (int handle, int queue, int frames, [System::Runtime::InteropServices::Out]int % frames_ret)

  Read data from a FIFO queue as array of uint16_t data frame arrays
- virtual array< array< uint32_t >^> ^ ChannelBlock_ReadAsFrameArrayUI32 (int handle, int frames, [System::Runtime::InteropServices::Out]int % frames_ret)

Read data from a FIFO queue as array of uint16_t data frame arrays

- virtual array< array< uint32_t >^>^ ChannelBlock_ReadAsFrameArrayUI32 (int handle, int queue, int frames, [System::Runtime::InteropServices::Out]int % frames_ret)

  Read data from a FIFO queue as array of uint16_t data frame arrays
- virtual array< array< int32_t >^>^ ChannelBlock_ReadAsFrameArrayI32 (int handle, int frames, [System::Runtime::InteropServices::Out]int % frames_ret)

  Read data from a FIFO queue as array of uint16_t data frame arrays
- virtual array< array< int32_t >^>^ ChannelBlock_ReadAsFrameArrayI32 (int handle, int queue, int frames, [System::Runtime::InteropServices::Out]int % frames_ret)

  Read data from a FIFO queue as array of uint16_t data frame arrays
- virtual System::Collections::Generic::Dictionary< int, array< uint16_t >^>^ ChannelBlock_ReadFramesDictUI16 (int handle, int frames, [System::Runtime::InteropServices::Out]int % frames_ret)

  Read data from a FIFO queue in uint16_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number
- virtual System::Collections::Generic::Dictionary< int, array< int16_t >^>^ ChannelBlock_ReadFramesDictI16 (int handle, int frames, [System::Runtime::InteropServices::Out]int % frames_ret)

  Read data from a FIFO queue in int16_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number
- virtual System::Collections::Generic::Dictionary< int, array< uint32_t >^>^ ChannelBlock_ReadFramesDictUI32 (int handle, int frames, [System::Runtime::InteropServices::Out]int % frames_ret)

  Read data from a FIFO queue in uint32_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number
- virtual System::Collections::Generic::Dictionary< int, array< int32_t >^>^ ChannelBlock_ReadFramesDictI32 (int handle, int frames, [System::Runtime::InteropServices::Out]int % frames_ret)

  Read data from a FIFO queue in int32_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number
- virtual System::Collections::Generic::Dictionary< int, array< uint16_t >^>^ GetGroupChannelDataUI16 (DacqGroupChannelEnumNet group, int frames, [System::Runtime::InteropServices::Out]int % frames_ret)

  Read data from a FIFO queue in uint16_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number
- virtual System::Collections::Generic::Dictionary< int, array< int16_t >^>^ GetGroupChannelDataI16 (DacqGroupChannelEnumNet group, int frames, [System::Runtime::InteropServices::Out]int % frames_ret)

  Read data from a FIFO queue in int16_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number
- virtual System::Collections::Generic::Dictionary< int, array< uint32_t >^>^ GetGroupChannelDataUI32 (DacqGroupChannelEnumNet group, int frames, [System::Runtime::InteropServices::Out]int % frames_ret)

  Read data from a FIFO queue in uint32_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number
- virtual System::Collections::Generic::Dictionary< int, array< int32_t >^>^ GetGroupChannelDataI32 (DacqGroupChannelEnumNet group, int frames, [System::Runtime::InteropServices::Out]int % frames_ret)

  Read data from a FIFO queue in int32_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number
- void SetupGroupDacqQueue (int queuesize, int threshold)
- 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

11.57.1 Detailed Description

Base class for data acquisition devices.

11.57.2 Constructor & Destructor Documentation

11.57.2.1 CMcsUsbDacqNet() CMcsUsbDacqNet ( )

11.57.2.2 ~CMcsUsbDacqNet() ~CMcsUsbDacqNet ( )

11.57.3 Member Function Documentation

11.57.3.1 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

nByteOffset	Number of bytes to start with.

Parameters

nChannelOffset	Number of channel to start with (counted in samplesize bytes).

Parameters

selectedChannels	List of channels to be collected in the FIFO.

Parameters

queuesize	Size of sample frames the FIFO can hold.

Parameters

threshold	Number of sample frames the FIFO must acquire before the callback function is called.

Parameters

samplesize	size of the datawords, either 16 or 32bit.

Returns

The handle to the Queue.

11.57.3.2 AddSelectedChannelsQueue() [2/4]

```
virtual int AddSelectedChannelsQueue (
        int nByteOffset,
        int nChannelOffset,
        array< bool >^ selectedChannels,
```

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

11.57.3.3 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

nByteOffset	Number of bytes to start with.

Parameters

nChannelOffset	Number of channel to start with (counted in samplesize bytes).

Parameters

nChannels	Number of channels to be collected in the FIFO.

Parameters

queuesize	Size of sample frames the FIFO can hold.

Parameters

threshold	Number of sample frames the FIFO must acquire before the callback function is called.

Parameters

samplesize	size of the datawords, either 16 or 32bit.

Returns

The handle to the Queue.

11.57.3.4 AddSelectedChannelsQueue() [4/4]

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

11.57.3.5 ChannelBlock_AvailFrames() [1/2]

```
virtual uint32_t ChannelBlock_AvailFrames (
    int handle )  [virtual]
```

Get the number of sample frames already available in the FIFO.

Parameters

handle	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.

11.57.3.6 ChannelBlock_AvailFrames() [2/2]

```
virtual uint32_t ChannelBlock_AvailFrames (
    int handle,
    int queue )  [virtual]
```

11.57.3.7 ChannelBlock_ReadAsFrameArrayI16() [1/2] virtual array<array<int16_t>^> ^ ChannelBlock_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

handle	Handle of the FIFO queue. Zero when the SetSelectedData call was used.

Parameters

frames	Number of sample frames to read.

Parameters

frames_ret	Number of sample frames which were read, might be smaller than frames.

Returns

Array of int16_t frame arrays.

11.57.3.8 ChannelBlock_ReadAsFrameArrayI16() [2/2] virtual array<array<int16_t>^> ^ ChannelBlock_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

handle	Handle of the FIFO queue. Zero when the SetSelectedData call was used.

Parameters

queue	Number of the sub queue.
frames	Number of sample frames to read.

Parameters

frames_ret	Number of sample frames which were read, might be smaller than frames.

Returns

Array of int16_t frame arrays.

11.57.3.9 ChannelBlock_ReadAsFrameArrayI32() [1/2]

```
virtual array<array<int32_t>^> ^ ChannelBlock_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

handle	Handle of the FIFO queue. Zero when the SetSelectedData call was used.

Parameters

frames	Number of sample frames to read.

Parameters

frames_ret	Number of sample frames which were read, might be smaller than frames.

Returns

Array of int32_t frame arrays.

11.57.3.10 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

handle	Handle of the FIFO queue. Zero when the SetSelectedData call was used.
queue	Number of the sub queue.
frames	Number of sample frames to read.

Parameters

frames_ret	Number of sample frames which were read, might be smaller than frames.

Returns

Array of int32_t frame arrays.

11.57.3.11 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

handle	Handle of the FIFO queue. Zero when the SetSelectedData call was used.

Parameters

frames	Number of sample frames to read.

Parameters

frames_ret	Number of sample frames which were read, might be smaller than frames.

Returns

Array of uint16_t frame arrays.

11.57.3.12 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

handle	Handle of the FIFO queue. Zero when the SetSelectedData call was used.

Parameters

queue	Number of the sub queue.
frames	Number of sample frames to read.

Parameters

frames_ret	Number of sample frames which were read, might be smaller than frames.

Returns

Array of uint16_t frame arrays.

11.57.3.13 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

handle	Handle of the FIFO queue. Zero when the SetSelectedData call was used.

Parameters

frames	Number of sample frames to read.

Parameters

frames_ret	Number of sample frames which were read, might be smaller than frames.

Returns

Array of uint32_t frame arrays.

11.57.3.14 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

handle	Handle of the FIFO queue. Zero when the SetSelectedData call was used.

Parameters

queue	Number of the sub queue.
frames	Number of sample frames to read.

Parameters

frames_ret	Number of sample frames which were read, might be smaller than frames.

Returns

Array of uint32_t frame arrays.

11.57.3.15 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

handle	Handle of the FIFO queue. Zero when the SetSelectedChannelsQueue call was used.

Parameters

frames	Number of sample frames to read.
frames_ret	Number of sample frames which were read, might be smaller than frames.

Returns

Dictonary of int16_t arrays and hardware channel as key.

11.57.3.16 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

handle	Handle of the FIFO queue. Zero when the SetSelectedChannelsQueue call was used.

Parameters

frames	Number of sample frames to read.
frames_ret	Number of sample frames which were read, might be smaller than frames.

Returns

Dictonary of int32_t arrays and hardware channel as key.

11.57.3.17 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

handle	Handle of the FIFO queue. Zero when the SetSelectedChannelsQueue call was used.

Parameters

frames	Number of sample frames to read.
frames_ret	Number of sample frames which were read, might be smaller than frames.

Returns

Dictonary of uint16_t arrays and hardware channel as key.

11.57.3.18 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

handle	Handle of the FIFO queue. Zero when the SetSelectedChannelsQueue call was used.

Parameters

frames	Number of sample frames to read.
frames_ret	Number of sample frames which were read, might be smaller than frames.

Returns

Dictonary of uint32_t arrays and hardware channel as key.

11.57.3.19 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

handle	Handle of the FIFO queue. Either zero when the SetSelectedData call was used or the channel number.

Parameters

buffer	Buffer to put the data from the device in.
frames_pos	Position in buffer where to put the data.
frames	Number of sample frames to read.

Parameters

frames_ret	Number of sample frames which were read, might be smaller than frames.

11.57.3.20 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

handle	Handle of the FIFO queue. Either zero when the SetSelectedData call was used or the channel number.

Parameters

frames	Number of sample frames to read.

Parameters

frames_ret	Number of sample frames which were read, might be smaller than frames.

11.57.3.21 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

handle	Handle of the FIFO queue. Either zero when the SetSelectedData call was used or the channel number.

Parameters

buffer	Buffer to put the data from the device in.
frames_pos	Position in buffer where to put the data.
frames	Number of sample frames to read.

Parameters

frames_ret	Number of sample frames which were read, might be smaller than frames.

11.57.3.22 ChannelBlock_ReadFramesI32() [2/2]

```
virtual array<int32_t> ^ ChannelBlock_Read↩
FramesI32 (
    int handle,
    int frames,
    [System::Runtime::InteropServices::Out] int % frames_ret )  [virtual]
```

Read data from a FIFO queue in uint32_t data format

Parameters

handle	Handle of the FIFO queue. Either zero when the SetSelectedData call was used or the channel number.

Parameters

frames	Number of sample frames to read.

Parameters

frames_ret	Number of sample frames which were read, might be smaller than frames.

11.57.3.23 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

handle	Handle of the FIFO queue. Either zero when the SetSelectedData call was used or the channel number.
buffer	Buffer to put the data from the device in.
frames_pos	Position in buffer where to put the data.
frames	Number of sample frames to read.

Parameters

frames_ret	Number of sample frames which were read, might be smaller than frames.

11.57.3.24 ChannelBlock_ReadFramesUI16() [2/2]

```
virtual array<uint16_t> ^ ChannelBlock_Read↩
FramesUI16 (
        int handle,
        int frames,
        [System::Runtime::InteropServices::Out] int % frames_ret )  [virtual]
```

Read data from a FIFO queue in uint16_t data format

Parameters

handle	Handle of the FIFO queue. Either zero when the SetSelectedData call was used or the channel number.

Parameters

frames	Number of sample frames to read.

Parameters

frames_ret	Number of sample frames which were read, might be smaller than frames.

Returns

Array of data from the device.

11.57.3.25 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

handle	Handle of the FIFO queue. Either zero when the SetSelectedData call was used or the channel number.

Parameters

buffer	Buffer to put the data from the device in.
frames_pos	Position in buffer where to put the data.
frames	Number of sample frames to read.

Parameters

frames_ret	Number of sample frames which were read, might be smaller than frames.

11.57.3.26 ChannelBlock_ReadFramesUI32() [2/2]

```
virtual array<uint32_t> ^ ChannelBlock_ReadFramesUI32 (
            int handle,
            int frames,
            [System::Runtime::InteropServices::Out] int % frames_ret )  [virtual]
```

Read data from a FIFO queue in uint32_t data format

Parameters

handle	Handle of the FIFO queue. Either zero when the SetSelectedData call was used or the channel number.

Parameters

frames	Number of sample frames to read.

Parameters

frames_ret	Number of sample frames which were read, might be smaller than frames.

11.57.3.27 ClearBuffers()

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

11.57.3.28 CMcsUsbDacqNet::GetFilterProperties()

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

11.57.3.29 GetAdapterType() virtual AdapterTypeEnumNet GetAdapterType ( ) [virtual]

Gets the adapter which is connected to the MEA2100 device.

Returns

AdapterTypeEnumNet which enumerates the possible adapters.

11.57.3.30 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.

11.57.3.31 GetAdcZero() virtual uint32_t GetAdcZero (uint32_t *virtualDevice*, DacqGroupChannelEnumNet *group*, [System::Runtime::InteropServices::Out] int% *adcz*) [virtual]

11.57.3.32 GetAnalogValueUnit() virtual uint32_t GetAnalogValueUnit (uint32_t *virtualDevice*, DacqGroupChannelEnumNet *group*, [System::Runtime::InteropServices::Out] AnalogUnitEnumNet% *unit*) [virtual]

11.57.3.33 GetChannelDataFillSize() int GetChannelDataFillSize ( )

11.57.3.34 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]

11.57.3.35 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.

11.57.3.36 GetDataFormat() virtual uint32_t GetDataFormat (uint32_t *virtualDevice*, DacqGroupChannelEnumNet *group*, [System::Runtime::InteropServices::Out] int% *numberOfBits*) [virtual]

11.57.3.37 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

virtualDevice	Virtual device to use.

Returns

DataModeEnumNet which enumerates the possible data modes.

11.57.3.38 GetDigitalSource() [1/7] void GetDigitalSource (DigitalTargetEnumNet *digitaltarget*, int32_t *NrChannel*, [System::Runtime::InteropServices::Out] CMOSMEA5000DigitalSourceEnumNet% *source*, [System::Runtime::InteropServices::Out] int% *bitnumber_offset*)

Gets the function/source of an digital output bit.

This overload is for the CMOSMEA5000 device.

Parameters

digitaltarget	The digital target to query.
NrChannel	The channel/bit of target to query.
source	The source/function assignd to the digital target.
bitnumber_offset	An offset / bit number with the source/function.

11.57.3.39 GetDigitalSource() [2/7]

```
void GetDigitalSource (
            DigitalTargetEnumNet digitaltarget,
            int32_t NrChannel,
            [System::Runtime::InteropServices::Out] DigitalSource< digitalsourceenum >^% source,
            [System::Runtime::InteropServices::Out] int% bitnumber_offset )
```

Gets the function/source of an digital output bit.

This is the templated generic implementation.

Parameters

digitaltarget	The digital target to query.
NrChannel	The channel/bit of target to query.
source	The source/function assignd to the digital target.
bitnumber_offset	An offset / bit number with the source/function.

11.57.3.40 GetDigitalSource() [3/7]

```
void GetDigitalSource (
            DigitalTargetEnumNet digitaltarget,
            int32_t NrChannel,
            [System::Runtime::InteropServices::Out] MEA2100_256DigitalSourceEnumNet% source,
            [System::Runtime::InteropServices::Out] int% bitnumber_offset )
```

Gets the function/source of an digital output bit.

This overload is for the MEA2100-256 device.

Parameters

digitaltarget	The digital target to query.
NrChannel	The channel/bit of target to query.
source	The source/function assignd to the digital target.
bitnumber_offset	An offset / bit number with the source/function.

11.57.3.41 GetDigitalSource() [4/7]

```
void GetDigitalSource (
            DigitalTargetEnumNet digitaltarget,
            int32_t NrChannel,
            [System::Runtime::InteropServices::Out] MEA2100DigitalSourceEnumNet% source,
            [System::Runtime::InteropServices::Out] int% bitnumber_offset )
```

Gets the function/source of an digital output bit.

This overload is for the MEA2100 device.

Parameters

digitaltarget	The digital target to query.
NrChannel	The channel/bit of target to query.
source	The source/function assignd to the digital target.
bitnumber_offset	An offset / bit number with the source/function.

11.57.3.42 GetDigitalSource() [5/7]

```
void GetDigitalSource (
    DigitalTargetEnumNet digitaltarget,
    int32_t NrChannel,
    [System::Runtime::InteropServices::Out] MultiwellDigitalSourceEnumNet% source,
    [System::Runtime::InteropServices::Out] int% bitnumber_offset )
```

Gets the function/source of an digital output bit.

This overload is for the Multiwell device.

Parameters

digitaltarget	The digital target to query.
NrChannel	The channel/bit of target to query.
source	The source/function assignd to the digital target.
bitnumber_offset	An offset / bit number with the source/function.

11.57.3.43 GetDigitalSource() [6/7]

```
void GetDigitalSource (
    DigitalTargetEnumNet digitaltarget,
    int32_t NrChannel,
    [System::Runtime::InteropServices::Out] SCUDigitalSourceEnumNet% source,
    [System::Runtime::InteropServices::Out] int% bitnumber_offset )
```

Gets the function/source of an digital output bit.

This overload is for the SCU device.

Parameters

digitaltarget	The digital target to query.
NrChannel	The channel/bit of target to query.
source	The source/function assignd to the digital target.
bitnumber_offset	An offset / bit number with the source/function.

11.57.3.44 GetDigitalSource() [7/7]

```
void GetDigitalSource (
    DigitalTargetEnumNet digitaltarget,
```

```
int32_t NrChannel,
[System::Runtime::InteropServices::Out] W2100DigitalSourceEnumNet% source,
[System::Runtime::InteropServices::Out] int% bitnumber_offset )
```

Gets the function/source of an digital output bit.

This overload is for the W2100 device.

Parameters

digitaltarget	The digital target to query.
NrChannel	The channel/bit of target to query.
source	The source/function assignd to the digital target.
bitnumber_offset	An offset / bit number with the source/function.

11.57.3.45 GetErrorMessage()

```
uint32_t GetErrorMessage (
[System::Runtime::InteropServices::Out] String^% errorString,
[System::Runtime::InteropServices::Out] int% info )
```

11.57.3.46 GetFilterProperty()

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

11.57.3.47 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

group	Group selector supported by the device.

Parameters

frames	Number of sample frames to read.
frames_ret	Number of sample frames which were read, might be smaller than frames.

Returns

Dictonary of int16_t arrays and hardware channel as key.

11.57.3.48 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

group	Group selector supported by the device.

Parameters

frames	Number of sample frames to read.
frames_ret	Number of sample frames which were read, might be smaller than frames.

Returns

Dictonary of int32_t arrays and hardware channel as key.

11.57.3.49 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

group	Group selector supported by the device.

Parameters

frames	Number of sample frames to read.
frames_ret	Number of sample frames which were read, might be smaller than frames.

Returns

Dictonary of uint16_t arrays and hardware channel as key.

11.57.3.50 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

group	Group selector supported by the device.

Parameters

frames	Number of sample frames to read.
frames_ret	Number of sample frames which were read, might be smaller than frames.

Returns

Dictonary of uint32_t arrays and hardware channel as key.

11.57.3.51 GetHardwareMaxRange()

```
virtual uint32_t GetHardwareMaxRange (
            uint32_t virtualDevice,
            DacqGroupChannelEnumNet group,
            [System::Runtime::InteropServices::Out] int% r,
            [System::Runtime::InteropServices::Out] int% rUnit )  [virtual]
```

11.57.3.52 GetHardwareMinRange() virtual uint32_t GetHardwareMinRange (uint32_t *virtualDevice*, DacqGroupChannelEnumNet *group*, [System::Runtime::InteropServices::Out] int% *r*, [System::Runtime::InteropServices::Out] int% *rUnit*) [virtual]

11.57.3.53 GetMaxSamplingFrequency() virtual uint32_t GetMaxSamplingFrequency (int *virtualDevice*) [virtual]

Gets the maximal sampling frequency of the device.

Returns

Sampling frequency in Hz.

11.57.3.54 GetMeaLayout() virtual MeaLayoutEnumNet GetMeaLayout ( ) [virtual]

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

Returns

MeaLayoutEnumNet which enumerates the MEA types.

11.57.3.55 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.

11.57.3.56 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

11.57.3.57 GetPoti() virtual uint32_t GetPoti (uint32_t *channel*, [System::Runtime::InteropServices::Out] uint32_t% *value*) [virtual]

11.57.3.58 GetResolutionPerDigit() virtual uint32_t GetResolutionPerDigit (uint32_t *virtualDevice*, DacqGroupChannelEnumNet *group*, [System::Runtime::InteropServices::Out] int% *res*, [System::Runtime::InteropServices::Out] int% *resUnit*) [virtual]

11.57.3.59 GetSamplerate() virtual int32_t GetSamplerate (unsigned int *virtualDevice*) [virtual]

Gets the sampling frequency of the device.

Returns

Sampling frequency in Hz.

11.57.3.60 GetVoltageRangeIndex() virtual uint32_t GetVoltageRangeIndex (unsigned int *virtualDevice*) [virtual]

11.57.3.61 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.

11.57.3.62 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.

11.57.3.63 HWInfo() CHWInfo ^ HWInfo ( )

11.57.3.64 SendStartDacq() [1/2] virtual void SendStartDacq ( ) [virtual]

Start sampling.

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

Start sampling.

Parameters

VirtualDacqMap	

11.57.3.66 SendStartStgAndDacq() virtual void SendStartStgAndDacq (uint32_t *trigger_map*, int *VirtualDacqMap*) [virtual]

Start sampling together with the STG.

Parameters

trigger_map	
VirtualDacqMap	

11.57.3.67 SendStopDacq() [1/2] virtual void SendStopDacq ( ) [virtual]

Stop sampling.

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

Stop sampling.

Parameters

VirtualDacqMap	

11.57.3.69 SendStopStgAndDacq() virtual void SendStopStgAndDacq (uint32_t *trigger_map*, int *VirtualDacqMap*) [virtual]

Stop sampling together with the STG.

Parameters

trigger_map	

11.57.3.70 SendStopStgAndDacqWithOptions() virtual void SendStopStgAndDacqWithOptions (uint32_t *trigger_map*, int *VirtualDacqMap*, int *options*) [virtual]

Stop sampling together with the STG and options.

Parameters

trigger_map	

Parameters

options	

Parameters

VirtualDacqMap	

11.57.3.71 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

dataMode	DataModeEnumNet enumerates the possible data modes.
virtualDevice	Virtual device to use.

11.57.3.72 SetDigitalSource() [1/7] void SetDigitalSource (DigitalTargetEnumNet *digitaltarget*, int32_t *NrChannel*, CMOSMEA5000DigitalSourceEnumNet *source*, int *bitnumber_offset*)

Sets the function/source of an digital output bit.

This overload is for the CMOSMEA5000 device.

Parameters

digitaltarget	The digital target to change.
NrChannel	The channel/bit of target to change.
source	The source/function to assign to the digital target.
bitnumber_offset	An offset / bit number with the source/function.

11.57.3.73 SetDigitalSource() [2/7] void SetDigitalSource (DigitalTargetEnumNet *digitaltarget*, int32_t *NrChannel*, DigitalSource< digitalsourceenum >^ *source*, int *bitnumber_offset*)

Sets the function/source of an digital output bit.

This is the templated generic implementation.

Parameters

digitaltarget	The digital target to change.
NrChannel	The channel/bit of target to change.
source	The source/function to assign to the digital target.
bitnumber_offset	An offset / bit number with the source/function.

11.57.3.74 SetDigitalSource() [3/7]

```
void SetDigitalSource (
    DigitalTargetEnumNet digitaltarget,
    int32_t NrChannel,
    MEA2100_256DigitalSourceEnumNet source,
    int bitnumber_offset )
```

Sets the function/source of an digital output bit.

This overload is for the MEA2100-256 device.

Parameters

digitaltarget	The digital target to change.
NrChannel	The channel/bit of target to change.
source	The source/function to assign to the digital target.
bitnumber_offset	An offset / bit number with the source/function.

11.57.3.75 SetDigitalSource() [4/7]

```
void SetDigitalSource (
    DigitalTargetEnumNet digitaltarget,
    int32_t NrChannel,
    MEA2100DigitalSourceEnumNet source,
    int bitnumber_offset )
```

Sets the function/source of an digital output bit.

This overload is for the MEA2100 device.

Parameters

digitaltarget	The digital target to change.
NrChannel	The channel/bit of target to change.
source	The source/function to assign to the digital target.
bitnumber_offset	An offset / bit number with the source/function.

11.57.3.76 SetDigitalSource() [5/7]

```
void SetDigitalSource (
    DigitalTargetEnumNet digitaltarget,
    int32_t NrChannel,
    MultiwellDigitalSourceEnumNet source,
    int bitnumber_offset )
```

Sets the function/source of an digital output bit.

This overload is for the Multiwell device.

Parameters

digitaltarget	The digital target to change.
NrChannel	The channel/bit of target to change.
source	The source/function to assign to the digital target.
bitnumber_offset	An offset / bit number with the source/function.

11.57.3.77 SetDigitalSource() [6/7] void SetDigitalSource (

```
DigitalTargetEnumNet digitaltarget,
int32_t NrChannel,
SCUDigitalSourceEnumNet source,
int bitnumber_offset )
```

Sets the function/source of an digital output bit.

This overload is for the SCU device.

Parameters

digitaltarget	The digital target to change.
NrChannel	The channel/bit of target to change.
source	The source/function to assign to the digital target.
bitnumber_offset	An offset / bit number with the source/function.

11.57.3.78 SetDigitalSource() [7/7] void SetDigitalSource (

```
DigitalTargetEnumNet digitaltarget,
int32_t NrChannel,
W2100DigitalSourceEnumNet source,
int bitnumber_offset )
```

Sets the function/source of an digital output bit.

This overload is for the W2100 device.

Parameters

digitaltarget	The digital target to change.
NrChannel	The channel/bit of target to change.
source	The source/function to assign to the digital target.
bitnumber_offset	An offset / bit number with the source/function.

11.57.3.79 SetPoti() virtual uint32_t SetPoti (

```
uint32_t channel,
uint32_t value,
bool write_nvram ) [virtual]
```

11.57.3.80 SetSamplerate() virtual void SetSamplerate (

```
int32_t rate,
```

```
            unsigned int oversample,
            unsigned int virtualDevice )  [virtual]
```

Sets the sampling frequency of the device.

Parameters

rate	Sampling frequency in Hz.

11.57.3.81 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 devided 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

selectedChannels	List of channels to be collected in the FIFO.

Parameters

queuesize	Size of sample frames the FIFO can hold.

Parameters

threshold	Number of sample frames the FIFO must acquire before the callback function is called.

Parameters

samplesize	size of the datawords, either 16 or 32bit.

Parameters

ChannelsInBlock	value obtained from GetChannelsInBlock.

11.57.3.82 SetSelectedChannels() [2/4] virtual void SetSelectedChannels (

```
array< bool >^ selectedChannels,
int queuesize,
int threshold,
SampleSizeNet samplesize,
SampleDstSizeNet sampleDstSize,
int ChannelsInBlock )  [virtual]
```

11.57.3.83 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 devided 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

nChannels	Number of channels to be collected in the FIFO.

Parameters

queuesize	Size of sample frames the FIFO can hold.

Parameters

threshold	Number of samples frames the FIFO must acquire before the callback function is called.

Parameters

samplesize	size of the datawords, either 16 or 32bit.
ChannelsInBlock	value obtained from GetChannelsInBlock.

11.57.3.84 SetSelectedChannels() [4/4] virtual void SetSelectedChannels (

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

11.57.3.85 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

selectedChannels	List of channels to be collected in the FIFO.

Parameters

queuesize	Size of sample frames the FIFO can hold.

Parameters

threshold	Number of sample frames the FIFO must acquire before the callback function is called.

Parameters

samplesize	size of the datawords, either 16 or 32bit.

Parameters

ChannelsInBlock	value obtained from GetChannelsInBlock.

11.57.3.86 SetSelectedChannelsQueue() [2/4]

```
virtual void SetSelectedChannelsQueue (
    array< bool >^ selectedChannels,
    int queuesize,
    int threshold,
    SampleSizeNet samplesize,
    SampleDstSizeNet sampleDstSize,
    int ChannelsInBlock )  [virtual]
```

11.57.3.87 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

nChannels	Number of channels to be collected in the FIFO.

Parameters

queuesize	Size of sample frames the FIFO can hold.

Parameters

threshold	Number of sample frames the FIFO must acquire before the callback function is called.

Parameters

samplesize	size of the datawords, either 16 or 32bit.

Parameters

ChannelsInBlock	value obtained from GetChannelsInBlock.

11.57.3.88 SetSelectedChannelsQueue() [4/4]

```
virtual void SetSelectedChannelsQueue (
    int nChannels,
    int queuesize,
    int threshold,
    SampleSizeNet samplesize,
    SampleDstSizeNet sampleDstSize,
    int ChannelsInBlock )  [virtual]
```

11.57.3.89 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

selectedChannels	List of channels to be collected in the FIFO.

Parameters

queuesize	Size of sample frames the FIFO can hold.

Parameters

threshold	Number of sample frames the FIFO must acquire before the callback function is called.

Parameters

samplesize	size of the datawords, either 16 or 32bit.
ChannelsInBlock	value obtained from GetChannelsInBlock.

11.57.3.90 SetSelectedData() [2/4]

```
virtual void SetSelectedData (
    array< bool >^ selectedChannels,
    int queuesize,
    int threshold,
    SampleSizeNet samplesize,
    SampleDstSizeNet sampleDstSize,
    int ChannelsInBlock )  [virtual]
```

11.57.3.91 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

nChannels	Number of channels to be collected in the FIFO.

Parameters

queuesize	Size of sample frames the FIFO can hold.

Parameters

threshold	Number of sample frames the FIFO must acquire before the callback function is called.

Parameters

samplesize	size of the datawords, either 16 or 32bit.
ChannelsInBlock	value obtained from GetChannelsInBlock.

11.57.3.92 SetSelectedData() [4/4]

```
virtual void SetSelectedData (
    int nChannels,
    int queuesize,
    int threshold,
    SampleSizeNet samplesize,
    SampleDstSizeNet sampleDstSize,
    int ChannelsInBlock )  [virtual]
```

11.57.3.93 SetupGroupDacqQueue() [1/2]

```
void SetupGroupDacqQueue (
    int queuesize,
    int threshold )
```

11.57.3.94 SetupGroupDacqQueue() [2/2]

```
void SetupGroupDacqQueue (
    int queuesize,
    int threshold,
    unsigned int virtualDevice )
```

11.57.3.95 SetVoltageRangeByIndex()

```
virtual void SetVoltageRangeByIndex (
    int32_t voltageRangeIndex,
    unsigned int virtualDevice )  [virtual]
```

Sets the voltage range on devices which support multiple voltage ranges.

Parameters

voltageRangeIndex	Voltage Range to use as index, smaller values are larger voltage ranges.

11.57.3.96 SetVoltageRangeInMicroVolt()

```
virtual void SetVoltageRangeInMicroVolt (
    int32_t voltageRange,
    unsigned int virtualDevice ) [virtual]
```

Sets the voltage range on devices which support multiple voltage ranges.

Parameters

voltageRange	Voltage Range to use in µV.

This replaces SetVoltageRange, where the value of the range was in mV!

11.57.3.97 StartDacq() [1/4]

```
virtual void StartDacq ( ) [virtual]
```

Start the data acquisition thread and sampling.

11.57.3.98 StartDacq() [2/4]

```
virtual void StartDacq (
    int32_t timeout ) [virtual]
```

Start the data acquisition thread and sampling.

Parameters

timeout	Timeout in ms.

11.57.3.99 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

timeout	Timeout in ms.

Parameters

numSubmittedUsbBuffers	Number of USB Buffers that are simultaniously submitted.

Parameters

numUsbBuffers	Number of USB Buffers to use.

Parameters

packetsInUrb	Packets in each URB.

11.57.3.100 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

numSubmittedUsbBuffers	Number of USB Buffers that are simultaniously submitted.

Parameters

timeout	Timeout in ms.

Parameters

numUsbBuffers	Number of USB Buffers to use.

Parameters

packetsInUrb	Packets in each URB.

Parameters

virtualDevice	Virtual Device to start.

11.57.3.101 StartLoop() [1/4] virtual void StartLoop ( ) [virtual]

Start the data acquisition thread.

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

Start the data acquisition thread.

Parameters

timeout	Timeout in ms.

11.57.3.103 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

timeout	Timeout in ms.

Parameters

numSubmittedUsbBuffers	Number of USB Buffers that are simultaniously submitted.

Parameters

numUsbBuffers	Number of USB Buffers to use.

Parameters

packetsInUrb	Packets in each URB.

11.57.3.104 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

numSubmittedUsbBuffers	Number of USB Buffers that are simultaniously submitted.

Parameters

timeout	Timeout in ms.

Parameters

numUsbBuffers	Number of USB Buffers to use.

Parameters

packetsInUrb	Packets in each URB.

Parameters

virtualDevice	Virtual Device to start.

11.57.3.105 StopDacq() [1/2] virtual void StopDacq ( ) [virtual]

Stop the data acquisition thread and sampling.

11.57.3.106 StopDacq() [2/2] virtual void StopDacq (uint32_t *virtualDevice*) [virtual]

Stop the data acquisition thread and sampling.

Parameters

virtualDevice	Virtual Device to start.

11.57.3.107 StopLoop() virtual void StopLoop ( ) [virtual]

11.57.4 Member Data Documentation

11.57.4.1 Error_Callback_Aquisition_Stopped const int Error_Callback_Aquisition_Stopped = 0x200 [static]

11.57.4.2 Error_Callback_Data_lost const int Error_Callback_Data_lost = 5 [static]

11.57.4.3 Error_Callback_Frames_Lost const int Error_Callback_Frames_Lost = 4 [static]

11.57.4.4 Error_Callback_Packet_Error

```
const int Error_Callback_Packet_Error = 1  [static]
```

11.57.4.5 Error_Callback_Queue_Full

```
const int Error_Callback_Queue_Full = 0x100  [static]
```

11.57.4.6 Error_Callback_RingQueue_Full

```
const int Error_Callback_RingQueue_Full = 3  [static]
```

11.57.5 Property Documentation

11.57.5.1 Samplerate

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

The sampling frequency of the device in Hz.

11.57.6 Event Documentation

11.57.6.1 ChannelDataEvent

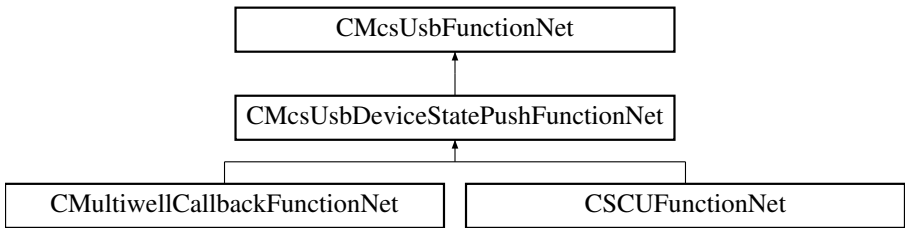
```
OnChannelData^ ChannelDataEvent  [add], [remove], [raise]
```

11.57.6.2 ErrorEvent

```
OnError^ ErrorEvent  [add], [remove], [raise]
```

11.58 CMcsUsbDeviceStatePushFunctionNet Class Reference

Inheritance diagram for CMcsUsbDeviceStatePushFunctionNet:



Public Member Functions

- void TriggerStatus ()

Protected Member Functions

- CMcsUsbDeviceStatePushFunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ p↩ Device)

Events

- OnMcsUsbDeviceState^ McsUsbDeviceStateEvent [add, remove, raise]

Additional Inherited Members

11.58.1 Constructor & Destructor Documentation

11.58.1.1 CMcsUsbDeviceStatePushFunctionNet() CMcsUsbDeviceStatePushFunctionNet (CMcsUsbNet^ *mcsusb*, CMcsUsbFunctionPointerContainer^ *pDevice*) [protected]

11.58.2 Member Function Documentation

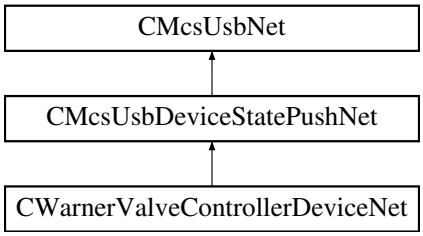
11.58.2.1 TriggerStatus() void TriggerStatus ( )

11.58.3 Event Documentation

11.58.3.1 McsUsbDeviceStateEvent OnMcsUsbDeviceState^ McsUsbDeviceStateEvent [add], [remove], [raise]

11.59 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

11.59.1 Constructor & Destructor Documentation

11.59.1.1 CMcsUsbDeviceStatePushNet() CMcsUsbDeviceStatePushNet (CMcsUsbPointerContainer^ *pDevice*) [protected]

11.59.2 Member Function Documentation

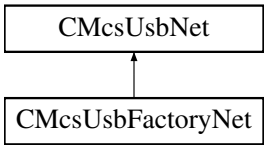
11.59.2.1 TriggerStatus() void TriggerStatus ( )

11.59.3 Event Documentation

11.59.3.1 McsUsbDeviceStateEvent OnMcsUsbDeviceState^ McsUsbDeviceStateEvent [add], [remove], [raise]

11.60 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 FX3MCSDataAddress = 0x40037E00
- static const uint32_t FX3MCSDataDeviceIdOffset = 0x4
- static const uint32_t FX3MCSDataVersionOffset = 0x8
- static const uint32_t FX3MCSDataIFB2ImageOffset = 0xC
- static const uint32_t FX3MCSDataIFB1ImageOffset = 0x2C

Additional Inherited Members

11.60.1 Constructor & Destructor Documentation

11.60.1.1 CMcsUsbFactoryNet() `CMcsUsbFactoryNet ( )`

11.60.1.2 ~CMcsUsbFactoryNet() `~CMcsUsbFactoryNet ( )`

11.60.2 Member Function Documentation

11.60.2.1 ChangeSerialNumber() `uint32_t ChangeSerialNumber ( String^ serial )`

11.60.2.2 Coldstart()

```
uint32_t Coldstart (
    CFirmwareDestinationNet dest )
```

11.60.2.3 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 )
```

11.60.2.4 DownloadFirmware()

```
array<uint8_t> ^ DownloadFirmware (
    CFirmwareDestinationNet Dest,
    uint32_t Address,
    uint32_t length )
```

11.60.2.5 FindFirmwareVersionMagicInBuffer()

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

11.60.2.6 GetChecksumFromFX3Image()

```
static uint32_t GetChecksumFromFX3Image (
    String^ FirmwareFile )  [static]
```

11.60.2.7 GetDestination() [1/2]

```
CFirmwareDestinationNet GetDestination (
    String^ Key )
```

11.60.2.8 GetDestination() [2/2]

```
CFirmwareDestinationNet GetDestination (
    unsigned int index )
```

11.60.2.9 GetDestinationDisplayLabel()

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

11.60.2.10 GetDestinationName() [1/2]

```
String ^ GetDestinationName (
    CFirmwareDestinationNet dest )
```

11.60.2.11 GetDestinationName() [2/2]

```
String ^ GetDestinationName (
    unsigned int index )
```

11.60.2.12 GetDestinationSerialNumber()

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

11.60.2.13 GetDestinationTargetAddress()

```
unsigned int GetDestinationTargetAddress (
    CFirmwareDestinationNet destination )
```

Gets the target base address for the destination.

Parameters

destination	The destination to be queried.

Returns

The base address as a 32 bit number, only the lower 16 bit represent the address.

11.60.2.14 GetFirmwareVersionFromFile() [1/2]

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

Retrives version info from a Firmware update file.

11.60.2.15 GetFirmwareVersionFromFile() [2/2]

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

11.60.2.16 GetFirmwareVersionFromHexFile()

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

11.60.2.17 GetNumDestinations() unsigned int GetNumDestinations ( )

11.60.2.18 GetUSBDeviceIDFromFX3Image() static uint32_t GetUSBDeviceIDFromFX3Image (String^ *FirmwareFile*) [static]

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

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

11.60.2.21 GetXilinxFlashOffset() int32_t GetXilinxFlashOffset (CFirmwareDestinationNet *dest*)

11.60.2.22 GetXilinxFlashReadCommand() uint32_t GetXilinxFlashReadCommand (CFirmwareDestinationNet *dest*)

11.60.2.23 LoadUserFirmware() [1/2] bool LoadUserFirmware (String^ *FirmwareFile*, CMcsUsbListEntryNet^ *listEntry*)

Send the DSP Firmware to the MEA21 device.

Parameters

FirmwareFile	Filename of the DSP Firmware (∗.bin) file.

Parameters

listEntry	Device to use for the connection. See CMcsUsbListNet.

11.60.2.24 LoadUserFirmware() [2/2]

```
bool LoadUserFirmware (
    String^ FirmwareFile,
    CMcsUsbListEntryNet^ listEntry,
    uint32_t LockMask )
```

11.60.2.25 ReadBlockFromFlash() [1/2]

```
array<unsigned char> ^ ReadBlockFromFlash (
    unsigned int FPGA,
    unsigned int Address )
```

11.60.2.26 ReadBlockFromFlash() [2/2]

```
void ReadBlockFromFlash (
    unsigned int FPGA,
    unsigned int Address,
    array< unsigned char >^ buffer,
    int position )
```

11.60.2.27 ReadBlockFromIFBGlobalEEprom()

```
array<unsigned char> ^ ReadBlockFromIFBGlobal↩
EEprom (
    unsigned int Address )
```

11.60.2.28 ReadBlockFromNVMEM()

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

11.60.2.29 SetDestinationSerialNumber()

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

11.60.2.30 UpdateFirmware() [1/5]

```
bool UpdateFirmware (
    String^ FirmwareFile,
    CMcsUsbListEntryNet^ listEntry,
    CFirmwareDestinationNet dest )
```

Flashes a firmware file to the device.

Parameters

FirmwareFile	Filename of the Firmware file.

Parameters

listEntry	Device to use for the connection.

11.60.2.31 UpdateFirmware() [2/5] bool UpdateFirmware (

```
String^ FirmwareFile,
CMcsUsbListEntryNet^ listEntry,
CFirmwareDestinationNet dest,
bool SkipWait )
```

Flashes a firmware file to the device.

Parameters

FirmwareFile	Filename of the Firmware file.

Parameters

listEntry	Device to use for the connection.

11.60.2.32 UpdateFirmware() [3/5] bool UpdateFirmware (

```
String^ FirmwareFile,
CMcsUsbListEntryNet^ listEntry,
CFirmwareDestinationNet dest,
bool SkipWait,
uint32_t LockMask )
```

11.60.2.33 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

FirmwareFile	Filename of the Firmware file.

11.60.2.34 UpdateFirmware() [5/5]

```
bool UpdateFirmware (
String^ FirmwareFile,
CMcsUsbListEntryNet^ listEntry,
CFirmwareDestinationNet Dest,
OnUpdateFirmwareStatusChange^ deleg,
OnUpdateFirmwareProgress^ progress,
bool SkipWait,
unsigned int LockMask )
```

11.60.3 Member Data Documentation

11.60.3.1 FX3MCSDataAddress

```
const uint32_t FX3MCSDataAddress = 0x40037E00 [static]
```

11.60.3.2 FX3MCSDataDeviceIdOffset

```
const uint32_t FX3MCSDataDeviceIdOffset = 0x4 [static]
```

11.60.3.3 FX3MCSDataIFB1ImageOffset

```
const uint32_t FX3MCSDataIFB1ImageOffset = 0x2C [static]
```

11.60.3.4 FX3MCSDataIFB2ImageOffset

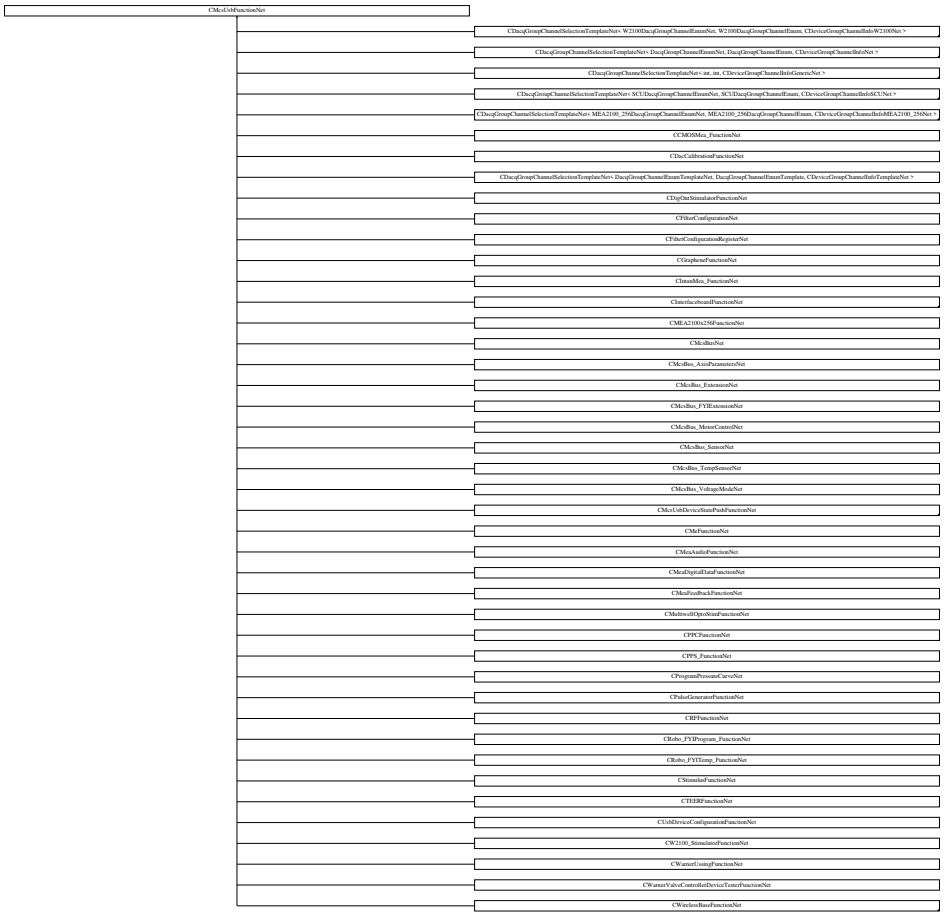
```
const uint32_t FX3MCSDataIFB2ImageOffset = 0xC [static]
```

11.60.3.5 FX3MCSDataVersionOffset

```
const uint32_t FX3MCSDataVersionOffset = 0x8 [static]
```

11.61 CMcsUsbFunctionNet Class Reference

Inheritance diagram for CMcsUsbFunctionNet:



Public Member Functions

- CMcsUsbFunctionNet (CMcsUsbNet^ mcsusb)
- virtual ~CMcsUsbFunctionNet (void)
- !CMcsUsbFunctionNet ()
- void ThrowCUsbExceptionNetOnError (uint32_t status)

Protected Member Functions

- CMcsUsbFunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ mcsusbfunction)

Protected Attributes

- CMcsUsbNet ^ m_pMcsUsb
- CMcsUsbFunction * m_pMcsUsbFunction

11.61.1 Constructor & Destructor Documentation

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

11.61.1.2 ~CMcsUsbFunctionNet() virtual ~CMcsUsbFunctionNet (void) [virtual]

11.61.1.3 "!CMcsUsbFunctionNet() !CMcsUsbFunctionNet ( )

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

11.61.2 Member Function Documentation

11.61.2.1 ThrowCUsbExceptionNetOnError() void ThrowCUsbExceptionNetOnError (uint32_t *status*)

11.61.3 Member Data Documentation

11.61.3.1 m_pMcsUsb CMcsUsbNet ^ m_pMcsUsb [protected]

11.61.3.2 m_pMcsUsbFunction CMcsUsbFunction* m_pMcsUsbFunction [protected]

11.62 CMcsUsbFunctionPointerContainer Class Reference

11.63 CMcsUsbListEntryNet Class Reference

McsUsbListEntryNet identifies a connected device.

Public Member Functions

- ~CMcsUsbListEntryNet ()
- virtual bool Equals (Object^ obj) override

  Checks weather two CMcsUsbListEntryNet represent the same USB device.
- void SetStringFormat (String ^ format)

  Specify the text the CMcsUsbListEntryNet.ToString() function should return. The special code N expands to the device name and S expands to the serial number of the device.
- virtual String ^ ToString () override

Static Public Member Functions

- static CMcsUsbListEntryNet ^ GetEntry ()

  Returns one CMcsUsbListEntryNet from the list of USB Devices connected to the computer.
- static CMcsUsbListEntryNet ^ GetEntry (DeviceEnumNet McsUsbDevice)

  Returns one CMcsUsbListEntryNet from the list of USB Devices connected to the computer.
- static CMcsUsbListEntryNet ^ GetEntry (DeviceEnumNet McsUsbDevice, unsigned int index)

  Returns one CMcsUsbListEntryNet from the list of USB Devices connected to the computer.
- static unsigned int GetEntryCount ()

  Returns the number of devices connected to the computer.
- static unsigned int GetEntryCount (DeviceEnumNet McsUsbDevice)

  Returns the number of devices connected to the computer.

Properties

- String^ Manufacturer `[get]`

  The Manufacturer ID of the device represented by this CMcsUsbListEntryNet.
- String^ Product `[get]`

  The Product ID of the device represented by this CMcsUsbListEntryNet.
- String^ DeviceName `[get]`

  The device name of the device represented by this CMcsUsbListEntryNet.
- String^ SerialNumber `[get]`

  The serial number of the device represented by this CMcsUsbListEntryNet.
- String^ HwVersion `[get]`

  The hardware revision of the device represented by this CMcsUsbListEntryNet.
- DeviceIdNet^ DeviceId `[get]`

11.63.1 Detailed Description

McsUsbListEntryNet identifies a connected device.

11.63.2 Constructor & Destructor Documentation

11.63.2.1 ~CMcsUsbListEntryNet() ~CMcsUsbListEntryNet ( )

11.63.3 Member Function Documentation

11.63.3.1 Equals() `virtual bool Equals ( Object^ obj ) [override], [virtual]`

Checks weather two CMcsUsbListEntryNet represent the same USB device.

Parameters

obj	The CMcsUsbListEntryNet to compare with.

11.63.3.2 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.

11.63.3.3 GetEntry() [2/3] static CMcsUsbListEntryNet ^ GetEntry (DeviceEnumNet *McsUsbDevice*) [static]

Returns one CMcsUsbListEntryNet from the list of USB Devices connected to the computer.

Parameters

McsUsbDevice	Specifies the type of devices to look for.

Returns

A CMcsUsbListEntryNet to be used to connect to the device.

11.63.3.4 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

McsUsbDevice	Specifies the type of devices to look for.

Parameters

index	number of the entry to use.

Returns

A CMcsUsbListEntryNet to be used to connect to the device.

11.63.3.5 GetEntryCount() [1/2] static unsigned int GetEntryCount ( ) [static]

Returns the number of devices connected to the computer.

Returns

The number of devices.

11.63.3.6 GetEntryCount() [2/2] static unsigned int GetEntryCount (DeviceEnumNet *McsUsbDevice*) [static]

Returns the number of devices connected to the computer.

Parameters

McsUsbDevice	Specifies the type of devices to look for.

Returns

The number of devices.

11.63.3.7 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

format	A String containing the format template. Default is "%N (%S)".

11.63.3.8 ToString() `virtual String ^ ToString ( ) [override], [virtual]`

11.63.4 Property Documentation

11.63.4.1 DeviceId `DeviceIdNet^ DeviceId [get]`

11.63.4.2 DeviceName `String^ DeviceName [get]`

The device name of the device represented by this CMcsUsbListEntryNet.

11.63.4.3 HwVersion `String^ HwVersion [get]`

The hardware revision of the device represented by this CMcsUsbListEntryNet.

11.63.4.4 Manufacturer `String^ Manufacturer [get]`

The Manufacturer ID of the device represented by this CMcsUsbListEntryNet.

11.63.4.5 Product `String^ Product [get]`

The Product ID of the device represented by this CMcsUsbListEntryNet.

11.63.4.6 SerialNumber `String^ SerialNumber [get]`

The serial number of the device represented by this CMcsUsbListEntryNet.

11.64 CMcsUsbListNet Class Reference

Class to handle a list of connected MCS USB devices.

Public Member Functions

- CMcsUsbListNet (DeviceEnumNet McsUsbDevice)

  Initializes a new instance of CMcsUsbListNet class.
- CMcsUsbListNet (array< DeviceIdNet^ >^ DeviceIdList)

  Initializes a new instance of CMcsUsbListNet class.
- ~CMcsUsbListNet ()

  Destructor: called by Dispose()
- !CMcsUsbListNet ()

  Finalizer: called by GC before collecting
- void SetStringFormat (String ^ format)

  Specify the text the CMcsUsbListEntryNet.ToString() function should return. The special code N expands to the device name and S expands to the serial number of the device.
- uint32_t GetNumberOfDevices ()

  Gets the number of devices currently in the list.
- CMcsUsbListEntryNet ^ GetUsbListEntry (unsigned int index)

  Returns one CMcsUsbListEntryNet from the list of USB Devices connected to the computer.
- array< CMcsUsbListEntryNet^ > ^ GetUsbListEntries ()

  Returns all entries from the list of USB Devices connected to the computer.
- bool IsDeviceTypeOf (CMcsUsbListEntryNet^ entry, DeviceEnumNet McsUsbDevice)

Properties

- uint32_t Count `[get]`

  Gets the number of devices currently in the list.

Events

- OnDeviceArrivalRemoval^ DeviceArrival
- OnDeviceArrivalRemoval^ DeviceRemoval

11.64.1 Detailed Description

Class to handle a list of connected MCS USB devices.

11.64.2 Constructor & Destructor Documentation

11.64.2.1 CMcsUsbListNet() [1/2] CMcsUsbListNet (DeviceEnumNet *McsUsbDevice*)

Initializes a new instance of CMcsUsbListNet class.

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

Initializes a new instance of CMcsUsbListNet class.

11.64.2.3 ~CMcsUsbListNet() ~CMcsUsbListNet ( )

Destructor: called by Dispose()

11.64.2.4 "!CMcsUsbListNet() !CMcsUsbListNet ( )

Finalizer: called by GC before collecting

11.64.3 Member Function Documentation

11.64.3.1 GetNumberOfDevices() uint32_t GetNumberOfDevices ( )

Gets the number of devices currently in the list.

Returns

The number of devices currently in the list.

11.64.3.2 GetUsbListEntries() array<CMcsUsbListEntryNet^> ^ GetUsbListEntries ( )

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

11.64.3.3 GetUsbListEntry() CMcsUsbListEntryNet ^ GetUsbListEntry (unsigned int *index*)

Returns one CMcsUsbListEntryNet from the list of USB Devices connected to the computer.

Parameters

index	number of the entry to use.

11.64.3.4 IsDeviceTypeOf() `bool IsDeviceTypeOf (`

```
CMcsUsbListEntryNet^ entry,
DeviceEnumNet McsUsbDevice )
```

11.64.3.5 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

format	A String containing the format template. Default is "%N (%S)".

11.64.4 Property Documentation

11.64.4.1 Count `uint32_t Count [get]`

Gets the number of devices currently in the list.

11.64.5 Event Documentation

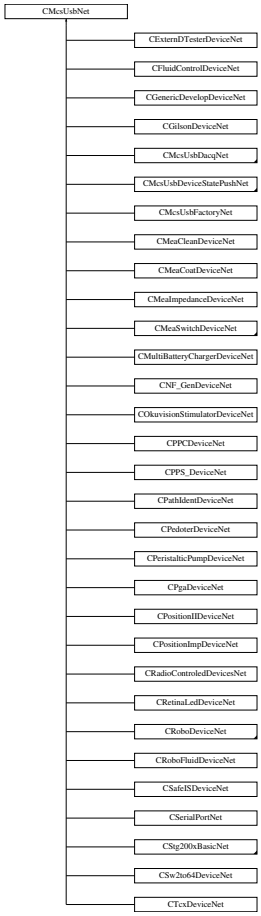
11.64.5.1 DeviceArrival `OnDeviceArrivalRemoval^ DeviceArrival`

11.64.5.2 DeviceRemoval `OnDeviceArrivalRemoval^ DeviceRemoval`

11.65 CMcsUsbNet Class Reference

Base class to handle MCS USB devices. All device classes are derived from this class. Functionality that is provided by all MCS devices is handled by this class.

Inheritance diagram for CMcsUsbNet:



Public Member Functions

- CMcsUsbNet ()

  Initializes a new instance of the base class to handle MCS USB devices.
- CMcsUsbNet (McsBusTypeEnumNet bustype)

  Initializes a new instance of the base class to handle MCS USB devices.
- virtual ~CMcsUsbNet ()
- !CMcsUsbNet ()
- DeviceEnumNet GetDeviceEnum ()
- virtual uint32_t Connect ()

  Opens a connection to the device.
- virtual uint32_t Connect (unsigned int LockMask)

  Opens a connection to the device.
- virtual uint32_t Connect (CMcsUsbListEntryNet^ entry)

  Opens a connection to the device.
- virtual uint32_t Connect (CMcsUsbListEntryNet^ entry, unsigned int LockMask)

  Opens a connection to the device.

- virtual uint32_t GetStatus ([System::Runtime::InteropServices::Out]uint32_t% iStatus)
- virtual bool IsConnected ()

  Check if a device is Connected.
- virtual void Disconnect ()

  Disconnect from a device.
- CMcsUsbListEntryNet ^ GetUsbListEntry ()
- virtual String ^ GetSerialNumber ()

  Query the Serial Number of the device.
- DriverVersionNet ^ GetVersion ()
- DriverVersionNet ^ GetVersion (CFirmwareDestinationNet dest)
- DeviceIdNet ^ GetDeviceId ()
- void MultibootSelectImage (unsigned int sector)

  Select the multiboot image specified by "sector" (range: 0..2) for IFB FPGA.
- 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.
- 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.
- uint32_t MultibootGetSelectedImage ()

  Gets sector index of selected FPGA boot image on IFB
- uint32_t GetMea21UsbPort ()

  Gets the USB port if an IFB that is used by this connection
- HeadstageIdEnumNet GetHeadstageID (uint32_t headstage)

  Gets the ID of a connected headstage.
- bool GetHeadstagePresent (uint32_t headstage)

  queries whether a headstage is present
- bool GetHeadstageActive (uint32_t headstage)

  queries whether a headstage is active
- void RescanHeadstage (uint32_t headstage)

  rescans and activates a headstage
- array< BYTE > ^ GetSoftwareKey (unsigned int index)
- void SetSoftwareKey (unsigned int index, array< BYTE >^ buffer)
- void RemoveSoftwareKey (unsigned int index)
- void AddSoftwareKey (String^ key)
- bool EmptyKey (String^ key)
- bool ValidKey (String^ key, [System::Runtime::InteropServices::Out]String^% serial_number)
- bool ValidKey (String^ key, uint8_t ProgrammID, uint8_t majorversion, [System::Runtime::InteropServices::Out]String^% serial_number)
- bool HasSoftwareKey (uint8_t ProgrammID, uint8_t majorversion)
- bool HasSoftwareKey (SoftwareKeyProgrammIdsNet::ProgrammIdsNet ProgrammID, uint8_t majorversion)
- String ^ GetSoftwareKeyString (uint8_t ProgrammID, uint8_t majorversion)
- String ^ GetSoftwareKeyString (SoftwareKeyProgrammIdsNet::ProgrammIdsNet ProgrammID, uint8_t majorversion)
- bool IsDeviceHighSpeedCapable ()
- bool IsDeviceHighSpeed ()
- McsUsbSpeedEnumNet GetDeviceCapableSpeed ()
- McsUsbSpeedEnumNet GetDeviceSpeed ()

  Query the Connection Speed of the device.
- unsigned int TxnTestMemoryWrite (unsigned short index)
- unsigned int TxnTestMemoryReadAndCheck (unsigned short index)
- void TxnSetSerialNumber (unsigned int number)
- unsigned int TxnGetSerialNumber ()
- unsigned int ReadRegister (unsigned int reg)
- array< uint32_t > ^ ReadRegister (unsigned int reg, int length)

- unsigned int ReadRegister32 (unsigned int adr)
- unsigned int ReadRegisterTimeSlot (unsigned int reg, int TimeSlot)
- void WriteRegister (unsigned int reg, unsigned int value)
- void WriteRegisterValue (unsigned int reg, unsigned int value)
- void WriteRegister32 (unsigned int adr, unsigned int value)
- void WriteRegister (unsigned int reg, array< unsigned int >^ values)
- void WriteRegisterArray (unsigned int reg, array< unsigned int >^ values)
- void WriteRegisterTimeSlot (unsigned int reg, unsigned int value, int TimeSlot)
- void WriteRegisterTimeSlot (unsigned int reg, array< unsigned int >^ values, int TimeSlot)
- bool ReadEepromRegisterPreconfig (uint32_t EEPROMBase, uint32_t DMA_reg, [System::Runtime::InteropServices::Out]uint32_t% DMA_value)
- bool ReadEepromRegisterPreconfig (uint32_t EEPROMBase, uint32_t DMA_reg, [System::Runtime::InteropServices::Out]uint32_t% DMA_value, uint32_t EEPROMSize)
- bool ReadEepromRegisterPreconfig (uint32_t EEPROMBase, uint32_t DMA_reg, [System::Runtime::InteropServices::Out]uint32_t% DMA_value, uint32_t EEPROMSize, uint32_t EepromStartAddress)
- void WriteEepromRegisterPreconfig (uint32_t EEPROMBase, uint32_t DMA_reg, uint32_t DMA_value)
- void WriteEepromRegisterPreconfig (uint32_t EEPROMBase, uint32_t DMA_reg, uint32_t DMA_value, uint32_t EEPROMSize)
- void WriteEepromRegisterPreconfig (uint32_t EEPROMBase, uint32_t DMA_reg, uint32_t DMA_value, uint32_t EEPROMSize, uint32_t EepromStartAddress)
- void EraseEepromRegisterPreconfig (uint32_t EEPROMBase, uint32_t DMA_reg)
- void EraseEepromRegisterPreconfig (uint32_t EEPROMBase, uint32_t DMA_reg, uint32_t EEPROMSize)
- void EraseEepromRegisterPreconfig (uint32_t EEPROMBase, uint32_t DMA_reg, uint32_t EEPROMSize, uint32_t EepromStartAddress)
- unsigned int GetLastUSBError ()
- void ThrowCUsbExceptionNetOnError (uint32_t status)
- bool GetDeviceCannotStallOutRequests ()
- String ^ GetHardwareRevision ()
- unsigned int GetFirmwareVersion (CFirmwareDestinationNet destination)

  Gets the firmware version for the destination.
- uint8_t GetNumConfigurations ()
- uint8_t GetConfiguration ()
- void SetConfiguration (uint8_t config)
- uint32_t GetDeviceRootHubVendorID ()

  Gets the Vendor ID of the USB root hub the device is connected to.
- UsbVendorIdEnumNet GetDeviceRootHubVendorEnum ()

  Gets the Vendor ID of the USB root hub the device is connected to.
- String ^ GetDeviceRootHubVendorName ()

  Gets the Vendor Name of the USB root hub the device is connected to.
- void EnableExceptions (bool enable)

  Enables or Disables Exceptions for calls to McsUsb Devices. If Exceptions are disabled, the return value of a command can be queries with the GetStatusOfLastCommand call instead.
- bool IsExceptionsEnabled ()
- uint32_t GetStatusOfLastCommand ()

  Gets the status of the last call to the McsUsb Library.
- uint32_t CyclePort ()
- void AssociateToThis (CMcsUsbNet^ device)

Static Public Member Functions

- static String ^ GetErrorText (unsigned int Status)

  Gets the error text string that belongs to a status number.

Static Public Attributes

- static const uint32_t Status_Crc = (0xE0100001L)
- static const uint32_t Status_Btstuff = (0xE0100002L)
- static const uint32_t Status_DataToggleMismatch = (0xE0100003L)
- static const uint32_t Status_Stall = (0xE0100004L)
- static const uint32_t Status_DevNotResponding = (0xE0100005L)
- static const uint32_t Status_PidCheckFailure = (0xE0100006L)
- static const uint32_t Status_UnexpectedPid = (0xE0100007L)
- static const uint32_t Status_DataOverrun = (0xE0100008L)
- static const uint32_t Status_DataUnderrun = (0xE0100009L)
- static const uint32_t Status_BufferOverrun = (0xE010000CL)
- static const uint32_t Status_BufferUnderrun = (0xE010000DL)
- static const uint32_t Status_NotAccessed = (0xE010000FL)
- static const uint32_t Status_Fifo = (0xE0100010L)
- static const uint32_t Status_EndpointHalted = (0xE0100030L)
- static const uint32_t Status_NoMemory = (0xE0100100L)
- static const uint32_t Status_InvalidUrbFunction = (0xE0100200L)
- static const uint32_t Status_InvalidParameter = (0xE0100300L)
- static const uint32_t Status_InvalidDeviceHandle = (0xE0100013L)
- static const uint32_t Status_InvalidHandle = (0xE0100012L)
- static const uint32_t Status_ErrorBusy = (0xE0100400L)
- static const uint32_t Status_RequestFailed = (0xE0100500L)
- static const uint32_t Status_InvalidPipeHandle = (0xE0100600L)
- static const uint32_t Status_NoBandwidth = (0xE0100700L)
- static const uint32_t Status_InternalHcError = (0xE0100800L)
- static const uint32_t Status_ErrorShortTransfer = (0xE0100900L)
- static const uint32_t Status_BadStartFrame = (0xE0100A00L)
- static const uint32_t Status_IsochRequestFailed = (0xE0100B00L)
- static const uint32_t Status_FrameControlOwned = (0xE0100C00L)
- static const uint32_t Status_ControlNotOwned = (0xE0100D00L)
- static const uint32_t Status_Canceled = (0xE0110000L)
- static const uint32_t Status_Canceling = (0xE0120000L)
- static const uint32_t Status_AlreadyConfigured = (0xE0110001L)
- static const uint32_t Status_Unconfigured = (0xE0110002L)
- static const uint32_t Status_NoSuchDevice = (0xE01F0002L)
- static const uint32_t Status_DeviceNotFound = (0xE01F0003L)
- static const uint32_t Status_NotSupported = (0xE01F0005L)
- static const uint32_t Status_IoPending = (0xE01F0006L)
- static const uint32_t Status_IoTimeout = (0xE01F0007L)
- static const uint32_t Status_DeviceRemoved = (0xE01F0008L)
- static const uint32_t Status_PipeNotLinked = (0xE01F0009L)
- static const uint32_t Status_ConnectedPipes = (0xE01F000AL)
- static const uint32_t Status_DeviceLocked = (0xE01F0010L)
- static const uint32_t Status_RequestMutexTimeout = (0xE01F0020L)
- static const uint32_t Status_RequestMutexFailed = (0xE01F0021L)
- static const uint32_t Status_LastUsbErrorMismatch = (0xE01F0022L)
- static const uint32_t WPAError_ScanningIsPending = ((0xA0220000L) | 0x0036)

Properties

- virtual String^ SerialNumber `[get]`

11.65.1 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.

11.65.2 Constructor & Destructor Documentation

11.65.2.1 CMcsUsbNet() [1/2] CMcsUsbNet ( )

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

11.65.2.2 CMcsUsbNet() [2/2] CMcsUsbNet (McsBusTypeEnumNet *bustype*)

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

Parameters

bustype	Type of device to use, either USB or PCI.

11.65.2.3 ~CMcsUsbNet() virtual ~CMcsUsbNet ( ) [virtual]

11.65.2.4 "!CMcsUsbNet() !CMcsUsbNet ( )

11.65.3 Member Function Documentation

11.65.3.1 AddSoftwareKey() void AddSoftwareKey (String^ *key*)

11.65.3.2 AssociateToThis() void AssociateToThis (CMcsUsbNet^ *device*)

11.65.3.3 Connect() [1/4] virtual uint32_t Connect ( ) [virtual]

Opens a connection to the device.

Returns

Error Status. 0 on success.

11.65.3.4 Connect() [2/4] virtual uint32_t Connect (CMcsUsbListEntryNet^ *entry*) [virtual]

Opens a connection to the device.

Parameters

entry	The Device List Entry for the device to be connected.

Returns

Error Status. 0 on success.

11.65.3.5 Connect() [3/4] virtual uint32_t Connect (CMcsUsbListEntryNet^ *entry*, unsigned int *LockMask*) [virtual]

Opens a connection to the device.

Parameters

entry	The Device List Entry for the device to be connected.
LockMask	The Lock Mask for this connection.

Returns

Error Status. 0 on success.

11.65.3.6 Connect() [4/4] virtual uint32_t Connect (unsigned int *LockMask*) [virtual]

Opens a connection to the device.

Parameters

LockMask	The Lock Mask for this connection.

Returns

Error Status. 0 on success.

11.65.3.7 CyclePort() `uint32_t CyclePort ( )`

11.65.3.8 Disconnect() `virtual void Disconnect ( ) [virtual]`

Disconnect from a device.

11.65.3.9 EmptyKey() `bool EmptyKey ( String^ key )`

11.65.3.10 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

enable	True to enable Exceptions, False to disable.

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

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

11.65.3.13 EraseEepromRegisterPreconfig() [3/3]

```
void EraseEepromRegisterPreconfig (
    uint32_t EEPROMBase,
    uint32_t DMA_reg,
    uint32_t EEPROMSize,
    uint32_t EepromStartAddress )
```

11.65.3.14 GetConfiguration()

```
uint8_t GetConfiguration ( )
```

11.65.3.15 GetDeviceCannotStallOutRequests()

```
bool GetDeviceCannotStallOutRequests ( )
```

11.65.3.16 GetDeviceCapableSpeed()

```
McsUsbSpeedEnumNet GetDeviceCapableSpeed ( )
```

11.65.3.17 GetDeviceEnum()

```
DeviceEnumNet GetDeviceEnum ( )
```

11.65.3.18 GetDeviceId()

```
DeviceIdNet ^ GetDeviceId ( )
```

11.65.3.19 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.

11.65.3.20 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.

11.65.3.21 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".

11.65.3.22 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.

11.65.3.23 GetErrorText() `static String ^ GetErrorText ( unsigned int Status ) [static]`

Gets the error text string that belongs to a status number.

Parameters

Status	The status number you want the text for.

Returns

The error text string that belongs to the status number.

11.65.3.24 GetFirmwareVersion() `unsigned int GetFirmwareVersion ( CFirmwareDestinationNet destination )`

Gets the firmware version for the destination.

Parameters

destination	The destination to be queried.

Returns

The firmware version as a 32 bit number, the upper 16 bit contain the majaor version number, the lower 16 bit the minor version number.

11.65.3.25 GetHardwareRevision() `String ^ GetHardwareRevision ( )`

11.65.3.26 GetHeadstageActive() `bool GetHeadstageActive ( uint32_t headstage )`

queries whether a headstage is active

Parameters

in	*headstage*	the headstage number (0 or 1)

Returns

true if the headstage is active

11.65.3.27 GetHeadstageID() `HeadstageIdEnumNet GetHeadstageID ( uint32_t headstage )`

Gets the ID of a connected headstage.

Parameters

in	*headstage*	the headstage number (0 or 1)

Returns

enumerated Headstage ID

11.65.3.28 GetHeadstagePresent() `bool GetHeadstagePresent ( uint32_t headstage )`

queries whether a headstage is present

Parameters

in	*headstage*	the headstage number (0 or 1)

Returns

true if the headstage is present

11.65.3.29 GetLastUSBError() `unsigned int GetLastUSBError ( )`

11.65.3.30 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

11.65.3.31 GetNumConfigurations() `uint8_t GetNumConfigurations ( )`

11.65.3.32 GetSerialNumber() `virtual String ^ GetSerialNumber ( ) [virtual]`

Query the Serial Number of the device.

Returns

The Serial Number.

11.65.3.33 GetSoftwareKey()

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

11.65.3.34 GetSoftwareKeyString() [1/2]

```
String ^ GetSoftwareKeyString (
  SoftwareKeyProgrammIdsNet::ProgrammIdsNet ProgrammID,
  uint8_t majorversion )
```

11.65.3.35 GetSoftwareKeyString() [2/2]

```
String ^ GetSoftwareKeyString (
  uint8_t ProgrammID,
  uint8_t majorversion )
```

11.65.3.36 GetStatus()

```
virtual uint32_t GetStatus (
  [System::Runtime::InteropServices::Out] uint32_t% iStatus ) [virtual]
```

11.65.3.37 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.

11.65.3.38 GetUsbListEntry() `CMcsUsbListEntryNet ^ GetUsbListEntry ( )`

11.65.3.39 GetVersion() [1/2] `DriverVersionNet ^ GetVersion ( )`

11.65.3.40 GetVersion() [2/2] `DriverVersionNet ^ GetVersion ( CFirmwareDestinationNet dest )`

11.65.3.41 HasSoftwareKey() [1/2] `bool HasSoftwareKey ( SoftwareKeyProgrammIdsNet::ProgrammIdsNet ProgrammID, uint8_t majorversion )`

11.65.3.42 HasSoftwareKey() [2/2] `bool HasSoftwareKey ( uint8_t ProgrammID, uint8_t majorversion )`

11.65.3.43 IsConnected() `virtual bool IsConnected ( ) [virtual]`

Check if a device is Connected.

Returns

true if the device is connected.

11.65.3.44 IsDeviceHighSpeed() `bool IsDeviceHighSpeed ( )`

11.65.3.45 IsDeviceHighSpeedCapable() `bool IsDeviceHighSpeedCapable ( )`

11.65.3.46 IsExceptionsEnabled() `bool IsExceptionsEnabled ( )`

11.65.3.47 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.

11.65.3.48 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.

11.65.3.49 MultibootGetSelectedImage() `uint32_t MultibootGetSelectedImage ( )`

Gets sector index of selected FPGA boot image on IFB

Returns

Sector index of image; range: 0..2

11.65.3.50 MultibootSelectImage() `void MultibootSelectImage ( unsigned int sector )`

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

Returns

Throws exception on error.

11.65.3.51 ReadEepromRegisterPreconfig() [1/3]

```
bool ReadEepromRegisterPreconfig (
    uint32_t EEPROMBase,
    uint32_t DMA_reg,
    [System::Runtime::InteropServices::Out] uint32_t% DMA_value )
```

11.65.3.52 ReadEepromRegisterPreconfig() [2/3]

```
bool ReadEepromRegisterPreconfig (
    uint32_t EEPROMBase,
    uint32_t DMA_reg,
    [System::Runtime::InteropServices::Out] uint32_t% DMA_value,
    uint32_t EEPROMSize )
```

11.65.3.53 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 )
```

11.65.3.54 ReadRegister() [1/2]

```
unsigned int ReadRegister (
    unsigned int reg )
```

11.65.3.55 ReadRegister() [2/2]

```
array<uint32_t> ^ ReadRegister (
    unsigned int reg,
    int length )
```

11.65.3.56 ReadRegister32()

```
unsigned int ReadRegister32 (
    unsigned int adr )
```

11.65.3.57 ReadRegisterTimeSlot()

```
unsigned int ReadRegisterTimeSlot (
    unsigned int reg,
    int TimeSlot )
```

11.65.3.58 RemoveSoftwareKey()

```
void RemoveSoftwareKey (
    unsigned int index )
```

11.65.3.59 RescanHeadstage()

```
void RescanHeadstage (
    uint32_t headstage )
```

rescans and activates a headstage

Parameters

in	*headstage*	the headstage number (0 or 1)

11.65.3.60 SetConfiguration()

```
void SetConfiguration (
    uint8_t config )
```

11.65.3.61 SetSoftwareKey()

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

11.65.3.62 ThrowCUsbExceptionNetOnError()

```
void ThrowCUsbExceptionNetOnError (
    uint32_t status )
```

11.65.3.63 TxnGetSerialNumber()

```
unsigned int TxnGetSerialNumber ( )
```

11.65.3.64 TxnSetSerialNumber()

```
void TxnSetSerialNumber (
    unsigned int number )
```

11.65.3.65 TxnTestMemoryReadAndCheck()

```
unsigned int TxnTestMemoryReadAndCheck (
    unsigned short index )
```

11.65.3.66 TxnTestMemoryWrite()

```
unsigned int TxnTestMemoryWrite (
    unsigned short index )
```

11.65.3.67 ValidKey() [1/2]

```
bool ValidKey (
    String^ key,
    [System::Runtime::InteropServices::Out] String^% serial_number )
```

11.65.3.68 ValidKey() [2/2] bool ValidKey (

```
String^ key,
uint8_t ProgrammID,
uint8_t majorversion,
[System::Runtime::InteropServices::Out] String^% serial_number )
```

11.65.3.69 WriteEepromRegisterPreconfig() [1/3] void WriteEepromRegisterPreconfig (

```
uint32_t EEPROMBase,
uint32_t DMA_reg,
uint32_t DMA_value )
```

11.65.3.70 WriteEepromRegisterPreconfig() [2/3] void WriteEepromRegisterPreconfig (

```
uint32_t EEPROMBase,
uint32_t DMA_reg,
uint32_t DMA_value,
uint32_t EEPROMSize )
```

11.65.3.71 WriteEepromRegisterPreconfig() [3/3] void WriteEepromRegisterPreconfig (

```
uint32_t EEPROMBase,
uint32_t DMA_reg,
uint32_t DMA_value,
uint32_t EEPROMSize,
uint32_t EepromStartAddress )
```

11.65.3.72 WriteRegister() [1/2] void WriteRegister (

```
unsigned int reg,
array< unsigned int >^ values )
```

11.65.3.73 WriteRegister() [2/2] void WriteRegister (

```
unsigned int reg,
unsigned int value )
```

11.65.3.74 WriteRegister32() void WriteRegister32 (

```
unsigned int adr,
unsigned int value )
```

11.65.3.75 WriteRegisterArray()

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

11.65.3.76 WriteRegisterTimeSlot() [1/2]

```
void WriteRegisterTimeSlot (
    unsigned int reg,
    array< unsigned int >^ values,
    int TimeSlot )
```

11.65.3.77 WriteRegisterTimeSlot() [2/2]

```
void WriteRegisterTimeSlot (
    unsigned int reg,
    unsigned int value,
    int TimeSlot )
```

11.65.3.78 WriteRegisterValue()

```
void WriteRegisterValue (
    unsigned int reg,
    unsigned int value )
```

11.65.4 Member Data Documentation

11.65.4.1 Status_AlreadyConfigured

```
const uint32_t Status_AlreadyConfigured = (0xE0110001L) [static]
```

11.65.4.2 Status_BadStartFrame

```
const uint32_t Status_BadStartFrame = (0xE0100A00L) [static]
```

11.65.4.3 Status_Btstuff

```
const uint32_t Status_Btstuff = (0xE0100002L) [static]
```

11.65.4.4 Status_BufferOverrun

```
const uint32_t Status_BufferOverrun = (0xE010000CL) [static]
```

11.65.4.5 Status_BufferUnderrun

```
const uint32_t Status_BufferUnderrun = (0xE010000DL) [static]
```

11.65.4.6 Status_Canceled

```
const uint32_t Status_Canceled = (0xE0110000L)  [static]
```

11.65.4.7 Status_Canceling

```
const uint32_t Status_Canceling = (0xE0120000L)  [static]
```

11.65.4.8 Status_ConnectedPipes

```
const uint32_t Status_ConnectedPipes = (0xE01F000AL)  [static]
```

11.65.4.9 Status_ControlNotOwned

```
const uint32_t Status_ControlNotOwned = (0xE0100D00L)  [static]
```

11.65.4.10 Status_Crc

```
const uint32_t Status_Crc = (0xE0100001L)  [static]
```

11.65.4.11 Status_DataOverrun

```
const uint32_t Status_DataOverrun = (0xE0100008L)  [static]
```

11.65.4.12 Status_DataToggleMismatch

```
const uint32_t Status_DataToggleMismatch = (0xE0100003L)
[static]
```

11.65.4.13 Status_DataUnderrun

```
const uint32_t Status_DataUnderrun = (0xE0100009L)  [static]
```

11.65.4.14 Status_DeviceLocked

```
const uint32_t Status_DeviceLocked = (0xE01F0010L)  [static]
```

11.65.4.15 Status_DeviceNotFound

```
const uint32_t Status_DeviceNotFound = (0xE01F0003L)  [static]
```

11.65.4.16 Status_DeviceRemoved

```
const uint32_t Status_DeviceRemoved = (0xE01F0008L)  [static]
```

11.65.4.17 Status_DevNotResponding

```
const uint32_t Status_DevNotResponding = (0xE0100005L) [static]
```

11.65.4.18 Status_EndpointHalted

```
const uint32_t Status_EndpointHalted = (0xE0100030L) [static]
```

11.65.4.19 Status_ErrorBusy

```
const uint32_t Status_ErrorBusy = (0xE0100400L) [static]
```

11.65.4.20 Status_ErrorShortTransfer

```
const uint32_t Status_ErrorShortTransfer = (0xE0100900L) [static]
```

11.65.4.21 Status_Fifo

```
const uint32_t Status_Fifo = (0xE0100010L) [static]
```

11.65.4.22 Status_FrameControlOwned

```
const uint32_t Status_FrameControlOwned = (0xE0100C00L) [static]
```

11.65.4.23 Status_InternalHcError

```
const uint32_t Status_InternalHcError = (0xE0100800L) [static]
```

11.65.4.24 Status_InvalidDeviceHandle

```
const uint32_t Status_InvalidDeviceHandle = (0xE0100013L) [static]
```

11.65.4.25 Status_InvalidHandle

```
const uint32_t Status_InvalidHandle = (0xE0100012L) [static]
```

11.65.4.26 Status_InvalidParameter

```
const uint32_t Status_InvalidParameter = (0xE0100300L) [static]
```

11.65.4.27 Status_InvalidPipeHandle

```
const uint32_t Status_InvalidPipeHandle = (0xE0100600L) [static]
```

11.65.4.28 Status_InvalidUrbFunction

```
const uint32_t Status_InvalidUrbFunction = (0xE0100200L) [static]
```

11.65.4.29 Status_IoPending

```
const uint32_t Status_IoPending = (0xE01F0006L) [static]
```

11.65.4.30 Status_IoTimeout

```
const uint32_t Status_IoTimeout = (0xE01F0007L) [static]
```

11.65.4.31 Status_IsochRequestFailed

```
const uint32_t Status_IsochRequestFailed = (0xE0100B00L) [static]
```

11.65.4.32 Status_LastUsbErrorMismatch

```
const uint32_t Status_LastUsbErrorMismatch = (0xE01↩
F0022L) [static]
```

11.65.4.33 Status_NoBandwidth

```
const uint32_t Status_NoBandwidth = (0xE0100700L) [static]
```

11.65.4.34 Status_NoMemory

```
const uint32_t Status_NoMemory = (0xE0100100L) [static]
```

11.65.4.35 Status_NoSuchDevice

```
const uint32_t Status_NoSuchDevice = (0xE01F0002L) [static]
```

11.65.4.36 Status_NotAccessed

```
const uint32_t Status_NotAccessed = (0xE010000FL) [static]
```

11.65.4.37 Status_NotSupported

```
const uint32_t Status_NotSupported = (0xE01F0005L) [static]
```

11.65.4.38 Status_PidCheckFailure

```
const uint32_t Status_PidCheckFailure = (0xE0100006L) [static]
```

11.65.4.39 Status_PipeNotLinked

```
const uint32_t Status_PipeNotLinked = (0xE01F0009L) [static]
```

11.65.4.40 Status_RequestFailed

```
const uint32_t Status_RequestFailed = (0xE0100500L) [static]
```

11.65.4.41 Status_RequestMutexFailed

```
const uint32_t Status_RequestMutexFailed = (0xE01F0021L) [static]
```

11.65.4.42 Status_RequestMutexTimeout

```
const uint32_t Status_RequestMutexTimeout = (0xE01↩F0020L) [static]
```

11.65.4.43 Status_Stall

```
const uint32_t Status_Stall = (0xE0100004L) [static]
```

11.65.4.44 Status_Unconfigured

```
const uint32_t Status_Unconfigured = (0xE0110002L) [static]
```

11.65.4.45 Status_UnexpectedPid

```
const uint32_t Status_UnexpectedPid = (0xE0100007L) [static]
```

11.65.4.46 WPAError_ScanningIsPending

```
const uint32_t WPAError_ScanningIsPending = ( (0x↩A0220000L) | 0x0036 ) [static]
```

11.65.5 Property Documentation

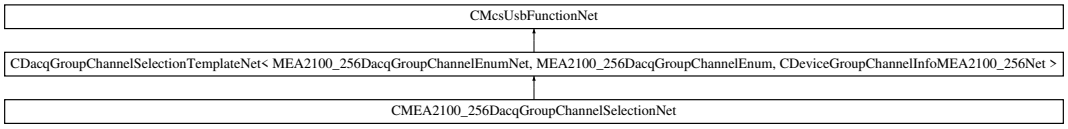
11.65.5.1 SerialNumber

```
virtual String^ SerialNumber [get]
```

11.66 CMcsUsbPointerContainer Class Reference

11.67 CMEA2100_256DacqGroupChannelSelectionNet Class Reference

Inheritance diagram for CMEA2100_256DacqGroupChannelSelectionNet:



Public Member Functions

- CMEA2100_256DacqGroupChannelSelectionNet (CMcsUsbNet^ mcsusb)

Additional Inherited Members

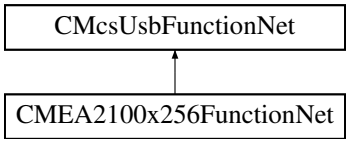
11.67.1 Constructor & Destructor Documentation

11.67.1.1 CMEA2100_256DacqGroupChannelSelectionNet() CMEA2100_256DacqGroupChannelSelectionNet (CMcsUsbNet^ *mcsusb*)

11.68 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^ mcsusb, CMcsUsbFunctionPointerContainer^ pMEA2100x256↩ FunctionPointerContainer)

  Initializes a new instance of the CMEA2100x256FunctionNet class.
- CMEA2100x256FunctionNet (CMcsUsbNet^ 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

11.68.1 Detailed Description

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

11.68.2 Constructor & Destructor Documentation

11.68.2.1 CMEA2100x256FunctionNet() [1/2]

```
CMEA2100x256FunctionNet (
    CMcsUsbNet^ mcsusb,
    CMcsUsbFunctionPointerContainer^ pMEA2100x256FunctionPointerContainer )
```

Initializes a new instance of the CMEA2100x256FunctionNet class.

11.68.2.2 CMEA2100x256FunctionNet() [2/2]

```
CMEA2100x256FunctionNet (
    CMcsUsbNet^ mcsusb )
```

11.68.2.3 ~CMEA2100x256FunctionNet()

```
virtual ~CMEA2100x256FunctionNet ( ) [virtual]
```

11.68.2.4 "!CMEA2100x256FunctionNet()

```
!CMEA2100x256FunctionNet ( )
```

11.68.3 Member Function Documentation

11.68.3.1 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.

11.68.3.2 SetLayoutConfiguration()

```
void SetLayoutConfiguration (
    StimulationLayoutConfigurationEnumNet LayoutConfiguration )
```

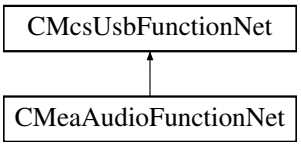
Sets the stimulation layout configuration. Can be single well, 6-well or 9-well. The number of DAC channels available per well is Mcs::Usb::CStg200xBasicNet::GetNumberOfAnalogChannels divided by Mcs::Usb::CStg200xBasicNet::GetNumberOfStimulationSourcesPerElectrode.

Parameters

LayoutConfiguration	The new stimulation layout configuration.

11.69 CMeaAudioFunctionNet Class Reference

Inheritance diagram for CMeaAudioFunctionNet:



Classes

- struct s_setaudionet

Public Member Functions

- CMeaAudioFunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ meaAudioFunction↩ PointerContainer)
- CMeaAudioFunctionNet (CMcsUsbNet^ mcsusb)
- virtual uint32_t GetNumberOfAudioChannels ()

  Gets the number of available audio channels.
- virtual uint32_t SetAudioChannels (array< s_setaudionet^ >^ channels)

  Sets the electrode to monitor and amplification for the audio channels.
- virtual uint32_t SetAudioChannels (array< s_setaudionet^ >^ 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^ >^% channels)

  Gets the electrode to monitor and amplification for the audio channels.
- virtual uint32_t GetAudioChannels ([System::Runtime::InteropServices::Out]array< s_setaudionet^ >^% channels, unsigned int virtualDevice)

  Gets the electrode to monitor and amplification for the audio channels.

Additional Inherited Members

11.69.1 Constructor & Destructor Documentation

11.69.1.1 CMeaAudioFunctionNet() [1/2]

```
CMeaAudioFunctionNet (
    CMcsUsbNet^ mcsusb,
    CMcsUsbFunctionPointerContainer^ meaAudioFunctionPointerContainer )
```

11.69.1.2 CMeaAudioFunctionNet() [2/2] CMeaAudioFunctionNet (CMcsUsbNet^ *mcsusb*)

11.69.2 Member Function Documentation

11.69.2.1 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

channels	Struct which contains the electrode (channel) and amplification on return.

Returns

Error Status. 0 on success.

11.69.2.2 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

channels	Struct which contains the electrode (channel) and amplification on return.

Parameters

virtualDevice	Virtual device to use.

Returns

Error Status. 0 on success.

11.69.2.3 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.

11.69.2.4 SetAudioChannels() [1/2] virtual uint32_t SetAudioChannels (array< s_setaudionet^>^ *channels*) [virtual]

Sets the electrode to monitor and amplification for the audio channels.

Parameters

channels	Struct which defines the electrode (channel) and amplification.

Returns

Error Status. 0 on success.

11.69.2.5 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

channels	Struct which defines the electrode (channel) and amplification.

Parameters

virtualDevice	Virtual device to use.

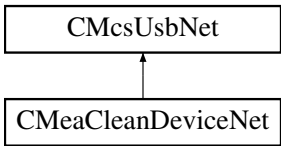
Returns

Error Status. 0 on success.

11.70 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

11.70.1 Detailed Description

CMeaCleanDeviceNet is the class to access the MEA Clean device.

11.70.2 Constructor & Destructor Documentation

11.70.2.1 CMeaCleanDeviceNet() `CMeaCleanDeviceNet ( )`

Initializes a new instance of the CMeaCleanDeviceNet class.

11.70.2.2 ~CMeaCleanDeviceNet() `virtual ~CMeaCleanDeviceNet ( ) [virtual]`

11.70.2.3 "!CMeaCleanDeviceNet() `!CMeaCleanDeviceNet ( )`

11.70.3 Member Function Documentation

11.70.3.1 GetCycle() `int32_t GetCycle ( )`

Gets the current cycle.

Returns

The cycle number.

11.70.3.2 GetCycles() `uint32_t GetCycles ( )`

Gets the number of cycles.

Returns

The number of cycles to run for.

11.70.3.3 GetMaxVoltage() `int32_t GetMaxVoltage ( )`

Gets the upper voltage level

Returns

The upper voltage level in mV.

11.70.3.4 GetMinVoltage() `int32_t GetMinVoltage ( )`

Gets the lower voltage level.

Returns

The lower voltage level in mV.

11.70.3.5 GetOutputVoltage() `int32_t GetOutputVoltage ( )`

Gets the output voltage.

Returns

The output voltage in mV.

11.70.3.6 GetSlope() `uint32_t GetSlope ( )`

Gets the voltage slope.

Returns

The voltage slope in mV/s.

11.70.3.7 IsRunning() `bool IsRunning ( )`

Gets if the MEA Clean device is running.

Returns

"true" when a run is in progress, otherwise "false".

11.70.3.8 SetCycles() `void SetCycles ( uint32_t cycles )`

Sets the number of cycles.

Parameters

cycles	The number of cycles to run for (0 .. 99).

11.70.3.9 SetMaxVoltage() `void SetMaxVoltage ( int32_t voltageMax )`

Sets the upper voltage level.

Parameters

voltageMax	The upper voltage level in mV (-1.6 .. 1.6 V).

11.70.3.10 SetMinVoltage() `void SetMinVoltage ( int32_t voltageMin )`

Sets the lower voltage level.

Parameters

voltageMin	The lower voltage level in mV (-1.6 .. 1.6 V).

11.70.3.11 SetSlope() `void SetSlope ( uint32_t voltageSlope )`

Sets the voltage slope.

Parameters

voltageSlope	The voltage slope in mV/s (range 0 .. 60 V/s).

11.70.3.12 Start() `void Start ( )`

Starts a MEA Clean run.

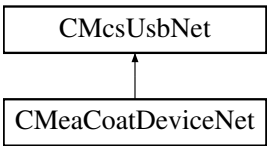
11.70.3.13 Stop() `void Stop ( )`

Stops a MEA Clean run.

11.71 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

11.71.1 Detailed Description

CMeaCoatDeviceNet is the class to access the MEA Coat device.

11.71.2 Constructor & Destructor Documentation

11.71.2.1 CMeaCoatDeviceNet() `CMeaCoatDeviceNet ( )`

Initializes a new instance of the CMeaCoatDeviceNet class.

11.71.2.2 ~CMeaCoatDeviceNet() `virtual ~CMeaCoatDeviceNet ( ) [virtual]`

11.71.2.3 "!CMeaCoatDeviceNet() `!CMeaCoatDeviceNet ( )`

11.71.3 Member Function Documentation

11.71.3.1 GetCurrentCycle() `int32_t GetCurrentCycle ( )`

Gets the current cycle.

Returns

The cycle number.

11.71.3.2 GetCycles() uint32_t GetCycles ( )

Gets the number of cycles.

Returns

The number of cycles to run for.

11.71.3.3 GetDuration() uint32_t GetDuration ( )

Gets the duration of a MEA Coat run.

Returns

The duration in ms.

11.71.3.4 GetMaxCurrent() uint32_t GetMaxCurrent ( )

Gets the limit of the current ramp (absolute value).

Returns

The limit of the current ramp in pA (absolute value).

11.71.3.5 GetOffsetCurrent() int32_t GetOffsetCurrent ( )

Gets the offset of the current.

Returns

The offset of the current in pA.

11.71.3.6 GetOutputCurrent() int32_t GetOutputCurrent ( )

Gets the output current.

Returns

The output current in pA.

11.71.3.7 GetPauseDuration() `uint32_t GetPauseDuration ( )`

Gets the duration of the pause between MEA Coat pulses.

Returns

The duration in ms.

11.71.3.8 GetSlope() `int32_t GetSlope ( )`

Gets the current slope.

Returns

The current slope in pA/s.

11.71.3.9 GetTimeInPause() `int32_t GetTimeInPause ( )`

Gets the time in the pause.

Returns

The time in the pause in ms.

11.71.3.10 GetTimeInPlateau() `int32_t GetTimeInPlateau ( )`

Gets the time in the plateau.

Returns

The time in the plateau in ms.

11.71.3.11 IsRunning() `bool IsRunning ( )`

Gets if the MEA Clean device is running.

Returns

"true" when a run is in progress, otherwise "false".

11.71.3.12 SetCycles() `void SetCycles ( uint32_t cycles )`

Sets the number of cycles.

Parameters

cycles	The number of cycles to run for (0 .. 99).

11.71.3.13 SetDuration() `void SetDuration ( uint32_t duration )`

Sets the duration of a MEA Coat run.

Parameters

duration	The duration in ms (range 0 .. 65 s).

11.71.3.14 SetMaxCurrent() `void SetMaxCurrent ( uint32_t currentMax )`

Sets the limit of the current ramp (absolute value).

Parameters

currentMax	The limit of the current ramp in pA (absolute value, 0 .. 18 nA).

11.71.3.15 SetOffsetCurrent() `void SetOffsetCurrent ( int32_t currentOffset )`

Sets the offset of the current.

Parameters

currentOffset	The offset of the current in pA (-10 .. 10 nA).

11.71.3.16 SetPauseDuration() `void SetPauseDuration ( uint32_t pauseDuration )`

Sets the duration of the pause between MEA Coat pulses.

Parameters

pauseDuration	The duration in ms (range 0 .. 65 s).

11.71.3.17 SetSlope() `void SetSlope ( int32_t currentSlope )`

Sets the current slope.

Parameters

currentSlope	The current slope in pA/s (range -65 .. 65 nA/s).

11.71.3.18 Start() `void Start ( )`

Starts a MEA Coat run.

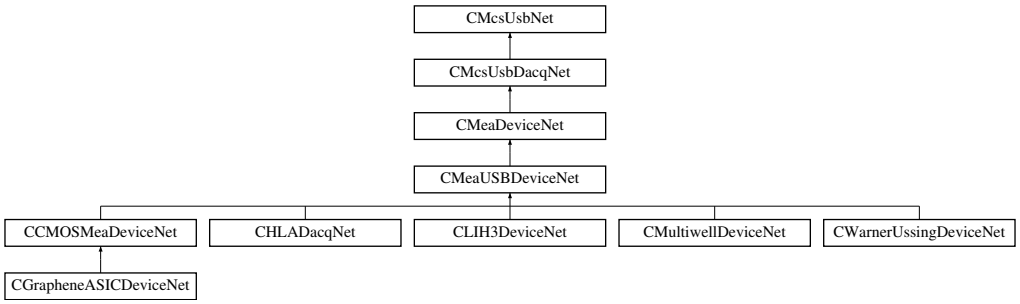
11.71.3.19 Stop() `void Stop ( )`

Stops a MEA Coat run.

11.72 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^ channelData, OnError^ 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

11.72.1 Detailed Description

Base class for MEA data acquisition devices.

There are two different device types for MEA data aquistion devices. There are the USB-MEA devices and the MC↩ _Card. In .NET both classes can be accessed by the contructor of the base class CMeaDeviceNet, which contructs 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

11.72.2 Constructor & Destructor Documentation

11.72.2.1 CMeaDeviceNet() [1/2] CMeaDeviceNet (McsBusTypeEnumNet *bustype*)

Initializes a new instance of CMeaDeviceNet class.

Parameters

bustype	Type of device to use, either USB or PCI.

11.72.2.2 CMeaDeviceNet() [2/2] CMeaDeviceNet (McsBusTypeEnumNet *bustype*, OnChannelData^ *channelData*, OnError^ *error*)

Initializes a new instance of CMeaDeviceNet class.

Parameters

bustype	Type of device to use, either USB or PCI.

Parameters

channelData	Callback to call when new data is available.

Parameters

error	Callback to call when an error occurred.

11.72.2.3 ~CMeaDeviceNet() ~CMeaDeviceNet ( )

11.72.3 Member Function Documentation

11.72.3.1 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

enable	True to enable, False to disable.
virtualDevice	virtual device to use.

Returns

Error Status. 0 on success.

11.72.3.2 EnableDigitalIn() [1/2] virtual uint32_t EnableDigitalIn (

bool *enable*,
unsigned int *virtualDevice*) [virtual]

Enable the digital data word in the datastream.

Parameters

enable	True to enable, False to disable.
virtualDevice	virtual device to use.

Returns

Error Status. 0 on success.

11.72.3.3 EnableDigitalIn() [2/2] virtual uint32_t EnableDigitalIn (

DigitalDatastreamEnableEnumNet *enable*,
unsigned int *virtualDevice*) [virtual]

Enable digital data words in the datastream.

Parameters

enable	True to enable, False to disable.
virtualDevice	virtual device to use.

Returns

Error Status. 0 on success.

11.72.3.4 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

enable	True to enable, False to disable.
virtualDevice	virtual device to use.

Returns

Error Status. 0 on success.

11.72.3.5 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.

11.72.3.6 GetEnumerationSpeed()

```
int32_t GetEnumerationSpeed ( )
```

11.72.3.7 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.

11.72.3.8 SetDigitalOut()

```
virtual void SetDigitalOut (
    unsigned int digout_value,
    int pulselength ) [virtual]
```

Generate a pulse on the digital output.

Parameters

digout_value	Bitmask to set on the digital out.

Parameters

pulselength	Pulselength in ms.

11.72.3.9 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

NumberOfChannels_HS1	Number of analog channels from the Headstage 1.

Parameters

NumberOfChannels_HS2	Number of analog channels from the Headstage 2.

Parameters

NumberOfChannels_DSP	Number of data words from the DSP.

Parameters

NumberOfChannels_IF	Number of analog channels from the Interfaceboard.

Parameters

virtualDevice	virtualDevice to use.

Returns

Error Status. 0 on success.

11.72.3.10 SetNumberOfChannels() [1/2] virtual uint32_t SetNumberOfChannels (int *NumberOfChannels*) [virtual]

Sets the number of analog channels in the datastream.

Parameters

NumberOfChannels	Number of analog channels.

Returns

Error Status. 0 on success.

11.72.3.11 SetNumberOfChannels() [2/2] virtual uint32_t SetNumberOfChannels (int *NumberOfChannels*, unsigned int *virtualDevice*) [virtual]

Sets the number of analog channels in the datastream.

Parameters

NumberOfChannels	Number of analog channels.
virtualDevice	virtual device to use.

Returns

Error Status. 0 on success.

11.72.3.12 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

mask	Bits in the digital dataword which are monitored for a match with value.

Parameters

value	Pattern which must match for the trigger to start.

Returns

Error Status. 0 on success.

11.72.3.13 SetTriggerPeriod() virtual uint32_t SetTriggerPeriod (int *samples*, unsigned int *virtualDevice*) [virtual]

Sets the maximum number of samples per trigger.

Parameters

samples	Number of samples to acquire after the trigger condition is met.

Returns

Error Status. 0 on success.

11.72.4 Property Documentation

11.72.4.1 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.

11.72.4.2 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.

11.72.4.3 MeaAudioFunctionNet `CMeaAudioFunctionNet^ MeaAudioFunctionNet [get]`

11.72.4.4 MeaDigitalDataFunctionNet `CMeaDigitalDataFunctionNet^ MeaDigitalDataFunctionNet [get]`

11.72.4.5 MeaFeedbackFunctionNet `CMeaFeedbackFunctionNet^ MeaFeedbackFunctionNet [get]`

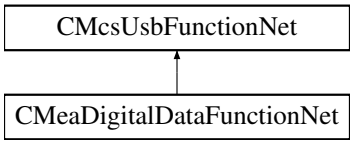
11.72.4.6 MeFunctionNet `CMeFunctionNet^ MeFunctionNet [get]`

11.72.4.7 W2100_FunctionNet `CW2100_FunctionNet^ W2100_FunctionNet [get]`

11.72.4.8 WClassicFunctionNet `CWClassicFunctionNet^ WClassicFunctionNet [get]`

11.73 CMeaDigitalDataFunctionNet Class Reference

Inheritance diagram for CMeaDigitalDataFunctionNet:



Public Member Functions

- CMeaDigitalDataFunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ meaDigital↩ FunctionPointerContainer)
- 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

11.73.1 Constructor & Destructor Documentation

11.73.1.1 CMeaDigitalDataFunctionNet() [1/2]

```
CMeaDigitalDataFunctionNet (
    CMcsUsbNet^ mcsusb,
    CMcsUsbFunctionPointerContainer^ meaDigitalFunctionPointerContainer )
```

11.73.1.2 CMeaDigitalDataFunctionNet() [2/2]

```
CMeaDigitalDataFunctionNet (
    CMcsUsbNet^ mcsusb )
```

11.73.2 Member Function Documentation

11.73.2.1 GetDigitalData()

```
unsigned int GetDigitalData ( )
```

Get the value of the digital output.

Returns

Value on the digital data register.

11.73.2.2 SetDigitalData() [1/2]

```
void SetDigitalData (
    unsigned int bit_number,
    bool value )
```

Generate a value on the digital output.

Parameters

bit_number	Bit number to change.

Parameters

value	Bit value.

11.73.2.3 SetDigitalData() [2/2] void SetDigitalData (

```
unsigned int digital_value,
unsigned int digital_value_mask )
```

Generate a value on the digital output.

Parameters

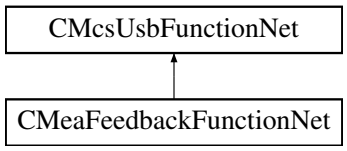
digital_value	Value to set.

Parameters

digital_value_mask	Mask for change.

11.74 CMeaFeedbackFunctionNet Class Reference

Inheritance diagram for CMeaFeedbackFunctionNet:



Public Member Functions

- CMeaFeedbackFunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ meaFeedback↩ FunctionNet)
- CMeaFeedbackFunctionNet (CMcsUsbNet^ 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 >^ parameters)
- void FeedbackSetFilterParameter32 (unsigned char filter, array< int >^ parameters)
- void FeedbackSetIIRFilterParameter (unsigned char filter, int length, array< double >^ parameters)
- void FeedbackSetMkFilter (unsigned char filter, String^ filtertype, double cheb_ribble, String^ passtype, int order, double alpha1, double alpha2)
- void FeedbackSetCreateFilter (unsigned char filter, CCreateFilterNet^ createFilter)
- void FeedbackSetChannelFilter (short channel, char filter)

- void FeedbackSetGlobalChannelFilter (char filter, unsigned short firstchannel, unsigned short lastchannel)
- void FeedbackSetFilterOff ()
- void FeedbackSetNumberOfSpikeDetectors (unsigned short number)
- void FeedbackSetSpikeDetectorThreshold (unsigned short position, unsigned short sourcechannel, unsigned short resultchannel, unsigned short trigger, unsigned short totzeit, int threshold1, int threshold2, short slope)
- void FeedbackSetNumberOfRateCounter (unsigned short number)
- void FeedbackSetRateCounter (unsigned short position, unsigned short sourcechannel, unsigned short resultchannel)
- void FeedbackSetNumberOfRateDetectors (unsigned short number)
- void FeedbackSetRateDetector (unsigned short position, unsigned short resultchannel, unsigned short trigger, unsigned short totzeit, unsigned short pulses, unsigned int duration1, unsigned int duration2)
- void FeedbackSetNumberOfLogics (unsigned short number)
- void FeedbackSetLogic (unsigned short position, array< unsigned short >^ sourcechannel, unsigned short resultchannel, unsigned int lookup)
- void FeedbackSetNumberOfTriggers (unsigned short number)
- void FeedbackSetTrigger (unsigned short position, unsigned short sourcechannel, unsigned short resultchannel, unsigned short trigger, unsigned short totzeit)
- void FeedbackSetAnalogSource (AnalogSourceEnumNet AnalogSource, unsigned int Channels, unsigned int Offset)
- void FeedbackSetTriggerTotzeitFactor (unsigned short trigger_factor, unsigned short totzeit_factor)
- void FeedbackGetTriggerTotzeitFactor ([System::Runtime::InteropServices::Out]unsigned short% trigger_↩ factor, [System::Runtime::InteropServices::Out]unsigned short% totzeit_factor)

Additional Inherited Members

11.74.1 Constructor & Destructor Documentation

11.74.1.1 CMeaFeedbackFunctionNet() [1/2]

```
CMeaFeedbackFunctionNet (
    CMcsUsbNet^ mcsusb,
    CMcsUsbFunctionPointerContainer^ meaFeedbackFunctionNet )
```

11.74.1.2 CMeaFeedbackFunctionNet() [2/2]

```
CMeaFeedbackFunctionNet (
    CMcsUsbNet^ mcsusb )
```

11.74.2 Member Function Documentation

11.74.2.1 FeedbackGetSampleTimerCount()

```
unsigned int FeedbackGetSampleTimerCount (
    [System::Runtime::InteropServices::Out] unsigned int% CurrentCount,
    [System::Runtime::InteropServices::Out] unsigned int% LastKnownCount,
    [System::Runtime::InteropServices::Out] bool% On )
```

11.74.2.2 FeedbackGetTriggerTotzeitFactor()

```
void FeedbackGetTriggerTotzeitFactor (
    [System::Runtime::InteropServices::Out] unsigned short% trigger_factor,
    [System::Runtime::InteropServices::Out] unsigned short% totzeit_factor )
```

11.74.2.3 FeedbackSetAnalogSource()

```
void FeedbackSetAnalogSource (
    AnalogSourceEnumNet AnalogSource,
    unsigned int Channels,
    unsigned int Offset )
```

11.74.2.4 FeedbackSetChannelFilter()

```
void FeedbackSetChannelFilter (
    short channel,
    char filter )
```

11.74.2.5 FeedbackSetCreateFilter()

```
void FeedbackSetCreateFilter (
    unsigned char filter,
    CCreateFilterNet^ createFilter )
```

11.74.2.6 FeedbackSetDigitalMapping()

```
void FeedbackSetDigitalMapping (
    unsigned short channel,
    unsigned short outmapping,
    unsigned short inmapping )
```

11.74.2.7 FeedbackSetFeedback()

```
void FeedbackSetFeedback (
    unsigned char on,
    unsigned short digoutmask,
    unsigned short diginmask )
```

11.74.2.8 FeedbackSetFilterOff()

```
void FeedbackSetFilterOff ( )
```

11.74.2.9 FeedbackSetFilterParameter()

```
void FeedbackSetFilterParameter (
    unsigned char filter,
    array< short >^ parameters )
```

11.74.2.10 FeedbackSetFilterParameter32()

```
void FeedbackSetFilterParameter32 (
  unsigned char filter,
  array< int >^ parameters )
```

11.74.2.11 FeedbackSetGlobalChannelFilter()

```
void FeedbackSetGlobalChannelFilter (
  char filter,
  unsigned short firstchannel,
  unsigned short lastchannel )
```

11.74.2.12 FeedbackSetIIRFilterParameter()

```
void FeedbackSetIIRFilterParameter (
  unsigned char filter,
  int length,
  array< double >^ parameters )
```

11.74.2.13 FeedbackSetLogic()

```
void FeedbackSetLogic (
  unsigned short position,
  array< unsigned short >^ sourcechannel,
  unsigned short resultchannel,
  unsigned int lookup )
```

11.74.2.14 FeedbackSetMkFilter()

```
void FeedbackSetMkFilter (
  unsigned char filter,
  String^ filtertype,
  double cheb_ribble,
  String^ passtype,
  int order,
  double alpha1,
  double alpha2 )
```

11.74.2.15 FeedbackSetNumberOfLogics()

```
void FeedbackSetNumberOfLogics (
  unsigned short number )
```

11.74.2.16 FeedbackSetNumberOfRateCounter()

```
void FeedbackSetNumberOfRateCounter (
  unsigned short number )
```

11.74.2.17 FeedbackSetNumberOfRateDetectors()

```
void FeedbackSetNumberOfRateDetectors (
    unsigned short number )
```

11.74.2.18 FeedbackSetNumberOfSpikeDetectors()

```
void FeedbackSetNumberOfSpikeDetectors (
    unsigned short number )
```

11.74.2.19 FeedbackSetNumberOfTriggers()

```
void FeedbackSetNumberOfTriggers (
    unsigned short number )
```

11.74.2.20 FeedbackSetRateCounter()

```
void FeedbackSetRateCounter (
    unsigned short position,
    unsigned short sourcechannel,
    unsigned short resultchannel )
```

11.74.2.21 FeedbackSetRateDetector()

```
void FeedbackSetRateDetector (
    unsigned short position,
    unsigned short resultchannel,
    unsigned short trigger,
    unsigned short totzeit,
    unsigned short pulses,
    unsigned int duration1,
    unsigned int duration2 )
```

11.74.2.22 FeedbackSetSpikeDetectorThreshold()

```
void FeedbackSetSpikeDetectorThreshold (
    unsigned short position,
    unsigned short sourcechannel,
    unsigned short resultchannel,
    unsigned short trigger,
    unsigned short totzeit,
    int threshold1,
    int threshold2,
    short slope )
```

11.74.2.23 FeedbackSetTrigger()

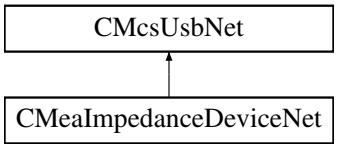
```
void FeedbackSetTrigger (
    unsigned short position,
    unsigned short sourcechannel,
    unsigned short resultchannel,
    unsigned short trigger,
    unsigned short totzeit )
```

11.74.2.24 FeedbackSetTriggerTotzeitFactor() `void FeedbackSetTriggerTotzeitFactor (`

```
unsigned short trigger_factor,
unsigned short totzeit_factor )
```

11.75 CMeaImpedanceDeviceNet Class Reference

Inheritance diagram for CMeaImpedanceDeviceNet:



Public Member Functions

- CMeaImpedanceDeviceNet ()
- ~CMeaImpedanceDeviceNet ()
- virtual void StartMeasurement (unsigned short channel)
- virtual unsigned short GetReady ()
- virtual unsigned short GetArraySize ()
- virtual array< unsigned short > ^ GetResult ()
- unsigned short GetAdapterCode ()
- virtual unsigned int GetImpedanceTestFrequency ()
- virtual void SetImpedanceTestFrequency (unsigned int TestFrequency_Hertz)

Additional Inherited Members

11.75.1 Constructor & Destructor Documentation

11.75.1.1 CMeaImpedanceDeviceNet() `CMeaImpedanceDeviceNet ( )`

11.75.1.2 ~CMeaImpedanceDeviceNet() `~CMeaImpedanceDeviceNet ( )`

11.75.2 Member Function Documentation

11.75.2.1 GetAdapterCode() `unsigned short GetAdapterCode ( )`

11.75.2.2 GetArraySize() virtual unsigned short GetArraySize ( ) [virtual]

11.75.2.3 GetImpedanceTestFrequency() virtual unsigned int GetImpedanceTestFrequency ( ) [virtual]

11.75.2.4 GetReady() virtual unsigned short GetReady ( ) [virtual]

11.75.2.5 GetResult() virtual array<unsigned short> ^ GetResult ( ) [virtual]

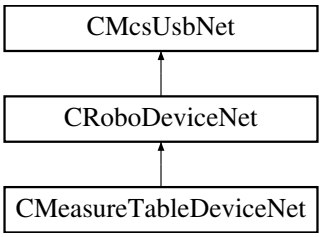
11.75.2.6 SetImpedanceTestFrequency() virtual void SetImpedanceTestFrequency (unsigned int *TestFrequency_Hertz*) [virtual]

11.75.2.7 StartMeasurement() virtual void StartMeasurement (unsigned short *channel*) [virtual]

11.76 CMeasureTableDeviceNet Class Reference

CMeasureTableDeviceNet is the to control the MCS HLA device

Inheritance diagram for CMeasureTableDeviceNet:



Public Member Functions

- CMeasureTableDeviceNet (void)

Properties

- CMcsBus_SensorNet^ Sensor [get]

Additional Inherited Members

11.76.1 Detailed Description

CMeasureTableDeviceNet is the to control the MCS HLA device

11.76.2 Constructor & Destructor Documentation

11.76.2.1 CMeasureTableDeviceNet() `CMeasureTableDeviceNet ( void )`

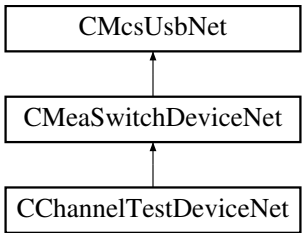
11.76.3 Property Documentation

11.76.3.1 Sensor `CMcsBus_SensorNet^ Sensor [get]`

11.77 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 he 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

11.77.1 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.

11.77.2 Constructor & Destructor Documentation

11.77.2.1 CMeaSwitchDeviceNet() `CMeaSwitchDeviceNet ( )`

Constructor.

11.77.2.2 ~CMeaSwitchDeviceNet() `~CMeaSwitchDeviceNet ( )`

Destructor.

11.77.3 Member Function Documentation

11.77.3.1 GetNumber() `unsigned short GetNumber ( )`

Gets the number of boards in the device.

The MEA-Switch are delivered with 64 or 128 channels

11.77.3.2 GetPattern() `array<unsigned char> ^ GetPattern ( )`

Gets the pattern of the switches that are currently set in the device as char array.

11.77.3.3 GetPatternBool() `array<bool> ^ GetPatternBool ( )`

Gets the pattern of the switches that are currently set in he device as bools.

11.77.3.4 SetPattern() void SetPattern (array< unsigned char >^ *pattern*)

Sets the pattern of switches from a char array.

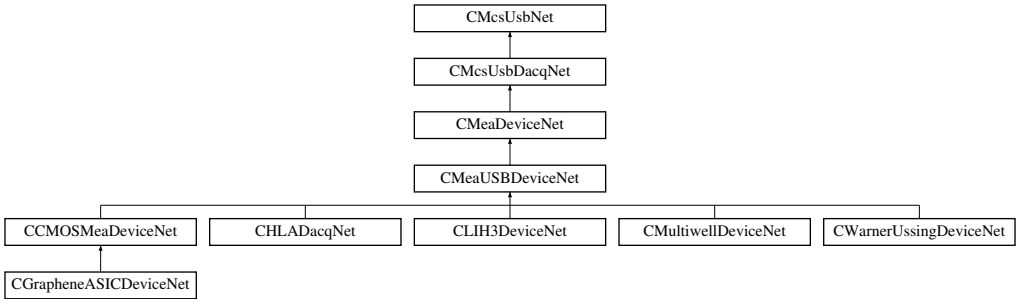
11.77.3.5 SetPatternBool() void SetPatternBool (array< bool >^ *pattern*)

Sets the pattern of switches from a.

11.78 CMeaUSBDeviceNet Class Reference

Class for data acquisition via ME and MEA USB amplifiers

Inheritance diagram for CMeaUSBDeviceNet:



Public Member Functions

- CMeaUSBDeviceNet (OnChannelData^ channelData, OnError^ error)
  Initializes a new instance of CMeaDeviceNet class.
- CMeaUSBDeviceNet ()
  Initializes a new instance of CMeaDeviceNet class.
- ~CMeaUSBDeviceNet ()
- void SetVoltageRangeForIndex (uint16_t index, uint32_t range)

Additional Inherited Members

11.78.1 Detailed Description

Class for data acquisition via ME and MEA USB amplifiers

11.78.2 Constructor & Destructor Documentation

11.78.2.1 CMeaUSBDeviceNet() [1/2] CMeaUSBDeviceNet (OnChannelData^ *channelData*, OnError^ *error*)

Initializes a new instance of CMeaDeviceNet class.

Parameters

channelData	Handler to call when new data is available.

Parameters

error	Handler to call when an error occurs.

11.78.2.2 CMeaUSBDeviceNet() [2/2] CMeaUSBDeviceNet ( )

Initializes a new instance of CMeaDeviceNet class.

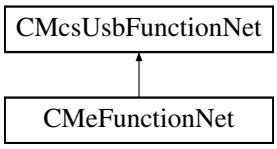
11.78.2.3 ~CMeaUSBDeviceNet() ~CMeaUSBDeviceNet ( )

11.78.3 Member Function Documentation

11.78.3.1 SetVoltageRangeForIndex() void SetVoltageRangeForIndex (uint16_t *index*, uint32_t *range*)

11.79 CMeFunctionNet Class Reference

Inheritance diagram for CMeFunctionNet:



Public Member Functions

- CMeFunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ meFunctionPointer↩ Container)

  Initializes a new instance of the CDacCalibrationFunctionNet class.
- CMeFunctionNet (CMcsUsbNet^ mcsusb)
- virtual ~CMeFunctionNet (void)
- !CMeFunctionNet (void)
- void SetTransformer (unsigned int index, bool onoff)

Additional Inherited Members

11.79.1 Detailed Description

11.79.2 Constructor & Destructor Documentation

11.79.2.1 CMeFunctionNet() [1/2]

```
CMeFunctionNet (
    CMcsUsbNet^ mcsusb,
    CMcsUsbFunctionPointerContainer^ meFunctionPointerContainer )
```

Initializes a new instance of the CDacCalibrationFunctionNet class.

11.79.2.2 CMeFunctionNet() [2/2]

```
CMeFunctionNet (
    CMcsUsbNet^ mcsusb )
```

11.79.2.3 ~CMeFunctionNet()

```
virtual ~CMeFunctionNet (
    void ) [virtual]
```

11.79.2.4 "!CMeFunctionNet()

```
!CMeFunctionNet (
    void )
```

11.79.3 Member Function Documentation

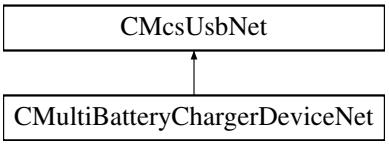
11.79.3.1 SetTransformer()

```
void SetTransformer (
    unsigned int index,
    bool onoff )
```

11.80 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: µAh
- uint32_t GetChargeCapacity (uint32_t NrChannel)

  gets the charge capacity; unit: µAh
- 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

11.80.1 Detailed Description

CMultiBatteryChargerDeviceNet is the class to access the MBC-08 device.

11.80.2 Constructor & Destructor Documentation

11.80.2.1 CMultiBatteryChargerDeviceNet() `CMultiBatteryChargerDeviceNet ( )`

Initializes a new instance of the CMultiBatteryChargerDeviceNet class.

11.80.2.2 ~CMultiBatteryChargerDeviceNet() `virtual ~CMultiBatteryChargerDeviceNet ( ) [virtual]`

11.80.2.3 "!CMultiBatteryChargerDeviceNet() `!CMultiBatteryChargerDeviceNet ( )`

11.80.3 Member Function Documentation

11.80.3.1 CapacityTest() `void CapacityTest ( uint32_t NrChannel )`

start capacity test on channel

Parameters

NrChannel	the channel number

11.80.3.2 ChannelReset() `void ChannelReset ( uint32_t NrChannel )`

cancel charging and capacity test functions; check if battery is connected

Parameters

NrChannel	the channel number

11.80.3.3 GetBatteryVoltage() `uint32_t GetBatteryVoltage ( uint32_t NrChannel )`

gets the battery voltage; unit: mV

Parameters

NrChannel	the channel number

Returns

the battery voltage in mV

11.80.3.4 GetChannels() `uint32_t GetChannels ( )`

gets number of channels

Returns

number of channels

11.80.3.5 GetChannelState() `MbcChannelStateEnumNet GetChannelState ( uint32_t NrChannel )`

gets the channel state: IdleNoBattery, IdleChargeFinished, CapacityTestPreCharge, CapacityTestDischarge, StorageCharge, LowCurrentCharge, HighCurrentCharge

Parameters

NrChannel	the channel number

Returns

the current state

11.80.3.6 GetChargeCapacity() `uint32_t GetChargeCapacity ( uint32_t NrChannel )`

gets the charge capacity; unit: µAh

Parameters

NrChannel	the channel number

Returns

the capacity in uAh

11.80.3.7 GetChargeCurrent() `uint32_t GetChargeCurrent ( uint32_t NrChannel )`

gets the charge current; unit: mA

Parameters

NrChannel	the channel number

Returns

the measured charge current in mA

11.80.3.8 GetChargingMode() `MbcChargingModeEnumNet GetChargingMode ( uint32_t NrChannel )`

gets the charging mode: StorageCharge, LowCurrentCharge and HighCurrentCharge

Parameters

NrChannel	the channel number

Returns

the charging mode

11.80.3.9 GetChargingPCoefficient() `uint32_t GetChargingPCoefficient ( )`

gets the p-coefficient for charging in mA/V / nominal charging current

Returns

the p-coefficient

11.80.3.10 GetDischargeCapacity() uint32_t GetDischargeCapacity (uint32_t *NrChannel*)

gets the discharge capacity; unit: µAh

Parameters

NrChannel	the channel number

Returns

the capacity in uAh

11.80.3.11 GetDischargeCurrent() uint32_t GetDischargeCurrent (uint32_t *NrChannel*)

gets the discharge current; unit: mA

Parameters

NrChannel	the channel number

Returns

the measured discharge current in mA

11.80.3.12 GetDischargeCurrentSetPoint() uint32_t GetDischargeCurrentSetPoint (uint32_t *NrChannel*)

gets the setpoint for the discharge current; unit: mA

Parameters

NrChannel	the channel number

Returns

the discharge current in mA

11.80.3.13 GetFinalDischargeVoltage() uint32_t GetFinalDischargeVoltage (uint32_t *NrChannel*)

gets the final discharge voltage; unit: mV

Parameters

NrChannel	the channel number

Returns

the battery voltage in mV at the end of discharge

11.80.3.14 GetRatedCapacity() MbcRatedCapacityEnumNet GetRatedCapacity (uint32_t *NrChannel*)

gets the rated capacity

Parameters

NrChannel	the channel number

Returns

the capacity

11.80.3.15 SetChargingMode() void SetChargingMode (uint32_t *NrChannel*, MbcChargingModeEnumNet *NewOperatingMode*)

sets the charging mode: StorageCharge, LowCurrentCharge and HighCurrentCharge

Parameters

NrChannel	the channel number
NewOperatingMode	the charging mode

11.80.3.16 SetChargingPCoefficient() void SetChargingPCoefficient (uint32_t *pCoefficient*)

sets the p-coefficient for charging in mA/V / nominal charging current

Parameters

pCoefficient	the p-coefficient

11.80.3.17 SetDischargeCurrentSetPoint()

```
void SetDischargeCurrentSetPoint (
    uint32_t NrChannel,
    uint32_t DischargeCurrent_mA )
```

sets the setpoint for the discharge current; unit: mA

Parameters

NrChannel	the channel number
DischargeCurrent_mA	the discharge current in mA

11.80.3.18 SetFinalDischargeVoltage()

```
void SetFinalDischargeVoltage (
    uint32_t NrChannel,
    uint32_t FinalDischargeVoltage_mV )
```

sets the final discharge voltage; unit: mV

Parameters

NrChannel	the channel number
FinalDischargeVoltage_mV	the battery voltage in mV at the end of discharge

11.80.3.19 SetRatedCapacity()

```
void SetRatedCapacity (
    uint32_t NrChannel,
    MbcRatedCapacityEnumNet NewRatedCapacity )
```

sets the rated capacity

Parameters

NrChannel	the channel number
NewRatedCapacity	the capacity

11.80.3.20 SetRatedCapacityVolatile()

```
void SetRatedCapacityVolatile (
    uint32_t NrChannel,
    MbcRatedCapacityEnumNet NewRatedCapacity )
```

sets the rated capacity (i.e. charge current) without storing it persistently

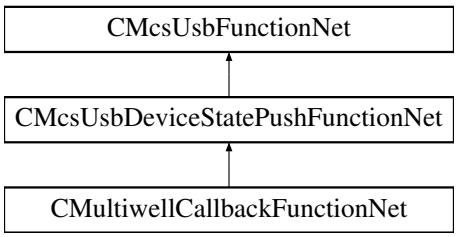
Parameters

NrChannel	the channel number
NewRatedCapacity	the capacity

11.81 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^ pMultiwell↩ CallbackFunctionPointerContainer)

  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

11.81.1 Detailed Description

CMultiwellCallbackFunctionNet is the class to access the Multiwell-Mini-Stimulator

11.81.2 Constructor & Destructor Documentation

11.81.2.1 CMultiwellCallbackFunctionNet() [1/2]

```
CMultiwellCallbackFunctionNet (
    CMcsUsbNet^ mcsusb,
    CMcsUsbFunctionPointerContainer^ pMultiwellCallbackFunctionPointerContainer )
```

Initializes a new instance of the CMultiwellCallbackFunctionNet class.

11.81.2.2 CMultiwellCallbackFunctionNet() [2/2]

```
CMultiwellCallbackFunctionNet ( CMcsUsbNet^ mcsusb )
```

11.81.2.3 ~CMultiwellCallbackFunctionNet()

```
virtual ~CMultiwellCallbackFunctionNet ( ) [virtual]
```

11.81.2.4 "!CMultiwellCallbackFunctionNet()

```
!CMultiwellCallbackFunctionNet ( )
```

11.81.3 Member Function Documentation

11.81.3.1 GetPlateClampStateByHeadstage()

```
PlateClampEnumNet GetPlateClampStateByHeadstage ( uint32_t Headstage )
```

Gets the state of the plate

Parameters

Headstage	The headstage number

Returns

The plate state

11.81.3.2 OnGetPlateClampStateByHeadstage()

```
delegate void OnGetPlateClampStateByHeadstage ( uint32_t Headstage,
PlateClampEnumNet plateState )
```

11.81.4 Event Documentation

11.81.4.1 GetPlateClampStateByHeadstageEvent

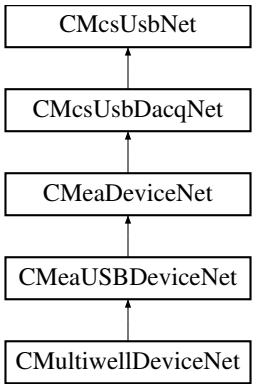
```
OnGetPlateClampStateByHeadstage^ GetPlateClamp↩
StateByHeadstageEvent [add], [remove], [raise]
```

Event fires when the plate state for the headstage number has changed

11.82 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

11.82.1 Detailed Description

CMultiwellDeviceNet is the class to access the Multiwell device.

11.82.2 Constructor & Destructor Documentation

11.82.2.1 CMultiwellDeviceNet() `CMultiwellDeviceNet ( )`

Initializes a new instance of the CMultiwellDeviceNet class.

11.82.2.2 ~CMultiwellDeviceNet() `virtual ~CMultiwellDeviceNet ( ) [virtual]`

11.82.2.3 "!CMultiwellDeviceNet() `!CMultiwellDeviceNet ( )`

11.82.3 Member Function Documentation

11.82.3.1 ClosePlateClamp() `void ClosePlateClamp ( )`

Closes the plate clamp.

11.82.3.2 GetPlateClampLockState() `uint32_t GetPlateClampLockState ( )`

Gets the state of the plate clamp lock.

Returns

the state of the plate lock (unlocked/locked)

11.82.3.3 GetPlateClampState() [1/2] `PlateClampEnumNet GetPlateClampState ( )`

Gets the state of the Multiwell plate clamp.

Returns

the state of the plate clamp (open/closed)

11.82.3.4 GetPlateClampState() [2/2] `PlateClampEnumNet GetPlateClampState ( uint32_t Headstage )`

Gets the state of the plate

Parameters

Headstage	The headstage number

Returns

The plate state

11.82.3.5 GetPlateMux() [1/2] `uint32_t GetPlateMux ( )`

Gets the selected quarter of the electrodes on a high density Multiwell plate.

Returns

the selected quarter

11.82.3.6 GetPlateMux() [2/2] uint32_t GetPlateMux (uint32_t *Headstage*)

Gets the selected quarter of the electrodes on a high density Multiwell plate.

Parameters

Headstage	The headstage to query.

Returns

the selected quarter

11.82.3.7 GetPlateType() [1/2] MultiwellPlateTypeEnumNet GetPlateType ( )

Gets the plate type.

Returns

the plate type

11.82.3.8 GetPlateType() [2/2] MultiwellPlateTypeEnumNet GetPlateType (uint32_t *Headstage*)

Gets the plate type.

Parameters

Headstage	The headstage to query.

Returns

the plate type

11.82.3.9 GetPowerMuxPlate() bool GetPowerMuxPlate (uint32_t *Headstage*)

On the Multiwell Mini device, Query if Power to the MUX Plate is On or Off.

Parameters

Headstage	The headstage to query.

Returns

"true" Power is On, "false" Power is Off

11.82.3.10 GetTouchPadEnable() `bool GetTouchPadEnable ( uint32_t Headstage )`

Manual opening/closing of plate clamp via touch pad can be disabled.

Parameters

Headstage	The headstage to query.

Returns

"true" when plate clamp can be driven manually, otherwise "false".

11.82.3.11 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

Headstage	The headstage to be affected.

Returns

Clamp distance reduction in um.

11.82.3.12 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".

11.82.3.13 IsPlateTypeValid() [2/2] bool IsPlateTypeValid (uint32_t *Headstage*)

Checks whether the plate type is valid, meaning all pins have contact.

Parameters

Headstage	The headstage to query.

Returns

"true" when all pins have contact, otherwise "false".

11.82.3.14 LockPlateClamp() void LockPlateClamp ( )

Locks the plate clamp.

11.82.3.15 OpenPlateClamp() void OpenPlateClamp ( )

Opens the plate clamp.

11.82.3.16 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

Headstage	The headstage to query.
muxSelection	the selected quarter

11.82.3.17 SetPlateMux() [2/2] void SetPlateMux (uint32_t *muxSelection*)

Selects a one quarter of the electrodes on a high density Multiwell plate.

Parameters

muxSelection	the selected quarter

11.82.3.18 SetPlateType() [1/2] void SetPlateType (MultiwellPlateTypeEnumNet *plateType*)

Sets the plate type.

Parameters

plateType	the plate type

11.82.3.19 SetPlateType() [2/2] void SetPlateType (uint32_t *Headstage*, MultiwellPlateTypeEnumNet *plateType*)

Sets the plate type.

Parameters

Headstage	The headstage to query.
plateType	the plate type

11.82.3.20 SetPowerMuxPlate() void SetPowerMuxPlate (uint32_t *Headstage*, bool *powerOn*)

On the Multiwell Mini device, turn Power to the MUX Plate On or Off.

Parameters

Headstage	The headstage to query.
powerOn	"true" to turn Power On, "false" to turn Power Off

11.82.3.21 SetTouchPadEnable() void SetTouchPadEnable (uint32_t *Headstage*, bool *Enable*)

Enables or disables manual opening/closing of plate clamp via touch pad.

Parameters

Headstage	The headstage to be affected.
Enable	"true" when plate clamp shall be driven manually, otherwise "false".

11.82.3.22 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

Headstage	The headstage to be affected.
CoverLipThickness_um	Clamp distance reduction in um. Range: 200um..400um; Typical value: 300um

11.82.3.23 StopPlateClamp() `void StopPlateClamp ( )`

Stops the plate clamp movement.

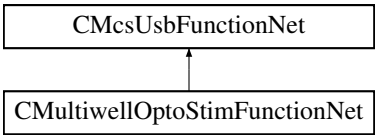
11.82.3.24 UnlockPlateClamp() `void UnlockPlateClamp ( )`

Unlocks the plate clamp.

11.83 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^ mcsusb, CMcsUsbFunctionPointerContainer^ pMultiwell↩ OptoStimFunctionPointerContainer)

  Initializes a new instance of the CMultiwellOptoStimFunctionNet class.
- CMultiwellOptoStimFunctionNet (CMcsUsbNet^ mcsusb)
- virtual ~CMultiwellOptoStimFunctionNet ()
- !CMultiwellOptoStimFunctionNet ()
- uint32_t GetWaveLengthInNanometer (uint16_t channel)
- uint32_t GetAbsMaxCurrentInMicroAmp (uint16_t channel)
- uint32_t GetMaxDurationHighCurrentInMicroSec (uint16_t channel)

- uint32_t GetMaxDutyCycleHighCurrent (uint16_t channel)
- uint32_t GetPermanentCurrentInMicroAmp (uint16_t channel)
- uint32_t GetColorRgb (uint16_t channel)
- String ^ GetColorStr (uint16_t channel)
- void SetWaveLengthInNanometer (uint16_t channel, uint32_t WaveLength_nm)
- void SetAbsMaxCurrentInMicroAmp (uint16_t channel, uint32_t AbsoluteMaxCurrent_uA)
- void SetMaxDurationHighCurrentInMicroSec (uint16_t channel, uint32_t AbsoluteMaxDuration_us)
- void SetMaxDutyCycleHighCurrent (uint16_t channel, uint32_t MaxDutyCycleHighCurrent)
- void SetPermanentCurrentInMicroAmp (uint16_t channel, uint32_t PermanentCurrent_uA)
- void SetColorRgb (uint16_t channel, uint32_t ColorRGB)
- void SetColorStr (uint16_t channel, String^ ColorString)

Additional Inherited Members

11.83.1 Detailed Description

CMultiwellOptoStimFunctionNet is the class to access the optical properties of the Multiwell Optostim device

11.83.2 Constructor & Destructor Documentation

11.83.2.1 CMultiwellOptoStimFunctionNet() [1/2]

```
CMultiwellOptoStimFunctionNet (
    CMcsUsbNet^ mcsusb,
    CMcsUsbFunctionPointerContainer^ pMultiwellOptoStimFunctionPointerContainer )
```

Initializes a new instance of the CMultiwellOptoStimFunctionNet class.

11.83.2.2 CMultiwellOptoStimFunctionNet() [2/2]

```
CMultiwellOptoStimFunctionNet (
    CMcsUsbNet^ mcsusb )
```

11.83.2.3 ~CMultiwellOptoStimFunctionNet()

```
virtual ~CMultiwellOptoStimFunctionNet ( ) [virtual]
```

11.83.2.4 "!CMultiwellOptoStimFunctionNet()

```
!CMultiwellOptoStimFunctionNet ( )
```

11.83.3 Member Function Documentation

11.83.3.1 GetAbsMaxCurrentInMicroAmp()

```
uint32_t GetAbsMaxCurrentInMicroAmp (
    uint16_t channel )
```

Parameters

channel	the (analog) channel number

Returns

absolute max. current; unit: uA

11.83.3.2 GetColorRgb() `uint32_t GetColorRgb ( uint16_t channel )`

Parameters

channel	the (analog) channel number

Returns

RGB-value of LED color

11.83.3.3 GetColorStr() `String ^ GetColorStr ( uint16_t channel )`

Parameters

channel	the (analog) channel number

Returns

LED color as string

11.83.3.4 GetMaxDurationHighCurrentInMicroSec() `uint32_t GetMaxDurationHighCurrentInMicroSec ( uint16_t channel )`

Parameters

channel	the (analog) channel number

Returns

max. duration the LED can stand the abs. max current; unit: us

11.83.3.5 GetMaxDutyCycleHighCurrent() uint32_t GetMaxDutyCycleHighCurrent (uint16_t *channel*)

Parameters

channel	the (analog) channel number

Returns

max. duty cycle at max. current; unit: 100∗%

11.83.3.6 GetPermanentCurrentInMicroAmp() uint32_t GetPermanentCurrentInMicroAmp (uint16_t *channel*)

Parameters

channel	the (analog) channel number

Returns

max. current the LED can stand when always switched on; unit: uA

11.83.3.7 GetWaveLengthInNanometer() uint32_t GetWaveLengthInNanometer (uint16_t *channel*)

Parameters

channel	the (analog) channel number

Returns

wavelength of this channel's LEDs; unit: nm

11.83.3.8 SetAbsMaxCurrentInMicroAmp() void SetAbsMaxCurrentInMicroAmp (uint16_t *channel*, uint32_t *AbsoluteMaxCurrent_uA*)

Parameters

channel	the (analog) channel number
AbsoluteMaxCurrent_uA	absolute max. current; unit: uA

11.83.3.9 SetColorRgb()

```
void SetColorRgb (
    uint16_t channel,
    uint32_t ColorRGB )
```

Parameters

channel	the (analog) channel number
ColorRGB	RGB-value of LED color

11.83.3.10 SetColorStr()

```
void SetColorStr (
    uint16_t channel,
    String^ ColorString )
```

Parameters

channel	the (analog) channel number
ColorString	LED color as string

11.83.3.11 SetMaxDurationHighCurrentInMicroSec()

```
void SetMaxDurationHighCurrentInMicroSec (
    uint16_t channel,
    uint32_t AbsoluteMaxDuration_us )
```

Parameters

channel	the (analog) channel number
AbsoluteMaxDuration_us	max. duration the LED can stand the abs. max current; unit: us

11.83.3.12 SetMaxDutyCycleHighCurrent()

```
void SetMaxDutyCycleHighCurrent (
    uint16_t channel,
    uint32_t MaxDutyCycleHighCurrent )
```

Parameters

channel	the (analog) channel number
MaxDutyCycleHighCurrent	max. duty cycle at max. current; unit: 100∗%

11.83.3.13 SetPermanentCurrentInMicroAmp()

```
void SetPermanentCurrentInMicroAmp (
```

```
uint16_t channel,
uint32_t PermanentCurrent_uA )
```

Parameters

channel	the (analog) channel number
PermanentCurrent_uA	max. current the LED can stand when always switched on; unit: uA

11.83.3.14 SetWaveLengthInNanometer() `void SetWaveLengthInNanometer (`

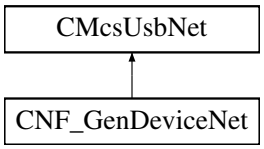
```
uint16_t channel,
uint32_t WaveLength_nm )
```

Parameters

channel	the (analog) channel number
WaveLength_nm	wavelength of this channel's LEDs; unit: nm

11.84 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

11.84.1 Constructor & Destructor Documentation

11.84.1.1 CNF_GenDeviceNet() `CNF_GenDeviceNet (`

```
void )
```

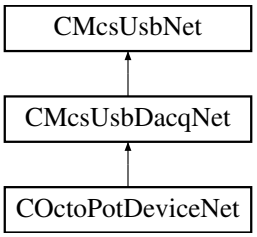
11.84.1.2 ~CNF_GenDeviceNet() ~CNF_GenDeviceNet (
void)

11.84.2 Member Function Documentation

11.84.2.1 Set_Values() void Set_Values (
unsigned int *frequency*,
unsigned int *amplitude*)

11.85 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

11.85.1 Constructor & Destructor Documentation

11.85.1.1 COctoPotDeviceNet() [1/2] COctoPotDeviceNet (void)

11.85.1.2 COctoPotDeviceNet() [2/2] COctoPotDeviceNet (OnChannelData^ *channelData*, OnError^ *error*)

11.85.2 Member Function Documentation

11.85.2.1 BurnAdcOffset() uint32_t BurnAdcOffset ( )

11.85.2.2 BurnDacOffset() uint32_t BurnDacOffset ( )

11.85.2.3 EnableChecksum() uint32_t EnableChecksum (bool *enable*)

11.85.2.4 EnableDigitalIn() uint32_t EnableDigitalIn (bool *enable*)

11.85.2.5 EnableTimestamp() uint32_t EnableTimestamp (bool *enable*)

11.85.2.6 GetAdcOffset() uint32_t GetAdcOffset (unsigned int *channel*, [System::Runtime::InteropServices::Out] int ^ *offset*)

11.85.2.7 GetDacOffset()

```
uint32_t GetDacOffset (
    unsigned int channel,
    [System::Runtime::InteropServices::Out] int ^ offset )
```

11.85.2.8 PatternListStart()

```
uint32_t PatternListStart (
    int channelmap )
```

11.85.2.9 RampStart()

```
uint32_t RampStart (
    int channelmap )
```

11.85.2.10 ResetAdcOffset()

```
uint32_t ResetAdcOffset (
    unsigned int channel )
```

11.85.2.11 ResetDacOffset()

```
uint32_t ResetDacOffset (
    unsigned int channel )
```

11.85.2.12 SetAdcOffset()

```
uint32_t SetAdcOffset (
    unsigned int channel,
    int offset )
```

11.85.2.13 SetAmplificationSwitch()

```
uint32_t SetAmplificationSwitch (
    unsigned int channel,
    unsigned int state )
```

11.85.2.14 SetBathclamp()

```
uint32_t SetBathclamp (
    unsigned int block,
    bool enable )
```

11.85.2.15 SetChannelSwitch()

```
uint32_t SetChannelSwitch (
    unsigned int channel,
    unsigned int state )
```

11.85.2.16 SetDacAutoControl()

```
uint32_t SetDacAutoControl (
    unsigned int channel )
```

11.85.2.17 SetDacOffset()

```
uint32_t SetDacOffset (
    unsigned int channel,
    int offset )
```

11.85.2.18 SetDacValue()

```
uint32_t SetDacValue (
    int channel,
    int value )
```

11.85.2.19 SetNumberOfChannels()

```
uint32_t SetNumberOfChannels (
    unsigned int NumberOfChannels )
```

11.85.2.20 SetOutputRate()

```
uint32_t SetOutputRate (
    uint32_t rate )
```

11.85.2.21 SetPatternListEntry()

```
uint32_t SetPatternListEntry (
    unsigned int channel,
    unsigned int position,
    unsigned int duration,
    int value )
```

11.85.2.22 SetPidParameter()

```
uint32_t SetPidParameter (
    unsigned int channel,
    int const_p,
    int const_i,
    int shift_p,
    int shift_i )
```

11.85.2.23 SetRampParameter()

```
uint32_t SetRampParameter (
    unsigned int channel,
    int start,
    int min,
    int max,
    int slope,
    int slope2,
    int pause,
    unsigned int samples )
```

11.85.2.24 SetSineParameter() uint32_t SetSineParameter (

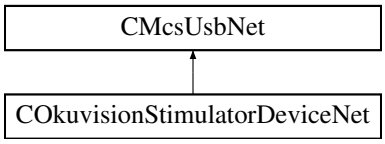
```
unsigned int channel,
int amplitude )
```

11.85.2.25 SineStart() uint32_t SineStart (

```
int channelmap )
```

11.86 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

11.86.1 Constructor & Destructor Documentation

11.86.1.1 COkuvisionStimulatorDeviceNet()

```
COkuvisionStimulatorDeviceNet (
    void  )
```

11.86.1.2 ~COkuvisionStimulatorDeviceNet()

```
~COkuvisionStimulatorDeviceNet (
    void  )
```

11.86.2 Member Function Documentation

11.86.2.1 GetCheckVoltage()

```
int GetCheckVoltage (
    int channel )
```

11.86.2.2 GetCurrentFactor()

```
int GetCurrentFactor (
    int channel )
```

11.86.2.3 GetDACOffset()

```
int GetDACOffset (
    int channel,
    int part )
```

11.86.2.4 GetMaxPower()

```
int GetMaxPower (
    int channel )
```

11.86.2.5 GetMaxVoltage()

```
int GetMaxVoltage (
    int channel )
```

11.86.2.6 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 )
```

11.86.2.7 GetRTC() [1/2]

```
DateTime GetRTC ( )
```

11.86.2.8 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 )
```

11.86.2.9 GetStimulatorStatus()

```
void GetStimulatorStatus (
    [System::Runtime::InteropServices::Out] int% startstop,
    [System::Runtime::InteropServices::Out] int% last_error,
    [System::Runtime::InteropServices::Out] int% battery_status )
```

11.86.2.10 GetVoltage()

```
int GetVoltage (
    int channel )
```

11.86.2.11 SetCheckVoltage()

```
void SetCheckVoltage (
    int channel,
    int voltage )
```

11.86.2.12 SetCurrentFactor()

```
void SetCurrentFactor (
    int channel,
    int factor )
```

11.86.2.13 SetDACOffset()

```
void SetDACOffset (
    int channel,
    int part,
    int offset )
```

11.86.2.14 SetMaxPower()

```
void SetMaxPower (
    int channel,
    int power )
```

11.86.2.15 SetMaxVoltage()

```
void SetMaxVoltage (
    int channel,
    int voltage )
```

11.86.2.16 SetPulseform()

```
void SetPulseform (
    int channel,
    int current,
    int pulsewidth,
    int periode,
    int duration )
```

11.86.2.17 SetRTC() [1/2]

```
void SetRTC (
    DateTime timestamp )
```

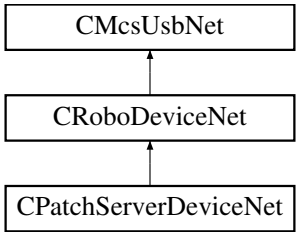
11.86.2.18 SetRTC() [2/2]

```
void SetRTC (
    uint8_t year,
    uint8_t month,
    uint8_t day,
    uint8_t hour,
    uint8_t minute,
    uint8_t second )
```

11.87 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

11.87.1 Detailed Description

CPatchServerDeviceNet is the class to control the MCS PatchServer device

11.87.2 Constructor & Destructor Documentation

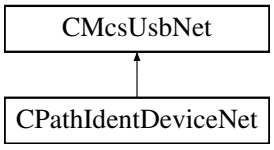
11.87.2.1 CPatchServerDeviceNet() CPatchServerDeviceNet (void)

11.87.3 Property Documentation

11.87.3.1 Sensor CMcsBus_SensorNet^ Sensor [get]

11.88 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::Interop↩ Services::Out] unsigned int% amplitude)

Additional Inherited Members

11.88.1 Constructor & Destructor Documentation

11.88.1.1 CPathIdentDeviceNet() CPathIdentDeviceNet (
void)

11.88.1.2 ~CPathIdentDeviceNet() ~CPathIdentDeviceNet (
void)

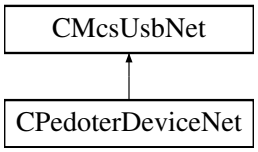
11.88.2 Member Function Documentation

11.88.2.1 Measure() void Measure (
[System::Runtime::InteropServices::Out] unsigned int% *phase*,
[System::Runtime::InteropServices::Out] unsigned int% *amplitude*)

11.88.2.2 Set_Values() void Set_Values (
unsigned int *frequency*,
unsigned int *amplitude*)

11.89 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

11.89.1 Detailed Description

11.89.2 Constructor & Destructor Documentation

11.89.2.1 CPedoterDeviceNet() `CPedoterDeviceNet ( )`

Initializes a new instance of the CPedoterDeviceNet class.

11.89.2.2 ~CPedoterDeviceNet() `virtual ~CPedoterDeviceNet ( ) [virtual]`

11.89.2.3 "!CPedoterDeviceNet() `!CPedoterDeviceNet ( )`

11.89.3 Member Function Documentation

11.89.3.1 GetCommand() `uint32_t GetCommand ( uint16_t Argument )`

Get value from the pedoter device

Parameters

Argument	argument

Returns

value

11.89.3.2 SetCommand() `void SetCommand ( uint16_t Argument, uint32_t pData )`

Set value on the pedoter device

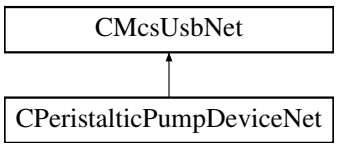
Parameters

Argument	argument
pData	value

11.90 CPeristalticPumpDeviceNet Class Reference

CPeristalticPumpDeviceNet is the class to control a Persistaltic Pump.

Inheritance diagram for CPeristalticPumpDeviceNet:



Public Member Functions

- CPeristalticPumpDeviceNet (void)

  Initialize a new instance of the CPeristalticPumpDeviceNet class.
- ~CPeristalticPumpDeviceNet (void)

Properties

- CMcsBus_MotorControlNet^ McsBus_MotorControl `[get]`

Additional Inherited Members

11.90.1 Detailed Description

CPeristalticPumpDeviceNet is the class to control a Persistaltic Pump.

11.90.2 Constructor & Destructor Documentation

11.90.2.1 CPeristalticPumpDeviceNet() `CPeristalticPumpDeviceNet ( void )`

Initialize a new instance of the CPeristalticPumpDeviceNet class.

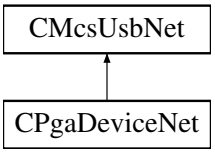
11.90.2.2 ~CPeristalticPumpDeviceNet() ~CPeristalticPumpDeviceNet (void)

11.90.3 Property Documentation

11.90.3.1 McsBus_MotorControl CMcsBus_MotorControlNet^ McsBus_MotorControl [get]

11.91 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

11.91.1 Constructor & Destructor Documentation

11.91.1.1 CPgaDeviceNet() CPgaDeviceNet ( )

11.91.1.2 ~CPgaDeviceNet() ~CPgaDeviceNet ( )

11.91.2 Member Function Documentation

11.91.2.1 ApplyGains() uint32_t ApplyGains ( )

11.91.2.2 DefineAmplification() uint32_t DefineAmplification (

```
int index,
int amplification,
int poti1,
int poti2 )
```

11.91.2.3 DefineFrequencyRange() uint32_t DefineFrequencyRange (

```
int index,
int low,
int high,
int channels,
int gain )
```

11.91.2.4 DefineNumAmplifications() uint32_t DefineNumAmplifications (

```
int number )
```

11.91.2.5 DefineNumFrequencyRanges() uint32_t DefineNumFrequencyRanges (

```
int rnum )
```

11.91.2.6 GetAmplification() uint32_t GetAmplification (

```
int index,
[System::Runtime::InteropServices::Out] int% amplification,
[System::Runtime::InteropServices::Out] int% poti1,
[System::Runtime::InteropServices::Out] int% poti2 )
```

11.91.2.7 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 )
```

11.91.2.8 GetGain()

```
uint32_t GetGain (
    int channel,
    [System::Runtime::InteropServices::Out] int% Gain,
    [System::Runtime::InteropServices::Out] int% poti1,
    [System::Runtime::InteropServices::Out] int% poti2 )
```

11.91.2.9 GetNumAmplifications()

```
uint32_t GetNumAmplifications (
    [System::Runtime::InteropServices::Out] int% number )
```

11.91.2.10 GetNumFrequencyRanges()

```
uint32_t GetNumFrequencyRanges (
    [System::Runtime::InteropServices::Out] int% numRanges )
```

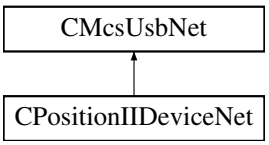
11.91.2.11 SetGain()

```
uint32_t SetGain (
    int channel,
    int Gain,
    int poti1,
    int poti2 )
```

11.92 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::Interop↩ Services::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::Interop↩ Services::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^ RFFunction [get]

Additional Inherited Members

11.92.1 Detailed Description

CPositionIIDeviceNet is the class to control PositionII devices

11.92.2 Constructor & Destructor Documentation

11.92.2.1 CPositionIIDeviceNet() CPositionIIDeviceNet ( )

Initializes a new instance of the CPositionIIDeviceNet class.

11.92.2.2 ~CPositionIIDeviceNet() virtual ~CPositionIIDeviceNet ( ) [virtual]

11.92.2.3 "!CPositionIIDeviceNet() !CPositionIIDeviceNet ( )

11.92.3 Member Function Documentation

11.92.3.1 GetCoilCommunication() uint32_t GetCoilCommunication (uint16_t *coil*)

get if the communication to the coil is working

Parameters

coil	the coil

Returns

is communicating

11.92.3.2 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

coil	the coil
index	the debug entry index number
voltage	the voltage applied
numberofpulses	the number of pulses detected
mediantime	the median time between pulses

11.92.3.3 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

index	the event index number
year	the year
month	the month
day	the day
hour	the hour
minute	the minute
second	the second
coil	the coil
type	the event type
value	the even value

11.92.3.4 GetImplantCurrentSetpoint() uint32_t GetImplantCurrentSetpoint (

```
uint16_t coil )
```

sets the implant current setpoint

Parameters

coil	the coil

Returns

the current

11.92.3.5 GetImplantResult() `uint32_t GetImplantResult ( uint16_t coil )`

gets the last result of the implant pulse trigger

Parameters

coil	the coil

Returns

the result

11.92.3.6 GetImplantState() `uint32_t GetImplantState ( uint16_t coil )`

gets the implantat state

Parameters

coil	the coil

Returns

the state

11.92.3.7 GetOnOff() `uint32_t GetOnOff ( uint16_t coil )`

get if the coil is switched on/off

Parameters

coil	the coil

Returns

0 = off, 1 = on

11.92.3.8 GetPowerStrength() uint32_t GetPowerStrength (uint16_t *coil*)

sets the power for the trigger pulses

Parameters

coil	the coil

Returns

the power

11.92.3.9 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

year	the year
month	the month
day	the day
hour	the hour
minute	the minute
second	the second

11.92.3.10 GetStateDebugData() uint32_t GetStateDebugData (uint16_t *coil*)

get the debug queue state

Parameters

coil	the coil

Returns

the state

11.92.3.11 GetStateEventData() `uint32_t GetStateEventData ( )`

get the event queue state

Returns

the state

11.92.3.12 SetImplantCurrentSetpoint() `void SetImplantCurrentSetpoint (`

```
uint16_t coil,
uint32_t current )
```

gets the implant current setpoint

Parameters

coil	the coil
current	the current

11.92.3.13 SetPowerStrength() `void SetPowerStrength (`

```
uint16_t coil,
uint32_t power )
```

gets the power for the trigger pulses

Parameters

coil	the coil
power	the power

11.92.3.14 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

year	the year
month	the month
day	the day
hour	the hour
minute	the minute
second	the second

11.92.3.15 SetStateDebugData() `void SetStateDebugData (`

```
uint16_t coil,
uint32_t state )
```

clears/starts/stops the debug queue for a certain coil

Parameters

coil	the coil
state	clear/start/stop

11.92.3.16 SetStateEventData() `void SetStateEventData (`

```
uint32_t state )
```

clears/starts/stops the event queue for a certain coil

Parameters

state	clear/start/stop

11.92.3.17 SwitchOnOff() `void SwitchOnOff (`

```
uint16_t coil,
uint32_t on )
```

switched the coild on of

Parameters

coil	the coil
on	0 = off, 1 = on

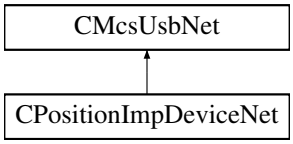
11.92.4 Property Documentation

11.92.4.1 RFFunction `CRFFunctionNet^ RFFunction [get]`

11.93 CPositionImpDeviceNet Class Reference

CPositionImpDeviceNet is the class to access the Position/Imp devices

Inheritance diagram for CPositionImpDeviceNet:



Public Member Functions

- CPositionImpDeviceNet ()
  Initializes a new instance of the CPositionImpDeviceNet class.
- virtual ~CPositionImpDeviceNet ()
- !CPositionImpDeviceNet ()
- void ConnectImp (uint32_t id)
  Connect to a Imp device with a certain ID
- uint32_t ConnectedImp ()
  The ID of the connected Imp device
- int32_t GetRFFrequency ()
  Gets currently used RF frequency
- void SetRFFrequency (int32_t frequency)
  Sets the current RF frequency
- uint32_t GetDeviceList (int32_t index)
  Gets the device list
- void SetDeviceList (int32_t index, uint32_t id)
  Sets the device list
- uint32_t GetImpId ()
  Gets the ID of the impedance measure device
- void SetImpId (uint32_t id)
  Sets the ID of the impedance measure device

Additional Inherited Members

11.93.1 Detailed Description

CPositionImpDeviceNet is the class to access the Position/Imp devices

11.93.2 Constructor & Destructor Documentation

11.93.2.1 CPositionImpDeviceNet() `CPositionImpDeviceNet ( )`

Initializes a new instance of the CPositionImpDeviceNet class.

11.93.2.2 ~CPositionImpDeviceNet() `virtual ~CPositionImpDeviceNet ( ) [virtual]`

11.93.2.3 "!CPositionImpDeviceNet() `!CPositionImpDeviceNet ( )`

11.93.3 Member Function Documentation

11.93.3.1 ConnectedImp() `uint32_t ConnectedImp ( )`

The ID of the connected Imp device

Returns

The ID

11.93.3.2 ConnectImp() `void ConnectImp ( uint32_t id )`

Connect to a Imp device with a certain ID

Parameters

id	The ID

11.93.3.3 GetDeviceList() `uint32_t GetDeviceList ( int32_t index )`

Gets the device list

Parameters

index	the index

Returns

the ID

11.93.3.4 GetImpId() `uint32_t GetImpId ( )`

Gets the ID of the impedance measure device

Returns

the ID

11.93.3.5 GetRFFrequency() `int32_t GetRFFrequency ( )`

Gets currently used RF frequency

Returns

The frequency

11.93.3.6 SetDeviceList() `void SetDeviceList (`

```
int32_t index,
uint32_t id )
```

Sets the device list

Parameters

index	the index
id	the ID

11.93.3.7 SetImpId() `void SetImpId (`

```
uint32_t id )
```

Sets the ID of the impedance measure device

Parameters

id	the ID

11.93.3.8 SetRFFrequency() `void SetRFFrequency (`

`int32_t` *frequency*)

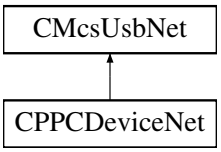
Sets the current RF frequency

Parameters

frequency	The frequency

11.94 CPPCDeviceNet Class Reference

Inheritance diagram for CPPCDeviceNet:



Public Member Functions

- CPPCDeviceNet (void)

Properties

- CPPCFunctionNet^ PPCFunction `[get]`
- CMcsBusNet^ McsBus `[get]`
- CMcsBus_MotorControlNet^ McsBus_MotorControl `[get]`
- CMcsBus_SensorNet^ McsBus_Sensor `[get]`

Additional Inherited Members

11.94.1 Constructor & Destructor Documentation

11.94.1.1 CPPCDeviceNet() `CPPCDeviceNet (`

`void`)

11.94.2 Property Documentation

11.94.2.1 McsBus

CMcsBusNet^ McsBus [get]

11.94.2.2 McsBus_MotorControl

CMcsBus_MotorControlNet^ McsBus_MotorControl [get]

11.94.2.3 McsBus_Sensor

CMcsBus_SensorNet^ McsBus_Sensor [get]

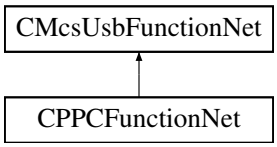
11.94.2.4 PPCFunction

CPPCFunctionNet^ PPCFunction [get]

11.95 CPPCFunctionNet Class Reference

CPPCFunctionNet is the class to access the PPC (high precision Patch Peristalic patch Pump

Inheritance diagram for CPPCFunctionNet:



Public Member Functions

- CPPCFunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ pPPCFunctionPointer↩ Container)

  Initializes a new instance of the CPPCFunctionNet class.
- CPPCFunctionNet (CMcsUsbNet^ 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

11.95.1 Detailed Description

CPPCFunctionNet is the class to access the PPC (high precision Patch Peristalic patch Pump

11.95.2 Constructor & Destructor Documentation

11.95.2.1 CPPCFunctionNet() [1/2]

```
CPPCFunctionNet (
    CMcsUsbNet^ mcsusb,
    CMcsUsbFunctionPointerContainer^ pPPCFunctionPointerContainer )
```

Initializes a new instance of the CPPCFunctionNet class.

11.95.2.2 CPPCFunctionNet() [2/2] CPPCFunctionNet (CMcsUsbNet^ *mcsusb*)

11.95.2.3 ~CPPCFunctionNet() virtual ~CPPCFunctionNet ( ) [virtual]

11.95.2.4 "!CPPCFunctionNet() !CPPCFunctionNet ( )

11.95.3 Member Function Documentation

11.95.3.1 FirePressurePulse() void FirePressurePulse (int32_t *duration*, int32_t *nextpressure*)

Fire a pressure pulse from the reservoir

Parameters

duration	The pulse duration (valves open)
nextpressure	The next pressure

11.95.3.2 GetAnalogVoltage() uint16_t GetAnalogVoltage (uint16_t *channel*)

Reads the current analog voltage

Parameters

channel	The Channel Number

Returns

The Analog Voltage

11.95.3.3 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

channel	The Channel Number
min_voltage	The voltage that should be seen as the minimum voltage
max_voltage	The voltage that should be seen as the maximum voltage

11.95.3.4 GetDigitalIn() `uint16_t GetDigitalIn (`

```
uint16_t channel )
```

Reads the digital input state

Parameters

channel	The Channel Number

Returns

The Digital State

11.95.3.5 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

channel	The Channel Number
lower_pressure	The lower border of the pressure range
upper_pressure	The upper border of the pressure range

11.95.3.6 GetPumpModeType() `PP_Pump_Mode_Type_EnumNet GetPumpModeType (`

```
uint16_t channel )
```

Reads the Pump Mode Type.

Parameters

channel	The Channel Number

Returns

The Pump Mode Type.

11.95.3.7 GetPumpSpeedUnit() `int GetPumpSpeedUnit (`

`uint16_t` *channel* `)`

Reads the Pump Speed Unit

Parameters

channel	The Channel Number

Returns

The Speed Unit

11.95.3.8 GetSupplyVoltage() `uint16_t GetSupplyVoltage ( )`

Reads the current supply voltage in mV

Returns

The supply voltage

11.95.3.9 GetValveActive() `int GetValveActive (`

`uint16_t` *valve* `)`

Gets the valve active/inactive state

Parameters

valve	The valve number

Returns

The valve state

11.95.3.10 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

task	The task state
wait	The wait state

11.95.3.11 LoadPressure() `void LoadPressure ( int32_t pressure, uint32_t options )`

Loads the reservoir with a pressure

Parameters

pressure	The pressure
options	The options: end with 0=regulate on patch 1=regulate on reservoir

11.95.3.12 MeasureReservoir() `int32_t MeasureReservoir ( )`

Measures the reservoir pressure

Returns

The pressure

11.95.3.13 SetAnalogVoltageRange() `void SetAnalogVoltageRange ( uint16_t channel, uint16_t min_voltage, uint16_t max_voltage )`

Writes the Analog Input Voltage Range

Parameters

channel	The Channel Number
min_voltage	The voltage that should be seen as the minimum voltage
max_voltage	The voltage that should be seen as the maximum voltage

11.95.3.14 SetPressureOffset() `void SetPressureOffset ( )`

Sets the pressure offset

11.95.3.15 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

channel	The Channel Number
lower_pressure	The lower border of the pressure range
upper_pressure	The upper border of the pressure range

11.95.3.16 SetPumpModeType() `void SetPumpModeType (`

```
uint16_t channel,
PP_Pump_Mode_Type_EnumNet PumpMode )
```

Writes the config string from the device.

Parameters

channel	The Channel Number
PumpMode	The Pump Mode Type.

11.95.3.17 SetPumpSpeedUnit() `void SetPumpSpeedUnit (`

```
uint16_t channel,
int SpeedUnit )
```

Writes the Pump Speed Unit

Parameters

channel	The Channel Number
SpeedUnit	The Speed Unit

11.95.3.18 SetValveActive() `void SetValveActive (`

```
uint16_t valve,
int valveActive )
```

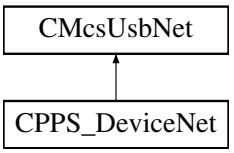
Sets the valve active/inactive state

Parameters

valve	The valve number
valveActive	The valve state

11.96 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

11.96.1 Constructor & Destructor Documentation

11.96.1.1 CPPS_DeviceNet() CPPS_DeviceNet (void)

11.96.2 Property Documentation

11.96.2.1 McsBus CMcsBusNet^ McsBus [get]

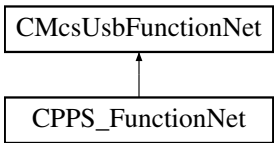
11.96.2.2 McsBus_MotorControl CMcsBus_MotorControlNet^ McsBus_MotorControl [get]

11.96.2.3 McsBus_Sensor CMcsBus_SensorNet^ McsBus_Sensor [get]

11.96.2.4 PPS_Function `CPPS_FunctionNet^ PPS_Function [get]`

11.97 CPPS_FunctionNet Class Reference

Inheritance diagram for CPPS_FunctionNet:



Public Member Functions

- CPPS_FunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ cPPS_FunctionPointer↩ Container)
- 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

11.97.1 Constructor & Destructor Documentation

11.97.1.1 CPPS_FunctionNet() [1/2] CPPS_FunctionNet (

```
CMcsUsbNet^ mcsusb,
CMcsUsbFunctionPointerContainer^ cPPS_FunctionPointerContainer )
```

11.97.1.2 CPPS_FunctionNet() [2/2] CPPS_FunctionNet (

```
CMcsUsbNet^ mcsusb )
```

11.97.2 Member Function Documentation

11.97.2.1 GetAnalogVoltage() unsigned short GetAnalogVoltage (

```
unsigned short index )
```

11.97.2.2 GetAnalogVoltages() void GetAnalogVoltages (

```
unsigned short index,
[System::Runtime::InteropServices::Out] unsigned short% minvoltage,
[System::Runtime::InteropServices::Out] unsigned short% maxvoltage )
```

11.97.2.3 GetBubbleState() unsigned short GetBubbleState ( )

11.97.2.4 GetDigitalIn() unsigned short GetDigitalIn (

```
unsigned short index )
```

11.97.2.5 GetPumpCouple() unsigned int GetPumpCouple ( )

11.97.2.6 GetPumpEnableSpeedRatio() unsigned int GetPumpEnableSpeedRatio ( )

11.97.2.7 GetPumpFastOnOff() unsigned int GetPumpFastOnOff (

```
unsigned short index )
```

11.97.2.8 GetPumpFastSpeed()

```
short GetPumpFastSpeed (
    unsigned short index )
```

11.97.2.9 GetPumpFunctionSpeeds()

```
void GetPumpFunctionSpeeds (
    unsigned short index,
    [System::Runtime::InteropServices::Out] short% offspeed,
    [System::Runtime::InteropServices::Out] short% onspeed )
```

11.97.2.10 GetPumpManualOnOff()

```
unsigned int GetPumpManualOnOff (
    unsigned short index )
```

11.97.2.11 GetPumpMaxSpeed()

```
unsigned short GetPumpMaxSpeed (
    unsigned short index )
```

11.97.2.12 GetPumpModeType()

```
PP_Pump_Mode_Type_EnumNet GetPumpModeType (
    unsigned short index )
```

11.97.2.13 GetPumpSpeedRatio()

```
int GetPumpSpeedRatio ( )
```

11.97.2.14 GetPumpSpeedUnit()

```
int GetPumpSpeedUnit (
    unsigned short index )
```

11.97.2.15 GetSupplyVoltage()

```
unsigned short GetSupplyVoltage ( )
```

11.97.2.16 GetUseBubble()

```
unsigned int GetUseBubble (
    unsigned short index )
```

11.97.2.17 SetAnalogVoltages()

```
void SetAnalogVoltages (
    unsigned short index,
    unsigned short minvoltage,
    unsigned short maxvoltage )
```

11.97.2.18 SetPumpCouple()

```
void SetPumpCouple (
    unsigned int i )
```

11.97.2.19 SetPumpEnableSpeedRatio()

```
void SetPumpEnableSpeedRatio (
    unsigned int enable )
```

11.97.2.20 SetPumpFastOnOff()

```
void SetPumpFastOnOff (
    unsigned short index,
    unsigned int onoff )
```

11.97.2.21 SetPumpFastSpeed()

```
void SetPumpFastSpeed (
    unsigned short index,
    short fastspeed )
```

11.97.2.22 SetPumpFunctionSpeeds()

```
void SetPumpFunctionSpeeds (
    unsigned short index,
    short offspeed,
    short onspeed )
```

11.97.2.23 SetPumpManualOnOff()

```
void SetPumpManualOnOff (
    unsigned short index,
    unsigned int onoff )
```

11.97.2.24 SetPumpMaxSpeed()

```
void SetPumpMaxSpeed (
    unsigned short index,
    unsigned short maxspeed )
```

11.97.2.25 SetPumpModeType() void SetPumpModeType (

```
unsigned short index,
PP_Pump_Mode_Type_EnumNet type )
```

11.97.2.26 SetPumpSpeedRatio() void SetPumpSpeedRatio (

```
int ratio )
```

11.97.2.27 SetPumpSpeedUnit() void SetPumpSpeedUnit (

```
unsigned short index,
int speedunit )
```

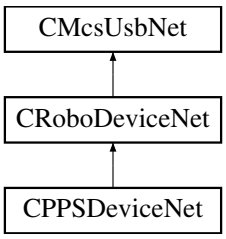
11.97.2.28 SetUseBubble() void SetUseBubble (

```
unsigned short index,
unsigned int usebubble )
```

11.98 CPPSDeviceNet Class Reference

CPPS4plus1DeviceNet is the to control the MCS HLA device

Inheritance diagram for CPPSDeviceNet:



Public Member Functions

- CPPSDeviceNet (void)

Additional Inherited Members

11.98.1 Detailed Description

CPPS4plus1DeviceNet is the to control the MCS HLA device

11.98.2 Constructor & Destructor Documentation

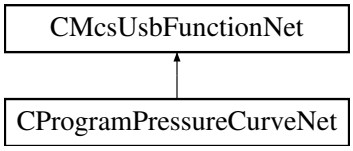
11.98.2.1 CPPSDeviceNet() `CPPSDeviceNet (`

`void )`

11.99 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

11.99.1 Detailed Description

CProgramPressureCurveNet is the class to program pressure curves

11.99.2 Constructor & Destructor Documentation

11.99.2.1 CProgramPressureCurveNet() `CProgramPressureCurveNet (`

`CMcsUsbNet^ mcsusb )`

Initializes a new instance of the CPulseGeneratorFunctionNet class.

11.99.2.2 ~CProgramPressureCurveNet() virtual ~CProgramPressureCurveNet (void) [virtual]

11.99.2.3 "!CProgramPressureCurveNet() !CProgramPressureCurveNet (void)

11.99.3 Member Function Documentation

11.99.3.1 GetRepeats() unsigned int GetRepeats (unsigned char *busnumber*, unsigned char *busaddress*, int32_t *channel*)

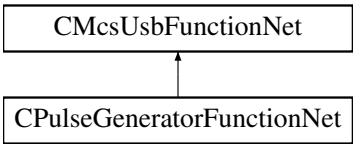
11.99.3.2 Program() void Program (unsigned char *busnumber*, unsigned char *busaddress*, int32_t *channel*, array< int32_t >^ *pressures*, array< int32_t >^ *steps*, array< int16_t >^ *durations*)

11.99.3.3 SetRepeats() void SetRepeats (unsigned char *busnumber*, unsigned char *busaddress*, int32_t *channel*, uint32_t *repeats*)

11.100 CPulseGeneratorFunctionNet Class Reference

CPulseGeneratorFunctionNet is the class to control the pulse generator for video tracking

Inheritance diagram for CPulseGeneratorFunctionNet:



Public Member Functions

- CPulseGeneratorFunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ pPulse↩ GeneratorFunctionPointerContainer)

  Initializes a new instance of the CPulseGeneratorFunctionNet class.
- CPulseGeneratorFunctionNet (CMcsUsbNet^ mcsusb)
- virtual ~CPulseGeneratorFunctionNet ()
- !CPulseGeneratorFunctionNet ()
- int32_t GetPeriod (int32_t generator_number)

  Reads the generator period
- void SetPeriod (int32_t generator_number, int32_t period_in_samples)

  Writes the generator period
- int32_t GetPulseLength (int32_t generator_number)

  Reads the generator pulse length
- void SetPulseLength (int32_t generator_number, int32_t pulselength_in_10us)

  Writes the generator pulse length
- void GetModeSelect (int32_t generator_number, [System::Runtime::InteropServices::Out]PulseGenerator_Mode_EnumNet% mode, [System::Runtime::InteropServices::Out]int32_t% digitalchannel)

  Reads the generator mode
- void SetModeSelect (int32_t generator_number, PulseGenerator_Mode_EnumNet mode, int32_t digitalchannel)

  Writes the generator mode

Additional Inherited Members

11.100.1 Detailed Description

CPulseGeneratorFunctionNet is the class to control the pulse generator for video tracking

11.100.2 Constructor & Destructor Documentation

11.100.2.1 CPulseGeneratorFunctionNet() [1/2]

```
CPulseGeneratorFunctionNet (
    CMcsUsbNet^ mcsusb,
    CMcsUsbFunctionPointerContainer^ pPulseGeneratorFunctionPointerContainer )
```

Initializes a new instance of the CPulseGeneratorFunctionNet class.

11.100.2.2 CPulseGeneratorFunctionNet() [2/2]

```
CPulseGeneratorFunctionNet (
    CMcsUsbNet^ mcsusb )
```

11.100.2.3 ~CPulseGeneratorFunctionNet()

```
virtual ~CPulseGeneratorFunctionNet ( ) [virtual]
```

11.100.2.4 "!CPulseGeneratorFunctionNet() !CPulseGeneratorFunctionNet ( )

11.100.3 Member Function Documentation

11.100.3.1 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

generator_number	The generator number
mode	The generator mode
digitalchannel	The digital in channel used as gate

11.100.3.2 GetPeriod() int32_t GetPeriod (

```
int32_t generator_number )
```

Reads the generator period

Parameters

generator_number	The generator number

Returns

The period

11.100.3.3 GetPulseLength() int32_t GetPulseLength (

```
int32_t generator_number )
```

Reads the generator pulse length

Parameters

generator_number	The generator number

Returns

The pulse length

11.100.3.4 SetModeSelect() `void SetModeSelect (`

```
int32_t generator_number,
PulseGenerator_Mode_EnumNet mode,
int32_t digitalchannel )
```

Writes the generator mode

Parameters

generator_number	The generator number
mode	The generator mode
digitalchannel	The digital in channel used as gate

11.100.3.5 SetPeriod() `void SetPeriod (`

```
int32_t generator_number,
int32_t period_in_samples )
```

Writes the generator period

Parameters

generator_number	The generator number
period_in_samples	The period

11.100.3.6 SetPulseLength() `void SetPulseLength (`

```
int32_t generator_number,
int32_t pulselength_in_10us )
```

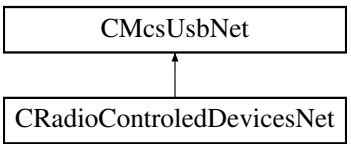
Writes the generator pulse length

Parameters

generator_number	The generator number
pulselength_in_10us	The pulse length

11.101 CRadioControledDevicesNet Class Reference

Inheritance diagram for CRadioControledDevicesNet:



Public Member Functions

- CRadioControledDevicesNet (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

- CRadioControledDevicesNet (CRadioControledDevices *pRadioControled)

Additional Inherited Members

11.101.1 Constructor & Destructor Documentation

11.101.1.1 CRadioControledDevicesNet() [1/2] CRadioControledDevicesNet (void)

11.101.1.2 CRadioControledDevicesNet() [2/2] CRadioControledDevicesNet (CRadioControledDevices * *pRadioControled*) [protected]

11.101.2 Member Function Documentation

11.101.2.1 ConnectDevice() void ConnectDevice (unsigned short *sn*)

11.101.2.2 DisConnectDevice() void DisConnectDevice ( )

11.101.2.3 GetDeviceNames() `array<unsigned short> ^ GetDeviceNames ( )`

11.101.2.4 GetFrequency() `unsigned short GetFrequency ( )`

11.101.2.5 HasRadioControl() `bool HasRadioControl ( )`

11.101.2.6 SetFrequency() `void SetFrequency ( unsigned short frequency )`

11.101.2.7 StillConnected() `bool StillConnected ( )`

11.102 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

11.102.1 Constructor & Destructor Documentation

11.102.1.1 CRegionOfInterestRect() `CRegionOfInterestRect ( int left, int top, int right, int bottom )`

11.102.2 Member Function Documentation

11.102.2.1 DeepCopy() `CRegionOfInterestRect ^ DeepCopy ( )`

11.102.3 Member Data Documentation

11.102.3.1 m_Bottom `int m_Bottom`

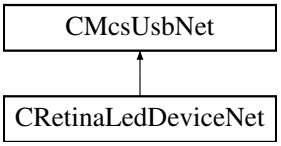
11.102.3.2 m_Left `int m_Left`

11.102.3.3 m_Right `int m_Right`

11.102.3.4 m_Top `int m_Top`

11.103 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

11.103.1 Constructor & Destructor Documentation

11.103.1.1 CRetinaLedDeviceNet() `CRetinaLedDeviceNet ( )`

11.103.1.2 ~CRetinaLedDeviceNet() `~CRetinaLedDeviceNet ( )`

11.103.2 Member Function Documentation

11.103.2.1 AddLoopEntry() `unsigned int AddLoopEntry (`

```
unsigned short repeats,
unsigned short steps_back )
```

11.103.2.2 AddTableEntry() `unsigned int AddTableEntry (`

```
unsigned long long pattern )
```

11.103.2.3 ClearTable() `unsigned int ClearTable ( )`

11.103.2.4 GetTablepointer() `unsigned int GetTablepointer (`

```
int % position )
```

11.103.2.5 SetLED() `unsigned int SetLED (`

```
unsigned long long pattern )
```

11.103.2.6 SetLumi() `unsigned int SetLumi (`

```
int lumi )
```

11.103.2.7 SetPersistency() unsigned int SetPersistency (unsigned int *persistency*)

11.103.2.8 SetRepeat() unsigned int SetRepeat (int *repeat*)

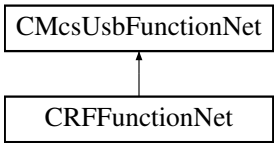
11.103.2.9 SetTablepointer() unsigned int SetTablepointer (int *position*)

11.103.2.10 SetTrigger() unsigned int SetTrigger (int *enable*)

11.104 CRFFunctionNet Class Reference

CRFFunctionNet is the class to control RF devices

Inheritance diagram for CRFFunctionNet:



Public Member Functions

- CRFFunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ pRFFunctionPointer↩ Container)

  Initializes a new instance of the CRFFunctionNet class.
- CRFFunctionNet (CMcsUsbNet^ 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 > ^ GetAvailableDeviceListEx (int list_Length)

  get a list of available devices
- array< uint32_t > ^ GetAvailableDeviceList ()

get a list of available devices

- array< uint32_t > ^ GetAvailableStateListEx (int list_Length)

  get a list of the states of the available devices
- array< uint32_t > ^ GetAvailableStateList ()

  get a list of the states of the available devices
- void Connect (uint32_t sn)

  connect to a RF device, use 0 to disconnect
- uint32_t GetConnectedDevice ()

  get connect RF device, 0 = no device connected
- uint32_t GetState ()

  get connection state
- void SetTestMode (uint32_t mode)

  set test mode
- uint32_t GetTestMode ()

  gets test mode

Additional Inherited Members

11.104.1 Detailed Description

CRFFunctionNet is the class to control RF devices

11.104.2 Constructor & Destructor Documentation

11.104.2.1 CRFFunctionNet() [1/2]

```
CRFFunctionNet (
    CMcsUsbNet^ mcsusb,
    CMcsUsbFunctionPointerContainer^ pRFFunctionPointerContainer )
```

Initializes a new instance of the CRFFunctionNet class.

11.104.2.2 CRFFunctionNet() [2/2]

```
CRFFunctionNet (
    CMcsUsbNet^ mcsusb )
```

11.104.2.3 ~CRFFunctionNet()

```
virtual ~CRFFunctionNet ( ) [virtual]
```

11.104.2.4 "!CRFFunctionNet()

```
!CRFFunctionNet ( )
```

11.104.3 Member Function Documentation

11.104.3.1 Connect()

```
void Connect (
    uint32_t sn )
```

connect to a RF device, use 0 to disconnect

Parameters

sn	the serial number

11.104.3.2 GetAvailableDeviceList() `array<uint32_t> ^ GetAvailableDeviceList ( )`

get a list of available devices

Returns

array of devices

11.104.3.3 GetAvailableDeviceListEx() `array<uint32_t> ^ GetAvailableDeviceListEx ( int list_Length )`

get a list of available devices

Parameters

list_Length	The maximal length of list.

Returns

array of devices

11.104.3.4 GetAvailableStateList() `array<uint32_t> ^ GetAvailableStateList ( )`

get a list of the states of the available devices

Returns

array of states

11.104.3.5 GetAvailableStateListEx() `array<uint32_t> ^ GetAvailableStateListEx ( int list_Length )`

get a list of the states of the available devices

Parameters

list_Length	The maximal length of list.

Returns

array of states

11.104.3.6 GetBaseFrequency() `uint32_t GetBaseFrequency ( CFirmwareDestinationNet destination )`

gets the base advertise frequency

Parameters

destination	the destination to query

Returns

the frequency

11.104.3.7 GetConnectedDevice() `uint32_t GetConnectedDevice ( )`

get connect RF device, 0 = no device connected

Returns

the serial number

11.104.3.8 GetState() `uint32_t GetState ( )`

get connection state

Returns

the state

11.104.3.9 GetTestMode() uint32_t GetTestMode ( )

gets test mode

Returns

the mode

11.104.3.10 GetWorkingFrequency() uint32_t GetWorkingFrequency ( )

gets the working frequency

Returns

the frequency

11.104.3.11 SetBaseFrequency() void SetBaseFrequency (CFirmwareDestinationNet *destination*, uint32_t *frequency*)

sets the base advertise frequency

Parameters

destination	the destination to set
frequency	the frequency

11.104.3.12 SetTestMode() void SetTestMode (uint32_t *mode*)

set test mode

Parameters

mode	the mode

11.104.3.13 SetWorkingFrequency() void SetWorkingFrequency (uint32_t *frequency*)

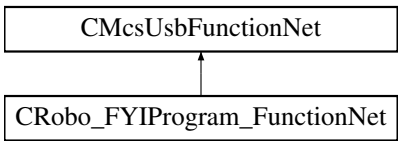
sets the working frequency

Parameters

frequency	the frequency

11.105 CRobo_FYIProgram_FunctionNet Class Reference

Inheritance diagram for CRobo_FYIProgram_FunctionNet:



Public Member Functions

- CRobo_FYIProgram_FunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ robo_↩ FYIProgram_FunctionPointerContainer)
- CRobo_FYIProgram_FunctionNet (CMcsUsbNet^ 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

11.105.1 Constructor & Destructor Documentation

11.105.1.1 CRobo_FYIProgram_FunctionNet() [1/2]

```
CRobo_FYIProgram_FunctionNet (
    CMcsUsbNet^ mcsusb,
    CMcsUsbFunctionPointerContainer^ robo_FYIProgram_FunctionPointerContainer )
```

11.105.1.2 CRobo_FYIProgram_FunctionNet() [2/2]

```
CRobo_FYIProgram_FunctionNet (
    CMcsUsbNet^ mcsusb )
```

11.105.2 Member Function Documentation

11.105.2.1 GetLength() int GetLength (unsigned char *index*)

11.105.2.2 GetState() int GetState ( )

11.105.2.3 GetValve1() unsigned int GetValve1 (unsigned char *index*)

11.105.2.4 GetValve2() unsigned int GetValve2 (unsigned char *index*)

11.105.2.5 SetLength() void SetLength (unsigned char *index*, int *length*)

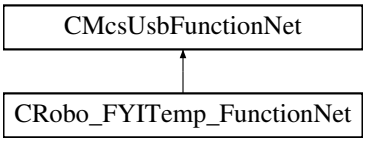
11.105.2.6 SetValve1() void SetValve1 (unsigned char *index*, unsigned int *valve1*)

11.105.2.7 SetValve2() void SetValve2 (unsigned char *index*, unsigned int *valve2*)

11.105.2.8 Start() void Start ( )

11.106 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

11.106.1 Constructor & Destructor Documentation

11.106.1.1 CRobo_FYITemp_FunctionNet() `CRobo_FYITemp_FunctionNet (`

`CMcsUsbNet^ mcsusb )`

11.106.2 Member Function Documentation

11.106.2.1 GetICoeff() `int GetICoeff (`

`unsigned char index )`

11.106.2.2 GetMaxPower() `int GetMaxPower (`

`unsigned char index )`

11.106.2.3 GetPCoeff() `int GetPCoeff (`

`unsigned char index )`

11.106.2.4 GetRegulatorOnOff() `int GetRegulatorOnOff (`

`unsigned char index )`

11.106.2.5 GetSollTemp()

```
int GetSollTemp (
    unsigned char index )
```

11.106.2.6 SetICoeff()

```
void SetICoeff (
    unsigned char index,
    int icoeff )
```

11.106.2.7 SetMaxPower()

```
void SetMaxPower (
    unsigned char index,
    int power )
```

11.106.2.8 SetPCoeff()

```
void SetPCoeff (
    unsigned char index,
    int pcoeff )
```

11.106.2.9 SetRegulatorOnOff()

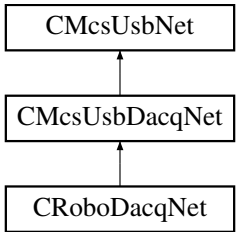
```
void SetRegulatorOnOff (
    unsigned char index,
    int onoff )
```

11.106.2.10 SetSollTemp()

```
void SetSollTemp (
    unsigned char index,
    int temp )
```

11.107 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 GetUVOffset ()
- int GetUCOffset ()
- int GetICOffset ()
- int GetCrossTalkOffset ()
- int GetXGain ()
- int GetCrossTalkOptimum ()
- unsigned int GetRecordingNumber ()
- int GetResistanceC ()
- int GetResistanceV ()
- int GetCapacityC ()
- int GetCapacityV ()
- int GetCapacityX ()
- int GetUV ()
- int GetUC ()
- int GetIC ()
- int GetNUV_MS ()
- int GetNUC_MS ()
- int GetNIC_MS ()
- void SetAllDigout (uint32_t value)
- uint32_t GetAllDigout ()
- void SetCommand (unsigned char command, int value)
- int GetCommand (unsigned char command)
- void SetDigout (uint16_t index, bool enable)
- bool GetDigout (uint16_t index)
- void TableDefBegin ()
- void TableDefEnd ()
- void Table_Wait (unsigned int tableWait)
- void SetDownsampleFactor (int index, int downsample_factor)
- void SetFilterCoeffs (int index, array< int >^ coeffs)
- void SetNoFilterCoeffs (int index)
- int GetDownsampleFactor (int index)
- array< int > ^ GetFilterCoeffs (int index)
- void Emu_SetElectrodeResists (int emuElectrodeResist)
- void Emu_SetCellResists (int emuCellResist)
- void Emu_SetCellCapacity (int emuCellCapacity)
- void Emu_SetCellPotential (int emuCellPotential)
- void Emu_SetNoise (int emuNoise)
- int Emu_GetElectrodeResists ()
- int Emu_GetCellResists ()
- int Emu_GetCellCapacity ()
- int Emu_GetCellPotential ()
- int Emu_GetNoise ()
- void SetDisplayText (int index, String^ displayText)
- void SetScreen (int screen)

- void UpdateDisplay ()
- String ^ GetDisplayText (int index)
- int GetScreen ()
- int GetUpdateDisplay ()

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_ContinousDacq = 0
- static const unsigned int VirtualDevice_TableRun = 1

Additional Inherited Members

11.107.1 Constructor & Destructor Documentation

11.107.1.1 CRoboDacqNet() [1/2] CRoboDacqNet (void)

11.107.1.2 CRoboDacqNet() [2/2] CRoboDacqNet (CRoboDeviceNet^ *robodevice*)

11.107.2 Member Function Documentation

11.107.2.1 CancelTableLoop() void CancelTableLoop ( )

11.107.2.2 CancelTableLoopAndStopTable() void CancelTableLoopAndStopTable ( )

11.107.2.3 ClampAmpRestart() void ClampAmpRestart ( )

11.107.2.4 DoRamp()

```
void DoRamp (
    int startValue,
    int endValue,
    int duration,
    int mode )
```

11.107.2.5 Emu_GetCellCapacity()

```
int Emu_GetCellCapacity ( )
```

11.107.2.6 Emu_GetCellPotential()

```
int Emu_GetCellPotential ( )
```

11.107.2.7 Emu_GetCellResists()

```
int Emu_GetCellResists ( )
```

11.107.2.8 Emu_GetElectrodeResists()

```
int Emu_GetElectrodeResists ( )
```

11.107.2.9 Emu_GetNoise()

```
int Emu_GetNoise ( )
```

11.107.2.10 Emu_SetCellCapacity()

```
void Emu_SetCellCapacity (
    int emuCellCapacity )
```

11.107.2.11 Emu_SetCellPotential()

```
void Emu_SetCellPotential (
    int emuCellPotential )
```

11.107.2.12 Emu_SetCellResists()

```
void Emu_SetCellResists (
    int emuCellResist )
```

11.107.2.13 Emu_SetElectrodeResists()

```
void Emu_SetElectrodeResists (
    int emuElectrodeResist )
```

11.107.2.14 Emu_SetNoise() void Emu_SetNoise (int *emuNoise*)

11.107.2.15 GetAllDigout() uint32_t GetAllDigout ( )

11.107.2.16 GetCapacityC() int GetCapacityC ( )

11.107.2.17 GetCapacityV() int GetCapacityV ( )

11.107.2.18 GetCapacityX() int GetCapacityX ( )

11.107.2.19 GetClampAmpSerialNumber() unsigned int GetClampAmpSerialNumber ( )

11.107.2.20 GetCommand() int GetCommand (unsigned char *command*)

11.107.2.21 GetConfigurationBit() bool GetConfigurationBit (unsigned short *bit*)

11.107.2.22 GetConfigurationBitAxc() bool GetConfigurationBitAxc ( )

11.107.2.23 GetConfigurationBitBlu_Led() bool GetConfigurationBitBlu_Led ( )

11.107.2.24 GetConfigurationBitBlu_LedToggleFast() bool GetConfigurationBitBlu_LedToggleFast ( )

11.107.2.25 GetConfigurationBitBlu_LedToggleSlow() bool GetConfigurationBitBlu_LedToggleSlow ( )

11.107.2.26 GetConfigurationBitCC_Gen() bool GetConfigurationBitCC_Gen ( )

11.107.2.27 GetConfigurationBitCV_Gen() bool GetConfigurationBitCV_Gen ( )

11.107.2.28 GetConfigurationBitRC_Gen() bool GetConfigurationBitRC_Gen ( )

11.107.2.29 GetConfigurationBitRed_Led() bool GetConfigurationBitRed_Led ( )

11.107.2.30 GetConfigurationBitRed_LedSaturation() bool GetConfigurationBitRed_LedSaturation ( )

11.107.2.31 GetConfigurationBitRed_LedToggleFast() bool GetConfigurationBitRed_LedToggleFast ( )

11.107.2.32 GetConfigurationBitRed_LedToggleSlow() bool GetConfigurationBitRed_LedToggleSlow ( )

11.107.2.33 GetConfigurationBitRelais() bool GetConfigurationBitRelais ( )

11.107.2.34 GetConfigurationBitRV_Gen() bool GetConfigurationBitRV_Gen ( )

11.107.2.35 GetConfigurationBits() unsigned int GetConfigurationBits ( )

11.107.2.36 GetConfigurationBitStream() bool GetConfigurationBitStream ( )

11.107.2.37 GetConfigurationBitSupply() bool GetConfigurationBitSupply ( )

11.107.2.38 GetCrossTalkOffset() int GetCrossTalkOffset ( )

11.107.2.39 GetCrossTalkOptimum() int GetCrossTalkOptimum ( )

11.107.2.40 GetDigout() bool GetDigout (uint16_t *index*)

11.107.2.41 GetDisplayText() String ^ GetDisplayText (int *index*)

11.107.2.42 GetDownsampleFactor() int GetDownsampleFactor (int *index*)

11.107.2.43 GetFilter() int GetFilter ( )

11.107.2.44 GetFilterCoeffs() array<int> ^ GetFilterCoeffs (int *index*)

11.107.2.45 GetIC() int GetIC ( )

11.107.2.46 GetIClamp() int GetIClamp ( )

11.107.2.47 GetICOffset() int GetICOffset ( )

11.107.2.48 GetIGain() int GetIGain ( )

11.107.2.49 GetNIC_MS() int GetNIC_MS ( )

11.107.2.50 GetNUC_MS() int GetNUC_MS ( )

11.107.2.51 GetNUV_MS() int GetNUV_MS ( )

11.107.2.52 GetPGain() int GetPGain ( )

11.107.2.53 GetRecordingNumber() unsigned int GetRecordingNumber ( )

11.107.2.54 GetResistanceC() int GetResistanceC ( )

11.107.2.55 GetResistanceV() int GetResistanceV ( )

11.107.2.56 GetScreen() int GetScreen ( )

11.107.2.57 GetSimulation() unsigned int GetSimulation ( )

11.107.2.58 GetUC() int GetUC ( )

11.107.2.59 GetUClamp() int GetUClamp ( )

11.107.2.60 GetUCOffset() int GetUCOffset ( )

11.107.2.61 GetUpdateDisplay() int GetUpdateDisplay ( )

11.107.2.62 GetUV() int GetUV ( )

11.107.2.63 GetUVOffset() int GetUVOffset ( )

11.107.2.64 GetXGain() int GetXGain ( )

11.107.2.65 RunTable() [1/2] void RunTable ( )

11.107.2.66 RunTable() [2/2] void RunTable (int *timeout*)

11.107.2.67 SetAllDigout() void SetAllDigout (uint32_t *value*)

11.107.2.68 SetCommand() void SetCommand (unsigned char *command*, int *value*)

11.107.2.69 SetConfigurationBit()

```
void SetConfigurationBit (
    unsigned short bit,
    bool value )
```

11.107.2.70 SetConfigurationBitAxc()

```
void SetConfigurationBitAxc (
    bool value )
```

11.107.2.71 SetConfigurationBitBlu_Led()

```
void SetConfigurationBitBlu_Led (
    bool value )
```

11.107.2.72 SetConfigurationBitBlu_LedToggleFast()

```
void SetConfigurationBitBlu_LedToggleFast (
    bool value )
```

11.107.2.73 SetConfigurationBitBlu_LedToggleSlow()

```
void SetConfigurationBitBlu_LedToggleSlow (
    bool value )
```

11.107.2.74 SetConfigurationBitCC_Gen()

```
void SetConfigurationBitCC_Gen (
    bool value )
```

11.107.2.75 SetConfigurationBitCV_Gen()

```
void SetConfigurationBitCV_Gen (
    bool value )
```

11.107.2.76 SetConfigurationBitRC_Gen()

```
void SetConfigurationBitRC_Gen (
    bool value )
```

11.107.2.77 SetConfigurationBitRed_Led()

```
void SetConfigurationBitRed_Led (
    bool value )
```

11.107.2.78 SetConfigurationBitRed_LedSaturation()

```
void SetConfigurationBitRed_LedSaturation (
    bool value )
```

11.107.2.79 SetConfigurationBitRed_LedToggleFast()

```
void SetConfigurationBitRed_LedToggleFast (
    bool value )
```

11.107.2.80 SetConfigurationBitRed_LedToggleSlow()

```
void SetConfigurationBitRed_LedToggleSlow (
    bool value )
```

11.107.2.81 SetConfigurationBitRelais()

```
void SetConfigurationBitRelais (
    bool value )
```

11.107.2.82 SetConfigurationBitRV_Gen()

```
void SetConfigurationBitRV_Gen (
    bool value )
```

11.107.2.83 SetConfigurationBitStream()

```
void SetConfigurationBitStream (
    bool value )
```

11.107.2.84 SetConfigurationBitSupply()

```
void SetConfigurationBitSupply (
    bool value )
```

11.107.2.85 SetCrossTalkOffset()

```
void SetCrossTalkOffset (
    int CrossTalk )
```

11.107.2.86 SetCrossTalkOptimum()

```
void SetCrossTalkOptimum (
    int cxOptimum )
```

11.107.2.87 SetDigout()

```
void SetDigout (
    uint16_t index,
    bool enable )
```

11.107.2.88 SetDisplayText()

```
void SetDisplayText (
    int index,
    String^ displayText )
```

11.107.2.89 SetDownsampleFactor()

```
void SetDownsampleFactor (
    int index,
    int downsample_factor )
```

11.107.2.90 SetFilter()

```
void SetFilter (
    int filter )
```

11.107.2.91 SetFilterCoeffs()

```
void SetFilterCoeffs (
    int index,
    array< int >^ coeffs )
```

11.107.2.92 SetIClamp()

```
void SetIClamp (
    int iClamp )
```

11.107.2.93 SetICOffset()

```
void SetICOffset (
    int ICOffset )
```

11.107.2.94 SetIGain()

```
void SetIGain (
    int iGain )
```

11.107.2.95 SetNoFilterCoeffs()

```
void SetNoFilterCoeffs (
    int index )
```

11.107.2.96 SetPGain() void SetPGain (

int *pGain*)

11.107.2.97 SetRecordingNumber() void SetRecordingNumber (

unsigned int *recordingNumber*)

11.107.2.98 SetScreen() void SetScreen (

int *screen*)

11.107.2.99 SetSimulation() void SetSimulation (

unsigned int *enable*)

11.107.2.100 SetTriggerMaskValue() void SetTriggerMaskValue (

unsigned int *mask,*
unsigned int *value,*
unsigned int *virtualDevice*)

11.107.2.101 SetUClamp() void SetUClamp (

int *uClamp*)

11.107.2.102 SetUCOffset() void SetUCOffset (

int *UCOffset*)

11.107.2.103 SetUVOffset() void SetUVOffset (

int *UVOffset*)

11.107.2.104 SetXGain() void SetXGain (

int *xGain*)

11.107.2.105 StopTable() [1/2] void StopTable ( )

11.107.2.106 StopTable() [2/2] void StopTable (int *timeout*)

11.107.2.107 Table_Wait() void Table_Wait (unsigned int *tableWait*)

11.107.2.108 TableDefBegin() void TableDefBegin ( )

11.107.2.109 TableDefEnd() void TableDefEnd ( )

11.107.2.110 UpdateDisplay() void UpdateDisplay ( )

11.107.3 Member Data Documentation

11.107.3.1 TriggerMask_Default const unsigned int TriggerMask_Default = 0xFF00 [static]

11.107.3.2 TriggerValue_MoveAbs const unsigned int TriggerValue_MoveAbs = COMMAND_ROBO_↩ MOVEABS [static]

11.107.3.3 TriggerValue_StartQueue const unsigned int TriggerValue_StartQueue = COMMAND_ROBO_↩ QUEUE [static]

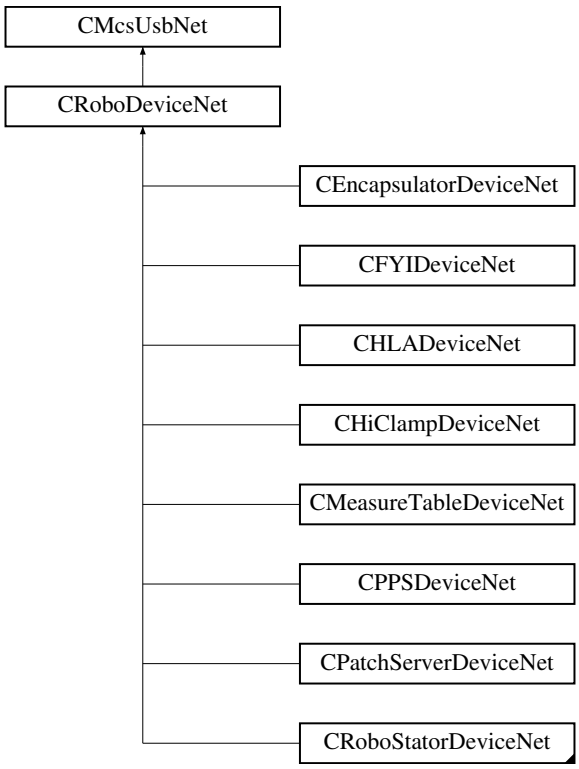
11.107.3.4 VirtualDevice_ContinousDacq const unsigned int VirtualDevice_ContinousDacq = 0 [static]

11.107.3.5 VirtualDevice_TableRun `const unsigned int VirtualDevice_TableRun = 1 [static]`

11.108 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^ McsBus `[get]`
- CMcsBus_MotorControlNet^ McsBus_MotorControl `[get]`
- RoboMainLowLevelCommands^ RoboMainLowLevelCommand `[get]`

Events

- RoboStatusEventDelegate^ RoboStatusEvent

Additional Inherited Members

11.108.1 Detailed Description

CRoboDeviceNet is the base class for all Robo platform based devices

11.108.2 Constructor & Destructor Documentation

11.108.2.1 CRoboDeviceNet() `CRoboDeviceNet (`

`void )`

11.108.2.2 ~CRoboDeviceNet() `~CRoboDeviceNet (`

`void )`

11.108.3 Member Function Documentation

11.108.3.1 CancelPoolLoop() `void CancelPoolLoop ( )`

11.108.3.2 CancelPoolLoopAndStopMovement() void CancelPoolLoopAndStopMovement ( )

11.108.3.3 EnableQueue() void EnableQueue (
bool *enable*)

11.108.3.4 FindReference() [1/2] void FindReference (
unsigned char *busaddress*,
char *axes*)

11.108.3.5 FindReference() [2/2] void FindReference (
unsigned char *busaddress*,
char *axes*,
int *timeout*)

Searches the reference position of the motor

Parameters

busaddress	Address of the McsBus
axes	Bit pattern of axes to drive
timeout	Timeout of maximal waiting for the end of the command (-1 is forever)

11.108.3.6 GetAirpressure() unsigned int GetAirpressure ( )

11.108.3.7 GetAirpressureLimit() void GetAirpressureLimit (
[System::Runtime::InteropServices::Out] unsigned int% *lowerpressure*,
[System::Runtime::InteropServices::Out] unsigned int% *upperpressure*)

11.108.3.8 GetAirValve() unsigned int GetAirValve ( )

11.108.3.9 GetCurrentAirvalve() unsigned int GetCurrentAirvalve ( )

11.108.3.10 GetCurrentAirvalveLimit()

```
void GetCurrentAirvalveLimit (
 [System::Runtime::InteropServices::Out] unsigned int% lowercurrent,
 [System::Runtime::InteropServices::Out] unsigned int% uppercurrent )
```

11.108.3.11 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

busaddress	Address of the McsBus
axes	Bit pattern of axes to drive
x	Current position of first axis if pattern in axes is set
y	Current position of second axis if pattern in axes is set

11.108.3.12 GetErrorAirpressure()

```
unsigned int GetErrorAirpressure ( )
```

11.108.3.13 GetErrorCurrentAirvalve()

```
unsigned int GetErrorCurrentAirvalve ( )
```

11.108.3.14 GetErrorVoltage12V()

```
unsigned int GetErrorVoltage12V ( )
```

11.108.3.15 GetErrorVoltage5V()

```
unsigned int GetErrorVoltage5V ( )
```

11.108.3.16 GetErrorVoltageAirvalve()

```
unsigned int GetErrorVoltageAirvalve ( )
```

11.108.3.17 GetErrorVoltageRs485A()

```
unsigned int GetErrorVoltageRs485A ( )
```

11.108.3.18 GetErrorVoltageRs485B() unsigned int GetErrorVoltageRs485B ( )

11.108.3.19 GetErrorVoltageValves() unsigned int GetErrorVoltageValves ( )

11.108.3.20 GetInMovement() bool GetInMovement ( )

Low level command, gets the internal state "In Movement"

11.108.3.21 GetMinPressure() int GetMinPressure ( )

11.108.3.22 GetMovementError() uint32_t GetMovementError ( )

Low level command, gets the error of the last movement end

11.108.3.23 GetVoltage12V() unsigned int GetVoltage12V ( )

11.108.3.24 GetVoltage12VLimit() void GetVoltage12VLimit (

```
[System::Runtime::InteropServices::Out] unsigned int% lowervoltage,
[System::Runtime::InteropServices::Out] unsigned int% uppervoltage )
```

11.108.3.25 GetVoltage5V() unsigned int GetVoltage5V ( )

11.108.3.26 GetVoltage5VLimit() void GetVoltage5VLimit (

```
[System::Runtime::InteropServices::Out] unsigned int% lowervoltage,
[System::Runtime::InteropServices::Out] unsigned int% uppervoltage )
```

11.108.3.27 GetVoltageAirvalve() unsigned int GetVoltageAirvalve ( )

11.108.3.28 GetVoltageAirvalveLimit()

```
void GetVoltageAirvalveLimit (
 [System::Runtime::InteropServices::Out] unsigned int% lowervoltage,
 [System::Runtime::InteropServices::Out] unsigned int% uppervoltage )
```

11.108.3.29 GetVoltageRs485A()

```
unsigned int GetVoltageRs485A ( )
```

11.108.3.30 GetVoltageRs485ALimit()

```
void GetVoltageRs485ALimit (
 [System::Runtime::InteropServices::Out] unsigned int% lowervoltage,
 [System::Runtime::InteropServices::Out] unsigned int% uppervoltage )
```

11.108.3.31 GetVoltageRs485B()

```
unsigned int GetVoltageRs485B ( )
```

11.108.3.32 GetVoltageRs485BLimit()

```
void GetVoltageRs485BLimit (
 [System::Runtime::InteropServices::Out] unsigned int% lowervoltage,
 [System::Runtime::InteropServices::Out] unsigned int% uppervoltage )
```

11.108.3.33 GetVoltageValves()

```
unsigned int GetVoltageValves ( )
```

11.108.3.34 GetVoltageValvesLimit()

```
void GetVoltageValvesLimit (
 [System::Runtime::InteropServices::Out] unsigned int% lowervoltage,
 [System::Runtime::InteropServices::Out] unsigned int% uppervoltage )
```

11.108.3.35 IsQueueEnabled()

```
bool IsQueueEnabled ( )
```

11.108.3.36 IsQueueStarted()

```
bool IsQueueStarted ( )
```

11.108.3.37 MoveAbs() [1/4]

```
void MoveAbs (
 unsigned char busaddress,
 char axes,
 array< int >^ pos )
```

Moves the motor to the new absolute position

Parameters

busaddress	Address of the McsBus
axes	Bit pattern of axes to drive
pos	Positions of the axis 0 to 3, if pattern in axes is set

11.108.3.38 MoveAbs() [2/4]

```
void MoveAbs (
    unsigned char busaddress,
    char axes,
    array< int >^ pos,
    int timeout )
```

Moves the motor to the new absolute position

Parameters

busaddress	Address of the McsBus
axes	Bit pattern of axes to drive
pos	Positions of the axis 0 to 3, if pattern in axes is set
timeout	Timeout of maximal waiting for the end of the command (-1 is forever)

11.108.3.39 MoveAbs() [3/4]

```
void MoveAbs (
    unsigned char busaddress,
    char axes,
    int x,
    int y )
```

Moves the motor to the new absolute position

Parameters

busaddress	Address of the McsBus
axes	Bit pattern of axes to drive
x	Position of first axis, if pattern in axes is set
y	Position of second axis if pattern in axes is set

11.108.3.40 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

busaddress	Address of the McsBus
axes	Bit pattern of axes to drive
x	Position of first axis, if pattern in axes is set
y	Position of second axis if pattern in axes is set
timeout	Timeout of maximal waiting for the end of the command (-1 is forever)

11.108.3.41 NullCommand()

```
void NullCommand (
    unsigned int marker )
```

11.108.3.42 SetAirpressureLimit()

```
void SetAirpressureLimit (
    unsigned int lowerpressure,
    unsigned int upperpressure )
```

11.108.3.43 SetAirValve()

```
void SetAirValve (
    unsigned int onoff )
```

11.108.3.44 SetCurrentAirvalveLimit()

```
void SetCurrentAirvalveLimit (
    unsigned int lowercurrent,
    unsigned int uppercurrent )
```

11.108.3.45 SetCurrentAndAir() [1/2]

```
void SetCurrentAndAir (
    unsigned char busaddress,
    char axes,
    unsigned short onoff )
```

11.108.3.46 SetCurrentAndAir() [2/2]

```
void SetCurrentAndAir (
    unsigned char busaddress,
    char axes,
    unsigned short onoff,
    int timeout )
```

11.108.3.47 SetInMovement() void SetInMovement ( )

Low level command, sets the internal state to "In Movement"

11.108.3.48 SetMinPressure() void SetMinPressure (int *pressure*)

11.108.3.49 SetVoltage12VLimit() void SetVoltage12VLimit (unsigned int *lowervoltage*, unsigned int *uppervoltage*)

11.108.3.50 SetVoltage5VLimit() void SetVoltage5VLimit (unsigned int *lowervoltage*, unsigned int *uppervoltage*)

11.108.3.51 SetVoltageAirvalveLimit() void SetVoltageAirvalveLimit (unsigned int *lowervoltage*, unsigned int *uppervoltage*)

11.108.3.52 SetVoltageRs485ALimit() void SetVoltageRs485ALimit (unsigned int *lowervoltage*, unsigned int *uppervoltage*)

11.108.3.53 SetVoltageRs485BLimit() void SetVoltageRs485BLimit (unsigned int *lowervoltage*, unsigned int *uppervoltage*)

11.108.3.54 SetVoltageValvesLimit() void SetVoltageValvesLimit (unsigned int *lowervoltage*, unsigned int *uppervoltage*)

11.108.3.55 StartQueue() void StartQueue (bool *start*)

11.108.3.56 StopMovement() [1/2] void StopMovement (unsigned char *busaddress*, char *axes*)

11.108.3.57 StopMovement() [2/2] void StopMovement (unsigned char *busaddress*, char *axes*, int *timeout*)

Stops the current movement

Parameters

busaddress	Address of the McsBus
axes	Bit pattern of axes to drive
timeout	Timeout of maximal waiting for the end of the command (-1 is forever)

11.108.3.58 WaitTimer() void WaitTimer (uint32_t *waittime*, int *timeout*)

11.108.4 Member Data Documentation

11.108.4.1 Axes_I const char Axes_I = 2 [static]

Bit pattern for i axis for axes argument

11.108.4.2 Axes_X const char Axes_X = 1 [static]

Bit pattern for x axis for axes argument

11.108.4.3 Axes_Y `const char Axes_Y = 2 [static]`

Bit pattern for y axis for axes argument

11.108.4.4 Axes_Z `const char Axes_Z = 1 [static]`

Bit pattern for z axis for axes argument

11.108.4.5 Axis_I `const byte Axis_I = 1 [static]`

Axis number of i for axis argument

11.108.4.6 Axis_X `const byte Axis_X = 0 [static]`

Axis number of x for axis argument

11.108.4.7 Axis_Y `const byte Axis_Y = 1 [static]`

Axis number of y for axis argument

11.108.4.8 Axis_Z `const byte Axis_Z = 0 [static]`

Axis number of z for axis argument

11.108.4.9 McsBus_XY `const byte McsBus_XY = 1 [static]`

McsBus address for the xy-plane

11.108.4.10 McsBus_ZI `const byte McsBus_ZI = 2 [static]`

McsBus address for the z and i axes

11.108.4.11 RoboError_AnotherMaster

```
const uint32_t RoboError_AnotherMaster = ( (0xA0110000L) | 0x0005 ) [static]
```

11.108.4.12 RoboError_Base

```
const uint32_t RoboError_Base = (0xA0110000L) [static]
```

11.108.4.13 RoboError_CannotEscapeEndSwitch

```
const uint32_t RoboError_CannotEscapeEndSwitch = ( (0xA0110000L) | 0x0009 ) [static]
```

11.108.4.14 RoboError_CommandAlreadyInProgress

```
const uint32_t RoboError_CommandAlreadyInProgress = ( (0xA0110000L) | 0x000A ) [static]
```

11.108.4.15 RoboError_CommandNotPossible

```
const uint32_t RoboError_CommandNotPossible = ( (0xA0110000L) | 0x0014 ) [static]
```

11.108.4.16 RoboError_CommunicationTimeout

```
const uint32_t RoboError_CommunicationTimeout = ( (0xA0110000L) | 0x0004 ) [static]
```

11.108.4.17 RoboError_DacqNotReady

```
const uint32_t RoboError_DacqNotReady = ( (0xA0110000L) | 0x0015 ) [static]
```

11.108.4.18 RoboError_DLLMovementTimeout

```
const uint32_t RoboError_DLLMovementTimeout = ( (0xA0110000L) | 0x1001) [static]
```

11.108.4.19 RoboError_FindReferenceMethod

```
const uint32_t RoboError_FindReferenceMethod = ( (0xA0110000L) | 0x0006 ) [static]
```

11.108.4.20 RoboError_GilsonCommandPending

```
const uint32_t RoboError_GilsonCommandPending = ( (0xA0110000L) | 0x0011 ) [static]
```

11.108.4.21 RoboError_GilsonTimeout

```
const uint32_t RoboError_GilsonTimeout = ( (0xA0110000L) | 0x000F ) [static]
```

11.108.4.22 RoboError_GilsonWrondID

```
const uint32_t RoboError_GilsonWrondID = ( (0xA0110000L) | 0x0010 ) [static]
```

11.108.4.23 RoboError_McsBus_UnknownCommand

```
const uint32_t RoboError_McsBus_Unknown↩
Command = ( (0xA0110000L) | 0x003F) [static]
```

11.108.4.24 RoboError_NoEndSwitch

```
const uint32_t RoboError_NoEndSwitch = ( (0xA0110000L) | 0x0008 ) [static]
```

11.108.4.25 RoboError_NoMoreData

```
const uint32_t RoboError_NoMoreData = ( (0xA0110000L) | 0x0016 ) [static]
```

11.108.4.26 RoboError_NoReference

```
const uint32_t RoboError_NoReference = ( (0xA0110000L) | 0x000B ) [static]
```

11.108.4.27 RoboError_NoSpeedOrAcceleration

```
const uint32_t RoboError_NoSpeedOrAcceleration = ( (0xA0110000L) | 0x0007 ) [static]
```

11.108.4.28 RoboError_OverPressure

```
const uint32_t RoboError_OverPressure = ( (0xA0110000L) | 0x000C ) [static]
```

11.108.4.29 RoboError_ParameterNotAllowed

```
const uint32_t RoboError_ParameterNotAllowed = ( (0xA0110000L) | 0x0012 ) [static]
```

11.108.4.30 RoboError_PeristalticTimeout

```
const uint32_t RoboError_PeristalticTimeout = ( (0x↩
A0110000L) | 0x000E ) [static]
```

11.108.4.31 RoboError_Phase0OutOfRange

```
const uint32_t RoboError_Phase0OutOfRange = ( (0xA0110000L) | 0x000D ) [static]
```

11.108.4.32 RoboError_PollLoopCanceled

```
const uint32_t RoboError_PollLoopCanceled = ( (0xA0110000L) | 0x1002) [static]
```

11.108.4.33 RoboError_PollLoopCanceledAndStopMovement

```
const uint32_t RoboError_PollLoopCanceledAndStopMovement = ( (0xA0110000L) | 0x1003) [static]
```

11.108.4.34 RoboError_Pressure

```
const uint32_t RoboError_Pressure = ( (0xA0110000L) | 0x0002 ) [static]
```

11.108.4.35 RoboError_RangeExceeded

```
const uint32_t RoboError_RangeExceeded = ( (0xA0110000L) | 0x0003 ) [static]
```

11.108.4.36 RoboError_StateChangeNotPossible

```
const uint32_t RoboError_StateChangeNotPossible = ( (0xA0110000L) | 0x0013 ) [static]
```

11.108.4.37 RoboError_Timeout

```
const uint32_t RoboError_Timeout = ( (0xA0110000L) | 0x0001 ) [static]
```

11.108.4.38 RoboError_UnknownCommand

```
const uint32_t RoboError_UnknownCommand = ( (0xA0110000L) ) [static]
```

11.108.5 Property Documentation

11.108.5.1 McsBus

```
CMcsBusNet^ McsBus [get]
```

11.108.5.2 McsBus_MotorControl CMcsBus_MotorControlNet^ McsBus_MotorControl [get]

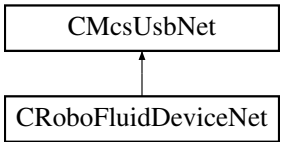
11.108.5.3 RoboMainLowLevelCommand RoboMainLowLevelCommands^ RoboMainLowLevelCommand [get]

11.108.6 Event Documentation

11.108.6.1 RoboStatusEvent RoboStatusEventDelegate^ RoboStatusEvent

11.109 CRoboFluidDeviceNet Class Reference

Inheritance diagram for CRoboFluidDeviceNet:



Public Member Functions

- CRoboFluidDeviceNet (void)
- ~CRoboFluidDeviceNet (void)
- void SetValve (int value)

  Open or Close valves.
- void SetSingleValve (int valve, bool onoff)

  Opens or Closes a valve.
- int GetValve ()

  Query the state of the values.
- bool GetSingleValve (int valve)

  Query the state of a valve.
- void CloseAllValves ()
- void PumpOn (int index, short speed)
- void SetPumpSpeed (int index, short speed)
- void PumpOff (int index)
- short GetPumpSpeed (int index)
- bool IsPumpMotorOn (int index)

Protected Attributes

- CRoboFluidDevice * m_pRoboFluidDevice
- CMcsBus_MotorControlNet ^ m_pMcsBus_MotorControlNet

Properties

- CMcsBus_MotorControlNet^ McsBus_MotorControl [get]

Additional Inherited Members

11.109.1 Constructor & Destructor Documentation

11.109.1.1 CRoboFluidDeviceNet() CRoboFluidDeviceNet (void)

11.109.1.2 ~CRoboFluidDeviceNet() ~CRoboFluidDeviceNet (void)

11.109.2 Member Function Documentation

11.109.2.1 CloseAllValves() void CloseAllValves ( )

11.109.2.2 GetPumpSpeed() short GetPumpSpeed (int *index*)

11.109.2.3 GetSingleValve() bool GetSingleValve (int *valve*)

Query the state of a valve.

Parameters

valve	number of valve /*!

Returns

state of the valve

11.109.2.4 GetValve() int GetValve ( )

Query the state of the values.

Returns

the current state of the valves as a bit pattern.

11.109.2.5 IsPumpMotorOn() bool IsPumpMotorOn (int *index*)

11.109.2.6 PumpOff() void PumpOff (int *index*)

11.109.2.7 PumpOn() void PumpOn (int *index*, short *speed*)

11.109.2.8 SetPumpSpeed() void SetPumpSpeed (int *index*, short *speed*)

11.109.2.9 SetSingleValve() void SetSingleValve (int *valve*, bool *onoff*)

Opens or Closes a valve.

Parameters

valve	number of valve to be changed /∗!
onoff	open or close the valve

11.109.2.10 SetValve() void SetValve (int *value*)

Open or Close valves.

Parameters

value	bit pattern of valves which should be open.

11.109.3 Member Data Documentation

11.109.3.1 m_pMcsBus_MotorControlNet `CMcsBus_MotorControlNet ^ m_pMcsBus_MotorControlNet [protected]`

11.109.3.2 m_pRoboFluidDevice `CRoboFluidDevice* m_pRoboFluidDevice [protected]`

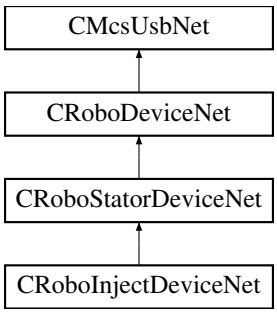
11.109.4 Property Documentation

11.109.4.1 McsBus_MotorControl `CMcsBus_MotorControlNet^ McsBus_MotorControl [get]`

11.110 CRoboInjectDeviceNet Class Reference

CRoboInjectDeviceNet is the to control the MCS RoboInject device

Inheritance diagram for CRoboInjectDeviceNet:



Public Member Functions

- CRoboInjectDeviceNet (void)

Additional Inherited Members

11.110.1 Detailed Description

CRoboInjectDeviceNet is the to control the MCS RoboInject device

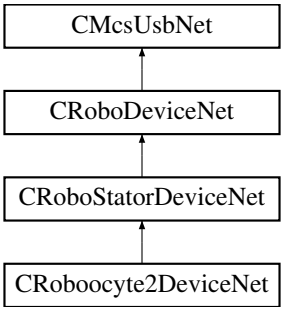
11.110.2 Constructor & Destructor Documentation

11.110.2.1 CRoboInjectDeviceNet() `CRoboInjectDeviceNet ( void )`

11.111 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

11.111.1 Detailed Description

CRoboocyte2DeviceNet is the class to control the MCS Roboocyte2 device

11.111.2 Constructor & Destructor Documentation

11.111.2.1 CRoboocyte2DeviceNet() CRoboocyte2DeviceNet (void)

11.111.3 Member Function Documentation

11.111.3.1 GetAxisLED() bool GetAxisLED ( )

11.111.3.2 GetGilsonDevice() CGilsonDeviceNet ^ GetGilsonDevice ( )

11.111.3.3 GetMcsBus_Extension() CMcsBus_ExtensionNet ^ GetMcsBus_Extension ( )

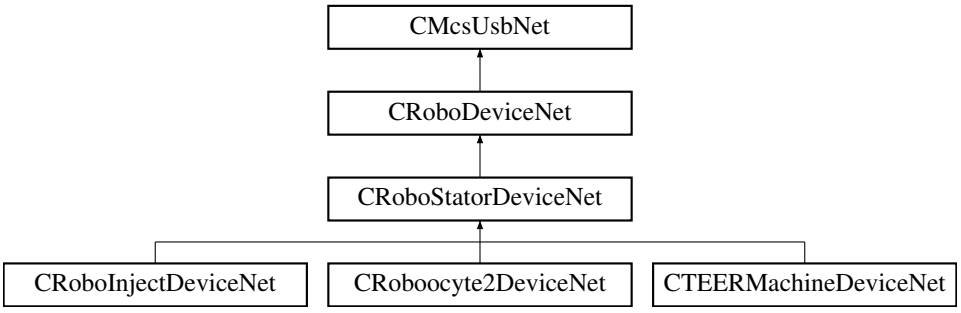
11.111.3.4 GetRoboDacq() CRoboDacqNet ^ GetRoboDacq ( )

11.111.3.5 GetRoboFluidDevice() CRoboFluidDeviceNet ^ GetRoboFluidDevice ( )

11.111.3.6 SetAxisLED() void SetAxisLED (bool *onoff*)

11.112 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::Interop↩ Services::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^ RoboMainStatorLowLevelCommand [get]

Additional Inherited Members

11.112.1 Constructor & Destructor Documentation

11.112.1.1 CRoboStatorDeviceNet() CRoboStatorDeviceNet (void)

11.112.2 Member Function Documentation

11.112.2.1 FindReferenceI() [1/2] void FindReferenceI ( )

11.112.2.2 FindReferenceI() [2/2] void FindReferenceI (int *timeout*)

11.112.2.3 FindReferenceXY() [1/2] void FindReferenceXY ( )

11.112.2.4 FindReferenceXY() [2/2] void FindReferenceXY (int *timeout*)

11.112.2.5 FindReferenceZ() [1/2] void FindReferenceZ ( )

11.112.2.6 FindReferenceZ() [2/2] void FindReferenceZ (int *timeout*)

11.112.2.7 GetCurrentPositionI() void GetCurrentPositionI ([System::Runtime::InteropServices::Out] int% *i*)

11.112.2.8 GetCurrentPositionXY()

```
void GetCurrentPositionXY (
    [System::Runtime::InteropServices::Out] int% x,
    [System::Runtime::InteropServices::Out] int% y )
```

11.112.2.9 GetCurrentPositionZ()

```
void GetCurrentPositionZ (
    [System::Runtime::InteropServices::Out] int% z )
```

11.112.2.10 HasRefI()

```
unsigned char HasRefI ( )
```

11.112.2.11 HasRefXY()

```
unsigned char HasRefXY ( )
```

11.112.2.12 HasRefZ()

```
unsigned char HasRefZ ( )
```

11.112.2.13 MoveAbsI() [1/2]

```
void MoveAbsI (
    int i )
```

11.112.2.14 MoveAbsI() [2/2]

```
void MoveAbsI (
    int i,
    int timeout )
```

11.112.2.15 MoveAbsXY() [1/2]

```
void MoveAbsXY (
    int x,
    int y )
```

11.112.2.16 MoveAbsXY() [2/2]

```
void MoveAbsXY (
    int x,
    int y,
    int timeout )
```

11.112.2.17 MoveAbsZ() [1/2] void MoveAbsZ (

```
int z )
```

11.112.2.18 MoveAbsZ() [2/2] void MoveAbsZ (

```
int z,
int timeout )
```

11.112.2.19 SetAccelerationI() void SetAccelerationI (

```
int a )
```

11.112.2.20 SetAccelerationNativeI() void SetAccelerationNativeI (

```
int a )
```

11.112.2.21 SetAccelerationNativeXY() void SetAccelerationNativeXY (

```
int a )
```

11.112.2.22 SetAccelerationNativeZ() void SetAccelerationNativeZ (

```
int a )
```

11.112.2.23 SetAccelerationXY() void SetAccelerationXY (

```
int a )
```

11.112.2.24 SetAccelerationZ() void SetAccelerationZ (

```
int a )
```

11.112.2.25 SetCurrentAndAirXY() [1/2] void SetCurrentAndAirXY (

```
unsigned short onoff )
```

11.112.2.26 SetCurrentAndAirXY() [2/2] void SetCurrentAndAirXY (

```
unsigned short onoff,
int timeout )
```

11.112.2.27 SetSpeedI() void SetSpeedI (

```
int v )
```

11.112.2.28 SetSpeedNativeI() void SetSpeedNativeI (

```
int v )
```

11.112.2.29 SetSpeedNativeXY() void SetSpeedNativeXY (

```
int v )
```

11.112.2.30 SetSpeedNativeZ() void SetSpeedNativeZ (

```
int v )
```

11.112.2.31 SetSpeedXY() void SetSpeedXY (

```
int v )
```

11.112.2.32 SetSpeedZ() void SetSpeedZ (

```
int v )
```

11.112.2.33 SetVelocityI() void SetVelocityI (

```
int v )
```

11.112.2.34 SetVelocityXY() void SetVelocityXY (

```
int v )
```

11.112.2.35 SetVelocityZ() void SetVelocityZ (int *v*)

11.112.2.36 StopMovementI() [1/2] void StopMovementI ( )

11.112.2.37 StopMovementI() [2/2] void StopMovementI (int *timeout*)

11.112.2.38 StopMovementXY() [1/2] void StopMovementXY ( )

11.112.2.39 StopMovementXY() [2/2] void StopMovementXY (int *timeout*)

11.112.2.40 StopMovementZ() [1/2] void StopMovementZ ( )

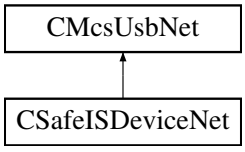
11.112.2.41 StopMovementZ() [2/2] void StopMovementZ (int *timeout*)

11.112.3 Property Documentation

11.112.3.1 RoboMainStatorLowLevelCommand RoboMainStatorLowLevelCommands^ RoboMainStator↩ LowLevelCommand [get]

11.113 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

11.113.1 Detailed Description

11.113.2 Constructor & Destructor Documentation

11.113.2.1 CSafeISDeviceNet() `CSafeISDeviceNet (void )`

Initializes a new instance of the CSafeISDeviceNet class.

11.113.2.2 ~CSafeISDeviceNet() `~CSafeISDeviceNet (void )`

Releases unmanaged resources and performs other cleanup operations before the CSafeISDeviceNet is reclaimed by garbage collection.

11.113.3 Member Function Documentation

11.113.3.1 SetAdcChannels() `void SetAdcChannels ( unsigned char channels )`

Sets the ADC channels you want to be sampled

Parameters

channels	The bitmap of the 8 channels. Set bit to 1 for the channels you want measure

11.113.3.2 SetAdcSamplePos() `void SetAdcSamplePos ( array< unsigned short >^ positions )`

Sets the sample position of the ADC.

Parameters

positions	The positions in units of 0.1µs.

11.113.3.3 SetDacMode() `void SetDacMode ( unsigned char mode )`

Sets the DAC mode.

Parameters

mode	The mode: 0 = Impedance ; 1 = Amperometry

11.113.3.4 SetDacPeriode() `void SetDacPeriode ( unsigned int periode )`

Sets the DAC periode.

Parameters

periode	The periode in units of 10µs.

11.113.3.5 SetDacPulseform() `void SetDacPulseform ( array< short >^ pulseform )`

Sets the DAC pulseform.

Parameters

pulseform	The pulseform.

11.113.3.6 SetSwitches() `void SetSwitches ( unsigned short switches )`

Sets the switches for all electrodes on the device. Do not use during measurement

Parameters

switches	The switches: See Schematics for the meaning

11.113.4 Property Documentation

11.113.4.1 DacqDevice `CMcsUsbDacqNet^ DacqDevice [get]`

Gets the CMcsUsbDacqNet. Use this to control the data aquisition.

11.113.4.2 FluidControlDevice `CFluidControlDeviceNet^ FluidControlDevice [get]`

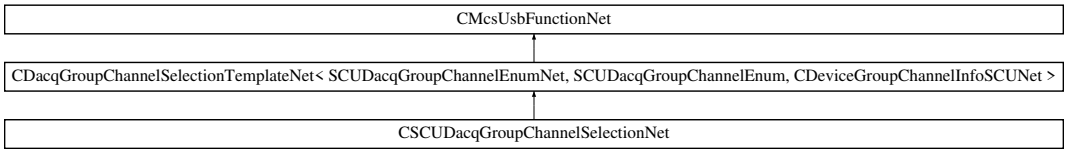
Gets the CFluidControlDeviceNet. Use this to control the valves. Only SetSingleValve is implemented for CSafeISDeviceNet.

11.113.4.3 RoboDevice `CRoboDeviceNet^ RoboDevice [get]`

Gets the CRoboDeviceNet. Use this to control the syringe.

11.114 CSCUDacqGroupChannelSelectionNet Class Reference

Inheritance diagram for CSCUDacqGroupChannelSelectionNet:



Public Member Functions

- CSCUDacqGroupChannelSelectionNet (CMcsUsbNet^ mcsusb)

Additional Inherited Members

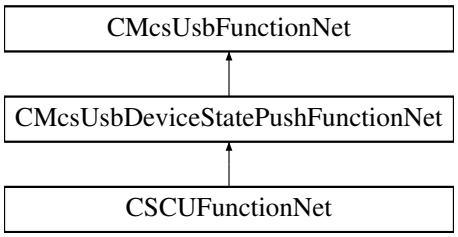
11.114.1 Constructor & Destructor Documentation

11.114.1.1 CSCUDacqGroupChannelSelectionNet() `CSCUDacqGroupChannelSelectionNet ( CMcsUsbNet^ mcsusb )`

11.115 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^ pSCUFunctionPointer↩ Container)

  Initializes a new instance of the CSCUFunctionNet class.
- CSCUFunctionNet (CMcsUsbNet^ mcsusb)
- virtual ~CSCUFunctionNet ()
- !CSCUFunctionNet ()
- uint32_t GetAvailableHeadstages ()

  Gets a bitmap of available headstages.
- bool IsInDacqLegacyMode ()

  Is the SCU in legacy mode
- void SetDacqLegacyMode (bool enable)

  Enable the SCU legacy mode
- uint32_t GetMaxStimulusChannelsPerHeadstage ()

  Gets the maximal number of stimulation channels a headstage can have.
- uint32_t GetMaxNumberOfHeadstages ()

  Gets the maximal number of headstages.
- SCU_HeadstageIdEnumNet GetHeadstageID (uint32_t Headstage)

Gets the headstage fpga ID.

- bool IsHeadstageAvailable (uint32_t Headstage)

  Checks whether the given headstage is available.
- void PowerHS (uint32_t Headstage, bool power)

  Power the HS
- bool IsHSPowered (uint32_t Headstage)

  Is the HS powered
- bool HasHSPowerSwitch ()

  Has SCU HS power switch
- String ^ GetHeadstageSerialNumber (uint32_t Headstage)

  Gets the serial number of a given headstage.
- uint32_t GetHeadstageNumberOfAnalogChannels (uint32_t Headstage)

  Gets the number of analog channels for a given headstage.
- 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 GetHeadstageGainInPermille (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

11.115.1 Detailed Description

CSCUFunctionNet is the class to control the SCU device

11.115.2 Constructor & Destructor Documentation

11.115.2.1 CSCUFunctionNet() [1/2] CSCUFunctionNet (CMcsUsbNet^ *mcsusb*, CMcsUsbFunctionPointerContainer^ *pSCUFunctionPointerContainer*)

Initializes a new instance of the CSCUFunctionNet class.

11.115.2.2 CSCUFunctionNet() [2/2] CSCUFunctionNet (CMcsUsbNet^ *mcsusb*)

11.115.2.3 ~CSCUFunctionNet() virtual ~CSCUFunctionNet ( ) [virtual]

11.115.2.4 "!CSCUFunctionNet() !CSCUFunctionNet ( )

11.115.3 Member Function Documentation

11.115.3.1 AutomaticAnalogOut() void AutomaticAnalogOut (bool *automatic*)

Sets automatic source channel selection

Parameters

automatic	Automatic

11.115.3.2 EnableAnalogOut() void EnableAnalogOut (bool *enable*)

Enables AnalogOut globally

Parameters

enable	Enable

11.115.3.3 GetAnalogOutADCRange() `uint32_t GetAnalogOutADCRange ( )`

Gets the analog out ADC range

Returns

Range

11.115.3.4 GetAnalogOutChannels() `uint32_t GetAnalogOutChannels ( uint32_t out_channel )`

Get the connected source channel number for a certain output channel

Parameters

out_channel	Output channel number

Returns

Source channel number

11.115.3.5 GetAnalogOutDACRange() `AnalogOut_DAC_Range_EnumNet GetAnalogOutDACRange ( )`

Gets the analog out DAC range

Returns

Range

11.115.3.6 GetAvailableHeadstages() `uint32_t GetAvailableHeadstages ( )`

Gets a bitmap of available headstages.

Returns

The bitmap of available headstages.

11.115.3.7 GetFilterProperties() `array<CFilterPropertyNet^> ^ GetFilterProperties ( SCUDacqGroupChannelEnumNet GroupID, int filterConfigurations_Length )`

Gets multiple filter properties

Parameters

GroupID	The group ID
filterConfigurations_Length	The maximal length of filterConfigurations.

Returns

array of filter properties

11.115.3.8 GetFilterProperty() `CFilterPropertyNet ^ GetFilterProperty ( SCUDacqGroupChannelEnumNet GroupID, uint32_t FilterNumber )`

Gets the filter property

Parameters

GroupID	The group ID
FilterNumber	The filter number

Returns

The filter property

11.115.3.9 GetHeadstageAdcBits() `uint32_t GetHeadstageAdcBits ( uint32_t Headstage )`

Gets the Number of ADC bits for a given headstage.

Parameters

Headstage	The headstage to query.

Returns

The number of bits the ADC has for the given headstage.

11.115.3.10 GetHeadstageAdcRangeInMicroVolt() `uint32_t GetHeadstageAdcRangeInMicroVolt ( uint32_t Headstage )`

Gets the ADC Range in uV for a given headstage.

Parameters

Headstage	The headstage to query.

Returns

The ADC Range in uV for the given headstage.

11.115.3.11 GetHeadstageDacBits() `uint32_t GetHeadstageDacBits ( uint32_t Headstage )`

Gets the Number of DAC bits for a given headstage.

Parameters

Headstage	The headstage to query.

Returns

The number of bits the DAC has for the given headstage.

11.115.3.12 GetHeadstageDacCurrentRangeInMicroAmpere() `uint32_t GetHeadstageDacCurrentRangeInMicroAmpere ( uint32_t Headstage )`

Gets the DAC Current Range in uA for a given headstage.

Parameters

Headstage	The headstage to query.

Returns

The DAC Current Range in uA for the given headstage.

11.115.3.13 GetHeadstageDacCurrentResolutionInNanoAmpere() `uint32_t GetHeadstageDacCurrentResolutionInNanoAmpere ( uint32_t Headstage )`

Gets the DAC Current Resolution in nA for a given headstage.

Parameters

Headstage	The headstage to query.

Returns

The DAC Current Resolution in nA for the given headstage.

11.115.3.14 GetHeadstageDacVoltageRangeInMilliVolt()

```
uint32_t GetHeadstageDacVoltageRangeInMilliVolt (
            uint32_t Headstage )
```

Gets the DAC Voltage Range in mV for a given headstage.

Parameters

Headstage	The headstage to query.

Returns

The DAC Voltage Range in mV for the given headstage.

11.115.3.15 GetHeadstageDacVoltageResolutionInMicroVolt()

```
uint32_t GetHeadstageDacVoltageResolutionInMicroVolt (
            uint32_t Headstage )
```

Gets the DAC Voltage Resolution in uV for a given headstage.

Parameters

Headstage	The headstage to query.

Returns

The DAC Voltage Resolution in uV for the given headstage.

11.115.3.16 GetHeadstageFrameCyclesToCompare()

```
uint32_t GetHeadstageFrameCyclesToCompare (
            uint32_t Headstage )
```

Gets the frame cycles to compare of a given headstage.

Parameters

Headstage	The headstage to query.

Returns

The samplerate in Hz for the given headstage.

11.115.3.17 GetHeadstageGainInPermille() `uint32_t GetHeadstageGainInPermille ( uint32_t Headstage )`

Gets the gain factor in permille for a given headstage.

Parameters

Headstage	The headstage to query.

Returns

The gain factor in permille for the given headstage.

11.115.3.18 GetHeadstageID() `SCU_HeadstageIdEnumNet GetHeadstageID ( uint32_t Headstage )`

Gets the headstage fpga ID.

Parameters

Headstage	The headstage to query.

Returns

The headstage fpga ID.

11.115.3.19 GetHeadstageLinkSpeed() `uint32_t GetHeadstageLinkSpeed ( uint32_t Headstage )`

Gets the Link speed of a given headstage.

Parameters

Headstage	The headstage to query.

Returns

The samplerate in Hz for the given headstage.

11.115.3.20 GetHeadstageNumberOfAnalogChannels() `uint32_t GetHeadstageNumberOfAnalogChannels (`

```
uint32_t Headstage )
```

Gets the number of analog channels for a given headstage.

Parameters

Headstage	The headstage to query.

Returns

The number of analog channels the headstage has.

11.115.3.21 GetHeadstageNumberOfStimulationChannels() `uint32_t GetHeadstageNumberOfStimulation↩ Channels (`

```
uint32_t Headstage )
```

Gets the number of stimulation channels for a given headstage.

Parameters

Headstage	The headstage to query.

Returns

The number of stimulation channels the headstage has.

11.115.3.22 GetHeadstagePowerStateAtStart() `bool GetHeadstagePowerStateAtStart (`

```
uint32_t Headstage )
```

Gets the Power Status at SCU Power on of a given headstage.

Parameters

Headstage	The headstage to query.

Returns

The Power State at startup for the given headstage: bool false -> off, bool true -> on.

11.115.3.23 GetHeadstageSamplerate() `uint32_t GetHeadstageSamplerate ( uint32_t Headstage )`

Gets the Samplerate of a given headstage.

Parameters

Headstage	The headstage to query.

Returns

The samplerate in Hz for the given headstage.

11.115.3.24 GetHeadstageSerialNumber() `String ^ GetHeadstageSerialNumber ( uint32_t Headstage )`

Gets the serial number of a given headstage.

Parameters

Headstage	The headstage to query.

Returns

The serial number of the headstage.

11.115.3.25 GetMaxNumberOfHeadstages() `uint32_t GetMaxNumberOfHeadstages ( )`

Gets the maximal number of headstages.

Returns

The maximal number of headstages.

11.115.3.26 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.

11.115.3.27 GetReferenceElectrodeMode() `ReferenceElectrodeModeEnumNet GetReferenceElectrodeMode ( uint32_t Headstage )`

Gets the mode for the reference electrode

Parameters

Headstage	The headstage number

Returns

The mode

11.115.3.28 GetReferenceElectrodeSwitchState() `ReferenceElectrodeSwitchPositionEnumNet GetReferenceElectrodeSwitchState ( uint32_t Headstage )`

Gets the position of the switch for the reference electrode

Parameters

Headstage	The headstage number

Returns

The switch position

11.115.3.29 HasAnalogOut() `bool HasAnalogOut ( )`

Has AnalogOut hardware

Returns

Enabled

11.115.3.30 HasGalvanicIsolation() bool HasGalvanicIsolation ( )

Has galvanic isolated hardware

Returns

Enabled

11.115.3.31 HasHSPowerSwitch() bool HasHSPowerSwitch ( )

Has SCU HS power switch

Returns

Has Switch

11.115.3.32 IsAnalogOutEnabled() bool IsAnalogOutEnabled ( )

Is AnalogOut enabled

Returns

Enabled

11.115.3.33 IsAutomaticAnalogOut() bool IsAutomaticAnalogOut ( )

Is Automatic source channel selection selected

Returns

Automatic

11.115.3.34 IsHeadstageAvailable() bool IsHeadstageAvailable (uint32_t *Headstage*)

Checks whether the given headstage is available.

Parameters

Headstage	The headstage to query.

Returns

'true' if the headstage is connected.

11.115.3.35 IsHSPowered() `bool IsHSPowered (`

`uint32_t` *Headstage*)

Is the HS powered

Parameters

Headstage	The headstage to query.

Returns

'true' if the headstage is powered.

11.115.3.36 IsInDacqLegacyMode() `bool IsInDacqLegacyMode ( )`

Is the SCU in legacy mode

Returns

Is Enabled

11.115.3.37 OnGetAvailableHeadstages() `delegate void OnGetAvailableHeadstages (`

`uint32_t` *AvailableHeadstages*)

11.115.3.38 OnIsHeadstageAvailable() `delegate void OnIsHeadstageAvailable (`

`uint32_t` *Headstage*,
`bool` *available*)

11.115.3.39 PowerHS() `void PowerHS (`

`uint32_t` *Headstage*,
`bool` *power*)

Power the HS

Parameters

Headstage	The headstage to query.
power	'true' if the headstage is powered.

11.115.3.40 SetAnalogOutADCRange() `void SetAnalogOutADCRange (`

```
uint32_t range )
```

Sets the analog out ADC range

Parameters

range	Range

11.115.3.41 SetAnalogOutChannels() `void SetAnalogOutChannels (`

```
uint32_t out_channel,
uint32_t source_channel )
```

Set the source channel number for a certain output channel

Parameters

out_channel	Output channel number
source_channel	Source channel number

11.115.3.42 SetAnalogOutDACRange() `void SetAnalogOutDACRange (`

```
AnalogOut_DAC_Range_EnumNet range )
```

Sets the analog out DAC range

Parameters

range	Range

11.115.3.43 SetDacqLegacyMode() `void SetDacqLegacyMode (`

```
bool enable )
```

Enable the SCU legacy mode

Parameters

enable	Enable

11.115.3.44 SetHeadstageFrameCyclesToComparePermanent() `void SetHeadstageFrameCyclesTo↩ ComparePermanent (`

```
            uint32_t Headstage,
            uint32_t FrameCycles )
```

Sets the frame cycles to compare permanent on a given headstage.

Parameters

Headstage	The headstage number
FrameCycles	The samplerate in Hz for the given headstage.

11.115.3.45 SetHeadstageLinkSpeedPermanent() `void SetHeadstageLinkSpeedPermanent (`

```
            uint32_t Headstage,
            uint32_t LinkSpeed )
```

Sets the Link speed permanent on a given headstage.

Parameters

Headstage	The headstage number
LinkSpeed	The samplerate in Hz for the given headstage.

11.115.3.46 SetHeadstageNumberOfAnalogChannelsPermanent() `void SetHeadstageNumberOfAnalog↩ ChannelsPermanent (`

```
            uint32_t Headstage,
            uint32_t NumberOfchannels )
```

Sets the number of analog channels permanent for a given headstage.

Parameters

Headstage	The headstage number
NumberOfchannels	The number of analog channels the headstage has to transmit

11.115.3.47 SetHeadstagePowerStateAtStart() `void SetHeadstagePowerStateAtStart (`

```
uint32_t Headstage,
bool Powerstatus )
```

Sets the Power Status at SCU Power on of a given headstage.

Parameters

Headstage	The headstage number
Powerstatus	The Power State at startup for the given headstage: bool false -> off, bool true -> on.

11.115.3.48 SetHeadstageSampleratePermanent() `void SetHeadstageSampleratePermanent (`

```
uint32_t Headstage,
uint32_t Samplerate )
```

Sets the samplerate permanent on a given headstage.

Parameters

Headstage	The headstage number
Samplerate	The samplerate in Hz for the given headstage.

11.115.3.49 SetReferenceElectrodeMode() `void SetReferenceElectrodeMode (`

```
uint32_t Headstage,
ReferenceElectrodeModeEnumNet NewValue )
```

Sets the mode for the reference electrode

Parameters

Headstage	The headstage number
NewValue	The mode

11.115.3.50 SetReferenceElectrodeSwitchState() `void SetReferenceElectrodeSwitchState (`

```
uint32_t Headstage,
ReferenceElectrodeSwitchPositionEnumNet NewSwitchPos )
```

Sets the position of the switch for the reference electrode

Parameters

Headstage	The headstage number
NewSwitchPos	The switch position

11.115.4 Event Documentation

11.115.4.1 GetAvailableHeadstagesEvent `OnGetAvailableHeadstages^ GetAvailableHeadstagesEvent [add], [remove], [raise]`

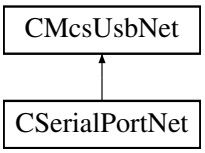
Event fires when the bitmap of available headstages has changed

11.115.4.2 IsHeadstageAvailableEvent `OnIsHeadstageAvailable^ IsHeadstageAvailableEvent [add], [remove], [raise]`

Event fires when 'true' if the headstage is connected for the headstage to query has changed

11.116 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

11.116.1 Constructor & Destructor Documentation

11.116.1.1 CSerialPortNet() `CSerialPortNet ( void )`

11.116.2 Member Function Documentation

11.116.2.1 GetBytesAvailable()

```
int GetBytesAvailable (
    void  )
```

11.116.2.2 Receive() [1/2]

```
array<byte> ^ Receive (
    int length )
```

11.116.2.3 Receive() [2/2]

```
array<byte> ^ Receive (
    void  )
```

11.116.2.4 ReceiveString() [1/2]

```
String ^ ReceiveString (
    int length )
```

11.116.2.5 ReceiveString() [2/2]

```
String ^ ReceiveString (
    void  )
```

11.116.2.6 Send() [1/2]

```
void Send (
    array< byte >^ buffer )
```

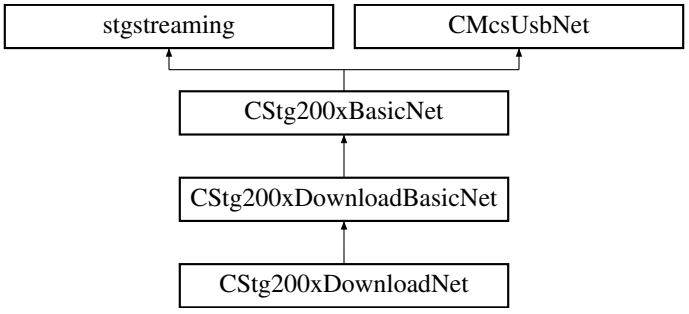
11.116.2.7 Send() [2/2]

```
void Send (
    String^ command )
```

11.117 CStg200xBasicNet Class Reference

Base class for the Stg200x.

Inheritance diagram for CStg200xBasicNet:



Public Member Functions

- virtual ~CStg200xBasicNet ()

  The destructor.
- void SetOutputRate (uint32_t rate)

  Change the output rate of the STG. Valid rates are from 1000 Hz to 50000 Hz.
- uint32_t GetOutputRate ()

  Queries the output rate of the STG. Valid rates are from 1000 Hz to 50000 Hz.
- void SendStart (uint32_t triggermap)

  Start (Trigger) the STG. The startup delay is in the range of a few ms.
- void SendStop (uint32_t triggermap)

  Stop some or all triggers of the STG.
- void SendStop (uint32_t triggermap, int options)

  Stop some or all triggers of the STG.
- void GetStgVersionInfo ([Out]String^% SwVersion, [Out]String^% HwVersion)

  Queries software and hardware version.
- virtual int32_t GetDACResolution ()

  Gets number of bits of the DAC resolution.
- virtual int32_t GetTimeResolutionInNanoSeconds ()

  Gets the time resolution of the STG.
- virtual int32_t GetVoltageRangeInMicroVolt (uint32_t channel)

  Gets the Voltage Range of the specified channel in Microvolts.
- virtual int32_t GetVoltageResolutionInMicroVolt (uint32_t channel)

  Gets the Voltage Resolution of the specified channel in Microvolts.
- virtual int32_t GetCurrentRangeInNanoAmp (uint32_t channel)

  Gets the Current Range of the specified channel in Nanoamps.
- virtual int32_t GetCurrentResolutionInNanoAmp (uint32_t channel)

  Gets the Current Resolution of the specified channel in Nanoamps.
- virtual int32_t GetCurrentResolutionInPicoAmp (uint32_t channel)

  Gets the Current Resolution of the specified channel in Picoamps.
- void GetStgProgramInfo ([Out]bool% IsProgrammed, [Out]System::Runtime::InteropServices::ComTypes::↩ FILETIME% timestamp, [Out]String^% filename, [Out]Guid% guid)

  Queries Download information from the STG. If download information was stored by the use of SetStgProgramInfo, this function can be used to retrieve it.
- void GetStgProgramInfo ([Out]bool% IsProgrammed, [Out]DateTime% timestamp, [Out]String^% filename, [Out]Guid% guid)

  Queries Download information from the STG. If download information was stored by the use of SetStgProgramInfo, this function can be used to retrieve it.
- void SetStgProgramInfo (DateTime timestamp, String^ filename, Guid guid)

  Store Download information in the STG. This function can be used to store the filename and timestamp of the last download for later query.
- uint32_t GetAvailableMemory ()

  Gets the amount of memory available in the currently selected segment of the STG.
- uint32_t GetTotalMemory ()

  Gets the total amount of memory available on the STG (all segments).
- virtual uint32_t GetNumberOfAnalogChannels ()

  Gets the Number of available analog channels of the device.
- virtual uint32_t GetNumberOfSyncoutChannels ()

  Gets the Number of available syncout channels of the device.
- virtual uint32_t GetNumberOfTriggerInputs ()

  Gets the Number of trigger inputs of the device.
- virtual uint32_t GetNumberOfHWDACPaths ()

Gets the Number of HW Stimulation DACs of the device.

- virtual uint32_t GetNumberOfStimulationSourcesPerElectrode ()

  Gets the number of stimulation sources (DACs) per electrode.
- virtual void SetVoltageMode (unsigned int channel)

  Sets a channel to voltage mode (STG3008-FA and STG400x only).
- virtual void SetCurrentMode (unsigned int channel)

  Sets a channel to current mode (STG3008-FA and STG400x only).
- virtual void SetVoltageMode ()

  Sets all channels to voltage mode (STG3008-FA and STG400x only).
- virtual void SetCurrentMode ()

  Sets all channels to current mode (STG3008-FA and STG400x only).
- virtual System::Collections::Generic::List< int32_t > ^ GetVoltageRangeListInMicroVolt (uint32_t channel)

  Gets a list of current ranges supported by the device (STG5 only).
- virtual uint32_t GetNumberOfVoltageRangeIndexes ()

  Gets the number of voltage ranges (STG5 only).
- virtual uint32_t GetVoltageRangeInMicroVoltByIndex (uint32_t channel, uint32_t index)

  Gets the voltage range for the given channel and index (STG5 only).
- virtual uint32_t GetVoltageRangeSelectedIndex (uint32_t channel)

  Gets the currently selected range index for the voltage output (not used yet).
- virtual void SetVoltageRangeSelectedIndex (uint32_t channel, uint32_t rangeIndex)

  Sets the range index for the voltage output (not used yet).
- virtual System::Collections::Generic::List< int32_t > ^ GetCurrentRangeListInNanoAmp (uint32_t channel)

  Gets a list of current ranges supported by the device (STG5 only).
- virtual uint32_t GetNumberOfCurrentRangeIndexes ()

  Gets the number of current ranges (STG5 only).
- virtual uint32_t GetCurrentRangeInNanoAmpByIndex (uint32_t channel, uint32_t index)

  Gets the current range for the given channel and index (STG5 only).
- virtual uint32_t GetCurrentRangeSelectedIndex (uint32_t channel)

  Gets the currently selected range index for the current output (STG5 only).
- virtual void SetCurrentRangeSelectedIndex (uint32_t channel, uint32_t rangeIndex)

  Sets the range index for the current output (STG5 only).
- virtual void SetMeasurementMode (unsigned int channel)

  Sets a channel to measurement mode (STG3008-FA).
- virtual void SetFAAmplification (Stg3008FilterAmpAmplificationEnumNet amplification)

  Sets the amplification of the filter amplifier (STG3008-FA).
- virtual Stg3008FilterAmpAmplificationEnumNet GetFAAmplification ()

  Gets the currently selected amplification of the filter amplifier (STG3008-FA).
- bool GetCanDisableCalibration (uint32_t channel)

  Gets whether the channel can disable autocalibration.
- bool GetCanDisableCurrentswitch (uint32_t channel)

  Gets whether the channel can disable its current switch.
- bool GetHasLowCurrentswitchLatency (uint32_t channel)

  Gets whether the channel has a low latency current switch.
- virtual void SetAutocalibrationDisabled (unsigned int channel, bool disable)

  Sets the autocalibration configuration.
- virtual bool GetAutocalibrationDisabled (unsigned int channel)

  Gets the autocalibration configuration.
- virtual void SetElectrodeMode (uint32_t electrode, array< ElectrodeModeEnumNet >^ mode)

  Puts an electrode in either automatic or manual mode.
- virtual void SetElectrodeMode (uint32_t electrode, ElectrodeModeEnumNet mode)

  Puts an electrode in either automatic or manual mode.

- virtual void SetElectrodeMode (uint32_t Scu_HS, uint32_t electrode, array< ElectrodeModeEnumNet >^ mode)

  Puts an electrode in either automatic or manual mode.
- virtual void SetElectrodeMode (uint32_t Scu_HS, uint32_t electrode, ElectrodeModeEnumNet mode)

  Puts an electrode in either automatic or manual mode.
- virtual uint32_t GetElectrodeMode (uint32_t electrode)

  Gets the mode an electrode is in.
- virtual uint32_t GetElectrodeMode (uint32_t Scu_HS, uint32_t electrode)

  Gets the mode an electrode is in.
- virtual void SetElectrodeDacMux (uint32_t electrode, uint32_t listmodeIndex, array< ElectrodeDacMuxEnumNet >^ dacMux)

  Defines the DAC to use for an electrode.
- virtual void SetElectrodeDacMux (uint32_t electrode, uint32_t listmodeIndex, ElectrodeDacMuxEnumNet dacMux)

  Defines the DAC to use for an electrode.
- virtual void SetElectrodeDacMux (uint32_t Scu_HS, uint32_t electrode, uint32_t listmodeIndex, ElectrodeDacMuxEnumNet dacMux)

  Defines the DAC to use for an electrode.
- virtual void SetElectrodeDacMux (uint32_t Scu_HS, uint32_t electrode, uint32_t listmodeIndex, array< ElectrodeDacMuxEnumNet >^ dacMux)

  Defines the DAC to use for an electrode.
- virtual ElectrodeDacMuxEnumNet GetElectrodeDacMux (uint32_t electrode, uint32_t listmodeIndex)

  Gets the DAC which is used for an electrode.
- virtual ElectrodeDacMuxEnumNet GetElectrodeDacMux (uint32_t Scu_HS, uint32_t electrode, uint32_t listmodeIndex)

  Gets the DAC which is used for an electrode.
- virtual void SetElectrodeEnable (uint32_t electrode, uint32_t listmodeIndex, array< bool >^ enable)

  Enables or disables the stimulation switch for an electrode.
- virtual void SetElectrodeEnable (uint32_t electrode, uint32_t listmodeIndex, bool enable)

  Enables or disables the stimulation switch for an electrode.
- virtual void SetElectrodeEnable (uint32_t Scu_HS, uint32_t electrode, uint32_t listmodeIndex, bool enable)

  Enables or disables the stimulation switch for an electrode.
- virtual void SetElectrodeEnable (uint32_t Scu_HS, uint32_t electrode, uint32_t listmodeIndex, array< bool >^ enable)

  Enables or disables the stimulation switch for an electrode.
- virtual bool GetElectrodeEnable (uint32_t electrode, uint32_t listmodeIndex)

  Gets weather an electrode is enabled or disabled for stimulation.
- virtual bool GetElectrodeEnable (uint32_t Scu_HS, uint32_t electrode, uint32_t listmodeIndex)

  Gets weather an electrode is enabled or disabled for stimulation.
- virtual void SetExternalElectrodeEnable (uint32_t electrode, uint32_t listmodeIndex, array< bool >^ enable)

  Enables or disables the stimulation switch for an external electrode.
- virtual void SetExternalElectrodeEnable (uint32_t electrode, uint32_t listmodeIndex, bool enable)

  Enables or disables the stimulation switch for an external electrode.
- virtual bool GetExternalElectrodeEnable (uint32_t electrode, uint32_t listmodeIndex)

  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 outport port, can be Monitor, Manual or SyncOut
- virtual Stg200xDigoutModeEnumNet GetDigoutMode ()

  Gets the operation mode of the digital outport port, can be Monitor, Manual or SyncOut
- virtual void SetDigoutValue (uint32_t digoutValue)

  Sets the Value on the digital output port when in manual mode.
- virtual uint32_t GetDigoutValue ()

  Gets the Value on the digital output port.
- virtual uint32_t GetDiginValue ()

  Gets the Value on the digital input port.
- virtual void SetSyncoutMap (uint32_t channel, uint32_t syncoutMap)

  Sets the mapping between external syncout outputs and internal syncout channels.
- virtual uint32_t GetSyncoutMap (uint32_t channel)

  Gets the mapping between external syncout outputs and internal syncout channels.
- virtual bool HasDigitalPort ()

  Queries if the device has a digital port.
- virtual bool HasFilterAmp ()

  Queries if the device has a filter amplifier. Currently the STG3008-FA supports this feature.
- virtual bool LegacyHasOutputMap ()

  Queries if the device can map its stimulation channels. Legacy command, do not use.

Additional Inherited Members

11.117.1 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.

11.117.2 Constructor & Destructor Documentation

11.117.2.1 ~CStg200xBasicNet() `virtual ~CStg200xBasicNet ( ) [virtual]`

The destructor.

11.117.3 Member Function Documentation

11.117.3.1 GetAutocalibrationDisabled() `virtual bool GetAutocalibrationDisabled ( unsigned int channel ) [virtual]`

Gets the autocalibration configuration.

Parameters

channel	The channel number.

Returns

`true` if autocalibration is disabled.

11.117.3.2 GetAvailableMemory() `uint32_t GetAvailableMemory ( )`

Gets the amount of memory available in the currently selected segment of the STG.

Returns

The memory available in the currently selected segment in bytes.

11.117.3.3 GetBlankingEnable() [1/2] `virtual bool GetBlankingEnable ( uint32_t electrode ) [virtual]`

Gets whether an electrode should be blanked while stimulation is in progress.

Parameters

electrode	The electrode number.

Returns

true if blanking is enabled while stimulation is in progress.

11.117.3.4 GetBlankingEnable() [2/2] `virtual bool GetBlankingEnable ( uint32_t Scu_HS, uint32_t electrode ) [virtual]`

Gets whether an electrode should be blanked while stimulation is in progress.

Parameters

Scu_HS	The SCU headstage number.

Parameters

electrode	The electrode number.

Returns

true if blanking is enabled while stimulation is in progress.

11.117.3.5 GetCanDisableCalibration() `bool GetCanDisableCalibration ( uint32_t channel )`

Gets whether the channel can disable autocalibration.

Parameters

channel	The channel which is queried.

Returns

True if autocalibration can be disabled.

11.117.3.6 GetCanDisableCurrentswitch() `bool GetCanDisableCurrentswitch ( uint32_t channel )`

Gets whether the channel can disable its current switch.

Parameters

channel	The channel which is queried.

Returns

True if the current switch can be disabled.

11.117.3.7 GetCurrentRangeInNanoAmp() `virtual int32_t GetCurrentRangeInNanoAmp ( uint32_t channel ) [virtual]`

Gets the Current Range of the specified channel in Nanoamps.

Parameters

channel	Channel which is queried.

Returns

The Current Range of the specified channel in Nanoamps.

11.117.3.8 GetCurrentRangeInNanoAmpByIndex()

```
virtual uint32_t GetCurrentRangeInNanoAmpByIndex (
            uint32_t channel,
            uint32_t index ) [virtual]
```

Gets the current range for the given channel and index (STG5 only).

Parameters

channel	The channel.
index	The channel.

Returns

The current range in uA.

11.117.3.9 GetCurrentRangeListInNanoAmp()

```
virtual System::Collections::Generic::List<int32_t> ^ GetCurrentRangeListInNanoAmp (
            uint32_t channel ) [virtual]
```

Gets a list of current ranges supported by the device (STG5 only).

11.117.3.10 GetCurrentRangeSelectedIndex()

```
virtual uint32_t GetCurrentRangeSelectedIndex (
            uint32_t channel ) [virtual]
```

Gets the currently selected range index for the current output (STG5 only).

Parameters

channel	The channel to change.

Returns

The currently selected range index.

11.117.3.11 GetCurrentResolutionInNanoAmp()

```
virtual int32_t GetCurrentResolutionInNanoAmp (
            uint32_t channel ) [virtual]
```

Gets the Current Resolution of the specified channel in Nanoamps.

Parameters

channel	Channel which is queried.

Returns

The Current Resolution of the specified channel in Nanoamps.

11.117.3.12 GetCurrentResolutionInPicoAmp() `virtual int32_t GetCurrentResolutionInPicoAmp ( uint32_t channel ) [virtual]`

Gets the Current Resolution of the specified channel in Picoamps.

Parameters

channel	Channel which is queried.

Returns

The Current Resolution of the specified channel in Picoamps.

11.117.3.13 GetDacAmplificationFactor() `virtual double GetDacAmplificationFactor ( uint32_t DacNumber ) [virtual]`

Get the amplification factor for a DAC.

Parameters

DacNumber	The number of the DAC.

Returns

the amplifcation factor for the DAC queried, range is from -1.99999 to +1.99999.

11.117.3.14 GetDACResolution() `virtual int32_t GetDACResolution ( ) [virtual]`

Gets number of bits of the DAC resolution.

Returns

The DAC resolution in bits.

11.117.3.15 GetDiginValue() virtual uint32_t GetDiginValue ( ) [virtual]

Gets the Value on the digital input port.

Returns

The current value on the digital inputs.

11.117.3.16 GetDigoutMode() virtual Stg200xDigoutModeEnumNet GetDigoutMode ( ) [virtual]

Gets the operation mode of the digital outport port, can be Monitor, Manual or SyncOut

Returns

The current operation mode.

11.117.3.17 GetDigoutValue() virtual uint32_t GetDigoutValue ( ) [virtual]

Gets the Value on the digital output port.

Returns

The current value on the digital outputs.

11.117.3.18 GetElectrodeDacMux() [1/2] virtual ElectrodeDacMuxEnumNet GetElectrodeDacMux (uint32_t *electrode*, uint32_t *listmodeIndex*) [virtual]

Gets the DAC which is used for an electrode.

Parameters

electrode	The electrode number.
listmodeIndex	The index for listmode.

Returns

The DAC in use, can be 1, 2 or 3. If the electrode is grounded 0 is returned.

11.117.3.19 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

Scu_HS	The SCU headstage number.

Parameters

electrode	The electrode number.
listmodeIndex	The index for listmode.

Returns

The DAC in use, can be 1, 2 or 3. If the electrode is grounded 0 is returned.

11.117.3.20 GetElectrodeEnable() [1/2]

```
virtual bool GetElectrodeEnable (
    uint32_t electrode,
    uint32_t listmodeIndex ) [virtual]
```

Gets weather an electrode is enabled or disabled for stimulation.

Parameters

electrode	The electrode number.
listmodeIndex	The index for listmode.

Returns

true if the electrode is enabled, false if it is disabled.

11.117.3.21 GetElectrodeEnable() [2/2]

```
virtual bool GetElectrodeEnable (
    uint32_t Scu_HS,
    uint32_t electrode,
    uint32_t listmodeIndex ) [virtual]
```

Gets weather an electrode is enabled or disabled for stimulation.

Parameters

Scu_HS	The SCU headstage number.

Parameters

electrode	The electrode number.
listmodeIndex	The index for listmode.

Returns

true if the electrode is enabled, false if it is disabled.

11.117.3.22 GetElectrodeMode() [1/2] virtual uint32_t GetElectrodeMode (uint32_t *electrode*) [virtual]

Gets the mode an electrode is in.

Parameters

electrode	The electrode number.

Returns

0 for automatic and 3 for manual mode.

11.117.3.23 GetElectrodeMode() [2/2] virtual uint32_t GetElectrodeMode (uint32_t *Scu_HS*, uint32_t *electrode*) [virtual]

Gets the mode an electrode is in.

Parameters

Scu_HS	The SCU headstage number.

Parameters

electrode	The electrode number.

Returns

0 for automatic and 3 for manual mode.

11.117.3.24 GetEnableAmplifierProtectionSwitch() [1/2] virtual bool GetEnableAmplifierProtection↩ Switch (

```
            uint32_t electrode )  [virtual]
```

Gets whether the Amplifier Protection Switch is openend while stimulation is in progress.

Parameters

electrode	The electrode number.

Returns

true if the switch is to be opened, false if it is closed while stimulation is in progress.

11.117.3.25 GetEnableAmplifierProtectionSwitch() [2/2] virtual bool GetEnableAmplifierProtection↩ Switch (

```
            uint32_t Scu_HS,
            uint32_t electrode )  [virtual]
```

Gets whether the Amplifier Protection Switch is openend while stimulation is in progress.

Parameters

Scu_HS	The SCU headstage number.

Parameters

electrode	The electrode number.

Returns

true if the switch is to be opened, false if it is closed while stimulation is in progress.

11.117.3.26 GetExternalElectrodeEnable() `virtual bool GetExternalElectrodeEnable ( uint32_t electrode, uint32_t listmodeIndex ) [virtual]`

Gets weather an electrode is enabled or disabled for stimulation.

Parameters

electrode	The electrode number.
listmodeIndex	The index for listmode.

Returns

true if the electrode is enabled, false if it is disabled.

11.117.3.27 GetFAAmplification() `virtual Stg3008FilterAmpAmplificationEnumNet GetFAAmplification ( ) [virtual]`

Gets the currently selected amplification of the filter amplifier (STG3008-FA).

Returns

True the currently selected amplification.

11.117.3.28 GetHasLowCurrentswitchLatency() `bool GetHasLowCurrentswitchLatency ( uint32_t channel )`

Gets whether the channel has a low latency current switch.

Parameters

channel	The channel which is queried.

Returns

True if the channel has a low latency current switch.

11.117.3.29 GetHeadstage() virtual uint32_t GetHeadstage ( ) [virtual]

11.117.3.30 GetListmodeIndexRange() virtual void GetListmodeIndexRange (unsigned int *electrodeGroup*, unsigned int & *startIndex*, unsigned int & *endIndex*, unsigned int & *mode*) [virtual]

Query the range of list mode indexes to use for the given electrode group.

Parameters

electrodeGroup	The electrodegroup for which the range is queried.

Parameters

startIndex	The index of the first active element in the listmode list.
endIndex	The index of the last active element in the listmode list.
mode	0 for "start with startIndex", 1 for "start with endIndex".

11.117.3.31 GetListmodeTriggerSource() virtual TriggerSourceEnumNet GetListmodeTriggerSource (unsigned int *electrodeGroup*) [virtual]

Query the currently active signal which triggers the transition from one list mode entry to the next. For triggersource use the Enum which corresponds to the device in use, for example use SCUDigitalSourceEnumNet and cast to TriggerSourceEnumNet if working with an SCU device.

Parameters

electrodeGroup	The electrodegroup for which the triggersource is queried.

Returns

One of the possible sources for the transition.

11.117.3.32 GetNumberOfAnalogChannels() virtual uint32_t GetNumberOfAnalogChannels ( ) [virtual]

Gets the Number of available analog channels of the device.

Returns

The number of analog channels.

11.117.3.33 GetNumberOfCurrentRangeIndexes() `virtual uint32_t GetNumberOfCurrentRangeIndexes ( ) [virtual]`

Gets the number of current ranges (STG5 only).

Returns

The number of current ranges available on the device.

11.117.3.34 GetNumberOfHWDACPaths() `virtual uint32_t GetNumberOfHWDACPaths ( ) [virtual]`

Gets the Number of HW Stimulation DACs of the device.

Returns

The number of independent HW Stimulation outputs.

11.117.3.35 GetNumberOfStimulationElectrodes() `virtual uint32_t GetNumberOfStimulationElectrodes ( ) [virtual]`

11.117.3.36 GetNumberOfStimulationSourcesPerElectrode() `virtual uint32_t GetNumberOfStimulation↩ SourcesPerElectrode ( ) [virtual]`

Gets the number of stimulation sources (DACs) per electrode.

Returns

The number of stimulation sources (DACs) per electrode.

11.117.3.37 GetNumberOfSyncoutChannels() `virtual uint32_t GetNumberOfSyncoutChannels ( ) [virtual]`

Gets the Number of available syncout channels of the device.

Returns

The number of analog channels.

11.117.3.38 GetNumberOfTriggerInputs() `virtual uint32_t GetNumberOfTriggerInputs ( ) [virtual]`

Gets the Number of trigger inputs of the device.

Returns

The number of trigger inputs.

11.117.3.39 GetNumberOfVoltageRangeIndexes() `virtual uint32_t GetNumberOfVoltageRangeIndexes ( ) [virtual]`

Gets the number of voltage ranges (STG5 only).

Returns

The number of voltage ranges available on the device.

11.117.3.40 GetOutputRate() `uint32_t GetOutputRate ( )`

Queries the output rate of the STG. Valid rates are from 1000 Hz to 50000 Hz.

Returns

Returns the current output rate in Hz.

11.117.3.41 GetStgProgramInfo() `[1/2]` `void GetStgProgramInfo (`

```
[Out] bool% IsProgrammed,
[Out] DateTime% timestamp,
[Out] String^% filename,
[Out] Guid% guid )
```

Queries Download information from the STG. If download information was stored by the use of SetStgProgramInfo, this function can be used to retrieve it.

Parameters

IsProgrammed	Flag wether download information is valid.
timestamp	The timestamp of last download.
filename	The filename of the downlaoded waveform.
guid	A GUID.

11.117.3.42 GetStgProgramInfo() [2/2]

```
void GetStgProgramInfo (
    [Out] bool% IsProgrammed,
    [Out] System::Runtime::InteropServices::ComTypes::FILETIME% timestamp,
    [Out] String^% filename,
    [Out] Guid% guid )
```

Queries Download information from the STG. If download information was stored by the use of SetStgProgramInfo, this function can be used to retrieve it.

Parameters

IsProgrammed	Flag wether download information is valid.
timestamp	The timestamp of last download.
filename	The filename of the downlaoded waveform.

11.117.3.43 GetStgVersionInfo()

```
void GetStgVersionInfo (
    [Out] String^% SwVersion,
    [Out] String^% HwVersion )
```

Queries software and hardware version.

Parameters

SwVersion	The current Software Version of the STG.
HwVersion	The Hardware Revision of the STG.

11.117.3.44 GetSyncoutMap()

```
virtual uint32_t GetSyncoutMap (
    uint32_t channel )  [virtual]
```

Gets the mapping between external syncout outputs and internal syncout channels.

Parameters

channel	The external syncout output channel number (zero based).

Returns

The bitmap of internal syncout channels mapped to channel.

11.117.3.45 GetTimeResolutionInNanoSeconds()

```
virtual int32_t GetTimeResolutionInNanoSeconds (
)  [virtual]
```

Gets the time resolution of the STG.

Returns

The time resolution in nanoseconds.

11.117.3.46 GetTotalMemory() `uint32_t GetTotalMemory ( )`

Gets the total amount of memory available on the STG (all segments).

Returns

The total memory available on the STG in bytes.

11.117.3.47 GetTriggerSource() `virtual TriggerSourceEnumNet GetTriggerSource ( unsigned int triggernum ) [virtual]`

11.117.3.48 GetVoltageRangeInMicroVolt() `virtual int32_t GetVoltageRangeInMicroVolt ( uint32_t channel ) [virtual]`

Gets the Voltage Range of the specified channel in Microvolts.

Parameters

channel	Channel which is queried.

Returns

The Voltage Range of the specified channel in Microvolts.

11.117.3.49 GetVoltageRangeInMicroVoltByIndex() `virtual uint32_t GetVoltageRangeInMicroVoltByIndex ( uint32_t channel, uint32_t index ) [virtual]`

Gets the voltage range for the given channel and index (STG5 only).

Parameters

channel	The channel.
index	The channel.

Returns

The voltage range.

11.117.3.50 GetVoltageRangeListInMicroVolt()

```
virtual System::Collections::Generic::List<int32_t> ^ GetVoltageRangeListInMicroVolt (
            uint32_t channel ) [virtual]
```

Gets a list of current ranges supported by the device (STG5 only).

11.117.3.51 GetVoltageRangeSelectedIndex()

```
virtual uint32_t GetVoltageRangeSelectedIndex (
            uint32_t channel ) [virtual]
```

Gets the currently selected range index for the voltage output (not used yet).

Parameters

channel	The channel to change.

Returns

The currently selected range index.

11.117.3.52 GetVoltageResolutionInMicroVolt()

```
virtual int32_t GetVoltageResolutionInMicroVolt (
            uint32_t channel ) [virtual]
```

Gets the Voltage Resolution of the specified channel in Microvolts.

Parameters

channel	Channel which is queried.

Returns

The Voltage Resolution of the specified channel in Microvolts.

11.117.3.53 HasDigitalPort()

```
virtual bool HasDigitalPort ( ) [virtual]
```

Queries if the device has a digital port.

Returns

true if the device has a digital port.

11.117.3.54 HasFilterAmp() virtual bool HasFilterAmp ( ) [virtual]

Queries if the device has a filter amplifier. Currently the STG3008-FA supports this feature.

Returns

true if the device has a a filter amplifier.

11.117.3.55 LegacyHasOutputMap() virtual bool LegacyHasOutputMap ( ) [virtual]

Queries if the device can map its stimulation channels. Legacy command, do not use.

Returns

true if the device can map its stimulation channels.

11.117.3.56 ListModeSendStart() virtual void ListModeSendStart (unsigned int *electrodeGroupMask*) [virtual]

Activate (arm) the Listmode for the selected electrode groups.

Parameters

electrodeGroupMask	The bitmask of electrode groups for which the listmode is activated.

11.117.3.57 ListModeSendStop() virtual void ListModeSendStop (unsigned int *electrodeGroupMask*) [virtual]

Deactivate the Listmode for the selected electrode groups.

Parameters

electrodeGroupMask	The bitmask of electrodegroups for which the listmode is deactivated.

11.117.3.58 SendStart() void SendStart (uint32_t *triggermap*)

Start (Trigger) the STG. The startup delay is in the range of a few ms.

Parameters

triggermap	A bitmap of triggers which will be started.

11.117.3.59 SendStop() [1/2] void SendStop (uint32_t *triggermap*)

Stop some or all triggers of the STG.

Parameters

triggermap	A bitmap of triggers which will be stopped.

11.117.3.60 SendStop() [2/2] void SendStop (uint32_t *triggermap*, int *options*)

Stop some or all triggers of the STG.

Parameters

triggermap	A bitmap of triggers which will be stopped.
options	bitmap of options, currently only STOP_OPTION_SAVESTOP (0x80) is defined, which bypasses the stop commands when a syncout assossiated 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..

11.117.3.61 SetAutocalibrationDisabled() virtual void SetAutocalibrationDisabled (unsigned int *channel*, bool *disable*) [virtual]

Sets the autocalibration configuration.

Parameters

channel	The channel number.
disable	`true` if autocalibration is to be disabled.

11.117.3.62 SetBlankingEnable() [1/4] virtual void SetBlankingEnable (uint32_t *electrode*, array< bool >^ *enable*) [virtual]

Defines whether an electrode should be blanked while stimulation is in progress.

Parameters

electrode	The electrode number.
enable	True if the switch is to be opened, false if it is to remain closed while stimulation is in progress.

11.117.3.63 SetBlankingEnable() [2/4]

```
virtual void SetBlankingEnable (
    uint32_t electrode,
    bool enable ) [virtual]
```

Defines whether an electrode should be blanked while stimulation is in progress.

Parameters

electrode	The electrode number.
enable	True if the switch is to be opened, false if it is to remain closed while stimulation is in progress.

11.117.3.64 SetBlankingEnable() [3/4]

```
virtual void SetBlankingEnable (
    uint32_t Scu_HS,
    uint32_t electrode,
    array< bool >^ enable ) [virtual]
```

Defines whether an electrode should be blanked while stimulation is in progress.

Parameters

Scu_HS	The SCU headstage number.

Parameters

electrode	The electrode number.
enable	True if the switch is to be opened, false if it is to remain closed while stimulation is in progress.

11.117.3.65 SetBlankingEnable() [4/4]

```
virtual void SetBlankingEnable (
    uint32_t Scu_HS,
    uint32_t electrode,
    bool enable ) [virtual]
```

Defines whether an electrode should be blanked while stimulation is in progress.

Parameters

Scu_HS	The SCU headstage number.

Parameters

electrode	The electrode number.
enable	True if the switch is to be opened, false if it is to remain closed while stimulation is in progress.

11.117.3.66 SetCurrentMode() [1/2] virtual void SetCurrentMode ( ) [virtual]

Sets all channels to current mode (STG3008-FA and STG400x only).

11.117.3.67 SetCurrentMode() [2/2] virtual void SetCurrentMode (unsigned int *channel*) [virtual]

Sets a channel to current mode (STG3008-FA and STG400x only).

Parameters

channel	The channel to change.

11.117.3.68 SetCurrentRangeSelectedIndex() virtual void SetCurrentRangeSelectedIndex (uint32_t *channel*, uint32_t *rangeIndex*) [virtual]

Sets the range index for the current output (STG5 only).

Parameters

channel	The channel to change.
rangeIndex	The new range index.

11.117.3.69 SetDacAmplificationFactor() virtual void SetDacAmplificationFactor (uint32_t *DacNumber*, double *Factor*) [virtual]

Set the amplification factor for a DAC.

Parameters

DacNumber	The number of the DAC.
Factor	the amplifcation factor for that DAC, range is from -1.99999 to +1.99999.

11.117.3.70 SetDigoutMode() `virtual void SetDigoutMode (`

`Stg200xDigoutModeEnumNet` *digoutMode*) `[virtual]`

Sets the operation mode of the digital outport port, can be Monitor, Manual or SyncOut

Parameters

digoutMode	The new operation mode.

11.117.3.71 SetDigoutValue() `virtual void SetDigoutValue (`

`uint32_t` *digoutValue*) `[virtual]`

Sets the Value on the digital output port when in manual mode.

Parameters

digoutValue	The new value on the digital outputs.

11.117.3.72 SetElectrodeDacMux() `[1/4]` `virtual void SetElectrodeDacMux (`

`uint32_t` *electrode*,
`uint32_t` *listmodeIndex*,
`array< ElectrodeDacMuxEnumNet >^` *dacMux*) `[virtual]`

Defines the DAC to use for an electrode.

Parameters

electrode	The electrode number.

Parameters

listmodeIndex	The index for listmode.
dacMux	The DAC to use, can be ElectrodeDacMuxEnumNet.Stg1 (1), ElectrodeDacMuxEnumNet.Stg2 (2) or ElectrodeDacMuxEnumNet.Stg3 (3). To ground an electrode, use ElectrodeDacMuxEnumNet.Ground (0).

11.117.3.73 SetElectrodeDacMux() [2/4] virtual void SetElectrodeDacMux (

```
uint32_t electrode,
uint32_t listmodeIndex,
ElectrodeDacMuxEnumNet dacMux ) [virtual]
```

Defines the DAC to use for an electrode.

Parameters

electrode	The electrode number.

Parameters

listmodeIndex	The index for listmode.
dacMux	The DAC to use, can be ElectrodeDacMuxEnumNet.Stg1 (1), ElectrodeDacMuxEnumNet.Stg2 (2) or ElectrodeDacMuxEnumNet.Stg3 (3). To ground an electrode, use ElectrodeDacMuxEnumNet.Ground (0).

11.117.3.74 SetElectrodeDacMux() [3/4] virtual void SetElectrodeDacMux (

```
uint32_t Scu_HS,
uint32_t electrode,
uint32_t listmodeIndex,
array< ElectrodeDacMuxEnumNet >^ dacMux ) [virtual]
```

Defines the DAC to use for an electrode.

Parameters

Scu_HS	The SCU headstage number.

Parameters

electrode	The electrode number.

Parameters

listmodeIndex	The index for listmode.
dacMux	The DAC to use, can be ElectrodeDacMuxEnumNet.Stg1 (1), ElectrodeDacMuxEnumNet.Stg2 (2) or ElectrodeDacMuxEnumNet.Stg3 (3). To ground an electrode, use ElectrodeDacMuxEnumNet.Ground (0).

11.117.3.75 SetElectrodeDacMux() [4/4]

```
virtual void SetElectrodeDacMux (
    uint32_t Scu_HS,
    uint32_t electrode,
    uint32_t listmodeIndex,
    ElectrodeDacMuxEnumNet dacMux ) [virtual]
```

Defines the DAC to use for an electrode.

Parameters

Scu_HS	The SCU headstage number.

Parameters

electrode	The electrode number.

Parameters

listmodeIndex	The index for listmode.
dacMux	The DAC to use, can be ElectrodeDacMuxEnumNet.Stg1 (1), ElectrodeDacMuxEnumNet.Stg2 (2) or ElectrodeDacMuxEnumNet.Stg3 (3). To ground an electrode, use ElectrodeDacMuxEnumNet.Ground (0).

11.117.3.76 SetElectrodeEnable() [1/4]

```
virtual void SetElectrodeEnable (
    uint32_t electrode,
    uint32_t listmodeIndex,
    array< bool >^ enable ) [virtual]
```

Enables or disables the stimulation switch for an electrode.

Parameters

electrode	The electrode number.

Parameters

listmodeIndex	The index for listmode.
enable	1 to enable the electrode, 0 to disable.

11.117.3.77 SetElectrodeEnable() [2/4]

```
virtual void SetElectrodeEnable (
    uint32_t electrode,
    uint32_t listmodeIndex,
    bool enable )  [virtual]
```

Enables or disables the stimulation switch for an electrode.

Parameters

electrode	The electrode number.

Parameters

listmodeIndex	The index for listmode.
enable	1 to enable the electrode, 0 to disable.

11.117.3.78 SetElectrodeEnable() [3/4]

```
virtual void SetElectrodeEnable (
    uint32_t Scu_HS,
    uint32_t electrode,
    uint32_t listmodeIndex,
    array< bool >^ enable )  [virtual]
```

Enables or disables the stimulation switch for an electrode.

Parameters

Scu_HS	The SCU headstage number.

Parameters

electrode	The electrode number.

Parameters

listmodeIndex	The index for listmode.
enable	1 to enable the electrode, 0 to disable.

11.117.3.79 SetElectrodeEnable() [4/4]

```
virtual void SetElectrodeEnable (
    uint32_t Scu_HS,
    uint32_t electrode,
    uint32_t listmodeIndex,
    bool enable ) [virtual]
```

Enables or disables the stimulation switch for an electrode.

Parameters

Scu_HS	The SCU headstage number.

Parameters

electrode	The electrode number.

Parameters

listmodeIndex	The index for listmode.
enable	1 to enable the electrode, 0 to disable.

11.117.3.80 SetElectrodeMode() [1/4]

```
virtual void SetElectrodeMode (
    uint32_t electrode,
    array< ElectrodeModeEnumNet >^ mode ) [virtual]
```

Puts an electrode in either automatic or manual mode.

Parameters

electrode	The electrode number.

Returns

0 for automatic and 3 for manual mode.

11.117.3.81 SetElectrodeMode() [2/4]

```
virtual void SetElectrodeMode (
    uint32_t electrode,
    ElectrodeModeEnumNet mode ) [virtual]
```

Puts an electrode in either automatic or manual mode.

Parameters

electrode	The electrode number.

Parameters

mode	0 for automatic and 3 for manual mode.

11.117.3.82 SetElectrodeMode() [3/4]

```
virtual void SetElectrodeMode (
    uint32_t Scu_HS,
    uint32_t electrode,
    array< ElectrodeModeEnumNet >^ mode ) [virtual]
```

Puts an electrode in either automatic or manual mode.

Parameters

Scu_HS	The SCU headstage number.

Parameters

electrode	The electrode number.

Returns

0 for automatic and 3 for manual mode.

11.117.3.83 SetElectrodeMode() [4/4]

```
virtual void SetElectrodeMode (
            uint32_t Scu_HS,
            uint32_t electrode,
            ElectrodeModeEnumNet mode ) [virtual]
```

Puts an electrode in either automatic or manual mode.

Parameters

Scu_HS	The SCU headstage number.

Parameters

electrode	The electrode number.

Parameters

mode	0 for automatic and 3 for manual mode.

11.117.3.84 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

electrode	The electrode number.
enable	True if the switch is to be opened, false if it is to remain closed while stimulation is in progress.

11.117.3.85 SetEnableAmplifierProtectionSwitch() [2/4]

```
virtual void SetEnableAmplifierProtectionSwitch (
            uint32_t electrode,
            bool enable ) [virtual]
```

Defines whether the Amplifier Protection Switch is openend while stimulation is in progress.

Parameters

electrode	The electrode number.
enable	True if the switch is to be opened, false if it is to remain closed while stimulation is in progress.

11.117.3.86 SetEnableAmplifierProtectionSwitch() [3/4]

```
virtual void SetEnableAmplifierProtectionSwitch (
            uint32_t Scu_HS,
            uint32_t electrode,
            array< bool >^ enable ) [virtual]
```

Defines whether the Amplifier Protection Switch is openend while stimulation is in progress.

Parameters

Scu_HS	The SCU headstage number.

Parameters

electrode	The electrode number.
enable	True if the switch is to be opened, false if it is to remain closed while stimulation is in progress.

11.117.3.87 SetEnableAmplifierProtectionSwitch() [4/4]

```
virtual void SetEnableAmplifierProtectionSwitch (
            uint32_t Scu_HS,
            uint32_t electrode,
            bool enable ) [virtual]
```

Defines whether the Amplifier Protection Switch is openend while stimulation is in progress.

Parameters

Scu_HS	The SCU headstage number.

Parameters

electrode	The electrode number.
enable	True if the switch is to be opened, false if it is to remain closed while stimulation is in progress.

11.117.3.88 SetExternalElectrodeEnable() [1/2]

```
virtual void SetExternalElectrodeEnable (
    uint32_t electrode,
    uint32_t listmodeIndex,
    array< bool >^ enable ) [virtual]
```

Enables or disables the stimulation switch for an external electrode.

Parameters

electrode	The electrode number.

Parameters

listmodeIndex	The index for listmode.
enable	1 to enable the electrode, 0 to disable.

11.117.3.89 SetExternalElectrodeEnable() [2/2]

```
virtual void SetExternalElectrodeEnable (
    uint32_t electrode,
    uint32_t listmodeIndex,
    bool enable ) [virtual]
```

Enables or disables the stimulation switch for an external electrode.

Parameters

electrode	The electrode number.

Parameters

listmodeIndex	The index for listmode.
enable	1 to enable the electrode, 0 to disable.

11.117.3.90 SetFAAmplification() virtual void SetFAAmplification (Stg3008FilterAmpAmplificationEnumNet *amplification*) [virtual]

Sets the amplification of the filter amplifier (STG3008-FA).

11.117.3.91 SetHeadstage() virtual void SetHeadstage (unsigned int *headstage*) [virtual]

11.117.3.92 SetListmodeIndexRange() virtual void SetListmodeIndexRange (unsigned int *electrodeGroup*, unsigned int *startIndex*, unsigned int *endIndex*, unsigned int *mode*) [virtual]

Define the range of list mode indexes to use for the given electrode group.

Parameters

electrodeGroup	The electrodegroup for which the range is defined.

Parameters

startIndex	The index of the first active element in the listmode list.
endIndex	The index of the last active element in the listmode list.
mode	0 for "start with startIndex", 1 for "start with endIndex".

11.117.3.93 SetListmodeTriggerSource() [1/2] virtual void SetListmodeTriggerSource (unsigned int *electrodeGroup*, TriggerSourceEnumNet *triggersource*) [virtual]

Define the signal which triggers the transition from one list mode entry to the next. After reaching the last entry in the list, the first entry is selected. For triggersource use the Enum which corresponds to the device in use, for example use *SCUDigitalSourceEnumNet* and cast to *TriggerSourceEnumNet* if working with an SCU device.

Parameters

electrodeGroup	The electrodegroup for which the triggersource is defined.

Parameters

triggersource	One of the possible sources for the transition.

11.117.3.94 SetListmodeTriggerSource() [2/2]

```
virtual void SetListmodeTriggerSource (
    unsigned int electrodeGroup,
    TriggerSourceEnumNet triggersource,
    int bitnumOffset ) [virtual]
```

Define the signal which triggers the transition from one list mode entry to the next. After reaching the last entry in the list, the first entry is selected. For triggersource use the Enum which corresponds to the device in use, for example use *SCUDigitalSourceEnumNet* and cast to *TriggerSourceEnumNet* if working with an SCU device.

Parameters

electrodeGroup	The electrodegroup for which the triggersource is defined.

Parameters

triggersource	One of the possible sources for the transition.
bitnumOffset	Number to add to the numeric value of the *triggersource* enum.

11.117.3.95 SetMeasurementMode()

```
virtual void SetMeasurementMode (
    unsigned int channel ) [virtual]
```

Sets a channel to measurement mode (STG3008-FA).

Parameters

channel	The channel to change.

11.117.3.96 SetOutputRate()

```
void SetOutputRate (
    uint32_t rate )
```

Change the output rate of the STG. Valid rates are from 1000 Hz to 50000 Hz.

Parameters

rate	The new output rate in Hz.

11.117.3.97 SetStgProgramInfo()

```
void SetStgProgramInfo (
    DateTime timestamp,
    String^ filename,
    Guid guid )
```

Store Download information in the STG. This function can be used to store the filename and timestamp of the last download for later query.

Parameters

timestamp	The timestamp of last download.
filename	The filename of the downlaoded waveform.

11.117.3.98 SetSyncoutMap()

```
virtual void SetSyncoutMap (
    uint32_t channel,
    uint32_t syncoutMap )  [virtual]
```

Sets the mapping between external syncout outputs and internal syncout channels.

Parameters

channel	The external syncout output channel number (zero based).
syncoutMap	A bitmap of internal syncout channels to map to channel.

11.117.3.99 SetTriggerSource() [1/3]

```
virtual void SetTriggerSource (
    unsigned int triggernum,
    DigitalSource< digitalsourceenum >^ triggersource,
    int bitnum_offset )  [virtual]
```

11.117.3.100 SetTriggerSource() [2/3]

```
virtual void SetTriggerSource (
    unsigned int triggernum,
    TriggerSourceEnumNet triggersource )  [virtual]
```

11.117.3.101 SetTriggerSource() [3/3]

```
virtual void SetTriggerSource (
    unsigned int triggernum,
    TriggerSourceEnumNet triggersource,
    int bitnum_offset )  [virtual]
```

11.117.3.102 SetVoltageMode() [1/2] virtual void SetVoltageMode ( ) [virtual]

Sets all channels to voltage mode (STG3008-FA and STG400x only).

11.117.3.103 SetVoltageMode() [2/2] virtual void SetVoltageMode (unsigned int *channel*) [virtual]

Sets a channel to voltage mode (STG3008-FA and STG400x only).

Parameters

channel	The channel to change.

11.117.3.104 SetVoltageRangeSelectedIndex() virtual void SetVoltageRangeSelectedIndex (uint32_t *channel*, uint32_t *rangeIndex*) [virtual]

Sets the range index for the voltage output (not used yet).

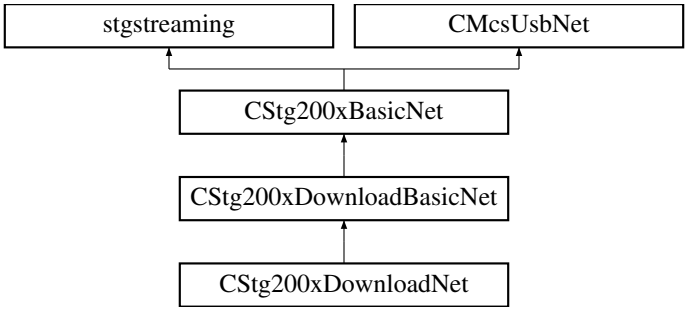
Parameters

channel	The channel to change.
rangeIndex	The new range index.

11.118 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 µs. The STG has a resolution of 20 µs. 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 µs. The STG has a resolution of 20 µs.
- 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^ Stimulus [get]

Additional Inherited Members

11.118.1 Detailed Description

CStg200xDownloadBasicNet is the base class to control the download mode of the MCS STG device.

11.118.2 Member Function Documentation

11.118.2.1 ClearChannelData() virtual void ClearChannelData (uint32_t *channel*) [virtual]

Delete a stimulus pattern for a channel from STG memory

Parameters

channel	Specifies the channel to clear.

11.118.2.2 ClearSyncData() virtual void ClearSyncData (uint32_t *channel*) [virtual]

Delete a SyncOut pattern for a channel from STG memory.

Parameters

channel	Specifies the syncout channel to clear.

11.118.2.3 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()

11.118.2.4 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.

11.118.2.5 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.

11.118.2.6 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

Channel	channel for the amount of interested usage.

Returns

Returns the usage in STG memory.

11.118.2.7 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

Channel	channel for the amount of interested usage.

Returns

Returns the usage in STG memory.

11.118.2.8 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

sweeps	on return contains the number of sweeps for each trigger.
triggers	on return contains the number of trigger events seen for each trigger.

11.118.2.9 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

channelmap	For each trigger, a bitmap of channels that belong to this trigger.

Parameters

syncoutmap	For each trigger, a bitmap of syncouts that belong to this trigger.
repeat	For each trigger, define the number of times this trigger should be repeated.

11.118.2.10 ResetStatus()

```
void ResetStatus (
    uint32_t triggermap )
```

Reset the status flag.

Parameters

triggermap	bitmap of trigger for which to reset the status.

11.118.2.11 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 µs. The STG has a resolution of 20 µs.

Parameters

channel	Specifies the channel to append the data to.
pData	A list of datapoints.
tData	A list of durations as int64_t. The time is given in units of µs.

11.118.2.12 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 µs. The STG has a resolution of 20 µs. If your application can not handle 64 bit integers, use the STG200x_SendSyncData32() call instead.

Parameters

channel	Specifies the sync output channel to append the data to.
pData	A list of datapoints.
tData	A list of durations as int64_t. The time is given in units of µs.

11.118.2.13 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

trigger	The trigger to change.
same_trigger	Action for successive triggers in Normal Mode, and for triggers to the currently selected segment in Multi - File Mode.
other_trigger	Action for successive triggers in Multi-File Mode for a trigger on a segment not currently selected.Not used in Normal Mode.

11.118.2.14 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

same_trigger	Action for successive triggers in Normal Mode, and for triggers to the currently selected segment in Multi - File Mode.
other_trigger	Action for successive triggers in Multi-File Mode for a trigger on a segment not currently selected.Not used in Normal Mode.

11.118.2.15 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

first_trigger	The number of the first trigger to change.

Parameters

channelmap	For each trigger, a bitmap of channels that belong to this trigger.

Parameters

syncoutmap	For each trigger, a bitmap of syncouts that belong to this trigger.
repeat	For each trigger, define the number of times this trigger should be repeated.

11.118.2.16 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

trigger	The trigger to change.

Parameters

channelmap	A bitmap of channels that belong to this trigger.

Parameters

syncoutmap	A bitmap of syncouts that belong to this trigger.
repeat	The number of times this trigger should be repeated.

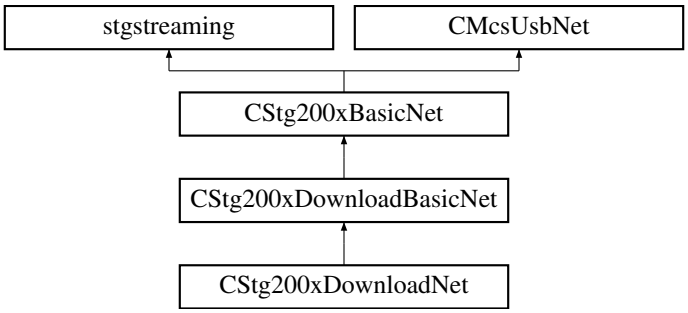
11.118.3 Property Documentation

11.118.3.1 Stimulus `CStimulusFunctionNet^ Stimulus [get]`

11.119 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 SegmentDefine (array< uint32_t >^ segment_list)

  Defines the segment memory layout of the STG.
- void SegmentStart (uint32_t triggermap, uint32_t segment, Stg200xSegmentFlagsEnumNet segmentflags)

  Switchs segment and starts trigger.
- void SegmentSelect (uint32_t segment, Stg200xSegmentFlagsEnumNet segmentflags)

  Switchs segment.
- void EnableMultiFileMode (Stg200xMultiFileSubmodeEnumNet submode)

Enable the Multi-File mode of the STG.

- void DisableMultiFileMode ()

  Disable the Multi-File mode of the STG

- StgStatusNet ^ QueryTriggerstatus ()
- void SetOutputMap (array< uint32_t >^ ChannelLayout)
- int32_t GetModuleTemp (unsigned int channel)
- uint32_t GetModuleCurrent (unsigned int channel)

Events

- OnStgPollStatus^ Stg200xPollStatusEvent [add, remove, raise]
- OnMwPollStatus^ MwPollStatusEvent [add, remove, raise]

Additional Inherited Members

11.119.1 Detailed Description

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

11.119.2 Constructor & Destructor Documentation

11.119.2.1 CStg200xDownloadNet() [1/2] CStg200xDownloadNet ( )

Use this constructor if you do not want to use the status callback.

11.119.2.2 CStg200xDownloadNet() [2/2] CStg200xDownloadNet (OnStgPollStatus^ *pollStatus*)

Use this constructor if you want to use the status callback.

11.119.2.3 ~CStg200xDownloadNet() ~CStg200xDownloadNet ( )

11.119.3 Member Function Documentation

11.119.3.1 ClearChannel_PrepareAndSendData()

```
void ClearChannel_PrepareAndSendData (
    uint32_t channel,
    array< int32_t >^ amplitude,
    array< uint64_t >^ duration,
    STG_DestinationEnumNet destType,
    bool doClear )
```

Prepare and append data to a given channel on the STG.

Each datapoint is represented by an signed 32bit integer value. When using voltage stimulation, the values are in multiple of 1 uV, thus the possible range is += 2000 V. When using current stimulation, the values are in multiple of 1 nA, this the possible range is += 2000 mA.

The duration is given as a list of 64 bit integers. Durations are given in units of µs. The STG has a resolution of 20 µs.

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

channel	The channel number to send data to.

Parameters

amplitude	A list of amplitudes in units of µV and nA in voltage and current mode, respectively.

Parameters

duration	A list of durations in units of µs.
destType	specifies wheather the data is for syncout, current or voltage stimulation.

11.119.3.2 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.

11.119.3.3 EnableMultiFileMode() `void EnableMultiFileMode (`

`Stg200xMultiFileSubmodeEnumNet` *submode*)

Enable the Multi-File mode of the STG.

In Multi-File mode, the trigger inputs switch between segments. To use this mode, define up to as many segments as trigger inputs are available and fill each segment with a stimulus pattern.

Now a trigger on trigger input 1 switches the STG to the first segment and starts all triggers in this segment. Likewise, a trigger on trigger input 2, 3 and 4 selects the respective segment and start all triggers in this segment So the Multi-File Mode can be used to predefine up to four different stimuli which can be selected without the need for a computer connection.

Parameters

submode	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.

11.119.3.4 GetModuleCurrent() `uint32_t GetModuleCurrent (`

`unsigned int` *channel*)

11.119.3.5 GetModuleTemp() `int32_t GetModuleTemp (`

`unsigned int` *channel*)

11.119.3.6 PrepareAndAppendData() `void PrepareAndAppendData (`

`uint32_t` *channel*,
`array< int32_t >^` *amplitude*,
`array< uint64_t >^` *duration*,
`STG_DestinationEnumNet` *destType*)

Prepare and append data to a given channel on the STG.

Each datapoint is represented by an signed 32bit integer value. When using voltage stimulation, the values are in multiple of 1 uV, thus the possible range is += 2000 V. When using current stimulation, the values are in multiple of 1 nA, this the possible range is += 2000 mA.

The duration is given as a list of 64 bit integers. Durations are given in units of µs. The STG has a resolution of 20 µs.

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

channel	The channel number to send data to.

Parameters

amplitude	A list of amplitudes in units of µV and nA in voltage and current mode, respectively.

Parameters

duration	A list of durations in units of µs.
destType	specifies wheather the data is for syncout, current or voltage stimulation.

11.119.3.7 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 uV, thus the possible range is += 2000 V. When using current stimulation, the values are in multiple of 1 nA, this the possible range is += 2000 mA.

The duration is given as a list of 64 bit integers. Durations are given in units of µs. The STG has a resolution of 20 µs.

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

channel	The channel number to send data to.

Parameters

amplitude	A list of amplitudes in units of µV and nA in voltage and current mode, respectively.

Parameters

duration	A list of durations in units of µs.
destType	specifies wheather the data is for syncout, current or voltage stimulation.

11.119.3.8 QueryTriggerstatus() `StgStatusNet ^ QueryTriggerstatus ( )`

11.119.3.9 SegmentDefine() `void SegmentDefine ( array< uint32_t >^ segment_list )`

Defines the segment memory layout of the STG.

On reset, the STG has one segment containing all available memory.

With this command, the STG memory can be devided into several segments. Each segment can be filled with stimulus data.

Parameters

segment_list	The List of memory sizes (one per segment).

11.119.3.10 SegmentSelect() `void SegmentSelect ( uint32_t segment, Stg200xSegmentFlagsEnumNet segmentflags )`

Switchs segment.

Parameters

segment	The number of the segment to select.

Parameters

segmentflags	A bitmap of flags, bit 1: assign all channels to the trigger number equal to the segment.

11.119.3.11 SegmentStart() void SegmentStart (uint32_t *triggermap*, uint32_t *segment*, Stg200xSegmentFlagsEnumNet *segmentflags*)

Switchs segment and starts trigger.

Parameters

triggermap	A bitmap of triggers that will be started.

Parameters

segment	The number of the segment to select.

Parameters

segmentflags	A bitmap of flags, bit 1: assign all channels to the trigger number equal to the segment.

11.119.3.12 SetOutputMap() void SetOutputMap (array< uint32_t >^ *ChannelLayout*)

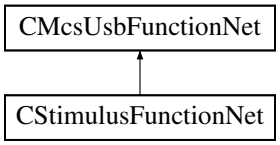
11.119.4 Event Documentation

11.119.4.1 MwPollStatusEvent OnMwPollStatus^ MwPollStatusEvent [add], [remove], [raise]

11.119.4.2 Stg200xPollStatusEvent OnStgPollStatus^ Stg200xPollStatusEvent [add], [remove], [raise]

11.120 CStimulusFunctionNet Class Reference

Inheritance diagram for CStimulusFunctionNet:



Classes

- class SidebandData
- class StimulusDeviceDataAndUnrolledData

Public Member Functions

- CStimulusFunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ stimulusFunction↩ PointerContainer)
- CStimulusFunctionNet (CMcsUsbNet^ 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 >^ 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)
- 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.

- int GetCurrentResolutionInPicoAmp (uint32_t channel)

  Gets the Current Resolution of the specified channel in Picoamps.

- void SetupTrigger (uint32_t first_trigger, array< uint32_t >^ channelmap, array< uint32_t >^ syncoutmap, array< uint32_t >^ repeat)

  Configures the trigger settings for the STG. Note that all memory segments have their own trigger setting.

- void SetupTriggerSingle (uint32_t trigger, uint32_t channelmap, uint32_t syncoutmap, uint32_t repeat)

  Configures the trigger settings for the STG. Note that all memory segments have their own trigger setting.

- uint32_t GetTotalMemory ()

  Get the total amount of memory available on the STG (all segments).

- uint32_t GetAvailableMemory ()

  Get the amount of memory available in the currently selected segment of the STG.

- int GetNumberOfAnalogChannels ()

  Get the number of STG channels.

Events

- OnStgPollStatus^ PollStatusEvent

Additional Inherited Members

11.120.1 Constructor & Destructor Documentation

11.120.1.1 CStimulusFunctionNet() [1/2]

```
CStimulusFunctionNet (
    CMcsUsbNet^ mcsusb,
    CMcsUsbFunctionPointerContainer^ stimulusFunctionPointerContainer )
```

11.120.1.2 CStimulusFunctionNet() [2/2]

```
CStimulusFunctionNet (
    CMcsUsbNet^ mcsusb )
```

11.120.2 Member Function Documentation

11.120.2.1 ClearChannel_PrepareAndSendData()

```
void ClearChannel_PrepareAndSendData (
    uint32_t channel,
    array< int32_t >^ amplitude,
    array< uint64_t >^ duration,
    STG_DestinationEnumNet destType,
    bool doClear )
```

11.120.2.2 ClearChannelData()

```
void ClearChannelData (
    int channel )
```

Delete a Stimulus Pattern from STG memory

Parameters

channel	specifies the channel to clear.

11.120.2.3 ClearMultiplexedData()

```
void ClearMultiplexedData ( )
```

Clears the Stimulation Memory in the STG device.

11.120.2.4 ClearSyncData()

```
void ClearSyncData (
    int channel )
```

Delete a Syncout Pattern from STG memory

Parameters

channel	specifies the channel to clear.

11.120.2.5 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 µs. The STG has a resolution of 20 µs.

Parameters

StimulusActive	A list of datapoints which define weather the Stimulus is active or idle at that time as int32.

Parameters

Duration	A list of durations as uint64. The time is given in units of µs.
Bit0Time	Time in µs for which Bit 0 (Blanking) is to be extended.

Parameters

Bit3Time	Time in µs for which Bit 3 (Stimulus Enable) is to be extended.

Parameters

Bit4Time	Time in µs for which Bit 4 (Stimulus Selector) is to be extended.

Returns

Error Status. 0 on success.

11.120.2.6 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.

11.120.2.7 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.

11.120.2.8 GetCurrentRangeInNanoAmp() `int GetCurrentRangeInNanoAmp ( uint32_t channel )`

Gets the Current Range of the specified channel in Nanoamps.

Parameters

channel	Channel which is queried.

Returns

The Current Range of the specified channel in Nanoamps.

11.120.2.9 GetCurrentResolutionInNanoAmp() `int GetCurrentResolutionInNanoAmp ( uint32_t channel )`

Gets the Current Resolution of the specified channel in Nanoamps.

Parameters

channel	Channel which is queried.

Returns

The Current Resolution of the specified channel in Nanoamps.

11.120.2.10 GetCurrentResolutionInPicoAmp() `int GetCurrentResolutionInPicoAmp ( uint32_t channel )`

Gets the Current Resolution of the specified channel in Picoamps.

Parameters

channel	Channel which is queried.

Returns

The Current Resolution of the specified channel in Picoamps.

11.120.2.11 GetDACResolution() `int GetDACResolution ( )`

Gets number of bits of the DAC resolution.

Returns

The DAC resolution in bits.

11.120.2.12 GetMultiplexedDataChannelsInBlock() `int GetMultiplexedDataChannelsInBlock ( )`

Gets the number of stimulus data channels to send per time slice. Might be greater than the number of configured channels. Fill unused channels with dummy data in SendMultiplexedData

11.120.2.13 GetNumberOfAnalogChannels() `int GetNumberOfAnalogChannels ( )`

Get the number of STG channels.

Returns

The number of STG channels.

11.120.2.14 GetTotalMemory() `uint32_t GetTotalMemory ( )`

Get the total amount of memory available on the STG (all segments).

Returns

The total memory available on the STG in bytes.

11.120.2.15 GetVoltageRangeInMicroVolt() `int GetVoltageRangeInMicroVolt ( uint32_t channel )`

Gets the Voltage Range of the specified channel in Microvolts.

Parameters

channel	Channel which is queried.

Returns

The Voltage Range of the specified channel in Microvolts.

11.120.2.16 GetVoltageResolutionInMicroVolt()

```
int GetVoltageResolutionInMicroVolt (
    uint32_t channel )
```

Gets the Voltage Resolution of the specified channel in Microvolts.

Parameters

channel	Channel which is queried.

Returns

The Voltage Resolution of the specified channel in Microvolts.

11.120.2.17 PrepareAndAppendData()

```
void PrepareAndAppendData (
    uint32_t channel,
    array< int32_t >^ amplitude,
    array< uint64_t >^ duration,
    STG_DestinationEnumNet destType )
```

Prepare and append data to a given channel on the STG.

Each datapoint is represented by an signed 32bit integer value. When using voltage stimulation, the values are in multiple of 1 uV, thus the possible range is += 2000 V. When using current stimulation, the values are in multiple of 1 nA, this the possible range is += 2000 mA.

The duration is given as a list of 64 bit integers. Durations are given in units of µs. The STG has a resolution of 20 µs.

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

channel	The channel number to send data to.

Parameters

amplitude	A list of amplitudes in units of µV and nA in voltage and current mode, respectively.

Parameters

duration	A list of durations in units of µs.
destType	specifies wheather the data is for syncout, current or voltage stimulation.

Returns

Error Status. 0 on success.

11.120.2.18 PrepareAndSendData()

```
void PrepareAndSendData (
    uint32_t channel,
    array< int32_t >^ amplitude,
    array< uint64_t >^ duration,
    STG_DestinationEnumNet destType )
```

Prepare and send data to a given channel on the STG. Previous data sent to that channel is erased first.

Each datapoint is represented by an signed 32bit integer value. When using voltage stimulation, the values are in multiple of 1 uV, thus the possible range is += 2000 V. When using current stimulation, the values are in multiple of 1 nA, this the possible range is += 2000 mA.

The duration is given as a list of 64 bit integers. Durations are given in units of µs. The STG has a resolution of 20 µs.

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

channel	The channel number to send data to.

Parameters

amplitude	A list of amplitudes in units of µV and nA in voltage and current mode, respectively.

Parameters

duration	A list of durations in units of µs.
destType	specifies wheather the data is for syncout, current or voltage stimulation.

Returns

Error Status. 0 on success.

11.120.2.19 PrepareData()

```
StimulusDeviceDataAndUnrolledData ^ PrepareData (
    int channel,
    array< int32_t >^ amplitude,
    array< uint64_t >^ duration,
    STG_DestinationEnumNet destType )
```

11.120.2.20 SendMultiplexedData()

```
void SendMultiplexedData (
    array< uint16_t >^ data )
```

Sends stimulus data in multiplexed form. All 16 bits words for the enabled DAC and digital channels are muxed together per time slice.

Parameters

data	Array of data to be sent.

11.120.2.21 SendPreparedData()

```
void SendPreparedData (
    int channel,
    StimulusDeviceDataAndUnrolledData^ device_data_and_unrolled,
    STG_DestinationEnumNet destType )
```

11.120.2.22 SendStart()

```
void SendStart (
    uint32_t triggermap )
```

Start (Trigger) the STG. The startup delay is in the range of a few ms.

Parameters

triggermap	A bitmap of triggers which will be started.

11.120.2.23 SendStop() [1/2] void SendStop (uint32_t *triggermap*)

Stop some or all triggers of the STG.

Parameters

triggermap	A bitmap of triggers which will be stopped.

11.120.2.24 SendStop() [2/2] void SendStop (uint32_t *triggermap*, int *options*)

Stop some or all triggers of the STG.

Parameters

triggermap	A bitmap of triggers which will be stopped.
options	bitmap of options, currently only STOP_OPTION_SAVESTOP (0x80) is defined, which bypasses the stop commands when a syncout assossiated 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..

11.120.2.25 SetupTrigger() void SetupTrigger (uint32_t *first_trigger*, array< uint32_t >^ *channelmap*, array< uint32_t >^ *syncoutmap*, array< uint32_t >^ *repeat*)

Configures the trigger settings for the STG. Note that all memory segments have their own trigger setting.

Parameters

first_trigger	The number of the first trigger to change.

Parameters

channelmap	For each trigger, a bitmap of channels that belong to this trigger.

Parameters

syncoutmap	For each trigger, a bitmap of syncouts that belong to this trigger.
repeat	For each trigger, define the number of times this trigger should be repeated.

11.120.2.26 SetupTriggerSingle() `void SetupTriggerSingle (`

```
uint32_t trigger,
uint32_t channelmap,
uint32_t syncoutmap,
uint32_t repeat )
```

Configures the trigger settings for the STG. Note that all memory segments have their own trigger setting.

Parameters

trigger	The trigger to change.

Parameters

channelmap	A bitmap of channels that belong to this trigger.

Parameters

syncoutmap	A bitmap of syncouts that belong to this trigger.
repeat	The number of times this trigger should be repeated.

11.120.2.27 StartPoll() `void StartPoll ( )`

Starts the interrupt fetching thread and delivers events

11.120.2.28 StopPoll() `void StopPoll ( )`

Stops the interrupt fetching thread and delivers events

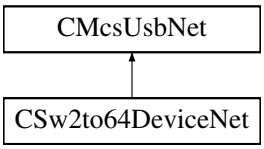
11.120.3 Event Documentation

11.120.3.1 PollStatusEvent `OnStgPollStatus^ PollStatusEvent`

11.121 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

11.121.1 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.

11.121.2 Constructor & Destructor Documentation

11.121.2.1 CSw2to64DeviceNet() `CSw2to64DeviceNet ( )`

11.121.2.2 ~CSw2to64DeviceNet() `~CSw2to64DeviceNet ( )`

11.121.3 Member Function Documentation

11.121.3.1 GetChannel() `unsigned char GetChannel ( unsigned short index )`

Gets one current switch position.

Parameters

in	*index*	number of channel to read the switch position from

Returns

switch position of desired channel

11.121.3.2 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

11.121.3.3 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

11.121.3.4 SetChannel() `void SetChannel ( unsigned short index, unsigned char pattern )`

Sets one switch position.

Parameters

in	*index*	number of channel to write the switch position to
in	*pattern*	switch position of the channel

11.121.3.5 SetChannels() `void SetChannels ( array< unsigned char >^ pattern )`

Sets the switch positions from a char array.

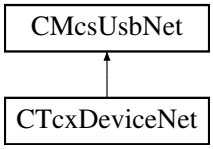
Parameters

in	*pattern*	array of char with the size of the number of channels, each char has the setting of a channel

11.122 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 ℃.
- int GetValueHires (unsigned int channel)

  Gets the temperate of the specified channel in units of 0.01 ℃.
- int GetHeaterTemp (unsigned int channel)

  Gets the temperate of the specified heater in units of 0.1 ℃.
- int GetHeaterLimit (unsigned int device)

  Gets the temperate limit of the specified heater in units of 0.1 ℃.
- double GetMaxHeaterPowerMultiwell ()

  queries the max. heater power that the Multiwell temperature controller will apply; unit: W; useful range: 5.2W..7.6W

- void SetMaxHeaterPowerMultiwell (double MaxPowerWatt)

  sets the max. heater power that the Multiwell temperature controller will apply; unit: W; useful range: 5.2W..7.6W
- bool GetHasThermocouple ()

  Gets weather the device supports a thermocouple.
- bool GetEnableHeaterLimit (unsigned int device)
- bool GetEnableThermocouple (unsigned int device)
- TcxSensorTypeEnumNet GetSensorType (unsigned int device)
- String ^ GetUnit (unsigned int channel)
- unsigned int GetBoardTemp ()

  Gets the temperate of the mainboard in units of 0.1 ℃.
- unsigned int GetVolti (unsigned int channel)
- unsigned int GetNumDevices ()
- void SetSetpoint (unsigned int channel, int sp)

  Sets the target temperate of specified channel in units of 0.1 ℃.
- void SetDevice (unsigned int channel, int device)
- void SetOnOff (unsigned int channel, bool on)

  Switches the specified channel on or off.
- void SetCalibration (unsigned int channel, int calib)
- void SetP (unsigned int device, int p_coeff)

  Sets the P-coefficient of the specified device.
- void SetI (unsigned int device, int i_coeff)

  Sets the I-coefficient of the specified device.
- void SetD (unsigned int device, int d_coeff)

  Sets the D-coefficient of the specified device.
- void SetMaxP (unsigned int device, int maxp)

  Sets the maximum heater power of the specified device.
- void SetHeaterLimit (unsigned int device, int heater_limit)
- void SetEnableHeaterLimit (unsigned int device, bool enable)
- void SetEnableThermocouple (unsigned int device, bool enable)
- void SetSensorType (unsigned int device, TcxSensorTypeEnumNet type)
- void SetDevname (unsigned int device, String^ Devicename)
- int GetSetpoint (unsigned int channel)

  Gets the target temperate of specified channel in units of 0.1 ℃.
- 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

11.122.1 Detailed Description

Class to control a Temperature Controller (TCX)

11.122.2 Constructor & Destructor Documentation

11.122.2.1 CTcxDeviceNet() CTcxDeviceNet ( )

Initializes a new instance of CTcxDeviceNet class.

11.122.2.2 ~CTcxDeviceNet() ~CTcxDeviceNet ( )

11.122.3 Member Function Documentation

11.122.3.1 CalibrateThermocouple() void CalibrateThermocouple (unsigned int *channel*)

11.122.3.2 FactoryReset() void FactoryReset ( )

11.122.3.3 GetBoardTemp() unsigned int GetBoardTemp ( )

Gets the temperate of the mainboard in units of 0.1 ℃.

11.122.3.4 GetCalibration() int GetCalibration (unsigned int *channel*)

11.122.3.5 GetCalibrationDecp() int GetCalibrationDecp (unsigned int *channel*)

11.122.3.6 GetCalibrationMax() int GetCalibrationMax (unsigned int *channel*)

11.122.3.7 GetCalibrationMin() int GetCalibrationMin (unsigned int *channel*)

11.122.3.8 GetCurrent() int GetCurrent (unsigned int *channel*)

11.122.3.9 GetD() int GetD (unsigned int *device*)

Gets the D-coefficient of the specified device.

11.122.3.10 GetDDecp() int GetDDecp (unsigned int *device*)

11.122.3.11 GetDevice() int GetDevice (unsigned int *channel*)

11.122.3.12 GetDeviceType() TcxDeviceTypeEnumNet GetDeviceType ( )

11.122.3.13 GetDevname() String ^ GetDevname (unsigned int *device*)

11.122.3.14 GetDMax() int GetDMax (unsigned int *device*)

11.122.3.15 GetDMin() int GetDMin (unsigned int *device*)

11.122.3.16 GetDuty() int GetDuty (unsigned int *channel*)

Gets the duty cycle of the heating element.

Parameters

channel	The channel number.

Returns

The duty cycle in percent, the value of 320 ∗ 64 corresponds to 100 %.

11.122.3.17 GetEnableHeaterLimit() `bool GetEnableHeaterLimit (`

`unsigned int` *device* `)`

11.122.3.18 GetEnableThermocouple() `bool GetEnableThermocouple (`

`unsigned int` *device* `)`

11.122.3.19 GetHasThermocouple() `bool GetHasThermocouple ( )`

Gets weather the device supports a thermocouple.

11.122.3.20 GetHeaterLimit() `int GetHeaterLimit (`

`unsigned int` *device* `)`

Gets the temperate limit of the specified heater in units of 0.1 ℃.

11.122.3.21 GetHeaterTemp() `int GetHeaterTemp (`

`unsigned int` *channel* `)`

Gets the temperate of the specified heater in units of 0.1 ℃.

11.122.3.22 GetI() `int GetI (`

`unsigned int` *device* `)`

Gets the I-coefficient of the specified device.

11.122.3.23 GetIDecp() `int GetIDecp ( unsigned int device )`

11.122.3.24 GetIMax() `int GetIMax ( unsigned int device )`

11.122.3.25 GetIMin() `int GetIMin ( unsigned int device )`

11.122.3.26 GetIOut() `int GetIOut ( unsigned int channel )`

Gets the current through the heating element.

Parameters

channel	The channel number.

Returns

The current in units of mA.

11.122.3.27 GetMaxHeaterPowerMultiwell() `double GetMaxHeaterPowerMultiwell ( )`

queries the max. heater power that the Multiwell temperature controller will apply; unit: W; useful range: 5.2W..7.6W

11.122.3.28 GetMaxP() `int GetMaxP ( unsigned int device )`

Gets the maximum heater power of the specified device.

11.122.3.29 GetMaxpDecp() `int GetMaxpDecp ( unsigned int device )`

11.122.3.30 GetMaxpMax() int GetMaxpMax (unsigned int *device*)

11.122.3.31 GetMaxpMin() int GetMaxpMin (unsigned int *device*)

11.122.3.32 GetNumControlChannels() unsigned int GetNumControlChannels ( )

Gets the number of channels the device can control/regulate.

11.122.3.33 GetNumDevices() unsigned int GetNumDevices ( )

11.122.3.34 GetNumMeasureChannels() unsigned int GetNumMeasureChannels ( )

Gets the number of channels the device can measure.

11.122.3.35 GetOnOff() int GetOnOff (unsigned int *channel*)

Gets if the specified channel is on or off.

11.122.3.36 GetP() int GetP (unsigned int *device*)

Gets the P-coefficient of the specified device.

11.122.3.37 GetPDecp() int GetPDecp (unsigned int *device*)

11.122.3.38 GetPMax() int GetPMax (unsigned int *device*)

11.122.3.39 GetPMin() int GetPMin (unsigned int *device*)

11.122.3.40 GetPOut() int GetPOut (unsigned int *channel*)

Gets the output power of the heating element.

Parameters

channel	The channel number.

Returns

The resistance in units of mW.

11.122.3.41 GetPwrOut() `int GetPwrOut (`

`unsigned int` *channel* `)`

11.122.3.42 GetPwrSet() `int GetPwrSet (`

`unsigned int` *channel* `)`

11.122.3.43 GetRes1() `int GetRes1 (`

`unsigned int` *channel* `)`

11.122.3.44 GetRes2() `int GetRes2 (`

`unsigned int` *channel* `)`

11.122.3.45 GetResS() `int GetResS (`

`unsigned int` *channel* `)`

11.122.3.46 GetResX() `int GetResX (`

`unsigned int` *channel* `)`

11.122.3.47 GetROut() `int GetROut (`

`unsigned int` *channel* `)`

Gets the resistance of the heating element.

Parameters

channel	The channel number.

Returns

The resistance in units of 0.1 Ohm.

11.122.3.48 GetSensorType() `TcxSensorTypeEnumNet GetSensorType (`

`unsigned int device )`

11.122.3.49 GetSetpoint() `int GetSetpoint (`

`unsigned int channel )`

Gets the target temperate of specified channel in units of 0.1 ℃.

11.122.3.50 GetSetpointDecp() `int GetSetpointDecp (`

`unsigned int channel )`

11.122.3.51 GetSetpointMax() `int GetSetpointMax (`

`unsigned int channel )`

11.122.3.52 GetSetpointMin() `int GetSetpointMin (`

`unsigned int channel )`

11.122.3.53 GetThermocoupleCalibration() `int GetThermocoupleCalibration (`

`unsigned int channel )`

11.122.3.54 GetThermocoupleNanovoltPerKelvin() `unsigned int GetThermocoupleNanovoltPerKelvin (`

`unsigned int channel )`

Gets the proportional constant for the thermocouple.

Parameters

channel	Thermocouple channel number.

Returns

The proportional constant in Nanovolt per Kelvin.

11.122.3.55 GetThermocoupleReferenceTemp() `int GetThermocoupleReferenceTemp ( unsigned int channel )`

11.122.3.56 GetThermocoupleTemp() `int GetThermocoupleTemp ( unsigned int channel )`

11.122.3.57 GetThermocoupleTempAbs() `int GetThermocoupleTempAbs ( unsigned int channel )`

11.122.3.58 GetUnit() `String ^ GetUnit ( unsigned int channel )`

11.122.3.59 GetUOut() `int GetUOut ( unsigned int channel )`

Gets the voltage on the heating element.

Parameters

channel	The channel number.

Returns

The voltage in units of mV.

11.122.3.60 GetValue()

```
int GetValue (
    unsigned int channel )
```

Gets the temperate of the specified channel in units of 0.1℃.

11.122.3.61 GetValueHires()

```
int GetValueHires (
    unsigned int channel )
```

Gets the temperate of the specified channel in units of 0.01℃.

11.122.3.62 GetVolti()

```
unsigned int GetVolti (
    unsigned int channel )
```

11.122.3.63 SetCalibration()

```
void SetCalibration (
    unsigned int channel,
    int calib )
```

11.122.3.64 SetD()

```
void SetD (
    unsigned int device,
    int d_coeff )
```

Sets the D-coefficient of the specified device.

11.122.3.65 SetDevice()

```
void SetDevice (
    unsigned int channel,
    int device )
```

11.122.3.66 SetDeviceType()

```
void SetDeviceType (
    TcxDeviceTypeEnumNet devicetype )
```

11.122.3.67 SetDevname()

```
void SetDevname (
    unsigned int device,
    String^ Devicename )
```

11.122.3.68 SetEnableHeaterLimit()

```
void SetEnableHeaterLimit (
    unsigned int device,
    bool enable )
```

11.122.3.69 SetEnableThermocouple()

```
void SetEnableThermocouple (
    unsigned int device,
    bool enable )
```

11.122.3.70 SetHeaterLimit()

```
void SetHeaterLimit (
    unsigned int device,
    int heater_limit )
```

11.122.3.71 SetI()

```
void SetI (
    unsigned int device,
    int i_coeff )
```

Sets the I-coefficient of the specified device.

11.122.3.72 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

11.122.3.73 SetMaxP()

```
void SetMaxP (
    unsigned int device,
    int maxp )
```

Sets the maximum heater power of the specified device.

11.122.3.74 SetOnOff()

```
void SetOnOff (
    unsigned int channel,
    bool on )
```

Switches the specified channel on or off.

Parameters

channel	The channel number.

11.122.3.75 SetP() `void SetP (`

```
unsigned int device,
int p_coeff )
```

Sets the P-coefficient of the specified device.

11.122.3.76 SetSensorType() `void SetSensorType (`

```
unsigned int device,
TcxSensorTypeEnumNet type )
```

11.122.3.77 SetSetpoint() `void SetSetpoint (`

```
unsigned int channel,
int sp )
```

Sets the target temperate of specified channel in units of 0.1 °C.

11.122.3.78 SetThermocoupleNanovoltPerKelvin() `void SetThermocoupleNanovoltPerKelvin (`

```
unsigned int channel,
unsigned int value )
```

Sets the proportional constant for the thermocouple.

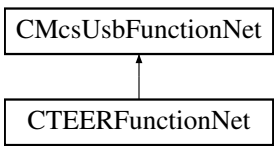
Parameters

channel	Thermocouple channel number.
value	Proportinal constant in Nanovolt per Kelvin.

11.123 CTEERFunctionNet Class Reference

CTEERFunctionNet is the class to control the TEER device

Inheritance diagram for CTEERFunctionNet:



Public Member Functions

- CTEERFunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ pTEERFunctionPointer↩ Container)

  Initializes a new instance of the CTEERFunctionNet class.
- CTEERFunctionNet (CMcsUsbNet^ mcsusb)
- virtual ~CTEERFunctionNet ()
- !CTEERFunctionNet ()
- uint32_t GetPeriod_us ()

  gets the period of TEER stimulation in us
- void SetPeriod_us (uint32_t period_us)

  sets the period of TEER stimulation in us
- uint32_t GetAmplitude_nA ()

  gets TEER stimulation amplitude in nA
- void SetAmplitude_nA (uint32_t Amplitude_nA)

  sets TEER stimulation amplitude in nA
- TeerWaveformEnumNet GetWaveform ()

  gets TEER stimulation waveform (sine/rect)
- void SetWaveform (TeerWaveformEnumNet Waveform)

  sets TEER stimulation waveform (sine/rect)
- TeerClampModeEnumNet GetClampMode ()

  gets TEER clamp mode (voltage/current)
- void SetClampMode (TeerClampModeEnumNet ClampMode)

  sets TEER clamp mode (voltage/current)
- void StartSampling (uint32_t NumberOfCycles)

  starts TEER stimulation (duration: n cycles) and samples during last cycle
- void StopSampling ()

  stops TEER stimulation and sampling
- uint32_t IsSamplingFinished ()

  returns false iff stimulation/sampling is going on, otherwise true
- void SetControllerParams (uint32_t P, uint32_t I, uint32_t D)

  sets PID controller parameters for voltage clamp mode
- void GetControllerParams ([System::Runtime::InteropServices::Out]uint32_t% P, [System::Runtime::↩ InteropServices::Out]uint32_t% I, [System::Runtime::InteropServices::Out]uint32_t% D)

  gets PID controller parameters for voltage clamp mode
- array< int32_t > ^ GetSampleBufferChunk (int Buffer_Length)

  private function to query max. 100 bytes of sample buffer; called internally
- array< int32_t > ^ GetSampleVoltageBuffer_uV (int Buffer_Length)

  returns voltage sample buffer (max. 500 values); unit: uV
- uint32_t GetMaxChunkSize_Byte ()

  private function to be called internally only
- uint32_t GetBytesPerSample ()

  private function to be called internally only
- uint32_t GetNumberOfAvailableSamples ()

  private function to be called internally only
- void SetBufferIndex (uint32_t NewBufferIndex)

  pre-selects sample buffer to be tranferred by GetSampleVoltageBuffer_uV()
- uint32_t GetAdapterCode ()

  gets the adapter code
- uint32_t GetRotaryPositionCode ()

  gets the rotary position code

- void SetExternalLED (uint32_t NewState)

  sets the external LED
- void SetCurrentEnable (bool NewCurrentEnable)

  when disabled, no current will flow through chamber
- bool GetCurrentEnable ()

  when disabled, no current will flow through chamber
- int32_t GetUptimeSeconds ()

  returns time in seconds since device was powered up
- void StartInternalCalibration ()

  starts determination of internal DAC-offset; result is used internally; NON-BLOCKING call
- bool IsInternalCalibrationFinished ()

  queries whether internal calibration has finished
- int GetDacZero ()

  returns DAC-offset (result of internal calibration); use to check for plausibility only
- void CancelInternalCalibration ()

  in case the internal calibration "hangs", this will cancel it
- void SetLiquidResistance (int32_t NewLiquidResistance_Ohm)

  sets the resistance of the liquid in ohms
- int32_t GetLiquidResistance ()

  gets the resitance of the liquid in ohms
- int GetScaleFactorU1 ()

  returns U1 scale factor times 10^6 (result of internal calibration)
- int GetScaleFactorU2 ()

  returns U2 scale factor times 10^6 (result of internal calibration)
- int GetAdcOffsetU1 ()

  returns ADC offset of U1 channel (result of internal calibration)
- int GetAdcOffsetU2 ()

  returns ADC offset of U2 channel (result of internal calibration)
- int GetFrameErrorCounter ()

  returns number of times (since bootup) sample memory got overwritten
- int GetSampleRate ()

  returns sample rate in Hz

Additional Inherited Members

11.123.1 Detailed Description

CTEERFunctionNet is the class to control the TEER device

11.123.2 Constructor & Destructor Documentation

11.123.2.1 CTEERFunctionNet() [1/2]

```
CTEERFunctionNet (
    CMcsUsbNet^ mcsusb,
    CMcsUsbFunctionPointerContainer^ pTEERFunctionPointerContainer )
```

Initializes a new instance of the CTEERFunctionNet class.

11.123.2.2 CTEERFunctionNet() [2/2] CTEERFunctionNet (CMcsUsbNet^ *mcsusb*)

11.123.2.3 ~CTEERFunctionNet() virtual ~CTEERFunctionNet ( ) [virtual]

11.123.2.4 "!CTEERFunctionNet() !CTEERFunctionNet ( )

11.123.3 Member Function Documentation

11.123.3.1 CancelInternalCalibration() void CancelInternalCalibration ( )

in case the internal calibration "hangs", this will cancel it

11.123.3.2 GetAdapterCode() uint32_t GetAdapterCode ( )

gets the adapter code

Returns

the adapter code

11.123.3.3 GetAdcOffsetU1() int GetAdcOffsetU1 ( )

returns ADC offset of U1 channel (result of internal calibration)

Returns

the ADC offset for U1

11.123.3.4 GetAdcOffsetU2() int GetAdcOffsetU2 ( )

returns ADC offset of U2 channel (result of internal calibration)

Returns

the ADC offset for U2

11.123.3.5 GetAmplitude_nA() `uint32_t GetAmplitude_nA ( )`

gets TEER stimulation amplitude in nA

Returns

current stimulation amplitude in nA

11.123.3.6 GetBytesPerSample() `uint32_t GetBytesPerSample ( )`

private function to be called internally only

Returns

not documented

11.123.3.7 GetClampMode() `TeerClampModeEnumNet GetClampMode ( )`

gets TEER clamp mode (voltage/current)

Returns

current TEER clamp mode

11.123.3.8 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

P	the P value
I	the I value
D	the D value

11.123.3.9 GetCurrentEnable() `bool GetCurrentEnable ( )`

when disabled, no current will flow through chamber

Returns

false when disabled, true when enabled

11.123.3.10 GetDacZero() `int GetDacZero ( )`

returns DAC-offset (result of internal calibration); use to check for plausibility only

Returns

the DAC offset

11.123.3.11 GetFrameErrorCounter() `int GetFrameErrorCounter ( )`

returns number of times (since bootup) sample memory got overwritten

Returns

the number of errors

11.123.3.12 GetLiquidResistance() `int32_t GetLiquidResistance ( )`

gets the resitance of the liquid in ohms

Returns

the resistance in ohms

11.123.3.13 GetMaxChunkSize_Byte() `uint32_t GetMaxChunkSize_Byte ( )`

private function to be called internally only

Returns

not documented

11.123.3.14 GetNumberOfAvailableSamples() `uint32_t GetNumberOfAvailableSamples ( )`

private function to be called internally only

Returns

not documented

11.123.3.15 GetPeriod_us() `uint32_t GetPeriod_us ( )`

gets the period of TEER stimulation in us

Returns

the period in us

11.123.3.16 GetRotaryPositionCode() `uint32_t GetRotaryPositionCode ( )`

gets the rotary position code

Returns

the rotary position code

11.123.3.17 GetSampleBufferChunk() `array<int32_t> ^ GetSampleBufferChunk ( int *Buffer_Length* )`

private function to query max. 100 bytes of sample buffer; called internally

Parameters

Buffer_Length	The maximal length of Buffer.

Returns

not documented

11.123.3.18 GetSampleRate() `int GetSampleRate ( )`

returns sample rate in Hz

Returns

the sample rate in Hz

11.123.3.19 GetSampleVoltageBuffer_uV() `array<int32_t> ^ GetSampleVoltageBuffer_uV ( int Buffer_Length )`

returns voltage sample buffer (max. 500 values); unit: uV

Parameters

Buffer_Length	The maximal length of Buffer.

Returns

the voltage sample buffer

11.123.3.20 GetScaleFactorU1() `int GetScaleFactorU1 ( )`

returns U1 scale factor times 10^6 (result of internal calibration)

Returns

the U1 scale factor

11.123.3.21 GetScaleFactorU2() `int GetScaleFactorU2 ( )`

returns U2 scale factor times 10^6 (result of internal calibration)

Returns

the U2 scale factor

11.123.3.22 GetUptimeSeconds() `int32_t GetUptimeSeconds ( )`

returns time in seconds since device was powered up

Returns

seconds since power-on

11.123.3.23 GetWaveform() `TeerWaveformEnumNet GetWaveform ( )`

gets TEER stimulation waveform (sine/rect)

Returns

waveform enum

11.123.3.24 IsInternalCalibrationFinished() `bool IsInternalCalibrationFinished ( )`

queries whether internal calibration has finished

Returns

true if calibration has finished

11.123.3.25 IsSamplingFinished() `uint32_t IsSamplingFinished ( )`

returns false iff stimulation/sampling is going on, otherwise true

Returns

true if sampling is finished

11.123.3.26 SetAmplitude_nA() `void SetAmplitude_nA ( uint32_t Amplitude_nA )`

sets TEER stimulation amplitude in nA

Parameters

Amplitude_nA	new stimulation amplitude in nA

11.123.3.27 SetBufferIndex() `void SetBufferIndex ( uint32_t NewBufferIndex )`

pre-selects sample buffer to be tranferred by GetSampleVoltageBuffer_uV()

Parameters

NewBufferIndex	0 - chamber voltage; 1 - compliance voltage

11.123.3.28 SetClampMode() void SetClampMode (TeerClampModeEnumNet *ClampMode*)

sets TEER clamp mode (voltage/current)

Parameters

ClampMode	new TEER clamp mode

11.123.3.29 SetControllerParams() void SetControllerParams (uint32_t *P*, uint32_t *I*, uint32_t *D*)

sets PID controller parameters for voltage clamp mode

Parameters

P	the P value
I	the I value
D	the D value

11.123.3.30 SetCurrentEnable() void SetCurrentEnable (bool *NewCurrentEnable*)

when disabled, no current will flow through chamber

Parameters

NewCurrentEnable	disabled when false, enabled when true

11.123.3.31 SetExternalLED() void SetExternalLED (uint32_t *NewState*)

sets the external LED

Parameters

NewState	state

11.123.3.32 SetLiquidResistance() `void SetLiquidResistance ( int32_t NewLiquidResistance_Ohm )`

sets the resistance of the liquid in ohms

Parameters

NewLiquidResistance_Ohm	the resistance in ohms

11.123.3.33 SetPeriod_us() `void SetPeriod_us ( uint32_t period_us )`

sets the period of TEER stimulation in us

Parameters

period_us	the period in us

11.123.3.34 SetWaveform() `void SetWaveform ( TeerWaveformEnumNet Waveform )`

sets TEER stimulation waveform (sine/rect)

Parameters

Waveform	waveform enum

11.123.3.35 StartInternalCalibration() `void StartInternalCalibration ( )`

starts determination of internal DAC-offset; result is used internally; NON-BLOCKING call

11.123.3.36 StartSampling() `void StartSampling ( uint32_t NumberOfCycles )`

starts TEER stimulation (duration: n cycles) and samples during last cycle

Parameters

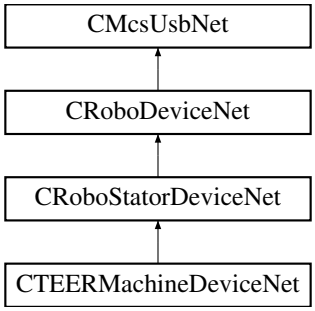
NumberOfCycles	number of cycles (sine or rect) to output (0 - loop forever)

11.123.3.37 StopSampling() `void StopSampling ( )`

stops TEER stimulation and sampling

11.124 CTEERMachineDeviceNet Class Reference

Inheritance diagram for CTEERMachineDeviceNet:



Public Member Functions

- CTEERMachineDeviceNet ()
- ~CTEERMachineDeviceNet ()

Properties

- CTEERFunctionNet^ TEERFunctionNet `[get]`

Additional Inherited Members

11.124.1 Constructor & Destructor Documentation

11.124.1.1 CTEERMachineDeviceNet() `CTEERMachineDeviceNet ( )`

11.124.1.2 ~CTEERMachineDeviceNet() `~CTEERMachineDeviceNet ( )`

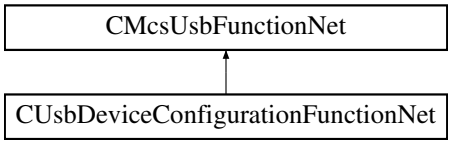
11.124.2 Property Documentation

11.124.2.1 TEERFunctionNet `CTEERFunctionNet^ TEERFunctionNet [get]`

11.125 CUsbDeviceConfigurationFunctionNet Class Reference

CUsbDeviceConfigurationFunctionNet is the class to configure the USB firmware

Inheritance diagram for CUsbDeviceConfigurationFunctionNet:



Public Member Functions

- CUsbDeviceConfigurationFunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ p↩ UsbDeviceConfigurationFunctionPointerContainer)

  Initializes a new instance of the CUsbDeviceConfigurationFunctionNet class.
- CUsbDeviceConfigurationFunctionNet (CMcsUsbNet^ mcsusb)
- virtual ~CUsbDeviceConfigurationFunctionNet ()
- !CUsbDeviceConfigurationFunctionNet ()
- void SetDeviceName (String^ name)

  sets the USB device name for configurable devices
- void SetDeviceId (ProductIdEnumNet id)

  sets the USB device name for configurable devices

Additional Inherited Members

11.125.1 Detailed Description

CUsbDeviceConfigurationFunctionNet is the class to configure the USB firmware

11.125.2 Constructor & Destructor Documentation

11.125.2.1 CUsbDeviceConfigurationFunctionNet() `[1/2]` `CUsbDeviceConfigurationFunctionNet (`

```
CMcsUsbNet^ mcsusb,
CMcsUsbFunctionPointerContainer^ pUsbDeviceConfigurationFunctionPointerContainer
)
```

Initializes a new instance of the CUsbDeviceConfigurationFunctionNet class.

11.125.2.2 CUsbDeviceConfigurationFunctionNet() [2/2] CUsbDeviceConfigurationFunctionNet (CMcsUsbNet^ *mcsusb*)

11.125.2.3 ~CUsbDeviceConfigurationFunctionNet() virtual ~CUsbDeviceConfigurationFunctionNet ( ) [virtual]

11.125.2.4 "!CUsbDeviceConfigurationFunctionNet() !CUsbDeviceConfigurationFunctionNet ( )

11.125.3 Member Function Documentation

11.125.3.1 SetDeviceId() void SetDeviceId (ProductIdEnumNet *id*)

sets the USB device name for configurable devices

Parameters

id	

11.125.3.2 SetDeviceName() void SetDeviceName (String^ *name*)

sets the USB device name for configurable devices

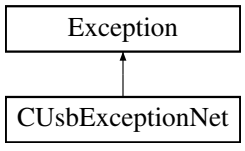
Parameters

name	

11.126 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]

11.126.1 Detailed Description

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

11.126.2 Constructor & Destructor Documentation

11.126.2.1 CUsbExceptionNet() [1/2] CUsbExceptionNet (uint32_t *status*)

Constructor of a CUsbException.

Parameters

status	the status number

11.126.2.2 CUsbExceptionNet() [2/2] CUsbExceptionNet (uint32_t *status*, String^ *message*)

11.126.3 Property Documentation

11.126.3.1 Status uint32_t Status [get]

11.127 CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet Class Reference

Public Member Functions

- CVoltageRangeInfoNet (int vr, String^ vrString)

Public Attributes

- int VoltageRangeInMicroVolt
- String ^ VoltageRangeDisplayStringMilliVolt

11.127.1 Constructor & Destructor Documentation

11.127.1.1 CVoltageRangeInfoNet() `CVoltageRangeInfoNet (`

```
int vr,
String^ vrString )
```

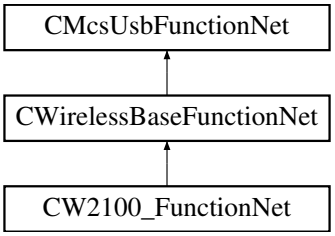
11.127.2 Member Data Documentation

11.127.2.1 VoltageRangeDisplayStringMilliVolt `String ^ VoltageRangeDisplayStringMilliVolt`

11.127.2.2 VoltageRangeInMicroVolt `int VoltageRangeInMicroVolt`

11.128 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 GetFPGAFirmwareType (int TimeSlotNr)

Static Public Member Functions

- static void ClearUserDefinedNameCache ()
- static void ClearUserDefinedNameCache (unsigned short ID)
- static void ClearStimulusParametersCache ()
- static void ClearStimulusParametersCache (unsigned short ID)

Properties

- CW2100_StimulatorFunctionNet^ Stimulator [get]
- CPulseGeneratorFunctionNet^ PulseGenerator [get]

Additional Inherited Members

11.128.1 Constructor & Destructor Documentation

11.128.1.1 CW2100_FunctionNet() [1/2]

```
CW2100_FunctionNet (
    CMcsUsbNet^ mcsusb,
    CMcsUsbFunctionPointerContainer^ w2100_FunctionPointerContainer )
```

11.128.1.2 CW2100_FunctionNet() [2/2]

```
CW2100_FunctionNet (
    CMcsUsbNet^ mcsusb )
```

11.128.2 Member Function Documentation

11.128.2.1 ClearStimulusParametersCache() [1/2]

```
static void ClearStimulusParametersCache ( )
[static]
```

11.128.2.2 ClearStimulusParametersCache() [2/2]

```
static void ClearStimulusParametersCache (
    unsigned short ID )  [static]
```

11.128.2.3 ClearUserDefinedNameCache() [1/2]

```
static void ClearUserDefinedNameCache ( )  [static]
```

11.128.2.4 ClearUserDefinedNameCache() [2/2]

```
static void ClearUserDefinedNameCache (
    unsigned short ID )  [static]
```

11.128.2.5 DeselectAllHeadstages()

```
void DeselectAllHeadstages ( )
```

11.128.2.6 DeselectHeadstage()

```
void DeselectHeadstage (
    int TimeSlotNr )
```

11.128.2.7 GetAccelGyroCurrentRate()

```
int GetAccelGyroCurrentRate (
    int TimeSlotNr )
```

11.128.2.8 GetAccelGyroDesiredRate()

```
int GetAccelGyroDesiredRate (
    int TimeSlotNr )
```

11.128.2.9 GetAccelGyroEnabled()

```
W2100_Accel_Gyro_Select_EnumNet GetAccelGyroEnabled (
    int TimeSlotNr )
```

11.128.2.10 GetAccelRange()

```
int GetAccelRange (
    int TimeSlotNr )
```

11.128.2.11 GetAnalogOutChannel()

```
unsigned int GetAnalogOutChannel (
    [System::Runtime::InteropServices::Out] int % automatic,
    unsigned short index )
```

11.128.2.12 GetAnalogOutFilter()

```
array<unsigned int> ^ GetAnalogOutFilter (
    [System::Runtime::InteropServices::Out] int % automatic )
```

11.128.2.13 GetAudioChannels()

```
array<AudioChannelsNet^> ^ GetAudioChannels ( )
```

11.128.2.14 GetAvailableHeadstages()

```
array<HeadStageIDType^> ^ GetAvailableHeadstages (
    unsigned int max_length )
```

11.128.2.15 GetBatteryState()

```
BatteryState ^ GetBatteryState (
    int TimeSlotNr )
```

11.128.2.16 GetDacRange()

```
AnalogOut_DAC_Range_EnumNet GetDacRange ( )
```

11.128.2.17 GetFilterProperties()

```
array<CFilterPropertyNet^> ^ GetFilterProperties (
    W2100DacqGroupChannelEnumNet GroupID )
```

11.128.2.18 GetFilterProperty()

```
CFilterPropertyNet ^ GetFilterProperty (
    W2100DacqGroupChannelEnumNet GroupID,
    unsigned int index )
```

11.128.2.19 GetFPGAFirmwareType()

```
unsigned int GetFPGAFirmwareType (
    int TimeSlotNr )
```

11.128.2.20 GetGyroRange()

```
int GetGyroRange (
    int TimeSlotNr )
```

11.128.2.21 GetHeadstageOnOff()

```
unsigned short GetHeadstageOnOff (
    int TimeSlotNr )
```

11.128.2.22 GetHeadstageSamplingActive()

```
bool GetHeadstageSamplingActive (
    int TimeSlotNr )
```

11.128.2.23 GetMultiHeadstageMode()

```
bool GetMultiHeadstageMode ( )
```

11.128.2.24 GetPicFirmwareType()

```
unsigned int GetPicFirmwareType (
            int TimeSlotNr )
```

11.128.2.25 GetSelectedChannels()

```
array<BYTE> ^ GetSelectedChannels (
            int TimeSlotNr )
```

11.128.2.26 GetSelectedHeadstageState()

```
HeadStageIDTypeState ^ GetSelectedHeadstageState (
            int TimeSlotNr )
```

11.128.2.27 GetStimulusParametersCache() [1/2]

```
W2100_StimulusParametersNet ^ GetStimulus↩
ParametersCache (
            unsigned int typeValue )
```

11.128.2.28 GetStimulusParametersCache() [2/2]

```
uint32_t GetStimulusParametersCache (
            unsigned int typeValue,
            [System::Runtime::InteropServices::Out] W2100_StimulusParametersNet^% Stimulus↩
Parameters )
```

11.128.2.29 GetStimulusParametersFromSelectedHS()

```
W2100_StimulusParametersNet ^ GetStimulus↩
ParametersFromSelectedHS (
            int TimeSlotNr )
```

11.128.2.30 GetStiumlusParameters()

```
W2100_StimulusParametersNet ^ GetStiumlusParameters (
            unsigned short ID )
```

11.128.2.31 GetUserDefinedName()

```
System::String ^ GetUserDefinedName (
            unsigned short ID )
```

11.128.2.32 GetUserDefinedNameCache() [1/2]

```
System::String ^ GetUserDefinedNameCache (
            unsigned short ID )
```

11.128.2.33 GetUserDefinedNameCache() [2/2] uint32_t GetUserDefinedNameCache (

```
unsigned short ID,
[System::Runtime::InteropServices::Out] System::String^% Name )
```

11.128.2.34 GetUserDefinedNameFromSelectedHS() System::String ^ GetUserDefinedNameFrom↩ SelectedHS (

```
int TimeSlotNr )
```

11.128.2.35 SelectHeadstage() void SelectHeadstage (

```
unsigned int IDorEntry,
int TimeSlotNr )
```

11.128.2.36 SetAccelGyroDesiredRate() void SetAccelGyroDesiredRate (

```
int rate,
int TimeSlotNr )
```

11.128.2.37 SetAccelGyroEnabled() void SetAccelGyroEnabled (

```
W2100_Accel_Gyro_Select_EnumNet enable,
int TimeSlotNr )
```

11.128.2.38 SetAccelRange() void SetAccelRange (

```
int range,
int TimeSlotNr )
```

11.128.2.39 SetAnalogOutChannel() void SetAnalogOutChannel (

```
int automatic,
unsigned short index,
unsigned int Channel )
```

11.128.2.40 SetAnalogOutFilter() void SetAnalogOutFilter (

```
int automatic,
array< unsigned int >^ Coeffs )
```

11.128.2.41 SetAudioChannels()

```
void SetAudioChannels (
    array< AudioChannelsNet^>^ channels )
```

11.128.2.42 SetDacRange()

```
void SetDacRange (
    AnalogOut_DAC_Range_EnumNet range )
```

11.128.2.43 SetGyroRange()

```
void SetGyroRange (
    int range,
    int TimeSlotNr )
```

11.128.2.44 SetHeadstageOnOff()

```
void SetHeadstageOnOff (
    unsigned short On,
    int TimeSlotNr )
```

11.128.2.45 SetHeadstageSamplingActive()

```
void SetHeadstageSamplingActive (
    bool Active,
    int TimeSlotNr )
```

11.128.2.46 SetHeadstageToSleep()

```
void SetHeadstageToSleep (
    unsigned int Sleep16ms,
    int TimeSlotNr )
```

11.128.2.47 SetMultiHeadstageMode()

```
void SetMultiHeadstageMode (
    bool Mode )
```

11.128.2.48 SetSelectedChannels()

```
void SetSelectedChannels (
    array< BYTE >^ channels,
    int TimeSlotNr )
```

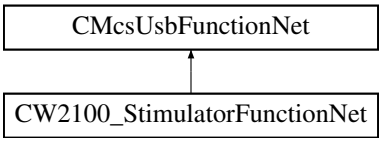
11.128.3 Property Documentation

11.128.3.1 PulseGenerator CPulseGeneratorFunctionNet^ PulseGenerator [get]

11.128.3.2 Stimulator CW2100_StimulatorFunctionNet^ Stimulator [get]

11.129 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.
- int GetCurrentResolutionInPicoAmp (uint32_t channel)

  Gets the Current Resolution of the specified channel in Picoamps.
- uint32_t GetNumberOfAnalogChannels ()
- uint32_t GetNumberOfSyncoutChannels ()
- uint32_t GetNumberOfTriggerInputs ()
- void SelectTimeSlot (int TimeSlotNr)

- int GetTimeSlot ()
- uint32_t GetStimulationPatternMemory ()
- uint32_t GetBoostPreTime ()
- uint32_t GetBoostAlwaysOnMode ()
- void SetDigitalStimulatorTrigger (int TimeSlotNr, DigitalStimulatorTriggerEventEnumNet trigger_event, int trigger_number, W2100DigitalSourceEnumNet digstream_source, int bitnumber_offset)
- void GetDigitalStimulatorTrigger (int TimeSlotNr, DigitalStimulatorTriggerEventEnumNet trigger_event, int trigger_number, [System::Runtime::InteropServices::Out]W2100DigitalSourceEnumNet% digstream_source, [System::Runtime::InteropServices::Out]int% bitnumber_offset)
- void SetDigitalStimulatorTriggerSlope (int TimeSlotNr, DigitalStimulatorTriggerEventEnumNet trigger_event, int trigger_number, DigitalStimulatorTriggerSlopeEnumNet slope)
- DigitalStimulatorTriggerSlopeEnumNet GetDigitalStimulatorTriggerSlope (int TimeSlotNr, DigitalStimulatorTriggerEventEnumNe 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^ PollStatusEvent

Additional Inherited Members

11.129.1 Constructor & Destructor Documentation

11.129.1.1 CW2100_StimulatorFunctionNet() `CW2100_StimulatorFunctionNet ( CMcsUsbNet^ mcsusb )`

11.129.2 Member Function Documentation

11.129.2.1 ClearChannelData() `void ClearChannelData ( int channel )`

Delete a Stimulus Pattern from STG memory

Parameters

channel	specifies the channel to clear.

11.129.2.2 GetBoostAlwaysOnMode() uint32_t GetBoostAlwaysOnMode ( )

11.129.2.3 GetBoostPreTime() uint32_t GetBoostPreTime ( )

11.129.2.4 GetCurrentRangeInNanoAmp() int GetCurrentRangeInNanoAmp (uint32_t *channel*)

Gets the Current Range of the specified channel in Nanoamps.

Parameters

channel	Channel which is queried.

Returns

The Current Range of the specified channel in Nanoamps.

11.129.2.5 GetCurrentResolutionInNanoAmp() int GetCurrentResolutionInNanoAmp (uint32_t *channel*)

Gets the Current Resolution of the specified channel in Nanoamps.

Parameters

channel	Channel which is queried.

Returns

The Current Resolution of the specified channel in Nanoamps.

11.129.2.6 GetCurrentResolutionInPicoAmp() int GetCurrentResolutionInPicoAmp (uint32_t *channel*)

Gets the Current Resolution of the specified channel in Picoamps.

Parameters

channel	Channel which is queried.

Returns

The Current Resolution of the specified channel in Picoamps.

11.129.2.7 GetDACResolution() `int GetDACResolution ( )`

Gets number of bits of the DAC resolution.

Returns

The DAC resolution in bits.

11.129.2.8 GetDigitalStimulatorTrigger() `void GetDigitalStimulatorTrigger (`

```
int TimeSlotNr,
DigitalStimulatorTriggerEventEnumNet trigger_event,
int trigger_number,
[System::Runtime::InteropServices::Out] W2100DigitalSourceEnumNet% digstream_source,
[System::Runtime::InteropServices::Out] int% bitnumber_offset )
```

11.129.2.9 GetDigitalStimulatorTriggerSlope() `DigitalStimulatorTriggerSlopeEnumNet GetDigitalStimulatorTriggerSlope (`

```
int TimeSlotNr,
DigitalStimulatorTriggerEventEnumNet trigger_event,
int trigger_number )
```

11.129.2.10 GetNumberOfAnalogChannels() `uint32_t GetNumberOfAnalogChannels ( )`

11.129.2.11 GetNumberOfSyncoutChannels() `uint32_t GetNumberOfSyncoutChannels ( )`

11.129.2.12 GetNumberOfTriggerInputs() `uint32_t GetNumberOfTriggerInputs ( )`

11.129.2.13 GetStimulationPatternMemory() uint32_t GetStimulationPatternMemory ( )

11.129.2.14 GetTimeResolutionInNanoSeconds() int GetTimeResolutionInNanoSeconds ( )

Gets number of bits of the DAC resolution.

Returns

The time resolution in ns.

11.129.2.15 GetTimeSlot() int GetTimeSlot ( )

11.129.2.16 GetVoltageRangeInMicroVolt() int GetVoltageRangeInMicroVolt (uint32_t *channel*)

Gets the Voltage Range of the specified channel in Microvolts.

Parameters

channel	Channel which is queried.

Returns

The Voltage Range of the specified channel in Microvolts.

11.129.2.17 GetVoltageResolutionInMicroVolt() int GetVoltageResolutionInMicroVolt (uint32_t *channel*)

Gets the Voltage Resolution of the specified channel in Microvolts.

Parameters

channel	Channel which is queried.

Returns

The Voltage Resolution of the specified channel in Microvolts.

11.129.2.18 PrepareData()

```
CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData ^ PrepareData (
        int channel,
        array< int32_t >^ amplitude,
        array< uint64_t >^ duration,
        STG_DestinationEnumNet destType,
        uint32_t repeat )
```

11.129.2.19 PrepareDataSync()

```
CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData ^ PrepareDataSync (
        int channel,
        array< int32_t >^ amplitude,
        array< uint32_t >^ Sync,
        array< uint64_t >^ duration,
        STG_DestinationEnumNet destType,
        uint32_t repeat )
```

11.129.2.20 SelectTimeSlot()

```
void SelectTimeSlot (
        int TimeSlotNr )
```

11.129.2.21 SendPreparedData()

```
void SendPreparedData (
        int channel,
        CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData^ device_data_and_unrolled,
        STG_DestinationEnumNet destType )
```

11.129.2.22 SendStart()

```
void SendStart (
        uint32_t triggermap )
```

Start (Trigger) the STG. The startup delay is in the range of a few ms.

Parameters

triggermap	A bitmap of triggers which will be started.

11.129.2.23 SendStop()

```
void SendStop (
        uint32_t triggermap )
```

Stop some or all triggers of the STG.

Parameters

triggermap	A bitmap of triggers which will be stopped.

11.129.2.24 SetDigitalStimulatorTrigger()

```
void SetDigitalStimulatorTrigger (
    int TimeSlotNr,
    DigitalStimulatorTriggerEventEnumNet trigger_event,
    int trigger_number,
    W2100DigitalSourceEnumNet digstream_source,
    int bitnumber_offset )
```

11.129.2.25 SetDigitalStimulatorTriggerSlope()

```
void SetDigitalStimulatorTriggerSlope (
    int TimeSlotNr,
    DigitalStimulatorTriggerEventEnumNet trigger_event,
    int trigger_number,
    DigitalStimulatorTriggerSlopeEnumNet slope )
```

11.129.2.26 StartPoll()

```
void StartPoll ( )
```

11.129.2.27 StopPoll()

```
void StopPoll ( )
```

11.129.3 Member Data Documentation

11.129.3.1 BOOST_BIT

```
const uint32_t BOOST_BIT = (1 << 0)  [static]
```

11.129.3.2 GND_SWITCH_BIT

```
const uint32_t GND_SWITCH_BIT = (1 << 1)  [static]
```

11.129.3.3 SYNC_BIT0

```
const uint32_t SYNC_BIT0 = (1 << 2)  [static]
```

11.129.3.4 SYNC_BIT1

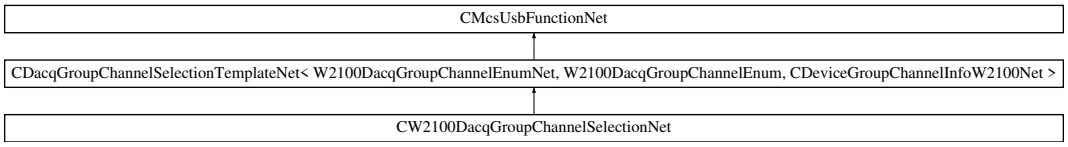
```
const uint32_t SYNC_BIT1 = (1 << 3)  [static]
```

11.129.4 Event Documentation

11.129.4.1 PollStatusEvent `OnStgPollStatus^ PollStatusEvent`

11.130 CW2100DacqGroupChannelSelectionNet Class Reference

Inheritance diagram for CW2100DacqGroupChannelSelectionNet:



Public Member Functions

- CW2100DacqGroupChannelSelectionNet (CMcsUsbNet^ mcsusb)

Additional Inherited Members

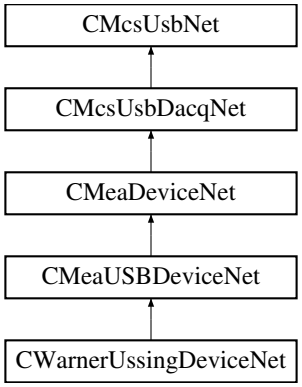
11.130.1 Constructor & Destructor Documentation

11.130.1.1 CW2100DacqGroupChannelSelectionNet() `CW2100DacqGroupChannelSelectionNet ( CMcsUsbNet^ mcsusb )`

11.131 CWarnerUssingDeviceNet Class Reference

CWarnerUssingDeviceNet is the class to control the Ussing device

Inheritance diagram for CWarnerUssingDeviceNet:



Public Member Functions

- CWarnerUssingDeviceNet ()

  Initializes a new instance of the CWarnerUssingDeviceNet class.
- virtual ~CWarnerUssingDeviceNet ()
- !CWarnerUssingDeviceNet ()

Properties

- CWarnerUssingFunctionNet^ WarnerUssingFunction [get]

Additional Inherited Members

11.131.1 Detailed Description

CWarnerUssingDeviceNet is the class to control the Ussing device

11.131.2 Constructor & Destructor Documentation

11.131.2.1 CWarnerUssingDeviceNet() CWarnerUssingDeviceNet ( )

Initializes a new instance of the CWarnerUssingDeviceNet class.

11.131.2.2 ~CWarnerUssingDeviceNet() virtual ~CWarnerUssingDeviceNet ( ) [virtual]

11.131.2.3 "!CWarnerUssingDeviceNet() !CWarnerUssingDeviceNet ( )

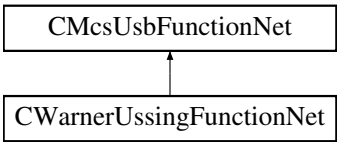
11.131.3 Property Documentation

11.131.3.1 WarnerUssingFunction CWarnerUssingFunctionNet^ WarnerUssingFunction [get]

11.132 CWarnerUssingFunctionNet Class Reference

CWarnerUssingFunctionNet is the class to control the Ussing device

Inheritance diagram for CWarnerUssingFunctionNet:



Public Member Functions

- CWarnerUssingFunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ pWarner↩ UssingFunctionPointerContainer)

  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 NewIdle↩ Offset)

  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

11.132.1 Detailed Description

CWarnerUssingFunctionNet is the class to control the Ussing device

11.132.2 Constructor & Destructor Documentation

11.132.2.1 CWarnerUssingFunctionNet() [1/2] CWarnerUssingFunctionNet (CMcsUsbNet^ *mcsusb*, CMcsUsbFunctionPointerContainer^ *pWarnerUssingFunctionPointerContainer*)

Initializes a new instance of the CWarnerUssingFunctionNet class.

11.132.2.2 CWarnerUssingFunctionNet() [2/2] CWarnerUssingFunctionNet (CMcsUsbNet^ *mcsusb*)

11.132.2.3 ~CWarnerUssingFunctionNet() virtual ~CWarnerUssingFunctionNet ( ) [virtual]

11.132.2.4 "!CWarnerUssingFunctionNet() !CWarnerUssingFunctionNet ( )

11.132.3 Member Function Documentation

11.132.3.1 CompensateElectrodeOffset() bool CompensateElectrodeOffset (int32_t *ChamberId*)

blocking call to compensate electrode offset of one chamber; returns true when successful

Parameters

Chamber↩ Id	index of hardware chamber slot (zero-based)

Returns

true if compensation succeeded

11.132.3.2 GetAvailableChambers() `array<int> ^ GetAvailableChambers ( )`

returns array with (zero-based) ChamberIds of all available chambers

11.132.3.3 GetChannelsCountOfChamber() `int32_t GetChannelsCountOfChamber ( int32_t ChamberId )`

gets number of channels in datastream from chamber amp with given index

Parameters

ChamberId	index of hardware chamber slot (zero-based)

Returns

return value of zero means that amp is not placed

11.132.3.4 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

ChamberId	index of hardware chamber slot (zero-based)

Returns

the current clamp mode

11.132.3.5 GetComplianceVoltageThreshold() `int32_t GetComplianceVoltageThreshold ( int32_t ChamberId )`

returns compliance voltage threshold in uV; when Uc is above, current source is overloaded

Parameters

ChamberId	index of hardware chamber slot (zero-based)

Returns

the compliance voltage threshold in uV

11.132.3.6 GetDacPampsPerDigitHighCurrentRange() `uint32_t GetDacPampsPerDigitHighCurrentRange (`

```
int32_t ChamberId )
```

get the resolution of the high current mode

Parameters

ChamberId	index of hardware chamber slot (zero-based)

Returns

unit: pA/digit in high current mode

11.132.3.7 GetDacPampsPerDigitLowCurrentRange() `uint32_t GetDacPampsPerDigitLowCurrentRange (`

```
int32_t ChamberId )
```

get the resolution of the low current mode

Parameters

ChamberId	index of hardware chamber slot (zero-based)

Returns

pA/digit in low current mode

11.132.3.8 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

ChamberId	index of hardware chamber slot (zero-based)

Returns

the zero value of the DAC

11.132.3.9 GetHighCurrentRange() `uint32_t GetHighCurrentRange ( int32_t ChamberId )`

query the range of the high current mode

Parameters

ChamberId	index of hardware chamber slot (zero-based)

Returns

low current range in nA

11.132.3.10 GetIdleModeOffset() `int32_t GetIdleModeOffset ( int32_t ChamberId, UssingClampModeEnumNet ClampMode )`

gets the offset (voltage or current) that will be applied when clamping is DISABLED

Parameters

ChamberId	index of hardware chamber slot (zero-based)
ClampMode	voltage or current clamp stimulation

Returns

unit: nA or uV

11.132.3.11 GetLiquidResistance() `int32_t GetLiquidResistance ( int32_t ChamberId )`

gets the resistance of the liquid

Parameters

ChamberId	index of hardware chamber slot (zero-based)

Returns

the liquid restistance in ohm

11.132.3.12 GetLowCurrentRange() `uint32_t GetLowCurrentRange ( int32_t ChamberId )`

query the range of the low current mode

Parameters

ChamberId	index of hardware chamber slot (zero-based)

Returns

low current range in nA

11.132.3.13 GetNumberOfAvailableChambers() `int32_t GetNumberOfAvailableChambers ( )`

gets number of actually connected chamber amps

Returns

the number of actually connected chambers

11.132.3.14 GetNumberOfHardwareSlotsForChambers() `int32_t GetNumberOfHardwareSlotsForChambers ( )`

gets number of physical hardware slots for chambers amps

Returns

the number of hardware chamber slots on the backplane

11.132.3.15 GetU1Offset() `int32_t GetU1Offset ( int32_t ChamberId )`

- diagnostic function only -

Parameters

ChamberId	index of hardware chamber slot (zero-based)

Returns

U1 offset

11.132.3.16 GetU1Reference() `int32_t GetU1Reference ( int32_t ChamberId )`

- diagnostic function only -

Parameters

ChamberId	index of hardware chamber slot (zero-based)

Returns

U1 reference

11.132.3.17 GetU2Offset() `int32_t GetU2Offset ( int32_t ChamberId )`

- diagnostic function only -

Parameters

ChamberId	index of hardware chamber slot (zero-based)

Returns

U2 offset

11.132.3.18 GetU2Reference() `int32_t GetU2Reference ( int32_t ChamberId )`

- diagnostic function only -

Parameters

ChamberId	index of hardware chamber slot (zero-based)

Returns

U2 reference

11.132.3.19 GetUnitDescription() `String ^ GetUnitDescription (`

```
int32_t ChamberId,
int32_t ChannelId )
```

gets the description for the unit

Parameters

ChamberId	index of hardware chamber slot (zero-based)
ChannelId	index of channel (zero-based)

Returns

the description of the unix

11.132.3.20 GetUnitExponent() `int32_t GetUnitExponent (`

```
int32_t ChamberId,
int32_t ChannelId )
```

gets the unit exponent for specified chamber and channel

Parameters

ChamberId	index of hardware chamber slot (zero-based)
ChannelId	index of channel (zero-based)

Returns

example: return value -9 means that amps/volts per digit is in nano

11.132.3.21 GetUnitName()

```
UssingUnitEnumNet GetUnitName (
    int32_t ChamberId,
    int32_t ChannelId )
```

gets the channel's unit name

Parameters

ChamberId	index of hardware chamber slot (zero-based)
ChannelId	index of channel (zero-based)

Returns

the name of the unit

11.132.3.22 GetUnitsPerDigit()

```
uint32_t GetUnitsPerDigit (
    int32_t ChamberId,
    int32_t ChannelId )
```

gets amps/volts per digit for specified chamber and channel

Parameters

ChamberId	index of hardware chamber slot (zero-based)
ChannelId	index of channel (zero-based)

Returns

amps/volts per digit

11.132.3.23 GetUptimeSeconds()

```
int32_t GetUptimeSeconds (
    int32_t ChamberId )
```

gets the uptime in seconds

Parameters

ChamberId	index of hardware chamber slot (zero-based)

Returns

seconds since power-on

11.132.3.24 GetVoltageClampControllerParam_D() `uint32_t GetVoltageClampControllerParam_D ( int32_t ChamberId )`

gets D value of PID controller;

Parameters

ChamberId	index of hardware chamber slot (zero-based)

Returns

the D value

11.132.3.25 GetVoltageClampControllerParam_I() `uint32_t GetVoltageClampControllerParam_I ( int32_t ChamberId )`

gets I value of PID controller;

Parameters

ChamberId	index of hardware chamber slot (zero-based)

Returns

the I value

11.132.3.26 GetVoltageClampControllerParam_P() `uint32_t GetVoltageClampControllerParam_P ( int32_t ChamberId )`

gets P value of PID controller;

Parameters

ChamberId	index of hardware chamber slot (zero-based)

Returns

the P value

11.132.3.27 IsChamberAvailable() bool IsChamberAvailable (int32_t *ChamberId*)

checks whether chamber amp is connected to slot

Parameters

ChamberId	index of hardware chamber slot (zero-based)

Returns

true if the chamber is available

11.132.3.28 IsHighCurrentMode() bool IsHighCurrentMode (int32_t *ChamberId*)

Parameters

ChamberId	index of hardware chamber slot (zero-based)

Returns

true if in hight current mode

11.132.3.29 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

ChamberId	index of hardware chamber slot (zero-based)

Returns

true if finished

11.132.3.30 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

ChamberId	index of hardware chamber slot (zero-based)
ClampMode	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

11.132.3.31 SetClampMode()

```
void SetClampMode (
    int32_t ChamberId,
    UssingClampModeEnumNet NewClampMode )
```

sets clamp mode (voltage, current or open clamp)

Parameters

ChamberId	index of hardware chamber slot (zero-based)
NewClampMode	the clamp mode to use

11.132.3.32 SetEnablePulse()

```
void SetEnablePulse (
    int32_t ChamberId,
    UssingClampModeEnumNet ClampMode,
    bool Enable )
```

enable pulse of given chamber and mode (voltage/current clamp) of this chamber

Parameters

ChamberId	index of hardware chamber slot (zero-based)
ClampMode	voltage or current clamp stimulation
Enable	when ENABLED, previously defined pulse pattern will be applied, otherwise the chamber current/voltage will be kept at specified offset level

11.132.3.33 SetHighCurrentMode()

```
void SetHighCurrentMode (
    int32_t ChamberId )
```

switch to high-current mode

Parameters

ChamberId	index of hardware chamber slot (zero-based)

11.132.3.34 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

ChamberId	index of hardware chamber slot (zero-based)
ClampMode	voltage or current clamp stimulation
NewIdleOffset	unit: nA or uV

11.132.3.35 SetLiquidResistance() `void SetLiquidResistance (`

```
int32_t ChamberId,
int32_t NewLiquidResistance_Ohm )
```

sets the resistance of the liquid

Parameters

ChamberId	index of hardware chamber slot (zero-based)
NewLiquidResistance_Ohm	the liquid resistiance in ohm

11.132.3.36 SetLowCurrentMode() `void SetLowCurrentMode (`

```
int32_t ChamberId )
```

switch to low-current mode

Parameters

ChamberId	index of hardware chamber slot (zero-based)

11.132.3.37 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

ChamberId	index of hardware chamber slot (zero-based); send pattern to connected amp
StgMode	voltage or current clamp stimulation
NumberOfRepetitions	number of repetitions for pulse pattern (-1 for infinite; n means pattern is applied n+1 times)
Amplitudes	amplitude; unit in voltage clamp: uV; unit in current clamp: nA
Durations	duration in 100us; CAUTION: first element is applied only one; auto-loop back to second element after last one
PulseMarker	defines values on digital channel for each step (positive: digital channel "01", neg: "10", zero: "00")

11.132.3.38 SetVoltageClampControllerParam_D()

```
void SetVoltageClampControllerParam_D (
int32_t ChamberId,
uint32_t D )
```

sets D value of PID controller;

Parameters

ChamberId	index of hardware chamber slot (zero-based)
D	useful range: 0..700

11.132.3.39 SetVoltageClampControllerParam_I()

```
void SetVoltageClampControllerParam_I (
int32_t ChamberId,
uint32_t I )
```

sets I value of PID controller;

Parameters

ChamberId	index of hardware chamber slot (zero-based)
I	useful range: 80000..120000

11.132.3.40 SetVoltageClampControllerParam_P() `void SetVoltageClampControllerParam_P ( int32_t ChamberId, uint32_t P )`

sets P value of PID controller;

Parameters

ChamberId	index of hardware chamber slot (zero-based)
P	useful value: 130000

11.132.3.41 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...)

11.132.3.42 WaitForChamber() `bool WaitForChamber ( int32_t ChamberId )`

blocking call that waits for chamber boot-up calibration to complete

Parameters

ChamberId	index of hardware chamber slot (zero-based)

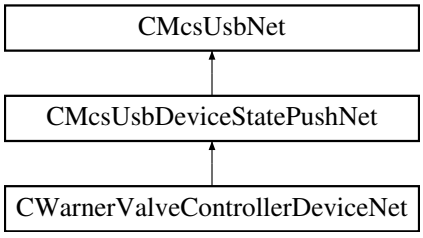
Returns

returns false when calibration fails (e.g. timeout...)

11.133 CWarnerValveControllerDeviceNet Class Reference

CWarnerValveControllerDeviceNet is the class to access the Warner Valve Controller

Inheritance diagram for CWarnerValveControllerDeviceNet:



Public Member Functions

- delegate void OnGetValveActive (uint16_t valve, int valveActive)
- delegate void OnGetValveManualState (uint16_t valve, int32_t valveManualState)
- delegate void OnGetValveManualGroup (uint16_t valve, int32_t valveManualGroup)
- delegate void OnGetValveMode (uint16_t valve, WvcValveModeEnumNet ValveMode)
- delegate void OnGetAnalogThresholdLow (uint16_t valve, int32_t threshold)
- delegate void OnGetAnalogThresholdHigh (uint16_t valve, int32_t threshold)
- delegate void OnGetDigitalPortDirection (uint16_t port, PortDirectionEnumNet direction)
- delegate void OnIsValveDigitalInInverted (uint16_t valve, bool isInverted)
- delegate void OnGetValveDigitalInPort (uint16_t valve, uint32_t digitalInPort)
- delegate void OnIsDigitalOutPortInverted (uint16_t digitalOutPort, bool isInverted)
- delegate void OnGetDigitalOutPortValve (uint16_t digitalOutPort, uint32_t valve)
- delegate void OnIsValveOpen (uint16_t valve, bool valveOpen)
- delegate void OnIsValveOpenInDigitalMode (uint16_t valve, bool valveOpen)
- delegate void OnIsValveOpenInAnalogMode (uint16_t valve, bool valveOpen)
- delegate void OnGetAnalogVoltage (int32_t voltage)
- delegate void OnTableEntryChanged (uint16_t tableNumber)
- delegate void OnGetTableNamebyIndex (uint16_t tableNumber, String^ tableName)
- delegate void OnGetActiveRunningTableNumber (uint32_t tableNumber)
- delegate void OnGetCurrentNumberOfValves (int32_t numberOfValves)
- delegate void OnGetValveBoardRevision (uint32_t revision)
- delegate void OnGetValveLedOn (bool ledon)
- delegate void OnGetDisplayMode (WvcDisplayModeEnumNet DisplayMode)
- CWarnerValveControllerDeviceNet ()

  Initializes a new instance of the CWarnerValveControllerDeviceNet class.
- virtual ~CWarnerValveControllerDeviceNet ()
- !CWarnerValveControllerDeviceNet ()
- int GetValveActive (uint16_t valve)

  Gets the valve active/inactive state
- void SetValveActive (uint16_t valve, int valveActive)

  Sets the valve active/inactive state
- uint32_t GetValvesActiveMap ()

  Gets the valves active/inactive states
- void SetValvesActiveMap (uint32_t valvesActive)

  Sets the valve active/inactive state
- int32_t GetValveManualState (uint16_t valve)

  Gets the valve manual on/off state
- void SetValveManualState (uint16_t valve, int32_t valveManualState)

  Sets the valve manual on/off state
- uint32_t GetValvesManualStateMap ()

  Gets the valves manual on/off states
- void SetValvesManualStateMap (uint32_t valveaManualState)

  Sets the valve manual on/off state
- int32_t GetValveManualGroup (uint16_t valve)

  Gets the valve manual group
- void SetValveManualGroup (uint16_t valve, int32_t valveManualGroup)

  Sets the valve manual group
- WvcValveModeEnumNet GetValveMode (uint16_t valve)

  Reads the valve mode
- void SetValveMode (uint16_t valve, WvcValveModeEnumNet ValveMode)

  Writes the valve mode
- int32_t GetAnalogThresholdLow (uint16_t valve)

Gets the lower threshold for the analog in port per valve

- void SetAnalogThresholdLow (uint16_t valve, int32_t threshold)

  Sets the lower threshold for the analog in port per valve
- int32_t GetAnalogThresholdHigh (uint16_t valve)

  Gets the upper threshold for the analog in port per valve
- void SetAnalogThresholdHigh (uint16_t valve, int32_t threshold)

  Sets the upper threshold for the analog in port per valve
- PortDirectionEnumNet GetDigitalPortDirection (uint16_t port)

  Gets the direction of a digital port
- void SetDigitalPortDirection (uint16_t port, PortDirectionEnumNet direction)

  Sets the direction of a digital port
- bool IsValveDigitalInInverted (uint16_t valve)

  Is digital in inverted
- void SetValveDigitalInInvert (uint16_t valve, bool isInverted)

  Invert digital in
- uint32_t GetValveDigitalInPort (uint16_t valve)

  Gets the number of the digital in port which is mapped to a valve
- void SetValveDigitalInPort (uint16_t valve, uint32_t digitalInPort)

  Map a digital in port to a valve
- bool IsDigitalOutPortInverted (uint16_t digitalOutPort)

  Gets the number of the valve which is mapped to a digital out port
- void SetDigitalOutPortInvert (uint16_t digitalOutPort, bool isInverted)

  Map a valve to a digital out port
- uint32_t GetDigitalOutPortValve (uint16_t digitalOutPort)

  Gets the number of the valve which is mapped to a digital out port
- void SetDigitalOutPortValve (uint16_t digitalOutPort, uint32_t valve)

  Map a valve to a digital out port
- void SetDefault ()

  Sets the settings of the valve controller to default
- bool IsValveOpen (uint16_t valve)

  Is valve open
- bool IsValveOpenInDigitalMode (uint16_t valve)

  True, if the valve would be open when the device is in digital mode
- bool IsValveOpenInAnalogMode (uint16_t valve)

  True, if the valve would be open when the device is in analog mode
- int32_t GetAnalogVoltage ()

  Reads the voltage on the analog in port
- void GetValveTableEntry (uint16_t valve, uint16_t index, [System::Runtime::InteropServices::Out]uint32_t% duration, [System::Runtime::InteropServices::Out]bool% state)

  Read an entry from the valve protocol table
- void SetValveTableEntry (uint16_t valve, uint16_t index, uint32_t duration, bool state)

  Write an entry to the valve protocol table
- void ClearValveTable (uint16_t valve)

  Clear the valve protocol table
- void LoadValveTable ()

  Load the current table from permanent memory
- void StoreValveTable ()

  Store the current table in permanent memory
- String ^ GetTableNamebyIndex (uint16_t tableNumber)

  Get the name of a protocol table
- String ^ GetTableName ()

Get the name of the current protocol table

- void SetTableName (String^ tableName)

  Set the name of the current protocol table

- uint32_t GetActiveRunningTableNumber ()

  Gets the number of the table that is active for running

- void SetActiveRunningTableNumber (uint32_t tableNumber)

  Sets the number of the tanle that is active for running

- uint32_t GetCurrentEditTableNumber ()

  Gets the number of the table that is current for editing

- void SetCurrentEditTableNumber (uint32_t tableNumber)

  Sets the number of the table that is current for editing

- void ClearTableName ()

  Clear the name of current protocol table

- void SetTableStep (uint16_t valve, int32_t steps)

  Skips the table protocol for a valve by steps

- void SetTableStepAll (int32_t steps)

  Skips the table protocol for all valves by steps

- int32_t GetTotalNumberOfValves ()

  Get the total number of valves in the system

- int32_t GetTotalNumberOfDigitalPorts ()

  Get the total number of digital ports in the system

- int32_t GetTotalTableSize ()

  Get the total table size in the system

- int32_t GetTotalNumberOfTables ()

  Get the total number of tables in the system

- int32_t GetCurrentNumberOfValves ()

  Get the current number of valves connected to the system

- uint32_t GetValveBoardRevision ()

  Gets the revision code of the valve board

- bool GetValveLedOn ()

  Gets the LED state of the valve board

- void SetValveLedOn (bool ledon)

  Gets the LED state of the valve board

- WvcDisplayModeEnumNet GetDisplayMode ()

  Reads the display mode

- void SetDisplayMode (WvcDisplayModeEnumNet DisplayMode, int32_t lockTimeMs)

  Writes the display mode

- String ^ GetValveBoardRevisionString ()

  Gets the revision name of the valve board

- void GetValveCurrent ([System::Runtime::InteropServices::Out]int16_t% switch_current, [System::Runtime↩
  ::InteropServices::Out]int16_t% hold_current)

  Gets the valve currents

- void SetValveCurrent (int16_t switch_current, int16_t hold_current)

  Sets the valve currents different from the default

Events

- OnGetValveActive^ GetValveActiveEvent [add, remove, raise]

  Event fires when the valve state for the valve number has changed
- OnGetValveManualState^ GetValveManualStateEvent [add, remove, raise]

  Event fires when the manual valve state for the valve number has changed
- OnGetValveManualGroup^ GetValveManualGroupEvent [add, remove, raise]

  Event fires when the manual valve group for the valve number has changed
- OnGetValveMode^ GetValveModeEvent [add, remove, raise]

  Event fires when the valve mode for the valve number has changed
- OnGetAnalogThresholdLow^ GetAnalogThresholdLowEvent [add, remove, raise]

  Event fires when the threshold in mV for the valve number has changed
- OnGetAnalogThresholdHigh^ GetAnalogThresholdHighEvent [add, remove, raise]

  Event fires when the threshold in mV for the valve number has changed
- OnGetDigitalPortDirection^ GetDigitalPortDirectionEvent [add, remove, raise]

  Event fires when the direction for the port number has changed
- OnIsValveDigitalInInverted^ IsValveDigitalInInvertedEvent [add, remove, raise]

  Event fires when is inverted for the valve number has changed
- OnGetValveDigitalInPort^ GetValveDigitalInPortEvent [add, remove, raise]

  Event fires when the digital in port for the valve number has changed
- OnIsDigitalOutPortInverted^ IsDigitalOutPortInvertedEvent [add, remove, raise]

  Event fires when is inverted for the digital out port has changed
- OnGetDigitalOutPortValve^ GetDigitalOutPortValveEvent [add, remove, raise]

  Event fires when the valve number for the digital out port has changed
- OnIsValveOpen^ IsValveOpenEvent [add, remove, raise]

  Event fires when is open for the valve number has changed
- OnIsValveOpenInDigitalMode^ IsValveOpenInDigitalModeEvent [add, remove, raise]

  Event fires when is open for the valve number has changed
- OnIsValveOpenInAnalogMode^ IsValveOpenInAnalogModeEvent [add, remove, raise]

  Event fires when is open for the valve number has changed
- OnGetAnalogVoltage^ GetAnalogVoltageEvent [add, remove, raise]

  Event fires when the voltage in mV has changed
- OnTableEntryChanged^ TableEntryChangedEvent [add, remove, raise]

  Event fires when an entry of a table changed
- OnGetTableNamebyIndex^ GetTableNamebyIndexEvent [add, remove, raise]

  Event fires when the name of the table for the table number has changed
- OnGetActiveRunningTableNumber^ GetActiveRunningTableNumberEvent [add, remove, raise]

  Event fires when the table number has changed
- OnGetCurrentNumberOfValves^ GetCurrentNumberOfValvesEvent [add, remove, raise]

  Event fires when the number of valves has changed
- OnGetValveBoardRevision^ GetValveBoardRevisionEvent [add, remove, raise]

  Event fires when the revision code has changed
- OnGetValveLedOn^ GetValveLedOnEvent [add, remove, raise]

  Event fires when the LED state has changed
- OnGetDisplayMode^ GetDisplayModeEvent [add, remove, raise]

  Event fires when the display mode has changed

Additional Inherited Members

11.133.1 Detailed Description

CWarnerValveControllerDeviceNet is the class to access the Warner Valve Controller

11.133.2 Constructor & Destructor Documentation

11.133.2.1 CWarnerValveControllerDeviceNet() `CWarnerValveControllerDeviceNet ( )`

Initializes a new instance of the CWarnerValveControllerDeviceNet class.

11.133.2.2 ~CWarnerValveControllerDeviceNet() `virtual ~CWarnerValveControllerDeviceNet ( ) [virtual]`

11.133.2.3 "!CWarnerValveControllerDeviceNet() `!CWarnerValveControllerDeviceNet ( )`

11.133.3 Member Function Documentation

11.133.3.1 ClearTableName() `void ClearTableName ( )`

Clear the name of current protocol table

11.133.3.2 ClearValveTable() `void ClearValveTable ( uint16_t valve )`

Clear the valve protocol table

Parameters

valve	The valve number

11.133.3.3 GetActiveRunningTableNumber() `uint32_t GetActiveRunningTableNumber ( )`

Gets the number of the table that is active for running

Returns

The table number

11.133.3.4 GetAnalogThresholdHigh() int32_t GetAnalogThresholdHigh (uint16_t *valve*)

Gets the upper threshold for the analog in port per valve

Parameters

valve	The valve number

Returns

The threshold in mV

11.133.3.5 GetAnalogThresholdLow() int32_t GetAnalogThresholdLow (uint16_t *valve*)

Gets the lower threshold for the analog in port per valve

Parameters

valve	The valve number

Returns

The threshold in mV

11.133.3.6 GetAnalogVoltage() int32_t GetAnalogVoltage ( )

Reads the voltage on the analog in port

Returns

The voltage in mV

11.133.3.7 GetCurrentEditTableNumber() uint32_t GetCurrentEditTableNumber ( )

Gets the number of the table that is current for editing

Returns

The table number

11.133.3.8 GetCurrentNumberOfValves() `int32_t GetCurrentNumberOfValves ( )`

Get the current number of valves connected to the system

Returns

The number of valves

11.133.3.9 GetDigitalOutPortValve() `uint32_t GetDigitalOutPortValve ( uint16_t digitalOutPort )`

Gets the number of the valve which is mapped to a digital out port

Parameters

digitalOutPort	The digital out port

Returns

The valve number

11.133.3.10 GetDigitalPortDirection() `PortDirectionEnumNet GetDigitalPortDirection ( uint16_t port )`

Gets the direction of a digital port

Parameters

port	The port number

Returns

the direction

11.133.3.11 GetDisplayMode() `WvcDisplayModeEnumNet GetDisplayMode ( )`

Reads the display mode

Returns

The display mode

11.133.3.12 GetTableName() `String ^ GetTableName ( )`

Get the name of the current protocol table

Returns

The name of the table

11.133.3.13 GetTableNamebyIndex() `String ^ GetTableNamebyIndex ( uint16_t tableNumber )`

Get the name of a protocol table

Parameters

tableNumber	The table number

Returns

The name of the table

11.133.3.14 GetTotalNumberOfDigitalPorts() `int32_t GetTotalNumberOfDigitalPorts ( )`

Get the total number of digital ports in the system

Returns

The number of digital ports

11.133.3.15 GetTotalNumberOfTables() `int32_t GetTotalNumberOfTables ( )`

Get the total number of tables in the system

Returns

The number of tables

11.133.3.16 GetTotalNumberOfValves() `int32_t GetTotalNumberOfValves ( )`

Get the total number of valves in the system

Returns

The number of valves

11.133.3.17 GetTotalTableSize() `int32_t GetTotalTableSize ( )`

Get the total table size in the system

Returns

The table size

11.133.3.18 GetValveActive() `int GetValveActive ( uint16_t valve )`

Gets the valve active/inactive state

Parameters

valve	The valve number

Returns

The valve state

11.133.3.19 GetValveBoardRevision() `uint32_t GetValveBoardRevision ( )`

Gets the revision code of the valve board

Returns

The revision code

11.133.3.20 GetValveBoardRevisionString() `String ^ GetValveBoardRevisionString ( )`

Gets the revision name of the valve board

Returns

The revision name

11.133.3.21 GetValveCurrent() `void GetValveCurrent (`

```
[System::Runtime::InteropServices::Out] int16_t% switch_current,
[System::Runtime::InteropServices::Out] int16_t% hold_current )
```

Gets the valve currents

Parameters

switch_current	The switch current (in DAC units)
hold_current	The hold current (in DAC units)

11.133.3.22 GetValveDigitalInPort() `uint32_t GetValveDigitalInPort (`

```
uint16_t valve )
```

Gets the number of the digital in port which is mapped to a valve

Parameters

valve	The valve number

Returns

The digital in port

11.133.3.23 GetValveLedOn() `bool GetValveLedOn ( )`

Gets the LED state of the valve board

Returns

The LED state

11.133.3.24 GetValveManualGroup() `int32_t GetValveManualGroup (`

```
uint16_t valve )
```

Gets the valve manual group

Parameters

valve	The valve number

Returns

The manual valve group

11.133.3.25 GetValveManualState() `int32_t GetValveManualState ( uint16_t valve )`

Gets the valve manual on/off state

Parameters

valve	The valve number

Returns

The manual valve state

11.133.3.26 GetValveMode() `WvcValveModeEnumNet GetValveMode ( uint16_t valve )`

Reads the valve mode

Parameters

valve	The valve number

Returns

The valve mode

11.133.3.27 GetValvesActiveMap() `uint32_t GetValvesActiveMap ( )`

Gets the valves active/inactive states

Returns

The valves states

11.133.3.28 GetValvesManualStateMap() uint32_t GetValvesManualStateMap ( )

Gets the valves manual on/off states

Returns

The manual valves states

11.133.3.29 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

valve	The valve number
index	The index in the table
duration	the duration in ms
state	the state

11.133.3.30 IsDigitalOutPortInverted() bool IsDigitalOutPortInverted (

```
uint16_t digitalOutPort )
```

Gets the number of the valve which is mapped to a digital out port

Parameters

digitalOutPort	The digital out port

Returns

is inverted

11.133.3.31 IsValveDigitalInInverted() bool IsValveDigitalInInverted (

```
uint16_t valve )
```

Is digital in inverted

Parameters

valve	The valve number

Returns

is inverted

11.133.3.32 IsValveOpen() `bool IsValveOpen ( uint16_t valve )`

Is valve open

Parameters

valve	The valve number

Returns

is open

11.133.3.33 IsValveOpenInAnalogMode() `bool IsValveOpenInAnalogMode ( uint16_t valve )`

True, if the valve would be open when the device is in analog mode

Parameters

valve	The valve number

Returns

is open

11.133.3.34 IsValveOpenInDigitalMode() `bool IsValveOpenInDigitalMode ( uint16_t valve )`

True, if the valve would be open when the device is in digital mode

Parameters

valve	The valve number

Returns

is open

11.133.3.35 LoadValveTable() void LoadValveTable ( )

Load the current table from permanent memory

11.133.3.36 OnGetActiveRunningTableNumber() delegate void OnGetActiveRunningTableNumber (uint32_t *tableNumber*)

11.133.3.37 OnGetAnalogThresholdHigh() delegate void OnGetAnalogThresholdHigh (uint16_t *valve*, int32_t *threshold*)

11.133.3.38 OnGetAnalogThresholdLow() delegate void OnGetAnalogThresholdLow (uint16_t *valve*, int32_t *threshold*)

11.133.3.39 OnGetAnalogVoltage() delegate void OnGetAnalogVoltage (int32_t *voltage*)

11.133.3.40 OnGetCurrentNumberOfValves() delegate void OnGetCurrentNumberOfValves (int32_t *numberOfValves*)

11.133.3.41 OnGetDigitalOutPortValve() delegate void OnGetDigitalOutPortValve (uint16_t *digitalOutPort*, uint32_t *valve*)

11.133.3.42 OnGetDigitalPortDirection() delegate void OnGetDigitalPortDirection (uint16_t *port*, PortDirectionEnumNet *direction*)

11.133.3.43 OnGetDisplayMode() delegate void OnGetDisplayMode (WvcDisplayModeEnumNet *DisplayMode*)

11.133.3.44 OnGetTableNamebyIndex()

```
delegate void OnGetTableNamebyIndex (
    uint16_t tableNumber,
    String^ tableName )
```

11.133.3.45 OnGetValveActive()

```
delegate void OnGetValveActive (
    uint16_t valve,
    int valveActive )
```

11.133.3.46 OnGetValveBoardRevision()

```
delegate void OnGetValveBoardRevision (
    uint32_t revision )
```

11.133.3.47 OnGetValveDigitalInPort()

```
delegate void OnGetValveDigitalInPort (
    uint16_t valve,
    uint32_t digitalInPort )
```

11.133.3.48 OnGetValveLedOn()

```
delegate void OnGetValveLedOn (
    bool ledon )
```

11.133.3.49 OnGetValveManualGroup()

```
delegate void OnGetValveManualGroup (
    uint16_t valve,
    int32_t valveManualGroup )
```

11.133.3.50 OnGetValveManualState()

```
delegate void OnGetValveManualState (
    uint16_t valve,
    int32_t valveManualState )
```

11.133.3.51 OnGetValveMode()

```
delegate void OnGetValveMode (
    uint16_t valve,
    WvcValveModeEnumNet ValveMode )
```

11.133.3.52 OnIsDigitalOutPortInverted()

```
delegate void OnIsDigitalOutPortInverted (
    uint16_t digitalOutPort,
    bool isInverted )
```

11.133.3.53 OnIsValveDigitalInInverted()

```
delegate void OnIsValveDigitalInInverted (
    uint16_t valve,
    bool isInverted )
```

11.133.3.54 OnIsValveOpen()

```
delegate void OnIsValveOpen (
    uint16_t valve,
    bool valveOpen )
```

11.133.3.55 OnIsValveOpenInAnalogMode()

```
delegate void OnIsValveOpenInAnalogMode (
    uint16_t valve,
    bool valveOpen )
```

11.133.3.56 OnIsValveOpenInDigitalMode()

```
delegate void OnIsValveOpenInDigitalMode (
    uint16_t valve,
    bool valveOpen )
```

11.133.3.57 OnTableEntryChanged()

```
delegate void OnTableEntryChanged (
    uint16_t tableNumber )
```

11.133.3.58 SetActiveRunningTableNumber()

```
void SetActiveRunningTableNumber (
    uint32_t tableNumber )
```

Sets the number of the tanle that is active for running

Parameters

tableNumber	The table number

11.133.3.59 SetAnalogThresholdHigh()

```
void SetAnalogThresholdHigh (
    uint16_t valve,
    int32_t threshold )
```

Sets the upper threshold for the analog in port per valve

Parameters

valve	The valve number
threshold	The threshold in mV

11.133.3.60 SetAnalogThresholdLow() `void SetAnalogThresholdLow (`

```
uint16_t valve,
int32_t threshold )
```

Sets the lower threshold for the analog in port per valve

Parameters

valve	The valve number
threshold	The threshold in mV

11.133.3.61 SetCurrentEditTableNumber() `void SetCurrentEditTableNumber (`

```
uint32_t tableNumber )
```

Sets the number of the table that is current for editing

Parameters

tableNumber	The table number

11.133.3.62 SetDefault() `void SetDefault ( )`

Sets the settings of the valve controller to default

11.133.3.63 SetDigitalOutPortInvert() `void SetDigitalOutPortInvert (`

```
uint16_t digitalOutPort,
bool isInverted )
```

Map a valve to a digital out port

Parameters

digitalOutPort	The digital out port
isInverted	True if digital out is to be inverted

11.133.3.64 SetDigitalOutPortValve() `void SetDigitalOutPortValve (`

```
uint16_t digitalOutPort,
uint32_t valve )
```

Map a valve to a digital out port

Parameters

digitalOutPort	The digital out port
valve	The valve number

11.133.3.65 SetDigitalPortDirection() `void SetDigitalPortDirection (`

```
uint16_t port,
PortDirectionEnumNet direction )
```

Sets the direction of a digital port

Parameters

port	The port number
direction	the direction

11.133.3.66 SetDisplayMode() `void SetDisplayMode (`

```
WvcDisplayModeEnumNet DisplayMode,
int32_t lockTimeMs )
```

Writes the display mode

Parameters

DisplayMode	The display mode
lockTimeMs	Locks the display for ms

11.133.3.67 SetTableName() `void SetTableName (`

```
String^ tableName )
```

Set the name of the current protocol table

Parameters

tableName	The name of the table

11.133.3.68 SetTableStep() `void SetTableStep (`

```
uint16_t valve,
int32_t steps )
```

Skips the table protocol for a valve by steps

Parameters

valve	The valve number
steps	Number of steps

11.133.3.69 SetTableStepAll() `void SetTableStepAll (`

```
int32_t steps )
```

Skips the table protocol for all valves by steps

Parameters

steps	Number of steps

11.133.3.70 SetValveActive() `void SetValveActive (`

```
uint16_t valve,
int valveActive )
```

Sets the valve active/inactive state

Parameters

valve	The valve number
valveActive	The valve state

11.133.3.71 SetValveCurrent() `void SetValveCurrent (`

```
int16_t switch_current,
int16_t hold_current )
```

Sets the valve currents different from the default

Parameters

switch_current	The switch current (in DAC units); -1 sets the device default current
hold_current	The hold current (in DAC units); -1 sets the device default current

11.133.3.72 SetValveDigitalInInvert() `void SetValveDigitalInInvert (`

```
uint16_t valve,
bool isInverted )
```

Invert digital in

Parameters

valve	The valve number
isInverted	True if digital in is to be inverted

11.133.3.73 SetValveDigitalInPort() `void SetValveDigitalInPort (`

```
uint16_t valve,
uint32_t digitalInPort )
```

Map a digital in port to a valve

Parameters

valve	The valve number
digitalInPort	The digital in port

11.133.3.74 SetValveLedOn() `void SetValveLedOn (`

```
bool ledon )
```

Gets the LED state of the valve board

Parameters

ledon	The LED state

11.133.3.75 SetValveManualGroup() `void SetValveManualGroup (`

```
uint16_t valve,
int32_t valveManualGroup )
```

Sets the valve manual group

Parameters

valve	The valve number
valveManualGroup	The manual valve group

11.133.3.76 SetValveManualState() `void SetValveManualState (`

```
uint16_t valve,
int32_t valveManualState )
```

Sets the valve manual on/off state

Parameters

valve	The valve number
valveManualState	The manual valve state

11.133.3.77 SetValveMode() `void SetValveMode (`

```
uint16_t valve,
WvcValveModeEnumNet ValveMode )
```

Writes the valve mode

Parameters

valve	The valve number
ValveMode	The valve mode

11.133.3.78 SetValvesActiveMap() `void SetValvesActiveMap (`

```
uint32_t valvesActive )
```

Sets the valve active/inactive state

Parameters

valvesActive	The valves states

11.133.3.79 SetValvesManualStateMap() `void SetValvesManualStateMap (`

```
uint32_t valveaManualState )
```

Sets the valve manual on/off state

Parameters

valveaManualState	The manual valves states

11.133.3.80 SetValveTableEntry() void SetValveTableEntry (uint16_t *valve*, uint16_t *index*, uint32_t *duration*, bool *state*)

Write an entry to the valve protocol table

Parameters

valve	The valve number
index	The index in the table
duration	the duration in ms
state	the state

11.133.3.81 StoreValveTable() void StoreValveTable ( )

Store the current table in permanent memory

11.133.4 Event Documentation

11.133.4.1 GetActiveRunningTableNumberEvent OnGetActiveRunningTableNumber^ GetActiveRunningTableNumberEvent [add], [remove], [raise]

Event fires when the table number has changed

11.133.4.2 GetAnalogThresholdHighEvent OnGetAnalogThresholdHigh^ GetAnalogThresholdHighEvent [add], [remove], [raise]

Event fires when the threshold in mV for the valve number has changed

11.133.4.3 GetAnalogThresholdLowEvent OnGetAnalogThresholdLow^ GetAnalogThresholdLowEvent [add], [remove], [raise]

Event fires when the threshold in mV for the valve number has changed

11.133.4.4 GetAnalogVoltageEvent OnGetAnalogVoltage^ GetAnalogVoltageEvent [add], [remove], [raise]

Event fires when the voltage in mV has changed

11.133.4.5 GetCurrentNumberOfValvesEvent OnGetCurrentNumberOfValves^ GetCurrentNumberOf↩ ValvesEvent [add], [remove], [raise]

Event fires when the number of valves has changed

11.133.4.6 GetDigitalOutPortValveEvent OnGetDigitalOutPortValve^ GetDigitalOutPortValveEvent [add], [remove], [raise]

Event fires when the valve number for the digital out port has changed

11.133.4.7 GetDigitalPortDirectionEvent OnGetDigitalPortDirection^ GetDigitalPortDirectionEvent [add], [remove], [raise]

Event fires when the direction for the port number has changed

11.133.4.8 GetDisplayModeEvent OnGetDisplayMode^ GetDisplayModeEvent [add], [remove], [raise]

Event fires when the display mode has changed

11.133.4.9 GetTableNamebyIndexEvent OnGetTableNamebyIndex^ GetTableNamebyIndexEvent [add], [remove], [raise]

Event fires when the name of the table for the table number has changed

11.133.4.10 GetValveActiveEvent OnGetValveActive^ GetValveActiveEvent [add], [remove], [raise]

Event fires when the valve state for the valve number has changed

11.133.4.11 GetValveBoardRevisionEvent

OnGetValveBoardRevision^ GetValveBoardRevisionEvent [add], [remove], [raise]

Event fires when the revision code has changed

11.133.4.12 GetValveDigitalInPortEvent

OnGetValveDigitalInPort^ GetValveDigitalInPortEvent [add], [remove], [raise]

Event fires when the digital in port for the valve number has changed

11.133.4.13 GetValveLedOnEvent

OnGetValveLedOn^ GetValveLedOnEvent [add], [remove], [raise]

Event fires when the LED state has changed

11.133.4.14 GetValveManualGroupEvent

OnGetValveManualGroup^ GetValveManualGroupEvent [add], [remove], [raise]

Event fires when the manual valve group for the valve number has changed

11.133.4.15 GetValveManualStateEvent

OnGetValveManualState^ GetValveManualStateEvent [add], [remove], [raise]

Event fires when the manual valve state for the valve number has changed

11.133.4.16 GetValveModeEvent

OnGetValveMode^ GetValveModeEvent [add], [remove], [raise]

Event fires when the valve mode for the valve number has changed

11.133.4.17 IsDigitalOutPortInvertedEvent

OnIsDigitalOutPortInverted^ IsDigitalOutPortInverted↩Event [add], [remove], [raise]

Event fires when is inverted for the digital out port has changed

11.133.4.18 IsValveDigitalInInvertedEvent

`OnIsValveDigitalInInverted^ IsValveDigitalInInvertedEvent [add], [remove], [raise]`

Event fires when is inverted for the valve number has changed

11.133.4.19 IsValveOpenEvent

`OnIsValveOpen^ IsValveOpenEvent [add], [remove], [raise]`

Event fires when is open for the valve number has changed

11.133.4.20 IsValveOpenInAnalogModeEvent

`OnIsValveOpenInAnalogMode^ IsValveOpenInAnalogModeEvent [add], [remove], [raise]`

Event fires when is open for the valve number has changed

11.133.4.21 IsValveOpenInDigitalModeEvent

`OnIsValveOpenInDigitalMode^ IsValveOpenInDigitalModeEvent [add], [remove], [raise]`

Event fires when is open for the valve number has changed

11.133.4.22 TableEntryChangedEvent

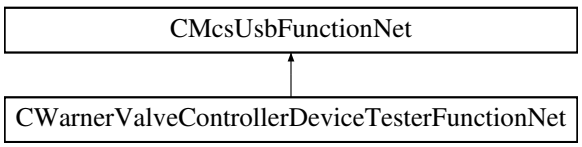
`OnTableEntryChanged^ TableEntryChangedEvent [add], [remove], [raise]`

Event fires when an entry of a table changed

11.134 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^ mcsusb, CMcsUsbFunctionPointerContainer^ pWarnerValveControllerDeviceTesterFunctionPointerContainer)

  Initializes a new instance of the CWarnerValveControllerDeviceTesterFunctionNet class.
- CWarnerValveControllerDeviceTesterFunctionNet (CMcsUsbNet^ 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

11.134.1 Detailed Description

CWarnerValveControllerDeviceTesterFunctionNet is the class to access the functions for the Warner Valve Controller Device Tester

11.134.2 Constructor & Destructor Documentation

11.134.2.1 CWarnerValveControllerDeviceTesterFunctionNet() [1/2]

```
CWarnerValveControllerDeviceTesterFunctionNet
(
            CMcsUsbNet^ mcsusb,
            CMcsUsbFunctionPointerContainer^ pWarnerValveControllerDeviceTesterFunction↩
PointerContainer )
```

Initializes a new instance of the CWarnerValveControllerDeviceTesterFunctionNet class.

11.134.2.2 CWarnerValveControllerDeviceTesterFunctionNet() [2/2]

```
CWarnerValveControllerDeviceTesterFunctionNet
(
            CMcsUsbNet^ mcsusb )
```

11.134.2.3 ~CWarnerValveControllerDeviceTesterFunctionNet() virtual ~CWarnerValveControllerDeviceTesterFunctio ( ) [virtual]

11.134.2.4 "!CWarnerValveControllerDeviceTesterFunctionNet() !CWarnerValveControllerDeviceTesterFunctionNet ( )

11.134.3 Member Function Documentation

11.134.3.1 GetIO() uint32_t GetIO ( )

Gets the output from the io ports

Returns

The manual valves states

11.134.3.2 GetSync() uint32_t GetSync ( )

Gets the output from the sync port

Returns

The sync state

11.134.3.3 SetADC() void SetADC (uint32_t *onoff*)

Sets the ADC port of the tester

Parameters

onoff	The port state

11.134.3.4 SetIO() void SetIO (uint32_t *io*)

Sets the input to the io ports

Parameters

io	The manual valves states

11.134.3.5 SetIODirection() `void SetIODirection ( int32_t direction )`

Sets the direction of the IO test ports

Parameters

direction	The 16bit direction map: 1=IN 0=OUT

11.134.3.6 SetTrigger() `void SetTrigger ( uint32_t trigger )`

Sets the input to the trigger port

Parameters

trigger	The trigger state

11.134.3.7 SetTriggerSyncDirection() `void SetTriggerSyncDirection ( uint32_t direction )`

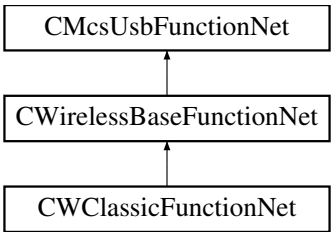
Sets the direction of the trigger/sync test port

Parameters

direction	The direction: 1=IN 0=OUT

11.135 CWClassicFunctionNet Class Reference

Inheritance diagram for CWClassicFunctionNet:



Public Member Functions

- CWClassicFunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ wClassicFuntion↩ PointerContainer)
- CWClassicFunctionNet (CMcsUsbNet^ mcsusb)
- uint32_t ResetChannelmap (unsigned int virtualDevice)
- uint32_t SetChannelmap (unsigned char position, unsigned char channel, unsigned int Device)
- void SetHWSelectedChannels (array< bool >^ channels, unsigned int Device)
- void SetRFLostBehaviour (uint8_t stoponfailure, unsigned int Device)
- void SetHeadstageOnOff (uint16_t onoff)
- USHORT GetHeadstageOnOff ()
- void SetRFFrequencyHeadstage (uint8_t receiver_nb, unsigned short frequency)
- unsigned short GetRFFrequencyHeadstage (uint8_t receiver_nb)
- void SetRFFrequencyReceiver (uint8_t receiver_nb, uint8_t configuration, unsigned short frequency)
- void SetRFFrequencyReceiverEeprom (uint8_t receiver_nb, uint8_t configuration, unsigned short frequency)
- unsigned short GetRFFrequencyReceiver (uint8_t receiver_nb, uint8_t configuration)
- void SetSerialNumberHeadstage (unsigned short number)
- unsigned short GetSerialNumberHeadstage ()
- void SetSelectedHeadstage (uint8_t number)
- uint8_t GetSelectedHeadstage ()
- void ScanForHeadstages ()
- uint8_t GetScanHeadstagesResult (int max_wait_for_ms)
- void SetFilterParametersHeadstage (unsigned short index, array< int >^ buffer)
- array< int > ^ GetFilterParametersHeadstage (unsigned short index)
- bool GetHasRedLedHeadstage ()
- void SetHasChecksum (unsigned int has, unsigned int Device)
- unsigned int GetHasChecksum (unsigned int Device)
- void SetResetFilter (unsigned int reset, unsigned int Device)
- unsigned int GetResetFilter (unsigned int Device)
- void SetWPAType (unsigned short type, unsigned int Device)
- unsigned short GetWPAType (unsigned int Device)
- void SetWPADebugMode (unsigned int mode, unsigned int Device)
- unsigned int GetWPADebugMode (unsigned int Device)
- void SetRFPower (unsigned short power)
- unsigned short GetRFPower ()
- unsigned int GetRFConnectionStatus ()

Additional Inherited Members

11.135.1 Constructor & Destructor Documentation

11.135.1.1 CWClassicFunctionNet() [1/2] CWClassicFunctionNet (CMcsUsbNet^ *mcsusb*, CMcsUsbFunctionPointerContainer^ *wClassicFuntionPointerContainer*)

11.135.1.2 CWClassicFunctionNet() [2/2] CWClassicFunctionNet (CMcsUsbNet^ *mcsusb*)

11.135.2 Member Function Documentation

11.135.2.1 GetFilterParametersHeadstage() array<int> ^ GetFilterParametersHeadstage (unsigned short *index*)

11.135.2.2 GetHasChecksum() unsigned int GetHasChecksum (unsigned int *Device*)

11.135.2.3 GetHasRedLedHeadstage() bool GetHasRedLedHeadstage ( )

11.135.2.4 GetHeadstageOnOff() USHORT GetHeadstageOnOff ( )

11.135.2.5 GetResetFilter() unsigned int GetResetFilter (unsigned int *Device*)

11.135.2.6 GetRFConnectionStatus() unsigned int GetRFConnectionStatus ( )

11.135.2.7 GetRFFrequencyHeadstage() unsigned short GetRFFrequencyHeadstage (uint8_t *receiver_nb*)

11.135.2.8 GetRFFrequencyReceiver()

```
unsigned short GetRFFrequencyReceiver (
    uint8_t receiver_nb,
    uint8_t configuration )
```

11.135.2.9 GetRFPower()

```
unsigned short GetRFPower ( )
```

11.135.2.10 GetScanHeadstagesResult()

```
uint8_t GetScanHeadstagesResult (
    int max_wait_for_ms )
```

11.135.2.11 GetSelectedHeadstage()

```
uint8_t GetSelectedHeadstage ( )
```

11.135.2.12 GetSerialNumberHeadstage()

```
unsigned short GetSerialNumberHeadstage ( )
```

11.135.2.13 GetWPADebugMode()

```
unsigned int GetWPADebugMode (
    unsigned int Device )
```

11.135.2.14 GetWPAType()

```
unsigned short GetWPAType (
    unsigned int Device )
```

11.135.2.15 ResetChannelmap()

```
uint32_t ResetChannelmap (
    unsigned int virtualDevice )
```

11.135.2.16 ScanForHeadstages()

```
void ScanForHeadstages ( )
```

11.135.2.17 SetChannelmap()

```
uint32_t SetChannelmap (
    unsigned char position,
    unsigned char channel,
    unsigned int Device )
```

11.135.2.18 SetFilterParametersHeadstage()

```
void SetFilterParametersHeadstage (
    unsigned short index,
    array< int >^ buffer )
```

11.135.2.19 SetHasChecksum()

```
void SetHasChecksum (
    unsigned int has,
    unsigned int Device )
```

11.135.2.20 SetHeadstageOnOff()

```
void SetHeadstageOnOff (
    uint16_t onoff )
```

11.135.2.21 SetHWSelectedChannels()

```
void SetHWSelectedChannels (
    array< bool >^ channels,
    unsigned int Device )
```

11.135.2.22 SetResetFilter()

```
void SetResetFilter (
    unsigned int reset,
    unsigned int Device )
```

11.135.2.23 SetRFFrequencyHeadstage()

```
void SetRFFrequencyHeadstage (
    uint8_t receiver_nb,
    unsigned short frequency )
```

11.135.2.24 SetRFFrequencyReceiver()

```
void SetRFFrequencyReceiver (
    uint8_t receiver_nb,
    uint8_t configuration,
    unsigned short frequency )
```

11.135.2.25 SetRFFrequencyReceiverEeprom()

```
void SetRFFrequencyReceiverEeprom (
    uint8_t receiver_nb,
    uint8_t configuration,
    unsigned short frequency )
```

11.135.2.26 SetRFLostBehaviour()

```
void SetRFLostBehaviour (
    uint8_t stoponfailure,
    unsigned int Device )
```

11.135.2.27 SetRFPower()

```
void SetRFPower (
    unsigned short power )
```

11.135.2.28 SetSelectedHeadstage()

```
void SetSelectedHeadstage (
    uint8_t number )
```

11.135.2.29 SetSerialNumberHeadstage()

```
void SetSerialNumberHeadstage (
    unsigned short number )
```

11.135.2.30 SetWPADebugMode()

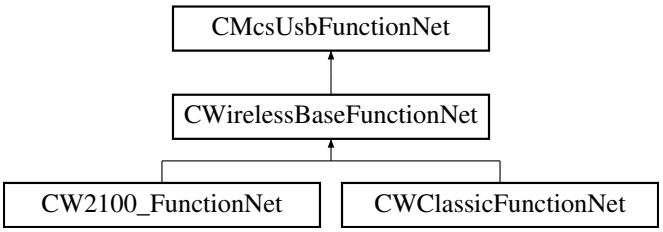
```
void SetWPADebugMode (
    unsigned int mode,
    unsigned int Device )
```

11.135.2.31 SetWPAType()

```
void SetWPAType (
    unsigned short type,
    unsigned int Device )
```

11.136 CWirelessBaseFunctionNet Class Reference

Inheritance diagram for CWirelessBaseFunctionNet:



Public Member Functions

- CWirelessBaseFunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ mcsusbfunction)

Static Public Member Functions

- static String ^ CreateWirelessHeadstageSerialNumberString (unsigned short ID)

Additional Inherited Members

11.136.1 Constructor & Destructor Documentation

11.136.1.1 CWirelessBaseFunctionNet() `CWirelessBaseFunctionNet (`

```
CMcsUsbNet^ mcsusb,
CMcsUsbFunctionPointerContainer^ mcsusbfunction )
```

11.136.2 Member Function Documentation

11.136.2.1 CreateWirelessHeadstageSerialNumberString() `static String ^ CreateWirelessHeadstageSerialNumberString (`

```
unsigned short ID )  [static]
```

11.137 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

11.137.1 Detailed Description

Device Id.

11.137.2 Constructor & Destructor Documentation

11.137.2.1 DeviceIdNet() [1/3] DeviceIdNet ( )

11.137.2.2 DeviceIdNet() [2/3] DeviceIdNet (

```
VendorIdEnumNet vendor,
ProductIdEnumNet product,
int bcd,
McsBusTypeEnumNet bustype )
```

11.137.2.3 DeviceIdNet() [3/3] DeviceIdNet (

```
DeviceIdNet% deviceId )
```

11.137.3 Member Function Documentation

11.137.3.1 operator=() DeviceIdNet operator= (

```
DeviceIdNet% deviceId )
```

11.137.4 Member Data Documentation

11.137.4.1 BcdDevice int BcdDevice

11.137.4.2 BusType McsBusTypeEnumNet BusType

11.137.4.3 IdProduct ProductIdEnumNet IdProduct

11.137.4.4 IdVendor VendorIdEnumNet IdVendor

11.138 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]`

11.138.1 Constructor & Destructor Documentation

11.138.1.1 DigitalSource() [1/2] DigitalSource ( )

11.138.1.2 DigitalSource() [2/2] DigitalSource (digitalsourceenum *source*)

11.138.2 Member Function Documentation

11.138.2.1 MaxBitNumber() [1/2] int MaxBitNumber ( )

11.138.2.2 MaxBitNumber() [2/2] int MaxBitNumber (digitalsourceenum *Source*)

11.138.2.3 MaxBitNumberStatic() static int MaxBitNumberStatic (digitalsourceenum *Source*) [static]

11.138.2.4 size() `static int size ( ) [static]`

11.138.3 Property Documentation

11.138.3.1 Source `digitalsourceenum Source [get], [set]`

11.139 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]`

11.139.1 Constructor & Destructor Documentation

11.139.1.1 DigitalSourceGeneral() [1/2] `DigitalSourceGeneral ( Type^ type )`

11.139.1.2 DigitalSourceGeneral() [2/2] `DigitalSourceGeneral ( Type^ type, int Source )`

11.139.2 Member Function Documentation

11.139.2.1 MaxBitNumber() [1/3] int MaxBitNumber ( )

11.139.2.2 MaxBitNumber() [2/3] int MaxBitNumber (int *Source*)

11.139.2.3 MaxBitNumber() [3/3] static int MaxBitNumber (Type^ *type*, int *Source*) [static]

11.139.2.4 size() static int size (Type^ *type*) [static]

11.139.3 Property Documentation

11.139.3.1 Source int Source [get], [set]

11.140 DriverVersionNet Class Reference

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

Public Member Functions

- DriverVersionNet ()
  Contructor.
- ~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)

11.140.1 Detailed Description

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

11.140.2 Constructor & Destructor Documentation

11.140.2.1 DriverVersionNet() `DriverVersionNet ( )`

Contructor.

11.140.2.2 ~DriverVersionNet() `~DriverVersionNet ( )`

Destructor.

11.140.3 Member Function Documentation

11.140.3.1 DriverVersionNet::FormatVersion()

```
static String ^ DriverVersionNet::FormatVersion (
    unsigned int v )  [static]
```

11.140.3.2 GetDestinationCode()

```
CFirmwareDestinationNet GetDestinationCode (
    unsigned int index )
```

Get CFirmwareDestinationNet.

Parameters

index	by index of firmware destination

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

Get firmware destination name.

Parameters

dest	by CFirmwareDestionationNet

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

Get firmware destination name.

Parameters

index	by index of firmware destination

11.140.3.5 GetMajor() [1/2] unsigned int GetMajor (CFirmwareDestinationNet *dest*)

Get the major version number of firmware destination.

Parameters

dest	by CFirmwareDestionationNet

11.140.3.6 GetMajor() [2/2] unsigned int GetMajor (unsigned int *index*)

Get the major version number of firmware destination.

Parameters

index	by index of firmware destination

11.140.3.7 GetMinor() [1/2] unsigned int GetMinor (CFirmwareDestinationNet *dest*)

Get the minor version number of firmware destination.

Parameters

dest	by CFirmwareDestionationNet

11.140.3.8 GetMinor() [2/2] unsigned int GetMinor (unsigned int *index*)

Get the minor version number of firmware destination.

Parameters

index	by index of firmware destination

11.140.3.9 GetNumEntries() unsigned int GetNumEntries ( )

Get the number of available firmware destinations.

11.140.3.10 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

dest	by CFirmwareDestionationNet

11.140.3.11 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

index	by index of firmware destination

11.140.3.12 GetStatus() [1/2] unsigned int GetStatus (CFirmwareDestinationNet *dest*)

Get status of firmware destination.

Parameters

dest	by CFirmwareDestionationNet

11.140.3.13 GetStatus() [2/2] unsigned int GetStatus (unsigned int *index*)

Get status of firmware destination.

Parameters

index	by index of firmware destination

11.140.3.14 GetVersionInt() [1/2] unsigned int GetVersionInt (CFirmwareDestinationNet *dest*)

Get the version number of firmware destination (major in high word, minor in low word)

Parameters

dest	by CFirmwareDestionationNet

11.140.3.15 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

index	by index of firmware destination

11.140.3.16 GetVersionString() [1/2] String ^ GetVersionString (CFirmwareDestinationNet *dest*)

Get the version as a string in the format Major.Minor.

Parameters

dest	by CFirmwareDestionationNet

11.140.3.17 GetVersionString() [2/2] String ^ GetVersionString (unsigned int *index*)

Get the version as a string in the format Major.Minor.

Parameters

index	by index of firmware

11.141 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")

11.141.1 Member Data Documentation

11.141.1.1 Altera `String ^ Altera = gcnew String( "Altera" ) [static]`

11.141.1.2 Bootstrap `String ^ Bootstrap = gcnew String( "Bootstrap" ) [static]`

11.141.1.3 BUS1_MCSBUS1 `String ^ BUS1_MCSBUS1 = gcnew String( "Bus1McsBus1" ) [static]`

11.141.1.4 BUS1_MCSBUS2 `String ^ BUS1_MCSBUS2 = gcnew String( "Bus1McsBus2" ) [static]`

11.141.1.5 DSP `String ^ DSP = gcnew String( "DSP" ) [static]`

11.141.1.6 FPGA2 `String ^ FPGA2 = gcnew String( "FPGA2" ) [static]`

11.141.1.7 FPGA3 `String ^ FPGA3 = gcnew String( "FPGA3" ) [static]`

11.141.1.8 FPGA4 `String ^ FPGA4 = gcnew String( "FPGA4" ) [static]`

11.141.1.9 FPGA5 `String ^ FPGA5 = gcnew String( "FPGA5" ) [static]`

11.141.1.10 FPGA6

```
String ^ FPGA6 = gcnew String( "FPGA6" )  [static]
```

11.141.1.11 MCSBUS1

```
String ^ MCSBUS1 = gcnew String( "McsBus1" )  [static]
```

11.141.1.12 MCSBUS10

```
String ^ MCSBUS10 = gcnew String( "McsBus10" )  [static]
```

11.141.1.13 MCSBUS11

```
String ^ MCSBUS11 = gcnew String( "McsBus11" )  [static]
```

11.141.1.14 MCSBUS12

```
String ^ MCSBUS12 = gcnew String( "McsBus12" )  [static]
```

11.141.1.15 MCSBUS13

```
String ^ MCSBUS13 = gcnew String( "McsBus13" )  [static]
```

11.141.1.16 MCSBUS2

```
String ^ MCSBUS2 = gcnew String( "McsBus2" )  [static]
```

11.141.1.17 MCSBUS3

```
String ^ MCSBUS3 = gcnew String( "McsBus3" )  [static]
```

11.141.1.18 MCSBUS4

```
String ^ MCSBUS4 = gcnew String( "McsBus4" )  [static]
```

11.141.1.19 MCSBUS5

```
String ^ MCSBUS5 = gcnew String( "McsBus5" )  [static]
```

11.141.1.20 MCSBUS6

```
String ^ MCSBUS6 = gcnew String( "McsBus6" )  [static]
```

11.141.1.21 MCSBUS7

```
String ^ MCSBUS7 = gcnew String( "McsBus7" )  [static]
```

11.141.1.22 MCSBUS8

```
String ^ MCSBUS8 = gcnew String( "McsBus8" )  [static]
```

11.141.1.23 MCSBUS9

```
String ^ MCSBUS9 = gcnew String( "McsBus9" )  [static]
```

11.141.1.24 MCU1

```
String ^ MCU1 = gcnew String( "MCU1" )  [static]
```

11.141.1.25 PIC

```
String ^ PIC = gcnew String( "PIC" )  [static]
```

11.141.1.26 PIC2

```
String ^ PIC2 = gcnew String( "PIC2" )  [static]
```

11.141.1.27 PIC3

```
String ^ PIC3 = gcnew String( "PIC3" )  [static]
```

11.141.1.28 PIC4

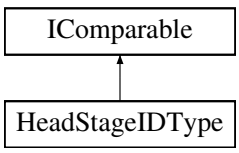
```
String ^ PIC4 = gcnew String( "PIC4" )  [static]
```

11.141.1.29 USB

```
String ^ USB = gcnew String( "USB" )  [static]
```

11.142 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 HasOptoCurrentMessurement [get]

11.142.1 Member Enumeration Documentation

11.142.1.1 HeadstageTypeEnum enum HeadstageTypeEnum [strong]

Enumerator

Unknown	
MeasuringOnly	
OpticalStimulation	
ElectricalStimulation	

11.142.2 Constructor & Destructor Documentation

11.142.2.1 HeadStageIDType() HeadStageIDType (

```
unsigned int entry,
CW2100_FunctionNet^ device )
```

11.142.3 Member Function Documentation

11.142.3.1 CompareTo() virtual Int32 CompareTo (

```
Object^ obj )  [virtual]
```

11.142.3.2 Equals() virtual bool Equals (

```
Object^ obj )  [override], [virtual]
```

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

11.142.4 Property Documentation

11.142.4.1 Entry unsigned int Entry [get]

11.142.4.2 HasIMU bool HasIMU [get]

11.142.4.3 HasOptoCurrentMessurement bool HasOptoCurrentMessurement [get]

11.142.4.4 HeadstageType HeadstageTypeEnum HeadstageType [get]

11.142.4.5 ID unsigned short ID [get]

11.142.4.6 NumberOfAnalogChannels `int NumberOfAnalogChannels [get]`

11.142.4.7 NumberOfStimulationChannels `int NumberOfStimulationChannels [get]`

11.142.4.8 SN `System:: String^ SN [get]`

11.142.4.9 StimulusParameters `W2100_StimulusParametersNet^ StimulusParameters [get]`

11.142.4.10 Type `System:: String^ Type [get]`

11.142.4.11 TypeValue `unsigned int TypeValue [get]`

11.142.4.12 UserDefinedName `System:: String^ UserDefinedName [get]`

11.142.4.13 Valid `bool Valid [get]`

11.142.4.14 W16IsW14 `bool W16IsW14 [get]`

11.143 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]

11.143.1 Constructor & Destructor Documentation

11.143.1.1 HeadstageIDTypeObject() HeadstageIDTypeObject (HeadStageIDType^ *idType*)

11.143.2 Member Function Documentation

11.143.2.1 Equals() virtual bool Equals (Object^ *obj*) [override], [virtual]

11.143.2.2 GetHashCode() virtual int GetHashCode ( ) [override], [virtual]

11.143.2.3 ToString() virtual String ^ ToString ( ) [override], [virtual]

11.143.3 Member Data Documentation

11.143.3.1 _AdditionalText String ^ _AdditionalText

11.143.3.2 _IdType HeadStageIDType ^ _IdType

11.143.4 Property Documentation

11.143.4.1 AdditionalText `String^ AdditionalText [get], [set]`

11.143.4.2 IdType `HeadStageIDType^ IdType [get]`

11.144 HeadStageIDTypeState Class Reference

Properties

- unsigned int State `[get]`
- HeadStageIDType^ IdType `[get]`
- bool ControlState `[get]`
- bool DataState `[get]`

11.144.1 Property Documentation

11.144.1.1 ControlState `bool ControlState [get]`

11.144.1.2 DataState `bool DataState [get]`

11.144.1.3 IdType `HeadStageIDType^ IdType [get]`

11.144.1.4 State `unsigned int State [get]`

11.145 mkfilterNet Class Reference

Static Public Member Functions

- static int mkfilter (String^ filtertype, double value, String^ passtype, int order, double alpha1, double alpha2, [System::Runtime::InteropServices::Out] array< double >^% xcoeffs, [System::Runtime::InteropServices::Out] array< double >^% ycoeffs)
- static int mkfilter_MCS (int SamplesPerSecond, double R1, double R2, double C, double Amplification, double Correction, [System::Runtime::InteropServices::Out] array< double >^% xcoeffs, [System::Runtime::InteropServices::Out] array< double >^% ycoeffs)
- static int mkfilter_MCS (int SamplesPerSecond, double R1, double R2, double C, double Correction, [System::Runtime::InteropServices::Out] array< double >^% xcoeffs, [System::Runtime::InteropServices::Out] array< double >^% ycoeffs)
- static int mkfilter_MCS_k (int SamplesPerSecond, double R1, double R2, double C, double Amplification, double Correction, [System::Runtime::InteropServices::Out] array< double >^% coeffs)
- static int mkfilter_MCS_k (int SamplesPerSecond, double R1, double R2, double C, double Correction, [System::Runtime::InteropServices::Out] array< double >^% coeffs)
- static void mkfilter_coef_in_one_set (int n, [System::Runtime::InteropServices::In] array< double >^ xcoeffs, [System::Runtime::InteropServices::In] array< double >^ ycoeffs, [System::Runtime::InteropServices::Out] array< double >^% out_coeffs)
- static void mkfilter_scale_coef_in_one_set (int n, double scale, [System::Runtime::InteropServices::In] array< double >^ xcoeffs, [System::Runtime::InteropServices::In] array< double >^ ycoeffs, [System::Runtime::InteropServices::Out] array< double >^% out_coeffs)
- static void mkfilter_normalize_coeffs_short (short maxvalue, [System::Runtime::InteropServices::In] array< double >^ coeffs, [System::Runtime::InteropServices::Out] array< short >^% out_coeffs)
- static void mkfilter_normalize_coeffs_int (int maxvalue, [System::Runtime::InteropServices::In] array< double >^ coeffs, [System::Runtime::InteropServices::Out] array< int >^% out_coeffs)
- static void mkfilter_normalize_scale_coeffs_int (int maxvalue, [System::Runtime::InteropServices::In] array< double >^ coeffs, [System::Runtime::InteropServices::Out] array< int >^% out_coeffs)
- static double mkfilter_highpass_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)

11.145.1 Member Function Documentation

11.145.1.1 mkfilter()

```
static int mkfilter (
    String^ filtertype,
    double value,
    String^ passtype,
    int order,
    double alpha1,
    double alpha2,
    [System::Runtime::InteropServices::Out] array< double >^% xcoeffs,
    [System::Runtime::InteropServices::Out] array< double >^% ycoeffs ) [static]
```

11.145.1.2 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]
```

11.145.1.3 mkfilter_highpass_coeff()

```
static double mkfilter_highpass_coeff (
    int SamplesPerSecond,
    double Frequency ) [static]
```

11.145.1.4 mkfilter_highpass_frequency_from_coeff()

```
static double mkfilter_highpass_frequency_from_coeff (
    int SamplesPerSecond,
    double coeff ) [static]
```

11.145.1.5 mkfilter_highpass_frequency_from_k()

```
static double mkfilter_highpass_frequency_from_k (
    int SamplesPerSecond,
    double k ) [static]
```

11.145.1.6 mkfilter_highpass_k()

```
static double mkfilter_highpass_k (
    int SamplesPerSecond,
    double Frequency ) [static]
```

11.145.1.7 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]
```

11.145.1.8 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]
```

11.145.1.9 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]
```

11.145.1.10 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]
```

11.145.1.11 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]
```

11.145.1.12 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]
```

11.145.1.13 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]
```

11.145.1.14 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]
```

11.146 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)

11.146.1 Member Function Documentation

11.146.1.1 FindReferencePhase0() [1/2]

```
void FindReferencePhase0 (
    unsigned char busaddress,
    char axes )
```

11.146.1.2 FindReferencePhase0() [2/2]

```
void FindReferencePhase0 (
    unsigned char busaddress,
    char axes,
    int timeout )
```

11.146.1.3 GetAxisConfig()

```
unsigned int GetAxisConfig (
    unsigned char busaddress,
    char axes )
```

11.146.1.4 GetHWConfig()

```
unsigned int GetHWConfig ( )
```

11.146.1.5 GetHWRevision()

```
unsigned int GetHWRevision ( )
```

11.146.1.6 GetMaxNoPressure()

```
unsigned int GetMaxNoPressure ( )
```

11.146.1.7 GetMaxNoPressureWaitTime()

```
unsigned int GetMaxNoPressureWaitTime ( )
```

11.146.1.8 GetMaxPressureWaitTime()

```
unsigned int GetMaxPressureWaitTime ( )
```

11.146.1.9 GetMinNoPressureWaitTime()

```
unsigned int GetMinNoPressureWaitTime ( )
```

11.146.1.10 GetMinPressure()

```
unsigned int GetMinPressure ( )
```

11.146.1.11 GetMinPressureWaitTime()

```
unsigned int GetMinPressureWaitTime ( )
```

11.146.1.12 GetParameter() [1/2]

```
void GetParameter (
    unsigned short command,
    unsigned short index,
    [System::Runtime::InteropServices::Out] unsigned int% value )
```

11.146.1.13 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 )
```

11.146.1.14 GetPhases()

```
void GetPhases (
    unsigned char busaddress,
    char axes,
    [System::Runtime::InteropServices::Out] unsigned short% phase0,
    [System::Runtime::InteropServices::Out] unsigned short% lastphase )
```

11.146.1.15 GetSearchReferenceFastAccel()

```
unsigned short GetSearchReferenceFastAccel (
    unsigned char busaddress,
    char axes )
```

11.146.1.16 GetSearchReferenceFastSpeed()

```
unsigned short GetSearchReferenceFastSpeed (
    unsigned char busaddress,
    char axes )
```

11.146.1.17 GetSearchReferenceFineAccel()

```
unsigned short GetSearchReferenceFineAccel (
    unsigned char busaddress,
    char axes )
```

11.146.1.18 GetSearchReferenceFineSpeed()

```
unsigned short GetSearchReferenceFineSpeed (
    unsigned char busaddress,
    char axes )
```

11.146.1.19 GetSearchReferenceMethod()

```
unsigned int GetSearchReferenceMethod (
    unsigned char busaddress,
    char axes )
```

11.146.1.20 GetSearchReferenceMoveOut()

```
int GetSearchReferenceMoveOut (
    unsigned char busaddress,
    char axes )
```

11.146.1.21 GetSearchReferenceOffsetPos()

```
int GetSearchReferenceOffsetPos (
    unsigned char busaddress,
    char axes )
```

11.146.1.22 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

index	address offset of parameter; range: 0..15
value	data buffer

11.146.1.23 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

index	address offset of parameter; range: 0..15
value	data buffer

11.146.1.24 HasRef()

```
unsigned char HasRef (
    unsigned char busaddress,
    char axes )
```

11.146.1.25 SetAxisConfig()

```
void SetAxisConfig (
    unsigned char busaddress,
    char axes,
    unsigned int config )
```

11.146.1.26 SetHWConfig()

```
void SetHWConfig (
    unsigned int config )
```

11.146.1.27 SetHWRevision()

```
void SetHWRevision (
    unsigned int revision )
```

11.146.1.28 SetMaxNoPressure()

```
void SetMaxNoPressure (
    unsigned int pressure )
```

11.146.1.29 SetMaxNoPressureWaitTime()

```
void SetMaxNoPressureWaitTime (
    unsigned int t )
```

11.146.1.30 SetMaxPressureWaitTime()

```
void SetMaxPressureWaitTime (
    unsigned int t )
```

11.146.1.31 SetMinNoPressureWaitTime()

```
void SetMinNoPressureWaitTime (
    unsigned int t )
```

11.146.1.32 SetMinPressure()

```
void SetMinPressure (
    unsigned int pressure )
```

11.146.1.33 SetMinPressureWaitTime()

```
void SetMinPressureWaitTime (
    unsigned int t )
```

11.146.1.34 SetParameter() [1/2]

```
void SetParameter (
    unsigned short command,
    unsigned short index,
    unsigned int value )
```

11.146.1.35 SetParameter() [2/2]

```
void SetParameter (
    unsigned short command,
    unsigned short index,
    unsigned int value1,
    unsigned int value2 )
```

11.146.1.36 SetSearchReferenceFastAccel()

```
void SetSearchReferenceFastAccel (
    unsigned char busaddress,
    char axes,
    unsigned short accel )
```

11.146.1.37 SetSearchReferenceFastSpeed()

```
void SetSearchReferenceFastSpeed (
    unsigned char busaddress,
    char axes,
    unsigned short speed )
```

11.146.1.38 SetSearchReferenceFineAccel()

```
void SetSearchReferenceFineAccel (
    unsigned char busaddress,
    char axes,
    unsigned short accel )
```

11.146.1.39 SetSearchReferenceFineSpeed()

```
void SetSearchReferenceFineSpeed (
    unsigned char busaddress,
    char axes,
    unsigned short speed )
```

11.146.1.40 SetSearchReferenceMethod()

```
void SetSearchReferenceMethod (
    unsigned char busaddress,
    char axes,
    unsigned int method )
```

11.146.1.41 SetSearchReferenceMoveOut()

```
void SetSearchReferenceMoveOut (
    unsigned char busaddress,
    char axes,
    int move )
```

11.146.1.42 SetSearchReferenceOffsetPos()

```
void SetSearchReferenceOffsetPos (
    unsigned char busaddress,
    char axes,
    int offsetpos )
```

11.146.1.43 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

index	address offset of parameter; range: 0..15
value	data to be stored

11.146.1.44 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

index	address offset of parameter; range: 0..15
value	data to be stored

11.147 CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands Class Reference

Public Member Functions

- void FindReferencePhase0XY ()
- void FindReferencePhase0XY (int timeout)

11.147.1 Member Function Documentation

11.147.1.1 FindReferencePhase0XY() [1/2] void FindReferencePhase0XY ( )

11.147.1.2 FindReferencePhase0XY() [2/2] void FindReferencePhase0XY (int *timeout*)

11.148 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

11.148.1 Constructor & Destructor Documentation

11.148.1.1 s_FilterAttributesNet() s_FilterAttributesNet (s_FilterAttributes *attrib*)

11.148.2 Member Function Documentation

11.148.2.1 ToCpp() `s_FilterAttributes ToCpp ( )`

11.148.3 Member Data Documentation

11.148.3.1 CommaPositionA `uint32_t CommaPositionA`

11.148.3.2 CommaPositionB `uint32_t CommaPositionB`

11.148.3.3 PostCommaA `uint32_t PostCommaA`

11.148.3.4 PostCommaB `uint32_t PostCommaB`

11.148.3.5 PreCommaA `uint32_t PreCommaA`

11.148.3.6 PreCommaB `uint32_t PreCommaB`

11.149 CMeaAudioFunctionNet::s_setaudionet Struct Reference

Public Attributes

- int channel
- int amplification

11.149.1 Member Data Documentation

11.149.1.1 amplification `int amplification`

11.149.1.2 channel `int channel`

11.150 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]`

11.150.1 Constructor & Destructor Documentation

11.150.1.1 SidebandData() `SidebandData ( )`

11.150.1.2 ~SidebandData() `~SidebandData ( )`

Destructor: called by Dispose()

11.150.1.3 "!SidebandData() `!SidebandData ( )`

Finalizer: called by GC before collecting

11.150.2 Property Documentation

11.150.2.1 Duration `array< uint64_t>^ Duration [get]`

11.150.2.2 Sideband `array< int32_t>^ Sideband [get]`

11.151 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

11.151.1 Member Function Documentation

11.151.1.1 FromIntPtr() `static StgStatusNet ^ FromIntPtr ( IntPtr stgstatus ) [static]`

11.151.1.2 FromPtr() `static StgStatusNet ^ FromPtr ( stgstatus_t * stgstatus ) [static]`

11.151.2 Member Data Documentation

11.151.2.1 ListOfChangedTriggers `array<uint32_t> ^ ListOfChangedTriggers`

11.151.2.2 TiggerStatus `array<Stg200xTriggerStatusEnumNet> ^ TiggerStatus`

11.152 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]

11.152.1 Constructor & Destructor Documentation

11.152.1.1 StimulusDeviceDataAndUnrolledData() StimulusDeviceDataAndUnrolledData ( )

11.152.1.2 ~StimulusDeviceDataAndUnrolledData() ~StimulusDeviceDataAndUnrolledData ( )

Destructor: called by Dispose()

11.152.1.3 "!StimulusDeviceDataAndUnrolledData() !StimulusDeviceDataAndUnrolledData ( )

Finalizer: called by GC before collecting

11.152.2 Property Documentation

11.152.2.1 DeviceData array< uint8_t>^ DeviceData [get]

11.152.2.2 DeviceDataLength int DeviceDataLength [get]

11.152.2.3 UnrolledAmplitude array< int32_t>^ UnrolledAmplitude [get]

11.152.2.4 UnrolledDuration array< uint64_t>^ UnrolledDuration [get]

11.152.2.5 UnrolledSync `array< uint32_t>^ UnrolledSync [get]`

11.153 usbSetupPacket_t Class Reference

Public Attributes

- uint8_t bmRequestType
- uint8_t bRequest
- uint16_t wValue
- uint16_t wIndex
- uint16_t wLength

11.153.1 Member Data Documentation

11.153.1.1 bmRequestType `uint8_t bmRequestType`

11.153.1.2 bRequest `uint8_t bRequest`

11.153.1.3 wIndex `uint16_t wIndex`

11.153.1.4 wLength `uint16_t wLength`

11.153.1.5 wValue `uint16_t wValue`

11.154 W2100_StimulusParametersNet Struct Reference

Public Attributes

- int DACResolution
- int TimeResolutionInNanoSeconds
- int VoltageRangeInMicroVolt
- int VoltageResolutionInMicroVolt
- int CurrentRangeInNanoAmp
- int CurrentResolutionInNanoAmp

11.154.1 Member Data Documentation

11.154.1.1 CurrentRangeInNanoAmp `int CurrentRangeInNanoAmp`

11.154.1.2 CurrentResolutionInNanoAmp `int CurrentResolutionInNanoAmp`

11.154.1.3 DACResolution `int DACResolution`

11.154.1.4 TimeResolutionInNanoSeconds `int TimeResolutionInNanoSeconds`

11.154.1.5 VoltageRangeInMicroVolt `int VoltageRangeInMicroVolt`

11.154.1.6 VoltageResolutionInMicroVolt `int VoltageResolutionInMicroVolt`

Index

!CDacCalibrationFunctionNet
CDacCalibrationFunctionNet, 124
!CDigOutStimulatorFunctionNet
CDigOutStimulatorFunctionNet, 134
!CExternDTesterDeviceNet
CExternDTesterDeviceNet, 138
!CGrapheneFunctionNet
CGrapheneFunctionNet, 179
!CInterfaceboard2FunctionNet
CInterfaceboard2FunctionNet, 197
!CInterfaceboardFunctionNet
CInterfaceboardFunctionNet, 199
!CLIH3DeviceNet
CLIH3DeviceNet, 202
!CMEA2100x256FunctionNet
CMEA2100x256FunctionNet, 345
!CMcsUsbFunctionNet
CMcsUsbFunctionNet, 314
!CMcsUsbListNet
CMcsUsbListNet, 321
!CMcsUsbNet
CMcsUsbNet, 327
!CMeFunctionNet
CMeFunctionNet, 379
!CMeaCleanDeviceNet
CMeaCleanDeviceNet, 350
!CMeaCoatDeviceNet
CMeaCoatDeviceNet, 354
!CMultiBatteryChargerDeviceNet
CMultiBatteryChargerDeviceNet, 381
!CMultiwellCallbackFunctionNet
CMultiwellCallbackFunctionNet, 388
!CMultiwellDeviceNet
CMultiwellDeviceNet, 390
!CMultiwellOptoStimFunctionNet
CMultiwellOptoStimFunctionNet, 397
!CPPCFunctionNet
CPPCFunctionNet, 431
!CPedoterDeviceNet
CPedoterDeviceNet, 412
!CPositionIIDeviceNet
CPositionIIDeviceNet, 418
!CPositionImpDeviceNet
CPositionImpDeviceNet, 426
!CProgramPressureCurveNet
CProgramPressureCurveNet, 445
!CPulseGeneratorFunctionNet
CPulseGeneratorFunctionNet, 446
!CRFFunctionNet
CRFFunctionNet, 454
!CSCUFunctionNet
CSCUFunctionNet, 510
!CTEERFunctionNet
CTEERFunctionNet, 609
!CUsbDeviceConfigurationFunctionNet
CUsbDeviceConfigurationFunctionNet, 619
!CWarnerUssingDeviceNet
CWarnerUssingDeviceNet, 637
!CWarnerUssingFunctionNet
CWarnerUssingFunctionNet, 640
!CWarnerValveControllerDeviceNet
CWarnerValveControllerDeviceNet, 658
!CWarnerValveControllerDeviceTesterFunctionNet
CWarnerValveControllerDeviceTesterFunctionNet, 680
!SidebandData
CStimulusFunctionNet::SidebandData, 718
!StimulusDeviceDataAndUnrolledData
CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, 720
_AdditionalText
HeadstageIDTypeObject, 704
_IdType
HeadstageIDTypeObject, 704
~CCMOSMeaDeviceNet
CCMOSMeaDeviceNet, 118
~CChannelTestDeviceNet
CChannelTestDeviceNet, 105
~CCreateFilterNet
CCreateFilterNet, 121
~CDacCalibrationFunctionNet
CDacCalibrationFunctionNet, 124
~CDigOutStimulatorFunctionNet
CDigOutStimulatorFunctionNet, 134
~CExternDTesterDeviceNet
CExternDTesterDeviceNet, 138
~CFilterCoefficientsNet
CFilterCoefficientsNet, 140
~CFilterPropertyNet
CFilterPropertyNet, 146
~CFluidControlDeviceNet
CFluidControlDeviceNet, 148
~CGenericDevelopDeviceNet
CGenericDevelopDeviceNet, 162
~CGilsonDeviceNet
CGilsonDeviceNet, 174
~CGrapheneASICDeviceNet
CGrapheneASICDeviceNet, 175
~CGrapheneFunctionNet
CGrapheneFunctionNet, 178
~CInterfaceboard2FunctionNet
CInterfaceboard2FunctionNet, 197
~CInterfaceboardFunctionNet
CInterfaceboardFunctionNet, 199
~CLIH3DeviceNet
CLIH3DeviceNet, 202
~CMEA2100x256FunctionNet
CMEA2100x256FunctionNet, 345
~CMcsBusNet
CMcsBusNet, 248

~CMcsBus_AxisParametersNet
CMcsBus_AxisParametersNet, 210
~CMcsBus_ExtensionNet
CMcsBus_ExtensionNet, 211
~CMcsBus_FYIExtensionNet
CMcsBus_FYIExtensionNet, 212
~CMcsBus_MotorControlNet
CMcsBus_MotorControlNet, 217
~CMcsBus_SensorNet
CMcsBus_SensorNet, 233
~CMcsBus_TempSensorNet
CMcsBus_TempSensorNet, 242
~CMcsBus_VoltageModeNet
CMcsBus_VoltageModeNet, 244
~CMcsUsbDacqNet
CMcsUsbDacqNet, 258
~CMcsUsbFactoryNet
CMcsUsbFactoryNet, 306
~CMcsUsbFunctionNet
CMcsUsbFunctionNet, 314
~CMcsUsbListEntryNet
CMcsUsbListEntryNet, 315
~CMcsUsbListNet
CMcsUsbListNet, 321
~CMcsUsbNet
CMcsUsbNet, 327
~CMeFunctionNet
CMeFunctionNet, 379
~CMeaCleanDeviceNet
CMeaCleanDeviceNet, 350
~CMeaCoatDeviceNet
CMeaCoatDeviceNet, 354
~CMeaDeviceNet
CMeaDeviceNet, 360
~CMeaImpedanceDeviceNet
CMeaImpedanceDeviceNet, 373
~CMeaSwitchDeviceNet
CMeaSwitchDeviceNet, 376
~CMeaUSBDeviceNet
CMeaUSBDeviceNet, 378
~CMultiBatteryChargerDeviceNet
CMultiBatteryChargerDeviceNet, 381
~CMultiwellCallbackFunctionNet
CMultiwellCallbackFunctionNet, 388
~CMultiwellDeviceNet
CMultiwellDeviceNet, 390
~CMultiwellOptoStimFunctionNet
CMultiwellOptoStimFunctionNet, 397
~CNF_GenDeviceNet
CNF_GenDeviceNet, 401
~COkuvisionStimulatorDeviceNet
COkuvisionStimulatorDeviceNet, 407
~CPPCFunctionNet
CPPCFunctionNet, 431
~CPathIdentDeviceNet
CPathIdentDeviceNet, 411
~CPedoterDeviceNet
CPedoterDeviceNet, 412
~CPeristalticPumpDeviceNet
CPeristalticPumpDeviceNet, 413
~CPgaDeviceNet
CPgaDeviceNet, 414
~CPositionIIDeviceNet
CPositionIIDeviceNet, 418
~CPositionImpDeviceNet
CPositionImpDeviceNet, 426
~CProgramPressureCurveNet
CProgramPressureCurveNet, 444
~CPulseGeneratorFunctionNet
CPulseGeneratorFunctionNet, 446
~CRFFunctionNet
CRFFunctionNet, 454
~CRetinaLedDeviceNet
CRetinaLedDeviceNet, 452
~CRoboDeviceNet
CRoboDeviceNet, 479
~CRoboFluidDeviceNet
CRoboFluidDeviceNet, 493
~CSCUFunctionNet
CSCUFunctionNet, 510
~CSafeISDeviceNet
CSafeISDeviceNet, 504
~CStg200xBasicNet
CStg200xBasicNet, 530
~CStg200xDownloadNet
CStg200xDownloadNet, 574
~CSw2to64DeviceNet
CSw2to64DeviceNet, 592
~CTEERFunctionNet
CTEERFunctionNet, 609
~CTEERMachineDeviceNet
CTEERMachineDeviceNet, 617
~CTcxDeviceNet
CTcxDeviceNet, 596
~CUsbDeviceConfigurationFunctionNet
CUsbDeviceConfigurationFunctionNet, 619
~CWarnerUssingDeviceNet
CWarnerUssingDeviceNet, 637
~CWarnerUssingFunctionNet
CWarnerUssingFunctionNet, 640
~CWarnerValveControllerDeviceNet
CWarnerValveControllerDeviceNet, 658
~CWarnerValveControllerDeviceTesterFunctionNet
CWarnerValveControllerDeviceTesterFunctionNet, 679
~DriverVersionNet
DriverVersionNet, 692
~SidebandData
CStimulusFunctionNet::SidebandData, 718
~StimulusDeviceDataAndUnrolledData
CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, 720

A
CFilterCoefficientsNet, 141
Mcs::Usb, 70
AccelOnly

Mcs::Usb, 93
AdapterTypeEnumNet
Mcs::Usb, 52
AdditionalText
HeadstageIDTypeObject, 705
AddLoopEntry
CRetinaLedDeviceNet, 452
AddSelectedChannelsQueue
CMcsUsbDacqNet, 258–261
AddSoftwareKey
CMcsUsbNet, 327
AddTableEntry
CRetinaLedDeviceNet, 452
ALA_VC3
Mcs::Usb, 78, 92
ALA_VC3_DEVICE
Mcs::Usb, 64
ALTERA
Mcs::Usb, 56
Altera
FirmwareDestinationNames, 698
ALTERA_BASE
Mcs::Usb, 56
ALTERA_BOOTSTRAP
Mcs::Usb, 57
ALTERA_GOLD
Mcs::Usb, 56
ALTERA_TARGET1
Mcs::Usb, 57
ALTERA_TARGET2
Mcs::Usb, 57
ALTERA_TARGET3
Mcs::Usb, 57
AlwaysOn
Mcs::Usb, 81
Ampere
Mcs::Usb, 54, 92
amplification
CMeaAudioFunctionNet::s_setaudionet, 717
CW2100_FunctionNet::AudioChannelsNet, 97
AmplifierSettle
CIntanMea_FunctionNet, 195
AMS_Dongle
Mcs::Usb, 79
Analog
Mcs::Usb, 78, 95
AnalogGain
CMeaDeviceNet, 365
AnalogGroup
Mcs::Usb, 62
AnalogOut_DAC_Range_EnumNet
Mcs::Usb, 53
AnalogSource_HS1
Mcs::Usb, 53
AnalogSource_HS2
Mcs::Usb, 53
AnalogSource_IF
Mcs::Usb, 53
AnalogSourceEnumNet
Mcs::Usb, 53
AnalogUnitEnumNet
Mcs::Usb, 53
Any
Mcs::Usb, 78, 92
ApplyGains
CPgaDeviceNet, 415
AreTransistorVoltagesSet
CCMOSMea_FunctionNet, 108
Armed
Mcs::Usb, 86
ASMedia
Mcs::Usb, 91
AssociateToThis
CMcsUsbNet, 327
AudioTestChannelGroup
Mcs::Usb, 62, 72, 84, 94
AutomaticAnalogOut
CSCUFunctionNet, 510
Aux
Mcs::Usb, 64
AuxIn
Mcs::Usb, 57, 73, 74, 76, 84, 88, 95
AuxPort
Mcs::Usb, 60
Axes_I
CRoboDeviceNet, 487
Axes_X
CRoboDeviceNet, 487
Axes_Y
CRoboDeviceNet, 487
Axes_Z
CRoboDeviceNet, 488
Axis_I
CRoboDeviceNet, 488
Axis_X
CRoboDeviceNet, 488
Axis_Y
CRoboDeviceNet, 488
Axis_Z
CRoboDeviceNet, 488

B
CFilterCoefficientsNet, 141
Mcs::Usb, 70
BatteryState, 98
Charge, 98
ChargeRegionString, 98
ChargeString, 98
Voltage, 98
VoltageString, 98
BcdDevice
DeviceIdNet, 688
BeginImpedanceCheck
CIntanMea_FunctionNet, 195
Bessel
Mcs::Usb, 67
Bessel3dBFilterBandPassNet, 98

Bessel3dBFilterBandPassNet, 99
Bessel3dBFilterHighPassNet, 99
Bessel3dBFilterHighPassNet, 99
Bessel3dBFilterLowPassNet, 99
Bessel3dBFilterLowPassNet, 100
BesselConstDelayFilterBandPassNet, 100
BesselConstDelayFilterBandPassNet, 100
BesselConstDelayFilterHighPassNet, 101
BesselConstDelayFilterHighPassNet, 101
BesselConstDelayFilterLowPassNet, 101
BesselConstDelayFilterLowPassNet, 101
BesselLegacyFilterHighPassNet, 102
BesselLegacyFilterHighPassNet, 102
BesselLegacyFilterLowPassNet, 102
BesselLegacyFilterLowPassNet, 103
BMI
Mcs::Usb, 88
bmRequestType
usbSetupPacket_t, 721
BOOST_BIT
CW2100_StimulatorFunctionNet, 635
Bootstrap
FirmwareDestinationNames, 698
Mcs::Usb, 55
BootstrapOtherCypress
Mcs::Usb, 56
Both
Mcs::Usb, 93
Break
Mcs::Usb, 82
bRequest
usbSetupPacket_t, 721
BurnAdcOffset
COctoPotDeviceNet, 403
BurnDacOffset
CDacCalibrationFunctionNet, 124
COctoPotDeviceNet, 403
BUS0MCSBUS0
Mcs::Usb, 55
BUS0MCSBUS1
Mcs::Usb, 54
BUS0MCSBUS10
Mcs::Usb, 54
BUS0MCSBUS11
Mcs::Usb, 54
BUS0MCSBUS12
Mcs::Usb, 54
BUS0MCSBUS13
Mcs::Usb, 54
BUS0MCSBUS14
Mcs::Usb, 54
BUS0MCSBUS15
Mcs::Usb, 55
BUS0MCSBUS2
Mcs::Usb, 54
BUS0MCSBUS3
Mcs::Usb, 54
BUS0MCSBUS4
Mcs::Usb, 54
BUS0MCSBUS5
Mcs::Usb, 54
BUS0MCSBUS6
Mcs::Usb, 54
BUS0MCSBUS7
Mcs::Usb, 54
BUS0MCSBUS8
Mcs::Usb, 54
BUS0MCSBUS9
Mcs::Usb, 54
BUS1_MCSBUS1
FirmwareDestinationNames, 698
BUS1_MCSBUS2
FirmwareDestinationNames, 698
BUS1MCSBUS0
Mcs::Usb, 55
BUS1MCSBUS1
Mcs::Usb, 55
BUS1MCSBUS10
Mcs::Usb, 55
BUS1MCSBUS11
Mcs::Usb, 55
BUS1MCSBUS12
Mcs::Usb, 55
BUS1MCSBUS13
Mcs::Usb, 55
BUS1MCSBUS14
Mcs::Usb, 55
BUS1MCSBUS15
Mcs::Usb, 55
BUS1MCSBUS2
Mcs::Usb, 55
BUS1MCSBUS3
Mcs::Usb, 55
BUS1MCSBUS4
Mcs::Usb, 55
BUS1MCSBUS5
Mcs::Usb, 55
BUS1MCSBUS6
Mcs::Usb, 55
BUS1MCSBUS7
Mcs::Usb, 55
BUS1MCSBUS8
Mcs::Usb, 55
BUS1MCSBUS9
Mcs::Usb, 55
BUS2MCSBUS0
Mcs::Usb, 55
BUS2MCSBUS1
Mcs::Usb, 55
BUS2MCSBUS10
Mcs::Usb, 55
BUS2MCSBUS11
Mcs::Usb, 55
BUS2MCSBUS12
Mcs::Usb, 55
BUS2MCSBUS13

Mcs::Usb, 55
BUS2MCSBUS14
Mcs::Usb, 55
BUS2MCSBUS15
Mcs::Usb, 55
BUS2MCSBUS2
Mcs::Usb, 55
BUS2MCSBUS3
Mcs::Usb, 55
BUS2MCSBUS4
Mcs::Usb, 55
BUS2MCSBUS5
Mcs::Usb, 55
BUS2MCSBUS6
Mcs::Usb, 55
BUS2MCSBUS7
Mcs::Usb, 55
BUS2MCSBUS8
Mcs::Usb, 55
BUS2MCSBUS9
Mcs::Usb, 55
BUSNUMBER0
Mcs::Usb, 54
BUSNUMBER1
Mcs::Usb, 55
BUSNUMBER2
Mcs::Usb, 55
BusType
DeviceIdNet, 688
Butterworth
Mcs::Usb, 67
ButterworthFilterBandPassNet, 103
ButterworthFilterBandPassNet, 103
ButterworthFilterHighPassNet, 103
ButterworthFilterHighPassNet, 104
ButterworthFilterLowPassNet, 104
ButterworthFilterLowPassNet, 104

CalibrateThermocouple
CFluidControlDeviceNet, 149
CTcxDeviceNet, 596
Campden_Ci4600EphysVideoDataIntegrator
Mcs::Usb, 78
CancelInternalCalibration
CTEERFunctionNet, 609
CancelPoolLoop
CRoboDeviceNet, 479
CancelPoolLoopAndStopMovement
CRoboDeviceNet, 479
CancelTableLoop
CRoboDacqNet, 464
CancelTableLoopAndStopTable
CRoboDacqNet, 464
CapacityTest
CMultiBatteryChargerDeviceNet, 381
CatchAmp
Mcs::Usb, 77
CatchAmpGetAdcMean
CMcsBus_SensorNet, 233
CatchAmpGetAdcValue
CMcsBus_SensorNet, 233
CatchAmpGetAdcValueH
CMcsBus_SensorNet, 233
CatchAmpGetAdcValueL
CMcsBus_SensorNet, 233
CatchAmpGetDacAmplitude
CMcsBus_SensorNet, 234
CatchAmpGetDacEnable
CMcsBus_SensorNet, 234
CatchAmpGetDacOffset
CMcsBus_SensorNet, 234
CatchAmpGetPwmEnable
CMcsBus_SensorNet, 234
CatchAmpSetDacAmplitude
CMcsBus_SensorNet, 234
CatchAmpSetDacEnable
CMcsBus_SensorNet, 234
CatchAmpSetDacOffset
CMcsBus_SensorNet, 234
CatchAmpSetPwmEnable
CMcsBus_SensorNet, 234
CChannelTestDeviceNet, 105
~CChannelTestDeviceNet, 105
CChannelTestDeviceNet, 105
SetAmplitude, 105
SetAttenuation, 105
SetFrequency, 105
SetWaveform, 106
CCMOSMea_FunctionNet, 106
AreTransistorVoltagesSet, 108
CCMOSMea_FunctionNet, 108
ClearSTGOutput, 108
DetectChipType, 108
EnableChannelsInGroup, 108, 109
GetADCInputOffset, 109
GetBath, 109
GetBathMode, 109
GetEnabledChannelsInGroup, 109
GetGate, 109
GetGNDI, 109
GetGroupADCBits, 109
GetGroupChannelBitmaskBySelect, 110
GetGroupChannelBitmaskHS1NCBathCurrent, 110
GetGroupChannelBitmaskHS1NCCol2Current, 110
GetGroupChannelBitmaskHS1NChipTemp, 110
GetGroupChannelBitmaskHS1Sidebands, 111
GetGroupChannelBitmaskHS1TriggerStatus, 111
GetGroupChannelBitmaskIFDigChannels, 111
GetGroupChannelBitmaskInterfaceADC, 111
GetGroupChannelBitmaskPacketFrameContext, 112
GetGroupChannelBitmaskSTG1DACSignal, 112
GetGroupDCOffset, 112
GetGroupID, 112
GetGroupNumberOfChannels, 113

GetGroupResolutionPerDigit, 113
GetGroupSampleSize, 113
GetGroupType, 113
GetGroupUnit, 113, 114
GetMaxNumOfColumns, 114
GetNeurochipMemoryData, 114
GetNeurochipMemorySize, 114
GetNumberOfSupportedGroups, 114
GetSourceBulk, 114
GetSourceDrain, 114
GetSourceGate, 114
GetStimulusSites, 115
GetVDD3I, 115
GetVDDI, 115
IsChipPowered, 115
IsGateFloating, 115
PowerChip, 115
SetADCInputOffset, 115
SetBath, 115
SetBathMode, 115
SetGate, 115
SetGateFloating, 116
SetGateToVOP, 116
SetNeurochipMemoryData, 116
SetSourceBulk, 116
SetSourceDrain, 116
SetSourceGate, 116
SetStimulusSites, 116
UpdateTransistorVoltages, 116
VOPSTimerSetResetTimes, 116, 117
CCMOSMeaDeviceNet, 117
~CCMOSMeaDeviceNet, 118
CCMOSMeaDeviceNet, 118
CMosMea, 120
GetAvailableBaseSamplerates, 118
GetBaseSamplerate, 118
GetChannelDataI16, 118
GetChannelDataI32, 118
GetChannelDataUI16, 119
GetChannelDataUI32, 119
GetCMOSDataDictionary, 119
GetMaxReadableColumns, 119
SetBaseSamplerate, 119
SetRegionOfInterests, 119
Stimulus, 120
UpdateChannelBlock, 119
CCMOSMeaDeviceNet::CRegionOfInterestRect, 450
CRegionOfInterestRect, 450
DeepCopy, 450
m_Bottom, 451
m_Left, 451
m_Right, 451
m_Top, 451
CCreateFilterNet, 120
~CCreateFilterNet, 121
CCreateFilterNet, 121
CutoffOrCenterFrequency, 122
FindFilter, 122
GetBiQuad, 122
GetBiQuads, 122
GetCpp, 122
NumCoefSets, 122
Order, 122
SampleRate, 122
Scale, 122
WidthFrequency, 123
CDacCalibrationFunctionNet, 123
!CDacCalibrationFunctionNet, 124
~CDacCalibrationFunctionNet, 124
BurnDacOffset, 124
CDacCalibrationFunctionNet, 123
GetDacOffset, 124
SetDacOffset, 124
CDacqGroupChannelGenericSelectionNet, 125
CDacqGroupChannelGenericSelectionNet, 125
CDacqGroupChannelSelectionNet, 125
CDacqGroupChannelSelectionNet, 126
CDacqGroupChannelSelectionTemplateNet
CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet >, 126
CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet >, 126
CDacqGroupChannelSelectionTemplateNet, 126
EnableChannelsInGroup, 127
GetDeviceGroupChannelInfos, 127
GetEnabledChannelsInGroup, 127
GetGroupID, 127
GetGroupNumberOfChannels, 128
GetGroupSampleSize, 128
GetGroupType, 128
GetNumberOfSupportedGroups, 128
CDeviceGroupChannelInfoGenericNet, 129
CDeviceGroupChannelInfoGenericNet, 129
CDeviceGroupChannelInfoMEA2100_256Net, 129
CDeviceGroupChannelInfoMEA2100_256Net, 129
CDeviceGroupChannelInfoNet, 130
CDeviceGroupChannelInfoNet, 130
CDeviceGroupChannelInfoSCUNet, 130
CDeviceGroupChannelInfoSCUNet, 131
CDeviceGroupChannelInfoTemplateNet
CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumTemplateNet >, 131
CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumTemplateNet >, 131
CDeviceGroupChannelInfoTemplateNet, 131
GroupID, 131
GroupType, 131
NumberOfChannels, 131
CDeviceGroupChannelInfoW2100Net, 132
CDeviceGroupChannelInfoW2100Net, 132
CDigOutStimulatorFunctionNet, 132
!CDigOutStimulatorFunctionNet, 134

~CDigOutStimulatorFunctionNet, 134
CDigOutStimulatorFunctionNet, 133
ClearChannel, 134
GetGlobalRepeat, 134
GetNumberOfChannels, 134
GetStartTriggerSlope, 134
GetStopTriggerSlope, 135
PrepareChannelData, 135
SendChannelData, 135
SetGlobalRepeat, 136
SetStartTriggerSlope, 136
SetStopTriggerSlope, 136
CEncapsulatorDeviceNet, 137
CEncapsulatorDeviceNet, 137
GetRoboFluidDevice, 137
CExternDTesterDeviceNet, 137
!CExternDTesterDeviceNet, 138
~CExternDTesterDeviceNet, 138
CExternDTesterDeviceNet, 138
Read, 138
Read2, 139
Write, 139
Write2, 139
CFilterCoefficientsNet, 139
~CFilterCoefficientsNet, 140
A, 141
B, 141
CFilterCoefficientsNet, 140
GetUintA, 141
GetUintB, 141
GetUints, 141
IsEqual, 141
CFilterCoefficientsNet::s_FilterAttributesNet, 716
CommaPositionA, 717
CommaPositionB, 717
PostCommaA, 717
PostCommaB, 717
PreCommaA, 717
PreCommaB, 717
s_FilterAttributesNet, 716
ToCpp, 716
CFilterConfigurationNet, 141
CFilterConfigurationNet, 142
EraseFilterParameterPermanent, 142
GetDefaultFilterAttributes, 142
GetFilterAttributes, 142, 143
GetHighpassFilterEnable, 143
ResetHighpassFilter, 143
SetFilterParameter, 143
SetFilterParameterPermanent, 143
SetHighpassFilterEnable, 143
CFilterConfigurationRegisterNet, 144
CFilterConfigurationRegisterNet, 144
EraseFilterParameterPermanent, 144
SetFilterParameter, 145
SetFilterParameterPermanent, 145
CFilterPropertyNet, 145
~CFilterPropertyNet, 146
CFilterPropertyNet, 146
CornerFrequency, 146
CornerFrequencymHz, 146
FilterActive, 146
FilterBand, 147
FilterFamily, 147
FilterType, 147
Order, 147
ToString, 146
CFirmwareDestinationNet
Mcs::Usb, 54
CFluidControlDeviceNet, 147
~CFluidControlDeviceNet, 148
CalibrateThermocouple, 149
CFluidControlDeviceNet, 148
GetAdc, 150
GetDigin, 150
GetDigout, 150
GetPWM, 150
GetReferenceTemperature, 150
GetSingleValve, 151
GetThermocoupleCalibration, 151
GetThermocoupleNanovoltPerKelvin, 151
GetThermocoupleTemperature, 152
GetValve, 152
McsBus_VoltageMode, 154
SetDigout, 152
SetPWM, 152
SetSingleValve, 153
SetThermocoupleNanovoltPerKelvin, 153
SetValve, 153
CFYIDeviceNet, 154
CFYIDeviceNet, 154
FYIProgram, 155
FYITemp, 155
Sensor, 155
CGenericDevelopDeviceNet, 155
~CGenericDevelopDeviceNet, 162
CGenericDevelopDeviceNet, 162
ClosePipe, 162
FindEndpoints, 163
GetBuffer, 163
GetByteBuffer, 163
GetIntBuffer, 163
GetShortBuffer, 164
GetUByteBuffer, 165
GetUIntBuffer, 165
GetUShortBuffer, 166
OpenPipe, 167
ReadPipe, 167
ResetPipe, 168
SetBuffer, 168
SetByteBuffer, 168
SetIntBuffer, 169
SetShortBuffer, 169
SetUByteBuffer, 170
SetUIntBuffer, 170
SetUShortBuffer, 171

SetValue, 172
VendorInRequest, 172
VendorOutRequest, 172
WritePipe, 172
CGilsonDeviceNet, 173
~CGilsonDeviceNet, 174
CGilsonDeviceNet, 174
ConnectSlave, 174
GetLastAnswer, 174
m_pGilsonDevice, 174
SendBuffered, 174
SendImmediate, 174
SendImmediateGetResponse, 174
CGrapheneASICDeviceNet, 175
~CGrapheneASICDeviceNet, 175
CGrapheneASICDeviceNet, 175
GetAvailableBaseSamplerates, 175
GetBaseSamplerate, 176
SetBaseSamplerate, 176
SetRegionOfInterests, 176
CGrapheneFunctionNet, 176
!CGrapheneFunctionNet, 179
~CGrapheneFunctionNet, 178
CGrapheneFunctionNet, 178
GetADCGain, 179
GetADCOffset, 179, 180
GetCur2VolResistance, 180
GetDACOffset, 181
GetVdsVgs, 181
GetVdVs, 182
GetVdVsDAC, 182, 183
GetVoltageRange, 183
GetVoltageReached, 183
GetVoltageResolution, 184
SetADCGain, 184, 185
SetADCOffset, 185
SetDACOffset, 185, 186
SetVdsVgs, 186
SetVdVs, 187
SetVdVsDAC, 187
SetVoltageRange, 188
SetVoltageResolution, 188
ChangeSerialNumber
CMcsUsbFactoryNet, 306
channel
CMeaAudioFunctionNet::s_setaudionet, 717
CW2100_FunctionNet::AudioChannelsNet, 97
ChannelBlock_AvailFrames
CMcsUsbDacqNet, 261
ChannelBlock_ReadAsFrameArrayI16
CMcsUsbDacqNet, 261, 262
ChannelBlock_ReadAsFrameArrayI32
CMcsUsbDacqNet, 263, 264
ChannelBlock_ReadAsFrameArrayUI16
CMcsUsbDacqNet, 264, 265
ChannelBlock_ReadAsFrameArrayUI32
CMcsUsbDacqNet, 266
ChannelBlock_ReadFramesDictI16
CMcsUsbDacqNet, 267
ChannelBlock_ReadFramesDictI32
CMcsUsbDacqNet, 268
ChannelBlock_ReadFramesDictUI16
CMcsUsbDacqNet, 268
ChannelBlock_ReadFramesDictUI32
CMcsUsbDacqNet, 269
ChannelBlock_ReadFramesI16
CMcsUsbDacqNet, 269, 270
ChannelBlock_ReadFramesI32
CMcsUsbDacqNet, 271
ChannelBlock_ReadFramesUI16
CMcsUsbDacqNet, 272
ChannelBlock_ReadFramesUI32
CMcsUsbDacqNet, 273, 274
channeldata_current
Mcs::Usb, 87
channeldata_current_always_boost
Mcs::Usb, 87
channeldata_current_always_boost_own_sync
Mcs::Usb, 87
channeldata_current_own_boost_gnd_sync
Mcs::Usb, 87
channeldata_current_own_sync
Mcs::Usb, 87
channeldata_positive_current
Mcs::Usb, 87
channeldata_positive_current_own_boost_gnd_sync
Mcs::Usb, 87
channeldata_positive_current_own_sync
Mcs::Usb, 87
channeldata_positive_voltage
Mcs::Usb, 87
channeldata_voltage
Mcs::Usb, 87
ChannelDataEvent
CMcsUsbDacqNet, 302
ChannelPIC
Mcs::Usb, 55
ChannelReset
CMultiBatteryChargerDeviceNet, 381
ChannelTest
Mcs::Usb, 79
Charge
BatteryState, 98
ChargeRegionString
BatteryState, 98
ChargeString
BatteryState, 98
ChecksumAndPacketCounter
Mcs::Usb, 61
CHiClampDeviceNet, 189
CHiClampDeviceNet, 189
RoboDacq, 189
CHLADacqNet, 190
CHLADacqNet, 190
CHLADeviceNet, 190
CHLADeviceNet, 191

HLADacq, 191
SerialPort, 191
CHWInfo
CMcsUsbDacqNet::CHWInfo, 192
Ci4600Intan
Mcs::Usb, 53
CIntanMea_FunctionNet, 194
AmplifierSettle, 195
BeginImpedanceCheck, 195
CIntanMea_FunctionNet, 195
GetDSPHighPassByIndex, 195
GetImpedanceResult, 195
GetIntanRegister, 195
GetLowerFrequencyByIndex, 195
GetUpperFrequencyByIndex, 196
SetBandwidthByIndex, 196
SetDiagnosticMode, 196
SetDSPHighPassByIndex, 196
SetIntanRegister, 196
CInterfaceboard2FunctionNet, 196
!CInterfaceboard2FunctionNet, 197
∼CInterfaceboard2FunctionNet, 197
CInterfaceboard2FunctionNet, 197
GetIoVoltage, 197
SetIoVoltage, 198
CInterfaceboardFunctionNet, 198
!CInterfaceboardFunctionNet, 199
∼CInterfaceboardFunctionNet, 199
CInterfaceboardFunctionNet, 199
GetCardinalDacqSamplerate, 199
GetCardinalStgOutputrate, 199
SetCardinalDacqSamplerate, 199
SetCardinalStgOutputrate, 200
ClampAmpRestart
CRoboDacqNet, 464
ClampModeCurrent
Mcs::Usb, 89
ClampModeInternalCalibration
Mcs::Usb, 89
ClampModeOpen
Mcs::Usb, 89
ClampModeVoltage
Mcs::Usb, 89
ClearBuffers
CMcsUsbDacqNet, 274
ClearChannel
CDigOutStimulatorFunctionNet, 134
ClearChannel_PrepareAndSendData
CStg200xDownloadNet, 574
CStimulusFunctionNet, 582
ClearChannelData
CStg200xDownloadBasicNet, 566
CStimulusFunctionNet, 582
CW2100_StimulatorFunctionNet, 630
ClearMultiplexedData
CStimulusFunctionNet, 582
ClearSTGOutput
CCMOSMea_FunctionNet, 108
ClearStimulusParametersCache
CW2100_FunctionNet, 623
ClearSyncData
CStg200xDownloadBasicNet, 566
CStimulusFunctionNet, 582
ClearTable
CRetinaLedDeviceNet, 452
ClearTableName
CWarnerValveControllerDeviceNet, 658
ClearUserDefinedNameCache
CW2100_FunctionNet, 623
ClearValveTable
CWarnerValveControllerDeviceNet, 658
CLIH3DeviceNet, 200
!CLIH3DeviceNet, 202
∼CLIH3DeviceNet, 202
CLIH3DeviceNet, 202
DummyCommand, 202
EnableUserTrigger, 202
ErasePermanentAdcOffset, 203
ErasePermanentDacOffset, 203
GetAdcOffset, 203
GetAudioOutDacParameter, 203
GetDacIdleValue, 204
GetDacOffset, 204
GetDacqRunStatus, 204
GetDacUseIdleValue, 204
GetDigInState, 205
GetEEpromPage, 205
GetSampleInterval, 205
IsUserTriggerEnabled, 205
ReadClipping, 206
ReadUARTData, 206
SendCommand, 206
SetAdcOffset, 206
SetAdcOffsetPermanent, 207
SetAudioOutDacParameter, 207
SetDacIdleValue, 207
SetDacOffset, 207
SetDacOffsetPermanent, 208
SetDacUseIdleValue, 208
SetDigOutState, 208
SetEEpromPage, 208
SetSampleInterval, 209
StimulusFunction, 209
WriteUARTData, 209
Close
Mcs::Usb, 77
CloseAllValves
CRoboFluidDeviceNet, 493
ClosePipe
CGenericDevelopDeviceNet, 162
ClosePlateClamp
CMultiwellDeviceNet, 391
CMcsBus_AxisParametersNet, 209
∼CMcsBus_AxisParametersNet, 210
CMcsBus_AxisParametersNet, 210
GetAxisParametersSignedEeprom, 210

GetAxisParametersUnsignedEeprom, 210
SetAxisParametersEeprom, 210, 211
CMcsBus_ExtensionNet, 211
~CMcsBus_ExtensionNet, 211
CMcsBus_ExtensionNet, 211
GetLEDSwitch, 212
SetLEDSwitch, 212
CMcsBus_FYIExtensionNet, 212
~CMcsBus_FYIExtensionNet, 212
CMcsBus_FYIExtensionNet, 212
GetDIO, 213
GetSingleHeater, 213
GetValves, 213
SetDIO, 213
SetSingleHeater, 213
SetValves, 213
CMcsBus_MotorControlNet, 214
~CMcsBus_MotorControlNet, 217
CMcsBus_MotorControlNet, 217
GetMCAcceleration, 217
GetMCAccelerationEeprom, 217
GetMCAccelerationShortCommand, 217
GetMCAxisRevisionEeprom, 217
GetMCBreakCurrent, 217
GetMCBreakCurrentEeprom, 218
GetMCConfig, 218
GetMCConfigEeprom, 218
GetMCCurrent, 218
GetMCCurrentEeprom, 218
GetMCCurrentMode, 218
GetMCCurrentModeEeprom, 218
GetMCCurrentModeShortCommand, 219
GetMCCurrentPosition, 219
GetMCCurrentShortCommand, 219
GetMCCurrentSpeed, 219
GetMCMaxAcceleration, 219
GetMCMaxAccelerationEeprom, 219
GetMCMaxCurrent, 219
GetMCMaxCurrentEeprom, 220
GetMCMaxSpeed, 220
GetMCMaxSpeedEeprom, 220
GetMCMaxTravel, 220
GetMCMaxTravelEeprom, 220
GetMCMaxTravelShortCommand, 220
GetMCMovement, 220
GetMCNewPosition, 221
GetMCOutputOnOff, 221
GetMCPhase, 221
GetMCPhaseOffset, 221
GetMCReference, 221
GetMCReferenceCurrent, 221
GetMCReferenceCurrentEeprom, 221
GetMCRegulatorGain, 222
GetMCRegulatorGainEeprom, 222
GetMCScalingFactor, 222
GetMCScalingFactorEeprom, 222
GetMCSpeed, 222
GetMCSpeedEeprom, 222
GetMCSpeedShortCommand, 222
GetMCSpeedUnitEeprom, 223
GetMCStandbyCurrent, 223
GetMCStandbyCurrentEeprom, 223
GetMCStandbyTime, 223
GetMCStandbyTimeEeprom, 223
GetSubChannel, 223
SetMCAcceleration, 223
SetMCAccelerationEeprom, 224
SetMCAccelerationShortCommand, 224
SetMCAxisRevisionEeprom, 224
SetMCBreakCurrent, 224
SetMCBreakCurrentEeprom, 224
SetMCConfig, 224
SetMCConfigEeprom, 225
SetMCCurrent, 225
SetMCCurrentEeprom, 225
SetMCCurrentMode, 225
SetMCCurrentModeEeprom, 225
SetMCCurrentModeShortCommand, 225
SetMCCurrentPosition, 226
SetMCCurrentShortCommand, 226
SetMCMaxAcceleration, 226
SetMCMaxAccelerationEeprom, 226
SetMCMaxCurrent, 226
SetMCMaxCurrentEeprom, 226
SetMCMaxSpeed, 227
SetMCMaxSpeedEeprom, 227
SetMCMaxTravel, 227
SetMCMaxTravelEeprom, 227
SetMCMaxTravelShortCommand, 227
SetMCNewPosition, 227
SetMCOutputOnOff, 228
SetMCReference, 228
SetMCReferenceCurrent, 228
SetMCReferenceCurrentEeprom, 228
SetMCRegulatorGain, 228
SetMCRegulatorGainEeprom, 228
SetMCRotation, 229
SetMCScalingFactor, 229
SetMCScalingFactorEeprom, 229
SetMCSpeed, 229
SetMCSpeedEeprom, 229
SetMCSpeedShortCommand, 229
SetMCSpeedUnitEeprom, 230
SetMCStandbyCurrent, 230
SetMCStandbyCurrentEeprom, 230
SetMCStandbyTime, 230
SetMCStandbyTimeEeprom, 230
SetSubChannel, 230
StartMCMovement, 231
StopMCMovement, 231
CMcsBus_SensorNet, 231
~CMcsBus_SensorNet, 233
CatchAmpGetAdcMean, 233
CatchAmpGetAdcValue, 233
CatchAmpGetAdcValueH, 233
CatchAmpGetAdcValueL, 233

CatchAmpGetDacAmplitude, 234
CatchAmpGetDacEnable, 234
CatchAmpGetDacOffset, 234
CatchAmpGetPwmEnable, 234
CatchAmpSetDacAmplitude, 234
CatchAmpSetDacEnable, 234
CatchAmpSetDacOffset, 234
CatchAmpSetPwmEnable, 234
CMcsBus_SensorNet, 233
Get2AnalogInput, 235
Get2DigitalInput, 235
Get4ADC, 235
Get4ADCAverage, 235
Get4ADCCatchampAverageShift, 235
Get4ADCMode, 235
Get4DAC, 235
GetADCs, 235
GetADCsLoop, 236
GetBubbleStatus, 236
GetDACs, 236
GetDetectionThreshold, 236
GetDetectorValue, 236
GetLatency, 236
GetLatencyCounter, 236
GetMinimalThreshold, 236
GetMovePump, 237
GetPiezoState, 237
GetPressure, 237
GetPressureOffset, 237
GetRegulationTimeouts, 237
GetRegulatorFactor, 238
GetRegulatorOnOff, 238
GetRegulatorStatus, 238
GetRotatePump, 238
GetSamplePeriode, 238
GetSollPressure, 238
GetSyncState, 238
Set4ADCCatchampAverageShift, 239
Set4ADCMode, 239
Set4DAC, 239
SetDACs, 239
SetDetectionThreshold, 239
SetLatency, 239
SetMinimalThreshold, 239
SetMovePump, 240
SetPiezoState, 240
SetPressureOffset, 240
SetRegulationTimeouts, 240
SetRegulatorFactor, 240
SetRegulatorOnOff, 240
SetRotatePump, 240
SetSamplePeriode, 241
SetSollPressure, 241
StartSync, 241
TactSwitchGetState, 241
TactSwitchSetDisplay, 241
CMcsBus_TempSensorNet, 241
~CMcsBus_TempSensorNet, 242
CMcsBus_TempSensorNet, 242
GetNanoVoltsPerKelvin, 242
GetTemperatur, 242
GetThermoOffset, 242
GetThermoTemp, 243
GetThermoVoltage, 243
SetNanoVoltsPerKelvin, 243
SetThermoOffset, 243
CMcsBus_VoltageModeNet, 243
~CMcsBus_VoltageModeNet, 244
CMcsBus_VoltageModeNet, 244
GetVMMaxNegativeCurrent, 245
GetVMMaxNegativeCurrentEeprom, 245
GetVMMaxNegativeVoltage, 245
GetVMMaxNegativeVoltageEeprom, 245
GetVMMaxPositiveCurrent, 245
GetVMMaxPositiveCurrentEeprom, 245
GetVMMaxPositiveVoltage, 245
GetVMMaxPositiveVoltageEeprom, 246
GetVMOutputOnOff, 246
GetVMVoltage, 246
SetVMMaxNegativeCurrent, 246
SetVMMaxNegativeCurrentEeprom, 246
SetVMMaxNegativeVoltage, 246
SetVMMaxNegativeVoltageEeprom, 246
SetVMMaxPositiveCurrent, 247
SetVMMaxPositiveCurrentEeprom, 247
SetVMMaxPositiveVoltage, 247
SetVMMaxPositiveVoltageEeprom, 247
SetVMOutputOnOff, 247
SetVMVoltage, 247
CMcsBusNet, 248
~CMcsBusNet, 248
CMcsBusNet, 248
CMcsBusNet::GetMode, 249
CMcsBusNet::GetModeEeprom, 249
CMcsBusNet::SetMode, 249
CMcsBusNet::SetModeEeprom, 249
GetBusAddress, 249
GetBusAddressEeprom, 249
GetCommand, 249, 250
GetHWRevisionEeprom, 250
SetBusAddress, 250
SetBusAddressEeprom, 250
SetCommand, 251
SetHWRevisionEeprom, 251
CMcsBusNet::GetMode
CMcsBusNet, 249
CMcsBusNet::GetModeEeprom
CMcsBusNet, 249
CMcsBusNet::SetMode
CMcsBusNet, 249
CMcsBusNet::SetModeEeprom
CMcsBusNet, 249
CMcsUsbDacqNet, 252
~CMcsUsbDacqNet, 258
AddSelectedChannelsQueue, 258–261
ChannelBlock_AvailFrames, 261

ChannelBlock_ReadAsFrameArrayI16, 261, 262
ChannelBlock_ReadAsFrameArrayI32, 263, 264
ChannelBlock_ReadAsFrameArrayUI16, 264, 265
ChannelBlock_ReadAsFrameArrayUI32, 266
ChannelBlock_ReadFramesDictI16, 267
ChannelBlock_ReadFramesDictI32, 268
ChannelBlock_ReadFramesDictUI16, 268
ChannelBlock_ReadFramesDictUI32, 269
ChannelBlock_ReadFramesI16, 269, 270
ChannelBlock_ReadFramesI32, 271
ChannelBlock_ReadFramesUI16, 272
ChannelBlock_ReadFramesUI32, 273, 274
ChannelDataEvent, 302
ClearBuffers, 274
CMcsUsbDacqNet, 258
CMcsUsbDacqNet::GetFilterProperties, 274
Error_Callback_Aquisition_Stopped, 301
Error_Callback_Data_lost, 301
Error_Callback_Frames_Lost, 301
Error_Callback_Packet_Error, 301
Error_Callback_Queue_Full, 302
Error_Callback_RingQueue_Full, 302
ErrorEvent, 302
GetAdapterType, 274
GetAdcDataFormat, 275
GetAdcZero, 275
GetAnalogValueUnit, 275
GetChannelDataFillSize, 275
GetChannelLayout, 275
GetChannelsInBlock, 275
GetDataFormat, 276
GetDataMode, 276
GetDigitalSource, 276–278
GetErrorMessage, 279
GetFilterProperty, 279
GetGroupChannelDataI16, 279
GetGroupChannelDataI32, 280
GetGroupChannelDataUI16, 280
GetGroupChannelDataUI32, 281
GetHardwareMaxRange, 281
GetHardwareMinRange, 281
GetMaxSamplingFrequency, 282
GetMeaLayout, 282
GetMinSamplingFrequencyStepsize, 282
GetNumberOfDataBits, 282
GetPoti, 282
GetResolutionPerDigit, 283
GetSamplerate, 283
GetVoltageRangeIndex, 283
GetVoltageRangeInMicroVolt, 283
GetVoltageRangeInMilliVolt, 283
HWInfo, 283
Samplerate, 302
SendStartDacq, 284
SendStartStgAndDacq, 284
SendStopDacq, 284
SendStopStgAndDacq, 285
SendStopStgAndDacqWithOptions, 285
SetDataMode, 285
SetDigitalSource, 286–288
SetPoti, 288
SetSamplerate, 288
SetSelectedChannels, 290–292
SetSelectedChannelsQueue, 292–294
SetSelectedData, 294–296
SetupGroupDacqQueue, 296
SetVoltageRangeByIndex, 296
SetVoltageRangeInMicroVolt, 297
StartDacq, 297, 298
StartLoop, 299, 300
StopDacq, 301
StopLoop, 301
CMcsUsbDacqNet::CHWInfo, 191
CHWInfo, 192
GetAvailableSampleRates, 192
GetAvailableVoltageRangesInMicroVolt, 192
GetAvailableVoltageRangesInMicroVoltAndStringsInMilliVolt, 193
GetNumberOfHWADCChannels, 193
GetNumberOfHWDigitalChannels, 193
IsDigitalChannelDedicated, 194
CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet, 620
CVoltageRangeInfoNet, 621
VoltageRangeDisplayStringMilliVolt, 621
VoltageRangeInMicroVolt, 621
CMcsUsbDacqNet::GetFilterProperties
CMcsUsbDacqNet, 274
CMcsUsbDeviceStatePushFunctionNet, 302
CMcsUsbDeviceStatePushFunctionNet, 303
McsUsbDeviceStateEvent, 303
TriggerStatus, 303
CMcsUsbDeviceStatePushNet, 303
CMcsUsbDeviceStatePushNet, 304
McsUsbDeviceStateEvent, 304
TriggerStatus, 304
CMcsUsbFactoryNet, 304
~CMcsUsbFactoryNet, 306
ChangeSerialNumber, 306
CMcsUsbFactoryNet, 306
Coldstart, 306
CompareFirmware, 307
DownloadFirmware, 307
FindFirmwareVersionMagicInBuffer, 307
FX3MCSDataAddress, 312
FX3MCSDataDeviceIdOffset, 312
FX3MCSDataIFB1ImageOffset, 312
FX3MCSDataIFB2ImageOffset, 312
FX3MCSDataVersionOffset, 312
GetChecksumFromFX3Image, 307
GetDestination, 307
GetDestinationDisplayLabel, 307
GetDestinationName, 307, 308
GetDestinationSerialNumber, 308
GetDestinationTargetAddress, 308
GetFirmwareVersionFromFile, 308

GetFirmwareVersionFromHexFile, 308
GetNumDestinations, 308
GetUSBDeviceIDFromFX3Image, 309
GetUsercodeFromBitFile, 309
GetUsercodeFromFlash, 309
GetXilinxFlashOffset, 309
GetXilinxFlashReadCommand, 309
LoadUserFirmware, 309, 310
ReadBlockFromFlash, 310
ReadBlockFromIFBGlobalEEprom, 310
ReadBlockFromNVMEM, 310
SetDestinationSerialNumber, 310
UpdateFirmware, 310–312
CMcsUsbFunctionNet, 313
!CMcsUsbFunctionNet, 314
~CMcsUsbFunctionNet, 314
CMcsUsbFunctionNet, 313, 314
m_pMcsUsb, 314
m_pMcsUsbFunction, 314
ThrowCUsbExceptionNetOnError, 314
CMcsUsbFunctionPointerContainer, 314
CMcsUsbListEntryNet, 314
~CMcsUsbListEntryNet, 315
DeviceId, 319
DeviceName, 319
Equals, 316
GetEntry, 317
GetEntryCount, 318
HwVersion, 319
Manufacturer, 319
Product, 319
SerialNumber, 319
SetStringFormat, 318
ToString, 319
CMcsUsbListNet, 320
!CMcsUsbListNet, 321
~CMcsUsbListNet, 321
CMcsUsbListNet, 320
Count, 322
DeviceArrival, 322
DeviceRemoval, 322
GetNumberOfDevices, 321
GetUsbListEntries, 321
GetUsbListEntry, 321
IsDeviceTypeOf, 322
SetStringFormat, 322
CMcsUsbNet, 323
!CMcsUsbNet, 327
~CMcsUsbNet, 327
AddSoftwareKey, 327
AssociateToThis, 327
CMcsUsbNet, 327
Connect, 327, 328
CyclePort, 329
Disconnect, 329
EmptyKey, 329
EnableExceptions, 329
EraseEepromRegisterPreconfig, 329
GetConfiguration, 330
GetDeviceCannotStallOutRequests, 330
GetDeviceCapableSpeed, 330
GetDeviceEnum, 330
GetDeviceId, 330
GetDeviceRootHubVendorEnum, 330
GetDeviceRootHubVendorID, 330
GetDeviceRootHubVendorName, 330
GetDeviceSpeed, 331
GetErrorText, 331
GetFirmwareVersion, 331
GetHardwareRevision, 331
GetHeadstageActive, 332
GetHeadstageID, 332
GetHeadstagePresent, 332
GetLastUSBError, 332
GetMea21UsbPort, 333
GetNumConfigurations, 333
GetSerialNumber, 333
GetSoftwareKey, 333
GetSoftwareKeyString, 333
GetStatus, 333
GetStatusOfLastCommand, 333
GetUsbListEntry, 334
GetVersion, 334
HasSoftwareKey, 334
IsConnected, 334
IsDeviceHighSpeed, 334
IsDeviceHighSpeedCapable, 334
IsExceptionsEnabled, 335
MultibootGetCypressImageId, 335
MultibootGetImageId, 335
MultibootGetSelectedImage, 335
MultibootSelectImage, 335
ReadEepromRegisterPreconfig, 335, 336
ReadRegister, 336
ReadRegister32, 336
ReadRegisterTimeSlot, 336
RemoveSoftwareKey, 336
RescanHeadstage, 336
SerialNumber, 343
SetConfiguration, 337
SetSoftwareKey, 337
Status_AlreadyConfigured, 339
Status_BadStartFrame, 339
Status_Btstuff, 339
Status_BufferOverrun, 339
Status_BufferUnderrun, 339
Status_Canceled, 339
Status_Canceling, 340
Status_ConnectedPipes, 340
Status_ControlNotOwned, 340
Status_Crc, 340
Status_DataOverrun, 340
Status_DataToggleMismatch, 340
Status_DataUnderrun, 340
Status_DeviceLocked, 340
Status_DeviceNotFound, 340

Status_DeviceRemoved, 340
Status_DevNotResponding, 340
Status_EndpointHalted, 341
Status_ErrorBusy, 341
Status_ErrorShortTransfer, 341
Status_Fifo, 341
Status_FrameControlOwned, 341
Status_InternalHcError, 341
Status_InvalidDeviceHandle, 341
Status_InvalidHandle, 341
Status_InvalidParameter, 341
Status_InvalidPipeHandle, 341
Status_InvalidUrbFunction, 342
Status_IoPending, 342
Status_IoTimeout, 342
Status_IsochRequestFailed, 342
Status_LastUsbErrorMismatch, 342
Status_NoBandwidth, 342
Status_NoMemory, 342
Status_NoSuchDevice, 342
Status_NotAccessed, 342
Status_NotSupported, 342
Status_PidCheckFailure, 342
Status_PipeNotLinked, 343
Status_RequestFailed, 343
Status_RequestMutexFailed, 343
Status_RequestMutexTimeout, 343
Status_Stall, 343
Status_Unconfigured, 343
Status_UnexpectedPid, 343
ThrowCUsbExceptionNetOnError, 337
TxnGetSerialNumber, 337
TxnSetSerialNumber, 337
TxnTestMemoryReadAndCheck, 337
TxnTestMemoryWrite, 337
ValidKey, 337
WPAError_ScanningIsPending, 343
WriteEepromRegisterPreconfig, 338
WriteRegister, 338
WriteRegister32, 338
WriteRegisterArray, 338
WriteRegisterTimeSlot, 339
WriteRegisterValue, 339
CMcsUsbPointerContainer, 344
CMEA2100_256DacqGroupChannelSelectionNet, 344
CMEA2100_256DacqGroupChannelSelectionNet, 344
CMEA2100x256FunctionNet, 344
!CMEA2100x256FunctionNet, 345
~CMEA2100x256FunctionNet, 345
CMEA2100x256FunctionNet, 345
GetLayoutConfiguration, 345
SetLayoutConfiguration, 345
CMeaAudioFunctionNet, 346
CMeaAudioFunctionNet, 346
GetAudioChannels, 347
GetNumberOfAudioChannels, 347
SetAudioChannels, 348
CMeaAudioFunctionNet::s_setaudionet, 717
amplification, 717
channel, 717
CMeaCleanDeviceNet, 349
!CMeaCleanDeviceNet, 350
~CMeaCleanDeviceNet, 350
CMeaCleanDeviceNet, 350
GetCycle, 350
GetCycles, 350
GetMaxVoltage, 350
GetMinVoltage, 351
GetOutputVoltage, 351
GetSlope, 351
IsRunning, 351
SetCycles, 351
SetMaxVoltage, 352
SetMinVoltage, 352
SetSlope, 352
Start, 352
Stop, 352
CMeaCoatDeviceNet, 353
!CMeaCoatDeviceNet, 354
~CMeaCoatDeviceNet, 354
CMeaCoatDeviceNet, 354
GetCurrentCycle, 354
GetCycles, 354
GetDuration, 355
GetMaxCurrent, 355
GetOffsetCurrent, 355
GetOutputCurrent, 355
GetPauseDuration, 355
GetSlope, 356
GetTimeInPause, 356
GetTimeInPlateau, 356
IsRunning, 356
SetCycles, 356
SetDuration, 357
SetMaxCurrent, 357
SetOffsetCurrent, 357
SetPauseDuration, 357
SetSlope, 358
Start, 358
Stop, 358
CMeaDeviceNet, 358
~CMeaDeviceNet, 360
AnalogGain, 365
CMeaDeviceNet, 360
EnableChecksum, 360
EnableDigitalIn, 361
EnableTimestamp, 362
Gain, 365
GetAnalogGain, 362
GetEnumerationSpeed, 362
GetGain, 362
MeaAudioFunctionNet, 366
MeaDigitalDataFunctionNet, 366
MeaFeedbackFunctionNet, 366
MeFunctionNet, 366

SetDigitalOut, 362
SetNumberOfAnalogChannels, 363
SetNumberOfChannels, 364
SetTriggerMaskValue, 364
SetTriggerPeriod, 365
W2100_FunctionNet, 366
WClassicFunctionNet, 366
CMeaDigitalDataFunctionNet, 366
CMeaDigitalDataFunctionNet, 367
GetDigitalData, 367
SetDigitalData, 367, 368
CMeaFeedbackFunctionNet, 368
CMeaFeedbackFunctionNet, 369
FeedbackGetSampleTimerCount, 369
FeedbackGetTriggerTotzeitFactor, 369
FeedbackSetAnalogSource, 370
FeedbackSetChannelFilter, 370
FeedbackSetCreateFilter, 370
FeedbackSetDigitalMapping, 370
FeedbackSetFeedback, 370
FeedbackSetFilterOff, 370
FeedbackSetFilterParameter, 370
FeedbackSetFilterParameter32, 370
FeedbackSetGlobalChannelFilter, 371
FeedbackSetIIRFilterParameter, 371
FeedbackSetLogic, 371
FeedbackSetMkFilter, 371
FeedbackSetNumberOfLogics, 371
FeedbackSetNumberOfRateCounter, 371
FeedbackSetNumberOfRateDetectors, 371
FeedbackSetNumberOfSpikeDetectors, 372
FeedbackSetNumberOfTriggers, 372
FeedbackSetRateCounter, 372
FeedbackSetRateDetector, 372
FeedbackSetSpikeDetectorThreshold, 372
FeedbackSetTrigger, 372
FeedbackSetTriggerTotzeitFactor, 372
CMeaImpedanceDeviceNet, 373
~CMeaImpedanceDeviceNet, 373
CMeaImpedanceDeviceNet, 373
GetAdapterCode, 373
GetArraySize, 373
GetImpedanceTestFrequency, 374
GetReady, 374
GetResult, 374
SetImpedanceTestFrequency, 374
StartMeasurement, 374
CMeasureTableDeviceNet, 374
CMeasureTableDeviceNet, 375
Sensor, 375
CMeaSwitchDeviceNet, 375
~CMeaSwitchDeviceNet, 376
CMeaSwitchDeviceNet, 376
GetNumber, 376
GetPattern, 376
GetPatternBool, 376
SetPattern, 376
SetPatternBool, 377
CMeaUSBDeviceNet, 377
~CMeaUSBDeviceNet, 378
CMeaUSBDeviceNet, 377, 378
SetVoltageRangeForIndex, 378
CMeFunctionNet, 378
!CMeFunctionNet, 379
~CMeFunctionNet, 379
CMeFunctionNet, 379
SetTransformer, 379
CMosMea
CCMOSMeaDeviceNet, 120
CmosMea
Mcs::Usb, 69
CMOSMEA5000DigitalSourceEnumNet
Mcs::Usb, 57
CMOSMeaBathModeEnumNet
Mcs::Usb, 58
CmosMeaHeadstage
Mcs::Usb, 68
CMOSMeaHeadstage1NCBathCurrentEnumNet
Mcs::Usb, 58
CMOSMeaHeadstage1NCCol2CurrentEnumNet
Mcs::Usb, 58
CMOSMeaHeadstage1NChipTempEnumNet
Mcs::Usb, 58
CMOSMeaHS1SidebandEnumNet
Mcs::Usb, 60
CMOSMeaHS1TriggerStatusEnumNet
Mcs::Usb, 60
CmosmeaIFB2
Mcs::Usb, 68
CMOSMeaIFDigChannelEnumNet
Mcs::Usb, 60
CMOSMeaInterfaceADCEnumNet
Mcs::Usb, 60
CmosMeaInterfaceboard
Mcs::Usb, 68
CMOSMeaPacketFrameContextGroupEnumNet
Mcs::Usb, 61
CMOSMeaSTG1DACSignalEnumNet
Mcs::Usb, 61
CMOSMeaValueUnitEnumNet
Mcs::Usb, 61
CMultiBatteryChargerDeviceNet, 379
!CMultiBatteryChargerDeviceNet, 381
~CMultiBatteryChargerDeviceNet, 381
CapacityTest, 381
ChannelReset, 381
CMultiBatteryChargerDeviceNet, 381
GetBatteryVoltage, 382
GetChannels, 382
GetChannelState, 382
GetChargeCapacity, 382
GetChargeCurrent, 383
GetChargingMode, 383
GetChargingPCoefficient, 383
GetDischargeCapacity, 383
GetDischargeCurrent, 384

GetDischargeCurrentSetPoint, 384
GetFinalDischargeVoltage, 384
GetRatedCapacity, 385
SetChargingMode, 385
SetChargingPCoefficient, 385
SetDischargeCurrentSetPoint, 385
SetFinalDischargeVoltage, 386
SetRatedCapacity, 386
SetRatedCapacityVolatile, 386
CMultiwellCallbackFunctionNet, 387
!CMultiwellCallbackFunctionNet, 388
~CMultiwellCallbackFunctionNet, 388
CMultiwellCallbackFunctionNet, 387
GetPlateClampStateByHeadstage, 388
GetPlateClampStateByHeadstageEvent, 388
OnGetPlateClampStateByHeadstage, 388
CMultiwellDeviceNet, 389
!CMultiwellDeviceNet, 390
~CMultiwellDeviceNet, 390
ClosePlateClamp, 391
CMultiwellDeviceNet, 390
GetPlateClampLockState, 391
GetPlateClampState, 391
GetPlateMux, 391, 392
GetPlateType, 392
GetPowerMuxPlate, 392
GetTouchPadEnable, 393
GetVolatileClampOffset, 393
IsPlateTypeValid, 393
LockPlateClamp, 394
OpenPlateClamp, 394
SetPlateMux, 394
SetPlateType, 395
SetPowerMuxPlate, 395
SetTouchPadEnable, 395
SetVolatileClampOffset, 396
StopPlateClamp, 396
UnlockPlateClamp, 396
CMultiwellOptoStimFunctionNet, 396
!CMultiwellOptoStimFunctionNet, 397
~CMultiwellOptoStimFunctionNet, 397
CMultiwellOptoStimFunctionNet, 397
GetAbsMaxCurrentInMicroAmp, 397
GetColorRgb, 398
GetColorStr, 398
GetMaxDurationHighCurrentInMicroSec, 398
GetMaxDutyCycleHighCurrent, 398
GetPermanentCurrentInMicroAmp, 399
GetWaveLengthInNanometer, 399
SetAbsMaxCurrentInMicroAmp, 399
SetColorRgb, 400
SetColorStr, 400
SetMaxDurationHighCurrentInMicroSec, 400
SetMaxDutyCycleHighCurrent, 400
SetPermanentCurrentInMicroAmp, 400
SetWaveLengthInNanometer, 401
CNF_GenDeviceNet, 401
~CNF_GenDeviceNet, 401
CNF_GenDeviceNet, 401
Set_Values, 402
COctoPotDeviceNet, 402
BurnAdcOffset, 403
BurnDacOffset, 403
COctoPotDeviceNet, 403
EnableChecksum, 403
EnableDigitalIn, 403
EnableTimestamp, 403
GetAdcOffset, 403
GetDacOffset, 403
PatternListStart, 404
RampStart, 404
ResetAdcOffset, 404
ResetDacOffset, 404
SetAdcOffset, 404
SetAmplificationSwitch, 404
SetBathclamp, 404
SetChannelSwitch, 404
SetDacAutoControl, 404
SetDacOffset, 405
SetDacValue, 405
SetNumberOfChannels, 405
SetOutputRate, 405
SetPatternListEntry, 405
SetPidParameter, 405
SetRampParameter, 405
SetSineParameter, 405
SineStart, 406
COkuvisionStimulatorDeviceNet, 406
~COkuvisionStimulatorDeviceNet, 407
COkuvisionStimulatorDeviceNet, 407
GetCheckVoltage, 407
GetCurrentFactor, 407
GetDACOffset, 407
GetMaxPower, 407
GetMaxVoltage, 407
GetPulseform, 407
GetRTC, 408
GetStimulatorStatus, 408
GetVoltage, 408
SetCheckVoltage, 408
SetCurrentFactor, 408
SetDACOffset, 408
SetMaxPower, 409
SetMaxVoltage, 409
SetPulseform, 409
SetRTC, 409
Coldstart
CMcsUsbFactoryNet, 306
CommaPositionA
CFilterCoefficientsNet::s_FilterAttributesNet, 717
Mcs::Usb, 67
CommaPositionB
CFilterCoefficientsNet::s_FilterAttributesNet, 717
Mcs::Usb, 67
CompareFirmware
CMcsUsbFactoryNet, 307

CompareTo
HeadStageIDType, 702
CompensateElectrodeOffset
CWarnerUssingFunctionNet, 640
Connect
CMcsUsbNet, 327, 328
CRFFunctionNet, 454
ConnectDevice
CRadioControledDevicesNet, 449
ConnectedImp
CPositionImpDeviceNet, 426
ConnectImp
CPositionImpDeviceNet, 426
ConnectSlave
CGilsonDeviceNet, 174
ControlState
HeadStageIDTypeState, 705
CornerFrequency
CFilterPropertyNet, 146
CornerFrequencymHz
CFilterPropertyNet, 146
Count
CMcsUsbListNet, 322
CPatchServerDeviceNet, 409
CPatchServerDeviceNet, 410
Sensor, 410
CPathIdentDeviceNet, 410
∼CPathIdentDeviceNet, 411
CPathIdentDeviceNet, 411
Measure, 411
Set_Values, 411
CPedoterDeviceNet, 411
!CPedoterDeviceNet, 412
∼CPedoterDeviceNet, 412
CPedoterDeviceNet, 412
GetCommand, 412
SetCommand, 412
CPeristalticPumpDeviceNet, 413
∼CPeristalticPumpDeviceNet, 413
CPeristalticPumpDeviceNet, 413
McsBus_MotorControl, 414
CPgaDeviceNet, 414
∼CPgaDeviceNet, 414
ApplyGains, 415
CPgaDeviceNet, 414
DefineAmplification, 415
DefineFrequencyRange, 415
DefineNumAmplifications, 415
DefineNumFrequencyRanges, 415
GetAmplification, 415
GetFrequencyRange, 415
GetGain, 416
GetNumAmplifications, 416
GetNumFrequencyRanges, 416
SetGain, 416
CPositionIIDeviceNet, 416
!CPositionIIDeviceNet, 418
∼CPositionIIDeviceNet, 418
CPositionIIDeviceNet, 418
GetCoilCommunication, 418
GetDebugData, 418
GetEventData, 419
GetImplantCurrentSetpoint, 419
GetImplantResult, 420
GetImplantState, 420
GetOnOff, 420
GetPowerStrength, 421
GetRTC, 421
GetStateDebugData, 421
GetStateEventData, 422
RFFunction, 425
SetImplantCurrentSetpoint, 422
SetPowerStrength, 422
SetRTC, 422
SetStateDebugData, 424
SetStateEventData, 424
SwitchOnOff, 424
CPositionImpDeviceNet, 425
!CPositionImpDeviceNet, 426
∼CPositionImpDeviceNet, 426
ConnectedImp, 426
ConnectImp, 426
CPositionImpDeviceNet, 426
GetDeviceList, 426
GetImpId, 427
GetRFFrequency, 427
SetDeviceList, 427
SetImpId, 427
SetRFFrequency, 428
CPPCDeviceNet, 428
CPPCDeviceNet, 428
McsBus, 429
McsBus_MotorControl, 429
McsBus_Sensor, 429
PPCFunction, 429
CPPCFunctionNet, 429
!CPPCFunctionNet, 431
∼CPPCFunctionNet, 431
CPPCFunctionNet, 430
FirePressurePulse, 431
GetAnalogVoltage, 431
GetAnalogVoltageRange, 431
GetDigitalIn, 433
GetPressureRange, 433
GetPumpModeType, 433
GetPumpSpeedUnit, 434
GetSupplyVoltage, 434
GetValveActive, 434
IsBusy, 434
LoadPressure, 436
MeasureReservoir, 436
SetAnalogVoltageRange, 436
SetPressureOffset, 436
SetPressureRange, 436
SetPumpModeType, 437
SetPumpSpeedUnit, 437

SetValveActive, 437
CPPS_DeviceNet, 438
CPPS_DeviceNet, 438
McsBus, 438
McsBus_MotorControl, 438
McsBus_Sensor, 438
PPS_Function, 438
CPPS_FunctionNet, 439
CPPS_FunctionNet, 439, 440
GetAnalogVoltage, 440
GetAnalogVoltages, 440
GetBubbleState, 440
GetDigitalIn, 440
GetPumpCouple, 440
GetPumpEnableSpeedRatio, 440
GetPumpFastOnOff, 440
GetPumpFastSpeed, 440
GetPumpFunctionSpeeds, 441
GetPumpManualOnOff, 441
GetPumpMaxSpeed, 441
GetPumpModeType, 441
GetPumpSpeedRatio, 441
GetPumpSpeedUnit, 441
GetSupplyVoltage, 441
GetUseBubble, 441
SetAnalogVoltages, 441
SetPumpCouple, 442
SetPumpEnableSpeedRatio, 442
SetPumpFastOnOff, 442
SetPumpFastSpeed, 442
SetPumpFunctionSpeeds, 442
SetPumpManualOnOff, 442
SetPumpMaxSpeed, 442
SetPumpModeType, 442
SetPumpSpeedRatio, 443
SetPumpSpeedUnit, 443
SetUseBubble, 443
CPPSDeviceNet, 443
CPPSDeviceNet, 444
CProgramPressureCurveNet, 444
!CProgramPressureCurveNet, 445
~CProgramPressureCurveNet, 444
CProgramPressureCurveNet, 444
GetRepeats, 445
Program, 445
SetRepeats, 445
CPulseGeneratorFunctionNet, 445
!CPulseGeneratorFunctionNet, 446
~CPulseGeneratorFunctionNet, 446
CPulseGeneratorFunctionNet, 446
GetModeSelect, 447
GetPeriod, 447
GetPulseLength, 447
SetModeSelect, 448
SetPeriod, 448
SetPulseLength, 448
CRadioControledDevicesNet, 448
ConnectDevice, 449
CRadioControledDevicesNet, 449
DisConnectDevice, 449
GetDeviceNames, 449
GetFrequency, 450
HasRadioControl, 450
SetFrequency, 450
StillConnected, 450
CreateSideband
CStimulusFunctionNet, 582
CreateWirelessHeadstageSerialNumberString
CWirelessBaseFunctionNet, 687
CRegionOfInterestRect
CCMOSMeaDeviceNet::CRegionOfInterestRect, 450
CRetinaLedDeviceNet, 451
~CRetinaLedDeviceNet, 452
AddLoopEntry, 452
AddTableEntry, 452
ClearTable, 452
CRetinaLedDeviceNet, 452
GetTablepointer, 452
SetLED, 452
SetLumi, 452
SetPersistency, 452
SetRepeat, 453
SetTablepointer, 453
SetTrigger, 453
CRFFunctionNet, 453
!CRFFunctionNet, 454
~CRFFunctionNet, 454
Connect, 454
CRFFunctionNet, 454
GetAvailableDeviceList, 455
GetAvailableDeviceListEx, 455
GetAvailableStateList, 455
GetAvailableStateListEx, 455
GetBaseFrequency, 456
GetConnectedDevice, 456
GetState, 456
GetTestMode, 456
GetWorkingFrequency, 457
SetBaseFrequency, 457
SetTestMode, 457
SetWorkingFrequency, 457
CRobo_FYIProgram_FunctionNet, 458
CRobo_FYIProgram_FunctionNet, 458
GetLength, 458
GetState, 459
GetValve1, 459
GetValve2, 459
SetLength, 459
SetValve1, 459
SetValve2, 459
Start, 459
CRobo_FYITemp_FunctionNet, 459
CRobo_FYITemp_FunctionNet, 460
GetICoeff, 460
GetMaxPower, 460

GetPCoeff, 460
GetRegulatorOnOff, 460
GetSollTemp, 460
SetICoeff, 461
SetMaxPower, 461
SetPCoeff, 461
SetRegulatorOnOff, 461
SetSollTemp, 461
CRoboDacqNet, 461
CancelTableLoop, 464
CancelTableLoopAndStopTable, 464
ClampAmpRestart, 464
CRoboDacqNet, 464
DoRamp, 464
Emu_GetCellCapacity, 465
Emu_GetCellPotential, 465
Emu_GetCellResists, 465
Emu_GetElectrodeResists, 465
Emu_GetNoise, 465
Emu_SetCellCapacity, 465
Emu_SetCellPotential, 465
Emu_SetCellResists, 465
Emu_SetElectrodeResists, 465
Emu_SetNoise, 465
GetAllDigout, 466
GetCapacityC, 466
GetCapacityV, 466
GetCapacityX, 466
GetClampAmpSerialNumber, 466
GetCommand, 466
GetConfigurationBit, 466
GetConfigurationBitAxc, 466
GetConfigurationBitBlu_Led, 466
GetConfigurationBitBlu_LedToggleFast, 466
GetConfigurationBitBlu_LedToggleSlow, 466
GetConfigurationBitCC_Gen, 467
GetConfigurationBitCV_Gen, 467
GetConfigurationBitRC_Gen, 467
GetConfigurationBitRed_Led, 467
GetConfigurationBitRed_LedSaturation, 467
GetConfigurationBitRed_LedToggleFast, 467
GetConfigurationBitRed_LedToggleSlow, 467
GetConfigurationBitRelais, 467
GetConfigurationBitRV_Gen, 467
GetConfigurationBits, 467
GetConfigurationBitStream, 467
GetConfigurationBitSupply, 468
GetCrossTalkOffset, 468
GetCrossTalkOptimum, 468
GetDigout, 468
GetDisplayText, 468
GetDownsampleFactor, 468
GetFilter, 468
GetFilterCoeffs, 468
GetIC, 468
GetIClamp, 468
GetICOffset, 468
GetIGain, 469
GetNIC_MS, 469
GetNUC_MS, 469
GetNUV_MS, 469
GetPGain, 469
GetRecordingNumber, 469
GetResistanceC, 469
GetResistanceV, 469
GetScreen, 469
GetSimulation, 469
GetUC, 469
GetUClamp, 470
GetUCOffset, 470
GetUpdateDisplay, 470
GetUV, 470
GetUVOffset, 470
GetXGain, 470
RunTable, 470
SetAllDigout, 470
SetCommand, 470
SetConfigurationBit, 470
SetConfigurationBitAxc, 471
SetConfigurationBitBlu_Led, 471
SetConfigurationBitBlu_LedToggleFast, 471
SetConfigurationBitBlu_LedToggleSlow, 471
SetConfigurationBitCC_Gen, 471
SetConfigurationBitCV_Gen, 471
SetConfigurationBitRC_Gen, 471
SetConfigurationBitRed_Led, 471
SetConfigurationBitRed_LedSaturation, 471
SetConfigurationBitRed_LedToggleFast, 472
SetConfigurationBitRed_LedToggleSlow, 472
SetConfigurationBitRelais, 472
SetConfigurationBitRV_Gen, 472
SetConfigurationBitStream, 472
SetConfigurationBitSupply, 472
SetCrossTalkOffset, 472
SetCrossTalkOptimum, 472
SetDigout, 472
SetDisplayText, 473
SetDownsampleFactor, 473
SetFilter, 473
SetFilterCoeffs, 473
SetIClamp, 473
SetICOffset, 473
SetIGain, 473
SetNoFilterCoeffs, 473
SetPGain, 473
SetRecordingNumber, 474
SetScreen, 474
SetSimulation, 474
SetTriggerMaskValue, 474
SetUClamp, 474
SetUCOffset, 474
SetUVOffset, 474
SetXGain, 474
StopTable, 474, 475
Table_Wait, 475
TableDefBegin, 475

TableDefEnd, 475
TriggerMask_Default, 475
TriggerValue_MoveAbs, 475
TriggerValue_StartQueue, 475
UpdateDisplay, 475
VirtualDevice_ContinousDacq, 475
VirtualDevice_TableRun, 475
CRoboDeviceNet, 476
~CRoboDeviceNet, 479
Axes_I, 487
Axes_X, 487
Axes_Y, 487
Axes_Z, 488
Axis_I, 488
Axis_X, 488
Axis_Y, 488
Axis_Z, 488
CancelPoolLoop, 479
CancelPoolLoopAndStopMovement, 479
CRoboDeviceNet, 479
EnableQueue, 480
FindReference, 480
GetAirpressure, 480
GetAirpressureLimit, 480
GetAirValve, 480
GetCurrentAirvalve, 480
GetCurrentAirvalveLimit, 480
GetCurrentPosition, 481
GetErrorAirpressure, 481
GetErrorCurrentAirvalve, 481
GetErrorVoltage12V, 481
GetErrorVoltage5V, 481
GetErrorVoltageAirvalve, 481
GetErrorVoltageRs485A, 481
GetErrorVoltageRs485B, 481
GetErrorVoltageValves, 482
GetInMovement, 482
GetMinPressure, 482
GetMovementError, 482
GetVoltage12V, 482
GetVoltage12VLimit, 482
GetVoltage5V, 482
GetVoltage5VLimit, 482
GetVoltageAirvalve, 482
GetVoltageAirvalveLimit, 482
GetVoltageRs485A, 483
GetVoltageRs485ALimit, 483
GetVoltageRs485B, 483
GetVoltageRs485BLimit, 483
GetVoltageValves, 483
GetVoltageValvesLimit, 483
IsQueueEnabled, 483
IsQueueStarted, 483
McsBus, 491
McsBus_MotorControl, 491
McsBus_XY, 488
McsBus_ZI, 488
MoveAbs, 483, 484
NullCommand, 485
RoboError_AnotherMaster, 488
RoboError_Base, 489
RoboError_CannotEscapeEndSwitch, 489
RoboError_CommandAlreadyInProgress, 489
RoboError_CommandNotPossible, 489
RoboError_CommunicationTimeout, 489
RoboError_DacqNotReady, 489
RoboError_DLLMovementTimeout, 489
RoboError_FindReferenceMethod, 489
RoboError_GilsonCommandPending, 489
RoboError_GilsonTimeout, 489
RoboError_GilsonWrondID, 490
RoboError_McsBus_UnknownCommand, 490
RoboError_NoEndSwitch, 490
RoboError_NoMoreData, 490
RoboError_NoReference, 490
RoboError_NoSpeedOrAcceleration, 490
RoboError_OverPressure, 490
RoboError_ParameterNotAllowed, 490
RoboError_PeristalticTimeout, 490
RoboError_Phase0OutOfRange, 490
RoboError_PollLoopCanceled, 491
RoboError_PollLoopCanceledAndStopMovement, 491
RoboError_Pressure, 491
RoboError_RangeExceeded, 491
RoboError_StateChangeNotPossible, 491
RoboError_Timeout, 491
RoboError_UnknownCommand, 491
RoboMainLowLevelCommand, 492
RoboStatusEvent, 492
SetAirpressureLimit, 485
SetAirValve, 485
SetCurrentAirvalveLimit, 485
SetCurrentAndAir, 485
SetInMovement, 485
SetMinPressure, 486
SetVoltage12VLimit, 486
SetVoltage5VLimit, 486
SetVoltageAirvalveLimit, 486
SetVoltageRs485ALimit, 486
SetVoltageRs485BLimit, 486
SetVoltageValvesLimit, 486
StartQueue, 486
StopMovement, 487
WaitTimer, 487
CRoboDeviceNet::RoboMainLowLevelCommands, 709
FindReferencePhase0, 710
GetAxisConfig, 710
GetHWConfig, 710
GetHWRevision, 710
GetMaxNoPressure, 710
GetMaxNoPressureWaitTime, 710
GetMaxPressureWaitTime, 710
GetMinNoPressureWaitTime, 710
GetMinPressure, 710
GetMinPressureWaitTime, 711

GetParameter, 711
GetPhases, 711
GetSearchReferenceFastAccel, 711
GetSearchReferenceFastSpeed, 711
GetSearchReferenceFineAccel, 711
GetSearchReferenceFineSpeed, 711
GetSearchReferenceMethod, 712
GetSearchReferenceMoveOut, 712
GetSearchReferenceOffsetPos, 712
GetUserParameter, 712
HasRef, 713
SetAxisConfig, 713
SetHWConfig, 713
SetHWRevision, 713
SetMaxNoPressure, 713
SetMaxNoPressureWaitTime, 713
SetMaxPressureWaitTime, 713
SetMinNoPressureWaitTime, 713
SetMinPressure, 713
SetMinPressureWaitTime, 713
SetParameter, 714
SetSearchReferenceFastAccel, 714
SetSearchReferenceFastSpeed, 714
SetSearchReferenceFineAccel, 714
SetSearchReferenceFineSpeed, 714
SetSearchReferenceMethod, 714
SetSearchReferenceMoveOut, 715
SetSearchReferenceOffsetPos, 715
SetUserParameter, 715
CRoboFluidDeviceNet, 492
~CRoboFluidDeviceNet, 493
CloseAllValves, 493
CRoboFluidDeviceNet, 493
GetPumpSpeed, 493
GetSingleValve, 493
GetValve, 493
IsPumpMotorOn, 494
m_pMcsBus_MotorControlNet, 495
m_pRoboFluidDevice, 495
McsBus_MotorControl, 495
PumpOff, 494
PumpOn, 494
SetPumpSpeed, 494
SetSingleValve, 494
SetValve, 494
CRoboInjectDeviceNet, 495
CRoboInjectDeviceNet, 496
CRoboocyte2DeviceNet, 496
CRoboocyte2DeviceNet, 497
GetAxisLED, 497
GetGilsonDevice, 497
GetMcsBus_Extension, 497
GetRoboDacq, 497
GetRoboFluidDevice, 497
SetAxisLED, 497
CRoboStatorDeviceNet, 497
CRoboStatorDeviceNet, 499
FindReferenceI, 499
FindReferenceXY, 499
FindReferenceZ, 499
GetCurrentPositionI, 499
GetCurrentPositionXY, 499
GetCurrentPositionZ, 500
HasRefI, 500
HasRefXY, 500
HasRefZ, 500
MoveAbsI, 500
MoveAbsXY, 500
MoveAbsZ, 500, 501
RoboMainStatorLowLevelCommand, 503
SetAccelerationI, 501
SetAccelerationNativeI, 501
SetAccelerationNativeXY, 501
SetAccelerationNativeZ, 501
SetAccelerationXY, 501
SetAccelerationZ, 501
SetCurrentAndAirXY, 501
SetSpeedI, 502
SetSpeedNativeI, 502
SetSpeedNativeXY, 502
SetSpeedNativeZ, 502
SetSpeedXY, 502
SetSpeedZ, 502
SetVelocityI, 502
SetVelocityXY, 502
SetVelocityZ, 502
StopMovementI, 503
StopMovementXY, 503
StopMovementZ, 503
CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands, 716
FindReferencePhase0XY, 716
CSafeISDeviceNet, 503
~CSafeISDeviceNet, 504
CSafeISDeviceNet, 504
DacqDevice, 506
FluidControlDevice, 506
RoboDevice, 506
SetAdcChannels, 505
SetAdcSamplePos, 505
SetDacMode, 505
SetDacPeriode, 505
SetDacPulseform, 505
SetSwitches, 506
csCapacityTestDischarge
Mcs::Usb, 71
csCapacityTestPrecharge
Mcs::Usb, 71
csCharge
Mcs::Usb, 71
CSCUDacqGroupChannelSelectionNet, 506
CSCUDacqGroupChannelSelectionNet, 507
CSCUFunctionNet, 507
!CSCUFunctionNet, 510
~CSCUFunctionNet, 510
AutomaticAnalogOut, 510

CSCUFunctionNet, 510
EnableAnalogOut, 510
GetAnalogOutADCRange, 511
GetAnalogOutChannels, 511
GetAnalogOutDACRange, 511
GetAvailableHeadstages, 511
GetAvailableHeadstagesEvent, 524
GetFilterProperties, 511
GetFilterProperty, 512
GetHeadstageAdcBits, 512
GetHeadstageAdcRangeInMicroVolt, 512
GetHeadstageDacBits, 513
GetHeadstageDacCurrentRangeInMicroAmpere, 513
GetHeadstageDacCurrentResolutionInNanoAmpere, 513
GetHeadstageDacVoltageRangeInMilliVolt, 514
GetHeadstageDacVoltageResolutionInMicroVolt, 514
GetHeadstageFrameCyclesToCompare, 514
GetHeadstageGainInPermille, 515
GetHeadstageID, 515
GetHeadstageLinkSpeed, 515
GetHeadstageNumberOfAnalogChannels, 516
GetHeadstageNumberOfStimulationChannels, 516
GetHeadstagePowerStateAtStart, 516
GetHeadstageSamplerate, 517
GetHeadstageSerialNumber, 517
GetMaxNumberOfHeadstages, 517
GetMaxStimulusChannelsPerHeadstage, 517
GetReferenceElectrodeMode, 518
GetReferenceElectrodeSwitchState, 518
HasAnalogOut, 518
HasGalvanicIsolation, 518
HasHSPowerSwitch, 519
IsAnalogOutEnabled, 519
IsAutomaticAnalogOut, 519
IsHeadstageAvailable, 519
IsHeadstageAvailableEvent, 524
IsHSPowered, 520
IsInDacqLegacyMode, 520
OnGetAvailableHeadstages, 520
OnIsHeadstageAvailable, 520
PowerHS, 520
SetAnalogOutADCRange, 521
SetAnalogOutChannels, 521
SetAnalogOutDACRange, 521
SetDacqLegacyMode, 521
SetHeadstageFrameCyclesToComparePermanent, 522
SetHeadstageLinkSpeedPermanent, 522
SetHeadstageNumberOfAnalogChannelsPermanent, 522
SetHeadstagePowerStateAtStart, 522
SetHeadstageSampleratePermanent, 523
SetReferenceElectrodeMode, 523
SetReferenceElectrodeSwitchState, 523
csDischarge
Mcs::Usb, 71
CSerialPortNet, 524
CSerialPortNet, 524
GetBytesAvailable, 525
Receive, 525
ReceiveString, 525
Send, 525
csError
Mcs::Usb, 71
csIdleChargeFinished
Mcs::Usb, 71
csIdleNoBattery
Mcs::Usb, 71
csRefreshBattery
Mcs::Usb, 71
CStg200xBasicNet, 525
~CStg200xBasicNet, 530
GetAutocalibrationDisabled, 530
GetAvailableMemory, 531
GetBlankingEnable, 531
GetCanDisableCalibration, 532
GetCanDisableCurrentswitch, 532
GetCurrentRangeInNanoAmp, 532
GetCurrentRangeInNanoAmpByIndex, 533
GetCurrentRangeListInNanoAmp, 533
GetCurrentRangeSelectedIndex, 533
GetCurrentResolutionInNanoAmp, 533
GetCurrentResolutionInPicoAmp, 534
GetDacAmplificationFactor, 534
GetDACResolution, 534
GetDiginValue, 534
GetDigoutMode, 535
GetDigoutValue, 535
GetElectrodeDacMux, 535
GetElectrodeEnable, 536
GetElectrodeMode, 537
GetEnableAmplifierProtectionSwitch, 538
GetExternalElectrodeEnable, 539
GetFAAmplification, 539
GetHasLowCurrentswitchLatency, 539
GetHeadstage, 539
GetListmodeIndexRange, 540
GetListmodeTriggerSource, 540
GetNumberOfAnalogChannels, 540
GetNumberOfCurrentRangeIndexes, 541
GetNumberOfHWDACPaths, 541
GetNumberOfStimulationElectrodes, 541
GetNumberOfStimulationSourcesPerElectrode, 541
GetNumberOfSyncoutChannels, 541
GetNumberOfTriggerInputs, 541
GetNumberOfVoltageRangeIndexes, 542
GetOutputRate, 542
GetStgProgramInfo, 542
GetStgVersionInfo, 543
GetSyncoutMap, 543
GetTimeResolutionInNanoSeconds, 543
GetTotalMemory, 544

GetTriggerSource, 544
GetVoltageRangeInMicroVolt, 544
GetVoltageRangeInMicroVoltByIndex, 544
GetVoltageRangeListInMicroVolt, 545
GetVoltageRangeSelectedIndex, 545
GetVoltageResolutionInMicroVolt, 545
HasDigitalPort, 545
HasFilterAmp, 545
LegacyHasOutputMap, 546
ListModeSendStart, 546
ListModeSendStop, 546
SendStart, 546
SendStop, 547
SetAutocalibrationDisabled, 547
SetBlankingEnable, 547, 548
SetCurrentMode, 549
SetCurrentRangeSelectedIndex, 549
SetDacAmplificationFactor, 549
SetDigoutMode, 551
SetDigoutValue, 551
SetElectrodeDacMux, 551–553
SetElectrodeEnable, 553–555
SetElectrodeMode, 555, 557, 558
SetEnableAmplifierProtectionSwitch, 558, 559
SetExternalElectrodeEnable, 560
SetFAAmplification, 560
SetHeadstage, 561
SetListmodeIndexRange, 561
SetListmodeTriggerSource, 561, 562
SetMeasurementMode, 562
SetOutputRate, 562
SetStgProgramInfo, 563
SetSyncoutMap, 563
SetTriggerSource, 563
SetVoltageMode, 563, 564
SetVoltageRangeSelectedIndex, 564
CStg200xDownloadBasicNet, 564
ClearChannelData, 566
ClearSyncData, 566
DisableAutoReset, 566
EnableAutoReset, 566
ForceStatusEvent, 567
GetMemoryUsageDAC, 567
GetMemoryUsageSyncout, 567
GetSweepCount, 567
GetTrigger, 568
ResetStatus, 568
SendChannelData, 569
SendSyncData, 569
SetupRetriggerMode, 569, 570
SetupTrigger, 570
SetupTriggerSingle, 572
Stimulus, 573
CStg200xDownloadNet, 573
~CStg200xDownloadNet, 574
ClearChannel_PrepareAndSendData, 574
CStg200xDownloadNet, 574
DisableMultiFileMode, 575
EnableMultiFileMode, 575
GetModuleCurrent, 576
GetModuleTemp, 576
MwPollStatusEvent, 579
PrepareAndAppendData, 576
PrepareAndSendData, 577
QueryTriggerstatus, 578
SegmentDefine, 578
SegmentSelect, 578
SegmentStart, 579
SetOutputMap, 579
Stg200xPollStatusEvent, 579
CStimulusFunctionNet, 580
ClearChannel_PrepareAndSendData, 582
ClearChannelData, 582
ClearMultiplexedData, 582
ClearSyncData, 582
CreateSideband, 582
CStimulusFunctionNet, 581
ForceStatusEvent, 583
GetAvailableMemory, 583
GetCurrentRangeInNanoAmp, 584
GetCurrentResolutionInNanoAmp, 584
GetCurrentResolutionInPicoAmp, 584
GetDACResolution, 585
GetMultiplexedDataChannelsInBlock, 585
GetNumberOfAnalogChannels, 585
GetTotalMemory, 585
GetVoltageRangeInMicroVolt, 585
GetVoltageResolutionInMicroVolt, 586
PollStatusEvent, 591
PrepareAndAppendData, 586
PrepareAndSendData, 587
PrepareData, 588
SendMultiplexedData, 588
SendPreparedData, 588
SendStart, 588
SendStop, 589
SetupTrigger, 589
SetupTriggerSingle, 590
StartPoll, 590
StopPoll, 590
CStimulusFunctionNet::SidebandData, 718
!SidebandData, 718
~SidebandData, 718
Duration, 718
Sideband, 718
SidebandData, 718
CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, 719
!StimulusDeviceDataAndUnrolledData, 720
~StimulusDeviceDataAndUnrolledData, 720
DeviceData, 720
DeviceDataLength, 720
StimulusDeviceDataAndUnrolledData, 720
UnrolledAmplitude, 720
UnrolledDuration, 720
UnrolledSync, 720

CSw2to64DeviceNet, 591
~CSw2to64DeviceNet, 592
CSw2to64DeviceNet, 592
GetChannel, 592
GetChannels, 592
GetNumber, 592
SetChannel, 592
SetChannels, 593
CTcxDeviceNet, 593
~CTcxDeviceNet, 596
CalibrateThermocouple, 596
CTcxDeviceNet, 595
FactoryReset, 596
GetBoardTemp, 596
GetCalibration, 596
GetCalibrationDecp, 596
GetCalibrationMax, 596
GetCalibrationMin, 596
GetCurrent, 596
GetD, 597
GetDDecp, 597
GetDevice, 597
GetDeviceType, 597
GetDevname, 597
GetDMax, 597
GetDMin, 597
GetDuty, 597
GetEnableHeaterLimit, 598
GetEnableThermocouple, 598
GetHasThermocouple, 598
GetHeaterLimit, 598
GetHeaterTemp, 598
GetI, 598
GetIDecp, 598
GetIMax, 599
GetIMin, 599
GetIOut, 599
GetMaxHeaterPowerMultiwell, 599
GetMaxP, 599
GetMaxpDecp, 599
GetMaxpMax, 599
GetMaxpMin, 600
GetNumControlChannels, 600
GetNumDevices, 600
GetNumMeasureChannels, 600
GetOnOff, 600
GetP, 600
GetPDecp, 600
GetPMax, 600
GetPMin, 600
GetPOut, 600
GetPwrOut, 601
GetPwrSet, 601
GetRes1, 601
GetRes2, 601
GetResS, 601
GetResX, 601
GetROut, 601
GetSensorType, 602
GetSetpoint, 602
GetSetpointDecp, 602
GetSetpointMax, 602
GetSetpointMin, 602
GetThermocoupleCalibration, 602
GetThermocoupleNanovoltPerKelvin, 602
GetThermocoupleReferenceTemp, 603
GetThermocoupleTemp, 603
GetThermocoupleTempAbs, 603
GetUnit, 603
GetUOut, 603
GetValue, 603
GetValueHires, 604
GetVolti, 604
SetCalibration, 604
SetD, 604
SetDevice, 604
SetDeviceType, 604
SetDevname, 604
SetEnableHeaterLimit, 604
SetEnableThermocouple, 605
SetHeaterLimit, 605
SetI, 605
SetMaxHeaterPowerMultiwell, 605
SetMaxP, 605
SetOnOff, 605
SetP, 606
SetSensorType, 606
SetSetpoint, 606
SetThermocoupleNanovoltPerKelvin, 606
CTEERFunctionNet, 606
!CTEERFunctionNet, 609
~CTEERFunctionNet, 609
CancelInternalCalibration, 609
CTEERFunctionNet, 608
GetAdapterCode, 609
GetAdcOffsetU1, 609
GetAdcOffsetU2, 609
GetAmplitude_nA, 609
GetBytesPerSample, 610
GetClampMode, 610
GetControllerParams, 610
GetCurrentEnable, 610
GetDacZero, 611
GetFrameErrorCounter, 611
GetLiquidResistance, 611
GetMaxChunkSize_Byte, 611
GetNumberOfAvailableSamples, 611
GetPeriod_us, 612
GetRotaryPositionCode, 612
GetSampleBufferChunk, 612
GetSampleRate, 612
GetSampleVoltageBuffer_uV, 613
GetScaleFactorU1, 613
GetScaleFactorU2, 613
GetUptimeSeconds, 613
GetWaveform, 613

IsInternalCalibrationFinished, 614
IsSamplingFinished, 614
SetAmplitude_nA, 614
SetBufferIndex, 614
SetClampMode, 615
SetControllerParams, 615
SetCurrentEnable, 615
SetExternalLED, 615
SetLiquidResistance, 616
SetPeriod_us, 616
SetWaveform, 616
StartInternalCalibration, 616
StartSampling, 616
StopSampling, 617
CTEERMachineDeviceNet, 617
~CTEERMachineDeviceNet, 617
CTEERMachineDeviceNet, 617
TEERFunctionNet, 618
CurrentClamp
Mcs::Usb, 92
CurrentMeasure
Mcs::Usb, 58
CurrentRangeInNanoAmp
W2100_StimulusParametersNet, 722
CurrentResolutionInNanoAmp
W2100_StimulusParametersNet, 722
CUsbDeviceConfigurationFunctionNet, 618
!CUsbDeviceConfigurationFunctionNet, 619
~CUsbDeviceConfigurationFunctionNet, 619
CUsbDeviceConfigurationFunctionNet, 618
SetDeviceId, 619
SetDeviceName, 619
CUsbExceptionNet, 619
CUsbExceptionNet, 620
Status, 620
CutoffOrCenterFrequency
CCreateFilterNet, 122
CVoltageRangeInfoNet
CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet, 621
CW2100_FunctionNet, 621
ClearStimulusParametersCache, 623
ClearUserDefinedNameCache, 623
CW2100_FunctionNet, 623
DeselectAllHeadstages, 624
DeselectHeadstage, 624
GetAccelGyroCurrentRate, 624
GetAccelGyroDesiredRate, 624
GetAccelGyroEnabled, 624
GetAccelRange, 624
GetAnalogOutChannel, 624
GetAnalogOutFilter, 624
GetAudioChannels, 624
GetAvailableHeadstages, 624
GetBatteryState, 625
GetDacRange, 625
GetFilterProperties, 625
GetFilterProperty, 625
GetFPGAFirmwareType, 625
GetGyroRange, 625
GetHeadstageOnOff, 625
GetHeadstageSamplingActive, 625
GetMultiHeadstageMode, 625
GetPicFirmwareType, 625
GetSelectedChannels, 626
GetSelectedHeadstageState, 626
GetStimulusParametersCache, 626
GetStimulusParametersFromSelectedHS, 626
GetStiumlusParameters, 626
GetUserDefinedName, 626
GetUserDefinedNameCache, 626
GetUserDefinedNameFromSelectedHS, 627
PulseGenerator, 628
SelectHeadstage, 627
SetAccelGyroDesiredRate, 627
SetAccelGyroEnabled, 627
SetAccelRange, 627
SetAnalogOutChannel, 627
SetAnalogOutFilter, 627
SetAudioChannels, 627
SetDacRange, 628
SetGyroRange, 628
SetHeadstageOnOff, 628
SetHeadstageSamplingActive, 628
SetHeadstageToSleep, 628
SetMultiHeadstageMode, 628
SetSelectedChannels, 628
Stimulator, 629
CW2100_FunctionNet::AudioChannelsNet, 97
amplification, 97
channel, 97
dacqgroup, 97
CW2100_StimulatorFunctionNet, 629
BOOST_BIT, 635
ClearChannelData, 630
CW2100_StimulatorFunctionNet, 630
GetBoostAlwaysOnMode, 631
GetBoostPreTime, 631
GetCurrentRangeInNanoAmp, 631
GetCurrentResolutionInNanoAmp, 631
GetCurrentResolutionInPicoAmp, 631
GetDACResolution, 632
GetDigitalStimulatorTrigger, 632
GetDigitalStimulatorTriggerSlope, 632
GetNumberOfAnalogChannels, 632
GetNumberOfSyncoutChannels, 632
GetNumberOfTriggerInputs, 632
GetStimulationPatternMemory, 632
GetTimeResolutionInNanoSeconds, 633
GetTimeSlot, 633
GetVoltageRangeInMicroVolt, 633
GetVoltageResolutionInMicroVolt, 633
GND_SWITCH_BIT, 635
PollStatusEvent, 636
PrepareData, 633
PrepareDataSync, 634

SelectTimeSlot, 634
SendPreparedData, 634
SendStart, 634
SendStop, 634
SetDigitalStimulatorTrigger, 635
SetDigitalStimulatorTriggerSlope, 635
StartPoll, 635
StopPoll, 635
SYNC_BIT0, 635
SYNC_BIT1, 635
CW2100DacqGroupChannelSelectionNet, 636
CW2100DacqGroupChannelSelectionNet, 636
CWarnerUssingDeviceNet, 636
!CWarnerUssingDeviceNet, 637
~CWarnerUssingDeviceNet, 637
CWarnerUssingDeviceNet, 637
WarnerUssingFunction, 637
CWarnerUssingFunctionNet, 638
!CWarnerUssingFunctionNet, 640
~CWarnerUssingFunctionNet, 640
CompensateElectrodeOffset, 640
CWarnerUssingFunctionNet, 640
GetAvailableChambers, 641
GetChannelsCountOfChamber, 641
GetClampMode, 641
GetComplianceVoltageThreshold, 641
GetDacPampsPerDigitHighCurrentRange, 642
GetDacPampsPerDigitLowCurrentRange, 642
GetDacZero, 642
GetHighCurrentRange, 643
GetIdleModeOffset, 643
GetLiquidResistance, 643
GetLowCurrentRange, 644
GetNumberOfAvailableChambers, 644
GetNumberOfHardwareSlotsForChambers, 644
GetU1Offset, 644
GetU1Reference, 645
GetU2Offset, 645
GetU2Reference, 645
GetUnitDescription, 646
GetUnitExponent, 646
GetUnitName, 646
GetUnitsPerDigit, 647
GetUptimeSeconds, 647
GetVoltageClampControllerParam_D, 647
GetVoltageClampControllerParam_I, 648
GetVoltageClampControllerParam_P, 648
IsChamberAvailable, 648
IsHighCurrentMode, 649
IsInternalCalibrationFinished, 649
IsPulseEnabled, 649
SetClampMode, 650
SetEnablePulse, 650
SetHighCurrentMode, 650
SetIdleModeOffset, 651
SetLiquidResistance, 651
SetLowCurrentMode, 651
SetPulse, 651
SetVoltageClampControllerParam_D, 652
SetVoltageClampControllerParam_I, 652
SetVoltageClampControllerParam_P, 652
WaitForAllChambers, 653
WaitForChamber, 653
CWarnerValveControllerDeviceNet, 653
!CWarnerValveControllerDeviceNet, 658
~CWarnerValveControllerDeviceNet, 658
ClearTableName, 658
ClearValveTable, 658
CWarnerValveControllerDeviceNet, 658
GetActiveRunningTableNumber, 658
GetActiveRunningTableNumberEvent, 675
GetAnalogThresholdHigh, 658
GetAnalogThresholdHighEvent, 675
GetAnalogThresholdLow, 659
GetAnalogThresholdLowEvent, 675
GetAnalogVoltage, 659
GetAnalogVoltageEvent, 675
GetCurrentEditTableNumber, 659
GetCurrentNumberOfValves, 659
GetCurrentNumberOfValvesEvent, 676
GetDigitalOutPortValve, 660
GetDigitalOutPortValveEvent, 676
GetDigitalPortDirection, 660
GetDigitalPortDirectionEvent, 676
GetDisplayMode, 660
GetDisplayModeEvent, 676
GetTableName, 660
GetTableNamebyIndex, 661
GetTableNamebyIndexEvent, 676
GetTotalNumberOfDigitalPorts, 661
GetTotalNumberOfTables, 661
GetTotalNumberOfValves, 661
GetTotalTableSize, 662
GetValveActive, 662
GetValveActiveEvent, 676
GetValveBoardRevision, 662
GetValveBoardRevisionEvent, 676
GetValveBoardRevisionString, 662
GetValveCurrent, 662
GetValveDigitalInPort, 663
GetValveDigitalInPortEvent, 677
GetValveLedOn, 663
GetValveLedOnEvent, 677
GetValveManualGroup, 663
GetValveManualGroupEvent, 677
GetValveManualState, 664
GetValveManualStateEvent, 677
GetValveMode, 664
GetValveModeEvent, 677
GetValvesActiveMap, 664
GetValvesManualStateMap, 664
GetValveTableEntry, 665
IsDigitalOutPortInverted, 665
IsDigitalOutPortInvertedEvent, 677
IsValveDigitalInInverted, 665
IsValveDigitalInInvertedEvent, 677

IsValveOpen, 666
IsValveOpenEvent, 678
IsValveOpenInAnalogMode, 666
IsValveOpenInAnalogModeEvent, 678
IsValveOpenInDigitalMode, 666
IsValveOpenInDigitalModeEvent, 678
LoadValveTable, 666
OnGetActiveRunningTableNumber, 667
OnGetAnalogThresholdHigh, 667
OnGetAnalogThresholdLow, 667
OnGetAnalogVoltage, 667
OnGetCurrentNumberOfValves, 667
OnGetDigitalOutPortValve, 667
OnGetDigitalPortDirection, 667
OnGetDisplayMode, 667
OnGetTableNamebyIndex, 667
OnGetValveActive, 668
OnGetValveBoardRevision, 668
OnGetValveDigitalInPort, 668
OnGetValveLedOn, 668
OnGetValveManualGroup, 668
OnGetValveManualState, 668
OnGetValveMode, 668
OnIsDigitalOutPortInverted, 668
OnIsValveDigitalInInverted, 669
OnIsValveOpen, 669
OnIsValveOpenInAnalogMode, 669
OnIsValveOpenInDigitalMode, 669
OnTableEntryChanged, 669
SetActiveRunningTableNumber, 669
SetAnalogThresholdHigh, 669
SetAnalogThresholdLow, 670
SetCurrentEditTableNumber, 670
SetDefault, 670
SetDigitalOutPortInvert, 670
SetDigitalOutPortValve, 671
SetDigitalPortDirection, 671
SetDisplayMode, 671
SetTableName, 671
SetTableStep, 672
SetTableStepAll, 672
SetValveActive, 672
SetValveCurrent, 672
SetValveDigitalInInvert, 673
SetValveDigitalInPort, 673
SetValveLedOn, 673
SetValveManualGroup, 673
SetValveManualState, 674
SetValveMode, 674
SetValvesActiveMap, 674
SetValvesManualStateMap, 674
SetValveTableEntry, 674
StoreValveTable, 675
TableEntryChangedEvent, 678
CWarnerValveControllerDeviceTesterFunctionNet, 678
!CWarnerValveControllerDeviceTesterFunctionNet, 680
~CWarnerValveControllerDeviceTesterFunctionNet, 679
CWarnerValveControllerDeviceTesterFunctionNet, 679
GetIO, 680
GetSync, 680
SetADC, 680
SetIO, 680
SetIODirection, 681
SetTrigger, 681
SetTriggerSyncDirection, 681
CWClassicFunctionNet, 681
CWClassicFunctionNet, 682, 683
GetFilterParametersHeadstage, 683
GetHasChecksum, 683
GetHasRedLedHeadstage, 683
GetHeadstageOnOff, 683
GetResetFilter, 683
GetRFConnectionStatus, 683
GetRFFrequencyHeadstage, 683
GetRFFrequencyReceiver, 683
GetRFPower, 684
GetScanHeadstagesResult, 684
GetSelectedHeadstage, 684
GetSerialNumberHeadstage, 684
GetWPADebugMode, 684
GetWPAType, 684
ResetChannelmap, 684
ScanForHeadstages, 684
SetChannelmap, 684
SetFilterParametersHeadstage, 684
SetHasChecksum, 685
SetHeadstageOnOff, 685
SetHWSelectedChannels, 685
SetResetFilter, 685
SetRFFrequencyHeadstage, 685
SetRFFrequencyReceiver, 685
SetRFFrequencyReceiverEeprom, 685
SetRFLostBehaviour, 685
SetRFPower, 686
SetSelectedHeadstage, 686
SetSerialNumberHeadstage, 686
SetWPADebugMode, 686
SetWPAType, 686
CWirelessBaseFunctionNet, 686
CreateWirelessHeadstageSerialNumberString, 687
CWirelessBaseFunctionNet, 687
CyclePort
CMcsUsbNet, 329
Cypress
Mcs::Usb, 92
Cypress_FX1
Mcs::Usb, 78
Cypress_FX2
Mcs::Usb, 78
Cypress_FX3
Mcs::Usb, 78

DAC1Channel
Mcs::Usb, 61
DAC2Channel
Mcs::Usb, 61
DAC3Channel
Mcs::Usb, 61
DAC4Channel
Mcs::Usb, 61
DACQ1DigitalGroup
Mcs::Usb, 62
DacqDevice
CSafeISDeviceNet, 506
dacqgroup
CW2100_FunctionNet::AudioChannelsNet, 97
DacqGroupChannelEnumNet
Mcs::Usb, 62
DacqMeaGroupTypeEnumNet
Mcs::Usb, 62
DacqTrigger
Mcs::Usb, 65
DACResolution
W2100_StimulusParametersNet, 722
DataModeEnumNet
Mcs::Usb, 62
DataState
HeadStageIDTypeState, 705
DeepCopy
CCMOSMeaDeviceNet::CRegionOfInterestRect, 450
DefineAmplification
CPgaDeviceNet, 415
DefineFrequencyRange
CPgaDeviceNet, 415
DefineNumAmplifications
CPgaDeviceNet, 415
DefineNumFrequencyRanges
CPgaDeviceNet, 415
DeselectAllHeadstages
CW2100_FunctionNet, 624
DeselectHeadstage
CW2100_FunctionNet, 624
DEST_FX3_TARGET_MASK
Mcs::Usb, 57
DEST_TARGET1
Mcs::Usb, 57
DEST_TARGET10
Mcs::Usb, 57
DEST_TARGET11
Mcs::Usb, 57
DEST_TARGET12
Mcs::Usb, 57
DEST_TARGET13
Mcs::Usb, 57
DEST_TARGET14
Mcs::Usb, 57
DEST_TARGET15
Mcs::Usb, 57
DEST_TARGET2
Mcs::Usb, 57
DEST_TARGET3
Mcs::Usb, 57
DEST_TARGET4
Mcs::Usb, 57
DEST_TARGET5
Mcs::Usb, 57
DEST_TARGET6
Mcs::Usb, 57
DEST_TARGET7
Mcs::Usb, 57
DEST_TARGET8
Mcs::Usb, 57
DEST_TARGET9
Mcs::Usb, 57
DEST_TARGET_MASK
Mcs::Usb, 57
DetectChipType
CCMOSMea_FunctionNet, 108
DEVICE_NOT_FOUND
Mcs::Usb, 66
DeviceArrival
CMcsUsbListNet, 322
DeviceData
CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, 720
DeviceDataLength
CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, 720
DeviceEnumNet
Mcs::Usb, 63
DeviceHasNoHeadstage
Mcs::Usb, 69, 83
DeviceId
CMcsUsbListEntryNet, 319
DeviceIdNet, 687
BcdDevice, 688
BusType, 688
DeviceIdNet, 688
IdProduct, 688
IdVendor, 688
operator=, 688
DeviceName
CMcsUsbListEntryNet, 319
DeviceNotConnected
Mcs::Usb, 68, 69, 83
DeviceRemoval
CMcsUsbListNet, 322
DeviceRunStatus
Mcs::Usb, 58, 73, 75, 76, 84, 88, 95
DigDataFromReceiver
Mcs::Usb, 95
Digital
Mcs::Usb, 78, 95
DigitalData
Mcs::Usb, 58, 73, 75, 76, 84, 88, 95
DigitalDatastreamEnableEnumNet
Mcs::Usb, 64

DigitalGroup
Mcs::Usb, 62
DigitalIn
Mcs::Usb, 57, 64, 72, 74, 76, 84, 88, 94
DigitalInOfOutPort
Mcs::Usb, 57, 72, 74, 76, 84, 88, 94
DigitalInPort
Mcs::Usb, 60
DigitalInReserverd
Mcs::Usb, 64
DigitalMux
Mcs::Usb, 60
DigitalOut
Mcs::Usb, 64
DigitalOutReg
Mcs::Usb, 60
DigitalOutReserved
Mcs::Usb, 64
DigitalOutStimulator
Mcs::Usb, 58, 73, 75, 76, 84, 88, 95
DigitalPulse
Mcs::Usb, 57, 72, 74, 76, 84, 88, 95
DigitalReg
Mcs::Usb, 60
DigitalSource
DigitalSource< digitalsourceenum >, 689
DigitalSource< digitalsourceenum >, 689
DigitalSource, 689
MaxBitNumber, 689
MaxBitNumberStatic, 689
size, 689
Source, 690
DigitalSourceGeneral, 690
DigitalSourceGeneral, 690
MaxBitNumber, 690, 691
size, 691
Source, 691
DigitalStimulatorTriggerEventEnumNet
Mcs::Usb, 65
DigitalStimulatorTriggerSlopeEnumNet
Mcs::Usb, 65
DigitalTargetEnumNet
Mcs::Usb, 65
Digout
Mcs::Usb, 65
DigOutStim
Mcs::Usb, 64
DigOutStimulatorStartTrigger
Mcs::Usb, 65
DigOutStimulatorStopTrigger
Mcs::Usb, 65
Digstream
Mcs::Usb, 65
DigStreamFromReceiver
Mcs::Usb, 95
DigStreamToReceiver
Mcs::Usb, 65
Dilutor
Mcs::Usb, 80
DisableAutoReset
CStg200xDownloadBasicNet, 566
DisableMultiFileMode
CStg200xDownloadNet, 575
Disconnect
CMcsUsbNet, 329
DisConnectDevice
CRadioControledDevicesNet, 449
DongleS
Mcs::Usb, 79
DoRamp
CRoboDacqNet, 464
Dotriapot
Mcs::Usb, 78
DoubleToInt
Mcs::Usb, 67
DownloadFirmware
CMcsUsbFactoryNet, 307
DownloadOnly
Mcs::Usb, 86
DriverVersionNet, 691
~DriverVersionNet, 692
DriverVersionNet, 692
DriverVersionNet::FormatVersion, 692
GetDestinationCode, 693
GetDestinationName, 694
GetMajor, 694
GetMinor, 695
GetNumEntries, 695
GetSerialNumber, 695
GetStatus, 696
GetVersionInt, 696
GetVersionString, 697
DriverVersionNet::FormatVersion
DriverVersionNet, 692
DSP
FirmwareDestinationNames, 698
Mcs::Usb, 54
DSPAnalogGroup
Mcs::Usb, 72, 84
DSPDataGroup
Mcs::Usb, 62, 93
DSPDigitalGroup
Mcs::Usb, 72, 84
DummyCommand
CLIH3DeviceNet, 202
Duration
CStimulusFunctionNet::SidebandData, 718

eCube
Mcs::Usb, 80, 83
eCubeHeadstage
Mcs::Usb, 68
ElectricalStimulation
HeadStageIDType, 701
ElectrodeDacMuxEnumNet
Mcs::Usb, 65
ElectrodeModeEnumNet

Mcs::Usb, 66
ElectrodeOffset
Mcs::Usb, 92
emAutomatic
Mcs::Usb, 66
emManual
Mcs::Usb, 66
EmptyKey
CMcsUsbNet, 329
Emu_GetCellCapacity
CRoboDacqNet, 465
Emu_GetCellPotential
CRoboDacqNet, 465
Emu_GetCellResists
CRoboDacqNet, 465
Emu_GetElectrodeResists
CRoboDacqNet, 465
Emu_GetNoise
CRoboDacqNet, 465
Emu_SetCellCapacity
CRoboDacqNet, 465
Emu_SetCellPotential
CRoboDacqNet, 465
Emu_SetCellResists
CRoboDacqNet, 465
Emu_SetElectrodeResists
CRoboDacqNet, 465
Emu_SetNoise
CRoboDacqNet, 465
EnableAnalogOut
CSCUFunctionNet, 510
EnableAutoReset
CStg200xDownloadBasicNet, 566
EnableChannelsInGroup
CCMOSMea_FunctionNet, 108, 109
CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet >, 127
EnableChecksum
CMeaDeviceNet, 360
COctoPotDeviceNet, 403
EnableDigitalIn
CMeaDeviceNet, 361
COctoPotDeviceNet, 403
EnableExceptions
CMcsUsbNet, 329
EnableMultiFileMode
CStg200xDownloadNet, 575
EnableQueue
CRoboDeviceNet, 480
EnableTimestamp
CMeaDeviceNet, 362
COctoPotDeviceNet, 403
EnableUserTrigger
CLIH3DeviceNet, 202
Encapsulator
Mcs::Usb, 80
enCMosMeaChipType
Mcs::Usb, 66
EnSTG200x_STATUS
Mcs::Usb, 66
Entry
HeadStageIDType, 702
EOFAndCRC
Mcs::Usb, 61
Equals
CMcsUsbListEntryNet, 316
HeadStageIDType, 702
HeadstageIDTypeObject, 704
EraseEepromRegisterPreconfig
CMcsUsbNet, 329
EraseFilterParameterPermanent
CFilterConfigurationNet, 142
CFilterConfigurationRegisterNet, 144
ErasePermanentAdcOffset
CLIH3DeviceNet, 203
ErasePermanentDacOffset
CLIH3DeviceNet, 203
Error_Callback_Aquisition_Stopped
CMcsUsbDacqNet, 301
Error_Callback_Data_lost
CMcsUsbDacqNet, 301
Error_Callback_Frames_Lost
CMcsUsbDacqNet, 301
Error_Callback_Packet_Error
CMcsUsbDacqNet, 301
Error_Callback_Queue_Full
CMcsUsbDacqNet, 302
Error_Callback_RingQueue_Full
CMcsUsbDacqNet, 302
ErrorEvent
CMcsUsbDacqNet, 302
ExtendedMultiFileMode
Mcs::Usb, 86
ExternBCTester
Mcs::Usb, 79
ExternDTester
Mcs::Usb, 79
ExternSTester
Mcs::Usb, 79

FactoryReset
CTcxDeviceNet, 596
Falling
Mcs::Usb, 65
FCB
Mcs::Usb, 79
FCX
Mcs::Usb, 79
Feedback
Mcs::Usb, 57, 73, 74, 76, 84, 88, 95
FeedbackGetSampleTimerCount
CMeaFeedbackFunctionNet, 369
FeedbackGetTriggerTotzeitFactor
CMeaFeedbackFunctionNet, 369
FeedbackHigh

Mcs::Usb, 64
FeedbackLow
Mcs::Usb, 64
FeedbackReg
Mcs::Usb, 60
FeedbackSetAnalogSource
CMeaFeedbackFunctionNet, 370
FeedbackSetChannelFilter
CMeaFeedbackFunctionNet, 370
FeedbackSetCreateFilter
CMeaFeedbackFunctionNet, 370
FeedbackSetDigitalMapping
CMeaFeedbackFunctionNet, 370
FeedbackSetFeedback
CMeaFeedbackFunctionNet, 370
FeedbackSetFilterOff
CMeaFeedbackFunctionNet, 370
FeedbackSetFilterParameter
CMeaFeedbackFunctionNet, 370
FeedbackSetFilterParameter32
CMeaFeedbackFunctionNet, 370
FeedbackSetGlobalChannelFilter
CMeaFeedbackFunctionNet, 371
FeedbackSetIIRFilterParameter
CMeaFeedbackFunctionNet, 371
FeedbackSetLogic
CMeaFeedbackFunctionNet, 371
FeedbackSetMkFilter
CMeaFeedbackFunctionNet, 371
FeedbackSetNumberOfLogics
CMeaFeedbackFunctionNet, 371
FeedbackSetNumberOfRateCounter
CMeaFeedbackFunctionNet, 371
FeedbackSetNumberOfRateDetectors
CMeaFeedbackFunctionNet, 371
FeedbackSetNumberOfSpikeDetectors
CMeaFeedbackFunctionNet, 372
FeedbackSetNumberOfTriggers
CMeaFeedbackFunctionNet, 372
FeedbackSetRateCounter
CMeaFeedbackFunctionNet, 372
FeedbackSetRateDetector
CMeaFeedbackFunctionNet, 372
FeedbackSetSpikeDetectorThreshold
CMeaFeedbackFunctionNet, 372
FeedbackSetTrigger
CMeaFeedbackFunctionNet, 372
FeedbackSetTriggerTotzeitFactor
CMeaFeedbackFunctionNet, 372
FilterActive
CFilterPropertyNet, 146
FilterAttributeEnumNet
Mcs::Usb, 66
FilterBand
CFilterPropertyNet, 147
FilterBandEnumNet
Mcs::Usb, 67
FilterCalculationDirectionEnumNet
Mcs::Usb, 67
FilterFamily
CFilterPropertyNet, 147
FilterFamilyEnumNet
Mcs::Usb, 67
FilterType
CFilterPropertyNet, 147
FilterTypeEnumNet
Mcs::Usb, 67
FindEndpoints
CGenericDevelopDeviceNet, 163
FindFilter
CCreateFilterNet, 122
FindFirmwareVersionMagicInBuffer
CMcsUsbFactoryNet, 307
FindReference
CRoboDeviceNet, 480
FindReferenceI
CRoboStatorDeviceNet, 499
FindReferencePhase0
CRoboDeviceNet::RoboMainLowLevelCommands, 710
FindReferencePhase0XY
CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands, 716
FindReferenceXY
CRoboStatorDeviceNet, 499
FindReferenceZ
CRoboStatorDeviceNet, 499
Finished
Mcs::Usb, 86
FirePressurePulse
CPPCFunctionNet, 431
FirmwareDestinationNames, 697
Altera, 698
Bootstrap, 698
BUS1_MCSBUS1, 698
BUS1_MCSBUS2, 698
DSP, 698
FPGA2, 698
FPGA3, 698
FPGA4, 698
FPGA5, 698
FPGA6, 698
MCSBUS1, 699
MCSBUS10, 699
MCSBUS11, 699
MCSBUS12, 699
MCSBUS13, 699
MCSBUS2, 699
MCSBUS3, 699
MCSBUS4, 699
MCSBUS5, 699
MCSBUS6, 699
MCSBUS7, 699
MCSBUS8, 700
MCSBUS9, 700
MCU1, 700

PIC, 700
PIC2, 700
PIC3, 700
PIC4, 700
USB, 700
FluidControlDevice
CSafeISDeviceNet, 506
ForceStatusEvent
CStg200xDownloadBasicNet, 567
CStimulusFunctionNet, 583
FPGA10
Mcs::Usb, 56
FPGA10_BASE
Mcs::Usb, 56
FPGA10_GOLD
Mcs::Usb, 56
FPGA11
Mcs::Usb, 56
FPGA11_BASE
Mcs::Usb, 56
FPGA11_GOLD
Mcs::Usb, 56
FPGA12
Mcs::Usb, 56
FPGA12_BASE
Mcs::Usb, 56
FPGA12_GOLD
Mcs::Usb, 56
FPGA13
Mcs::Usb, 56
FPGA13_BASE
Mcs::Usb, 56
FPGA13_GOLD
Mcs::Usb, 56
FPGA14
Mcs::Usb, 56
FPGA14_BASE
Mcs::Usb, 56
FPGA14_GOLD
Mcs::Usb, 56
FPGA15
Mcs::Usb, 56
FPGA15_BASE
Mcs::Usb, 57
FPGA15_GOLD
Mcs::Usb, 56
FPGA16
Mcs::Usb, 56
FPGA16_BASE
Mcs::Usb, 57
FPGA16_GOLD
Mcs::Usb, 56
FPGA2
FirmwareDestinationNames, 698
Mcs::Usb, 56
FPGA2_BASE
Mcs::Usb, 56
FPGA2_GOLD
Mcs::Usb, 56
FPGA3
FirmwareDestinationNames, 698
Mcs::Usb, 56
FPGA3_BASE
Mcs::Usb, 56
FPGA3_GOLD
Mcs::Usb, 56
FPGA4
FirmwareDestinationNames, 698
Mcs::Usb, 56
FPGA4_BASE
Mcs::Usb, 56
FPGA4_GOLD
Mcs::Usb, 56
FPGA5
FirmwareDestinationNames, 698
Mcs::Usb, 56
FPGA5_BASE
Mcs::Usb, 56
FPGA5_GOLD
Mcs::Usb, 56
FPGA6
FirmwareDestinationNames, 698
Mcs::Usb, 56
FPGA6_BASE
Mcs::Usb, 56
FPGA6_GOLD
Mcs::Usb, 56
FPGA7
Mcs::Usb, 56
FPGA7_BASE
Mcs::Usb, 56
FPGA7_GOLD
Mcs::Usb, 56
FPGA8
Mcs::Usb, 56
FPGA8_BASE
Mcs::Usb, 56
FPGA8_GOLD
Mcs::Usb, 56
FPGA9
Mcs::Usb, 56
FPGA9_BASE
Mcs::Usb, 56
FPGA9_GOLD
Mcs::Usb, 56
FPGA_BASE
Mcs::Usb, 56
FPGA_BOOTSTRAP
Mcs::Usb, 57
FPGA_GOLD
Mcs::Usb, 56
FPGA_NORMAL
Mcs::Usb, 54
FpgaIdEnumNet
Mcs::Usb, 68
FrameContextGroup

Mcs::Usb, 62
FromIntPtr
StgStatusNet, 719
FromPtr
StgStatusNet, 719
FullCharge
Mcs::Usb, 71
FullSpeed
Mcs::Usb, 72
FunkDongleS
Mcs::Usb, 79
FX3MCSDataAddress
CMcsUsbFactoryNet, 312
FX3MCSDataDeviceIdOffset
CMcsUsbFactoryNet, 312
FX3MCSDataIFB1ImageOffset
CMcsUsbFactoryNet, 312
FX3MCSDataIFB2ImageOffset
CMcsUsbFactoryNet, 312
FX3MCSDataVersionOffset
CMcsUsbFactoryNet, 312
FYIProgram
CFYIDeviceNet, 155
FYITemp
CFYIDeviceNet, 155

Gain
CMeaDeviceNet, 365
Gain100
Mcs::Usb, 87
Gain1000
Mcs::Usb, 87
Gain200
Mcs::Usb, 87
Gain500
Mcs::Usb, 87
Gate
Mcs::Usb, 82
Gated_High_Active
Mcs::Usb, 81
Gated_Low_Active
Mcs::Usb, 81
GE2100
Mcs::Usb, 80
Generic
Mcs::Usb, 79
Get2AnalogInput
CMcsBus_SensorNet, 235
Get2DigitalInput
CMcsBus_SensorNet, 235
Get4ADC
CMcsBus_SensorNet, 235
Get4ADCAverage
CMcsBus_SensorNet, 235
Get4ADCCatchampAverageShift
CMcsBus_SensorNet, 235
Get4ADCMode
CMcsBus_SensorNet, 235
Get4DAC
CMcsBus_SensorNet, 235
GetAbsMaxCurrentInMicroAmp
CMultiwellOptoStimFunctionNet, 397
GetAccelGyroCurrentRate
CW2100_FunctionNet, 624
GetAccelGyroDesiredRate
CW2100_FunctionNet, 624
GetAccelGyroEnabled
CW2100_FunctionNet, 624
GetAccelRange
CW2100_FunctionNet, 624
GetActiveRunningTableNumber
CWarnerValveControllerDeviceNet, 658
GetActiveRunningTableNumberEvent
CWarnerValveControllerDeviceNet, 675
GetAdapterCode
CMeaImpedanceDeviceNet, 373
CTEERFunctionNet, 609
GetAdapterType
CMcsUsbDacqNet, 274
GetAdc
CFluidControlDeviceNet, 150
GetAdcDataFormat
CMcsUsbDacqNet, 275
GetADCGain
CGrapheneFunctionNet, 179
GetADCInputOffset
CCMOSMea_FunctionNet, 109
GetADCOffset
CGrapheneFunctionNet, 179, 180
GetAdcOffset
CLIH3DeviceNet, 203
COctoPotDeviceNet, 403
GetAdcOffsetU1
CTEERFunctionNet, 609
GetAdcOffsetU2
CTEERFunctionNet, 609
GetADCs
CMcsBus_SensorNet, 235
GetADCsLoop
CMcsBus_SensorNet, 236
GetAdcZero
CMcsUsbDacqNet, 275
GetAirpressure
CRoboDeviceNet, 480
GetAirpressureLimit
CRoboDeviceNet, 480
GetAirValve
CRoboDeviceNet, 480
GetAllDigout
CRoboDacqNet, 466
GetAmplification
CPgaDeviceNet, 415
GetAmplitude_nA
CTEERFunctionNet, 609
GetAnalogGain
CMeaDeviceNet, 362
GetAnalogOutADCRange

CSCUFunctionNet, 511
GetAnalogOutChannel
CW2100_FunctionNet, 624
GetAnalogOutChannels
CSCUFunctionNet, 511
GetAnalogOutDACRange
CSCUFunctionNet, 511
GetAnalogOutFilter
CW2100_FunctionNet, 624
GetAnalogThresholdHigh
CWarnerValveControllerDeviceNet, 658
GetAnalogThresholdHighEvent
CWarnerValveControllerDeviceNet, 675
GetAnalogThresholdLow
CWarnerValveControllerDeviceNet, 659
GetAnalogThresholdLowEvent
CWarnerValveControllerDeviceNet, 675
GetAnalogValueUnit
CMcsUsbDacqNet, 275
GetAnalogVoltage
CPPCFunctionNet, 431
CPPS_FunctionNet, 440
CWarnerValveControllerDeviceNet, 659
GetAnalogVoltageEvent
CWarnerValveControllerDeviceNet, 675
GetAnalogVoltageRange
CPPCFunctionNet, 431
GetAnalogVoltages
CPPS_FunctionNet, 440
GetArraySize
CMeaImpedanceDeviceNet, 373
GetAudioChannels
CMeaAudioFunctionNet, 347
CW2100_FunctionNet, 624
GetAudioOutDacParameter
CLIH3DeviceNet, 203
GetAutocalibrationDisabled
CStg200xBasicNet, 530
GetAvailableBaseSamplerates
CCMOSMeaDeviceNet, 118
CGrapheneASICDeviceNet, 175
GetAvailableChambers
CWarnerUssingFunctionNet, 641
GetAvailableDeviceList
CRFFunctionNet, 455
GetAvailableDeviceListEx
CRFFunctionNet, 455
GetAvailableHeadstages
CSCUFunctionNet, 511
CW2100_FunctionNet, 624
GetAvailableHeadstagesEvent
CSCUFunctionNet, 524
GetAvailableMemory
CStg200xBasicNet, 531
CStimulusFunctionNet, 583
GetAvailableSampleRates
CMcsUsbDacqNet::CHWInfo, 192
GetAvailableStateList
CRFFunctionNet, 455
GetAvailableStateListEx
CRFFunctionNet, 455
GetAvailableVoltageRangesInMicroVolt
CMcsUsbDacqNet::CHWInfo, 192
GetAvailableVoltageRangesInMicroVoltAndStringsInMilliVolt
CMcsUsbDacqNet::CHWInfo, 193
GetAxisConfig
CRoboDeviceNet::RoboMainLowLevelCommands, 710
GetAxisLED
CRoboocyte2DeviceNet, 497
GetAxisParametersSignedEeprom
CMcsBus_AxisParametersNet, 210
GetAxisParametersUnsignedEeprom
CMcsBus_AxisParametersNet, 210
GetBaseFrequency
CRFFunctionNet, 456
GetBaseSamplerate
CCMOSMeaDeviceNet, 118
CGrapheneASICDeviceNet, 176
GetBath
CCMOSMea_FunctionNet, 109
GetBathMode
CCMOSMea_FunctionNet, 109
GetBatteryState
CW2100_FunctionNet, 625
GetBatteryVoltage
CMultiBatteryChargerDeviceNet, 382
GetBiQuad
CCreateFilterNet, 122
GetBiQuads
CCreateFilterNet, 122
GetBlankingEnable
CStg200xBasicNet, 531
GetBoardTemp
CTcxDeviceNet, 596
GetBoostAlwaysOnMode
CW2100_StimulatorFunctionNet, 631
GetBoostPreTime
CW2100_StimulatorFunctionNet, 631
GetBubbleState
CPPS_FunctionNet, 440
GetBubbleStatus
CMcsBus_SensorNet, 236
GetBuffer
CGenericDevelopDeviceNet, 163
GetBusAddress
CMcsBusNet, 249
GetBusAddressEeprom
CMcsBusNet, 249
GetByteBuffer
CGenericDevelopDeviceNet, 163
GetBytesAvailable
CSerialPortNet, 525
GetBytesPerSample
CTEERFunctionNet, 610
GetCalibration

CTcxDeviceNet, 596
GetCalibrationDecp
CTcxDeviceNet, 596
GetCalibrationMax
CTcxDeviceNet, 596
GetCalibrationMin
CTcxDeviceNet, 596
GetCanDisableCalibration
CStg200xBasicNet, 532
GetCanDisableCurrentswitch
CStg200xBasicNet, 532
GetCapacityC
CRoboDacqNet, 466
GetCapacityV
CRoboDacqNet, 466
GetCapacityX
CRoboDacqNet, 466
GetCardinalDacqSamplerate
CInterfaceboardFunctionNet, 199
GetCardinalStgOutputrate
CInterfaceboardFunctionNet, 199
GetChannel
CSw2to64DeviceNet, 592
GetChannelDataFillSize
CMcsUsbDacqNet, 275
GetChannelDataI16
CCMOSMeaDeviceNet, 118
GetChannelDataI32
CCMOSMeaDeviceNet, 118
GetChannelDataUI16
CCMOSMeaDeviceNet, 119
GetChannelDataUI32
CCMOSMeaDeviceNet, 119
GetChannelLayout
CMcsUsbDacqNet, 275
GetChannels
CMultiBatteryChargerDeviceNet, 382
CSw2to64DeviceNet, 592
GetChannelsCountOfChamber
CWarnerUssingFunctionNet, 641
GetChannelsInBlock
CMcsUsbDacqNet, 275
GetChannelState
CMultiBatteryChargerDeviceNet, 382
GetChargeCapacity
CMultiBatteryChargerDeviceNet, 382
GetChargeCurrent
CMultiBatteryChargerDeviceNet, 383
GetChargingMode
CMultiBatteryChargerDeviceNet, 383
GetChargingPCoefficient
CMultiBatteryChargerDeviceNet, 383
GetChecksumFromFX3Image
CMcsUsbFactoryNet, 307
GetCheckVoltage
COkuvisionStimulatorDeviceNet, 407
GetClampAmpSerialNumber
CRoboDacqNet, 466
GetClampMode
CTEERFunctionNet, 610
CWarnerUssingFunctionNet, 641
GetCMOSDataDictionary
CCMOSMeaDeviceNet, 119
GetCoilCommunication
CPositionIIDeviceNet, 418
GetColorRgb
CMultiwellOptoStimFunctionNet, 398
GetColorStr
CMultiwellOptoStimFunctionNet, 398
GetCommand
CMcsBusNet, 249, 250
CPedoterDeviceNet, 412
CRoboDacqNet, 466
GetComplianceVoltageThreshold
CWarnerUssingFunctionNet, 641
GetConfiguration
CMcsUsbNet, 330
GetConfigurationBit
CRoboDacqNet, 466
GetConfigurationBitAxc
CRoboDacqNet, 466
GetConfigurationBitBlu_Led
CRoboDacqNet, 466
GetConfigurationBitBlu_LedToggleFast
CRoboDacqNet, 466
GetConfigurationBitBlu_LedToggleSlow
CRoboDacqNet, 466
GetConfigurationBitCC_Gen
CRoboDacqNet, 467
GetConfigurationBitCV_Gen
CRoboDacqNet, 467
GetConfigurationBitRC_Gen
CRoboDacqNet, 467
GetConfigurationBitRed_Led
CRoboDacqNet, 467
GetConfigurationBitRed_LedSaturation
CRoboDacqNet, 467
GetConfigurationBitRed_LedToggleFast
CRoboDacqNet, 467
GetConfigurationBitRed_LedToggleSlow
CRoboDacqNet, 467
GetConfigurationBitRelais
CRoboDacqNet, 467
GetConfigurationBitRV_Gen
CRoboDacqNet, 467
GetConfigurationBits
CRoboDacqNet, 467
GetConfigurationBitStream
CRoboDacqNet, 467
GetConfigurationBitSupply
CRoboDacqNet, 468
GetConnectedDevice
CRFFunctionNet, 456
GetControllerParams
CTEERFunctionNet, 610
GetCpp

CCreateFilterNet, 122
GetCrossTalkOffset
CRoboDacqNet, 468
GetCrossTalkOptimum
CRoboDacqNet, 468
GetCur2VolResistance
CGrapheneFunctionNet, 180
GetCurrent
CTcxDeviceNet, 596
GetCurrentAirvalve
CRoboDeviceNet, 480
GetCurrentAirvalveLimit
CRoboDeviceNet, 480
GetCurrentCycle
CMeaCoatDeviceNet, 354
GetCurrentEditTableNumber
CWarnerValveControllerDeviceNet, 659
GetCurrentEnable
CTEERFunctionNet, 610
GetCurrentFactor
COkuvisionStimulatorDeviceNet, 407
GetCurrentNumberOfValves
CWarnerValveControllerDeviceNet, 659
GetCurrentNumberOfValvesEvent
CWarnerValveControllerDeviceNet, 676
GetCurrentPosition
CRoboDeviceNet, 481
GetCurrentPositionI
CRoboStatorDeviceNet, 499
GetCurrentPositionXY
CRoboStatorDeviceNet, 499
GetCurrentPositionZ
CRoboStatorDeviceNet, 500
GetCurrentRangeInNanoAmp
CStg200xBasicNet, 532
CStimulusFunctionNet, 584
CW2100_StimulatorFunctionNet, 631
GetCurrentRangeInNanoAmpByIndex
CStg200xBasicNet, 533
GetCurrentRangeListInNanoAmp
CStg200xBasicNet, 533
GetCurrentRangeSelectedIndex
CStg200xBasicNet, 533
GetCurrentResolutionInNanoAmp
CStg200xBasicNet, 533
CStimulusFunctionNet, 584
CW2100_StimulatorFunctionNet, 631
GetCurrentResolutionInPicoAmp
CStg200xBasicNet, 534
CStimulusFunctionNet, 584
CW2100_StimulatorFunctionNet, 631
GetCycle
CMeaCleanDeviceNet, 350
GetCycles
CMeaCleanDeviceNet, 350
CMeaCoatDeviceNet, 354
GetD
CTcxDeviceNet, 597
GetDacAmplificationFactor
CStg200xBasicNet, 534
GetDacIdleValue
CLIH3DeviceNet, 204
GetDACOffset
CGrapheneFunctionNet, 181
COkuvisionStimulatorDeviceNet, 407
GetDacOffset
CDacCalibrationFunctionNet, 124
CLIH3DeviceNet, 204
COctoPotDeviceNet, 403
GetDacPampsPerDigitHighCurrentRange
CWarnerUssingFunctionNet, 642
GetDacPampsPerDigitLowCurrentRange
CWarnerUssingFunctionNet, 642
GetDacqRunStatus
CLIH3DeviceNet, 204
GetDacRange
CW2100_FunctionNet, 625
GetDACResolution
CStg200xBasicNet, 534
CStimulusFunctionNet, 585
CW2100_StimulatorFunctionNet, 632
GetDACs
CMcsBus_SensorNet, 236
GetDacUseIdleValue
CLIH3DeviceNet, 204
GetDacZero
CTEERFunctionNet, 611
CWarnerUssingFunctionNet, 642
GetDataFormat
CMcsUsbDacqNet, 276
GetDataMode
CMcsUsbDacqNet, 276
GetDDecp
CTcxDeviceNet, 597
GetDebugData
CPositionIIDeviceNet, 418
GetDefaultFilterAttributes
CFilterConfigurationNet, 142
GetDestination
CMcsUsbFactoryNet, 307
GetDestinationCode
DriverVersionNet, 693
GetDestinationDisplayLabel
CMcsUsbFactoryNet, 307
GetDestinationName
CMcsUsbFactoryNet, 307, 308
DriverVersionNet, 694
GetDestinationSerialNumber
CMcsUsbFactoryNet, 308
GetDestinationTargetAddress
CMcsUsbFactoryNet, 308
GetDetectionThreshold
CMcsBus_SensorNet, 236
GetDetectorValue
CMcsBus_SensorNet, 236
GetDevice

CTcxDeviceNet, 597
GetDeviceCannotStallOutRequests
CMcsUsbNet, 330
GetDeviceCapableSpeed
CMcsUsbNet, 330
GetDeviceEnum
CMcsUsbNet, 330
GetDeviceGroupChannelInfos
CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet >, 127
GetDeviceId
CMcsUsbNet, 330
GetDeviceList
CPositionImpDeviceNet, 426
GetDeviceNames
CRadioControledDevicesNet, 449
GetDeviceRootHubVendorEnum
CMcsUsbNet, 330
GetDeviceRootHubVendorID
CMcsUsbNet, 330
GetDeviceRootHubVendorName
CMcsUsbNet, 330
GetDeviceSpeed
CMcsUsbNet, 331
GetDeviceType
CTcxDeviceNet, 597
GetDevname
CTcxDeviceNet, 597
GetDigin
CFluidControlDeviceNet, 150
GetDigInState
CLIH3DeviceNet, 205
GetDiginValue
CStg200xBasicNet, 534
GetDigitalData
CMeaDigitalDataFunctionNet, 367
GetDigitalIn
CPPCFunctionNet, 433
CPPS_FunctionNet, 440
GetDigitalOutPortValve
CWarnerValveControllerDeviceNet, 660
GetDigitalOutPortValveEvent
CWarnerValveControllerDeviceNet, 676
GetDigitalPortDirection
CWarnerValveControllerDeviceNet, 660
GetDigitalPortDirectionEvent
CWarnerValveControllerDeviceNet, 676
GetDigitalSource
CMcsUsbDacqNet, 276–278
GetDigitalStimulatorTrigger
CW2100_StimulatorFunctionNet, 632
GetDigitalStimulatorTriggerSlope
CW2100_StimulatorFunctionNet, 632
GetDigout
CFluidControlDeviceNet, 150
CRoboDacqNet, 468
GetDigoutMode
CStg200xBasicNet, 535
GetDigoutValue
CStg200xBasicNet, 535
GetDIO
CMcsBus_FYIExtensionNet, 213
GetDischargeCapacity
CMultiBatteryChargerDeviceNet, 383
GetDischargeCurrent
CMultiBatteryChargerDeviceNet, 384
GetDischargeCurrentSetPoint
CMultiBatteryChargerDeviceNet, 384
GetDisplayMode
CWarnerValveControllerDeviceNet, 660
GetDisplayModeEvent
CWarnerValveControllerDeviceNet, 676
GetDisplayText
CRoboDacqNet, 468
GetDMax
CTcxDeviceNet, 597
GetDMin
CTcxDeviceNet, 597
GetDownsampleFactor
CRoboDacqNet, 468
GetDSPHighPassByIndex
CIntanMea_FunctionNet, 195
GetDuration
CMeaCoatDeviceNet, 355
GetDuty
CTcxDeviceNet, 597
GetEEpromPage
CLIH3DeviceNet, 205
GetElectrodeDacMux
CStg200xBasicNet, 535
GetElectrodeEnable
CStg200xBasicNet, 536
GetElectrodeMode
CStg200xBasicNet, 537
GetEnableAmplifierProtectionSwitch
CStg200xBasicNet, 538
GetEnabledChannelsInGroup
CCMOSMea_FunctionNet, 109
CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet >, 127
GetEnableHeaterLimit
CTcxDeviceNet, 598
GetEnableThermocouple
CTcxDeviceNet, 598
GetEntry
CMcsUsbListEntryNet, 317
GetEntryCount
CMcsUsbListEntryNet, 318
GetEnumerationSpeed
CMeaDeviceNet, 362
GetErrorAirpressure
CRoboDeviceNet, 481

GetErrorCurrentAirvalve
CRoboDeviceNet, 481
GetErrorMessage
CMcsUsbDacqNet, 279
GetErrorText
CMcsUsbNet, 331
GetErrorVoltage12V
CRoboDeviceNet, 481
GetErrorVoltage5V
CRoboDeviceNet, 481
GetErrorVoltageAirvalve
CRoboDeviceNet, 481
GetErrorVoltageRs485A
CRoboDeviceNet, 481
GetErrorVoltageRs485B
CRoboDeviceNet, 481
GetErrorVoltageValves
CRoboDeviceNet, 482
GetEventData
CPositionIIDeviceNet, 419
GetExternalElectrodeEnable
CStg200xBasicNet, 539
GetFAAmplification
CStg200xBasicNet, 539
GetFilter
CRoboDacqNet, 468
GetFilterAttributes
CFilterConfigurationNet, 142, 143
GetFilterCoeffs
CRoboDacqNet, 468
GetFilterParametersHeadstage
CWClassicFunctionNet, 683
GetFilterProperties
CSCUFunctionNet, 511
CW2100_FunctionNet, 625
GetFilterProperty
CMcsUsbDacqNet, 279
CSCUFunctionNet, 512
CW2100_FunctionNet, 625
GetFinalDischargeVoltage
CMultiBatteryChargerDeviceNet, 384
GetFirmwareVersion
CMcsUsbNet, 331
GetFirmwareVersionFromFile
CMcsUsbFactoryNet, 308
GetFirmwareVersionFromHexFile
CMcsUsbFactoryNet, 308
GetFPGAFirmwareType
CW2100_FunctionNet, 625
GetFrameErrorCounter
CTEERFunctionNet, 611
GetFrequency
CRadioControledDevicesNet, 450
GetFrequencyRange
CPgaDeviceNet, 415
GetGain
CMeaDeviceNet, 362
CPgaDeviceNet, 416
GetGate
CCMOSMea_FunctionNet, 109
GetGilsonDevice
CRoboocyte2DeviceNet, 497
GetGlobalRepeat
CDigOutStimulatorFunctionNet, 134
GetGNDI
CCMOSMea_FunctionNet, 109
GetGroupADCBits
CCMOSMea_FunctionNet, 109
GetGroupChannelBitmaskBySelect
CCMOSMea_FunctionNet, 110
GetGroupChannelBitmaskHS1NCBathCurrent
CCMOSMea_FunctionNet, 110
GetGroupChannelBitmaskHS1NCCol2Current
CCMOSMea_FunctionNet, 110
GetGroupChannelBitmaskHS1NChipTemp
CCMOSMea_FunctionNet, 110
GetGroupChannelBitmaskHS1Sidebands
CCMOSMea_FunctionNet, 111
GetGroupChannelBitmaskHS1TriggerStatus
CCMOSMea_FunctionNet, 111
GetGroupChannelBitmaskIFDigChannels
CCMOSMea_FunctionNet, 111
GetGroupChannelBitmaskInterfaceADC
CCMOSMea_FunctionNet, 111
GetGroupChannelBitmaskPacketFrameContext
CCMOSMea_FunctionNet, 112
GetGroupChannelBitmaskSTG1DACSignal
CCMOSMea_FunctionNet, 112
GetGroupChannelDataI16
CMcsUsbDacqNet, 279
GetGroupChannelDataI32
CMcsUsbDacqNet, 280
GetGroupChannelDataUI16
CMcsUsbDacqNet, 280
GetGroupChannelDataUI32
CMcsUsbDacqNet, 281
GetGroupDCOffset
CCMOSMea_FunctionNet, 112
GetGroupID
CCMOSMea_FunctionNet, 112
CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet >, 127
GetGroupNumberOfChannels
CCMOSMea_FunctionNet, 113
CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet >, 128
GetGroupResolutionPerDigit
CCMOSMea_FunctionNet, 113
GetGroupSampleSize
CCMOSMea_FunctionNet, 113
CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, Dac-

qGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet >, 128
GetGroupType
CCMOSMea_FunctionNet, 113
CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet >, 128
GetGroupUnit
CCMOSMea_FunctionNet, 113, 114
GetGyroRange
CW2100_FunctionNet, 625
GetHardwareMaxRange
CMcsUsbDacqNet, 281
GetHardwareMinRange
CMcsUsbDacqNet, 281
GetHardwareRevision
CMcsUsbNet, 331
GetHasChecksum
CWClassicFunctionNet, 683
GetHashCode
HeadstageIDTypeObject, 704
GetHasLowCurrentswitchLatency
CStg200xBasicNet, 539
GetHasRedLedHeadstage
CWClassicFunctionNet, 683
GetHasThermocouple
CTcxDeviceNet, 598
GetHeadstage
CStg200xBasicNet, 539
GetHeadstageActive
CMcsUsbNet, 332
GetHeadstageAdcBits
CSCUFunctionNet, 512
GetHeadstageAdcRangeInMicroVolt
CSCUFunctionNet, 512
GetHeadstageDacBits
CSCUFunctionNet, 513
GetHeadstageDacCurrentRangeInMicroAmpere
CSCUFunctionNet, 513
GetHeadstageDacCurrentResolutionInNanoAmpere
CSCUFunctionNet, 513
GetHeadstageDacVoltageRangeInMilliVolt
CSCUFunctionNet, 514
GetHeadstageDacVoltageResolutionInMicroVolt
CSCUFunctionNet, 514
GetHeadstageFrameCyclesToCompare
CSCUFunctionNet, 514
GetHeadstageGainInPermille
CSCUFunctionNet, 515
GetHeadstageID
CMcsUsbNet, 332
CSCUFunctionNet, 515
GetHeadstageLinkSpeed
CSCUFunctionNet, 515
GetHeadstageNumberOfAnalogChannels
CSCUFunctionNet, 516
GetHeadstageNumberOfStimulationChannels
CSCUFunctionNet, 516
GetHeadstageOnOff
CW2100_FunctionNet, 625
CWClassicFunctionNet, 683
GetHeadstagePowerStateAtStart
CSCUFunctionNet, 516
GetHeadstagePresent
CMcsUsbNet, 332
GetHeadstageSamplerate
CSCUFunctionNet, 517
GetHeadstageSamplingActive
CW2100_FunctionNet, 625
GetHeadstageSerialNumber
CSCUFunctionNet, 517
GetHeaterLimit
CTcxDeviceNet, 598
GetHeaterTemp
CTcxDeviceNet, 598
GetHighCurrentRange
CWarnerUssingFunctionNet, 643
GetHighpassFilterEnable
CFilterConfigurationNet, 143
GetHWConfig
CRoboDeviceNet::RoboMainLowLevelCommands, 710
GetHWRevision
CRoboDeviceNet::RoboMainLowLevelCommands, 710
GetHWRevisionEeprom
CMcsBusNet, 250
GetI
CTcxDeviceNet, 598
GetIC
CRoboDacqNet, 468
GetIClamp
CRoboDacqNet, 468
GetICoeff
CRobo_FYITemp_FunctionNet, 460
GetICOffset
CRoboDacqNet, 468
GetIDecp
CTcxDeviceNet, 598
GetIdleModeOffset
CWarnerUssingFunctionNet, 643
GetIGain
CRoboDacqNet, 469
GetIMax
CTcxDeviceNet, 599
GetIMin
CTcxDeviceNet, 599
GetImpedanceResult
CIntanMea_FunctionNet, 195
GetImpedanceTestFrequency
CMeaImpedanceDeviceNet, 374
GetImpId
CPositionImpDeviceNet, 427
GetImplantCurrentSetpoint
CPositionIIDeviceNet, 419

GetImplantResult
CPositionIIDeviceNet, 420
GetImplantState
CPositionIIDeviceNet, 420
GetInMovement
CRoboDeviceNet, 482
GetIntanRegister
CIntanMea_FunctionNet, 195
GetIntBuffer
CGenericDevelopDeviceNet, 163
GetIO
CWarnerValveControllerDeviceTesterFunctionNet, 680
GetIOut
CTcxDeviceNet, 599
GetIoVoltage
CInterfaceboard2FunctionNet, 197
GetLastAnswer
CGilsonDeviceNet, 174
GetLastUSBError
CMcsUsbNet, 332
GetLatency
CMcsBus_SensorNet, 236
GetLatencyCounter
CMcsBus_SensorNet, 236
GetLayoutConfiguration
CMEA2100x256FunctionNet, 345
GetLEDSwitch
CMcsBus_ExtensionNet, 212
GetLength
CRobo_FYIProgram_FunctionNet, 458
GetLiquidResistance
CTEERFunctionNet, 611
CWarnerUssingFunctionNet, 643
GetListmodeIndexRange
CStg200xBasicNet, 540
GetListmodeTriggerSource
CStg200xBasicNet, 540
GetLowCurrentRange
CWarnerUssingFunctionNet, 644
GetLowerFrequencyByIndex
CIntanMea_FunctionNet, 195
GetMajor
DriverVersionNet, 694
GetMaxChunkSize_Byte
CTEERFunctionNet, 611
GetMaxCurrent
CMeaCoatDeviceNet, 355
GetMaxDurationHighCurrentInMicroSec
CMultiwellOptoStimFunctionNet, 398
GetMaxDutyCycleHighCurrent
CMultiwellOptoStimFunctionNet, 398
GetMaxHeaterPowerMultiwell
CTcxDeviceNet, 599
GetMaxNoPressure
CRoboDeviceNet::RoboMainLowLevelCommands, 710
GetMaxNoPressureWaitTime
CRoboDeviceNet::RoboMainLowLevelCommands, 710
GetMaxNumberOfHeadstages
CSCUFunctionNet, 517
GetMaxNumOfColumns
CCMOSMea_FunctionNet, 114
GetMaxP
CTcxDeviceNet, 599
GetMaxpDecp
CTcxDeviceNet, 599
GetMaxpMax
CTcxDeviceNet, 599
GetMaxpMin
CTcxDeviceNet, 600
GetMaxPower
COkuvisionStimulatorDeviceNet, 407
CRobo_FYITemp_FunctionNet, 460
GetMaxPressureWaitTime
CRoboDeviceNet::RoboMainLowLevelCommands, 710
GetMaxReadableColumns
CCMOSMeaDeviceNet, 119
GetMaxSamplingFrequency
CMcsUsbDacqNet, 282
GetMaxStimulusChannelsPerHeadstage
CSCUFunctionNet, 517
GetMaxVoltage
CMeaCleanDeviceNet, 350
COkuvisionStimulatorDeviceNet, 407
GetMCAcceleration
CMcsBus_MotorControlNet, 217
GetMCAccelerationEeprom
CMcsBus_MotorControlNet, 217
GetMCAccelerationShortCommand
CMcsBus_MotorControlNet, 217
GetMCAxisRevisionEeprom
CMcsBus_MotorControlNet, 217
GetMCBreakCurrent
CMcsBus_MotorControlNet, 217
GetMCBreakCurrentEeprom
CMcsBus_MotorControlNet, 218
GetMCConfig
CMcsBus_MotorControlNet, 218
GetMCConfigEeprom
CMcsBus_MotorControlNet, 218
GetMCCurrent
CMcsBus_MotorControlNet, 218
GetMCCurrentEeprom
CMcsBus_MotorControlNet, 218
GetMCCurrentMode
CMcsBus_MotorControlNet, 218
GetMCCurrentModeEeprom
CMcsBus_MotorControlNet, 218
GetMCCurrentModeShortCommand
CMcsBus_MotorControlNet, 219
GetMCCurrentPosition
CMcsBus_MotorControlNet, 219
GetMCCurrentShortCommand

CMcsBus_MotorControlNet, 219
GetMCCurrentSpeed
CMcsBus_MotorControlNet, 219
GetMCMaxAcceleration
CMcsBus_MotorControlNet, 219
GetMCMaxAccelerationEeprom
CMcsBus_MotorControlNet, 219
GetMCMaxCurrent
CMcsBus_MotorControlNet, 219
GetMCMaxCurrentEeprom
CMcsBus_MotorControlNet, 220
GetMCMaxSpeed
CMcsBus_MotorControlNet, 220
GetMCMaxSpeedEeprom
CMcsBus_MotorControlNet, 220
GetMCMaxTravel
CMcsBus_MotorControlNet, 220
GetMCMaxTravelEeprom
CMcsBus_MotorControlNet, 220
GetMCMaxTravelShortCommand
CMcsBus_MotorControlNet, 220
GetMCMovement
CMcsBus_MotorControlNet, 220
GetMCNewPosition
CMcsBus_MotorControlNet, 221
GetMCOutputOnOff
CMcsBus_MotorControlNet, 221
GetMCPhase
CMcsBus_MotorControlNet, 221
GetMCPhaseOffset
CMcsBus_MotorControlNet, 221
GetMCReference
CMcsBus_MotorControlNet, 221
GetMCReferenceCurrent
CMcsBus_MotorControlNet, 221
GetMCReferenceCurrentEeprom
CMcsBus_MotorControlNet, 221
GetMCRegulatorGain
CMcsBus_MotorControlNet, 222
GetMCRegulatorGainEeprom
CMcsBus_MotorControlNet, 222
GetMcsBus_Extension
CRoboocyte2DeviceNet, 497
GetMCScalingFactor
CMcsBus_MotorControlNet, 222
GetMCScalingFactorEeprom
CMcsBus_MotorControlNet, 222
GetMCSpeed
CMcsBus_MotorControlNet, 222
GetMCSpeedEeprom
CMcsBus_MotorControlNet, 222
GetMCSpeedShortCommand
CMcsBus_MotorControlNet, 222
GetMCSpeedUnitEeprom
CMcsBus_MotorControlNet, 223
GetMCStandbyCurrent
CMcsBus_MotorControlNet, 223
GetMCStandbyCurrentEeprom
CMcsBus_MotorControlNet, 223
GetMCStandbyTime
CMcsBus_MotorControlNet, 223
GetMCStandbyTimeEeprom
CMcsBus_MotorControlNet, 223
GetMea21UsbPort
CMcsUsbNet, 333
GetMeaLayout
CMcsUsbDacqNet, 282
GetMemoryUsageDAC
CStg200xDownloadBasicNet, 567
GetMemoryUsageSyncout
CStg200xDownloadBasicNet, 567
GetMinimalThreshold
CMcsBus_SensorNet, 236
GetMinNoPressureWaitTime
CRoboDeviceNet::RoboMainLowLevelCommands, 710
GetMinor
DriverVersionNet, 695
GetMinPressure
CRoboDeviceNet, 482
CRoboDeviceNet::RoboMainLowLevelCommands, 710
GetMinPressureWaitTime
CRoboDeviceNet::RoboMainLowLevelCommands, 711
GetMinSamplingFrequencyStepsize
CMcsUsbDacqNet, 282
GetMinVoltage
CMeaCleanDeviceNet, 351
GetModeSelect
CPulseGeneratorFunctionNet, 447
GetModuleCurrent
CStg200xDownloadNet, 576
GetModuleTemp
CStg200xDownloadNet, 576
GetMovementError
CRoboDeviceNet, 482
GetMovePump
CMcsBus_SensorNet, 237
GetMultiHeadstageMode
CW2100_FunctionNet, 625
GetMultiplexedDataChannelsInBlock
CStimulusFunctionNet, 585
GetNanoVoltsPerKelvin
CMcsBus_TempSensorNet, 242
GetNeurochipMemoryData
CCMOSMea_FunctionNet, 114
GetNeurochipMemorySize
CCMOSMea_FunctionNet, 114
GetNIC_MS
CRoboDacqNet, 469
GetNUC_MS
CRoboDacqNet, 469
GetNumAmplifications
CPgaDeviceNet, 416
GetNumber

CMeaSwitchDeviceNet, 376
CSw2to64DeviceNet, 592
GetNumberOfAnalogChannels
CStg200xBasicNet, 540
CStimulusFunctionNet, 585
CW2100_StimulatorFunctionNet, 632
GetNumberOfAudioChannels
CMeaAudioFunctionNet, 347
GetNumberOfAvailableChambers
CWarnerUssingFunctionNet, 644
GetNumberOfAvailableSamples
CTEERFunctionNet, 611
GetNumberOfChannels
CDigOutStimulatorFunctionNet, 134
GetNumberOfCurrentRangeIndexes
CStg200xBasicNet, 541
GetNumberOfDataBits
CMcsUsbDacqNet, 282
GetNumberOfDevices
CMcsUsbListNet, 321
GetNumberOfHardwareSlotsForChambers
CWarnerUssingFunctionNet, 644
GetNumberOfHWADCChannels
CMcsUsbDacqNet::CHWInfo, 193
GetNumberOfHWDACPaths
CStg200xBasicNet, 541
GetNumberOfHWDigitalChannels
CMcsUsbDacqNet::CHWInfo, 193
GetNumberOfStimulationElectrodes
CStg200xBasicNet, 541
GetNumberOfStimulationSourcesPerElectrode
CStg200xBasicNet, 541
GetNumberOfSupportedGroups
CCMOSMea_FunctionNet, 114
CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet >, 128
GetNumberOfSyncoutChannels
CStg200xBasicNet, 541
CW2100_StimulatorFunctionNet, 632
GetNumberOfTriggerInputs
CStg200xBasicNet, 541
CW2100_StimulatorFunctionNet, 632
GetNumberOfVoltageRangeIndexes
CStg200xBasicNet, 542
GetNumConfigurations
CMcsUsbNet, 333
GetNumControlChannels
CTcxDeviceNet, 600
GetNumDestinations
CMcsUsbFactoryNet, 308
GetNumDevices
CTcxDeviceNet, 600
GetNumEntries
DriverVersionNet, 695
GetNumFrequencyRanges
CPgaDeviceNet, 416
GetNumMeasureChannels
CTcxDeviceNet, 600
GetNUV_MS
CRoboDacqNet, 469
GetOffsetCurrent
CMeaCoatDeviceNet, 355
GetOnOff
CPositionIIDeviceNet, 420
CTcxDeviceNet, 600
GetOutputCurrent
CMeaCoatDeviceNet, 355
GetOutputRate
CStg200xBasicNet, 542
GetOutputVoltage
CMeaCleanDeviceNet, 351
GetP
CTcxDeviceNet, 600
GetParameter
CRoboDeviceNet::RoboMainLowLevelCommands, 711
GetPattern
CMeaSwitchDeviceNet, 376
GetPatternBool
CMeaSwitchDeviceNet, 376
GetPauseDuration
CMeaCoatDeviceNet, 355
GetPCoeff
CRobo_FYITemp_FunctionNet, 460
GetPDecp
CTcxDeviceNet, 600
GetPeriod
CPulseGeneratorFunctionNet, 447
GetPeriod_us
CTEERFunctionNet, 612
GetPermanentCurrentInMicroAmp
CMultiwellOptoStimFunctionNet, 399
GetPGain
CRoboDacqNet, 469
GetPhases
CRoboDeviceNet::RoboMainLowLevelCommands, 711
GetPicFirmwareType
CW2100_FunctionNet, 625
GetPiezoState
CMcsBus_SensorNet, 237
GetPlateClampLockState
CMultiwellDeviceNet, 391
GetPlateClampState
CMultiwellDeviceNet, 391
GetPlateClampStateByHeadstage
CMultiwellCallbackFunctionNet, 388
GetPlateClampStateByHeadstageEvent
CMultiwellCallbackFunctionNet, 388
GetPlateMux
CMultiwellDeviceNet, 391, 392
GetPlateType
CMultiwellDeviceNet, 392
GetPMax

CTcxDeviceNet, 600
GetPMin
CTcxDeviceNet, 600
GetPoti
CMcsUsbDacqNet, 282
GetPOut
CTcxDeviceNet, 600
GetPowerMuxPlate
CMultiwellDeviceNet, 392
GetPowerStrength
CPositionIIDeviceNet, 421
GetPressure
CMcsBus_SensorNet, 237
GetPressureOffset
CMcsBus_SensorNet, 237
GetPressureRange
CPPCFunctionNet, 433
GetPulseform
COkuvisionStimulatorDeviceNet, 407
GetPulseLength
CPulseGeneratorFunctionNet, 447
GetPumpCouple
CPPS_FunctionNet, 440
GetPumpEnableSpeedRatio
CPPS_FunctionNet, 440
GetPumpFastOnOff
CPPS_FunctionNet, 440
GetPumpFastSpeed
CPPS_FunctionNet, 440
GetPumpFunctionSpeeds
CPPS_FunctionNet, 441
GetPumpManualOnOff
CPPS_FunctionNet, 441
GetPumpMaxSpeed
CPPS_FunctionNet, 441
GetPumpModeType
CPPCFunctionNet, 433
CPPS_FunctionNet, 441
GetPumpSpeed
CRoboFluidDeviceNet, 493
GetPumpSpeedRatio
CPPS_FunctionNet, 441
GetPumpSpeedUnit
CPPCFunctionNet, 434
CPPS_FunctionNet, 441
GetPWM
CFluidControlDeviceNet, 150
GetPwrOut
CTcxDeviceNet, 601
GetPwrSet
CTcxDeviceNet, 601
GetRatedCapacity
CMultiBatteryChargerDeviceNet, 385
GetReady
CMeaImpedanceDeviceNet, 374
GetRecordingNumber
CRoboDacqNet, 469
GetReferenceElectrodeMode
CSCUFunctionNet, 518
GetReferenceElectrodeSwitchState
CSCUFunctionNet, 518
GetReferenceTemperature
CFluidControlDeviceNet, 150
GetRegulationTimeouts
CMcsBus_SensorNet, 237
GetRegulatorFactor
CMcsBus_SensorNet, 238
GetRegulatorOnOff
CMcsBus_SensorNet, 238
CRobo_FYITemp_FunctionNet, 460
GetRegulatorStatus
CMcsBus_SensorNet, 238
GetRepeats
CProgramPressureCurveNet, 445
GetRes1
CTcxDeviceNet, 601
GetRes2
CTcxDeviceNet, 601
GetResetFilter
CWClassicFunctionNet, 683
GetResistanceC
CRoboDacqNet, 469
GetResistanceV
CRoboDacqNet, 469
GetResolutionPerDigit
CMcsUsbDacqNet, 283
GetResS
CTcxDeviceNet, 601
GetResult
CMeaImpedanceDeviceNet, 374
GetResX
CTcxDeviceNet, 601
GetRFConnectionStatus
CWClassicFunctionNet, 683
GetRFFrequency
CPositionImpDeviceNet, 427
GetRFFrequencyHeadstage
CWClassicFunctionNet, 683
GetRFFrequencyReceiver
CWClassicFunctionNet, 683
GetRFPower
CWClassicFunctionNet, 684
GetRoboDacq
CRoboocyte2DeviceNet, 497
GetRoboFluidDevice
CEncapsulatorDeviceNet, 137
CRoboocyte2DeviceNet, 497
GetRotaryPositionCode
CTEERFunctionNet, 612
GetRotatePump
CMcsBus_SensorNet, 238
GetROut
CTcxDeviceNet, 601
GetRTC
COkuvisionStimulatorDeviceNet, 408
CPositionIIDeviceNet, 421

GetSampleBufferChunk
CTEERFunctionNet, 612
GetSampleInterval
CLIH3DeviceNet, 205
GetSamplePeriode
CMcsBus_SensorNet, 238
GetSampleRate
CTEERFunctionNet, 612
GetSamplerate
CMcsUsbDacqNet, 283
GetSampleVoltageBuffer_uV
CTEERFunctionNet, 613
GetScaleFactorU1
CTEERFunctionNet, 613
GetScaleFactorU2
CTEERFunctionNet, 613
GetScanHeadstagesResult
CWClassicFunctionNet, 684
GetScreen
CRoboDacqNet, 469
GetSearchReferenceFastAccel
CRoboDeviceNet::RoboMainLowLevelCommands, 711
GetSearchReferenceFastSpeed
CRoboDeviceNet::RoboMainLowLevelCommands, 711
GetSearchReferenceFineAccel
CRoboDeviceNet::RoboMainLowLevelCommands, 711
GetSearchReferenceFineSpeed
CRoboDeviceNet::RoboMainLowLevelCommands, 711
GetSearchReferenceMethod
CRoboDeviceNet::RoboMainLowLevelCommands, 712
GetSearchReferenceMoveOut
CRoboDeviceNet::RoboMainLowLevelCommands, 712
GetSearchReferenceOffsetPos
CRoboDeviceNet::RoboMainLowLevelCommands, 712
GetSelectedChannels
CW2100_FunctionNet, 626
GetSelectedHeadstage
CWClassicFunctionNet, 684
GetSelectedHeadstageState
CW2100_FunctionNet, 626
GetSensorType
CTcxDeviceNet, 602
GetSerialNumber
CMcsUsbNet, 333
DriverVersionNet, 695
GetSerialNumberHeadstage
CWClassicFunctionNet, 684
GetSetpoint
CTcxDeviceNet, 602
GetSetpointDecp
CTcxDeviceNet, 602
GetSetpointMax
CTcxDeviceNet, 602
GetSetpointMin
CTcxDeviceNet, 602
GetShortBuffer
CGenericDevelopDeviceNet, 164
GetSimulation
CRoboDacqNet, 469
GetSingleHeater
CMcsBus_FYIExtensionNet, 213
GetSingleValve
CFluidControlDeviceNet, 151
CRoboFluidDeviceNet, 493
GetSlope
CMeaCleanDeviceNet, 351
CMeaCoatDeviceNet, 356
GetSoftwareKey
CMcsUsbNet, 333
GetSoftwareKeyString
CMcsUsbNet, 333
GetSollPressure
CMcsBus_SensorNet, 238
GetSollTemp
CRobo_FYITemp_FunctionNet, 460
GetSourceBulk
CCMOSMea_FunctionNet, 114
GetSourceDrain
CCMOSMea_FunctionNet, 114
GetSourceGate
CCMOSMea_FunctionNet, 114
GetStartTriggerSlope
CDigOutStimulatorFunctionNet, 134
GetState
CRFFunctionNet, 456
CRobo_FYIProgram_FunctionNet, 459
GetStateDebugData
CPositionIIDeviceNet, 421
GetStateEventData
CPositionIIDeviceNet, 422
GetStatus
CMcsUsbNet, 333
DriverVersionNet, 696
GetStatusOfLastCommand
CMcsUsbNet, 333
GetStgProgramInfo
CStg200xBasicNet, 542
GetStgVersionInfo
CStg200xBasicNet, 543
GetStimulationPatternMemory
CW2100_StimulatorFunctionNet, 632
GetStimulatorStatus
COkuvisionStimulatorDeviceNet, 408
GetStimulusParametersCache
CW2100_FunctionNet, 626
GetStimulusParametersFromSelectedHS
CW2100_FunctionNet, 626
GetStimulusSites
CCMOSMea_FunctionNet, 115

GetStiumlusParameters
CW2100_FunctionNet, 626
GetStopTriggerSlope
CDigOutStimulatorFunctionNet, 135
GetSubChannel
CMcsBus_MotorControlNet, 223
GetSupplyVoltage
CPPCFunctionNet, 434
CPPS_FunctionNet, 441
GetSweepCount
CStg200xDownloadBasicNet, 567
GetSync
CWarnerValveControllerDeviceTesterFunctionNet, 680
GetSyncoutMap
CStg200xBasicNet, 543
GetSyncState
CMcsBus_SensorNet, 238
GetTableName
CWarnerValveControllerDeviceNet, 660
GetTableNamebyIndex
CWarnerValveControllerDeviceNet, 661
GetTableNamebyIndexEvent
CWarnerValveControllerDeviceNet, 676
GetTablepointer
CRetinaLedDeviceNet, 452
GetTemperatur
CMcsBus_TempSensorNet, 242
GetTestMode
CRFFunctionNet, 456
GetThermocoupleCalibration
CFluidControlDeviceNet, 151
CTcxDeviceNet, 602
GetThermocoupleNanovoltPerKelvin
CFluidControlDeviceNet, 151
CTcxDeviceNet, 602
GetThermocoupleReferenceTemp
CTcxDeviceNet, 603
GetThermocoupleTemp
CTcxDeviceNet, 603
GetThermocoupleTempAbs
CTcxDeviceNet, 603
GetThermocoupleTemperature
CFluidControlDeviceNet, 152
GetThermoOffset
CMcsBus_TempSensorNet, 242
GetThermoTemp
CMcsBus_TempSensorNet, 243
GetThermoVoltage
CMcsBus_TempSensorNet, 243
GetTimeInPause
CMeaCoatDeviceNet, 356
GetTimeInPlateau
CMeaCoatDeviceNet, 356
GetTimeResolutionInNanoSeconds
CStg200xBasicNet, 543
CW2100_StimulatorFunctionNet, 633
GetTimeSlot
CW2100_StimulatorFunctionNet, 633
GetTotalMemory
CStg200xBasicNet, 544
CStimulusFunctionNet, 585
GetTotalNumberOfDigitalPorts
CWarnerValveControllerDeviceNet, 661
GetTotalNumberOfTables
CWarnerValveControllerDeviceNet, 661
GetTotalNumberOfValves
CWarnerValveControllerDeviceNet, 661
GetTotalTableSize
CWarnerValveControllerDeviceNet, 662
GetTouchPadEnable
CMultiwellDeviceNet, 393
GetTrigger
CStg200xDownloadBasicNet, 568
GetTriggerSource
CStg200xBasicNet, 544
GetU1Offset
CWarnerUssingFunctionNet, 644
GetU1Reference
CWarnerUssingFunctionNet, 645
GetU2Offset
CWarnerUssingFunctionNet, 645
GetU2Reference
CWarnerUssingFunctionNet, 645
GetUByteBuffer
CGenericDevelopDeviceNet, 165
GetUC
CRoboDacqNet, 469
GetUClamp
CRoboDacqNet, 470
GetUCOffset
CRoboDacqNet, 470
GetUintA
CFilterCoefficientsNet, 141
GetUintB
CFilterCoefficientsNet, 141
GetUIntBuffer
CGenericDevelopDeviceNet, 165
GetUints
CFilterCoefficientsNet, 141
GetUnit
CTcxDeviceNet, 603
GetUnitDescription
CWarnerUssingFunctionNet, 646
GetUnitExponent
CWarnerUssingFunctionNet, 646
GetUnitName
CWarnerUssingFunctionNet, 646
GetUnitsPerDigit
CWarnerUssingFunctionNet, 647
GetUOut
CTcxDeviceNet, 603
GetUpdateDisplay
CRoboDacqNet, 470
GetUpperFrequencyByIndex
CIntanMea_FunctionNet, 196

GetUptimeSeconds
CTEERFunctionNet, 613
CWarnerUssingFunctionNet, 647
GetUSBDeviceIDFromFX3Image
CMcsUsbFactoryNet, 309
GetUsbListEntries
CMcsUsbListNet, 321
GetUsbListEntry
CMcsUsbListNet, 321
CMcsUsbNet, 334
GetUseBubble
CPPS_FunctionNet, 441
GetUsercodeFromBitFile
CMcsUsbFactoryNet, 309
GetUsercodeFromFlash
CMcsUsbFactoryNet, 309
GetUserDefinedName
CW2100_FunctionNet, 626
GetUserDefinedNameCache
CW2100_FunctionNet, 626
GetUserDefinedNameFromSelectedHS
CW2100_FunctionNet, 627
GetUserParameter
CRoboDeviceNet::RoboMainLowLevelCommands, 712
GetUShortBuffer
CGenericDevelopDeviceNet, 166
GetUV
CRoboDacqNet, 470
GetUVOffset
CRoboDacqNet, 470
GetValue
CTcxDeviceNet, 603
GetValueHires
CTcxDeviceNet, 604
GetValve
CFluidControlDeviceNet, 152
CRoboFluidDeviceNet, 493
GetValve1
CRobo_FYIProgram_FunctionNet, 459
GetValve2
CRobo_FYIProgram_FunctionNet, 459
GetValveActive
CPPCFunctionNet, 434
CWarnerValveControllerDeviceNet, 662
GetValveActiveEvent
CWarnerValveControllerDeviceNet, 676
GetValveBoardRevision
CWarnerValveControllerDeviceNet, 662
GetValveBoardRevisionEvent
CWarnerValveControllerDeviceNet, 676
GetValveBoardRevisionString
CWarnerValveControllerDeviceNet, 662
GetValveCurrent
CWarnerValveControllerDeviceNet, 662
GetValveDigitalInPort
CWarnerValveControllerDeviceNet, 663
GetValveDigitalInPortEvent
CWarnerValveControllerDeviceNet, 677
GetValveLedOn
CWarnerValveControllerDeviceNet, 663
GetValveLedOnEvent
CWarnerValveControllerDeviceNet, 677
GetValveManualGroup
CWarnerValveControllerDeviceNet, 663
GetValveManualGroupEvent
CWarnerValveControllerDeviceNet, 677
GetValveManualState
CWarnerValveControllerDeviceNet, 664
GetValveManualStateEvent
CWarnerValveControllerDeviceNet, 677
GetValveMode
CWarnerValveControllerDeviceNet, 664
GetValveModeEvent
CWarnerValveControllerDeviceNet, 677
GetValves
CMcsBus_FYIExtensionNet, 213
GetValvesActiveMap
CWarnerValveControllerDeviceNet, 664
GetValvesManualStateMap
CWarnerValveControllerDeviceNet, 664
GetValveTableEntry
CWarnerValveControllerDeviceNet, 665
GetVDD3I
CCMOSMea_FunctionNet, 115
GetVDDI
CCMOSMea_FunctionNet, 115
GetVdsVgs
CGrapheneFunctionNet, 181
GetVdVs
CGrapheneFunctionNet, 182
GetVdVsDAC
CGrapheneFunctionNet, 182, 183
GetVersion
CMcsUsbNet, 334
GetVersionInt
DriverVersionNet, 696
GetVersionString
DriverVersionNet, 697
GetVMMaxNegativeCurrent
CMcsBus_VoltageModeNet, 245
GetVMMaxNegativeCurrentEeprom
CMcsBus_VoltageModeNet, 245
GetVMMaxNegativeVoltage
CMcsBus_VoltageModeNet, 245
GetVMMaxNegativeVoltageEeprom
CMcsBus_VoltageModeNet, 245
GetVMMaxPositiveCurrent
CMcsBus_VoltageModeNet, 245
GetVMMaxPositiveCurrentEeprom
CMcsBus_VoltageModeNet, 245
GetVMMaxPositiveVoltage
CMcsBus_VoltageModeNet, 245
GetVMMaxPositiveVoltageEeprom
CMcsBus_VoltageModeNet, 246
GetVMOutputOnOff

CMcsBus_VoltageModeNet, 246
GetVMVoltage
CMcsBus_VoltageModeNet, 246
GetVolatileClampOffset
CMultiwellDeviceNet, 393
GetVoltage
COkuvisionStimulatorDeviceNet, 408
GetVoltage12V
CRoboDeviceNet, 482
GetVoltage12VLimit
CRoboDeviceNet, 482
GetVoltage5V
CRoboDeviceNet, 482
GetVoltage5VLimit
CRoboDeviceNet, 482
GetVoltageAirvalve
CRoboDeviceNet, 482
GetVoltageAirvalveLimit
CRoboDeviceNet, 482
GetVoltageClampControllerParam_D
CWarnerUssingFunctionNet, 647
GetVoltageClampControllerParam_I
CWarnerUssingFunctionNet, 648
GetVoltageClampControllerParam_P
CWarnerUssingFunctionNet, 648
GetVoltageRange
CGrapheneFunctionNet, 183
GetVoltageRangeIndex
CMcsUsbDacqNet, 283
GetVoltageRangeInMicroVolt
CMcsUsbDacqNet, 283
CStg200xBasicNet, 544
CStimulusFunctionNet, 585
CW2100_StimulatorFunctionNet, 633
GetVoltageRangeInMicroVoltByIndex
CStg200xBasicNet, 544
GetVoltageRangeInMilliVolt
CMcsUsbDacqNet, 283
GetVoltageRangeListInMicroVolt
CStg200xBasicNet, 545
GetVoltageRangeSelectedIndex
CStg200xBasicNet, 545
GetVoltageReached
CGrapheneFunctionNet, 183
GetVoltageResolution
CGrapheneFunctionNet, 184
GetVoltageResolutionInMicroVolt
CStg200xBasicNet, 545
CStimulusFunctionNet, 586
CW2100_StimulatorFunctionNet, 633
GetVoltageRs485A
CRoboDeviceNet, 483
GetVoltageRs485ALimit
CRoboDeviceNet, 483
GetVoltageRs485B
CRoboDeviceNet, 483
GetVoltageRs485BLimit
CRoboDeviceNet, 483
GetVoltageValves
CRoboDeviceNet, 483
GetVoltageValvesLimit
CRoboDeviceNet, 483
GetVolti
CTcxDeviceNet, 604
GetWaveform
CTEERFunctionNet, 613
GetWaveLengthInNanometer
CMultiwellOptoStimFunctionNet, 399
GetWorkingFrequency
CRFFunctionNet, 457
GetWPADebugMode
CWClassicFunctionNet, 684
GetWPAType
CWClassicFunctionNet, 684
GetXGain
CRoboDacqNet, 470
GetXilinxFlashOffset
CMcsUsbFactoryNet, 309
GetXilinxFlashReadCommand
CMcsUsbFactoryNet, 309
GND_SWITCH_BIT
CW2100_StimulatorFunctionNet, 635
Graphene_ASIC
Mcs::Usb, 80
GrapheneASIC
Mcs::Usb, 83
GrapheneASICHeadstage
Mcs::Usb, 68
GrapheneProjectTestDevice
Mcs::Usb, 81
Ground
Mcs::Usb, 58, 66
GroupID
CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumTemplateNet >, 131
GroupType
CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumTemplateNet >, 131
GyroOnly
Mcs::Usb, 93

Hardware
Mcs::Usb, 68
HasAnalogOut
CSCUFunctionNet, 518
HasDigitalPort
CStg200xBasicNet, 545
HasFilterAmp
CStg200xBasicNet, 545
HasGalvanicIsolation
CSCUFunctionNet, 518
HasHSPowerSwitch
CSCUFunctionNet, 519
HasIMU
HeadStageIDType, 702
HasOptoCurrentMessurement
HeadStageIDType, 702

HasRadioControl
CRadioControledDevicesNet, 450
HasRef
CRoboDeviceNet::RoboMainLowLevelCommands, 713
HasRefI
CRoboStatorDeviceNet, 500
HasRefXY
CRoboStatorDeviceNet, 500
HasRefZ
CRoboStatorDeviceNet, 500
HasSoftwareKey
CMcsUsbNet, 334
Headstage1NCBathCurrentGroup
Mcs::Usb, 62
Headstage1NCCol2CurrentGroup
Mcs::Usb, 62
Headstage1NChipTempGroup
Mcs::Usb, 62
HeadstageElectrodeGroup
Mcs::Usb, 62
HeadstageIdEnumNet
Mcs::Usb, 69
HeadStageIDType, 700
CompareTo, 702
ElectricalStimulation, 701
Entry, 702
Equals, 702
HasIMU, 702
HasOptoCurrentMessurement, 702
HeadStageIDType, 701
HeadstageType, 702
HeadstageTypeEnum, 701
ID, 702
MeasuringOnly, 701
NumberOfAnalogChannels, 702
NumberOfStimulationChannels, 703
OpticalStimulation, 701
SN, 703
StimulusParameters, 703
ToString, 702
Type, 703
TypeValue, 703
Unknown, 701
UserDefinedName, 703
Valid, 703
W16IsW14, 703
HeadstageIDTypeObject, 703
_AdditionalText, 704
_IdType, 704
AdditionalText, 705
Equals, 704
GetHashCode, 704
HeadstageIDTypeObject, 704
IdType, 705
ToString, 704
HeadStageIDTypeState, 705
ControlState, 705
DataState, 705
IdType, 705
State, 705
HeadstageType
HeadStageIDType, 702
HeadstageTypeEnum
HeadStageIDType, 701
HEKA_LIH3_DEVICE
Mcs::Usb, 64
HekaEPC10Double
Mcs::Usb, 78
HekaEPC10Quadro
Mcs::Usb, 78
HekaEPC10Single
Mcs::Usb, 78
HekaEPC10Triple
Mcs::Usb, 78
HekaEPCLite
Mcs::Usb, 78
HekaITEV100
Mcs::Usb, 78
HekaLIH30
Mcs::Usb, 78
HekaLIH406
Mcs::Usb, 78
HekaLIH816
Mcs::Usb, 78
HekaPG610
Mcs::Usb, 78
HekaPG611
Mcs::Usb, 78
HekaPG612
Mcs::Usb, 78
HekaPG618
Mcs::Usb, 78
HekaPG690
Mcs::Usb, 78
HiClamp
Mcs::Usb, 80
HiClamp4Uart
Mcs::Usb, 80
Highpass
Mcs::Usb, 67
HighSpeed
Mcs::Usb, 72
HLA
Mcs::Usb, 78
HLADacq
CHLADeviceNet, 191
Hs1Digital
Mcs::Usb, 64
HS1DigitalData1
Mcs::Usb, 88
HS1ElectrodeGroup
Mcs::Usb, 72
HS1Sideband1
Mcs::Usb, 57, 73, 75, 76
HS1Sideband10

Mcs::Usb, 73
HS1Sideband11
Mcs::Usb, 73
HS1Sideband12
Mcs::Usb, 73
HS1Sideband13
Mcs::Usb, 73
HS1Sideband14
Mcs::Usb, 73
HS1Sideband15
Mcs::Usb, 73
HS1Sideband16
Mcs::Usb, 73
HS1Sideband17
Mcs::Usb, 73
HS1Sideband18
Mcs::Usb, 73
HS1Sideband2
Mcs::Usb, 58, 73, 75, 76
HS1Sideband3
Mcs::Usb, 58, 73, 75
HS1Sideband4
Mcs::Usb, 73, 75
HS1Sideband5
Mcs::Usb, 73, 75
HS1Sideband6
Mcs::Usb, 73, 75
HS1Sideband7
Mcs::Usb, 73
HS1Sideband8
Mcs::Usb, 73
HS1Sideband9
Mcs::Usb, 73
Hs1SidebandHigh
Mcs::Usb, 64
Hs1SidebandLow
Mcs::Usb, 64
Hs1Trigger
Mcs::Usb, 64
HS1Trigger10Status
Mcs::Usb, 73
HS1Trigger11Status
Mcs::Usb, 73
HS1Trigger12Status
Mcs::Usb, 73
HS1Trigger13Status
Mcs::Usb, 73
HS1Trigger14Status
Mcs::Usb, 73
HS1Trigger15Status
Mcs::Usb, 73
HS1Trigger16Status
Mcs::Usb, 73
HS1Trigger17Status
Mcs::Usb, 73
HS1Trigger18Status
Mcs::Usb, 73
HS1Trigger1Status
Mcs::Usb, 57, 73, 75, 76
HS1Trigger2Status
Mcs::Usb, 57, 73, 75, 76
HS1Trigger3Status
Mcs::Usb, 57, 73, 75
HS1Trigger4Status
Mcs::Usb, 73, 75
HS1Trigger5Status
Mcs::Usb, 73, 75
HS1Trigger6Status
Mcs::Usb, 73, 75
HS1Trigger7Status
Mcs::Usb, 73
HS1Trigger8Status
Mcs::Usb, 73
HS1Trigger9Status
Mcs::Usb, 73
Hs2Digital
Mcs::Usb, 65
HS2DigitalData1
Mcs::Usb, 88
HS2ElectrodeGroup
Mcs::Usb, 72
HS2Sideband1
Mcs::Usb, 74–76
HS2Sideband10
Mcs::Usb, 74
HS2Sideband11
Mcs::Usb, 74
HS2Sideband12
Mcs::Usb, 74
HS2Sideband13
Mcs::Usb, 74
HS2Sideband14
Mcs::Usb, 74
HS2Sideband15
Mcs::Usb, 74
HS2Sideband16
Mcs::Usb, 74
HS2Sideband17
Mcs::Usb, 74
HS2Sideband18
Mcs::Usb, 74
HS2Sideband2
Mcs::Usb, 74–76
HS2Sideband3
Mcs::Usb, 74, 75
HS2Sideband4
Mcs::Usb, 74, 75
HS2Sideband5
Mcs::Usb, 74, 75
HS2Sideband6
Mcs::Usb, 74, 75
HS2Sideband7
Mcs::Usb, 74
HS2Sideband8
Mcs::Usb, 74
HS2Sideband9

Mcs::Usb, 74
Hs2SidebandHigh
Mcs::Usb, 65
Hs2SidebandLow
Mcs::Usb, 65
Hs2Trigger
Mcs::Usb, 65
HS2Trigger10Status
Mcs::Usb, 74
HS2Trigger11Status
Mcs::Usb, 74
HS2Trigger12Status
Mcs::Usb, 74
HS2Trigger13Status
Mcs::Usb, 74
HS2Trigger14Status
Mcs::Usb, 74
HS2Trigger15Status
Mcs::Usb, 74
HS2Trigger16Status
Mcs::Usb, 74
HS2Trigger17Status
Mcs::Usb, 74
HS2Trigger18Status
Mcs::Usb, 74
HS2Trigger1Status
Mcs::Usb, 73, 75, 76
HS2Trigger2Status
Mcs::Usb, 73, 75, 76
HS2Trigger3Status
Mcs::Usb, 73, 75
HS2Trigger4Status
Mcs::Usb, 73, 75
HS2Trigger5Status
Mcs::Usb, 74, 75
HS2Trigger6Status
Mcs::Usb, 74, 75
HS2Trigger7Status
Mcs::Usb, 74
HS2Trigger8Status
Mcs::Usb, 74
HS2Trigger9Status
Mcs::Usb, 74
HWInfo
CMcsUsbDacqNet, 283
HwVersion
CMcsUsbListEntryNet, 319

ID
HeadStageIDType, 702
Idle
Mcs::Usb, 86
IdProduct
DeviceIdNet, 688
IdType
HeadstageIDTypeObject, 705
HeadStageIDTypeState, 705
IdVendor
DeviceIdNet, 688
IFB2GoldenInterfaceboard
Mcs::Usb, 69
IFB30GoldenInterfaceboard
Mcs::Usb, 69
IFChannel1
Mcs::Usb, 61
IFChannel2
Mcs::Usb, 61
IFChannel3
Mcs::Usb, 61
IFChannel4
Mcs::Usb, 61
IFChannel5
Mcs::Usb, 61
IFChannel6
Mcs::Usb, 61
IFChannel7
Mcs::Usb, 61
IFChannel8
Mcs::Usb, 61
IFDigChannelsGroup
Mcs::Usb, 62, 72, 84, 94
Ignore
Mcs::Usb, 82
IM16KRC
Mcs::Usb, 80
IM16S16KRA
Mcs::Usb, 80
IM16S8KRA
Mcs::Usb, 80
IM64KRB
Mcs::Usb, 80
IM64KRC
Mcs::Usb, 80
Input
Mcs::Usb, 77
Intel
Mcs::Usb, 91
InterfaceADCGroup
Mcs::Usb, 62, 72, 83, 93
InterfaceBoard2
Mcs::Usb, 68
IntToDouble
Mcs::Usb, 67
InvitroSignalCollectorUnit
Mcs::Usb, 69
InvivoSignalCollectorUnit
Mcs::Usb, 69
IoVoltageEnumNet
Mcs::Usb, 69
IS32KRA
Mcs::Usb, 80
IsAnalogOutEnabled
CSCUFunctionNet, 519
IsAutomaticAnalogOut
CSCUFunctionNet, 519
IsBusy
CPPCFunctionNet, 434

IsChamberAvailable
CWarnerUssingFunctionNet, 648
IsChipPowered
CCMOSMea_FunctionNet, 115
IsConnected
CMcsUsbNet, 334
IsDeviceHighSpeed
CMcsUsbNet, 334
IsDeviceHighSpeedCapable
CMcsUsbNet, 334
IsDeviceTypeOf
CMcsUsbListNet, 322
IsDigitalChannelDedicated
CMcsUsbDacqNet::CHWInfo, 194
IsDigitalOutPortInverted
CWarnerValveControllerDeviceNet, 665
IsDigitalOutPortInvertedEvent
CWarnerValveControllerDeviceNet, 677
IsEqual
CFilterCoefficientsNet, 141
IsExceptionsEnabled
CMcsUsbNet, 335
IsGateFloating
CCMOSMea_FunctionNet, 115
IsHeadstageAvailable
CSCUFunctionNet, 519
IsHeadstageAvailableEvent
CSCUFunctionNet, 524
IsHighCurrentMode
CWarnerUssingFunctionNet, 649
IsHSPowered
CSCUFunctionNet, 520
IsInDacqLegacyMode
CSCUFunctionNet, 520
IsInternalCalibrationFinished
CTEERFunctionNet, 614
CWarnerUssingFunctionNet, 649
IsPlateTypeValid
CMultiwellDeviceNet, 393
IsPulseEnabled
CWarnerUssingFunctionNet, 649
IsPumpMotorOn
CRoboFluidDeviceNet, 494
IsQueueEnabled
CRoboDeviceNet, 483
IsQueueStarted
CRoboDeviceNet, 483
IsRunning
CMeaCleanDeviceNet, 351
CMeaCoatDeviceNet, 356
IsSamplingFinished
CTEERFunctionNet, 614
IsUserTriggerEnabled
CLIH3DeviceNet, 205
IsValveDigitalInInverted
CWarnerValveControllerDeviceNet, 665
IsValveDigitalInInvertedEvent
CWarnerValveControllerDeviceNet, 677
IsValveOpen
CWarnerValveControllerDeviceNet, 666
IsValveOpenEvent
CWarnerValveControllerDeviceNet, 678
IsValveOpenInAnalogMode
CWarnerValveControllerDeviceNet, 666
IsValveOpenInAnalogModeEvent
CWarnerValveControllerDeviceNet, 678
IsValveOpenInDigitalMode
CWarnerValveControllerDeviceNet, 666
IsValveOpenInDigitalModeEvent
CWarnerValveControllerDeviceNet, 678

Kelvin
Mcs::Usb, 54

LastPosition
Mcs::Usb, 58, 74–76, 85, 88, 95
LegacyHasOutputMap
CStg200xBasicNet, 546
LegacyMeaUsb
Mcs::Usb, 78
LIH30_ADC_Channel_EnumNet
Mcs::Usb, 69
LIH30_DAC_Channel_EnumNet
Mcs::Usb, 70
LIH30_EPC10_Bus_EnumNet
Mcs::Usb, 70
LIH30ADCCtrl
Mcs::Usb, 69
LIH30ADCModulesGroup
Mcs::Usb, 62
LIH30Interfaceboard
Mcs::Usb, 68
ListModeSendStart
CStg200xBasicNet, 546
ListModeSendStop
CStg200xBasicNet, 546
ListOfChangedTriggers
StgStatusNet, 719
LoadPressure
CPPCFunctionNet, 436
LoadUserFirmware
CMcsUsbFactoryNet, 309, 310
LoadValveTable
CWarnerValveControllerDeviceNet, 666
Lock
Mcs::Usb, 77
LockPlateClamp
CMultiwellDeviceNet, 394
Lowpass
Mcs::Usb, 67
LowSpeed
Mcs::Usb, 72

M4M_Dock
Mcs::Usb, 68
M4M_ReUse
Mcs::Usb, 68

m_Bottom
CCMOSMeaDeviceNet::CRegionOfInterestRect, 451
m_Left
CCMOSMeaDeviceNet::CRegionOfInterestRect, 451
m_pGilsonDevice
CGilsonDeviceNet, 174
m_pMcsBus_MotorControlNet
CRoboFluidDeviceNet, 495
m_pMcsUsb
CMcsUsbFunctionNet, 314
m_pMcsUsbFunction
CMcsUsbFunctionNet, 314
m_pRoboFluidDevice
CRoboFluidDeviceNet, 495
m_Right
CCMOSMeaDeviceNet::CRegionOfInterestRect, 451
m_Top
CCMOSMeaDeviceNet::CRegionOfInterestRect, 451
Manual
Mcs::Usb, 78, 85, 95
Manufacturer
CMcsUsbListEntryNet, 319
MaxBitNumber
DigitalSource< digitalsourceenum >, 689
DigitalSourceGeneral, 690, 691
MaxBitNumberStatic
DigitalSource< digitalsourceenum >, 689
MBC08
Mcs::Usb, 79
MbcChannelStateEnumNet
Mcs::Usb, 71
MbcChargingModeEnumNet
Mcs::Usb, 71
MbcRatedCapacityEnumNet
Mcs::Usb, 71
MC_Card
Mcs::Usb, 78
MCS
Mcs::Usb, 92
Mcs, 23
Mcs::Usb, 23
A, 70
AccelOnly, 93
AdapterTypeEnumNet, 52
ALA_VC3, 78, 92
ALA_VC3_DEVICE, 64
ALTERA, 56
ALTERA_BASE, 56
ALTERA_BOOTSTRAP, 57
ALTERA_GOLD, 56
ALTERA_TARGET1, 57
ALTERA_TARGET2, 57
ALTERA_TARGET3, 57
AlwaysOn, 81
Ampere, 54, 92
AMS_Dongle, 79
Analog, 78, 95
AnalogGroup, 62
AnalogOut_DAC_Range_EnumNet, 53
AnalogSource_HS1, 53
AnalogSource_HS2, 53
AnalogSource_IF, 53
AnalogSourceEnumNet, 53
AnalogUnitEnumNet, 53
Any, 78, 92
Armed, 86
ASMedia, 91
AudioTestChannelGroup, 62, 72, 84, 94
Aux, 64
AuxIn, 57, 73, 74, 76, 84, 88, 95
AuxPort, 60
B, 70
Bessel, 67
BMI, 88
Bootstrap, 55
BootstrapOtherCypress, 56
Both, 93
Break, 82
BUS0MCSBUS0, 55
BUS0MCSBUS1, 54
BUS0MCSBUS10, 54
BUS0MCSBUS11, 54
BUS0MCSBUS12, 54
BUS0MCSBUS13, 54
BUS0MCSBUS14, 54
BUS0MCSBUS15, 55
BUS0MCSBUS2, 54
BUS0MCSBUS3, 54
BUS0MCSBUS4, 54
BUS0MCSBUS5, 54
BUS0MCSBUS6, 54
BUS0MCSBUS7, 54
BUS0MCSBUS8, 54
BUS0MCSBUS9, 54
BUS1MCSBUS0, 55
BUS1MCSBUS1, 55
BUS1MCSBUS10, 55
BUS1MCSBUS11, 55
BUS1MCSBUS12, 55
BUS1MCSBUS13, 55
BUS1MCSBUS14, 55
BUS1MCSBUS15, 55
BUS1MCSBUS2, 55
BUS1MCSBUS3, 55
BUS1MCSBUS4, 55
BUS1MCSBUS5, 55
BUS1MCSBUS6, 55
BUS1MCSBUS7, 55
BUS1MCSBUS8, 55
BUS1MCSBUS9, 55
BUS2MCSBUS0, 55
BUS2MCSBUS1, 55

BUS2MCSBUS10, 55
BUS2MCSBUS11, 55
BUS2MCSBUS12, 55
BUS2MCSBUS13, 55
BUS2MCSBUS14, 55
BUS2MCSBUS15, 55
BUS2MCSBUS2, 55
BUS2MCSBUS3, 55
BUS2MCSBUS4, 55
BUS2MCSBUS5, 55
BUS2MCSBUS6, 55
BUS2MCSBUS7, 55
BUS2MCSBUS8, 55
BUS2MCSBUS9, 55
BUSNUMBER0, 54
BUSNUMBER1, 55
BUSNUMBER2, 55
Butterworth, 67
Campden_Ci4600EphysVideoDataIntegrator, 78
CatchAmp, 77
CFirmwareDestinationNet, 54
channeldata_current, 87
channeldata_current_always_boost, 87
channeldata_current_always_boost_own_sync, 87
channeldata_current_own_boost_gnd_sync, 87
channeldata_current_own_sync, 87
channeldata_positive_current, 87
channeldata_positive_current_own_boost_gnd_sync, 87
channeldata_positive_current_own_sync, 87
channeldata_positive_voltage, 87
channeldata_voltage, 87
ChannelPIC, 55
ChannelTest, 79
ChecksumAndPacketCounter, 61
Ci4600Intan, 53
ClampModeCurrent, 89
ClampModeInternalCalibration, 89
ClampModeOpen, 89
ClampModeVoltage, 89
Close, 77
CmosMea, 69
CMOSMEA5000DigitalSourceEnumNet, 57
CMOSMeaBathModeEnumNet, 58
CmosMeaHeadstage, 68
CMOSMeaHeadstage1NCBathCurrentEnumNet, 58
CMOSMeaHeadstage1NCCol2CurrentEnumNet, 58
CMOSMeaHeadstage1NChipTempEnumNet, 58
CMOSMeaHS1SidebandEnumNet, 60
CMOSMeaHS1TriggerStatusEnumNet, 60
CmosmeaIFB2, 68
CMOSMeaIFDigChannelEnumNet, 60
CMOSMeaInterfaceADCEnumNet, 60
CmosMeaInterfaceboard, 68
CMOSMeaPacketFrameContextGroupEnumNet, 61
CMOSMeaSTG1DACSignalEnumNet, 61
CMOSMeaValueUnitEnumNet, 61
CommaPositionA, 67
CommaPositionB, 67
csCapacityTestDischarge, 71
csCapacityTestPrecharge, 71
csCharge, 71
csDischarge, 71
csError, 71
csIdleChargeFinished, 71
csIdleNoBattery, 71
csRefreshBattery, 71
CurrentClamp, 92
CurrentMeasure, 58
Cypress, 92
Cypress_FX1, 78
Cypress_FX2, 78
Cypress_FX3, 78
DAC1Channel, 61
DAC2Channel, 61
DAC3Channel, 61
DAC4Channel, 61
DACQ1DigitalGroup, 62
DacqGroupChannelEnumNet, 62
DacqMeaGroupTypeEnumNet, 62
DacqTrigger, 65
DataModeEnumNet, 62
DEST_FX3_TARGET_MASK, 57
DEST_TARGET1, 57
DEST_TARGET10, 57
DEST_TARGET11, 57
DEST_TARGET12, 57
DEST_TARGET13, 57
DEST_TARGET14, 57
DEST_TARGET15, 57
DEST_TARGET2, 57
DEST_TARGET3, 57
DEST_TARGET4, 57
DEST_TARGET5, 57
DEST_TARGET6, 57
DEST_TARGET7, 57
DEST_TARGET8, 57
DEST_TARGET9, 57
DEST_TARGET_MASK, 57
DEVICE_NOT_FOUND, 66
DeviceEnumNet, 63
DeviceHasNoHeadstage, 69, 83
DeviceNotConnected, 68, 69, 83
DeviceRunStatus, 58, 73, 75, 76, 84, 88, 95
DigDataFromReceiver, 95
Digital, 78, 95
DigitalData, 58, 73, 75, 76, 84, 88, 95
DigitalDatastreamEnableEnumNet, 64
DigitalGroup, 62
DigitalIn, 57, 64, 72, 74, 76, 84, 88, 94
DigitalInOfOutPort, 57, 72, 74, 76, 84, 88, 94
DigitalInPort, 60
DigitalInReserverd, 64

DigitalMux, 60
DigitalOut, 64
DigitalOutReg, 60
DigitalOutReserved, 64
DigitalOutStimulator, 58, 73, 75, 76, 84, 88, 95
DigitalPulse, 57, 72, 74, 76, 84, 88, 95
DigitalReg, 60
DigitalStimulatorTriggerEventEnumNet, 65
DigitalStimulatorTriggerSlopeEnumNet, 65
DigitalTargetEnumNet, 65
Digout, 65
DigOutStim, 64
DigOutStimulatorStartTrigger, 65
DigOutStimulatorStopTrigger, 65
Digstream, 65
DigStreamFromReceiver, 95
DigStreamToReceiver, 65
Dilutor, 80
DongleS, 79
Dotriapot, 78
DoubleToInt, 67
DownloadOnly, 86
DSP, 54
DSPAnalogGroup, 72, 84
DSPDataGroup, 62, 93
DSPDigitalGroup, 72, 84
eCube, 80, 83
eCubeHeadstage, 68
ElectrodeDacMuxEnumNet, 65
ElectrodeModeEnumNet, 66
ElectrodeOffset, 92
emAutomatic, 66
emManual, 66
Encapsulator, 80
enCMosMeaChipType, 66
EnSTG200x_STATUS, 66
EOFAndCRC, 61
ExtendedMultiFileMode, 86
ExternBCTester, 79
ExternDTester, 79
ExternSTester, 79
Falling, 65
FCB, 79
FCX, 79
Feedback, 57, 73, 74, 76, 84, 88, 95
FeedbackHigh, 64
FeedbackLow, 64
FeedbackReg, 60
FilterAttributeEnumNet, 66
FilterBandEnumNet, 67
FilterCalculationDirectionEnumNet, 67
FilterFamilyEnumNet, 67
FilterTypeEnumNet, 67
Finished, 86
FPGA10, 56
FPGA10_BASE, 56
FPGA10_GOLD, 56
FPGA11, 56
FPGA11_BASE, 56
FPGA11_GOLD, 56
FPGA12, 56
FPGA12_BASE, 56
FPGA12_GOLD, 56
FPGA13, 56
FPGA13_BASE, 56
FPGA13_GOLD, 56
FPGA14, 56
FPGA14_BASE, 56
FPGA14_GOLD, 56
FPGA15, 56
FPGA15_BASE, 57
FPGA15_GOLD, 56
FPGA16, 56
FPGA16_BASE, 57
FPGA16_GOLD, 56
FPGA2, 56
FPGA2_BASE, 56
FPGA2_GOLD, 56
FPGA3, 56
FPGA3_BASE, 56
FPGA3_GOLD, 56
FPGA4, 56
FPGA4_BASE, 56
FPGA4_GOLD, 56
FPGA5, 56
FPGA5_BASE, 56
FPGA5_GOLD, 56
FPGA6, 56
FPGA6_BASE, 56
FPGA6_GOLD, 56
FPGA7, 56
FPGA7_BASE, 56
FPGA7_GOLD, 56
FPGA8, 56
FPGA8_BASE, 56
FPGA8_GOLD, 56
FPGA9, 56
FPGA9_BASE, 56
FPGA9_GOLD, 56
FPGA_BASE, 56
FPGA_BOOTSTRAP, 57
FPGA_GOLD, 56
FPGA_NORMAL, 54
FpgaIdEnumNet, 68
FrameContextGroup, 62
FullCharge, 71
FullSpeed, 72
FunkDongleS, 79
Gain100, 87
Gain1000, 87
Gain200, 87
Gain500, 87
Gate, 82
Gated_High_Active, 81
Gated_Low_Active, 81
GE2100, 80

Generic, 79
Graphene_ASIC, 80
GrapheneASIC, 83
GrapheneASICHeadstage, 68
GrapheneProjectTestDevice, 81
Ground, 58, 66
GyroOnly, 93
Hardware, 68
Headstage1NCBathCurrentGroup, 62
Headstage1NCCol2CurrentGroup, 62
Headstage1NChipTempGroup, 62
HeadstageElectrodeGroup, 62
HeadstageIdEnumNet, 69
HEKA_LIH3_DEVICE, 64
HekaEPC10Double, 78
HekaEPC10Quadro, 78
HekaEPC10Single, 78
HekaEPC10Triple, 78
HekaEPCLite, 78
HekaITEV100, 78
HekaLIH30, 78
HekaLIH406, 78
HekaLIH816, 78
HekaPG610, 78
HekaPG611, 78
HekaPG612, 78
HekaPG618, 78
HekaPG690, 78
HiClamp, 80
HiClamp4Uart, 80
Highpass, 67
HighSpeed, 72
HLA, 78
Hs1Digital, 64
HS1DigitalData1, 88
HS1ElectrodeGroup, 72
HS1Sideband1, 57, 73, 75, 76
HS1Sideband10, 73
HS1Sideband11, 73
HS1Sideband12, 73
HS1Sideband13, 73
HS1Sideband14, 73
HS1Sideband15, 73
HS1Sideband16, 73
HS1Sideband17, 73
HS1Sideband18, 73
HS1Sideband2, 58, 73, 75, 76
HS1Sideband3, 58, 73, 75
HS1Sideband4, 73, 75
HS1Sideband5, 73, 75
HS1Sideband6, 73, 75
HS1Sideband7, 73
HS1Sideband8, 73
HS1Sideband9, 73
Hs1SidebandHigh, 64
Hs1SidebandLow, 64
Hs1Trigger, 64
HS1Trigger10Status, 73
HS1Trigger11Status, 73
HS1Trigger12Status, 73
HS1Trigger13Status, 73
HS1Trigger14Status, 73
HS1Trigger15Status, 73
HS1Trigger16Status, 73
HS1Trigger17Status, 73
HS1Trigger18Status, 73
HS1Trigger1Status, 57, 73, 75, 76
HS1Trigger2Status, 57, 73, 75, 76
HS1Trigger3Status, 57, 73, 75
HS1Trigger4Status, 73, 75
HS1Trigger5Status, 73, 75
HS1Trigger6Status, 73, 75
HS1Trigger7Status, 73
HS1Trigger8Status, 73
HS1Trigger9Status, 73
Hs2Digital, 65
HS2DigitalData1, 88
HS2ElectrodeGroup, 72
HS2Sideband1, 74–76
HS2Sideband10, 74
HS2Sideband11, 74
HS2Sideband12, 74
HS2Sideband13, 74
HS2Sideband14, 74
HS2Sideband15, 74
HS2Sideband16, 74
HS2Sideband17, 74
HS2Sideband18, 74
HS2Sideband2, 74–76
HS2Sideband3, 74, 75
HS2Sideband4, 74, 75
HS2Sideband5, 74, 75
HS2Sideband6, 74, 75
HS2Sideband7, 74
HS2Sideband8, 74
HS2Sideband9, 74
Hs2SidebandHigh, 65
Hs2SidebandLow, 65
Hs2Trigger, 65
HS2Trigger10Status, 74
HS2Trigger11Status, 74
HS2Trigger12Status, 74
HS2Trigger13Status, 74
HS2Trigger14Status, 74
HS2Trigger15Status, 74
HS2Trigger16Status, 74
HS2Trigger17Status, 74
HS2Trigger18Status, 74
HS2Trigger1Status, 73, 75, 76
HS2Trigger2Status, 73, 75, 76
HS2Trigger3Status, 73, 75
HS2Trigger4Status, 73, 75
HS2Trigger5Status, 74, 75
HS2Trigger6Status, 74, 75
HS2Trigger7Status, 74
HS2Trigger8Status, 74

HS2Trigger9Status, 74
Idle, 86
IFB2GoldenInterfaceboard, 69
IFB30GoldenInterfaceboard, 69
IFChannel1, 61
IFChannel2, 61
IFChannel3, 61
IFChannel4, 61
IFChannel5, 61
IFChannel6, 61
IFChannel7, 61
IFChannel8, 61
IFDigChannelsGroup, 62, 72, 84, 94
Ignore, 82
IM16KRC, 80
IM16S16KRA, 80
IM16S8KRA, 80
IM64KRB, 80
IM64KRC, 80
Input, 77
Intel, 91
InterfaceADCGroup, 62, 72, 83, 93
InterfaceBoard2, 68
IntToDouble, 67
InvitroSignalCollectorUnit, 69
InvivoSignalCollectorUnit, 69
IoVoltageEnumNet, 69
IS32KRA, 80
Kelvin, 54
LastPosition, 58, 74–76, 85, 88, 95
LegacyMeaUsb, 78
LIH30_ADC_Channel_EnumNet, 69
LIH30_DAC_Channel_EnumNet, 70
LIH30_EPC10_Bus_EnumNet, 70
LIH30ADCCtrl, 69
LIH30ADCModulesGroup, 62
LIH30Interfaceboard, 68
Lock, 77
Lowpass, 67
LowSpeed, 72
M4M_Dock, 68
M4M_ReUse, 68
Manual, 78, 85, 95
MBC08, 79
MbcChannelStateEnumNet, 71
MbcChargingModeEnumNet, 71
MbcRatedCapacityEnumNet, 71
MC_Card, 78
MCS, 92
MCS_ANY_BUS, 71
MCS_CHANNELTEST_DEVICE, 63
MCS_DEVICE_ANY, 63
MCS_DEVICE_USB, 63
MCS_DEVICE_USB_CYPRESS, 64
MCS_ENCAPSULATOR_DEVICE, 63
MCS_EXTERN_BC_TESTER_DEVICE, 64
MCS_EXTERN_D_TESTER_DEVICE, 64
MCS_FCX_DEVICE, 63
MCS_FYI_DEVICE, 63
MCS_GENERIC_DEVELOPMENT_DEVICE, 63
MCS_HICLAMP_DEVICE, 63
MCS_HLA_DEVICE, 63
MCS_MBC08_DEVICE, 64
MCS_MC_STIMULUS_DEVICE, 63
MCS_MCCARD_DEVICE, 63
MCS_MEA_CLEAN_DEVICE, 64
MCS_MEA_COAT_DEVICE, 64
MCS_MEA_DEVICE, 63
MCS_MEA_IMPEDANCE_DEVICE, 63
MCS_MEA_SWITCH_DEVICE, 63
MCS_MEASURETABLE_DEVICE, 63
MCS_MEAUSB_DEVICE, 63
MCS_NF_GEN_DEVICE, 64
MCS_OCTOPOT_DEVICE, 63
MCS_OKUVISION_STIMULATOR_DEVICE, 64
MCS_old, 92
MCS_PATCHSERVER_DEVICE, 63
MCS_PATHIDENT_DEVICE, 63
MCS_PCI_BUS, 71
MCS_PCX_DEVICE, 63
MCS_PEDOTER_DEVICE, 64
MCS_PERISTALTIC_PUMP_DEVICE, 64
MCS_PGA_DEVICE, 63
MCS_PPC_DEVICE, 64
MCS_PPS5_DEVICE, 64
MCS_PPS_DEVICE, 63
MCS_RETINA_AMS_DONGLE, 63
MCS_RETINA_LED_DEVICE, 63
MCS_ROBO_DEVICE, 63
MCS_ROBOINJECT_DEVICE, 63
MCS_ROBOOCYTE2_DEVICE, 63
MCS_SAFEIS_DEVICE, 64
MCS_SMARTIMPLANT_DEVICE, 64
MCS_SOFTWARE_DONGLE_DEVICE, 64
MCS_STG_DEVICE, 63
MCS_SW2TO64_DEVICE, 63
MCS_TCX_DEVICE, 63
MCS_TERSENS_DEVICE, 63
MCS_UNDEFINED_BUS, 71
MCS_USB_BUS, 71
MCSBUS0, 54
MCSBUS1, 54
MCSBUS10, 54
MCSBUS11, 54
MCSBUS12, 54
MCSBUS13, 54
MCSBUS14, 54
MCSBUS15, 54
MCSBUS2, 54
MCSBUS3, 54
MCSBUS4, 54
MCSBUS5, 54
MCSBUS6, 54
MCSBUS7, 54
MCSBUS8, 54
MCSBUS9, 54

McsBusTypeEnumNet, 71
McsUsbSpeedEnumNet, 72
MCU1, 54
ME128, 79
ME16, 80
ME2100, 80
Me2100_32PICiCE40, 83
Me2100_32PICiCE40Headstage, 68
Me2100_32Xilinx, 83
Me2100_32XilinxHeadstage, 68
Me2100Graphene16_32, 83
Me2100Graphene16_32Headstage, 68
Me2100Interfaceboard, 68
Me2100InvitroSignalCollectorUnit, 68
Me2100InvivoSignalCollectorUnit, 68
Me2100UPA32, 83
Me2100UPA32Headstage, 68
ME256, 79
ME32, 79
ME64, 79
MEA1060, 79
MEA120, 53
MEA2100, 80
Mea2100, 69
MEA2100_256, 80
Mea2100_256, 69
MEA2100_256DacqGroupChannelEnumNet, 72
MEA2100_256DigitalSourceEnumNet, 72
Mea2100_256Headstage, 68
Mea2100_256Interfaceboard, 68
MEA2100_32, 80
MEA2100_Lite, 80
Mea2100_Lite, 69
MEA2100_Mini, 80
MEA2100_Mini_Usb_develop, 80
MEA2100BetaScreen, 80
Mea2100BetaScreen, 83
Mea2100BetaScreenHeadstage, 68
MEA2100DigitalSourceEnumNet, 74
Mea2100Headstage, 68
Mea2100Interfaceboard, 68
Mea2100LiteHeadstage, 68
Mea2100Mini120, 83
Mea2100Mini120Headstage, 68
Mea2100Mini60ECP5, 83
Mea2100Mini60ECP5Headstage, 68
Mea2100Mini60PICiCE40, 83
Mea2100Mini60PICiCE40Headstage, 68
Mea2100MultiwellIFB2, 68
Mea2100STG, 68
MEA252, 53
MEA256, 80
MEA2x32, 53
MEA2x60, 53
MEA32, 53
MEA60, 53
MEA_2_252_2, 53
MEA_2_252_2_6Well, 53
MEA_2_252_2_9Well, 53
MEA_2_252_2_Test, 53
MEA_Clean, 79
MEA_Coat, 79
MEA_Impedance, 79
MEA_Sanofi, 79
MEA_Switch, 79
MEA_Switch_2_1, 79
MEA_Switch_4_2, 79
MeaLayoutEnumNet, 75
MeasureTable, 80
MicroAmpere, 62
MilliDegreeCelsius, 62
mIMEA60, 75
mIUnknown, 75
ModulA_ADC0, 70
ModulA_ADC1, 70
ModulA_ADC2, 70
ModulA_ADC3, 70
ModulA_DAC0, 70
ModulA_DAC1, 70
ModulB_ADC0, 70
ModulB_ADC1, 70
ModulB_ADC2, 70
ModulB_ADC3, 70
ModulB_DAC0, 70
ModulB_DAC1, 70
ModulC_ADC0, 70
ModulC_ADC1, 70
ModulC_ADC2, 70
ModulC_ADC3, 70
ModulC_DAC0, 70
ModulC_DAC1, 70
ModulD_ADC0, 70
ModulD_ADC1, 70
ModulD_ADC2, 70
ModulD_ADC3, 70
ModulD_DAC0, 70
ModulD_DAC1, 70
Monitor, 85
Movement, 82
Multiboot, 80
MultiFileMode, 86
Multiwell, 69, 80
Multiwell96, 53
Multiwell_ICC, 79
Multiwell_MEA_Mini, 80
MultiwellDigitalSourceEnumNet, 75
MultiwellHeadstage, 68
MultiwellInterfaceboard, 68
MultiwellMini, 83
MultiwellMiniHeadstage, 68
MultiwellOptoStim, 79
MultiwellPlateTypeEnumNet, 76
Mux, 64
MuxOtherDevice, 64
Nanion, 88
NanoAmpere, 62

NanoVolt, 62
NCBathCurrent, 58
NCCol2Current, 58
NChipTemperature, 60
Neptun, 81
NeuroChip, 80
NeurochipConfig, 80
NF_Gen, 80
NineWell, 87
nMos16LV, 66
nMos32LV, 66
nMos36LN, 66
nMos64LN, 66
No_Plate, 77
None, 53, 64, 78, 86, 91, 92
Normal, 77
NOT_CONNECTED, 66
NotApplicable, 53
NoUnit, 62
NTC10K, 88
Octopot, 78
Off, 81, 82, 92
off, 81
OK, 66
Okuvision_Stimulator, 79
OnChannelData, 95
OnDeviceArrivalRemoval, 96
One, 57, 73, 75, 76, 84, 88, 95
OnError, 96
OnMcsUsbDeviceState, 96
OnMcsUsbDeviceStateCallback, 96
OnMwPollStatus, 96
OnStg200xDataHandler, 96
OnStg200xErrorHandler, 96
OnStgPollStatus, 96
OnUpdateFirmwareProgress, 97
OnUpdateFirmwareStatusChange, 97
Open, 77
OpenClamp, 92
Output, 77
PacketFrameContextGroup, 62, 72, 84, 94
PatchServAdcModeEnumNet, 77
PatchServer, 80
PathIdent, 80
PC, 95
PCI, 92
PCX, 79
PeriodicPulse, 64
PeristalticPump, 79
PGA, 79
PIC, 55
PIC10, 55
PIC11, 55
PIC12, 55
PIC2, 55
PIC3, 55
PIC4, 55
PIC5, 55
PIC6, 55
PIC7, 55
PIC8, 55
PIC9, 55
PicoAmpere, 62
Plate_24W030MGA, 76
Plate_24W300_30_1152GBA, 76
Plate_24W300_30GBA, 77
Plate_24W300_30GBB, 77
Plate_24W300_30GMA, 76
Plate_24W700_100FMA, 76
Plate_24W700_100FMB, 76
Plate_24W700_100FMC, 76
Plate_24W700_100PBB, 76
Plate_72W500_100FMA, 76
Plate_72W500_100PMA, 76
Plate_96W300_80_1152FMA, 76
Plate_96W400_80_1152FMB, 76
Plate_96W700_100FMA, 76
Plate_96W700_100FMB, 76
Plate_96W700_100GBA, 77
Plate_96W700_100GBB, 77
Plate_96W700_100GBC, 76
Plate_96W700_100GBD, 76
Plate_96W700_100GMA, 76
Plate_96W700_100PBB, 76
Plate_Dummy, 76
Plate_Dummy_126, 76
Plate_ICB8_24W700_100P8A, 76
Plate_ICB8_96W700_100P8A, 76
PlateClampEnumNet, 77
PlateClampLockEnumNet, 77
PlusMinus10Volts, 53
PlusMinus2Comma5Volts, 53
PlusMinus5Volts, 53
PortDirectionEnumNet, 77
Pos900, 81
PositionBase, 80
PositionIIBase, 80
PositionIICentralUnit, 80
PositionImp, 80
PostCommaA, 67
PostCommaB, 67
PP_Pump_Mode_Type_EnumNet, 78
PPC, 79
PPS2, 79
PPS4plus1, 79
PPS5, 79
PPS5_DIG, 79
PreCommaA, 67
PreCommaB, 67
ProductIdEnumNet, 78
PT100, 88
PT1000, 88
PulseGenerator, 58, 73, 75, 76, 84, 88, 95
PulseGenerator_Mode_EnumNet, 81
rawdata, 87
RC, 67

rc100mAh, 71
rc200mAh, 71
rc300mAh, 71
rc30mAh, 71
rcGreater300mAh, 71
Rectangle, 89
Ref16, 81
Ref24, 81
Ref32, 81
Ref8, 81
Reference, 82
ReferenceElectrodeModeEnumNet, 81
ReferenceElectrodeSwitchPositionEnumNet, 81
RegisterHigh, 64
RegisterLow, 64
Regular, 88
Renesas, 91
Reserved1, 88
Reserved2, 88
Reserved3, 88
Reserved4, 88
Reserved5, 88
Restart, 82
Retina_LED, 79
RetriggerActionEnumNet, 82
Rising, 65
RoboCurrentModeEnumNet, 82
RoboInject, 80
Robooycte2, 80
RoboStatusEventDelegate, 97
Running, 86
SafeIS, 80
SampleDstSize16, 82
SampleDstSize32, 82
SampleDstSizeNet, 82
SampleSize16Signed, 82
SampleSize16Unsigned, 82
SampleSize24Signed, 83
SampleSize24Unsigned, 82
SampleSize32Signed, 83
SampleSize32Unsigned, 83
SampleSize64Signed, 83
SampleSize64Unsigned, 83
SampleSizeNet, 82
SBSVector1, 60
SBSVector2, 60
SBSVector3, 60
SBSVector4, 60
SCU1ElectrodeGroupHS1, 83
SCU1ElectrodeGroupHS2, 83
SCU1ElectrodeGroupHS3, 83
SCU1ElectrodeGroupHS4, 83
SCU1HS1Sideband1, 84
SCU1HS1Sideband1TriggerStatus, 84
SCU1HS1Sideband2, 84
SCU1HS1Sideband2TriggerStatus, 84
SCU1HS2Sideband1, 84
SCU1HS2Sideband1TriggerStatus, 84
SCU1HS2Sideband2, 84
SCU1HS2Sideband2TriggerStatus, 84
SCU1HS3Sideband1, 84
SCU1HS3Sideband1TriggerStatus, 84
SCU1HS3Sideband2, 84
SCU1HS3Sideband2TriggerStatus, 84
SCU1HS4Sideband1, 85
SCU1HS4Sideband1TriggerStatus, 84
SCU1HS4Sideband2, 85
SCU1HS4Sideband2TriggerStatus, 84
SCU1LEDStim1, 85
SCU1LEDStim1TriggerStatus, 84
SCU1LEDStim2, 85
SCU1LEDStim2TriggerStatus, 84
SCU1LEDStim3, 85
SCU1LEDStim3TriggerStatus, 84
SCU1LEDStim4, 85
SCU1LEDStim4TriggerStatus, 84
SCU2ElectrodeGroupHS1, 83
SCU2ElectrodeGroupHS2, 83
SCU2ElectrodeGroupHS3, 83
SCU2ElectrodeGroupHS4, 83
SCU2HS1Sideband1, 85
SCU2HS1Sideband1TriggerStatus, 85
SCU2HS1Sideband2, 85
SCU2HS1Sideband2TriggerStatus, 85
SCU2HS2Sideband1, 85
SCU2HS2Sideband1TriggerStatus, 85
SCU2HS2Sideband2, 85
SCU2HS2Sideband2TriggerStatus, 85
SCU2HS3Sideband1, 85
SCU2HS3Sideband1TriggerStatus, 85
SCU2HS3Sideband2, 85
SCU2HS3Sideband2TriggerStatus, 85
SCU2HS4Sideband1, 85
SCU2HS4Sideband1TriggerStatus, 85
SCU2HS4Sideband2, 85
SCU2HS4Sideband2TriggerStatus, 85
SCU2LEDStim1, 85
SCU2LEDStim1TriggerStatus, 85
SCU2LEDStim2, 85
SCU2LEDStim2TriggerStatus, 85
SCU2LEDStim3, 85
SCU2LEDStim3TriggerStatus, 85
SCU2LEDStim4, 85
SCU2LEDStim4TriggerStatus, 85
SCU_HeadstageIdEnumNet, 83
SCUDacqGroupChannelEnumNet, 83
SCUDigitalSourceEnumNet, 84
Settings, 95
Signed_16bit, 63
Signed_24bit, 63
Signed_32bit, 63
Sine, 89
Single, 82
SingleWell, 87
SixWell, 87
SmartImplant, 80

SOFAndCTRLword, 61
Software, 68
SoftwareDongle, 80
Standby, 82, 92
Start, 65
State, 92
STG, 78
Stg1, 66
STG1DACSignalGroup, 62, 72, 83
STG1SidebandsGroup, 62, 72, 84
STG1TriggerStatusGroup, 62, 72, 84
Stg2, 66
Stg200xDigoutModeEnumNet, 85
Stg200xMultiFileSubmodeEnumNet, 86
Stg200xSegmentFlagsEnumNet, 86
Stg200xTriggerStatusEnumNet, 86
STG2DACSignalGroup, 72, 83
STG2SidebandsGroup, 72, 84
STG2TriggerStatusGroup, 72, 84
Stg3, 66
STG3008_FA, 79
Stg3008FilterAmpAmplificationEnumNet, 86
STG4002, 78
STG4002_opto, 79
STG4004, 78
STG4004_opto, 79
STG4008, 78
STG4008_opto, 79
STG400x, 78
STG400x_opto, 79
STG5, 79
STG_DestinationEnumNet, 87
StgListModeTrigger, 65
StgTrigger, 65
Stimulation, 58
StimulationLayoutConfigurationEnumNet, 87
Stop, 65, 77, 82
StorageCharge, 71
SubtractFromAll, 81
SubtractFromAllOther, 81
SubtractFromReferenceElectrodeOnly, 81
SubtractionOff, 81
SuperSpeed, 72
Sw2to64, 79
SYNCOUT1, 85
SYNCOUT2, 85
SYNCOUT3, 85
SYNCOUT4, 85
SYNCOUT5, 85
SYNCOUT6, 85
SYNCOUT7, 85
SYNCOUT8, 85
syncoutdata, 87
SyncStart, 86
Table, 95
TBSI_127, 53
TBSI_15, 53
TBSI_31, 53
TBSI_5, 53
TBSI_63, 53
TBSI_Dacq, 80
TBSI_DACQDigitalSourceEnumNet, 87
TBSI_Reserved, 53
TbsiDacq, 69
TbsiDacqHeadstage, 68
TbsiDacqInterfaceboard, 68
TC01, 79
TC02, 79
TCX, 79
TcxDeviceTypeEnumNet, 88
TcxSensorTypeEnumNet, 88
TeerClampModeEnumNet, 88
TeerWaveformEnumNet, 89
Tersens, 78
Test_ADC_EPC10, 70
Test_DAC_EPC10, 70
Timestamp, 61
TouchTest, 95
Triggerbox_AMS, 79
Triggerbox_AMS3, 79
Triggerbox_IMS, 79
Triggerbox_R5, 79
TriggerOnly, 86
TriggerSourceEnumNet, 89
TriggerStatus1, 60
TriggerStatus2, 60
TriggerStatus3, 60
TriggerStatus4, 60
tsAuxIn1, 90
tsAuxIn2, 90
tsDACQCy1Dev1Runs, 91
tsDACQCy1Dev2Runs, 91
tsDACQCy2Dev1Runs, 91
tsDACQCy2Dev2Runs, 91
tsDigitalIn1, 89
tsDigitalIn10, 89
tsDigitalIn11, 89
tsDigitalIn12, 89
tsDigitalIn13, 89
tsDigitalIn14, 89
tsDigitalIn15, 89
tsDigitalIn16, 89
tsDigitalIn17, 89
tsDigitalIn18, 89
tsDigitalIn19, 89
tsDigitalIn2, 89
tsDigitalIn20, 89
tsDigitalIn21, 89
tsDigitalIn22, 90
tsDigitalIn23, 90
tsDigitalIn24, 90
tsDigitalIn25, 90
tsDigitalIn26, 90
tsDigitalIn27, 90
tsDigitalIn28, 90
tsDigitalIn29, 90

tsDigitalIn3, 89
tsDigitalIn30, 90
tsDigitalIn31, 90
tsDigitalIn32, 90
tsDigitalIn4, 89
tsDigitalIn5, 89
tsDigitalIn6, 89
tsDigitalIn7, 89
tsDigitalIn8, 89
tsDigitalIn9, 89
tsDigitalPuse0, 90
tsDigitalPuse1, 90
tsDigitalPuse10, 91
tsDigitalPuse11, 91
tsDigitalPuse12, 91
tsDigitalPuse13, 91
tsDigitalPuse14, 91
tsDigitalPuse15, 91
tsDigitalPuse16, 91
tsDigitalPuse17, 91
tsDigitalPuse18, 91
tsDigitalPuse19, 91
tsDigitalPuse2, 90
tsDigitalPuse20, 91
tsDigitalPuse21, 91
tsDigitalPuse22, 91
tsDigitalPuse23, 91
tsDigitalPuse24, 91
tsDigitalPuse25, 91
tsDigitalPuse26, 91
tsDigitalPuse27, 91
tsDigitalPuse28, 91
tsDigitalPuse29, 91
tsDigitalPuse3, 90
tsDigitalPuse30, 91
tsDigitalPuse31, 91
tsDigitalPuse4, 90
tsDigitalPuse5, 91
tsDigitalPuse6, 91
tsDigitalPuse7, 91
tsDigitalPuse8, 91
tsDigitalPuse9, 91
tsFeedback1, 90
tsFeedback10, 90
tsFeedback11, 90
tsFeedback12, 90
tsFeedback13, 90
tsFeedback14, 90
tsFeedback15, 90
tsFeedback16, 90
tsFeedback17, 90
tsFeedback18, 90
tsFeedback19, 90
tsFeedback2, 90
tsFeedback20, 90
tsFeedback21, 90
tsFeedback22, 90
tsFeedback23, 90
tsFeedback24, 90
tsFeedback25, 90
tsFeedback26, 90
tsFeedback27, 90
tsFeedback28, 90
tsFeedback29, 90
tsFeedback3, 90
tsFeedback30, 90
tsFeedback31, 90
tsFeedback32, 90
tsFeedback4, 90
tsFeedback5, 90
tsFeedback6, 90
tsFeedback7, 90
tsFeedback8, 90
tsFeedback9, 90
tsNone, 89
tsSidebandBit8, 91
tsTriggered, 91
Unknown, 53, 54, 67, 88, 91
unknown, 66
UnknownDest, 57
UnknownSpeed, 72
Unlock, 77
Unsigned_16bit, 63
Unsigned_24bit, 63
Unsigned_32bit, 63
UpdateTrigger, 86
USB, 54
USB_TARGET1, 57
USB_TARGET2, 57
USB_TARGET3, 57
UsbTest, 80
UsbVendorIdEnumNet, 91
User_ADC_0, 69
User_ADC_1, 69
User_ADC_2, 70
User_ADC_3, 70
User_ADC_4, 70
User_DAC_0, 70
User_DAC_1, 70
User_DAC_2, 70
UssingChamber, 69
UssingClampModeEnumNet, 92
UssingRail, 69
UssingUnitEnumNet, 92
VendorIdEnumNet, 92
Volt, 54, 92
Voltage_3V3, 69
Voltage_5V0, 69
VoltageClamp, 92
W2100, 80
W2100_Accel_Gyro_Select_EnumNet, 92
W2100DacqGroupChannelEnumNet, 93
W2100DigitalSourceEnumNet, 94
W2100IFB2, 68
W2100Interfaceboard, 68
W2100WirelessReceiver, 68, 69

W2100WirelessReceiverAnalog, 68, 69
Warner, 88
Warner_TEER_Machine, 81
Warner_Ussing, 81
WARNER_USSING_DEVICE, 64
Warner_Valve_Control, 81
WARNER_VALVE_CONTROL_DEVICE, 64
Whole_Cell_Patch, 80
WholeCellPatch, 83
WholeCellPatchHeadstage, 68
WirelessHeadStageAccDataRE1HS1, 93
WirelessHeadStageAccDataRE1HS2, 93
WirelessHeadStageAccDataRE1HS3, 93
WirelessHeadStageAccDataRE1HS4, 93
WirelessHeadStageAccDataRE2HS1, 93
WirelessHeadStageAccDataRE2HS2, 93
WirelessHeadStageAccDataRE2HS3, 93
WirelessHeadStageAccDataRE2HS4, 93
WirelessHeadStageAnalogRE1HS1, 93
WirelessHeadStageAnalogRE1HS2, 93
WirelessHeadStageAnalogRE1HS3, 93
WirelessHeadStageAnalogRE1HS4, 93
WirelessHeadStageAnalogRE2HS1, 93
WirelessHeadStageAnalogRE2HS2, 93
WirelessHeadStageAnalogRE2HS3, 93
WirelessHeadStageAnalogRE2HS4, 93
WirelessHeadStageGyroDataRE1HS1, 93
WirelessHeadStageGyroDataRE1HS2, 93
WirelessHeadStageGyroDataRE1HS3, 93
WirelessHeadStageGyroDataRE1HS4, 93
WirelessHeadStageGyroDataRE2HS1, 93
WirelessHeadStageGyroDataRE2HS2, 93
WirelessHeadStageGyroDataRE2HS3, 93
WirelessHeadStageGyroDataRE2HS4, 93
WirelessHeadStageOptoStimCurrentRE1HS1, 94
WirelessHeadStageOptoStimCurrentRE1HS2, 94
WirelessHeadStageOptoStimCurrentRE1HS3, 94
WirelessHeadStageOptoStimCurrentRE1HS4, 94
WirelessHeadStageOptoStimCurrentRE2HS1, 94
WirelessHeadStageOptoStimCurrentRE2HS2, 94
WirelessHeadStageOptoStimCurrentRE2HS3, 94
WirelessHeadStageOptoStimCurrentRE2HS4, 94
WirelessHeadStageReservedARE1HS1, 94
WirelessHeadStageReservedARE1HS2, 94
WirelessHeadStageReservedARE1HS3, 94
WirelessHeadStageReservedARE1HS4, 94
WirelessHeadStageReservedARE2HS1, 94
WirelessHeadStageReservedARE2HS2, 94
WirelessHeadStageReservedARE2HS3, 94
WirelessHeadStageReservedARE2HS4, 94
WirelessHeadStageReservedBRE1HS1, 94
WirelessHeadStageReservedBRE1HS2, 94
WirelessHeadStageReservedBRE1HS3, 94
WirelessHeadStageReservedBRE1HS4, 94
WirelessHeadStageReservedBRE2HS1, 94
WirelessHeadStageReservedBRE2HS2, 94
WirelessHeadStageReservedBRE2HS3, 94
WirelessHeadStageReservedBRE2HS4, 94
WirelessHeadStageReservedCRE1HS1, 94
WirelessHeadStageReservedCRE1HS2, 94
WirelessHeadStageReservedCRE1HS3, 94
WirelessHeadStageReservedCRE1HS4, 94
WirelessHeadStageReservedCRE2HS1, 94
WirelessHeadStageReservedCRE2HS2, 94
WirelessHeadStageReservedCRE2HS3, 94
WirelessHeadStageReservedCRE2HS4, 94
WirelessHeadStageStatusRE1HS1, 93
WirelessHeadStageStatusRE1HS2, 93
WirelessHeadStageStatusRE1HS3, 93
WirelessHeadStageStatusRE1HS4, 93
WirelessHeadStageStatusRE2HS1, 93
WirelessHeadStageStatusRE2HS2, 93
WirelessHeadStageStatusRE2HS3, 93
WirelessHeadStageStatusRE2HS4, 93
WirelessTestAdapter, 53
Work, 95
WPA16, 80
WPA32, 80
WPA4, 80
WPA8, 80
WvcDisplayModeEnumNet, 95
WvcValveModeEnumNet, 95
Zero, 57, 73, 74, 76, 84, 88, 95
MCS_ANY_BUS
Mcs::Usb, 71
MCS_CHANNELTEST_DEVICE
Mcs::Usb, 63
MCS_DEVICE_ANY
Mcs::Usb, 63
MCS_DEVICE_USB
Mcs::Usb, 63
MCS_DEVICE_USB_CYPRESS
Mcs::Usb, 64
MCS_ENCAPSULATOR_DEVICE
Mcs::Usb, 63
MCS_EXTERN_BC_TESTER_DEVICE
Mcs::Usb, 64
MCS_EXTERN_D_TESTER_DEVICE
Mcs::Usb, 64
MCS_FCX_DEVICE
Mcs::Usb, 63
MCS_FYI_DEVICE
Mcs::Usb, 63
MCS_GENERIC_DEVELOPMENT_DEVICE
Mcs::Usb, 63
MCS_HICLAMP_DEVICE
Mcs::Usb, 63
MCS_HLA_DEVICE
Mcs::Usb, 63
MCS_MBC08_DEVICE
Mcs::Usb, 64
MCS_MC_STIMULUS_DEVICE
Mcs::Usb, 63
MCS_MCCARD_DEVICE
Mcs::Usb, 63
MCS_MEA_CLEAN_DEVICE

Mcs::Usb, 64
MCS_MEA_COAT_DEVICE
Mcs::Usb, 64
MCS_MEA_DEVICE
Mcs::Usb, 63
MCS_MEA_IMPEDANCE_DEVICE
Mcs::Usb, 63
MCS_MEA_SWITCH_DEVICE
Mcs::Usb, 63
MCS_MEASURETABLE_DEVICE
Mcs::Usb, 63
MCS_MEAUSB_DEVICE
Mcs::Usb, 63
MCS_NF_GEN_DEVICE
Mcs::Usb, 64
MCS_OCTOPOT_DEVICE
Mcs::Usb, 63
MCS_OKUVISION_STIMULATOR_DEVICE
Mcs::Usb, 64
MCS_old
Mcs::Usb, 92
MCS_PATCHSERVER_DEVICE
Mcs::Usb, 63
MCS_PATHIDENT_DEVICE
Mcs::Usb, 63
MCS_PCI_BUS
Mcs::Usb, 71
MCS_PCX_DEVICE
Mcs::Usb, 63
MCS_PEDOTER_DEVICE
Mcs::Usb, 64
MCS_PERISTALTIC_PUMP_DEVICE
Mcs::Usb, 64
MCS_PGA_DEVICE
Mcs::Usb, 63
MCS_PPC_DEVICE
Mcs::Usb, 64
MCS_PPS5_DEVICE
Mcs::Usb, 64
MCS_PPS_DEVICE
Mcs::Usb, 63
MCS_RETINA_AMS_DONGLE
Mcs::Usb, 63
MCS_RETINA_LED_DEVICE
Mcs::Usb, 63
MCS_ROBO_DEVICE
Mcs::Usb, 63
MCS_ROBOINJECT_DEVICE
Mcs::Usb, 63
MCS_ROBOOCYTE2_DEVICE
Mcs::Usb, 63
MCS_SAFEIS_DEVICE
Mcs::Usb, 64
MCS_SMARTIMPLANT_DEVICE
Mcs::Usb, 64
MCS_SOFTWARE_DONGLE_DEVICE
Mcs::Usb, 64
MCS_STG_DEVICE
Mcs::Usb, 63
MCS_SW2TO64_DEVICE
Mcs::Usb, 63
MCS_TCX_DEVICE
Mcs::Usb, 63
MCS_TERSENS_DEVICE
Mcs::Usb, 63
MCS_UNDEFINED_BUS
Mcs::Usb, 71
MCS_USB_BUS
Mcs::Usb, 71
McsBus
CPPCDeviceNet, 429
CPPS_DeviceNet, 438
CRoboDeviceNet, 491
MCSBUS0
Mcs::Usb, 54
MCSBUS1
FirmwareDestinationNames, 699
Mcs::Usb, 54
MCSBUS10
FirmwareDestinationNames, 699
Mcs::Usb, 54
MCSBUS11
FirmwareDestinationNames, 699
Mcs::Usb, 54
MCSBUS12
FirmwareDestinationNames, 699
Mcs::Usb, 54
MCSBUS13
FirmwareDestinationNames, 699
Mcs::Usb, 54
MCSBUS14
Mcs::Usb, 54
MCSBUS15
Mcs::Usb, 54
MCSBUS2
FirmwareDestinationNames, 699
Mcs::Usb, 54
MCSBUS3
FirmwareDestinationNames, 699
Mcs::Usb, 54
MCSBUS4
FirmwareDestinationNames, 699
Mcs::Usb, 54
MCSBUS5
FirmwareDestinationNames, 699
Mcs::Usb, 54
MCSBUS6
FirmwareDestinationNames, 699
Mcs::Usb, 54
MCSBUS7
FirmwareDestinationNames, 699
Mcs::Usb, 54
MCSBUS8
FirmwareDestinationNames, 700
Mcs::Usb, 54
MCSBUS9

FirmwareDestinationNames, 700
Mcs::Usb, 54
McsBus_MotorControl
CPeristalticPumpDeviceNet, 414
CPPCDeviceNet, 429
CPPS_DeviceNet, 438
CRoboDeviceNet, 491
CRoboFluidDeviceNet, 495
McsBus_Sensor
CPPCDeviceNet, 429
CPPS_DeviceNet, 438
McsBus_VoltageMode
CFluidControlDeviceNet, 154
McsBus_XY
CRoboDeviceNet, 488
McsBus_ZI
CRoboDeviceNet, 488
McsBusTypeEnumNet
Mcs::Usb, 71
McsUsbDeviceStateEvent
CMcsUsbDeviceStatePushFunctionNet, 303
CMcsUsbDeviceStatePushNet, 304
McsUsbSpeedEnumNet
Mcs::Usb, 72
MCU1
FirmwareDestinationNames, 700
Mcs::Usb, 54
ME128
Mcs::Usb, 79
ME16
Mcs::Usb, 80
ME2100
Mcs::Usb, 80
Me2100_32PICiCE40
Mcs::Usb, 83
Me2100_32PICiCE40Headstage
Mcs::Usb, 68
Me2100_32Xilinx
Mcs::Usb, 83
Me2100_32XilinxHeadstage
Mcs::Usb, 68
Me2100Graphene16_32
Mcs::Usb, 83
Me2100Graphene16_32Headstage
Mcs::Usb, 68
Me2100Interfaceboard
Mcs::Usb, 68
Me2100InvitroSignalCollectorUnit
Mcs::Usb, 68
Me2100InvivoSignalCollectorUnit
Mcs::Usb, 68
Me2100UPA32
Mcs::Usb, 83
Me2100UPA32Headstage
Mcs::Usb, 68
ME256
Mcs::Usb, 79
ME32
Mcs::Usb, 79
ME64
Mcs::Usb, 79
MEA1060
Mcs::Usb, 79
MEA120
Mcs::Usb, 53
MEA2100
Mcs::Usb, 80
Mea2100
Mcs::Usb, 69
MEA2100_256
Mcs::Usb, 80
Mea2100_256
Mcs::Usb, 69
MEA2100_256DacqGroupChannelEnumNet
Mcs::Usb, 72
MEA2100_256DigitalSourceEnumNet
Mcs::Usb, 72
Mea2100_256Headstage
Mcs::Usb, 68
Mea2100_256Interfaceboard
Mcs::Usb, 68
MEA2100_32
Mcs::Usb, 80
MEA2100_Lite
Mcs::Usb, 80
Mea2100_Lite
Mcs::Usb, 69
MEA2100_Mini
Mcs::Usb, 80
MEA2100_Mini_Usb_develop
Mcs::Usb, 80
MEA2100BetaScreen
Mcs::Usb, 80
Mea2100BetaScreen
Mcs::Usb, 83
Mea2100BetaScreenHeadstage
Mcs::Usb, 68
MEA2100DigitalSourceEnumNet
Mcs::Usb, 74
Mea2100Headstage
Mcs::Usb, 68
Mea2100Interfaceboard
Mcs::Usb, 68
Mea2100LiteHeadstage
Mcs::Usb, 68
Mea2100Mini120
Mcs::Usb, 83
Mea2100Mini120Headstage
Mcs::Usb, 68
Mea2100Mini60ECP5
Mcs::Usb, 83
Mea2100Mini60ECP5Headstage
Mcs::Usb, 68
Mea2100Mini60PICiCE40
Mcs::Usb, 83
Mea2100Mini60PICiCE40Headstage

Mcs::Usb, 68
Mea2100MultiwellIFB2
Mcs::Usb, 68
Mea2100STG
Mcs::Usb, 68
MEA252
Mcs::Usb, 53
MEA256
Mcs::Usb, 80
MEA2x32
Mcs::Usb, 53
MEA2x60
Mcs::Usb, 53
MEA32
Mcs::Usb, 53
MEA60
Mcs::Usb, 53
MEA_2_252_2
Mcs::Usb, 53
MEA_2_252_2_6Well
Mcs::Usb, 53
MEA_2_252_2_9Well
Mcs::Usb, 53
MEA_2_252_2_Test
Mcs::Usb, 53
MEA_Clean
Mcs::Usb, 79
MEA_Coat
Mcs::Usb, 79
MEA_Impedance
Mcs::Usb, 79
MEA_Sanofi
Mcs::Usb, 79
MEA_Switch
Mcs::Usb, 79
MEA_Switch_2_1
Mcs::Usb, 79
MEA_Switch_4_2
Mcs::Usb, 79
MeaAudioFunctionNet
CMeaDeviceNet, 366
MeaDigitalDataFunctionNet
CMeaDeviceNet, 366
MeaFeedbackFunctionNet
CMeaDeviceNet, 366
MeaLayoutEnumNet
Mcs::Usb, 75
Measure
CPathIdentDeviceNet, 411
MeasureReservoir
CPPCFunctionNet, 436
MeasureTable
Mcs::Usb, 80
MeasuringOnly
HeadStageIDType, 701
MeFunctionNet
CMeaDeviceNet, 366
MicroAmpere
Mcs::Usb, 62
MilliDegreeCelsius
Mcs::Usb, 62
mkfilter
mkfilterNet, 706
mkfilter_coef_in_one_set
mkfilterNet, 706
mkfilter_highpass_coeff
mkfilterNet, 707
mkfilter_highpass_frequency_from_coeff
mkfilterNet, 707
mkfilter_highpass_frequency_from_k
mkfilterNet, 707
mkfilter_highpass_k
mkfilterNet, 707
mkfilter_MCS
mkfilterNet, 707
mkfilter_MCS_k
mkfilterNet, 707, 708
mkfilter_normalize_coeffs_int
mkfilterNet, 708
mkfilter_normalize_coeffs_short
mkfilterNet, 708
mkfilter_normalize_scale_coeffs_int
mkfilterNet, 708
mkfilter_scale_coef_in_one_set
mkfilterNet, 708
mkfilterNet, 706
mkfilter, 706
mkfilter_coef_in_one_set, 706
mkfilter_highpass_coeff, 707
mkfilter_highpass_frequency_from_coeff, 707
mkfilter_highpass_frequency_from_k, 707
mkfilter_highpass_k, 707
mkfilter_MCS, 707
mkfilter_MCS_k, 707, 708
mkfilter_normalize_coeffs_int, 708
mkfilter_normalize_coeffs_short, 708
mkfilter_normalize_scale_coeffs_int, 708
mkfilter_scale_coef_in_one_set, 708
mlMEA60
Mcs::Usb, 75
mlUnknown
Mcs::Usb, 75
ModulA_ADC0
Mcs::Usb, 70
ModulA_ADC1
Mcs::Usb, 70
ModulA_ADC2
Mcs::Usb, 70
ModulA_ADC3
Mcs::Usb, 70
ModulA_DAC0
Mcs::Usb, 70
ModulA_DAC1
Mcs::Usb, 70
ModulB_ADC0
Mcs::Usb, 70

ModulB_ADC1
Mcs::Usb, 70
ModulB_ADC2
Mcs::Usb, 70
ModulB_ADC3
Mcs::Usb, 70
ModulB_DAC0
Mcs::Usb, 70
ModulB_DAC1
Mcs::Usb, 70
ModulC_ADC0
Mcs::Usb, 70
ModulC_ADC1
Mcs::Usb, 70
ModulC_ADC2
Mcs::Usb, 70
ModulC_ADC3
Mcs::Usb, 70
ModulC_DAC0
Mcs::Usb, 70
ModulC_DAC1
Mcs::Usb, 70
ModulD_ADC0
Mcs::Usb, 70
ModulD_ADC1
Mcs::Usb, 70
ModulD_ADC2
Mcs::Usb, 70
ModulD_ADC3
Mcs::Usb, 70
ModulD_DAC0
Mcs::Usb, 70
ModulD_DAC1
Mcs::Usb, 70
Monitor
Mcs::Usb, 85
MoveAbs
CRoboDeviceNet, 483, 484
MoveAbsI
CRoboStatorDeviceNet, 500
MoveAbsXY
CRoboStatorDeviceNet, 500
MoveAbsZ
CRoboStatorDeviceNet, 500, 501
Movement
Mcs::Usb, 82
Multiboot
Mcs::Usb, 80
MultibootGetCypressImageId
CMcsUsbNet, 335
MultibootGetImageId
CMcsUsbNet, 335
MultibootGetSelectedImage
CMcsUsbNet, 335
MultibootSelectImage
CMcsUsbNet, 335
MultiFileMode
Mcs::Usb, 86
Multiwell
Mcs::Usb, 69, 80
Multiwell96
Mcs::Usb, 53
Multiwell_ICC
Mcs::Usb, 79
Multiwell_MEA_Mini
Mcs::Usb, 80
MultiwellDigitalSourceEnumNet
Mcs::Usb, 75
MultiwellHeadstage
Mcs::Usb, 68
MultiwellInterfaceboard
Mcs::Usb, 68
MultiwellMini
Mcs::Usb, 83
MultiwellMiniHeadstage
Mcs::Usb, 68
MultiwellOptoStim
Mcs::Usb, 79
MultiwellPlateTypeEnumNet
Mcs::Usb, 76
Mux
Mcs::Usb, 64
MuxOtherDevice
Mcs::Usb, 64
MwPollStatusEvent
CStg200xDownloadNet, 579

Nanion
Mcs::Usb, 88
NanoAmpere
Mcs::Usb, 62
NanoVolt
Mcs::Usb, 62
NCBathCurrent
Mcs::Usb, 58
NCCol2Current
Mcs::Usb, 58
NChipTemperature
Mcs::Usb, 60
Neptun
Mcs::Usb, 81
NeuroChip
Mcs::Usb, 80
NeurochipConfig
Mcs::Usb, 80
NF_Gen
Mcs::Usb, 80
NineWell
Mcs::Usb, 87
nMos16LV
Mcs::Usb, 66
nMos32LV
Mcs::Usb, 66
nMos36LN
Mcs::Usb, 66
nMos64LN
Mcs::Usb, 66

No_Plate
Mcs::Usb, 77
None
Mcs::Usb, 53, 64, 78, 86, 91, 92
Normal
Mcs::Usb, 77
NOT_CONNECTED
Mcs::Usb, 66
NotApplicable
Mcs::Usb, 53
NoUnit
Mcs::Usb, 62
NTC10K
Mcs::Usb, 88
NullCommand
CRoboDeviceNet, 485
NumberOfAnalogChannels
HeadStageIDType, 702
NumberOfChannels
CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumTemplateNet >, 131
NumberOfStimulationChannels
HeadStageIDType, 703
NumCoefSets
CCreateFilterNet, 122

Octopot
Mcs::Usb, 78
Off
Mcs::Usb, 81, 82, 92
off
Mcs::Usb, 81
OK
Mcs::Usb, 66
Okuvision_Stimulator
Mcs::Usb, 79
OnChannelData
Mcs::Usb, 95
OnDeviceArrivalRemoval
Mcs::Usb, 96
One
Mcs::Usb, 57, 73, 75, 76, 84, 88, 95
OnError
Mcs::Usb, 96
OnGetActiveRunningTableNumber
CWarnerValveControllerDeviceNet, 667
OnGetAnalogThresholdHigh
CWarnerValveControllerDeviceNet, 667
OnGetAnalogThresholdLow
CWarnerValveControllerDeviceNet, 667
OnGetAnalogVoltage
CWarnerValveControllerDeviceNet, 667
OnGetAvailableHeadstages
CSCUFunctionNet, 520
OnGetCurrentNumberOfValves
CWarnerValveControllerDeviceNet, 667
OnGetDigitalOutPortValve
CWarnerValveControllerDeviceNet, 667
OnGetDigitalPortDirection
CWarnerValveControllerDeviceNet, 667
OnGetDisplayMode
CWarnerValveControllerDeviceNet, 667
OnGetPlateClampStateByHeadstage
CMultiwellCallbackFunctionNet, 388
OnGetTableNamebyIndex
CWarnerValveControllerDeviceNet, 667
OnGetValveActive
CWarnerValveControllerDeviceNet, 668
OnGetValveBoardRevision
CWarnerValveControllerDeviceNet, 668
OnGetValveDigitalInPort
CWarnerValveControllerDeviceNet, 668
OnGetValveLedOn
CWarnerValveControllerDeviceNet, 668
OnGetValveManualGroup
CWarnerValveControllerDeviceNet, 668
OnGetValveManualState
CWarnerValveControllerDeviceNet, 668
OnGetValveMode
CWarnerValveControllerDeviceNet, 668
OnIsDigitalOutPortInverted
CWarnerValveControllerDeviceNet, 668
OnIsHeadstageAvailable
CSCUFunctionNet, 520
OnIsValveDigitalInInverted
CWarnerValveControllerDeviceNet, 669
OnIsValveOpen
CWarnerValveControllerDeviceNet, 669
OnIsValveOpenInAnalogMode
CWarnerValveControllerDeviceNet, 669
OnIsValveOpenInDigitalMode
CWarnerValveControllerDeviceNet, 669
OnMcsUsbDeviceState
Mcs::Usb, 96
OnMcsUsbDeviceStateCallback
Mcs::Usb, 96
OnMwPollStatus
Mcs::Usb, 96
OnStg200xDataHandler
Mcs::Usb, 96
OnStg200xErrorHandler
Mcs::Usb, 96
OnStgPollStatus
Mcs::Usb, 96
OnTableEntryChanged
CWarnerValveControllerDeviceNet, 669
OnUpdateFirmwareProgress
Mcs::Usb, 97
OnUpdateFirmwareStatusChange
Mcs::Usb, 97
Open
Mcs::Usb, 77
OpenClamp
Mcs::Usb, 92
OpenPipe
CGenericDevelopDeviceNet, 167
OpenPlateClamp

CMultiwellDeviceNet, 394
operator=
DeviceIdNet, 688
OpticalStimulation
HeadStageIDType, 701
Order
CCreateFilterNet, 122
CFilterPropertyNet, 147
Output
Mcs::Usb, 77

PacketFrameContextGroup
Mcs::Usb, 62, 72, 84, 94
PatchServAdcModeEnumNet
Mcs::Usb, 77
PatchServer
Mcs::Usb, 80
PathIdent
Mcs::Usb, 80
PatternListStart
COctoPotDeviceNet, 404
PC
Mcs::Usb, 95
PCI
Mcs::Usb, 92
PCX
Mcs::Usb, 79
PeriodicPulse
Mcs::Usb, 64
PeristalticPump
Mcs::Usb, 79
PGA
Mcs::Usb, 79
PIC
FirmwareDestinationNames, 700
Mcs::Usb, 55
PIC10
Mcs::Usb, 55
PIC11
Mcs::Usb, 55
PIC12
Mcs::Usb, 55
PIC2
FirmwareDestinationNames, 700
Mcs::Usb, 55
PIC3
FirmwareDestinationNames, 700
Mcs::Usb, 55
PIC4
FirmwareDestinationNames, 700
Mcs::Usb, 55
PIC5
Mcs::Usb, 55
PIC6
Mcs::Usb, 55
PIC7
Mcs::Usb, 55
PIC8
Mcs::Usb, 55
PIC9
Mcs::Usb, 55
PicoAmpere
Mcs::Usb, 62
Plate_24W030MGA
Mcs::Usb, 76
Plate_24W300_30_1152GBA
Mcs::Usb, 76
Plate_24W300_30GBA
Mcs::Usb, 77
Plate_24W300_30GBB
Mcs::Usb, 77
Plate_24W300_30GMA
Mcs::Usb, 76
Plate_24W700_100FMA
Mcs::Usb, 76
Plate_24W700_100FMB
Mcs::Usb, 76
Plate_24W700_100FMC
Mcs::Usb, 76
Plate_24W700_100PBB
Mcs::Usb, 76
Plate_72W500_100FMA
Mcs::Usb, 76
Plate_72W500_100PMA
Mcs::Usb, 76
Plate_96W300_80_1152FMA
Mcs::Usb, 76
Plate_96W400_80_1152FMB
Mcs::Usb, 76
Plate_96W700_100FMA
Mcs::Usb, 76
Plate_96W700_100FMB
Mcs::Usb, 76
Plate_96W700_100GBA
Mcs::Usb, 77
Plate_96W700_100GBB
Mcs::Usb, 77
Plate_96W700_100GBC
Mcs::Usb, 76
Plate_96W700_100GBD
Mcs::Usb, 76
Plate_96W700_100GMA
Mcs::Usb, 76
Plate_96W700_100PBB
Mcs::Usb, 76
Plate_Dummy
Mcs::Usb, 76
Plate_Dummy_126
Mcs::Usb, 76
Plate_ICB8_24W700_100P8A
Mcs::Usb, 76
Plate_ICB8_96W700_100P8A
Mcs::Usb, 76
PlateClampEnumNet
Mcs::Usb, 77
PlateClampLockEnumNet
Mcs::Usb, 77

PlusMinus10Volts
Mcs::Usb, 53
PlusMinus2Comma5Volts
Mcs::Usb, 53
PlusMinus5Volts
Mcs::Usb, 53
PollStatusEvent
CStimulusFunctionNet, 591
CW2100_StimulatorFunctionNet, 636
PortDirectionEnumNet
Mcs::Usb, 77
Pos900
Mcs::Usb, 81
PositionBase
Mcs::Usb, 80
PositionIIBase
Mcs::Usb, 80
PositionIICentralUnit
Mcs::Usb, 80
PositionImp
Mcs::Usb, 80
PostCommaA
CFilterCoefficientsNet::s_FilterAttributesNet, 717
Mcs::Usb, 67
PostCommaB
CFilterCoefficientsNet::s_FilterAttributesNet, 717
Mcs::Usb, 67
PowerChip
CCMOSMea_FunctionNet, 115
PowerHS
CSCUFunctionNet, 520
PP_Pump_Mode_Type_EnumNet
Mcs::Usb, 78
PPC
Mcs::Usb, 79
PPCFunction
CPPCDeviceNet, 429
PPS2
Mcs::Usb, 79
PPS4plus1
Mcs::Usb, 79
PPS5
Mcs::Usb, 79
PPS5_DIG
Mcs::Usb, 79
PPS_Function
CPPS_DeviceNet, 438
PreCommaA
CFilterCoefficientsNet::s_FilterAttributesNet, 717
Mcs::Usb, 67
PreCommaB
CFilterCoefficientsNet::s_FilterAttributesNet, 717
Mcs::Usb, 67
PrepareAndAppendData
CStg200xDownloadNet, 576
CStimulusFunctionNet, 586
PrepareAndSendData
CStg200xDownloadNet, 577
CStimulusFunctionNet, 587
PrepareChannelData
CDigOutStimulatorFunctionNet, 135
PrepareData
CStimulusFunctionNet, 588
CW2100_StimulatorFunctionNet, 633
PrepareDataSync
CW2100_StimulatorFunctionNet, 634
Product
CMcsUsbListEntryNet, 319
ProductIdEnumNet
Mcs::Usb, 78
Program
CProgramPressureCurveNet, 445
PT100
Mcs::Usb, 88
PT1000
Mcs::Usb, 88
PulseGenerator
CW2100_FunctionNet, 628
Mcs::Usb, 58, 73, 75, 76, 84, 88, 95
PulseGenerator_Mode_EnumNet
Mcs::Usb, 81
PumpOff
CRoboFluidDeviceNet, 494
PumpOn
CRoboFluidDeviceNet, 494

QueryTriggerstatus
CStg200xDownloadNet, 578

RampStart
COctoPotDeviceNet, 404
rawdata
Mcs::Usb, 87
RC
Mcs::Usb, 67
rc100mAh
Mcs::Usb, 71
rc200mAh
Mcs::Usb, 71
rc300mAh
Mcs::Usb, 71
rc30mAh
Mcs::Usb, 71
rcGreater300mAh
Mcs::Usb, 71
Read
CExternDTesterDeviceNet, 138
Read2
CExternDTesterDeviceNet, 139
ReadBlockFromFlash
CMcsUsbFactoryNet, 310
ReadBlockFromIFBGlobalEEprom
CMcsUsbFactoryNet, 310
ReadBlockFromNVMEM
CMcsUsbFactoryNet, 310
ReadClipping
CLIH3DeviceNet, 206

ReadEepromRegisterPreconfig
CMcsUsbNet, 335, 336
ReadPipe
CGenericDevelopDeviceNet, 167
ReadRegister
CMcsUsbNet, 336
ReadRegister32
CMcsUsbNet, 336
ReadRegisterTimeSlot
CMcsUsbNet, 336
ReadUARTData
CLIH3DeviceNet, 206
Receive
CSerialPortNet, 525
ReceiveString
CSerialPortNet, 525
Rectangle
Mcs::Usb, 89
Ref16
Mcs::Usb, 81
Ref24
Mcs::Usb, 81
Ref32
Mcs::Usb, 81
Ref8
Mcs::Usb, 81
Reference
Mcs::Usb, 82
ReferenceElectrodeModeEnumNet
Mcs::Usb, 81
ReferenceElectrodeSwitchPositionEnumNet
Mcs::Usb, 81
RegisterHigh
Mcs::Usb, 64
RegisterLow
Mcs::Usb, 64
Regular
Mcs::Usb, 88
RemoveSoftwareKey
CMcsUsbNet, 336
Renesas
Mcs::Usb, 91
RescanHeadstage
CMcsUsbNet, 336
Reserved1
Mcs::Usb, 88
Reserved2
Mcs::Usb, 88
Reserved3
Mcs::Usb, 88
Reserved4
Mcs::Usb, 88
Reserved5
Mcs::Usb, 88
ResetAdcOffset
COctoPotDeviceNet, 404
ResetChannelmap
CWClassicFunctionNet, 684
ResetDacOffset
COctoPotDeviceNet, 404
ResetHighpassFilter
CFilterConfigurationNet, 143
ResetPipe
CGenericDevelopDeviceNet, 168
ResetStatus
CStg200xDownloadBasicNet, 568
Restart
Mcs::Usb, 82
Retina_LED
Mcs::Usb, 79
RetriggerActionEnumNet
Mcs::Usb, 82
RFFunction
CPositionIIDeviceNet, 425
Rising
Mcs::Usb, 65
RoboCurrentModeEnumNet
Mcs::Usb, 82
RoboDacq
CHiClampDeviceNet, 189
RoboDevice
CSafeISDeviceNet, 506
RoboError_AnotherMaster
CRoboDeviceNet, 488
RoboError_Base
CRoboDeviceNet, 489
RoboError_CannotEscapeEndSwitch
CRoboDeviceNet, 489
RoboError_CommandAlreadyInProgress
CRoboDeviceNet, 489
RoboError_CommandNotPossible
CRoboDeviceNet, 489
RoboError_CommunicationTimeout
CRoboDeviceNet, 489
RoboError_DacqNotReady
CRoboDeviceNet, 489
RoboError_DLLMovementTimeout
CRoboDeviceNet, 489
RoboError_FindReferenceMethod
CRoboDeviceNet, 489
RoboError_GilsonCommandPending
CRoboDeviceNet, 489
RoboError_GilsonTimeout
CRoboDeviceNet, 489
RoboError_GilsonWrondID
CRoboDeviceNet, 490
RoboError_McsBus_UnknownCommand
CRoboDeviceNet, 490
RoboError_NoEndSwitch
CRoboDeviceNet, 490
RoboError_NoMoreData
CRoboDeviceNet, 490
RoboError_NoReference
CRoboDeviceNet, 490
RoboError_NoSpeedOrAcceleration
CRoboDeviceNet, 490

RoboError_OverPressure
CRoboDeviceNet, 490
RoboError_ParameterNotAllowed
CRoboDeviceNet, 490
RoboError_PeristalticTimeout
CRoboDeviceNet, 490
RoboError_Phase0OutOfRange
CRoboDeviceNet, 490
RoboError_PollLoopCanceled
CRoboDeviceNet, 491
RoboError_PollLoopCanceledAndStopMovement
CRoboDeviceNet, 491
RoboError_Pressure
CRoboDeviceNet, 491
RoboError_RangeExceeded
CRoboDeviceNet, 491
RoboError_StateChangeNotPossible
CRoboDeviceNet, 491
RoboError_Timeout
CRoboDeviceNet, 491
RoboError_UnknownCommand
CRoboDeviceNet, 491
RoboInject
Mcs::Usb, 80
RoboMainLowLevelCommand
CRoboDeviceNet, 492
RoboMainStatorLowLevelCommand
CRoboStatorDeviceNet, 503
Robooycte2
Mcs::Usb, 80
RoboStatusEvent
CRoboDeviceNet, 492
RoboStatusEventDelegate
Mcs::Usb, 97
Running
Mcs::Usb, 86
RunTable
CRoboDacqNet, 470

s_FilterAttributesNet
CFilterCoefficientsNet::s_FilterAttributesNet, 716
SafeIS
Mcs::Usb, 80
SampleDstSize16
Mcs::Usb, 82
SampleDstSize32
Mcs::Usb, 82
SampleDstSizeNet
Mcs::Usb, 82
SampleRate
CCreateFilterNet, 122
Samplerate
CMcsUsbDacqNet, 302
SampleSize16Signed
Mcs::Usb, 82
SampleSize16Unsigned
Mcs::Usb, 82
SampleSize24Signed
Mcs::Usb, 83
SampleSize24Unsigned
Mcs::Usb, 82
SampleSize32Signed
Mcs::Usb, 83
SampleSize32Unsigned
Mcs::Usb, 83
SampleSize64Signed
Mcs::Usb, 83
SampleSize64Unsigned
Mcs::Usb, 83
SampleSizeNet
Mcs::Usb, 82
SBSVector1
Mcs::Usb, 60
SBSVector2
Mcs::Usb, 60
SBSVector3
Mcs::Usb, 60
SBSVector4
Mcs::Usb, 60
Scale
CCreateFilterNet, 122
ScanForHeadstages
CWClassicFunctionNet, 684
SCU1ElectrodeGroupHS1
Mcs::Usb, 83
SCU1ElectrodeGroupHS2
Mcs::Usb, 83
SCU1ElectrodeGroupHS3
Mcs::Usb, 83
SCU1ElectrodeGroupHS4
Mcs::Usb, 83
SCU1HS1Sideband1
Mcs::Usb, 84
SCU1HS1Sideband1TriggerStatus
Mcs::Usb, 84
SCU1HS1Sideband2
Mcs::Usb, 84
SCU1HS1Sideband2TriggerStatus
Mcs::Usb, 84
SCU1HS2Sideband1
Mcs::Usb, 84
SCU1HS2Sideband1TriggerStatus
Mcs::Usb, 84
SCU1HS2Sideband2
Mcs::Usb, 84
SCU1HS2Sideband2TriggerStatus
Mcs::Usb, 84
SCU1HS3Sideband1
Mcs::Usb, 84
SCU1HS3Sideband1TriggerStatus
Mcs::Usb, 84
SCU1HS3Sideband2
Mcs::Usb, 84
SCU1HS3Sideband2TriggerStatus
Mcs::Usb, 84
SCU1HS4Sideband1
Mcs::Usb, 85

SCU1HS4Sideband1TriggerStatus
Mcs::Usb, 84
SCU1HS4Sideband2
Mcs::Usb, 85
SCU1HS4Sideband2TriggerStatus
Mcs::Usb, 84
SCU1LEDStim1
Mcs::Usb, 85
SCU1LEDStim1TriggerStatus
Mcs::Usb, 84
SCU1LEDStim2
Mcs::Usb, 85
SCU1LEDStim2TriggerStatus
Mcs::Usb, 84
SCU1LEDStim3
Mcs::Usb, 85
SCU1LEDStim3TriggerStatus
Mcs::Usb, 84
SCU1LEDStim4
Mcs::Usb, 85
SCU1LEDStim4TriggerStatus
Mcs::Usb, 84
SCU2ElectrodeGroupHS1
Mcs::Usb, 83
SCU2ElectrodeGroupHS2
Mcs::Usb, 83
SCU2ElectrodeGroupHS3
Mcs::Usb, 83
SCU2ElectrodeGroupHS4
Mcs::Usb, 83
SCU2HS1Sideband1
Mcs::Usb, 85
SCU2HS1Sideband1TriggerStatus
Mcs::Usb, 85
SCU2HS1Sideband2
Mcs::Usb, 85
SCU2HS1Sideband2TriggerStatus
Mcs::Usb, 85
SCU2HS2Sideband1
Mcs::Usb, 85
SCU2HS2Sideband1TriggerStatus
Mcs::Usb, 85
SCU2HS2Sideband2
Mcs::Usb, 85
SCU2HS2Sideband2TriggerStatus
Mcs::Usb, 85
SCU2HS3Sideband1
Mcs::Usb, 85
SCU2HS3Sideband1TriggerStatus
Mcs::Usb, 85
SCU2HS3Sideband2
Mcs::Usb, 85
SCU2HS3Sideband2TriggerStatus
Mcs::Usb, 85
SCU2HS4Sideband1
Mcs::Usb, 85
SCU2HS4Sideband1TriggerStatus
Mcs::Usb, 85
SCU2HS4Sideband2
Mcs::Usb, 85
SCU2HS4Sideband2TriggerStatus
Mcs::Usb, 85
SCU2LEDStim1
Mcs::Usb, 85
SCU2LEDStim1TriggerStatus
Mcs::Usb, 85
SCU2LEDStim2
Mcs::Usb, 85
SCU2LEDStim2TriggerStatus
Mcs::Usb, 85
SCU2LEDStim3
Mcs::Usb, 85
SCU2LEDStim3TriggerStatus
Mcs::Usb, 85
SCU2LEDStim4
Mcs::Usb, 85
SCU2LEDStim4TriggerStatus
Mcs::Usb, 85
SCU_HeadstageIdEnumNet
Mcs::Usb, 83
SCUDacqGroupChannelEnumNet
Mcs::Usb, 83
SCUDigitalSourceEnumNet
Mcs::Usb, 84
SegmentDefine
CStg200xDownloadNet, 578
SegmentSelect
CStg200xDownloadNet, 578
SegmentStart
CStg200xDownloadNet, 579
SelectHeadstage
CW2100_FunctionNet, 627
SelectTimeSlot
CW2100_StimulatorFunctionNet, 634
Send
CSerialPortNet, 525
SendBuffered
CGilsonDeviceNet, 174
SendChannelData
CDigOutStimulatorFunctionNet, 135
CStg200xDownloadBasicNet, 569
SendCommand
CLIH3DeviceNet, 206
SendImmediate
CGilsonDeviceNet, 174
SendImmediateGetResponse
CGilsonDeviceNet, 174
SendMultiplexedData
CStimulusFunctionNet, 588
SendPreparedData
CStimulusFunctionNet, 588
CW2100_StimulatorFunctionNet, 634
SendStart
CStg200xBasicNet, 546
CStimulusFunctionNet, 588
CW2100_StimulatorFunctionNet, 634

SendStartDacq
CMcsUsbDacqNet, 284
SendStartStgAndDacq
CMcsUsbDacqNet, 284
SendStop
CStg200xBasicNet, 547
CStimulusFunctionNet, 589
CW2100_StimulatorFunctionNet, 634
SendStopDacq
CMcsUsbDacqNet, 284
SendStopStgAndDacq
CMcsUsbDacqNet, 285
SendStopStgAndDacqWithOptions
CMcsUsbDacqNet, 285
SendSyncData
CStg200xDownloadBasicNet, 569
Sensor
CFYIDeviceNet, 155
CMeasureTableDeviceNet, 375
CPatchServerDeviceNet, 410
SerialNumber
CMcsUsbListEntryNet, 319
CMcsUsbNet, 343
SerialPort
CHLADeviceNet, 191
Set4ADCCatchampAverageShift
CMcsBus_SensorNet, 239
Set4ADCMode
CMcsBus_SensorNet, 239
Set4DAC
CMcsBus_SensorNet, 239
Set_Values
CNF_GenDeviceNet, 402
CPathIdentDeviceNet, 411
SetAbsMaxCurrentInMicroAmp
CMultiwellOptoStimFunctionNet, 399
SetAccelerationI
CRoboStatorDeviceNet, 501
SetAccelerationNativeI
CRoboStatorDeviceNet, 501
SetAccelerationNativeXY
CRoboStatorDeviceNet, 501
SetAccelerationNativeZ
CRoboStatorDeviceNet, 501
SetAccelerationXY
CRoboStatorDeviceNet, 501
SetAccelerationZ
CRoboStatorDeviceNet, 501
SetAccelGyroDesiredRate
CW2100_FunctionNet, 627
SetAccelGyroEnabled
CW2100_FunctionNet, 627
SetAccelRange
CW2100_FunctionNet, 627
SetActiveRunningTableNumber
CWarnerValveControllerDeviceNet, 669
SetADC
CWarnerValveControllerDeviceTesterFunctionNet, 680
SetAdcChannels
CSafeISDeviceNet, 505
SetADCGain
CGrapheneFunctionNet, 184, 185
SetADCInputOffset
CCMOSMea_FunctionNet, 115
SetADCOffset
CGrapheneFunctionNet, 185
SetAdcOffset
CLIH3DeviceNet, 206
COctoPotDeviceNet, 404
SetAdcOffsetPermanent
CLIH3DeviceNet, 207
SetAdcSamplePos
CSafeISDeviceNet, 505
SetAirpressureLimit
CRoboDeviceNet, 485
SetAirValve
CRoboDeviceNet, 485
SetAllDigout
CRoboDacqNet, 470
SetAmplificationSwitch
COctoPotDeviceNet, 404
SetAmplitude
CChannelTestDeviceNet, 105
SetAmplitude_nA
CTEERFunctionNet, 614
SetAnalogOutADCRange
CSCUFunctionNet, 521
SetAnalogOutChannel
CW2100_FunctionNet, 627
SetAnalogOutChannels
CSCUFunctionNet, 521
SetAnalogOutDACRange
CSCUFunctionNet, 521
SetAnalogOutFilter
CW2100_FunctionNet, 627
SetAnalogThresholdHigh
CWarnerValveControllerDeviceNet, 669
SetAnalogThresholdLow
CWarnerValveControllerDeviceNet, 670
SetAnalogVoltageRange
CPPCFunctionNet, 436
SetAnalogVoltages
CPPS_FunctionNet, 441
SetAttenuation
CChannelTestDeviceNet, 105
SetAudioChannels
CMeaAudioFunctionNet, 348
CW2100_FunctionNet, 627
SetAudioOutDacParameter
CLIH3DeviceNet, 207
SetAutocalibrationDisabled
CStg200xBasicNet, 547
SetAxisConfig

CRoboDeviceNet::RoboMainLowLevelCommands, 713
SetAxisLED
CRoboocyte2DeviceNet, 497
SetAxisParametersEeprom
CMcsBus_AxisParametersNet, 210, 211
SetBandwidthByIndex
CIntanMea_FunctionNet, 196
SetBaseFrequency
CRFFunctionNet, 457
SetBaseSamplerate
CCMOSMeaDeviceNet, 119
CGrapheneASICDeviceNet, 176
SetBath
CCMOSMea_FunctionNet, 115
SetBathclamp
COctoPotDeviceNet, 404
SetBathMode
CCMOSMea_FunctionNet, 115
SetBlankingEnable
CStg200xBasicNet, 547, 548
SetBuffer
CGenericDevelopDeviceNet, 168
SetBufferIndex
CTEERFunctionNet, 614
SetBusAddress
CMcsBusNet, 250
SetBusAddressEeprom
CMcsBusNet, 250
SetByteBuffer
CGenericDevelopDeviceNet, 168
SetCalibration
CTcxDeviceNet, 604
SetCardinalDacqSamplerate
CInterfaceboardFunctionNet, 199
SetCardinalStgOutputrate
CInterfaceboardFunctionNet, 200
SetChannel
CSw2to64DeviceNet, 592
SetChannelmap
CWClassicFunctionNet, 684
SetChannels
CSw2to64DeviceNet, 593
SetChannelSwitch
COctoPotDeviceNet, 404
SetChargingMode
CMultiBatteryChargerDeviceNet, 385
SetChargingPCoefficient
CMultiBatteryChargerDeviceNet, 385
SetCheckVoltage
COkuvisionStimulatorDeviceNet, 408
SetClampMode
CTEERFunctionNet, 615
CWarnerUssingFunctionNet, 650
SetColorRgb
CMultiwellOptoStimFunctionNet, 400
SetColorStr
CMultiwellOptoStimFunctionNet, 400
SetCommand
CMcsBusNet, 251
CPedoterDeviceNet, 412
CRoboDacqNet, 470
SetConfiguration
CMcsUsbNet, 337
SetConfigurationBit
CRoboDacqNet, 470
SetConfigurationBitAxc
CRoboDacqNet, 471
SetConfigurationBitBlu_Led
CRoboDacqNet, 471
SetConfigurationBitBlu_LedToggleFast
CRoboDacqNet, 471
SetConfigurationBitBlu_LedToggleSlow
CRoboDacqNet, 471
SetConfigurationBitCC_Gen
CRoboDacqNet, 471
SetConfigurationBitCV_Gen
CRoboDacqNet, 471
SetConfigurationBitRC_Gen
CRoboDacqNet, 471
SetConfigurationBitRed_Led
CRoboDacqNet, 471
SetConfigurationBitRed_LedSaturation
CRoboDacqNet, 471
SetConfigurationBitRed_LedToggleFast
CRoboDacqNet, 472
SetConfigurationBitRed_LedToggleSlow
CRoboDacqNet, 472
SetConfigurationBitRelais
CRoboDacqNet, 472
SetConfigurationBitRV_Gen
CRoboDacqNet, 472
SetConfigurationBitStream
CRoboDacqNet, 472
SetConfigurationBitSupply
CRoboDacqNet, 472
SetControllerParams
CTEERFunctionNet, 615
SetCrossTalkOffset
CRoboDacqNet, 472
SetCrossTalkOptimum
CRoboDacqNet, 472
SetCurrentAirvalveLimit
CRoboDeviceNet, 485
SetCurrentAndAir
CRoboDeviceNet, 485
SetCurrentAndAirXY
CRoboStatorDeviceNet, 501
SetCurrentEditTableNumber
CWarnerValveControllerDeviceNet, 670
SetCurrentEnable
CTEERFunctionNet, 615
SetCurrentFactor
COkuvisionStimulatorDeviceNet, 408
SetCurrentMode
CStg200xBasicNet, 549

SetCurrentRangeSelectedIndex
CStg200xBasicNet, 549
SetCycles
CMeaCleanDeviceNet, 351
CMeaCoatDeviceNet, 356
SetD
CTcxDeviceNet, 604
SetDacAmplificationFactor
CStg200xBasicNet, 549
SetDacAutoControl
COctoPotDeviceNet, 404
SetDacIdleValue
CLIH3DeviceNet, 207
SetDacMode
CSafeISDeviceNet, 505
SetDACOffset
CGrapheneFunctionNet, 185, 186
COkuvisionStimulatorDeviceNet, 408
SetDacOffset
CDacCalibrationFunctionNet, 124
CLIH3DeviceNet, 207
COctoPotDeviceNet, 405
SetDacOffsetPermanent
CLIH3DeviceNet, 208
SetDacPeriode
CSafeISDeviceNet, 505
SetDacPulseform
CSafeISDeviceNet, 505
SetDacqLegacyMode
CSCUFunctionNet, 521
SetDacRange
CW2100_FunctionNet, 628
SetDACs
CMcsBus_SensorNet, 239
SetDacUseIdleValue
CLIH3DeviceNet, 208
SetDacValue
COctoPotDeviceNet, 405
SetDataMode
CMcsUsbDacqNet, 285
SetDefault
CWarnerValveControllerDeviceNet, 670
SetDestinationSerialNumber
CMcsUsbFactoryNet, 310
SetDetectionThreshold
CMcsBus_SensorNet, 239
SetDevice
CTcxDeviceNet, 604
SetDeviceId
CUsbDeviceConfigurationFunctionNet, 619
SetDeviceList
CPositionImpDeviceNet, 427
SetDeviceName
CUsbDeviceConfigurationFunctionNet, 619
SetDeviceType
CTcxDeviceNet, 604
SetDevname
CTcxDeviceNet, 604
SetDiagnosticMode
CIntanMea_FunctionNet, 196
SetDigitalData
CMeaDigitalDataFunctionNet, 367, 368
SetDigitalOut
CMeaDeviceNet, 362
SetDigitalOutPortInvert
CWarnerValveControllerDeviceNet, 670
SetDigitalOutPortValve
CWarnerValveControllerDeviceNet, 671
SetDigitalPortDirection
CWarnerValveControllerDeviceNet, 671
SetDigitalSource
CMcsUsbDacqNet, 286–288
SetDigitalStimulatorTrigger
CW2100_StimulatorFunctionNet, 635
SetDigitalStimulatorTriggerSlope
CW2100_StimulatorFunctionNet, 635
SetDigout
CFluidControlDeviceNet, 152
CRoboDacqNet, 472
SetDigoutMode
CStg200xBasicNet, 551
SetDigOutState
CLIH3DeviceNet, 208
SetDigoutValue
CStg200xBasicNet, 551
SetDIO
CMcsBus_FYIExtensionNet, 213
SetDischargeCurrentSetPoint
CMultiBatteryChargerDeviceNet, 385
SetDisplayMode
CWarnerValveControllerDeviceNet, 671
SetDisplayText
CRoboDacqNet, 473
SetDownsampleFactor
CRoboDacqNet, 473
SetDSPHighPassByIndex
CIntanMea_FunctionNet, 196
SetDuration
CMeaCoatDeviceNet, 357
SetEEpromPage
CLIH3DeviceNet, 208
SetElectrodeDacMux
CStg200xBasicNet, 551–553
SetElectrodeEnable
CStg200xBasicNet, 553–555
SetElectrodeMode
CStg200xBasicNet, 555, 557, 558
SetEnableAmplifierProtectionSwitch
CStg200xBasicNet, 558, 559
SetEnableHeaterLimit
CTcxDeviceNet, 604
SetEnablePulse
CWarnerUssingFunctionNet, 650
SetEnableThermocouple
CTcxDeviceNet, 605
SetExternalElectrodeEnable

CStg200xBasicNet, 560
SetExternalLED
CTEERFunctionNet, 615
SetFAAmplification
CStg200xBasicNet, 560
SetFilter
CRoboDacqNet, 473
SetFilterCoeffs
CRoboDacqNet, 473
SetFilterParameter
CFilterConfigurationNet, 143
CFilterConfigurationRegisterNet, 145
SetFilterParameterPermanent
CFilterConfigurationNet, 143
CFilterConfigurationRegisterNet, 145
SetFilterParametersHeadstage
CWClassicFunctionNet, 684
SetFinalDischargeVoltage
CMultiBatteryChargerDeviceNet, 386
SetFrequency
CChannelTestDeviceNet, 105
CRadioControledDevicesNet, 450
SetGain
CPgaDeviceNet, 416
SetGate
CCMOSMea_FunctionNet, 115
SetGateFloating
CCMOSMea_FunctionNet, 116
SetGateToVOP
CCMOSMea_FunctionNet, 116
SetGlobalRepeat
CDigOutStimulatorFunctionNet, 136
SetGyroRange
CW2100_FunctionNet, 628
SetHasChecksum
CWClassicFunctionNet, 685
SetHeadstage
CStg200xBasicNet, 561
SetHeadstageFrameCyclesToComparePermanent
CSCUFunctionNet, 522
SetHeadstageLinkSpeedPermanent
CSCUFunctionNet, 522
SetHeadstageNumberOfAnalogChannelsPermanent
CSCUFunctionNet, 522
SetHeadstageOnOff
CW2100_FunctionNet, 628
CWClassicFunctionNet, 685
SetHeadstagePowerStateAtStart
CSCUFunctionNet, 522
SetHeadstageSampleratePermanent
CSCUFunctionNet, 523
SetHeadstageSamplingActive
CW2100_FunctionNet, 628
SetHeadstageToSleep
CW2100_FunctionNet, 628
SetHeaterLimit
CTcxDeviceNet, 605
SetHighCurrentMode
CWarnerUssingFunctionNet, 650
SetHighpassFilterEnable
CFilterConfigurationNet, 143
SetHWConfig
CRoboDeviceNet::RoboMainLowLevelCommands, 713
SetHWRevision
CRoboDeviceNet::RoboMainLowLevelCommands, 713
SetHWRevisionEeprom
CMcsBusNet, 251
SetHWSelectedChannels
CWClassicFunctionNet, 685
SetI
CTcxDeviceNet, 605
SetIClamp
CRoboDacqNet, 473
SetICoeff
CRobo_FYITemp_FunctionNet, 461
SetICOffset
CRoboDacqNet, 473
SetIdleModeOffset
CWarnerUssingFunctionNet, 651
SetIGain
CRoboDacqNet, 473
SetImpedanceTestFrequency
CMeaImpedanceDeviceNet, 374
SetImpId
CPositionImpDeviceNet, 427
SetImplantCurrentSetpoint
CPositionIIDeviceNet, 422
SetInMovement
CRoboDeviceNet, 485
SetIntanRegister
CIntanMea_FunctionNet, 196
SetIntBuffer
CGenericDevelopDeviceNet, 169
SetIO
CWarnerValveControllerDeviceTesterFunctionNet, 680
SetIODirection
CWarnerValveControllerDeviceTesterFunctionNet, 681
SetIoVoltage
CInterfaceboard2FunctionNet, 198
SetLatency
CMcsBus_SensorNet, 239
SetLayoutConfiguration
CMEA2100x256FunctionNet, 345
SetLED
CRetinaLedDeviceNet, 452
SetLEDSwitch
CMcsBus_ExtensionNet, 212
SetLength
CRobo_FYIProgram_FunctionNet, 459
SetLiquidResistance
CTEERFunctionNet, 616
CWarnerUssingFunctionNet, 651

SetListmodeIndexRange
CStg200xBasicNet, 561
SetListmodeTriggerSource
CStg200xBasicNet, 561, 562
SetLowCurrentMode
CWarnerUssingFunctionNet, 651
SetLumi
CRetinaLedDeviceNet, 452
SetMaxCurrent
CMeaCoatDeviceNet, 357
SetMaxDurationHighCurrentInMicroSec
CMultiwellOptoStimFunctionNet, 400
SetMaxDutyCycleHighCurrent
CMultiwellOptoStimFunctionNet, 400
SetMaxHeaterPowerMultiwell
CTcxDeviceNet, 605
SetMaxNoPressure
CRoboDeviceNet::RoboMainLowLevelCommands, 713
SetMaxNoPressureWaitTime
CRoboDeviceNet::RoboMainLowLevelCommands, 713
SetMaxP
CTcxDeviceNet, 605
SetMaxPower
COkuvisionStimulatorDeviceNet, 409
CRobo_FYITemp_FunctionNet, 461
SetMaxPressureWaitTime
CRoboDeviceNet::RoboMainLowLevelCommands, 713
SetMaxVoltage
CMeaCleanDeviceNet, 352
COkuvisionStimulatorDeviceNet, 409
SetMCAcceleration
CMcsBus_MotorControlNet, 223
SetMCAccelerationEeprom
CMcsBus_MotorControlNet, 224
SetMCAccelerationShortCommand
CMcsBus_MotorControlNet, 224
SetMCAxisRevisionEeprom
CMcsBus_MotorControlNet, 224
SetMCBreakCurrent
CMcsBus_MotorControlNet, 224
SetMCBreakCurrentEeprom
CMcsBus_MotorControlNet, 224
SetMCConfig
CMcsBus_MotorControlNet, 224
SetMCConfigEeprom
CMcsBus_MotorControlNet, 225
SetMCCurrent
CMcsBus_MotorControlNet, 225
SetMCCurrentEeprom
CMcsBus_MotorControlNet, 225
SetMCCurrentMode
CMcsBus_MotorControlNet, 225
SetMCCurrentModeEeprom
CMcsBus_MotorControlNet, 225
SetMCCurrentModeShortCommand
CMcsBus_MotorControlNet, 225
SetMCCurrentPosition
CMcsBus_MotorControlNet, 226
SetMCCurrentShortCommand
CMcsBus_MotorControlNet, 226
SetMCMaxAcceleration
CMcsBus_MotorControlNet, 226
SetMCMaxAccelerationEeprom
CMcsBus_MotorControlNet, 226
SetMCMaxCurrent
CMcsBus_MotorControlNet, 226
SetMCMaxCurrentEeprom
CMcsBus_MotorControlNet, 226
SetMCMaxSpeed
CMcsBus_MotorControlNet, 227
SetMCMaxSpeedEeprom
CMcsBus_MotorControlNet, 227
SetMCMaxTravel
CMcsBus_MotorControlNet, 227
SetMCMaxTravelEeprom
CMcsBus_MotorControlNet, 227
SetMCMaxTravelShortCommand
CMcsBus_MotorControlNet, 227
SetMCNewPosition
CMcsBus_MotorControlNet, 227
SetMCOutputOnOff
CMcsBus_MotorControlNet, 228
SetMCReference
CMcsBus_MotorControlNet, 228
SetMCReferenceCurrent
CMcsBus_MotorControlNet, 228
SetMCReferenceCurrentEeprom
CMcsBus_MotorControlNet, 228
SetMCRegulatorGain
CMcsBus_MotorControlNet, 228
SetMCRegulatorGainEeprom
CMcsBus_MotorControlNet, 228
SetMCRotation
CMcsBus_MotorControlNet, 229
SetMCScalingFactor
CMcsBus_MotorControlNet, 229
SetMCScalingFactorEeprom
CMcsBus_MotorControlNet, 229
SetMCSpeed
CMcsBus_MotorControlNet, 229
SetMCSpeedEeprom
CMcsBus_MotorControlNet, 229
SetMCSpeedShortCommand
CMcsBus_MotorControlNet, 229
SetMCSpeedUnitEeprom
CMcsBus_MotorControlNet, 230
SetMCStandbyCurrent
CMcsBus_MotorControlNet, 230
SetMCStandbyCurrentEeprom
CMcsBus_MotorControlNet, 230
SetMCStandbyTime
CMcsBus_MotorControlNet, 230
SetMCStandbyTimeEeprom

CMcsBus_MotorControlNet, 230
SetMeasurementMode
CStg200xBasicNet, 562
SetMinimalThreshold
CMcsBus_SensorNet, 239
SetMinNoPressureWaitTime
CRoboDeviceNet::RoboMainLowLevelCommands, 713
SetMinPressure
CRoboDeviceNet, 486
CRoboDeviceNet::RoboMainLowLevelCommands, 713
SetMinPressureWaitTime
CRoboDeviceNet::RoboMainLowLevelCommands, 713
SetMinVoltage
CMeaCleanDeviceNet, 352
SetModeSelect
CPulseGeneratorFunctionNet, 448
SetMovePump
CMcsBus_SensorNet, 240
SetMultiHeadstageMode
CW2100_FunctionNet, 628
SetNanoVoltsPerKelvin
CMcsBus_TempSensorNet, 243
SetNeurochipMemoryData
CCMOSMea_FunctionNet, 116
SetNoFilterCoeffs
CRoboDacqNet, 473
SetNumberOfAnalogChannels
CMeaDeviceNet, 363
SetNumberOfChannels
CMeaDeviceNet, 364
COctoPotDeviceNet, 405
SetOffsetCurrent
CMeaCoatDeviceNet, 357
SetOnOff
CTcxDeviceNet, 605
SetOutputMap
CStg200xDownloadNet, 579
SetOutputRate
COctoPotDeviceNet, 405
CStg200xBasicNet, 562
SetP
CTcxDeviceNet, 606
SetParameter
CRoboDeviceNet::RoboMainLowLevelCommands, 714
SetPattern
CMeaSwitchDeviceNet, 376
SetPatternBool
CMeaSwitchDeviceNet, 377
SetPatternListEntry
COctoPotDeviceNet, 405
SetPauseDuration
CMeaCoatDeviceNet, 357
SetPCoeff
CRobo_FYITemp_FunctionNet, 461
SetPeriod
CPulseGeneratorFunctionNet, 448
SetPeriod_us
CTEERFunctionNet, 616
SetPermanentCurrentInMicroAmp
CMultiwellOptoStimFunctionNet, 400
SetPersistency
CRetinaLedDeviceNet, 452
SetPGain
CRoboDacqNet, 473
SetPidParameter
COctoPotDeviceNet, 405
SetPiezoState
CMcsBus_SensorNet, 240
SetPlateMux
CMultiwellDeviceNet, 394
SetPlateType
CMultiwellDeviceNet, 395
SetPoti
CMcsUsbDacqNet, 288
SetPowerMuxPlate
CMultiwellDeviceNet, 395
SetPowerStrength
CPositionIIDeviceNet, 422
SetPressureOffset
CMcsBus_SensorNet, 240
CPPCFunctionNet, 436
SetPressureRange
CPPCFunctionNet, 436
SetPulse
CWarnerUssingFunctionNet, 651
SetPulseform
COkuvisionStimulatorDeviceNet, 409
SetPulseLength
CPulseGeneratorFunctionNet, 448
SetPumpCouple
CPPS_FunctionNet, 442
SetPumpEnableSpeedRatio
CPPS_FunctionNet, 442
SetPumpFastOnOff
CPPS_FunctionNet, 442
SetPumpFastSpeed
CPPS_FunctionNet, 442
SetPumpFunctionSpeeds
CPPS_FunctionNet, 442
SetPumpManualOnOff
CPPS_FunctionNet, 442
SetPumpMaxSpeed
CPPS_FunctionNet, 442
SetPumpModeType
CPPCFunctionNet, 437
CPPS_FunctionNet, 442
SetPumpSpeed
CRoboFluidDeviceNet, 494
SetPumpSpeedRatio
CPPS_FunctionNet, 443
SetPumpSpeedUnit
CPPCFunctionNet, 437

CPPS_FunctionNet, 443
SetPWM
CFluidControlDeviceNet, 152
SetRampParameter
COctoPotDeviceNet, 405
SetRatedCapacity
CMultiBatteryChargerDeviceNet, 386
SetRatedCapacityVolatile
CMultiBatteryChargerDeviceNet, 386
SetRecordingNumber
CRoboDacqNet, 474
SetReferenceElectrodeMode
CSCUFunctionNet, 523
SetReferenceElectrodeSwitchState
CSCUFunctionNet, 523
SetRegionOfInterests
CCMOSMeaDeviceNet, 119
CGrapheneASICDeviceNet, 176
SetRegulationTimeouts
CMcsBus_SensorNet, 240
SetRegulatorFactor
CMcsBus_SensorNet, 240
SetRegulatorOnOff
CMcsBus_SensorNet, 240
CRobo_FYITemp_FunctionNet, 461
SetRepeat
CRetinaLedDeviceNet, 453
SetRepeats
CProgramPressureCurveNet, 445
SetResetFilter
CWClassicFunctionNet, 685
SetRFFrequency
CPositionImpDeviceNet, 428
SetRFFrequencyHeadstage
CWClassicFunctionNet, 685
SetRFFrequencyReceiver
CWClassicFunctionNet, 685
SetRFFrequencyReceiverEeprom
CWClassicFunctionNet, 685
SetRFLostBehaviour
CWClassicFunctionNet, 685
SetRFPower
CWClassicFunctionNet, 686
SetRotatePump
CMcsBus_SensorNet, 240
SetRTC
COkuvisionStimulatorDeviceNet, 409
CPositionIIDeviceNet, 422
SetSampleInterval
CLIH3DeviceNet, 209
SetSamplePeriode
CMcsBus_SensorNet, 241
SetSamplerate
CMcsUsbDacqNet, 288
SetScreen
CRoboDacqNet, 474
SetSearchReferenceFastAccel
CRoboDeviceNet::RoboMainLowLevelCommands, 714
SetSearchReferenceFastSpeed
CRoboDeviceNet::RoboMainLowLevelCommands, 714
SetSearchReferenceFineAccel
CRoboDeviceNet::RoboMainLowLevelCommands, 714
SetSearchReferenceFineSpeed
CRoboDeviceNet::RoboMainLowLevelCommands, 714
SetSearchReferenceMethod
CRoboDeviceNet::RoboMainLowLevelCommands, 714
SetSearchReferenceMoveOut
CRoboDeviceNet::RoboMainLowLevelCommands, 715
SetSearchReferenceOffsetPos
CRoboDeviceNet::RoboMainLowLevelCommands, 715
SetSelectedChannels
CMcsUsbDacqNet, 290–292
CW2100_FunctionNet, 628
SetSelectedChannelsQueue
CMcsUsbDacqNet, 292–294
SetSelectedData
CMcsUsbDacqNet, 294–296
SetSelectedHeadstage
CWClassicFunctionNet, 686
SetSensorType
CTcxDeviceNet, 606
SetSerialNumberHeadstage
CWClassicFunctionNet, 686
SetSetpoint
CTcxDeviceNet, 606
SetShortBuffer
CGenericDevelopDeviceNet, 169
SetSimulation
CRoboDacqNet, 474
SetSineParameter
COctoPotDeviceNet, 405
SetSingleHeater
CMcsBus_FYIExtensionNet, 213
SetSingleValve
CFluidControlDeviceNet, 153
CRoboFluidDeviceNet, 494
SetSlope
CMeaCleanDeviceNet, 352
CMeaCoatDeviceNet, 358
SetSoftwareKey
CMcsUsbNet, 337
SetSollPressure
CMcsBus_SensorNet, 241
SetSollTemp
CRobo_FYITemp_FunctionNet, 461
SetSourceBulk
CCMOSMea_FunctionNet, 116
SetSourceDrain

CCMOSMea_FunctionNet, 116
SetSourceGate
CCMOSMea_FunctionNet, 116
SetSpeedI
CRoboStatorDeviceNet, 502
SetSpeedNativeI
CRoboStatorDeviceNet, 502
SetSpeedNativeXY
CRoboStatorDeviceNet, 502
SetSpeedNativeZ
CRoboStatorDeviceNet, 502
SetSpeedXY
CRoboStatorDeviceNet, 502
SetSpeedZ
CRoboStatorDeviceNet, 502
SetStartTriggerSlope
CDigOutStimulatorFunctionNet, 136
SetStateDebugData
CPositionIIDeviceNet, 424
SetStateEventData
CPositionIIDeviceNet, 424
SetStgProgramInfo
CStg200xBasicNet, 563
SetStimulusSites
CCMOSMea_FunctionNet, 116
SetStopTriggerSlope
CDigOutStimulatorFunctionNet, 136
SetStringFormat
CMcsUsbListEntryNet, 318
CMcsUsbListNet, 322
SetSubChannel
CMcsBus_MotorControlNet, 230
SetSwitches
CSafeISDeviceNet, 506
SetSyncoutMap
CStg200xBasicNet, 563
SetTableName
CWarnerValveControllerDeviceNet, 671
SetTablepointer
CRetinaLedDeviceNet, 453
SetTableStep
CWarnerValveControllerDeviceNet, 672
SetTableStepAll
CWarnerValveControllerDeviceNet, 672
SetTestMode
CRFFunctionNet, 457
SetThermocoupleNanovoltPerKelvin
CFluidControlDeviceNet, 153
CTcxDeviceNet, 606
SetThermoOffset
CMcsBus_TempSensorNet, 243
Settings
Mcs::Usb, 95
SetTouchPadEnable
CMultiwellDeviceNet, 395
SetTransformer
CMeFunctionNet, 379
SetTrigger
CRetinaLedDeviceNet, 453
CWarnerValveControllerDeviceTesterFunctionNet, 681
SetTriggerMaskValue
CMeaDeviceNet, 364
CRoboDacqNet, 474
SetTriggerPeriod
CMeaDeviceNet, 365
SetTriggerSource
CStg200xBasicNet, 563
SetTriggerSyncDirection
CWarnerValveControllerDeviceTesterFunctionNet, 681
SetUByteBuffer
CGenericDevelopDeviceNet, 170
SetUClamp
CRoboDacqNet, 474
SetUCOffset
CRoboDacqNet, 474
SetUIntBuffer
CGenericDevelopDeviceNet, 170
SetupGroupDacqQueue
CMcsUsbDacqNet, 296
SetupRetriggerMode
CStg200xDownloadBasicNet, 569, 570
SetupTrigger
CStg200xDownloadBasicNet, 570
CStimulusFunctionNet, 589
SetupTriggerSingle
CStg200xDownloadBasicNet, 572
CStimulusFunctionNet, 590
SetUseBubble
CPPS_FunctionNet, 443
SetUserParameter
CRoboDeviceNet::RoboMainLowLevelCommands, 715
SetUShortBuffer
CGenericDevelopDeviceNet, 171
SetUVOffset
CRoboDacqNet, 474
SetValue
CGenericDevelopDeviceNet, 172
SetValve
CFluidControlDeviceNet, 153
CRoboFluidDeviceNet, 494
SetValve1
CRobo_FYIProgram_FunctionNet, 459
SetValve2
CRobo_FYIProgram_FunctionNet, 459
SetValveActive
CPPCFunctionNet, 437
CWarnerValveControllerDeviceNet, 672
SetValveCurrent
CWarnerValveControllerDeviceNet, 672
SetValveDigitalInInvert
CWarnerValveControllerDeviceNet, 673
SetValveDigitalInPort
CWarnerValveControllerDeviceNet, 673

SetValveLedOn
CWarnerValveControllerDeviceNet, 673
SetValveManualGroup
CWarnerValveControllerDeviceNet, 673
SetValveManualState
CWarnerValveControllerDeviceNet, 674
SetValveMode
CWarnerValveControllerDeviceNet, 674
SetValves
CMcsBus_FYIExtensionNet, 213
SetValvesActiveMap
CWarnerValveControllerDeviceNet, 674
SetValvesManualStateMap
CWarnerValveControllerDeviceNet, 674
SetValveTableEntry
CWarnerValveControllerDeviceNet, 674
SetVdsVgs
CGrapheneFunctionNet, 186
SetVdVs
CGrapheneFunctionNet, 187
SetVdVsDAC
CGrapheneFunctionNet, 187
SetVelocityI
CRoboStatorDeviceNet, 502
SetVelocityXY
CRoboStatorDeviceNet, 502
SetVelocityZ
CRoboStatorDeviceNet, 502
SetVMMaxNegativeCurrent
CMcsBus_VoltageModeNet, 246
SetVMMaxNegativeCurrentEeprom
CMcsBus_VoltageModeNet, 246
SetVMMaxNegativeVoltage
CMcsBus_VoltageModeNet, 246
SetVMMaxNegativeVoltageEeprom
CMcsBus_VoltageModeNet, 246
SetVMMaxPositiveCurrent
CMcsBus_VoltageModeNet, 247
SetVMMaxPositiveCurrentEeprom
CMcsBus_VoltageModeNet, 247
SetVMMaxPositiveVoltage
CMcsBus_VoltageModeNet, 247
SetVMMaxPositiveVoltageEeprom
CMcsBus_VoltageModeNet, 247
SetVMOutputOnOff
CMcsBus_VoltageModeNet, 247
SetVMVoltage
CMcsBus_VoltageModeNet, 247
SetVolatileClampOffset
CMultiwellDeviceNet, 396
SetVoltage12VLimit
CRoboDeviceNet, 486
SetVoltage5VLimit
CRoboDeviceNet, 486
SetVoltageAirvalveLimit
CRoboDeviceNet, 486
SetVoltageClampControllerParam_D
CWarnerUssingFunctionNet, 652
SetVoltageClampControllerParam_I
CWarnerUssingFunctionNet, 652
SetVoltageClampControllerParam_P
CWarnerUssingFunctionNet, 652
SetVoltageMode
CStg200xBasicNet, 563, 564
SetVoltageRange
CGrapheneFunctionNet, 188
SetVoltageRangeByIndex
CMcsUsbDacqNet, 296
SetVoltageRangeForIndex
CMeaUSBDeviceNet, 378
SetVoltageRangeInMicroVolt
CMcsUsbDacqNet, 297
SetVoltageRangeSelectedIndex
CStg200xBasicNet, 564
SetVoltageResolution
CGrapheneFunctionNet, 188
SetVoltageRs485ALimit
CRoboDeviceNet, 486
SetVoltageRs485BLimit
CRoboDeviceNet, 486
SetVoltageValvesLimit
CRoboDeviceNet, 486
SetWaveform
CChannelTestDeviceNet, 106
CTEERFunctionNet, 616
SetWaveLengthInNanometer
CMultiwellOptoStimFunctionNet, 401
SetWorkingFrequency
CRFFunctionNet, 457
SetWPADebugMode
CWClassicFunctionNet, 686
SetWPAType
CWClassicFunctionNet, 686
SetXGain
CRoboDacqNet, 474
Sideband
CStimulusFunctionNet::SidebandData, 718
SidebandData
CStimulusFunctionNet::SidebandData, 718
Signed_16bit
Mcs::Usb, 63
Signed_24bit
Mcs::Usb, 63
Signed_32bit
Mcs::Usb, 63
Sine
Mcs::Usb, 89
SineStart
COctoPotDeviceNet, 406
Single
Mcs::Usb, 82
SingleWell
Mcs::Usb, 87
SixWell
Mcs::Usb, 87
size

DigitalSource< digitalsourceenum >, 689
DigitalSourceGeneral, 691
SmartImplant
Mcs::Usb, 80
SN
HeadStageIDType, 703
SOFAndCTRLword
Mcs::Usb, 61
Software
Mcs::Usb, 68
SoftwareDongle
Mcs::Usb, 80
Source
DigitalSource< digitalsourceenum >, 690
DigitalSourceGeneral, 691
Standby
Mcs::Usb, 82, 92
Start
CMeaCleanDeviceNet, 352
CMeaCoatDeviceNet, 358
CRobo_FYIProgram_FunctionNet, 459
Mcs::Usb, 65
StartDacq
CMcsUsbDacqNet, 297, 298
StartInternalCalibration
CTEERFunctionNet, 616
StartLoop
CMcsUsbDacqNet, 299, 300
StartMCMovement
CMcsBus_MotorControlNet, 231
StartMeasurement
CMeaImpedanceDeviceNet, 374
StartPoll
CStimulusFunctionNet, 590
CW2100_StimulatorFunctionNet, 635
StartQueue
CRoboDeviceNet, 486
StartSampling
CTEERFunctionNet, 616
StartSync
CMcsBus_SensorNet, 241
State
HeadStageIDTypeState, 705
Mcs::Usb, 92
Status
CUsbExceptionNet, 620
Status_AlreadyConfigured
CMcsUsbNet, 339
Status_BadStartFrame
CMcsUsbNet, 339
Status_Btstuff
CMcsUsbNet, 339
Status_BufferOverrun
CMcsUsbNet, 339
Status_BufferUnderrun
CMcsUsbNet, 339
Status_Canceled
CMcsUsbNet, 339
Status_Canceling
CMcsUsbNet, 340
Status_ConnectedPipes
CMcsUsbNet, 340
Status_ControlNotOwned
CMcsUsbNet, 340
Status_Crc
CMcsUsbNet, 340
Status_DataOverrun
CMcsUsbNet, 340
Status_DataToggleMismatch
CMcsUsbNet, 340
Status_DataUnderrun
CMcsUsbNet, 340
Status_DeviceLocked
CMcsUsbNet, 340
Status_DeviceNotFound
CMcsUsbNet, 340
Status_DeviceRemoved
CMcsUsbNet, 340
Status_DevNotResponding
CMcsUsbNet, 340
Status_EndpointHalted
CMcsUsbNet, 341
Status_ErrorBusy
CMcsUsbNet, 341
Status_ErrorShortTransfer
CMcsUsbNet, 341
Status_Fifo
CMcsUsbNet, 341
Status_FrameControlOwned
CMcsUsbNet, 341
Status_InternalHcError
CMcsUsbNet, 341
Status_InvalidDeviceHandle
CMcsUsbNet, 341
Status_InvalidHandle
CMcsUsbNet, 341
Status_InvalidParameter
CMcsUsbNet, 341
Status_InvalidPipeHandle
CMcsUsbNet, 341
Status_InvalidUrbFunction
CMcsUsbNet, 342
Status_IoPending
CMcsUsbNet, 342
Status_IoTimeout
CMcsUsbNet, 342
Status_IsochRequestFailed
CMcsUsbNet, 342
Status_LastUsbErrorMismatch
CMcsUsbNet, 342
Status_NoBandwidth
CMcsUsbNet, 342
Status_NoMemory
CMcsUsbNet, 342
Status_NoSuchDevice
CMcsUsbNet, 342

Status_NotAccessed
CMcsUsbNet, 342
Status_NotSupported
CMcsUsbNet, 342
Status_PidCheckFailure
CMcsUsbNet, 342
Status_PipeNotLinked
CMcsUsbNet, 343
Status_RequestFailed
CMcsUsbNet, 343
Status_RequestMutexFailed
CMcsUsbNet, 343
Status_RequestMutexTimeout
CMcsUsbNet, 343
Status_Stall
CMcsUsbNet, 343
Status_Unconfigured
CMcsUsbNet, 343
Status_UnexpectedPid
CMcsUsbNet, 343
STG
Mcs::Usb, 78
Stg1
Mcs::Usb, 66
STG1DACSignalGroup
Mcs::Usb, 62, 72, 83
STG1SidebandsGroup
Mcs::Usb, 62, 72, 84
STG1TriggerStatusGroup
Mcs::Usb, 62, 72, 84
Stg2
Mcs::Usb, 66
Stg200xDigoutModeEnumNet
Mcs::Usb, 85
Stg200xMultiFileSubmodeEnumNet
Mcs::Usb, 86
Stg200xPollStatusEvent
CStg200xDownloadNet, 579
Stg200xSegmentFlagsEnumNet
Mcs::Usb, 86
Stg200xTriggerStatusEnumNet
Mcs::Usb, 86
STG2DACSignalGroup
Mcs::Usb, 72, 83
STG2SidebandsGroup
Mcs::Usb, 72, 84
STG2TriggerStatusGroup
Mcs::Usb, 72, 84
Stg3
Mcs::Usb, 66
STG3008_FA
Mcs::Usb, 79
Stg3008FilterAmpAmplificationEnumNet
Mcs::Usb, 86
STG4002
Mcs::Usb, 78
STG4002_opto
Mcs::Usb, 79
STG4004
Mcs::Usb, 78
STG4004_opto
Mcs::Usb, 79
STG4008
Mcs::Usb, 78
STG4008_opto
Mcs::Usb, 79
STG400x
Mcs::Usb, 78
STG400x_opto
Mcs::Usb, 79
STG5
Mcs::Usb, 79
STG_DestinationEnumNet
Mcs::Usb, 87
StgListModeTrigger
Mcs::Usb, 65
StgStatusNet, 719
FromIntPtr, 719
FromPtr, 719
ListOfChangedTriggers, 719
TiggerStatus, 719
StgTrigger
Mcs::Usb, 65
StillConnected
CRadioControledDevicesNet, 450
Stimulation
Mcs::Usb, 58
StimulationLayoutConfigurationEnumNet
Mcs::Usb, 87
Stimulator
CW2100_FunctionNet, 629
Stimulus
CCMOSMeaDeviceNet, 120
CStg200xDownloadBasicNet, 573
StimulusDeviceDataAndUnrolledData
CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, 720
StimulusFunction
CLIH3DeviceNet, 209
StimulusParameters
HeadStageIDType, 703
Stop
CMeaCleanDeviceNet, 352
CMeaCoatDeviceNet, 358
Mcs::Usb, 65, 77, 82
StopDacq
CMcsUsbDacqNet, 301
StopLoop
CMcsUsbDacqNet, 301
StopMCMovement
CMcsBus_MotorControlNet, 231
StopMovement
CRoboDeviceNet, 487
StopMovementI
CRoboStatorDeviceNet, 503
StopMovementXY

CRoboStatorDeviceNet, 503
StopMovementZ
CRoboStatorDeviceNet, 503
StopPlateClamp
CMultiwellDeviceNet, 396
StopPoll
CStimulusFunctionNet, 590
CW2100_StimulatorFunctionNet, 635
StopSampling
CTEERFunctionNet, 617
StopTable
CRoboDacqNet, 474, 475
StorageCharge
Mcs::Usb, 71
StoreValveTable
CWarnerValveControllerDeviceNet, 675
SubtractFromAll
Mcs::Usb, 81
SubtractFromAllOther
Mcs::Usb, 81
SubtractFromReferenceElectrodeOnly
Mcs::Usb, 81
SubtractionOff
Mcs::Usb, 81
SuperSpeed
Mcs::Usb, 72
Sw2to64
Mcs::Usb, 79
SwitchOnOff
CPositionIIDeviceNet, 424
SYNC_BIT0
CW2100_StimulatorFunctionNet, 635
SYNC_BIT1
CW2100_StimulatorFunctionNet, 635
SYNCOUT1
Mcs::Usb, 85
SYNCOUT2
Mcs::Usb, 85
SYNCOUT3
Mcs::Usb, 85
SYNCOUT4
Mcs::Usb, 85
SYNCOUT5
Mcs::Usb, 85
SYNCOUT6
Mcs::Usb, 85
SYNCOUT7
Mcs::Usb, 85
SYNCOUT8
Mcs::Usb, 85
syncoutdata
Mcs::Usb, 87
SyncStart
Mcs::Usb, 86

Table
Mcs::Usb, 95
Table_Wait
CRoboDacqNet, 475
TableDefBegin
CRoboDacqNet, 475
TableDefEnd
CRoboDacqNet, 475
TableEntryChangedEvent
CWarnerValveControllerDeviceNet, 678
TactSwitchGetState
CMcsBus_SensorNet, 241
TactSwitchSetDisplay
CMcsBus_SensorNet, 241
TBSI_127
Mcs::Usb, 53
TBSI_15
Mcs::Usb, 53
TBSI_31
Mcs::Usb, 53
TBSI_5
Mcs::Usb, 53
TBSI_63
Mcs::Usb, 53
TBSI_Dacq
Mcs::Usb, 80
TBSI_DACQDigitalSourceEnumNet
Mcs::Usb, 87
TBSI_Reserved
Mcs::Usb, 53
TbsiDacq
Mcs::Usb, 69
TbsiDacqHeadstage
Mcs::Usb, 68
TbsiDacqInterfaceboard
Mcs::Usb, 68
TC01
Mcs::Usb, 79
TC02
Mcs::Usb, 79
TCX
Mcs::Usb, 79
TcxDeviceTypeEnumNet
Mcs::Usb, 88
TcxSensorTypeEnumNet
Mcs::Usb, 88
TeerClampModeEnumNet
Mcs::Usb, 88
TEERFunctionNet
CTEERMachineDeviceNet, 618
TeerWaveformEnumNet
Mcs::Usb, 89
Tersens
Mcs::Usb, 78
Test_ADC_EPC10
Mcs::Usb, 70
Test_DAC_EPC10
Mcs::Usb, 70
ThrowCUsbExceptionNetOnError
CMcsUsbFunctionNet, 314
CMcsUsbNet, 337
TiggerStatus

StgStatusNet, 719
TimeResolutionInNanoSeconds
W2100_StimulusParametersNet, 722
Timestamp
Mcs::Usb, 61
ToCpp
CFilterCoefficientsNet::s_FilterAttributesNet, 716
ToString
CFilterPropertyNet, 146
CMcsUsbListEntryNet, 319
HeadStageIDType, 702
HeadstageIDTypeObject, 704
TouchTest
Mcs::Usb, 95
Triggerbox_AMS
Mcs::Usb, 79
Triggerbox_AMS3
Mcs::Usb, 79
Triggerbox_IMS
Mcs::Usb, 79
Triggerbox_R5
Mcs::Usb, 79
TriggerMask_Default
CRoboDacqNet, 475
TriggerOnly
Mcs::Usb, 86
TriggerSourceEnumNet
Mcs::Usb, 89
TriggerStatus
CMcsUsbDeviceStatePushFunctionNet, 303
CMcsUsbDeviceStatePushNet, 304
TriggerStatus1
Mcs::Usb, 60
TriggerStatus2
Mcs::Usb, 60
TriggerStatus3
Mcs::Usb, 60
TriggerStatus4
Mcs::Usb, 60
TriggerValue_MoveAbs
CRoboDacqNet, 475
TriggerValue_StartQueue
CRoboDacqNet, 475
tsAuxIn1
Mcs::Usb, 90
tsAuxIn2
Mcs::Usb, 90
tsDACQCy1Dev1Runs
Mcs::Usb, 91
tsDACQCy1Dev2Runs
Mcs::Usb, 91
tsDACQCy2Dev1Runs
Mcs::Usb, 91
tsDACQCy2Dev2Runs
Mcs::Usb, 91
tsDigitalIn1
Mcs::Usb, 89
tsDigitalIn10
Mcs::Usb, 89
tsDigitalIn11
Mcs::Usb, 89
tsDigitalIn12
Mcs::Usb, 89
tsDigitalIn13
Mcs::Usb, 89
tsDigitalIn14
Mcs::Usb, 89
tsDigitalIn15
Mcs::Usb, 89
tsDigitalIn16
Mcs::Usb, 89
tsDigitalIn17
Mcs::Usb, 89
tsDigitalIn18
Mcs::Usb, 89
tsDigitalIn19
Mcs::Usb, 89
tsDigitalIn2
Mcs::Usb, 89
tsDigitalIn20
Mcs::Usb, 89
tsDigitalIn21
Mcs::Usb, 89
tsDigitalIn22
Mcs::Usb, 90
tsDigitalIn23
Mcs::Usb, 90
tsDigitalIn24
Mcs::Usb, 90
tsDigitalIn25
Mcs::Usb, 90
tsDigitalIn26
Mcs::Usb, 90
tsDigitalIn27
Mcs::Usb, 90
tsDigitalIn28
Mcs::Usb, 90
tsDigitalIn29
Mcs::Usb, 90
tsDigitalIn3
Mcs::Usb, 89
tsDigitalIn30
Mcs::Usb, 90
tsDigitalIn31
Mcs::Usb, 90
tsDigitalIn32
Mcs::Usb, 90
tsDigitalIn4
Mcs::Usb, 89
tsDigitalIn5
Mcs::Usb, 89
tsDigitalIn6
Mcs::Usb, 89
tsDigitalIn7
Mcs::Usb, 89
tsDigitalIn8

Mcs::Usb, 89
tsDigitalIn9
Mcs::Usb, 89
tsDigitalPuse0
Mcs::Usb, 90
tsDigitalPuse1
Mcs::Usb, 90
tsDigitalPuse10
Mcs::Usb, 91
tsDigitalPuse11
Mcs::Usb, 91
tsDigitalPuse12
Mcs::Usb, 91
tsDigitalPuse13
Mcs::Usb, 91
tsDigitalPuse14
Mcs::Usb, 91
tsDigitalPuse15
Mcs::Usb, 91
tsDigitalPuse16
Mcs::Usb, 91
tsDigitalPuse17
Mcs::Usb, 91
tsDigitalPuse18
Mcs::Usb, 91
tsDigitalPuse19
Mcs::Usb, 91
tsDigitalPuse2
Mcs::Usb, 90
tsDigitalPuse20
Mcs::Usb, 91
tsDigitalPuse21
Mcs::Usb, 91
tsDigitalPuse22
Mcs::Usb, 91
tsDigitalPuse23
Mcs::Usb, 91
tsDigitalPuse24
Mcs::Usb, 91
tsDigitalPuse25
Mcs::Usb, 91
tsDigitalPuse26
Mcs::Usb, 91
tsDigitalPuse27
Mcs::Usb, 91
tsDigitalPuse28
Mcs::Usb, 91
tsDigitalPuse29
Mcs::Usb, 91
tsDigitalPuse3
Mcs::Usb, 90
tsDigitalPuse30
Mcs::Usb, 91
tsDigitalPuse31
Mcs::Usb, 91
tsDigitalPuse4
Mcs::Usb, 90
tsDigitalPuse5
Mcs::Usb, 91
tsDigitalPuse6
Mcs::Usb, 91
tsDigitalPuse7
Mcs::Usb, 91
tsDigitalPuse8
Mcs::Usb, 91
tsDigitalPuse9
Mcs::Usb, 91
tsFeedback1
Mcs::Usb, 90
tsFeedback10
Mcs::Usb, 90
tsFeedback11
Mcs::Usb, 90
tsFeedback12
Mcs::Usb, 90
tsFeedback13
Mcs::Usb, 90
tsFeedback14
Mcs::Usb, 90
tsFeedback15
Mcs::Usb, 90
tsFeedback16
Mcs::Usb, 90
tsFeedback17
Mcs::Usb, 90
tsFeedback18
Mcs::Usb, 90
tsFeedback19
Mcs::Usb, 90
tsFeedback2
Mcs::Usb, 90
tsFeedback20
Mcs::Usb, 90
tsFeedback21
Mcs::Usb, 90
tsFeedback22
Mcs::Usb, 90
tsFeedback23
Mcs::Usb, 90
tsFeedback24
Mcs::Usb, 90
tsFeedback25
Mcs::Usb, 90
tsFeedback26
Mcs::Usb, 90
tsFeedback27
Mcs::Usb, 90
tsFeedback28
Mcs::Usb, 90
tsFeedback29
Mcs::Usb, 90
tsFeedback3
Mcs::Usb, 90
tsFeedback30
Mcs::Usb, 90
tsFeedback31

Mcs::Usb, 90
tsFeedback32
Mcs::Usb, 90
tsFeedback4
Mcs::Usb, 90
tsFeedback5
Mcs::Usb, 90
tsFeedback6
Mcs::Usb, 90
tsFeedback7
Mcs::Usb, 90
tsFeedback8
Mcs::Usb, 90
tsFeedback9
Mcs::Usb, 90
tsNone
Mcs::Usb, 89
tsSidebandBit8
Mcs::Usb, 91
tsTriggered
Mcs::Usb, 91
TxnGetSerialNumber
CMcsUsbNet, 337
TxnSetSerialNumber
CMcsUsbNet, 337
TxnTestMemoryReadAndCheck
CMcsUsbNet, 337
TxnTestMemoryWrite
CMcsUsbNet, 337
Type
HeadStageIDType, 703
TypeValue
HeadStageIDType, 703

Unknown
HeadStageIDType, 701
Mcs::Usb, 53, 54, 67, 88, 91
unknown
Mcs::Usb, 66
UnknownDest
Mcs::Usb, 57
UnknownSpeed
Mcs::Usb, 72
Unlock
Mcs::Usb, 77
UnlockPlateClamp
CMultiwellDeviceNet, 396
UnrolledAmplitude
CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, 720
UnrolledDuration
CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, 720
UnrolledSync
CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, 720
Unsigned_16bit
Mcs::Usb, 63
Unsigned_24bit
Mcs::Usb, 63
Unsigned_32bit
Mcs::Usb, 63
UpdateChannelBlock
CCMOSMeaDeviceNet, 119
UpdateDisplay
CRoboDacqNet, 475
UpdateFirmware
CMcsUsbFactoryNet, 310–312
UpdateTransistorVoltages
CCMOSMea_FunctionNet, 116
UpdateTrigger
Mcs::Usb, 86
USB
FirmwareDestinationNames, 700
Mcs::Usb, 54
USB_TARGET1
Mcs::Usb, 57
USB_TARGET2
Mcs::Usb, 57
USB_TARGET3
Mcs::Usb, 57
usbSetupPacket_t, 721
bmRequestType, 721
bRequest, 721
wIndex, 721
wLength, 721
wValue, 721
UsbTest
Mcs::Usb, 80
UsbVendorIdEnumNet
Mcs::Usb, 91
User_ADC_0
Mcs::Usb, 69
User_ADC_1
Mcs::Usb, 69
User_ADC_2
Mcs::Usb, 70
User_ADC_3
Mcs::Usb, 70
User_ADC_4
Mcs::Usb, 70
User_DAC_0
Mcs::Usb, 70
User_DAC_1
Mcs::Usb, 70
User_DAC_2
Mcs::Usb, 70
UserDefinedName
HeadStageIDType, 703
UssingChamber
Mcs::Usb, 69
UssingClampModeEnumNet
Mcs::Usb, 92
UssingRail
Mcs::Usb, 69
UssingUnitEnumNet
Mcs::Usb, 92

Valid
HeadStageIDType, 703
ValidKey
CMcsUsbNet, 337
VendorIdEnumNet
Mcs::Usb, 92
VendorInRequest
CGenericDevelopDeviceNet, 172
VendorOutRequest
CGenericDevelopDeviceNet, 172
VirtualDevice_ContinousDacq
CRoboDacqNet, 475
VirtualDevice_TableRun
CRoboDacqNet, 475
Volt
Mcs::Usb, 54, 92
Voltage
BatteryState, 98
Voltage_3V3
Mcs::Usb, 69
Voltage_5V0
Mcs::Usb, 69
VoltageClamp
Mcs::Usb, 92
VoltageRangeDisplayStringMilliVolt
CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet, 621
VoltageRangeInMicroVolt
CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet, 621
W2100_StimulusParametersNet, 722
VoltageResolutionInMicroVolt
W2100_StimulusParametersNet, 722
VoltageString
BatteryState, 98
VOPSTimerSetResetTimes
CCMOSMea_FunctionNet, 116, 117

W16IsW14
HeadStageIDType, 703
W2100
Mcs::Usb, 80
W2100_Accel_Gyro_Select_EnumNet
Mcs::Usb, 92
W2100_FunctionNet
CMeaDeviceNet, 366
W2100_StimulusParametersNet, 721
CurrentRangeInNanoAmp, 722
CurrentResolutionInNanoAmp, 722
DACResolution, 722
TimeResolutionInNanoSeconds, 722
VoltageRangeInMicroVolt, 722
VoltageResolutionInMicroVolt, 722
W2100DacqGroupChannelEnumNet
Mcs::Usb, 93
W2100DigitalSourceEnumNet
Mcs::Usb, 94
W2100IFB2
Mcs::Usb, 68
W2100Interfaceboard
Mcs::Usb, 68
W2100WirelessReceiver
Mcs::Usb, 68, 69
W2100WirelessReceiverAnalog
Mcs::Usb, 68, 69
WaitForAllChambers
CWarnerUssingFunctionNet, 653
WaitForChamber
CWarnerUssingFunctionNet, 653
WaitTimer
CRoboDeviceNet, 487
Warner
Mcs::Usb, 88
Warner_TEER_Machine
Mcs::Usb, 81
Warner_Ussing
Mcs::Usb, 81
WARNER_USSING_DEVICE
Mcs::Usb, 64
Warner_Valve_Control
Mcs::Usb, 81
WARNER_VALVE_CONTROL_DEVICE
Mcs::Usb, 64
WarnerUssingFunction
CWarnerUssingDeviceNet, 637
WClassicFunctionNet
CMeaDeviceNet, 366
Whole_Cell_Patch
Mcs::Usb, 80
WholeCellPatch
Mcs::Usb, 83
WholeCellPatchHeadstage
Mcs::Usb, 68
WidthFrequency
CCreateFilterNet, 123
wIndex
usbSetupPacket_t, 721
WirelessHeadStageAccDataRE1HS1
Mcs::Usb, 93
WirelessHeadStageAccDataRE1HS2
Mcs::Usb, 93
WirelessHeadStageAccDataRE1HS3
Mcs::Usb, 93
WirelessHeadStageAccDataRE1HS4
Mcs::Usb, 93
WirelessHeadStageAccDataRE2HS1
Mcs::Usb, 93
WirelessHeadStageAccDataRE2HS2
Mcs::Usb, 93
WirelessHeadStageAccDataRE2HS3
Mcs::Usb, 93
WirelessHeadStageAccDataRE2HS4
Mcs::Usb, 93
WirelessHeadStageAnalogRE1HS1
Mcs::Usb, 93
WirelessHeadStageAnalogRE1HS2
Mcs::Usb, 93

WirelessHeadStageAnalogRE1HS3
Mcs::Usb, 93
WirelessHeadStageAnalogRE1HS4
Mcs::Usb, 93
WirelessHeadStageAnalogRE2HS1
Mcs::Usb, 93
WirelessHeadStageAnalogRE2HS2
Mcs::Usb, 93
WirelessHeadStageAnalogRE2HS3
Mcs::Usb, 93
WirelessHeadStageAnalogRE2HS4
Mcs::Usb, 93
WirelessHeadStageGyroDataRE1HS1
Mcs::Usb, 93
WirelessHeadStageGyroDataRE1HS2
Mcs::Usb, 93
WirelessHeadStageGyroDataRE1HS3
Mcs::Usb, 93
WirelessHeadStageGyroDataRE1HS4
Mcs::Usb, 93
WirelessHeadStageGyroDataRE2HS1
Mcs::Usb, 93
WirelessHeadStageGyroDataRE2HS2
Mcs::Usb, 93
WirelessHeadStageGyroDataRE2HS3
Mcs::Usb, 93
WirelessHeadStageGyroDataRE2HS4
Mcs::Usb, 93
WirelessHeadStageOptoStimCurrentRE1HS1
Mcs::Usb, 94
WirelessHeadStageOptoStimCurrentRE1HS2
Mcs::Usb, 94
WirelessHeadStageOptoStimCurrentRE1HS3
Mcs::Usb, 94
WirelessHeadStageOptoStimCurrentRE1HS4
Mcs::Usb, 94
WirelessHeadStageOptoStimCurrentRE2HS1
Mcs::Usb, 94
WirelessHeadStageOptoStimCurrentRE2HS2
Mcs::Usb, 94
WirelessHeadStageOptoStimCurrentRE2HS3
Mcs::Usb, 94
WirelessHeadStageOptoStimCurrentRE2HS4
Mcs::Usb, 94
WirelessHeadStageReservedARE1HS1
Mcs::Usb, 94
WirelessHeadStageReservedARE1HS2
Mcs::Usb, 94
WirelessHeadStageReservedARE1HS3
Mcs::Usb, 94
WirelessHeadStageReservedARE1HS4
Mcs::Usb, 94
WirelessHeadStageReservedARE2HS1
Mcs::Usb, 94
WirelessHeadStageReservedARE2HS2
Mcs::Usb, 94
WirelessHeadStageReservedARE2HS3
Mcs::Usb, 94
WirelessHeadStageReservedARE2HS4
Mcs::Usb, 94
WirelessHeadStageReservedBRE1HS1
Mcs::Usb, 94
WirelessHeadStageReservedBRE1HS2
Mcs::Usb, 94
WirelessHeadStageReservedBRE1HS3
Mcs::Usb, 94
WirelessHeadStageReservedBRE1HS4
Mcs::Usb, 94
WirelessHeadStageReservedBRE2HS1
Mcs::Usb, 94
WirelessHeadStageReservedBRE2HS2
Mcs::Usb, 94
WirelessHeadStageReservedBRE2HS3
Mcs::Usb, 94
WirelessHeadStageReservedBRE2HS4
Mcs::Usb, 94
WirelessHeadStageReservedCRE1HS1
Mcs::Usb, 94
WirelessHeadStageReservedCRE1HS2
Mcs::Usb, 94
WirelessHeadStageReservedCRE1HS3
Mcs::Usb, 94
WirelessHeadStageReservedCRE1HS4
Mcs::Usb, 94
WirelessHeadStageReservedCRE2HS1
Mcs::Usb, 94
WirelessHeadStageReservedCRE2HS2
Mcs::Usb, 94
WirelessHeadStageReservedCRE2HS3
Mcs::Usb, 94
WirelessHeadStageReservedCRE2HS4
Mcs::Usb, 94
WirelessHeadStageStatusRE1HS1
Mcs::Usb, 93
WirelessHeadStageStatusRE1HS2
Mcs::Usb, 93
WirelessHeadStageStatusRE1HS3
Mcs::Usb, 93
WirelessHeadStageStatusRE1HS4
Mcs::Usb, 93
WirelessHeadStageStatusRE2HS1
Mcs::Usb, 93
WirelessHeadStageStatusRE2HS2
Mcs::Usb, 93
WirelessHeadStageStatusRE2HS3
Mcs::Usb, 93
WirelessHeadStageStatusRE2HS4
Mcs::Usb, 93
WirelessTestAdapter
Mcs::Usb, 53
wLength
usbSetupPacket_t, 721
Work
Mcs::Usb, 95
WPA16
Mcs::Usb, 80

WPA32
Mcs::Usb, 80
WPA4
Mcs::Usb, 80
WPA8
Mcs::Usb, 80
WPAError_ScanningIsPending
CMcsUsbNet, 343
Write
CExternDTesterDeviceNet, 139
Write2
CExternDTesterDeviceNet, 139
WriteEepromRegisterPreconfig
CMcsUsbNet, 338
WritePipe
CGenericDevelopDeviceNet, 172
WriteRegister
CMcsUsbNet, 338
WriteRegister32
CMcsUsbNet, 338
WriteRegisterArray
CMcsUsbNet, 338
WriteRegisterTimeSlot
CMcsUsbNet, 339
WriteRegisterValue
CMcsUsbNet, 339
WriteUARTData
CLIH3DeviceNet, 209
wValue
usbSetupPacket_t, 721
WvcDisplayModeEnumNet
Mcs::Usb, 95
WvcValveModeEnumNet
Mcs::Usb, 95

Zero
Mcs::Usb, 57, 73, 74, 76, 84, 88, 95