multichannel systems

McsUsbNet.dll

Version 5.1.23

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 . . . 1
1.3 Connecting to an MCS device . . . 1

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

3 Function Classes 3

4 Data ACQuisition (DACQ) Devices 4

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

6 STG200x & STG400x STimulus Generator 5
6.1 Introduction . . . 5
6.2 Download mode . . . 6
6.2.1 Memory Layout and Trigger Setup . . . 6
6.3 Streaming mode . . . 8
6.3.1 Memory Layout and Trigger Setup . . . 9

7 Namespace Index 11
7.1 Namespace List . . . 11

8 Hierarchical Index 11
8.1 Class Hierarchy . . . 11

9 Class Index 16
9.1 Class List . . . 16

10 Namespace Documentation 22
10.1 Mcs Namespace Reference . . . 22
10.2 Mcs::Usb Namespace Reference . . . 22
10.2.1 Enumeration Type Documentation . . . 50
10.2.2 Function Documentation . . . 90

11 Class Documentation 92
11.1 CW2100_FunctionNet::AudioChannelsNet Struct Reference . . . 92
11.1.1 Member Data Documentation . . . 92
11.2 BatteryState Class Reference . . . 92
11.2.1 Property Documentation . . . 92
11.3 BesselFilterHighPassNet Class Reference . . . 93

11.3.1 Constructor & Destructor Documentation . . . 93
11.4 BesselFilterLowPassNet Class Reference . . . 93
11.4.1 Constructor & Destructor Documentation . . . 94
11.5 ButterworthFilterHighPassNet Class Reference . . . 94
11.5.1 Constructor & Destructor Documentation . . . 94
11.6 ButterworthFilterLowPassNet Class Reference . . . 95
11.6.1 Constructor & Destructor Documentation . . . 95
11.7 CChannelTestDeviceNet Class Reference . . . 95
11.7.1 Constructor & Destructor Documentation . . . 96
11.7.2 Member Function Documentation . . . 96
11.8 CCMOSMea_FunctionNet Class Reference . . . 96
11.8.1 Constructor & Destructor Documentation . . . 98
11.8.2 Member Function Documentation . . . 99
11.9 CCMOSMeaDeviceNet Class Reference . . . 107
11.9.1 Constructor & Destructor Documentation . . . 108
11.9.2 Member Function Documentation . . . 108
11.9.3 Property Documentation . . . 110
11.10 CCreateFilterNet Class Reference . . . 110
11.10.1 Constructor & Destructor Documentation . . . 111
11.10.2 Member Function Documentation . . . 111
11.10.3 Property Documentation . . . 112
11.11 CDacCalibrationFunctionNet Class Reference . . . 112
11.11.1 Detailed Description . . . 113
11.11.2 Constructor & Destructor Documentation . . . 113
11.11.3 Member Function Documentation . . . 113
11.12 CDacqGroupChannelGenericSelectionNet Class Reference . . . 114
11.12.1 Constructor & Destructor Documentation . . . 115
11.13 CDacqGroupChannelSelectionNet Class Reference . . . 115
11.13.1 Constructor & Destructor Documentation . . . 115
11.14 CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet > Class Template Reference 115
11.14.1 Constructor & Destructor Documentation . . . 116
11.14.2 Member Function Documentation . . . 116
11.15 CDeviceGroupChannelInfoGenericNet Class Reference . . . 118
11.15.1 Constructor & Destructor Documentation . . . 118
11.16 CDeviceGroupChannelInfoMEA2100_256Net Class Reference . . . 119
11.16.1 Constructor & Destructor Documentation . . . 119
11.17 CDeviceGroupChannelInfoNet Class Reference . . . 119
11.17.1 Constructor & Destructor Documentation . . . 120
11.18 CDeviceGroupChannelInfoSCUNet Class Reference . . . 120
11.18.1 Constructor & Destructor Documentation . . . 120
11.19 CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumTemplateNet > Class Template Reference . . . 120

11.19.1 Constructor & Destructor Documentation . . . 121
11.19.2 Member Data Documentation . . . 121
11.20 CDeviceGroupChannelInfoW2100Net Class Reference . . . 121
11.20.1 Constructor & Destructor Documentation . . . 122
11.21 CDigOutStimulatorFunctionNet Class Reference . . . 122
11.21.1 Detailed Description . . . 123
11.21.2 Constructor & Destructor Documentation . . . 123
11.21.3 Member Function Documentation . . . 123
11.22 CEncapsulatorDeviceNet Class Reference . . . 126
11.22.1 Detailed Description . . . 126
11.22.2 Constructor & Destructor Documentation . . . 127
11.22.3 Member Function Documentation . . . 127
11.23 CExternDTesterDeviceNet Class Reference . . . 127
11.23.1 Detailed Description . . . 127
11.23.2 Constructor & Destructor Documentation . . . 128
11.23.3 Member Function Documentation . . . 128
11.24 CFilterCoefficientsNet Class Reference . . . 129
11.24.1 Constructor & Destructor Documentation . . . 129
11.24.2 Member Function Documentation . . . 130
11.24.3 Property Documentation . . . 131
11.25 CFilterConfigurationNet Class Reference . . . 131
11.25.1 Constructor & Destructor Documentation . . . 131
11.25.2 Member Function Documentation . . . 132
11.26 CFilterConfigurationRegisterNet Class Reference . . . 133
11.26.1 Constructor & Destructor Documentation . . . 133
11.26.2 Member Function Documentation . . . 133
11.27 CFilterPropertyNet Class Reference . . . 135
11.27.1 Constructor & Destructor Documentation . . . 135
11.27.2 Member Function Documentation . . . 135
11.27.3 Property Documentation . . . 135
11.28 CFluidControlDeviceNet Class Reference . . . 136
11.28.1 Detailed Description . . . 137
11.28.2 Constructor & Destructor Documentation . . . 138
11.28.3 Member Function Documentation . . . 138
11.28.4 Property Documentation . . . 143
11.29 CFYIDeviceNet Class Reference . . . 143
11.29.1 Detailed Description . . . 144
11.29.2 Constructor & Destructor Documentation . . . 144
11.29.3 Property Documentation . . . 144
11.30 CGenericDevelopDeviceNet Class Reference . . . 145
11.30.1 Detailed Description . . . 151
11.30.2 Constructor & Destructor Documentation . . . 152

11.30.3 Member Function Documentation . . . 152
11.31 CGilsonDeviceNet Class Reference . . . 163
11.31.1 Detailed Description . . . 164
11.31.2 Constructor & Destructor Documentation . . . 164
11.31.3 Member Function Documentation . . . 164
11.31.4 Member Data Documentation . . . 165
11.32 CGrapheneFunctionNet Class Reference . . . 165
11.32.1 Detailed Description . . . 167
11.32.2 Constructor & Destructor Documentation . . . 167
11.32.3 Member Function Documentation . . . 168
11.33 CHiClampDeviceNet Class Reference . . . 176
11.33.1 Detailed Description . . . 177
11.33.2 Constructor & Destructor Documentation . . . 177
11.33.3 Property Documentation . . . 177
11.34 CHLADacqNet Class Reference . . . 177
11.34.1 Constructor & Destructor Documentation . . . 178
11.35 CHLADeviceNet Class Reference . . . 178
11.35.1 Detailed Description . . . 178
11.35.2 Constructor & Destructor Documentation . . . 178
11.35.3 Property Documentation . . . 179
11.36 CMcsUsbDacqNet::CHWInfo Class Reference . . . 179
11.36.1 Detailed Description . . . 179
11.36.2 Constructor & Destructor Documentation . . . 180
11.36.3 Member Function Documentation . . . 180
11.37 CIntanMea_FunctionNet Class Reference . . . 182
11.37.1 Constructor & Destructor Documentation . . . 182
11.37.2 Member Function Documentation . . . 183
11.38 CInterfaceboard2FunctionNet Class Reference . . . 184
11.38.1 Detailed Description . . . 184
11.38.2 Constructor & Destructor Documentation . . . 185
11.38.3 Member Function Documentation . . . 185
11.39 CInterfaceboardFunctionNet Class Reference . . . 186
11.39.1 Detailed Description . . . 186
11.39.2 Constructor & Destructor Documentation . . . 186
11.39.3 Member Function Documentation . . . 187
11.40 CLIH3DeviceNet Class Reference . . . 188
11.40.1 Detailed Description . . . 190
11.40.2 Constructor & Destructor Documentation . . . 190
11.40.3 Member Function Documentation . . . 190
11.40.4 Property Documentation . . . 197
11.41 CMcsBus_AxisParametersNet Class Reference . . . 197
11.41.1 Constructor & Destructor Documentation . . . 198

11.41.2 Member Function Documentation . . . 198
11.42 CMcsBus_ExtensionNet Class Reference . . . 199
11.42.1 Constructor & Destructor Documentation . . . 199
11.42.2 Member Function Documentation . . . 200
11.43 CMcsBus_FYIExtensionNet Class Reference . . . 200
11.43.1 Constructor & Destructor Documentation . . . 200
11.43.2 Member Function Documentation . . . 201
11.44 CMcsBus_MotorControlNet Class Reference . . . 202
11.44.1 Constructor & Destructor Documentation . . . 205
11.44.2 Member Function Documentation . . . 205
11.45 CMcsBus_SensorNet Class Reference . . . 219
11.45.1 Constructor & Destructor Documentation . . . 221
11.45.2 Member Function Documentation . . . 221
11.46 CMcsBus_TempSensorNet Class Reference . . . 229
11.46.1 Constructor & Destructor Documentation . . . 230
11.46.2 Member Function Documentation . . . 230
11.47 CMcsBus_VoltageModeNet Class Reference . . . 231
11.47.1 Constructor & Destructor Documentation . . . 232
11.47.2 Member Function Documentation . . . 233
11.48 CMcsBusNet Class Reference . . . 236
11.48.1 Constructor & Destructor Documentation . . . 236
11.48.2 Member Function Documentation . . . 237
11.49 CMcsUsbDacqNet Class Reference . . . 240
11.49.1 Detailed Description . . . 245
11.49.2 Constructor & Destructor Documentation . . . 246
11.49.3 Member Function Documentation . . . 246
11.49.4 Member Data Documentation . . . 286
11.49.5 Property Documentation . . . 287
11.49.6 Event Documentation . . . 287
11.50 CMcsUsbDeviceStatePushFunctionNet Class Reference . . . 287
11.50.1 Constructor & Destructor Documentation . . . 288
11.50.2 Member Function Documentation . . . 288
11.50.3 Event Documentation . . . 288
11.51 CMcsUsbDeviceStatePushNet Class Reference . . . 288
11.51.1 Constructor & Destructor Documentation . . . 289
11.51.2 Member Function Documentation . . . 289
11.51.3 Event Documentation . . . 289
11.52 CMcsUsbFactoryNet Class Reference . . . 289
11.52.1 Constructor & Destructor Documentation . . . 291
11.52.2 Member Function Documentation . . . 291
11.52.3 Member Data Documentation . . . 297
11.53 CMcsUsbFunctionNet Class Reference . . . 298

11.53.1 Constructor & Destructor Documentation . . . 298
11.53.2 Member Function Documentation . . . 299
11.53.3 Member Data Documentation . . . 299
11.54 CMcsUsbFunctionPointerContainer Class Reference . . . 299
11.55 CMcsUsbListEntryNet Class Reference . . . 299
11.55.1 Detailed Description . . . 300
11.55.2 Constructor & Destructor Documentation . . . 300
11.55.3 Member Function Documentation . . . 301
11.55.4 Property Documentation . . . 304
11.56 CMcsUsbListNet Class Reference . . . 305
11.56.1 Detailed Description . . . 305
11.56.2 Constructor & Destructor Documentation . . . 305
11.56.3 Member Function Documentation . . . 306
11.56.4 Property Documentation . . . 307
11.56.5 Event Documentation . . . 307
11.57 CMcsUsbNet Class Reference . . . 308
11.57.1 Detailed Description . . . 312
11.57.2 Constructor & Destructor Documentation . . . 312
11.57.3 Member Function Documentation . . . 312
11.57.4 Member Data Documentation . . . 324
11.57.5 Property Documentation . . . 329
11.58 CMcsUsbPointerContainer Class Reference . . . 329
11.59 CMEA2100_256DacqGroupChannelSelectionNet Class Reference . . . 329
11.59.1 Constructor & Destructor Documentation . . . 329
11.60 CMEA2100x256FunctionNet Class Reference . . . 330
11.60.1 Detailed Description . . . 330
11.60.2 Constructor & Destructor Documentation . . . 330
11.60.3 Member Function Documentation . . . 331
11.61 CMeaAudioFunctionNet Class Reference . . . 331
11.61.1 Constructor & Destructor Documentation . . . 332
11.61.2 Member Function Documentation . . . 332
11.62 CMeaCleanDeviceNet Class Reference . . . 334
11.62.1 Detailed Description . . . 335
11.62.2 Constructor & Destructor Documentation . . . 335
11.62.3 Member Function Documentation . . . 336
11.63 CMeaCoatDeviceNet Class Reference . . . 338
11.63.1 Detailed Description . . . 339
11.63.2 Constructor & Destructor Documentation . . . 340
11.63.3 Member Function Documentation . . . 340
11.64 CMeaDeviceNet Class Reference . . . 344
11.64.1 Detailed Description . . . 345
11.64.2 Constructor & Destructor Documentation . . . 345

11.64.3 Member Function Documentation . . . 346
11.64.4 Property Documentation . . . 351
11.65 CMeaDigitalDataFunctionNet Class Reference . . . 352
11.65.1 Constructor & Destructor Documentation . . . 352
11.65.2 Member Function Documentation . . . 353
11.66 CMeaFeedbackFunctionNet Class Reference . . . 354
11.66.1 Constructor & Destructor Documentation . . . 355
11.66.2 Member Function Documentation . . . 355
11.67 CMeaImpedanceDeviceNet Class Reference . . . 358
11.67.1 Constructor & Destructor Documentation . . . 358
11.67.2 Member Function Documentation . . . 359
11.68 CMeasureTableDeviceNet Class Reference . . . 359
11.68.1 Detailed Description . . . 360
11.68.2 Constructor & Destructor Documentation . . . 360
11.68.3 Property Documentation . . . 360
11.69 CMeaSwitchDeviceNet Class Reference . . . 360
11.69.1 Detailed Description . . . 361
11.69.2 Constructor & Destructor Documentation . . . 361
11.69.3 Member Function Documentation . . . 361
11.70 CMeaUSBDeviceNet Class Reference . . . 362
11.70.1 Detailed Description . . . 363
11.70.2 Constructor & Destructor Documentation . . . 363
11.71 CMeFunctionNet Class Reference . . . 363
11.71.1 Detailed Description . . . 364
11.71.2 Constructor & Destructor Documentation . . . 364
11.71.3 Member Function Documentation . . . 364
11.72 CMultiBatteryChargerDeviceNet Class Reference . . . 365
11.72.1 Detailed Description . . . 366
11.72.2 Constructor & Destructor Documentation . . . 366
11.72.3 Member Function Documentation . . . 366
11.73 CMultiwellCallbackFunctionNet Class Reference . . . 372
11.73.1 Detailed Description . . . 372
11.73.2 Constructor & Destructor Documentation . . . 373
11.73.3 Member Function Documentation . . . 373
11.73.4 Event Documentation . . . 374
11.74 CMultiwellDeviceNet Class Reference . . . 374
11.74.1 Detailed Description . . . 375
11.74.2 Constructor & Destructor Documentation . . . 375
11.74.3 Member Function Documentation . . . 376
11.75 CMultiwellOptoStimFunctionNet Class Reference . . . 380
11.75.1 Detailed Description . . . 381
11.75.2 Constructor & Destructor Documentation . . . 381

11.75.3 Member Function Documentation . . . 381
11.76 CNF_GenDeviceNet Class Reference . . . 385
11.76.1 Constructor & Destructor Documentation . . . 385
11.76.2 Member Function Documentation . . . 385
11.77 COctoPotDeviceNet Class Reference . . . 386
11.77.1 Constructor & Destructor Documentation . . . 386
11.77.2 Member Function Documentation . . . 387
11.78 COkuvisionStimulatorDeviceNet Class Reference . . . 390
11.78.1 Constructor & Destructor Documentation . . . 391
11.78.2 Member Function Documentation . . . 391
11.79 CPatchServerDeviceNet Class Reference . . . 393
11.79.1 Detailed Description . . . 394
11.79.2 Constructor & Destructor Documentation . . . 394
11.79.3 Property Documentation . . . 394
11.80 CPathIdentDeviceNet Class Reference . . . 394
11.80.1 Constructor & Destructor Documentation . . . 395
11.80.2 Member Function Documentation . . . 395
11.81 CPedoterDeviceNet Class Reference . . . 395
11.81.1 Detailed Description . . . 396
11.81.2 Constructor & Destructor Documentation . . . 396
11.81.3 Member Function Documentation . . . 396
11.82 CPeristalticPumpDeviceNet Class Reference . . . 397
11.82.1 Detailed Description . . . 397
11.82.2 Constructor & Destructor Documentation . . . 397
11.82.3 Property Documentation . . . 398
11.83 CPgaDeviceNet Class Reference . . . 398
11.83.1 Constructor & Destructor Documentation . . . 398
11.83.2 Member Function Documentation . . . 399
11.84 CPositionIIDeviceNet Class Reference . . . 400
11.84.1 Detailed Description . . . 402
11.84.2 Constructor & Destructor Documentation . . . 402
11.84.3 Member Function Documentation . . . 402
11.84.4 Property Documentation . . . 409
11.85 CPositionImpDeviceNet Class Reference . . . 409
11.85.1 Detailed Description . . . 409
11.85.2 Constructor & Destructor Documentation . . . 410
11.85.3 Member Function Documentation . . . 410
11.86 CPPCDeviceNet Class Reference . . . 412
11.86.1 Constructor & Destructor Documentation . . . 412
11.86.2 Property Documentation . . . 413
11.87 CPPCFunctionNet Class Reference . . . 413
11.87.1 Detailed Description . . . 414

11.87.2 Constructor & Destructor Documentation . . . 414
11.87.3 Member Function Documentation . . . 415
11.88 CPPS_DeviceNet Class Reference . . . 422
11.88.1 Constructor & Destructor Documentation . . . 422
11.88.2 Property Documentation . . . 422
11.89 CPPS_FunctionNet Class Reference . . . 423
11.89.1 Constructor & Destructor Documentation . . . 423
11.89.2 Member Function Documentation . . . 424
11.90 CPPSDeviceNet Class Reference . . . 427
11.90.1 Detailed Description . . . 427
11.90.2 Constructor & Destructor Documentation . . . 428
11.91 CProgramPressureCurveNet Class Reference . . . 428
11.91.1 Detailed Description . . . 428
11.91.2 Constructor & Destructor Documentation . . . 428
11.91.3 Member Function Documentation . . . 429
11.92 CPulseGeneratorFunctionNet Class Reference . . . 429
11.92.1 Detailed Description . . . 430
11.92.2 Constructor & Destructor Documentation . . . 430
11.92.3 Member Function Documentation . . . 431
11.93 CRadioControledDevicesNet Class Reference . . . 432
11.93.1 Constructor & Destructor Documentation . . . 433
11.93.2 Member Function Documentation . . . 433
11.94 CCMOSMeaDeviceNet::CRegionOfInterestRect Class Reference . . . 434
11.94.1 Constructor & Destructor Documentation . . . 434
11.94.2 Member Function Documentation . . . 434
11.94.3 Member Data Documentation . . . 435
11.95 CRetinaLedDeviceNet Class Reference . . . 435
11.95.1 Constructor & Destructor Documentation . . . 436
11.95.2 Member Function Documentation . . . 436
11.96 CRFFunctionNet Class Reference . . . 437
11.96.1 Detailed Description . . . 438
11.96.2 Constructor & Destructor Documentation . . . 438
11.96.3 Member Function Documentation . . . 438
11.97 CRobo_FYIProgram_FunctionNet Class Reference . . . 442
11.97.1 Constructor & Destructor Documentation . . . 442
11.97.2 Member Function Documentation . . . 442
11.98 CRobo_FYITemp_FunctionNet Class Reference . . . 443
11.98.1 Constructor & Destructor Documentation . . . 444
11.98.2 Member Function Documentation . . . 444
11.99 CRoboDacqNet Class Reference . . . 445
11.99.1 Constructor & Destructor Documentation . . . 448
11.99.2 Member Function Documentation . . . 448

11.99.3 Member Data Documentation . . . 459
11.100 CRoboDeviceNet Class Reference . . . 460
11.100.1 Detailed Description . . . 463
11.100.2 Constructor & Destructor Documentation . . . 463
11.100.3 Member Function Documentation . . . 463
11.100.4 Member Data Documentation . . . 471
11.100.5 Property Documentation . . . 475
11.100.6 Event Documentation . . . 476
11.101 CRoboFluidDeviceNet Class Reference . . . 476
11.101.1 Constructor & Destructor Documentation . . . 477
11.101.2 Member Function Documentation . . . 477
11.101.3 Member Data Documentation . . . 479
11.101.4 Property Documentation . . . 479
11.102 CRoboInjectDeviceNet Class Reference . . . 479
11.102.1 Detailed Description . . . 480
11.102.2 Constructor & Destructor Documentation . . . 480
11.103 CRoboocyte2DeviceNet Class Reference . . . 480
11.103.1 Detailed Description . . . 480
11.103.2 Constructor & Destructor Documentation . . . 481
11.103.3 Member Function Documentation . . . 481
11.104 CRoboStatorDeviceNet Class Reference . . . 481
11.104.1 Constructor & Destructor Documentation . . . 483
11.104.2 Member Function Documentation . . . 483
11.104.3 Property Documentation . . . 487
11.105 CSafeISDeviceNet Class Reference . . . 487
11.105.1 Detailed Description . . . 488
11.105.2 Constructor & Destructor Documentation . . . 488
11.105.3 Member Function Documentation . . . 489
11.105.4 Property Documentation . . . 490
11.106 CSCUDacqGroupChannelSelectionNet Class Reference . . . 490
11.106.1 Constructor & Destructor Documentation . . . 491
11.107 CSCUFunctionNet Class Reference . . . 491
11.107.1 Detailed Description . . . 493
11.107.2 Constructor & Destructor Documentation . . . 493
11.107.3 Member Function Documentation . . . 494
11.107.4 Event Documentation . . . 505
11.108 CSerialPortNet Class Reference . . . 506
11.108.1 Constructor & Destructor Documentation . . . 506
11.108.2 Member Function Documentation . . . 506
11.109 CStg200xBasicNet Class Reference . . . 507
11.109.1 Detailed Description . . . 511
11.109.2 Constructor & Destructor Documentation . . . 512

11.109.3 Member Function Documentation . . . 512
11.110 CStg200xDownloadBasicNet Class Reference . . . 541
11.110.1 Detailed Description . . . 542
11.110.2 Member Function Documentation . . . 542
11.110.3 Property Documentation . . . 548
11.111 CStg200xDownloadNet Class Reference . . . 548
11.111.1 Detailed Description . . . 549
11.111.2 Constructor & Destructor Documentation . . . 549
11.111.3 Member Function Documentation . . . 550
11.111.4 Event Documentation . . . 554
11.112 CStimulusFunctionNet Class Reference . . . 555
11.112.1 Constructor & Destructor Documentation . . . 556
11.112.2 Member Function Documentation . . . 557
11.112.3 Event Documentation . . . 566
11.113 CSw2to64DeviceNet Class Reference . . . 566
11.113.1 Detailed Description . . . 566
11.113.2 Constructor & Destructor Documentation . . . 567
11.113.3 Member Function Documentation . . . 567
11.114 CTcxDeviceNet Class Reference . . . 568
11.114.1 Detailed Description . . . 570
11.114.2 Constructor & Destructor Documentation . . . 570
11.114.3 Member Function Documentation . . . 571
11.115 CTEERFunctionNet Class Reference . . . 581
11.115.1 Detailed Description . . . 583
11.115.2 Constructor & Destructor Documentation . . . 583
11.115.3 Member Function Documentation . . . 584
11.116 CTEERMachineDeviceNet Class Reference . . . 592
11.116.1 Constructor & Destructor Documentation . . . 592
11.116.2 Property Documentation . . . 593
11.117 CUsbDeviceConfigurationFunctionNet Class Reference . . . 593
11.117.1 Detailed Description . . . 593
11.117.2 Constructor & Destructor Documentation . . . 593
11.117.3 Member Function Documentation . . . 594
11.118 CUsbExceptionNet Class Reference . . . 594
11.118.1 Detailed Description . . . 595
11.118.2 Constructor & Destructor Documentation . . . 595
11.118.3 Property Documentation . . . 595
11.119 CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet Class Reference . . . 595
11.119.1 Constructor & Destructor Documentation . . . 596
11.119.2 Member Data Documentation . . . 596
11.120 CW2100_FunctionNet Class Reference . . . 596
11.120.1 Constructor & Destructor Documentation . . . 598

11.120.2 Member Function Documentation . . . 598
11.120.3 Property Documentation . . . 603
11.121 CW2100_StimulatorFunctionNet Class Reference . . . 604
11.121.1 Constructor & Destructor Documentation . . . 605
11.121.2 Member Function Documentation . . . 605
11.121.3 Member Data Documentation . . . 610
11.121.4 Event Documentation . . . 610
11.122 CW2100DacqGroupChannelSelectionNet Class Reference . . . 610
11.122.1 Constructor & Destructor Documentation . . . 610
11.123 CWarnerUssingDeviceNet Class Reference . . . 611
11.123.1 Detailed Description . . . 611
11.123.2 Constructor & Destructor Documentation . . . 611
11.123.3 Property Documentation . . . 612
11.124 CWarnerUssingFunctionNet Class Reference . . . 612
11.124.1 Detailed Description . . . 614
11.124.2 Constructor & Destructor Documentation . . . 614
11.124.3 Member Function Documentation . . . 614
11.125 CWarnerValveControllerDeviceNet Class Reference . . . 628
11.125.1 Detailed Description . . . 632
11.125.2 Constructor & Destructor Documentation . . . 633
11.125.3 Member Function Documentation . . . 633
11.125.4 Event Documentation . . . 650
11.126 CWarnerValveControllerDeviceTesterFunctionNet Class Reference . . . 653
11.126.1 Detailed Description . . . 654
11.126.2 Constructor & Destructor Documentation . . . 654
11.126.3 Member Function Documentation . . . 655
11.127 CWClassicFunctionNet Class Reference . . . 656
11.127.1 Constructor & Destructor Documentation . . . 657
11.127.2 Member Function Documentation . . . 658
11.128 CWirelessBaseFunctionNet Class Reference . . . 661
11.128.1 Constructor & Destructor Documentation . . . 662
11.128.2 Member Function Documentation . . . 662
11.129 DeviceIdNet Struct Reference . . . 662
11.129.1 Detailed Description . . . 662
11.129.2 Constructor & Destructor Documentation . . . 663
11.129.3 Member Function Documentation . . . 663
11.129.4 Member Data Documentation . . . 663
11.130 DigitalSource< digitalsourceenum > Class Template Reference . . . 664
11.130.1 Constructor & Destructor Documentation . . . 664
11.130.2 Member Function Documentation . . . 664
11.130.3 Property Documentation . . . 665
11.131 DigitalSourceGeneral Class Reference . . . 665

11.131.1 Constructor & Destructor Documentation . . . 665
11.131.2 Member Function Documentation . . . 665
11.131.3 Property Documentation . . . 666
11.132 DriverVersionNet Class Reference . . . 666
11.132.1 Detailed Description . . . 667
11.132.2 Constructor & Destructor Documentation . . . 667
11.132.3 Member Function Documentation . . . 667
11.133 FirmwareDestinationNames Class Reference . . . 672
11.133.1 Member Data Documentation . . . 673
11.134 HeadStageIDType Class Reference . . . 675
11.134.1 Member Enumeration Documentation . . . 676
11.134.2 Constructor & Destructor Documentation . . . 676
11.134.3 Member Function Documentation . . . 677
11.134.4 Property Documentation . . . 677
11.135 HeadstageIDTypeObject Class Reference . . . 678
11.135.1 Constructor & Destructor Documentation . . . 679
11.135.2 Member Function Documentation . . . 679
11.135.3 Member Data Documentation . . . 679
11.135.4 Property Documentation . . . 680
11.136 HeadStageIDTypeState Class Reference . . . 680
11.136.1 Property Documentation . . . 680
11.137 mkfilterNet Class Reference . . . 681
11.137.1 Member Function Documentation . . . 681
11.138 CRoboDeviceNet::RoboMainLowLevelCommands Class Reference . . . 684
11.138.1 Member Function Documentation . . . 685
11.139 CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands Class Reference . . . 691
11.139.1 Member Function Documentation . . . 691
11.140 CFilterCoefficientsNet::s_FilterAttributesNet Struct Reference . . . 691
11.140.1 Constructor & Destructor Documentation . . . 691
11.140.2 Member Function Documentation . . . 691
11.140.3 Member Data Documentation . . . 692
11.141 CMeaAudioFunctionNet::s_setaudionet Struct Reference . . . 692
11.141.1 Member Data Documentation . . . 692
11.142 CStimulusFunctionNet::SidebandData Class Reference . . . 693
11.142.1 Constructor & Destructor Documentation . . . 693
11.142.2 Property Documentation . . . 693
11.143 StgStatusNet Class Reference . . . 694
11.143.1 Member Function Documentation . . . 694
11.143.2 Member Data Documentation . . . 694
11.144 CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData Class Reference . . . 694
11.144.1 Constructor & Destructor Documentation . . . 695
11.144.2 Property Documentation . . . 695

11.145 usbSetupPacket_t Class Reference 696
11.145.1 Member Data Documentation 696
11.146 W2100_StimulusParametersNet Struct Reference 696
11.146.1 Member Data Documentation 697

Index **699**

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.SendSegmentDefine the segment sizes are assigned.

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

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

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

```
}
```

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

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

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

```
// Only 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 22

Mcs::Usb 22

8 Hierarchical Index

8.1 Class Hierarchy

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

CW2100_FunctionNet::AudioChannelsNet 92

BatteryState 92

CCreateFilterNet 110

- **BesselFilterHighPassNet** 93
- **BesselFilterLowPassNet** 93
- **ButterworthFilterHighPassNet** 94
- **ButterworthFilterLowPassNet** 95

CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumTemplateNet > 120

CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumNet > 120

- **CDeviceGroupChannelInfoNet** 119

CDeviceGroupChannelInfoTemplateNet< int > 120

- **CDeviceGroupChannelInfoGenericNet** 118

CDeviceGroupChannelInfoTemplateNet< MEA2100_256DacqGroupChannelEnumNet > 120

CDeviceGroupChannelInfoMEA2100_256Net 119

CDeviceGroupChannelInfoTemplateNet< SCUDacqGroupChannelEnumNet > 120

CDeviceGroupChannelInfoSCUNet 120

CDeviceGroupChannelInfoTemplateNet< W2100DacqGroupChannelEnumNet > 120

CDeviceGroupChannelInfoW2100Net 121

CFilterCoefficientsNet 129

CFilterPropertyNet 135

CMcsUsbDacqNet::CHWInfo 179

CMcsUsbFunctionNet 298

CDacqGroupChannelSelectionTemplateNet< W2100DacqGroupChannelEnumNet, W2100↩ DacqGroupChannelEnum, CDeviceGroupChannelInfoW2100Net > 115

CW2100DacqGroupChannelSelectionNet 610

CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumNet, DacqGroup↩ ChannelEnum, CDeviceGroupChannelInfoNet > 115

CDacqGroupChannelSelectionNet 115

CDacqGroupChannelSelectionTemplateNet< int, int, CDeviceGroupChannelInfoGenericNet > 115

CDacqGroupChannelGenericSelectionNet 114

CDacqGroupChannelSelectionTemplateNet< SCUDacqGroupChannelEnumNet, SCUDacq↩ GroupChannelEnum, CDeviceGroupChannelInfoSCUNet > 115

CSCUDacqGroupChannelSelectionNet 490

CDacqGroupChannelSelectionTemplateNet< MEA2100_256DacqGroupChannelEnumNet, MEA2100_256DacqGroupChannelEnum, CDeviceGroupChannelInfoMEA2100_256Net > 115

CMEA2100_256DacqGroupChannelSelectionNet 329

CCMOSMea_FunctionNet 96

CDacCalibrationFunctionNet 112

CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, Dacq↩ GroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet > 115

CDigOutStimulatorFunctionNet 122

CFilterConfigurationNet 131

CFilterConfigurationRegisterNet 133

CGrapheneFunctionNet 165

CIntanMea_FunctionNet 182

CInterfaceboardFunctionNet 186

CInterfaceboard2FunctionNet 184
CMEA2100x256FunctionNet 330
CMcsBusNet 236
CMcsBus_AxisParametersNet 197
CMcsBus_ExtensionNet 199
CMcsBus_FYIExtensionNet 200
CMcsBus_MotorControlNet 202
CMcsBus_SensorNet 219
CMcsBus_TempSensorNet 229
CMcsBus_VoltageModeNet 231
CMcsUsbDeviceStatePushFunctionNet 287
CMultiwellCallbackFunctionNet 372
CSCUFunctionNet 491
CMeFunctionNet 363
CMeaAudioFunctionNet 331
CMeaDigitalDataFunctionNet 352
CMeaFeedbackFunctionNet 354
CMultiwellOptoStimFunctionNet 380
CPPCFunctionNet 413
CPPS_FunctionNet 423
CProgramPressureCurveNet 428
CPulseGeneratorFunctionNet 429
CRFFunctionNet 437
CRobo_FYIProgram_FunctionNet 442
CRobo_FYITemp_FunctionNet 443
CStimulusFunctionNet 555
CTEERFunctionNet 581
CUsbDeviceConfigurationFunctionNet 593
CW2100_StimulatorFunctionNet 604
CWarnerUssingFunctionNet 612
CWarnerValveControllerDeviceTesterFunctionNet 653
CWirelessBaseFunctionNet 661

CW2100_FunctionNet 596
CWClassicFunctionNet 656
CMcsUsbFunctionPointerContainer 299
CMcsUsbListEntryNet 299
CMcsUsbListNet 305
CMcsUsbNet 308
CExternDTesterDeviceNet 127
CFluidControlDeviceNet 136
CGenericDevelopDeviceNet 145
CGilsonDeviceNet 163
CMcsUsbDacqNet 240
CMeaDeviceNet 344
CMeaUSBDeviceNet 362
CCMOSMeaDeviceNet 107
CHLADacqNet 177
CLIH3DeviceNet 188
CMultiwellDeviceNet 374
CWarnerUssingDeviceNet 611
COctoPotDeviceNet 386
CRoboDacqNet 445
CMcsUsbDeviceStatePushNet 288
CWarnerValveControllerDeviceNet 628
CMcsUsbFactoryNet 289
CMeaCleanDeviceNet 334
CMeaCoatDeviceNet 338
CMeaImpedanceDeviceNet 358
CMeaSwitchDeviceNet 360
CChannelTestDeviceNet 95
CMultiBatteryChargerDeviceNet 365
CNF_GenDeviceNet 385
COkuvisionStimulatorDeviceNet 390
CPPCDeviceNet 412

CPPS_DeviceNet 422
CPathIdentDeviceNet 394
CPedoterDeviceNet 395
CPeristalticPumpDeviceNet 397
CPgaDeviceNet 398
CPositionIIDeviceNet 400
CPositionImpDeviceNet 409
CRadioControledDevicesNet 432
CRetinaLedDeviceNet 435
CRoboDeviceNet 460
CEncapsulatorDeviceNet 126
CFYIDeviceNet 143
CHLADeviceNet 178
CHiClampDeviceNet 176
CMeasureTableDeviceNet 359
CPPSDeviceNet 427
CPatchServerDeviceNet 393
CRoboStatorDeviceNet 481
CRoboInjectDeviceNet 479
CRoboocyte2DeviceNet 480
CTEERMachineDeviceNet 592
CRoboFluidDeviceNet 476
CSafeISDeviceNet 487
CSerialPortNet 506
CStg200xBasicNet 507
CStg200xDownloadBasicNet 541
CStg200xDownloadNet 548
CSw2to64DeviceNet 566
CTcxDeviceNet 568
CMcsUsbPointerContainer 329
CCMOSMeaDeviceNet::CRegionOfInterestRect 434
CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet 595

DeviceIdNet 662

DigitalSource< digitalsourceenum > 664

DigitalSourceGeneral 665

DriverVersionNet 666
Exception

CUsbExceptionNet 594

FirmwareDestinationNames 672

HeadstageIDTypeObject 678

HeadStageIDTypeState 680
IComparable

HeadStageIDType 675

mkfilterNet 681

CRoboDeviceNet::RoboMainLowLevelCommands 684

CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands 691

CFilterCoefficientsNet::s_FilterAttributesNet 691

CMeaAudioFunctionNet::s_setaudionet 692

CStimulusFunctionNet::SidebandData 693

StgStatusNet 694
stgstreaming

CStg200xBasicNet 507

CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData 694

usbSetupPacket_t 696

W2100_StimulusParametersNet 696

9 Class Index

9.1 Class List

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

CW2100_FunctionNet::AudioChannelsNet 92

BatteryState 92

BesselFilterHighPassNet 93

BesselFilterLowPassNet 93

ButterworthFilterHighPassNet 94

ButterworthFilterLowPassNet 95

CChannelTestDeviceNet 95

CCMOSMea_FunctionNet 96

CCMOSMeaDeviceNet 107

CCreateFilterNet 110

CDacCalibrationFunctionNet
112

CDacqGroupChannelGenericSelectionNet 114

CDacqGroupChannelSelectionNet 115

CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate,
115

CDeviceGroupChannelInfoGenericNet 118

CDeviceGroupChannelInfoMEA2100_256Net 119

CDeviceGroupChannelInfoNet 119

CDeviceGroupChannelInfoSCUNet 120

CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumTemplateNet > 120

CDeviceGroupChannelInfoW2100Net 121

CDigOutStimulatorFunctionNet
CDigOutStimulatorFunctionNet is the class of the DigOut stimulator function class 122

CEncapsulatorDeviceNet
CEncapsulatorDeviceNet is the to control the MCS HiClamp device 126

CExternDTesterDeviceNet
CExternDTesterDeviceNet is the class to access the ExternD Tester (Handheld Device Tester D) 127

CFilterCoefficientsNet 129

CFilterConfigurationNet 131

CFilterConfigurationRegisterNet 133

CFilterPropertyNet 135

CFluidControlDeviceNet
CFluidControlDeviceNet is the class to control MCS FluidControl (FCB and FCX) device 136

CFYIDeviceNet
CFYIDeviceNet is the class to control the MCS FYI device 143

CGenericDevelopDeviceNet
CGenericDevelopDeviceNet is the class to use during development of a new device 145

CGilsonDeviceNet
CGilsonDeviceNet is the class to control a Gilson device 163

CGrapheneFunctionNet
CGrapheneFunctionNet is the class to control Graphene device functions 165

CHiClampDeviceNet
CHiClampDeviceNet is the to control the MCS HiClamp device 176

CHLADacqNet 177

CHLADeviceNet
CHLADeviceNet is the to control the MCS HLA device 178

CMcsUsbDacqNet::CHWInfo
Class to provide hardware information about the device 179

CIntanMea_FunctionNet 182

CInterfaceboard2FunctionNet
CInterfaceboard2FunctionNet is the class to control the Interfaceboard 184

CInterfaceboardFunctionNet
CInterfaceboardFunctionNet is the class to control the Interfaceboard 186

CLIH3DeviceNet
CLIH3DeviceNet is the class to access the HEKA LIH3 device 188

CMcsBus_AxisParametersNet 197

CMcsBus_ExtensionNet 199

CMcsBus_FYIExtensionNet 200

CMcsBus_MotorControlNet 202

CMcsBus_SensorNet 219

CMcsBus_TempSensorNet 229

CMcsBus_VoltageModeNet 231

CMcsBusNet 236

CMcsUsbDacqNet
Base class for data acquisition devices 240

CMcsUsbDeviceStatePushFunctionNet 287

CMcsUsbDeviceStatePushNet 288

CMcsUsbFactoryNet 289

CMcsUsbFunctionNet 298

CMcsUsbFunctionPointerContainer 299

CMcsUsbListEntryNet
McsUsbListEntryNet identifies a connected device 299

CMcsUsbListNet
Class to handle a list of connected MCS USB devices 305

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 308

CMcsUsbPointerContainer 329

CMEA2100_256DacqGroupChannelSelectionNet 329

CMEA2100x256FunctionNet
CMEA2100x256FunctionNet is the class to control the MEA2100-256 device needs #include "↩ Stg200xNet.h" to resolve documentation reference 330

CMeaAudioFunctionNet 331

CMeaCleanDeviceNet
CMeaCleanDeviceNet is the class to access the MEA Clean device 334

CMeaCoatDeviceNet
CMeaCoatDeviceNet is the class to access the MEA Coat device 338

CMeaDeviceNet
Base class for MEA data acquisition devices 344

CMeaDigitalDataFunctionNet 352

CMeaFeedbackFunctionNet 354

CMeaImpedanceDeviceNet 358

CMeasureTableDeviceNet
CMeasureTableDeviceNet is the to control the MCS HLA device 359

CMeaSwitchDeviceNet
The class to control the USB-MEA-Switch 360

CMeaUSBDeviceNet
Class for data acquisition via ME and MEA USB amplifiers 362

CMeFunctionNet
363

CMultiBatteryChargerDeviceNet
CMultiBatteryChargerDeviceNet is the class to access the MBC-08 device 365

CMultiwellCallbackFunctionNet
CMultiwellCallbackFunctionNet is the class to access the Multiwell-Mini-Stimulator 372

CMultiwellDeviceNet
CMultiwellDeviceNet is the class to access the Multiwell device 374

CMultiwellOptoStimFunctionNet
CMultiwellOptoStimFunctionNet is the class to access the optical properties of the Multiwell Optostim device 380

CNF_GenDeviceNet 385

COctoPotDeviceNet 386

COkuvisionStimulatorDeviceNet 390

CPatchServerDeviceNet
CPatchServerDeviceNet is the class to control the MCS PatchServer device 393

CPathIdentDeviceNet 394

CPedoterDeviceNet
395

CPeristalticPumpDeviceNet
CPeristalticPumpDeviceNet is the class to control a Persistaltic Pump 397

CPgaDeviceNet 398

CPositionIIDeviceNet
CPositionIIDeviceNet is the class to control PositionII devices 400

CPositionImpDeviceNet
CPositionImpDeviceNet is the class to access the Position/Imp devices 409

CPPCDeviceNet 412

CPPCFunctionNet
CPPCFunctionNet is the class to access the PPC (high precision Patch Peristalic patch Pump 413

CPPS_DeviceNet 422

CPPS_FunctionNet 423

CPPSDeviceNet
CPPS4plus1DeviceNet is the to control the MCS HLA device 427

CProgramPressureCurveNet
CProgramPressureCurveNet is the class to program pressure curves 428

CPulseGeneratorFunctionNet
CPulseGeneratorFunctionNet is the class to control the pulse generator for video tracking 429

CRadioControledDevicesNet 432

CCMOSMeaDeviceNet::CRegionOfInterestRect 434

CRetinaLedDeviceNet 435

CRFFunctionNet
CRFFunctionNet is the class to control RF devices 437

CRobo_FYIProgram_FunctionNet 442

CRobo_FYITemp_FunctionNet 443

CRoboDacqNet 445

CRoboDeviceNet
CRoboDeviceNet is the base class for all Robo platform based devices 460

CRoboFluidDeviceNet 476

CRoboInjectDeviceNet
CRoboInjectDeviceNet is the to control the MCS RoboInject device 479

CRoboocyte2DeviceNet
CRoboocyte2DeviceNet is the class to control the MCS Roboocyte2 device 480

CRoboStatorDeviceNet 481

CSafeISDeviceNet
487

CSCUDacqGroupChannelSelectionNet 490

CSCUFunctionNet
CSCUFunctionNet is the class to control the SCU device 491

CSerialPortNet 506

CStg200xBasicNet
Base class for the Stg200x 507

CStg200xDownloadBasicNet
CStg200xDownloadBasicNet is the base class to control the download mode of the MCS STG device 541

CStg200xDownloadNet
Main class for the STG download mode This class implements the STG download mode interface. 548

CStimulusFunctionNet 555

CSw2to64DeviceNet
The class to control the MCS-USB-Sw2to64 device 566

CTcxDeviceNet
Class to control a Temperature Controller (TCX) 568

CTEERFunctionNet
CTEERFunctionNet is the class to control the TEER device 581

CTEERMachineDeviceNet 592

CUsbDeviceConfigurationFunctionNet
CUsbDeviceConfigurationFunctionNet is the class to configure the USB firmware 593

CUsbExceptionNet
Exception class that is thrown in case of an USB error 594

CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet 595

CW2100_FunctionNet 596

CW2100_StimulatorFunctionNet 604

CW2100DacqGroupChannelSelectionNet 610

CWarnerUssingDeviceNet
CWarnerUssingDeviceNet is the class to control the Ussing device 611

CWarnerUssingFunctionNet
CWarnerUssingFunctionNet is the class to control the Ussing device 612

CWarnerValveControllerDeviceNet
CWarnerValveControllerDeviceNet is the class to access the Warner Valve Controller 628

CWarnerValveControllerDeviceTesterFunctionNet
CWarnerValveControllerDeviceTesterFunctionNet is the class to access the functions for the Warner Valve Controller Device Tester 653

CWClassicFunctionNet 656

CWirelessBaseFunctionNet 661

DeviceIdNet
Device Id 662

DigitalSource< digitalsourceenum > 664

DigitalSourceGeneral 665

DriverVersionNet
Class gives firmware versions of the device's firmware destinations 666

FirmwareDestinationNames 672

HeadStageIDType 675

HeadstageIDTypeObject 678

HeadStageIDTypeState 680

mkfilterNet 681

CRoboDeviceNet::RoboMainLowLevelCommands 684

CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands 691

CFilterCoefficientsNet::s_FilterAttributesNet 691

CMeaAudioFunctionNet::s_setaudionet 692

CStimulusFunctionNet::SidebandData 693

StgStatusNet 694

CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData 694

usbSetupPacket_t 696

W2100_StimulusParametersNet 696

10 Namespace Documentation

10.1 Mcs Namespace Reference

Namespaces

- Usb

10.2 Mcs::Usb Namespace Reference

Classes

- class CChannelTestDeviceNet
- class CCMOSMea_FunctionNet
- class CCMOSMeaDeviceNet
- class CCreateFilterNet
- class ButterworthFilterLowPassNet
- class ButterworthFilterHighPassNet
- class BesselFilterLowPassNet
- class BesselFilterHighPassNet

- class CDeviceGroupChannelInfoTemplateNet
- class CDeviceGroupChannelInfoGenericNet
- class CDeviceGroupChannelInfoNet
- class CDeviceGroupChannelInfoW2100Net
- class CDeviceGroupChannelInfoSCUNet
- class CDeviceGroupChannelInfoMEA2100_256Net
- class CDacqGroupChannelSelectionTemplateNet
- class CDacqGroupChannelGenericSelectionNet
- class CDacqGroupChannelSelectionNet
- class CW2100DacqGroupChannelSelectionNet
- class CSCUDacqGroupChannelSelectionNet
- class CMEA2100_256DacqGroupChannelSelectionNet
- class CDacCalibrationFunctionNet
- class CDigOutStimulatorFunctionNet

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

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

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

  CInterfaceboard2FunctionNet is the class to control the Interfaceboard
- class CInterfaceboardFunctionNet

  CInterfaceboardFunctionNet is the class to control the Interfaceboard
- class CLIH3DeviceNet

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

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

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

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

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

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

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

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

  CPositionIIDeviceNet is the class to control PositionII devices
- class CPositionImpDeviceNet

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

  CPPCFunctionNet is the class to access the PPC (high precision Patch 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 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 = 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 ,
  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_FlagShip_Core_2 = MCS_PRODUCT_ID_GRAPHENE_FLAGSHIP_CORE_2 ,
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 ,
BUSNUMBER1 = MCSUSB_DEST_BUSNUMBER1 ,
BUS1MCSBUS1 = MCSUSB_DEST_BUS1_MCSBUS1 ,
BUS1MCSBUS2 = MCSUSB_DEST_BUS1_MCSBUS2 ,
BUSNUMBER2 = MCSUSB_DEST_BUSNUMBER2 ,
BUS2MCSBUS1 = MCSUSB_DEST_BUS2_MCSBUS1 ,
BUS2MCSBUS2 = MCSUSB_DEST_BUS2_MCSBUS2 ,
PIC = MCSUSB_DEST_PIC ,
PIC2 = MCSUSB_DEST_PIC2 ,
PIC3 = MCSUSB_DEST_PIC3 ,
PIC4 = MCSUSB_DEST_PIC4 ,
PIC5 = MCSUSB_DEST_PIC5 ,
PIC6 = MCSUSB_DEST_PIC6 ,
PIC7 = MCSUSB_DEST_PIC7 ,
PIC8 = MCSUSB_DEST_PIC8 ,
PIC9 = MCSUSB_DEST_PIC9 ,

PIC10 = MCSUSB_DEST_PIC10 ,
PIC11 = MCSUSB_DEST_PIC11 ,
PIC12 = MCSUSB_DEST_PIC12 ,
ChannelPIC = MCSUSB_DEST_CHANNELPIC ,
Bootstrap = MCSUSB_DEST_BOOTSTRAP ,
BootstrapOtherCypress = MCSUSB_DEST_BOOTSTAP_OTHER_CYPRESS ,
ALTERA = MCSUSB_DEST_ALTERA ,
FPGA2 = MCSUSB_DEST_FPGA2 ,
FPGA3 = MCSUSB_DEST_FPGA3 ,
FPGA4 = MCSUSB_DEST_FPGA4 ,
FPGA5 = MCSUSB_DEST_FPGA5 ,
FPGA6 = MCSUSB_DEST_FPGA6 ,
FPGA7 = MCSUSB_DEST_FPGA7 ,
FPGA8 = MCSUSB_DEST_FPGA8 ,
FPGA9 = MCSUSB_DEST_FPGA9 ,
FPGA10 = MCSUSB_DEST_FPGA10 ,
FPGA11 = MCSUSB_DEST_FPGA11 ,
FPGA12 = MCSUSB_DEST_FPGA12 ,
FPGA13 = MCSUSB_DEST_FPGA13 ,
FPGA14 = MCSUSB_DEST_FPGA14 ,
FPGA15 = MCSUSB_DEST_FPGA15 ,
FPGA16 = MCSUSB_DEST_FPGA16 ,
FPGA_GOLD = XILINX_DEST_GOLDEN ,
ALTERA_GOLD = (MCSUSB_DEST_ALTERA | XILINX_DEST_GOLDEN) ,
FPGA2_GOLD = (MCSUSB_DEST_FPGA2 | XILINX_DEST_GOLDEN) ,
FPGA3_GOLD = (MCSUSB_DEST_FPGA3 | XILINX_DEST_GOLDEN) ,
FPGA4_GOLD = (MCSUSB_DEST_FPGA4 | XILINX_DEST_GOLDEN) ,
FPGA5_GOLD = (MCSUSB_DEST_FPGA5 | XILINX_DEST_GOLDEN) ,
FPGA6_GOLD = (MCSUSB_DEST_FPGA6 | XILINX_DEST_GOLDEN) ,
FPGA7_GOLD = (MCSUSB_DEST_FPGA7 | XILINX_DEST_GOLDEN) ,
FPGA8_GOLD = (MCSUSB_DEST_FPGA8 | XILINX_DEST_GOLDEN) ,
FPGA9_GOLD = (MCSUSB_DEST_FPGA9 | XILINX_DEST_GOLDEN) ,
FPGA10_GOLD = (MCSUSB_DEST_FPGA10 | XILINX_DEST_GOLDEN) ,
FPGA11_GOLD = (MCSUSB_DEST_FPGA11 | XILINX_DEST_GOLDEN) ,
FPGA12_GOLD = (MCSUSB_DEST_FPGA12 | XILINX_DEST_GOLDEN) ,
FPGA13_GOLD = (MCSUSB_DEST_FPGA13 | XILINX_DEST_GOLDEN) ,
FPGA14_GOLD = (MCSUSB_DEST_FPGA14 | XILINX_DEST_GOLDEN) ,
FPGA15_GOLD = (MCSUSB_DEST_FPGA15 | XILINX_DEST_GOLDEN) ,
FPGA16_GOLD = (MCSUSB_DEST_FPGA16 | XILINX_DEST_GOLDEN) ,
FPGA_BASE = XILINX_DEST_BASEIMAGE ,
ALTERA_BASE = (MCSUSB_DEST_ALTERA | XILINX_DEST_BASEIMAGE) ,
FPGA2_BASE = (MCSUSB_DEST_FPGA2 | XILINX_DEST_BASEIMAGE) ,
FPGA3_BASE = (MCSUSB_DEST_FPGA3 | XILINX_DEST_BASEIMAGE) ,
FPGA4_BASE = (MCSUSB_DEST_FPGA4 | XILINX_DEST_BASEIMAGE) ,
FPGA5_BASE = (MCSUSB_DEST_FPGA5 | XILINX_DEST_BASEIMAGE) ,
FPGA6_BASE = (MCSUSB_DEST_FPGA6 | XILINX_DEST_BASEIMAGE) ,
FPGA7_BASE = (MCSUSB_DEST_FPGA7 | XILINX_DEST_BASEIMAGE) ,
FPGA8_BASE = (MCSUSB_DEST_FPGA8 | XILINX_DEST_BASEIMAGE) ,
FPGA9_BASE = (MCSUSB_DEST_FPGA9 | XILINX_DEST_BASEIMAGE) ,
FPGA10_BASE = (MCSUSB_DEST_FPGA10 | XILINX_DEST_BASEIMAGE) ,
FPGA11_BASE = (MCSUSB_DEST_FPGA11 | XILINX_DEST_BASEIMAGE) ,
FPGA12_BASE = (MCSUSB_DEST_FPGA12 | XILINX_DEST_BASEIMAGE) ,
FPGA13_BASE = (MCSUSB_DEST_FPGA13 | XILINX_DEST_BASEIMAGE) ,
FPGA14_BASE = (MCSUSB_DEST_FPGA14 | XILINX_DEST_BASEIMAGE) ,
FPGA15_BASE = (MCSUSB_DEST_FPGA15 | XILINX_DEST_BASEIMAGE) ,
FPGA16_BASE = (MCSUSB_DEST_FPGA16 | XILINX_DEST_BASEIMAGE) ,
FPGA_BOOTSTRAP = XILINX_DEST_BOOTSTRAP ,
ALTERA_BOOTSTRAP = (MCSUSB_DEST_ALTERA | XILINX_DEST_BOOTSTRAP) ,

DEST_TARGET1 = FLASH_DEST_TARGET1 ,
DEST_TARGET2 = FLASH_DEST_TARGET2 ,
DEST_TARGET3 = FLASH_DEST_TARGET3 ,
DEST_TARGET4 = FLASH_DEST_TARGET4 ,
DEST_TARGET5 = FLASH_DEST_TARGET5 ,
DEST_TARGET6 = FLASH_DEST_TARGET6 ,
DEST_TARGET7 = FLASH_DEST_TARGET7 ,
DEST_TARGET8 = FLASH_DEST_TARGET8 ,
DEST_TARGET9 = FLASH_DEST_TARGET9 ,
DEST_TARGET10 = FLASH_DEST_TARGET10 ,
DEST_TARGET11 = FLASH_DEST_TARGET11 ,
DEST_TARGET12 = FLASH_DEST_TARGET12 ,
DEST_TARGET13 = FLASH_DEST_TARGET13 ,
DEST_TARGET14 = FLASH_DEST_TARGET14 ,
DEST_TARGET15 = FLASH_DEST_TARGET15 ,
DEST_TARGET_MASK = FPGA_DEST_TARGET_MASK ,
DEST_FX3_TARGET_MASK = FX3_DEST_TARGET_MASK ,
ALTERA_TARGET1 = (MCSUSB_DEST_ALTERA | FLASH_DEST_TARGET1) ,
ALTERA_TARGET2 = (MCSUSB_DEST_ALTERA | FLASH_DEST_TARGET2) ,
ALTERA_TARGET3 = (MCSUSB_DEST_ALTERA | FLASH_DEST_TARGET3) ,
USB_TARGET1 = (MCSUSB_DEST_USB | FLASH_DEST_TARGET1) ,
USB_TARGET2 = (MCSUSB_DEST_USB | FLASH_DEST_TARGET2) ,
USB_TARGET3 = (MCSUSB_DEST_USB | FLASH_DEST_TARGET3) ,
UnknownDest = MCSUSB_DEST_UNKNOWN }

Enumerates the destination processor for the firmware.

- enum class DigitalTargetEnumNet {
  Digout = (MEA_COMMAND << 16) + MEA_MEA21_DIGOUT_SOURCE ,
  Digstream = (MEA_COMMAND << 16) + MEA_MEA21_DIGSTREAM_SOURCE ,
  DacqTrigger = (MEA_COMMAND << 16) + MEA_MEA21_DACQTRIGGER_SOURCE ,
  StgTrigger = (STG200x_COMMAND << 16) + STG200x_TRIGGER_SOURCE ,
  StgListModeTrigger = (STG200x_COMMAND << 16) + STG200x_MEA21_LISTMODE_TRIGGERSOURCE ,
  DigOutStimulatorStartTrigger = (MEA_COMMAND << 16) + MEA_DIGOUT_STG_START_TRIGGER_↩ SOURCE ,
  DigOutStimulatorStopTrigger = (MEA_COMMAND << 16) + MEA_DIGOUT_STG_STOP_TRIGGER_↩ SOURCE ,
  DigStreamToReceiver = (MEA_COMMAND << 16) + MEA_DIGSTREAMTORECEIVER_SOURCE }

  Enumerates the Digital Targets for Digital Sources

- enum class DigitalSourceEnumNet {
  DigitalInOfOutPort = 0 ,
  DigitalIn = 16 ,
  DigitalPulse = 32 ,
  Feedback = 64 ,
  AuxIn = 96 ,
  Zero = 98 ,
  One = 99 ,
  HS1Trigger1Status = 100 ,
  HS1Trigger2Status = 102 ,
  HS1Trigger3Status = 104 ,
  HS1Trigger4Status = 106 ,
  HS1Trigger5Status = 108 ,
  HS1Trigger6Status = 110 ,
  HS1Sideband1 = 112 ,
  HS1Sideband2 = 128 ,
  HS1Sideband3 = 144 ,
  HS1Sideband4 = 160 ,
  HS1Sideband5 = 176 ,
  HS1Sideband6 = 192 ,

HS2Trigger1Status = 208 ,
HS2Trigger2Status = 210 ,
HS2Trigger3Status = 212 ,
HS2Trigger4Status = 214 ,
HS2Trigger5Status = 216 ,
HS2Trigger6Status = 218 ,
HS2Sideband1 = 220 ,
HS2Sideband2 = 236 ,
HS2Sideband3 = 252 ,
HS2Sideband4 = 268 ,
HS2Sideband5 = 284 ,
HS2Sideband6 = 300 ,
PulseGenerator = 316 ,
DigitalOutStimulator = 320 ,
DigitalData = 336 ,
DeviceRunStatus = 368 ,
LastPosition = 372 }

Enumerates the digital source of the MEA2100 device.

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

Enumerates the digital source of the W2100 device.

- enum class SCUDigitalSourceEnumNet {
DigitalInOfOutPort = (0x00 << 8) + 0 ,
DigitalIn = (0x00 << 8) + 16 ,
DigitalPulse = (0x01 << 8) ,
Feedback = (0x02 << 8) ,
AuxIn = (0x03 << 8) + 0 ,
Zero = (0x03 << 8) + 2 ,
One = (0x03 << 8) + 3 ,
PulseGenerator = (0x03 << 8) + 8 ,
DigitalOutStimulator = (0x03 << 8) + 16 ,
DigitalData = (0x04 << 8) ,
DeviceRunStatus = (0x05 << 8) + 0 ,
HS1Trigger1Status = (0x40 << 8) + 0 ,
HS1Trigger2Status = (0x40 << 8) + 2 ,
HS1Trigger3Status = (0x40 << 8) + 4 ,
HS1Trigger4Status = (0x40 << 8) + 6 ,
HS1Trigger5Status = (0x40 << 8) + 8 ,
HS1Trigger6Status = (0x40 << 8) + 10 ,
HS1Trigger7Status = (0x40 << 8) + 12 ,
HS1Trigger8Status = (0x40 << 8) + 14 ,
HS1Trigger9Status = (0x40 << 8) + 16 ,
HS1Trigger10Status = (0x40 << 8) + 18 ,
HS1Trigger11Status = (0x40 << 8) + 20 ,
HS1Trigger12Status = (0x40 << 8) + 22 ,

HS1Sideband1 = (0x42 << 8) ,
HS1Sideband2 = (0x43 << 8) ,
HS1Sideband3 = (0x44 << 8) ,
HS1Sideband4 = (0x45 << 8) ,
HS1Sideband5 = (0x46 << 8) ,
HS1Sideband6 = (0x47 << 8) ,
HS1Sideband7 = (0x48 << 8) ,
HS1Sideband8 = (0x49 << 8) ,
HS1Sideband9 = (0x4A << 8) ,
HS1Sideband10 = (0x4B << 8) ,
HS1Sideband11 = (0x4C << 8) ,
HS1Sideband12 = (0x4D << 8) ,
HS2Trigger1Status = (0x80 << 8) + 0 ,
HS2Trigger2Status = (0x80 << 8) + 2 ,
HS2Trigger3Status = (0x80 << 8) + 4 ,
HS2Trigger4Status = (0x80 << 8) + 6 ,
HS2Trigger5Status = (0x80 << 8) + 8 ,
HS2Trigger6Status = (0x80 << 8) + 10 ,
HS2Trigger7Status = (0x80 << 8) + 12 ,
HS2Trigger8Status = (0x80 << 8) + 14 ,
HS2Trigger9Status = (0x80 << 8) + 16 ,
HS2Trigger10Status = (0x80 << 8) + 18 ,
HS2Trigger11Status = (0x80 << 8) + 20 ,
HS2Trigger12Status = (0x80 << 8) + 22 ,
HS2Sideband1 = (0x82 << 8) ,
HS2Sideband2 = (0x83 << 8) ,
HS2Sideband3 = (0x84 << 8) ,
HS2Sideband4 = (0x85 << 8) ,
HS2Sideband5 = (0x86 << 8) ,
HS2Sideband6 = (0x87 << 8) ,
HS2Sideband7 = (0x88 << 8) ,
HS2Sideband8 = (0x89 << 8) ,
HS2Sideband9 = (0x8A << 8) ,
HS2Sideband10 = (0x8B << 8) ,
HS2Sideband11 = (0x8C << 8) ,
HS2Sideband12 = (0x8D << 8) ,
LastPosition = (0xFF << 8) }

Enumerates the digital source of the SCU device.

- enum class MEA2100_256DigitalSourceEnumNet {
DigitalInOfOutPort = (0x00 << 8) + 0 ,
DigitalIn = (0x00 << 8) + 16 ,
DigitalPulse = (0x01 << 8) ,
Feedback = (0x02 << 8) ,
AuxIn = (0x03 << 8) + 0 ,
Zero = (0x03 << 8) + 2 ,
One = (0x03 << 8) + 3 ,
DeviceRunStatus = (0x03 << 8) + 4 ,
PulseGenerator = (0x03 << 8) + 8 ,
DigitalOutStimulator = (0x03 << 8) + 16 ,
DigitalData = (0x04 << 8) ,
HS1Trigger1Status = (0x40 << 8) + 0 ,
HS1Trigger2Status = (0x40 << 8) + 2 ,
HS1Trigger3Status = (0x40 << 8) + 4 ,
HS1Trigger4Status = (0x40 << 8) + 6 ,
HS1Trigger5Status = (0x40 << 8) + 8 ,
HS1Trigger6Status = (0x40 << 8) + 10 ,
HS1Trigger7Status = (0x40 << 8) + 12 ,
HS1Trigger8Status = (0x40 << 8) + 14 ,

HS1Trigger9Status = (0x40 << 8) + 16 ,
HS1Trigger10Status = (0x40 << 8) + 18 ,
HS1Trigger11Status = (0x40 << 8) + 20 ,
HS1Trigger12Status = (0x40 << 8) + 22 ,
HS1Trigger13Status = (0x40 << 8) + 24 ,
HS1Trigger14Status = (0x40 << 8) + 26 ,
HS1Trigger15Status = (0x40 << 8) + 28 ,
HS1Trigger16Status = (0x40 << 8) + 30 ,
HS1Trigger17Status = (0x41 << 8) + 0 ,
HS1Trigger18Status = (0x41 << 8) + 2 ,
HS1Sideband1 = (0x42 << 8) ,
HS1Sideband2 = (0x43 << 8) ,
HS1Sideband3 = (0x44 << 8) ,
HS1Sideband4 = (0x45 << 8) ,
HS1Sideband5 = (0x46 << 8) ,
HS1Sideband6 = (0x47 << 8) ,
HS1Sideband7 = (0x48 << 8) ,
HS1Sideband8 = (0x49 << 8) ,
HS1Sideband9 = (0x4A << 8) ,
HS1Sideband10 = (0x4B << 8) ,
HS1Sideband11 = (0x4C << 8) ,
HS1Sideband12 = (0x4D << 8) ,
HS1Sideband13 = (0x4E << 8) ,
HS1Sideband14 = (0x4F << 8) ,
HS1Sideband15 = (0x50 << 8) ,
HS1Sideband16 = (0x51 << 8) ,
HS1Sideband17 = (0x52 << 8) ,
HS1Sideband18 = (0x53 << 8) ,
HS2Trigger1Status = (0x80 << 8) + 0 ,
HS2Trigger2Status = (0x80 << 8) + 2 ,
HS2Trigger3Status = (0x80 << 8) + 4 ,
HS2Trigger4Status = (0x80 << 8) + 6 ,
HS2Trigger5Status = (0x80 << 8) + 8 ,
HS2Trigger6Status = (0x80 << 8) + 10 ,
HS2Trigger7Status = (0x80 << 8) + 12 ,
HS2Trigger8Status = (0x80 << 8) + 14 ,
HS2Trigger9Status = (0x80 << 8) + 16 ,
HS2Trigger10Status = (0x80 << 8) + 18 ,
HS2Trigger11Status = (0x80 << 8) + 20 ,
HS2Trigger12Status = (0x80 << 8) + 22 ,
HS2Trigger13Status = (0x80 << 8) + 24 ,
HS2Trigger14Status = (0x80 << 8) + 26 ,
HS2Trigger15Status = (0x80 << 8) + 28 ,
HS2Trigger16Status = (0x80 << 8) + 30 ,
HS2Trigger17Status = (0x81 << 8) + 0 ,
HS2Trigger18Status = (0x81 << 8) + 2 ,
HS2Sideband1 = (0x82 << 8) ,
HS2Sideband2 = (0x83 << 8) ,
HS2Sideband3 = (0x84 << 8) ,
HS2Sideband4 = (0x85 << 8) ,
HS2Sideband5 = (0x86 << 8) ,
HS2Sideband6 = (0x87 << 8) ,
HS2Sideband7 = (0x88 << 8) ,
HS2Sideband8 = (0x89 << 8) ,
HS2Sideband9 = (0x8A << 8) ,
HS2Sideband10 = (0x8B << 8) ,
HS2Sideband11 = (0x8C << 8) ,
HS2Sideband12 = (0x8D << 8) ,

HS2Sideband13 = (0x8E << 8) ,
HS2Sideband14 = (0x8F << 8) ,
HS2Sideband15 = (0x90 << 8) ,
HS2Sideband16 = (0x91 << 8) ,
HS2Sideband17 = (0x92 << 8) ,
HS2Sideband18 = (0x93 << 8) ,
LastPosition = (0xFF << 8) }

Enumerates the digital source of the MEA2100-256 device.

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

Enumerates the digital source of the TBSI-DACQ device.

- enum class TriggerSourceEnumNet {
tsNone = 0 ,
tsDigitalIn1 = 1 ,
tsDigitalIn2 = 2 ,
tsDigitalIn3 = 3 ,
tsDigitalIn4 = 4 ,
tsDigitalIn5 = 5 ,
tsDigitalIn6 = 6 ,
tsDigitalIn7 = 7 ,
tsDigitalIn8 = 8 ,
tsDigitalIn9 = 9 ,
tsDigitalIn10 = 10 ,
tsDigitalIn11 = 11 ,
tsDigitalIn12 = 12 ,
tsDigitalIn13 = 13 ,
tsDigitalIn14 = 14 ,
tsDigitalIn15 = 15 ,
tsDigitalIn16 = 16 ,
tsDigitalIn17 = 17 ,
tsDigitalIn18 = 18 ,
tsDigitalIn19 = 19 ,
tsDigitalIn20 = 20 ,
tsDigitalIn21 = 21 ,
tsDigitalIn22 = 22 ,
tsDigitalIn23 = 23 ,
tsDigitalIn24 = 24 ,
tsDigitalIn25 = 25 ,
tsDigitalIn26 = 26 ,
tsDigitalIn27 = 27 ,
tsDigitalIn28 = 28 ,
tsDigitalIn29 = 29 ,
tsDigitalIn30 = 30 ,
tsDigitalIn31 = 31 ,
tsDigitalIn32 = 32 ,

tsFeedback1 = 33 ,
tsFeedback2 = 34 ,
tsFeedback3 = 35 ,
tsFeedback4 = 36 ,
tsFeedback5 = 37 ,
tsFeedback6 = 38 ,
tsFeedback7 = 39 ,
tsFeedback8 = 40 ,
tsFeedback9 = 41 ,
tsFeedback10 = 42 ,
tsFeedback11 = 43 ,
tsFeedback12 = 44 ,
tsFeedback13 = 45 ,
tsFeedback14 = 46 ,
tsFeedback15 = 47 ,
tsFeedback16 = 48 ,
tsFeedback17 = 49 ,
tsFeedback18 = 50 ,
tsFeedback19 = 51 ,
tsFeedback20 = 52 ,
tsFeedback21 = 53 ,
tsFeedback22 = 54 ,
tsFeedback23 = 55 ,
tsFeedback24 = 56 ,
tsFeedback25 = 57 ,
tsFeedback26 = 58 ,
tsFeedback27 = 59 ,
tsFeedback28 = 60 ,
tsFeedback29 = 61 ,
tsFeedback30 = 62 ,
tsFeedback31 = 63 ,
tsFeedback32 = 64 ,
tsAuxIn1 = 65 ,
tsAuxIn2 = 66 ,
tsDigitalPuse0 = 67 ,
tsDigitalPuse1 = 68 ,
tsDigitalPuse2 = 69 ,
tsDigitalPuse3 = 70 ,
tsDigitalPuse4 = 71 ,
tsDigitalPuse5 = 72 ,
tsDigitalPuse6 = 73 ,
tsDigitalPuse7 = 74 ,
tsDigitalPuse8 = 75 ,
tsDigitalPuse9 = 76 ,
tsDigitalPuse10 = 77 ,
tsDigitalPuse11 = 78 ,
tsDigitalPuse12 = 79 ,
tsDigitalPuse13 = 80 ,
tsDigitalPuse14 = 81 ,
tsDigitalPuse15 = 82 ,
tsDigitalPuse16 = 83 ,
tsDigitalPuse17 = 84 ,
tsDigitalPuse18 = 85 ,
tsDigitalPuse19 = 86 ,
tsDigitalPuse20 = 87 ,
tsDigitalPuse21 = 88 ,
tsDigitalPuse22 = 89 ,
tsDigitalPuse23 = 90 ,

tsDigitalPuse24 = 91 ,
tsDigitalPuse25 = 92 ,
tsDigitalPuse26 = 93 ,
tsDigitalPuse27 = 94 ,
tsDigitalPuse28 = 95 ,
tsDigitalPuse29 = 96 ,
tsDigitalPuse30 = 97 ,
tsDigitalPuse31 = 98 ,
tsTriggered = 99 ,
tsSidebandBit8 = 100 ,
tsDACQCy1Dev1Runs = 101 ,
tsDACQCy1Dev2Runs = 102 ,
tsDACQCy2Dev1Runs = 103 ,
tsDACQCy2Dev2Runs = 104 }

Enumerates the trigger source of the MEA2100 device.

- enum class AnalogSourceEnumNet {
AnalogSource_HS1 ,
AnalogSource_HS2 ,
AnalogSource_IF }

Enumerates the analog source of the MEA2100 device.

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

Enumerates the STG download mode trigger status

- enum class RetriggerActionEnumNet {
raStop = STG200x_RETRIGGER_STOP ,
raRestart = STG200x_RETRIGGER_RESTART ,
raIgnore = STG200x_RETRIGGER_IGNORE ,
raGate = STG200x_RETRIGGER_GATEMODE ,
raSingle = STG200x_RETRIGGER_SINGLE }

Enumerates possible retrigger actions for STG200x devices.

- enum class Stg200xSegmentFlagsEnumNet {
None = 0 ,
UpdateTrigger = SEGMENTFLAGS_UPDATETRIGGER ,
DownloadOnly = SEGMENTFLAGS_DOWNLOADONLY ,
TriggerOnly = SEGMENTFLAGS_TRIGGERONLY ,
SyncStart = SEGMENTFLAGS_SYNCSTART }

Enumerates Segmentflag options for STG400x devices.

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

Enumerates the DigoutMode on STG400x devices.

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

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

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

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

  Enumerates the adapter type of the MEA2100 device.
- enum class MeaLayoutEnumNet {
  mlUnknown = 0 ,
  mlMEA60 = 1 }

  Enumerates the MEA layout of the MEA2100 device.
- enum class DataModeEnumNet {
  Unsigned_16bit = 0 ,
  Unsigned_24bit = 2 ,
  Unsigned_32bit = 3 ,
  Signed_16bit = 8 ,
  Signed_24bit = 10 ,
  Signed_32bit = 11 }

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

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

  Enumerates the destination data format for ChannelBlock functions.
- enum class TcxDeviceTypeEnumNet {
  Unknown = 0 ,
  Regular = 1 ,

BMI = 2 ,
Nanion = 3 ,
Warner = 4 }

Enumerates the type of TCX devices.

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

Enumerates the sensor types for TCX devices

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

Enumerates the destination for STG downloads.

- enum class ElectrodeModeEnumNet {
emAutomatic = 0 ,
emManual = 3 }

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

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

Enumerates the setting of the Stimulation DAC Multiplexer.

- enum class DacqGroupChannelEnumNet {
HeadstageElectrodeGroup = 0x00 ,
InterfaceADCGroup = INTERFACEANALOGCHANNELSGROUP ,
DSPDataGroup = DSPDATACHANNELSGROUP ,
Headstage1NCBathCurrentGroup = 0x30 ,
Headstage1NCCol2CurrentGroup = 0x31 ,
Headstage1NChipTempGroup = 0x32 ,
STG1DACSignalGroup = 0x38 ,
LIH30UserADCGroup = 0x50 ,
LIH30TestADCGroup = 0x51 ,
LIH30ADCModulesGroup = 0x52 ,
IFDigChannelsGroup = INTERFACEDIGITALCHANNELSGROUP ,
STG1SidebandsGroup = 0x90 ,
STG1TriggerStatusGroup = 0x91 ,
DACQ1DigitalGroup = 0xA0 ,
AudioTestChannelGroup = AUDIOTESTCHANNELGROUP ,
PacketFrameContextGroup = PACKETFRAMECONTEXTGROUP }

Enumerates the Channel Groups of Datastream

- enum class W2100DacqGroupChannelEnumNet {
  InterfaceADCGroup = INTERFACEANALOGCHANNELSGROUP ,
  DSPDataGroup = DSPDATACHANNELSGROUP ,
  WirelessHeadStageAnalogRE1HS1 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 0 + 0 ,
  WirelessHeadStageStatusRE1HS1 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 0 + 1 ,
  WirelessHeadStageAnalogRE1HS2 = WIRELESSHEADSTAGEANALOGGROUPBASE + 0 + 2 + 0 ,
  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_100PBA = HS_PLATETYPE_60 ,
  Plate_Dummy_126 = HS_PLATETYPE_126 ,
  Plate_24W300_30GMA = HS_PLATETYPE_193 ,
  Plate_96W700_100GMA = 194 ,
  Plate_24W300_30GBA = HS_PLATETYPE_195 ,
  Plate_96W700_100GBA = HS_PLATETYPE_224 ,
  Plate_24W300_30GBB = HS_PLATETYPE_232 ,
  Plate_96W700_100GBB = HS_PLATETYPE_244 ,
  No_Plate = 255 }
- enum class 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 ,
GrapheneFlagshipCore2Headstage = FPGA_ID_GRAPHENE_FLAGSHIP_CORE_2_HS ,
WholeCellPatchHeadstage = FPGA_ID_WHOLE_CELL_PATCH_HS ,
InterfaceBoard2 = FPGA_ID_INTERFACEBOARD2 ,
W2100IFB2 = FPGA_ID_W2100_IFB2 ,
CmosmeaIFB2 = FPGA_ID_CMOS_MEA_IFB2 ,
Mea2100LiteHeadstage = FPGA_ID_MEA2100_LITE_HS ,
LIH30Interfaceboard = FPGA_ID_LIH30_USB_IF ,
LIH30ADCCtrl = FPGA_ID_LIH30_ADC_CTRL ,
UssingRail = FPGA_ID_USSING_RAIL ,
UssingChamber = FPGA_ID_USSING_CHAMBER ,
IFB2GoldenInterfaceboard = FPGA_ID_IFB2_GOLDEN ,
IFB30GoldenInterfaceboard = FPGA_ID_IFB30_GOLDEN ,
DeviceHasNoHeadstage = FPGA_ID_HAS_NO_HS }

- enum class HeadstageIdEnumNet {
  DeviceNotConnected = FPGA_ID_NOT_CONNECTED ,
  Mea2100 = FPGA_ID_MEA2100_HS ,
  Multiwell = FPGA_ID_HS_MULTIWELL ,
  TbsiDacq = FPGA_ID_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 ,
  GrapheneFlagshipCore2 = FPGA_ID_GRAPHENE_FLAGSHIP_CORE_2_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	

Enumerator

MCSBUS3	
MCSBUS4	
MCSBUS5	
MCSBUS6	
MCSBUS7	
MCSBUS8	
MCSBUS9	
MCSBUS10	
MCSBUS11	
MCSBUS12	
MCSBUS13	
MCSBUS14	
MCSBUS15	
MCSBUS0	
BUSNUMBER1	
BUS1MCSBUS1	
BUS1MCSBUS2	
BUSNUMBER2	
BUS2MCSBUS1	
BUS2MCSBUS2	
PIC	
PIC2	
PIC3	
PIC4	
PIC5	
PIC6	
PIC7	
PIC8	
PIC9	
PIC10	
PIC11	
PIC12	
ChannelPIC	
Bootstrap	
BootstrapOtherCypress	
ALTERA	
FPGA2	
FPGA3	
FPGA4	
FPGA5	
FPGA6	
FPGA7	
FPGA8	
FPGA9	
FPGA10	
FPGA11	
FPGA12	
FPGA13	
FPGA14	
FPGA15	

Enumerator

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	
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	

Enumerator

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 CMOSMeaBathModeEnumNet `enum CMOSMeaBathModeEnumNet [strong]`

Enumerations of CMOS MEA Bath Mode

Enumerator

Ground	
Stimulation	
CurrentMeasure	

10.2.1.7 CMOSMeaHeadstage1NCBathCurrentEnumNet `enum CMOSMeaHeadstage1NCBathCurrentEnumNet [strong]`

Enumerations of CMOS MEA HS Current Monitoring Channels Group Bitmask

Enumerator

NCBathCurrent	

10.2.1.8 CMOSMeaHeadstage1NCCol2CurrentEnumNet `enum CMOSMeaHeadstage1NCCol2CurrentEnumNet [strong]`

Enumerations of CMOS MEA HS Current Monitoring Channels Group Bitmask

Enumerator

NCCol2Current	

10.2.1.9 CMOSMeaHeadstage1NChipTempEnumNet enum CMOSMeaHeadstage1NChipTempEnumNet [strong]

Enumerations of CMOS MEA HS Temperature Monitoring Channels Group Bitmask

Enumerator

NChipTemperature	

10.2.1.10 CMOSMeaHS1SidebandEnumNet enum CMOSMeaHS1SidebandEnumNet [strong]

Enumerations of CMOS MEA HS STG Sideband Channels Group Bitmask

Enumerator

SBSVector1	
SBSVector2	
SBSVector3	
SBSVector4	

10.2.1.11 CMOSMeaHS1TriggerStatusEnumNet enum CMOSMeaHS1TriggerStatusEnumNet [strong]

Enumerations of CMOS MEA HS STG Trigger Status Channels Group Bitmask

Enumerator

TriggerStatus1	
TriggerStatus2	
TriggerStatus3	
TriggerStatus4	

10.2.1.12 CMOSMeaIFDigChannelEnumNet enum CMOSMeaIFDigChannelEnumNet [strong]

Enumerations of CMOS MEA IF Digital Channels Group Bitmask

Enumerator

DigitalMux	
DigitalInPort	
DigitalOutReg	

Enumerator

FeedbackReg	
DigitalReg	
AuxPort	

10.2.1.13 CMOSMeaInterfaceADCEnumNet `enum CMOSMeaInterfaceADCEnumNet [strong]`

Enumerations of CMOS MEA IF Analog Channels Group Bitmask

Enumerator

IFChannel1	
IFChannel2	
IFChannel3	
IFChannel4	
IFChannel5	
IFChannel6	
IFChannel7	
IFChannel8	

10.2.1.14 CMOSMeaPacketFrameContextGroupEnumNet `enum CMOSMeaPacketFrameContextGroupEnumNet [strong]`

Enumerations of CMOS MEA HS STG Trigger Status Channels Group Bitmask

Enumerator

SOFAndCTRLword	
ChecksumAndPacketCounter	
Timestamp	
EOFAndCRC	

10.2.1.15 CMOSMeaSTG1DACSignalEnumNet `enum CMOSMeaSTG1DACSignalEnumNet [strong]`

Enumerations of CMOS MEA DAC Stimulation Channels Group Bitmask

Enumerator

DAC1Channel	
DAC2Channel	
DAC3Channel	
DAC4Channel	

10.2.1.16 CMOSMeaValueUnitEnumNet enum CMOSMeaValueUnitEnumNet [strong]

Enumerations of CMOS MEA Units of Values in Data stream

Enumerator

NoUnit	
NanoVolt	
PicoAmpere	
NanoAmpere	
MicroAmpere	
MilliDegreeCelsius	

10.2.1.17 DacqGroupChannelEnumNet enum DacqGroupChannelEnumNet [strong]

Enumerates the Channel Groups of Datastream

Enumerator

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

10.2.1.18 DacqMeaGroupTypeEnumNet enum DacqMeaGroupTypeEnumNet [strong]

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

Enumerator

AnalogGroup	
DigitalGroup	
FrameContextGroup	

10.2.1.19 DataModeEnumNet `enum DataModeEnumNet [strong]`

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

Enumerator

Unsigned_16bit	
Unsigned_24bit	
Unsigned_32bit	
Signed_16bit	
Signed_24bit	
Signed_32bit	

10.2.1.20 DeviceEnumNet `enum DeviceEnumNet [strong]`

Enumerates the group of MCS devices to connect to.

Enumerator

MCS_DEVICE_ANY	To connect to any MCS device.
MCS_GENERIC_DEVELOPMENT_DEVICE	Please use this only for MCS internal development.
MCS_DEVICE_USB	To connect to any MCS USB device.
MCS_MCCARD_DEVICE	Connect to an MC_Card.
MCS_STG_DEVICE	Connect to an MCS device with STG capability.
MCS_MC_STIMULUS_DEVICE	Devices which should be accessible from MC_Stimulus.
MCS_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.

Enumerator

MCS_RETINA_AMS_DONGLE	Connect to an MCS Retina AMS Dongle (Radio device)
MCS_PATHIDENT_DEVICE	Connect to an MCS Pathident device.
MCS_ROBO_DEVICE	Connect to an MCS Robo Platform device.
MCS_ROBOOCYTE2_DEVICE	Connect to an MCS Roboocyte2 device.
MCS_ROBOINJECT_DEVICE	Connect to an MCS RoboInject device.
MCS_HICLAMP_DEVICE	Connect to an MCS HiClamp device.
MCS_PATCHSERVER_DEVICE	Connect to an MCS PatchServer device.
MCS_ENCAPSULATOR_DEVICE	Connect to an MCS Encapsulator device.
MCS_MEASURETABLE_DEVICE	Connect to an MCS Encapsulator device.
MCS_FYI_DEVICE	Connect to an MCS FYI device.
MCS_HLA_DEVICE	Connect to an MCS HLA device.
MCS_PPS_DEVICE	Connect to an MCS PPS device.
MCS_PPS5_DEVICE	Connect to an MCS PPS5 device.
MCS_OKUVISION_STIMULATOR_DEVICE	Connect to an Okuvision Stimulator device.
MCS_NF_GEN_DEVICE	Connect to an MCS NF-Gen device.
MCS_SAFEIS_DEVICE	Connect to an MCS SafeIS device.
MCS_PERISTALTIC_PUMP_DEVICE	Connect to an MCS PeristalticPump device.
MCS_EXTERN_BC_TESTER_DEVICE	Connect to an ExternBCTester device.
MCS_EXTERN_D_TESTER_DEVICE	Connect to an ExternDTester device.
MCS_SOFTWARE_DONGLE_DEVICE	Connect to an Software Dongle device.
MCS_MEA_CLEAN_DEVICE	Connect to a MEA Clean device.
MCS_MEA_COAT_DEVICE	Connect to a MEA Clean device.
MCS_SMARTIMPLANT_DEVICE	Connect to a SmartImplant device.
MCS_MBC08_DEVICE	Connect to a MultiBatteryCharger device.
MCS_PEDOTER_DEVICE	Connect to a Pedoter device.
MCS_PPC_DEVICE	COnnect to a PPC device.
WARNER_VALVE_CONTROL_DEVICE	COnnect to a Warner Valve Control device.
WARNER_USSING_DEVICE	COnnect to a Warner Valve Control device.
HEKA_LIH3_DEVICE	Connect to a HEKA LIH3 device.
ALA_VC3_DEVICE	Connect to an ALA VC3 Valve Commander.
MCS_DEVICE_USB_CYPRESS	Connect to a Cypress USB device.

10.2.1.21 DigitalDatastreamEnableEnumNet `enum DigitalDatastreamEnableEnumNet [strong]`

enumerates the streams available as digital datastream

Enumerator

None	No digital datastream.
Mux	16 bits from the standard MUX datastream.
MuxOtherDevice	The 16 bits of the standard MUX datastream used by the other virtual device.
DigitalInReserverd	The lower 16 bits of the Digital IN port, these ports are on the device by default used as Digital OUT, thus not available as Digital IN.
DigitalIn	The upper 16 bits of the Digital IN port, use when Digital IN datastream is needed.
DigitalOut	The lower 16 bits of the Digital OUT port, use when Digital OUT datastream is needed.

Enumerator

DigitalOutReserved	The upper 16 bits of the Digital OUT port, these ports are on the device by default used as Digital IN, thus not available as Digital OUT.
RegisterLow	The lower 16 bits of the digital register.
RegisterHigh	The upper 16 bits of the digital register.
FeedbackLow	The lower 16 bits of the realtime feedback register.
FeedbackHigh	The upper 16 bits of the realtime feedback register.
Aux	The 2 bits of the AUX port.
PeriodicPulse	The 8 bits of the Periodic Pulse Generator (Video-Sync).
DigOutStim	The 16 bits of the Digital Out Stimulator.
Hs1Digital	Headstage 1 digital signals.
Hs1Trigger	Headstage 1 trigger signals.
Hs1SidebandLow	Headstage 1 lower 16 bits of sideband data.
Hs1SidebandHigh	Headstage 1 upper 16 bits of sideband data.
Hs2Digital	Headstage 2 digital signals.
Hs2Trigger	Headstage 2 trigger signals.
Hs2SidebandLow	Headstage 2 lower 16 bits of sideband data.
Hs2SidebandHigh	Headstage 2 upper 16 bits of sideband data.

10.2.1.22 DigitalSourceEnumNet `enum DigitalSourceEnumNet [strong]`

Enumerates the digital source of the MEA2100 device.

Enumerator

DigitalInOfOutPort	
DigitalIn	
DigitalPulse	
Feedback	
AuxIn	
Zero	
One	
HS1Trigger1Status	
HS1Trigger2Status	
HS1Trigger3Status	
HS1Trigger4Status	
HS1Trigger5Status	
HS1Trigger6Status	
HS1Sideband1	
HS1Sideband2	
HS1Sideband3	
HS1Sideband4	
HS1Sideband5	
HS1Sideband6	
HS2Trigger1Status	
HS2Trigger2Status	

Enumerator

HS2Trigger3Status	
HS2Trigger4Status	
HS2Trigger5Status	
HS2Trigger6Status	
HS2Sideband1	
HS2Sideband2	
HS2Sideband3	
HS2Sideband4	
HS2Sideband5	
HS2Sideband6	
PulseGenerator	
DigitalOutStimulator	
DigitalData	
DeviceRunStatus	
LastPosition	

10.2.1.23 DigitalStimulatorTriggerEventEnumNet `enum DigitalStimulatorTriggerEventEnumNet [strong]`

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

Enumerator

Start	
Stop	

10.2.1.24 DigitalStimulatorTriggerSlopeEnumNet `enum DigitalStimulatorTriggerSlopeEnumNet [strong]`

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

Enumerator

Falling	
Rising	

10.2.1.25 DigitalTargetEnumNet `enum DigitalTargetEnumNet [strong]`

Enumerates the Digital Targets for Digital Sources

Enumerator

Digout	
Digstream	
DacqTrigger	
StgTrigger	
StgListModeTrigger	
DigOutStimulatorStartTrigger	
DigOutStimulatorStopTrigger	
DigStreamToReceiver	

10.2.1.26 ElectrodeDacMuxEnumNet `enum ElectrodeDacMuxEnumNet [strong]`

Enumerates the setting of the Stimulation DAC Multiplexer.

Enumerator

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

10.2.1.27 ElectrodeModeEnumNet `enum ElectrodeModeEnumNet [strong]`

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

Enumerator

emAutomatic	
emManual	

10.2.1.28 enCMosMeaChipType `enum enCMosMeaChipType [strong]`

Enumerator

unknown	
nMos16LV	
nMos32LV	
nMos36LN	
nMos64LN	

10.2.1.29 EnSTG200x_STATUS enum EnSTG200x_STATUS [strong]

Enumerator

OK	
NOT_CONNECTED	
DEVICE_NOT_FOUND	

10.2.1.30 FilterAttributeEnumNet enum FilterAttributeEnumNet [strong]

Enumerator

PreCommaB	
PostCommaB	
CommaPositionB	
PreCommaA	
PostCommaA	
CommaPositionA	

10.2.1.31 FilterBandEnumNet enum FilterBandEnumNet [strong]

Enumerator

Unknown	
Lowpass	
Highpass	

10.2.1.32 FilterCalculationDirectionEnumNet enum FilterCalculationDirectionEnumNet [strong]

Enumerator

DoubleToInt	
IntToDouble	

10.2.1.33 FilterFamilyEnumNet enum FilterFamilyEnumNet [strong]

Enumerator

Unknown	
Bessel	
Butterworth	
RC	

10.2.1.34 FilterTypeEnumNet `enum FilterTypeEnumNet [strong]`

Enumerator

Hardware	
Software	

10.2.1.35 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	

Enumerator

Mea2100Mini60ECP5Headstage	
eCubeHeadstage	
Me2100Graphene16_32Headstage	
GrapheneFlagshipCore2Headstage	
WholeCellPatchHeadstage	
InterfaceBoard2	
W2100IFB2	
CmosmeaIFB2	
Mea2100LiteHeadstage	
LIH30Interfaceboard	
LIH30ADCCtrl	
UssingRail	
UssingChamber	
IFB2GoldenInterfaceboard	
IFB30GoldenInterfaceboard	
DeviceHasNoHeadstage	

10.2.1.36 HeadstageIdEnumNet `enum HeadstageIdEnumNet [strong]`

Enumerator

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

10.2.1.37 IoVoltageEnumNet `enum IoVoltageEnumNet [strong]`

enumerates the I/O Voltages available on the IFB2

Enumerator

Voltage_3V3	
Voltage_5V0	

10.2.1.38 LIH30_ADC_Channel_EnumNet enum LIH30_ADC_Channel_EnumNet [strong]

Enumerator

User_ADC_0	
User_ADC_1	
User_ADC_2	
User_ADC_3	
User_ADC_4	
Test_ADC_EPC10	
ModulA_ADC0	
ModulA_ADC1	
ModulA_ADC2	
ModulA_ADC3	
ModulB_ADC0	
ModulB_ADC1	
ModulB_ADC2	
ModulB_ADC3	
ModulC_ADC0	
ModulC_ADC1	
ModulC_ADC2	
ModulC_ADC3	
ModulD_ADC0	
ModulD_ADC1	
ModulD_ADC2	
ModulD_ADC3	

10.2.1.39 LIH30_DAC_Channel_EnumNet enum LIH30_DAC_Channel_EnumNet [strong]

Enumerator

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

10.2.1.40 LIH30_EPC10_Bus_EnumNet enum LIH30_EPC10_Bus_EnumNet [strong]

Enumerator

A	
B	

10.2.1.41 MbcChannelStateEnumNet enum MbcChannelStateEnumNet [strong]

Enumerator

csIdleNoBattery	
csIdleChargeFinished	
csCapacityTestPrecharge	
csCapacityTestDischarge	
csRefreshBattery	
csCharge	
csDischarge	
csError	

10.2.1.42 MbcChargingModeEnumNet enum MbcChargingModeEnumNet [strong]

Enumerator

StorageCharge	
FullCharge	

10.2.1.43 MbcRatedCapacityEnumNet enum MbcRatedCapacityEnumNet [strong]

Enumerator

rc30mAh	
rc100mAh	
rc200mAh	
rc300mAh	
rcGreater300mAh	

10.2.1.44 McsBusTypeEnumNet enum McsBusTypeEnumNet [strong]

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

Enumerator

MCS_ANY_BUS	
MCS_UNDEFINED_BUS	
MCS_USB_BUS	
MCS_PCI_BUS	

10.2.1.45 McsUsbSpeedEnumNet `enum McsUsbSpeedEnumNet [strong]`

Enumerates the current connection speed of the device

Enumerator

LowSpeed	
FullSpeed	
HighSpeed	
SuperSpeed	
UnknownSpeed	

10.2.1.46 MEA2100_256DacqGroupChannelEnumNet `enum MEA2100_256DacqGroupChannelEnumNet [strong]`

Enumerates the MEA2100-256 Channel Groups of Datastream

Enumerator

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

10.2.1.47 MEA2100_256DigitalSourceEnumNet `enum MEA2100_256DigitalSourceEnumNet [strong]`

Enumerates the digital source of the MEA2100-256 device.

Enumerator

DigitalInOfOutPort	
DigitalIn	
DigitalPulse	
Feedback	
AuxIn	
Zero	
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	

Enumerator

HS2Trigger1Status	
HS2Trigger2Status	
HS2Trigger3Status	
HS2Trigger4Status	
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 MeaLayoutEnumNet `enum MeaLayoutEnumNet [strong]`

Enumerates the MEA layout of the MEA2100 device.

Enumerator

mlUnknown	
mlMEA60	

10.2.1.49 MultiwellPlateTypeEnumNet enum MultiwellPlateTypeEnumNet [strong]

Enumerator

Plate_Dummy	
Plate_24W700_100FMA	
Plate_24W030MGA	
Plate_72W500_100PMA	
Plate_72W500_100FMA	
Plate_24W700_100FMB	
Plate_96W700_100FMA	
Plate_96W300_80_1152FMA	
Plate_96W400_80_1152FMB	
Plate_24W300_30_1152GBA	
Plate_24W700_100FMC	
Plate_96W700_100FMB	
Plate_96W700_100GBC	
Plate_96W700_100GBD	
Plate_24W700_100PBA	
Plate_Dummy_126	
Plate_24W300_30GMA	
Plate_96W700_100GMA	
Plate_24W300_30GBA	
Plate_96W700_100GBA	
Plate_24W300_30GBB	
Plate_96W700_100GBB	
No_Plate	

10.2.1.50 PatchServAdcModeEnumNet enum PatchServAdcModeEnumNet [strong]

Enumerator

Normal	
CatchAmp	

10.2.1.51 PlateClampEnumNet enum PlateClampEnumNet [strong]

Enumerator

Close	
Open	
Stop	

10.2.1.52 PlateClampLockEnumNet enum PlateClampLockEnumNet [strong]

Enumerator

Lock	
Unlock	

10.2.1.53 PortDirectionEnumNet enum PortDirectionEnumNet [strong]

enumerates a port direction

Enumerator

Output	
Input	

10.2.1.54 PP_Pump_Mode_Type_EnumNet enum PP_Pump_Mode_Type_EnumNet [strong]

Enumerator

Manual	
Digital	
Analog	

10.2.1.55 ProductIdEnumNet enum ProductIdEnumNet [strong]

Enumerates the group of MCS devices to connect to.

Enumerator

Any	
None	
ALA_VC3	
Cypress_FX1	
Cypress_FX2	
Cypress_FX3	
MC_Card	
Campden_Ci4600EphysVideoDataIntegrator	
HekaLIH30	

Enumerator

HekaEPC10Single	
HekaEPC10Double	
HekaEPC10Triple	
HekaEPC10Quadro	
HekaLIH406	
HekaLIH816	
HekaITEV100	
HekaPG610	
HekaPG611	
HekaPG612	
HekaPG618	
HekaPG690	
HekaEPCLite	
STG	
Octopot	
Tersens	
Dotriapot	
HLA	
STG400x	
STG4002	
STG4004	
STG4008	
STG400x_opto	
STG4002_opto	
STG4004_opto	
STG4008_opto	
STG5	
STG3008_FA	
MultiwellOptoStim	
Generic	
PGA	
PCX	
TCX	
FCX	
FCB	
TC01	
TC02	
Retina_LED	
AMS_Dongle	
Okuvision_Stimulator	
ExternBCTester	
Triggerbox_IMS	
Triggerbox_AMS	
Triggerbox_AMS3	
ExternDTester	
FunkDongleS	
ExternSTester	
DongleS	
Triggerbox_R5	

Enumerator

MEA_Switch	
MEA_Impedance	
ChannelTest	
Sw2to64	
PeristalticPump	
MEA_Switch_2_1	
MEA_Switch_4_2	
PPS4plus1	
PPS5	
PPS2	
PPS5_DIG	
MEA_Clean	
MEA_Coat	
Multiwell_ICC	
MBC08	
PPC	
MEA1060	
MEA_Sanofi	
ME256	
ME128	
ME64	
ME32	
ME16	
MEA2100_Mini_Usb_develop	
MEA256	
MEA2100	
MEA2100_32	
MEA2100_Lite	
Multiwell	
MEA2100_256	
ME2100	
MEA2100BetaScreen	
MEA2100_Mini	
TBSI_Dacq	
Multiwell_MEA_Mini	
Whole_Cell_Patch	
eCube	
Graphene_FlagShip_Core_2	
GE2100	
Multiboot	
WPA8	
WPA4	
WPA16	
WPA32	
W2100	
NeuroChip	
UsbTest	
SoftwareDongle	
PathIdent	
NF_Gen	

Enumerator

SafeIS	
Encapsulator	
NeurochipConfig	
MeasureTable	
Robooycte2	
RoboInject	
HiClamp	
PatchServer	
Dilutor	
HiClamp4Uart	
IM16S16KRA	
IM64KRB	
IS32KRA	
IM64KRC	
IM16S8KRA	
IM16KRC	
SmartImplant	
PositionImp	
PositionBase	
PositionIICentralUnit	
PositionIIBase	
GrapheneProjectTestDevice	
Pos900	
Neptun	
Warner_Valve_Control	
Warner_TEER_Machine	
Warner_Ussing	

10.2.1.56 PulseGenerator_Mode_EnumNet `enum PulseGenerator_Mode_EnumNet [strong]`

Enumerator

Off	
AlwaysOn	
Gated_Low_Active	
Gated_High_Active	

10.2.1.57 ReferenceElectrodeModeEnumNet `enum ReferenceElectrodeModeEnumNet [strong]`

enumerates the electrode subtraction modes

Enumerator

SubtractionOff	
SubtractFromAllOther	
SubtractFromReferenceElectrodeOnly	
SubtractFromAll	

10.2.1.58 ReferenceElectrodeSwitchPositionEnumNet `enum ReferenceElectrodeSwitchPositionEnumNet [strong]`

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

Enumerator

off	
Ref8	
Ref16	
Ref24	
Ref32	

10.2.1.59 RetriggerActionEnumNet `enum RetriggerActionEnumNet [strong]`

Enumerates possible retrigger actions for STG200x devices.

Enumerator

raStop	
raRestart	
raIgnore	
raGate	
raSingle	

10.2.1.60 RoboCurrentModeEnumNet `enum RoboCurrentModeEnumNet [strong]`

Enumerator

Off	
Break	
Standby	
Reference	
Movement	

10.2.1.61 SampleDstSizeNet enum SampleDstSizeNet [strong]

Enumerates the destination data format for ChannelBlock functions.

Enumerator

SampleDstSize16	
SampleDstSize32	

10.2.1.62 SampleSizeNet enum SampleSizeNet [strong]

Enumerates the data format for ChannelBlock functions.

Enumerator

SampleSize16Unsigned	
SampleSize16Signed	
SampleSize24Unsigned	
SampleSize24Signed	
SampleSize32Unsigned	
SampleSize32Signed	
SampleSize64Unsigned	
SampleSize64Signed	

10.2.1.63 SCU_HeadstageIdEnumNet enum SCU_HeadstageIdEnumNet [strong]

Enumerator

DeviceNotConnected	
Me2100_32Xilinx	
Me2100_32PICiCE40	
Mea2100Mini60PICiCE40	
Mea2100BetaScreen	
Me2100UPA32	
MultiwellMini	
Mea2100Mini120	
Mea2100Mini60ECP5	
eCube	
Me2100Graphene16_32	
GrapheneFlagshipCore2	
WholeCellPatch	
DeviceHasNoHeadstage	

10.2.1.64 SCUDacqGroupChannelEnumNet enum SCUDacqGroupChannelEnumNet [strong]

Enumerates the SCU Channel Groups of Datastream

Enumerator

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

10.2.1.65 SCUDigitalSourceEnumNet enum SCUDigitalSourceEnumNet [strong]

Enumerates the digital source of the SCU device.

Enumerator

DigitalInOfOutPort	
DigitalIn	
DigitalPulse	
Feedback	
AuxIn	
Zero	
One	
PulseGenerator	
DigitalOutStimulator	
DigitalData	
DeviceRunStatus	
HS1Trigger1Status	

Enumerator

HS1Trigger2Status	
HS1Trigger3Status	
HS1Trigger4Status	
HS1Trigger5Status	
HS1Trigger6Status	
HS1Trigger7Status	
HS1Trigger8Status	
HS1Trigger9Status	
HS1Trigger10Status	
HS1Trigger11Status	
HS1Trigger12Status	
HS1Sideband1	
HS1Sideband2	
HS1Sideband3	
HS1Sideband4	
HS1Sideband5	
HS1Sideband6	
HS1Sideband7	
HS1Sideband8	
HS1Sideband9	
HS1Sideband10	
HS1Sideband11	
HS1Sideband12	
HS2Trigger1Status	
HS2Trigger2Status	
HS2Trigger3Status	
HS2Trigger4Status	
HS2Trigger5Status	
HS2Trigger6Status	
HS2Trigger7Status	
HS2Trigger8Status	
HS2Trigger9Status	
HS2Trigger10Status	
HS2Trigger11Status	
HS2Trigger12Status	
HS2Sideband1	
HS2Sideband2	
HS2Sideband3	
HS2Sideband4	
HS2Sideband5	
HS2Sideband6	
HS2Sideband7	
HS2Sideband8	
HS2Sideband9	
HS2Sideband10	
HS2Sideband11	
HS2Sideband12	
LastPosition	

10.2.1.66 Stg200xDigoutModeEnumNet enum Stg200xDigoutModeEnumNet [strong]

Enumerates the DigoutMode on STG400x devices.

Enumerator

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

10.2.1.67 Stg200xSegmentFlagsEnumNet enum Stg200xSegmentFlagsEnumNet [strong]

Enumerates Segmentflag options for STG400x devices.

Enumerator

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

10.2.1.68 Stg200xTriggerStatusEnumNet enum Stg200xTriggerStatusEnumNet [strong]

Enumerates the STG download mode trigger status

The STG maintains the status for each of the STG200x_NUM_TRIGGER triggers

Enumerator

Idle	
Running	
Finished	
Armed	

10.2.1.69 STG_DestinationEnumNet enum STG_DestinationEnumNet [strong]

Enumerates the destination for STG downloads.

Enumerator

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

10.2.1.70 StimulationLayoutConfigurationEnumNet enum StimulationLayoutConfigurationEnumNet [strong]

enumerates the layout configuration for the MEA2100-256 device

Enumerator

SingleWell	
SixWell	
NineWell	

10.2.1.71 TBSI_DACQDigitalSourceEnumNet enum TBSI_DACQDigitalSourceEnumNet [strong]

Enumerates the digital source of the TBSI-DACQ device.

Enumerator

DigitalInOfOutPort	
DigitalIn	
DigitalPulse	
Feedback	
AuxIn	
Zero	
One	

Enumerator

DeviceRunStatus	
PulseGenerator	
DigitalOutStimulator	
DigitalData	
HS1DigitalData1	
HS2DigitalData1	
LastPosition	

10.2.1.72 TcxDeviceTypeEnumNet `enum TcxDeviceTypeEnumNet [strong]`

Enumerates the type of TCX devices.

Enumerator

Unknown	
Regular	
BMI	
Nanion	
Warner	

10.2.1.73 TcxSensorTypeEnumNet `enum TcxSensorTypeEnumNet [strong]`

Enumerates the sensor types for TCX devices

Enumerator

Reserved5	
Reserved4	
Reserved3	
Reserved2	
Reserved1	
NTC10K	
PT1000	
PT100	

10.2.1.74 TeerClampModeEnumNet `enum TeerClampModeEnumNet [strong]`

Enumerator

ClampModeVoltage	

Enumerator

ClampModeCurrent	
ClampModeOpen	
ClampModeInternalCalibration	

10.2.1.75 TeerWaveformEnumNet `enum TeerWaveformEnumNet [strong]`

Enumerator

Rectangle	
Sine	

10.2.1.76 TriggerSourceEnumNet `enum TriggerSourceEnumNet [strong]`

Enumerates the trigger source of the MEA2100 device.

Enumerator

tsNone	
tsDigitalIn1	
tsDigitalIn2	
tsDigitalIn3	
tsDigitalIn4	
tsDigitalIn5	
tsDigitalIn6	
tsDigitalIn7	
tsDigitalIn8	
tsDigitalIn9	
tsDigitalIn10	
tsDigitalIn11	
tsDigitalIn12	
tsDigitalIn13	
tsDigitalIn14	
tsDigitalIn15	
tsDigitalIn16	
tsDigitalIn17	
tsDigitalIn18	
tsDigitalIn19	
tsDigitalIn20	
tsDigitalIn21	
tsDigitalIn22	
tsDigitalIn23	
tsDigitalIn24	

Enumerator

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	
tsDigitalPuse5	
tsDigitalPuse6	
tsDigitalPuse7	

Enumerator

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	

10.2.1.77 UsbVendorIdEnumNet `enum UsbVendorIdEnumNet [strong]`

Enumerator

Unknown	
None	
Renesas	
ASMedia	
Intel	

10.2.1.78 UssingClampModeEnumNet `enum UssingClampModeEnumNet [strong]`

Enumerator

VoltageClamp	
CurrentClamp	
OpenClamp	
Standby	
ElectrodeOffset	

10.2.1.79 UssingUnitEnumNet `enum UssingUnitEnumNet [strong]`

Enumerator

Volt	
Ampere	
State	

10.2.1.80 VendorIdEnumNet `enum VendorIdEnumNet [strong]`

Enumerates the group of MCS devices to connect to.

Enumerator

Any	
None	
MCS	
PCI	
Cypress	
ALA_VC3	

10.2.1.81 W2100_Accel_Gyro_Select_EnumNet `enum W2100_Accel_Gyro_Select_EnumNet [strong]`

enumerates the accelerometer configuration on the W2100 device

Enumerator

Off	
GyroOnly	
AccelOnly	
Both	

10.2.1.82 W2100DacqGroupChannelEnumNet enum W2100DacqGroupChannelEnumNet [strong]

Enumerates the W2100 Channel Groups of Datastream

Enumerator

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

Enumerator

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	

10.2.1.83 W2100DigitalSourceEnumNet `enum W2100DigitalSourceEnumNet [strong]`

Enumerates the digital source of the W2100 device.

Enumerator

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

10.2.1.84 WvcDisplayModeEnumNet enum WvcDisplayModeEnumNet [strong]

enumerates Wvc display mode

Enumerator

Work	
PC	
Settings	
TouchTest	

10.2.1.85 WvcValveModeEnumNet enum WvcValveModeEnumNet [strong]

enumerates Wvc valve mode

Enumerator

Manual	
Digital	
Analog	
Table	

10.2.2 Function Documentation

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

```
CMcsUsbDacqNet^ 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]`

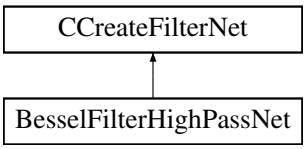
11.2.1.3 ChargeString `System:: String^ ChargeString [get]`

11.2.1.4 Voltage `unsigned int Voltage [get]`

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

11.3 BesselFilterHighPassNet Class Reference

Inheritance diagram for BesselFilterHighPassNet:



Public Member Functions

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

Additional Inherited Members

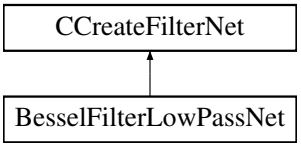
11.3.1 Constructor & Destructor Documentation

11.3.1.1 BesselFilterHighPassNet() `BesselFilterHighPassNet (`

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

11.4 BesselFilterLowPassNet Class Reference

Inheritance diagram for BesselFilterLowPassNet:



Public Member Functions

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

Additional Inherited Members

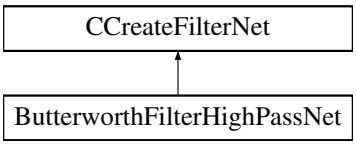
11.4.1 Constructor & Destructor Documentation

11.4.1.1 BesselFilterLowPassNet() BesselFilterLowPassNet (

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

11.5 ButterworthFilterHighPassNet Class Reference

Inheritance diagram for ButterworthFilterHighPassNet:



Public Member Functions

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

Additional Inherited Members

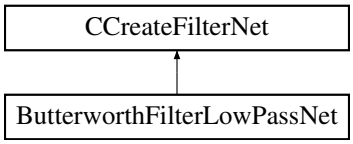
11.5.1 Constructor & Destructor Documentation

11.5.1.1 ButterworthFilterHighPassNet() ButterworthFilterHighPassNet (

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

11.6 ButterworthFilterLowPassNet Class Reference

Inheritance diagram for ButterworthFilterLowPassNet:



Public Member Functions

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

Additional Inherited Members

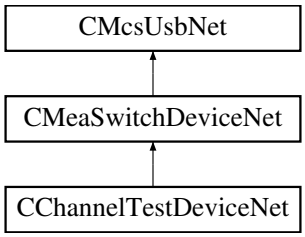
11.6.1 Constructor & Destructor Documentation

11.6.1.1 ButterworthFilterLowPassNet()

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

11.7 CChannelTestDeviceNet Class Reference

Inheritance diagram for CChannelTestDeviceNet:



Public Member Functions

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

Additional Inherited Members

11.7.1 Constructor & Destructor Documentation

11.7.1.1 CChannelTestDeviceNet() `CChannelTestDeviceNet ( )`

11.7.1.2 ~CChannelTestDeviceNet() `~CChannelTestDeviceNet ( )`

11.7.2 Member Function Documentation

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

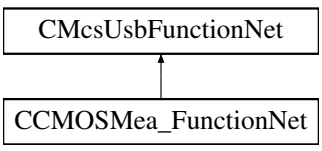
11.7.2.2 SetAttenuation() `void SetAttenuation ( unsigned int Attenuation )`

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

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

11.8 CCMOSMea_FunctionNet Class Reference

Inheritance diagram for CCMOSMea_FunctionNet:



Public Member Functions

- CCMOSMea_FunctionNet (CMcsUsbNet^ 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 ChannelNumber)
- uint32_t GetGroupChannelBitmaskBySelect (DacqGroupChannelEnumNet GroupID, uint32_t ChannelNumber, uint32_t virtualDevice)
- CMOSMeaInterfaceADCEnumNet GetGroupChannelBitmaskInterfaceADC (uint32_t ChannelNumber)
- CMOSMeaInterfaceADCEnumNet GetGroupChannelBitmaskInterfaceADC (uint32_t ChannelNumber, uint32_t virtualDevice)
- CMOSMeaIFDigChannelEnumNet GetGroupChannelBitmaskIFDigChannels (uint32_t ChannelNumber)
- CMOSMeaIFDigChannelEnumNet GetGroupChannelBitmaskIFDigChannels (uint32_t ChannelNumber, uint32_t virtualDevice)
- CMOSMeaHeadstage1NCBathCurrentEnumNet GetGroupChannelBitmaskHS1NCBathCurrent (uint32_t ChannelNumber)
- CMOSMeaHeadstage1NCBathCurrentEnumNet GetGroupChannelBitmaskHS1NCBathCurrent (uint32_t ChannelNumber, uint32_t virtualDevice)
- CMOSMeaHeadstage1NCCol2CurrentEnumNet GetGroupChannelBitmaskHS1NCCol2Current (uint32_t ChannelNumber)
- CMOSMeaHeadstage1NCCol2CurrentEnumNet GetGroupChannelBitmaskHS1NCCol2Current (uint32_t ChannelNumber, uint32_t virtualDevice)
- CMOSMeaHeadstage1NChipTempEnumNet GetGroupChannelBitmaskHS1NChipTemp (uint32_t ChannelNumber)
- CMOSMeaHeadstage1NChipTempEnumNet GetGroupChannelBitmaskHS1NChipTemp (uint32_t ChannelNumber, uint32_t virtualDevice)
- CMOSMeaSTG1DACSignalEnumNet GetGroupChannelBitmaskSTG1DACSignal (uint32_t 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

11.8.1 Constructor & Destructor Documentation

11.8.1.1 CCMOSMea_FunctionNet() [1/2] CCMOSMea_FunctionNet (

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

11.8.1.2 CCMOSMea_FunctionNet() [2/2] CCMOSMea_FunctionNet (

```
CMcsUsbNet^ mcsusb )
```

11.8.2 Member Function Documentation

11.8.2.1 AreTransistorVoltagesSet() bool AreTransistorVoltagesSet ( )

11.8.2.2 ClearSTGOutput() void ClearSTGOutput (

```
uint32_t Channel )
```

11.8.2.3 DetectChipType() enCMosMeaChipType DetectChipType ( )

11.8.2.4 EnableChannelsInGroup() [1/2] void EnableChannelsInGroup (

```
DacqGroupChannelEnumNet GroupID,
List< bool >^ EnabledChannelsBitMap )
```

11.8.2.5 EnableChannelsInGroup() [2/2] void EnableChannelsInGroup (

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

11.8.2.6 GetADCInputOffset() int32_t GetADCInputOffset ( )

11.8.2.7 GetBath() int32_t GetBath ( )

11.8.2.8 GetBathMode() CMOSMeaBathModeEnumNet GetBathMode ( )

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

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

11.8.2.11 GetGate() int32_t GetGate ( )

11.8.2.12 GetGNDI() int32_t GetGNDI ( )

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

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

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

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

11.8.2.17 GetGroupChannelBitmaskHS1NCBathCurrent() [1/2] CMOSMeaHeadstage1NCBathCurrentEnumNet

```
GetGroupChannelBitmaskHS1NCBathCurrent (
            uint32_t ChannelNumber )
```

11.8.2.18 GetGroupChannelBitmaskHS1NCBathCurrent() [2/2] CMOSMeaHeadstage1NCBathCurrentEnumNet

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

11.8.2.19 GetGroupChannelBitmaskHS1NCCol2Current() [1/2] CMOSMeaHeadstage1NCCol2CurrentEnumNet

```
GetGroupChannelBitmaskHS1NCCol2Current (
            uint32_t ChannelNumber )
```

11.8.2.20 GetGroupChannelBitmaskHS1NCCol2Current() [2/2] CMOSMeaHeadstage1NCCol2CurrentEnumNet

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

11.8.2.21 GetGroupChannelBitmaskHS1NChipTemp() [1/2] CMOSMeaHeadstage1NChipTempEnumNet GetGroupChannelBitmaskHS1NChipTemp

```
GetGroupChannelBitmaskHS1NChipTemp (
            uint32_t ChannelNumber )
```

11.8.2.22 GetGroupChannelBitmaskHS1NChipTemp() [2/2] CMOSMeaHeadstage1NChipTempEnumNet GetGroupChannelBitmaskHS1NChipTemp

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

11.8.2.23 GetGroupChannelBitmaskHS1Sidebands() [1/2] CMOSMeaHS1SidebandEnumNet GetGroupChannelBitmaskHS1Sidebands

```
GetGroupChannelBitmaskHS1Sidebands (
            uint32_t ChannelNumber )
```

11.8.2.24 GetGroupChannelBitmaskHS1Sidebands() [2/2] CMOSMeaHS1SidebandEnumNet GetGroupChannelBitmaskHS1Sidebands

```
GetGroupChannelBitmaskHS1Sidebands (
            uint32_t ChannelNumber,
            uint32_t virtualDevice )
```

11.8.2.25 GetGroupChannelBitmaskHS1TriggerStatus() [1/2] CMOSMeaHS1TriggerStatusEnumNet GetGroupChannelBitmaskHS1TriggerStatus (

```
            uint32_t ChannelNumber )
```

11.8.2.26 GetGroupChannelBitmaskHS1TriggerStatus() [2/2] CMOSMeaHS1TriggerStatusEnumNet GetGroupChannelBitmaskHS1TriggerStatus (

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

11.8.2.27 GetGroupChannelBitmaskIFDigChannels() [1/2] CMOSMeaIFDigChannelEnumNet GetGroupChannelBitmaskIFDigChannels (

```
            uint32_t ChannelNumber )
```

11.8.2.28 GetGroupChannelBitmaskIFDigChannels() [2/2] CMOSMeaIFDigChannelEnumNet GetGroupChannelBitmaskIFDigChannels (

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

11.8.2.29 GetGroupChannelBitmaskInterfaceADC() [1/2] CMOSMeaInterfaceADCEnumNet GetGroupChannelBitmaskInterfaceADC (

```
            uint32_t ChannelNumber )
```

11.8.2.30 GetGroupChannelBitmaskInterfaceADC() [2/2] CMOSMeaInterfaceADCEnumNet GetGroupChannelBitmaskInterfaceADC (

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

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

```
            uint32_t ChannelNumber )
```

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

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

11.8.2.33 GetGroupChannelBitmaskSTG1DACSignal() [1/2]

```
CMOSMeaSTG1DACSignalEnumNet GetGroupChannelBitmaskSTG1DACSignal (
            uint32_t ChannelNumber )
```

11.8.2.34 GetGroupChannelBitmaskSTG1DACSignal() [2/2]

```
CMOSMeaSTG1DACSignalEnumNet GetGroupChannelBitmaskSTG1DACSignal (
            uint32_t ChannelNumber,
            uint32_t virtualDevice )
```

11.8.2.35 GetGroupDCOffset() [1/2]

```
int32_t GetGroupDCOffset (
            DacqGroupChannelEnumNet GroupID )
```

11.8.2.36 GetGroupDCOffset() [2/2]

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

11.8.2.37 GetGroupID() [1/2]

```
DacqGroupChannelEnumNet GetGroupID (
            uint32_t Index )
```

11.8.2.38 GetGroupID() [2/2]

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

11.8.2.39 GetGroupNumberOfChannels() [1/2]

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

11.8.2.40 GetGroupNumberOfChannels() [2/2]

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

11.8.2.41 GetGroupResolutionPerDigit() [1/2]

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

11.8.2.42 GetGroupResolutionPerDigit() [2/2]

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

11.8.2.43 GetGroupSampleSize() [1/2]

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

11.8.2.44 GetGroupSampleSize() [2/2]

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

11.8.2.45 GetGroupType() [1/2]

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

11.8.2.46 GetGroupType() [2/2]

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

11.8.2.47 GetGroupUnit() [1/2]

```
CMOSMeaValueUnitEnumNet GetGroupUnit (
    DacqGroupChannelEnumNet GroupID )
```

11.8.2.48 GetGroupUnit() [2/2]

```
CMOSMeaValueUnitEnumNet GetGroupUnit (
    DacqGroupChannelEnumNet GroupID,
    uint32_t virtualDevice )
```

11.8.2.49 GetMaxNumOfColumns()

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

11.8.2.50 GetNeurochipMemoryData() [1/2]

```
uint32_t GetNeurochipMemoryData (
    uint16_t MemAddress )
```

11.8.2.51 GetNeurochipMemoryData() [2/2]

```
array<uint32_t> ^ GetNeurochipMemoryData (
    uint16_t MemAddress,
    uint32_t ReqestLength )
```

11.8.2.52 GetNeurochipMemorySize()

```
uint32_t GetNeurochipMemorySize ( )
```

11.8.2.53 GetNumberOfSupportedGroups() [1/2]

```
uint32_t GetNumberOfSupportedGroups ( )
```

11.8.2.54 GetNumberOfSupportedGroups() [2/2]

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

11.8.2.55 GetSourceBulk()

```
int32_t GetSourceBulk ( )
```

11.8.2.56 GetSourceDrain()

```
int32_t GetSourceDrain ( )
```

11.8.2.57 GetSourceGate()

```
int32_t GetSourceGate ( )
```

11.8.2.58 GetStimulusSites()

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

11.8.2.59 GetVDD3I()

```
int32_t GetVDD3I ( )
```

11.8.2.60 GetVDDI()

```
int32_t GetVDDI ( )
```

11.8.2.61 IsChipPowered()

```
bool IsChipPowered ( )
```

11.8.2.62 IsGateFloating() bool IsGateFloating ( )

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

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

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

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

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

11.8.2.68 SetGateFloating() void SetGateFloating ( )

11.8.2.69 SetGateToVOP() void SetGateToVOP ( )

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

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

11.8.2.72 SetSourceBulk()

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

11.8.2.73 SetSourceDrain()

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

11.8.2.74 SetSourceGate()

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

11.8.2.75 SetStimulusSites()

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

11.8.2.76 UpdateTransistorVoltages()

```
void UpdateTransistorVoltages ( )
```

11.8.2.77 VOPSTimerSetResetTimes() [1/2]

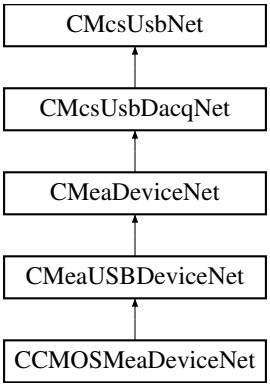
```
void VOPSTimerSetResetTimes (
    uint32_t ResetTime,
    uint32_t IntervalTime )
```

11.8.2.78 VOPSTimerSetResetTimes() [2/2]

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

11.9 CCMOSMeaDeviceNet Class Reference

Inheritance diagram for CCMOSMeaDeviceNet:



Classes

- class CRegionOfInterestRect

Public Member Functions

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

Properties

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

Additional Inherited Members

11.9.1 Constructor & Destructor Documentation

11.9.1.1 CCMOSMeaDeviceNet() `CCMOSMeaDeviceNet ( void )`

11.9.1.2 ~CCMOSMeaDeviceNet() `~CCMOSMeaDeviceNet ( )`

11.9.2 Member Function Documentation

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

11.9.2.2 GetBaseSamplerate() `int GetBaseSamplerate ( )`

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

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

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

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

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

11.9.2.8 GetMaxReadableColumns() `int GetMaxReadableColumns ( )`

11.9.2.9 SetBaseSamplerate()

```
void SetBaseSamplerate (
    int BaseSamplerate )
```

11.9.2.10 SetRegionOfInterests()

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

11.9.2.11 UpdateChannelBlock()

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

11.9.3 Property Documentation

11.9.3.1 CMosMea

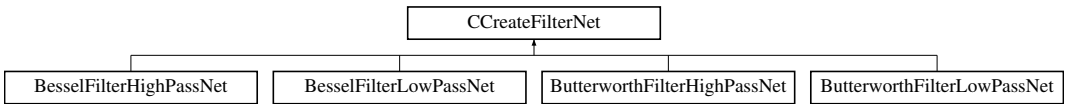
```
CCMOSMea_FunctionNet^ CMosMea [get]
```

11.9.3.2 Stimulus

```
CStimulusFunctionNet^ Stimulus [get]
```

11.10 CCreateFilterNet Class Reference

Inheritance diagram for CCreateFilterNet:



Public Member Functions

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

Static Public Member Functions

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

Protected Member Functions

- CCreateFilterNet (int numCoefSets, CCreateFilter *pCreateFilter)

Properties

- int NumCoefSets [get]
- int Order [get]
- double SampleRate [get]
- double CutoffFrequency [get]
- double Scale [get]

11.10.1 Constructor & Destructor Documentation

11.10.1.1 CCreateFilterNet() [1/2]

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

11.10.1.2 ~CCreateFilterNet()

```
~CCreateFilterNet ( )
```

11.10.1.3 CCreateFilterNet() [2/2]

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

11.10.2 Member Function Documentation

11.10.2.1 FindFilter() [1/2]

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

11.10.2.2 FindFilter() [2/2] static int FindFilter (

```
array< CFilterCoefficientsNet^>^ coef,
array< CCreateFilterNet^>^ param ) [static]
```

11.10.2.3 GetBiQuad() CFilterCoefficientsNet ^ GetBiQuad (

```
int index )
```

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

11.10.3 Property Documentation

11.10.3.1 CutoffFrequency double CutoffFrequency [get]

11.10.3.2 NumCoefSets int NumCoefSets [get]

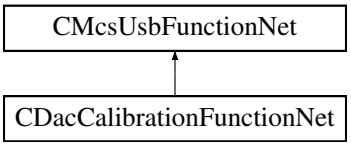
11.10.3.3 Order int Order [get]

11.10.3.4 SampleRate double SampleRate [get]

11.10.3.5 Scale double Scale [get]

11.11 CDacCalibrationFunctionNet Class Reference

Inheritance diagram for CDacCalibrationFunctionNet:



Public Member Functions

- CDacCalibrationFunctionNet (CMcsUsbNet^ 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

11.11.1 Detailed Description

11.11.2 Constructor & Destructor Documentation

11.11.2.1 CDacCalibrationFunctionNet() [1/2]

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

Initializes a new instance of the CDacCalibrationFunctionNet class.

11.11.2.2 CDacCalibrationFunctionNet() [2/2]

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

11.11.2.3 ~CDacCalibrationFunctionNet()

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

11.11.2.4 "!CDacCalibrationFunctionNet()

```
!CDacCalibrationFunctionNet ( )
```

11.11.3 Member Function Documentation

11.11.3.1 BurnDacOffset()

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

Writes the offset of a DAC channel to permanent memory.

Parameters

dacChannel	The DAC channel number.

11.11.3.2 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.

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

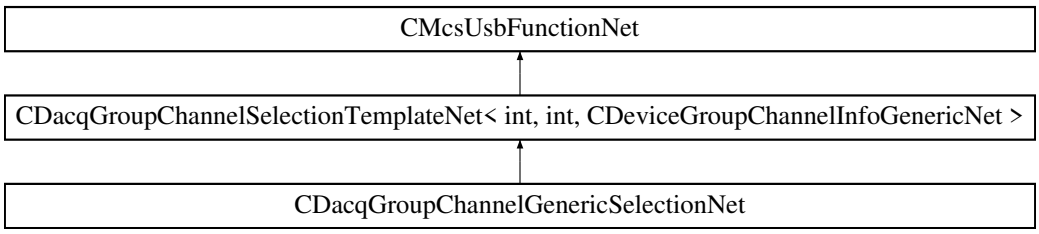
Sets the offset of a DAC channel.

Parameters

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

11.12 CDacqGroupChannelGenericSelectionNet Class Reference

Inheritance diagram for CDacqGroupChannelGenericSelectionNet:



Public Member Functions

- CDacqGroupChannelGenericSelectionNet (CMcsUsbNet^ mcsusb)

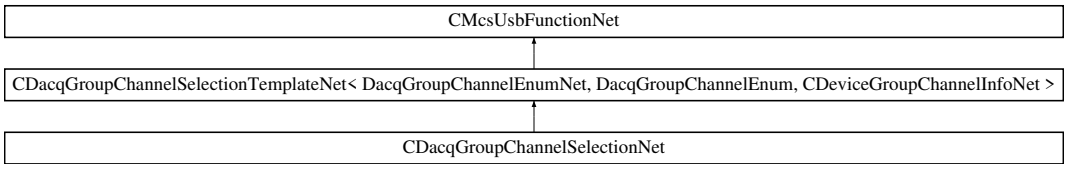
Additional Inherited Members

11.12.1 Constructor & Destructor Documentation

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

11.13 CDacqGroupChannelSelectionNet Class Reference

Inheritance diagram for CDacqGroupChannelSelectionNet:



Public Member Functions

- CDacqGroupChannelSelectionNet (CMcsUsbNet^ mcsusb)

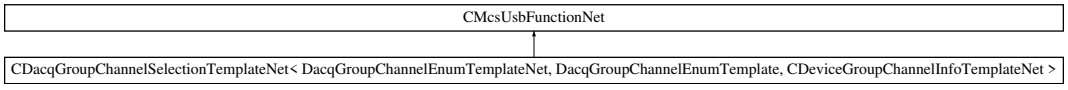
Additional Inherited Members

11.13.1 Constructor & Destructor Documentation

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

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

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



Public Member Functions

- CDacqGroupChannelSelectionTemplateNet (CMcsUsbNet^ 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

11.14.1 Constructor & Destructor Documentation

11.14.1.1 CDacqGroupChannelSelectionTemplateNet()

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

11.14.2 Member Function Documentation

11.14.2.1 EnableChannelsInGroup() [1/2]

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

11.14.2.2 EnableChannelsInGroup() [2/2]

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

11.14.2.3 GetDeviceGroupChannelInfos() [1/2]

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

11.14.2.4 GetDeviceGroupChannelInfos() [2/2]

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

11.14.2.5 GetEnabledChannelsInGroup() [1/2]

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

11.14.2.6 GetEnabledChannelsInGroup() [2/2]

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

11.14.2.7 GetGroupID() [1/2]

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

11.14.2.8 GetGroupID() [2/2]

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

11.14.2.9 GetGroupNumberOfChannels() [1/2]

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

11.14.2.10 GetGroupNumberOfChannels() [2/2]

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

11.14.2.11 GetGroupSampleSize() [1/2]

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

11.14.2.12 GetGroupSampleSize() [2/2]

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

11.14.2.13 GetGroupType() [1/2]

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

11.14.2.14 GetGroupType() [2/2]

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

11.14.2.15 GetNumberOfSupportedGroups() [1/2]

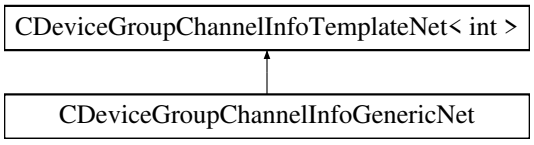
```
uint32_t GetNumberOfSupportedGroups ( )
```

11.14.2.16 GetNumberOfSupportedGroups() [2/2]

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

11.15 CDeviceGroupChannelInfoGenericNet Class Reference

Inheritance diagram for CDeviceGroupChannelInfoGenericNet:



Public Member Functions

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

Additional Inherited Members

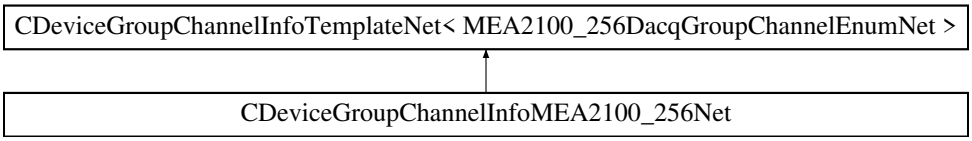
11.15.1 Constructor & Destructor Documentation

11.15.1.1 CDeviceGroupChannelInfoGenericNet()

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

11.16 CDeviceGroupChannelInfoMEA2100_256Net Class Reference

Inheritance diagram for CDeviceGroupChannelInfoMEA2100_256Net:



Public Member Functions

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

Additional Inherited Members

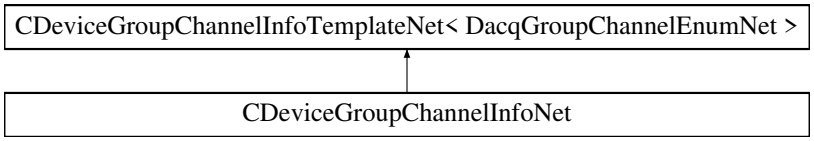
11.16.1 Constructor & Destructor Documentation

11.16.1.1 CDeviceGroupChannelInfoMEA2100_256Net()

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

11.17 CDeviceGroupChannelInfoNet Class Reference

Inheritance diagram for CDeviceGroupChannelInfoNet:



Public Member Functions

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

Additional Inherited Members

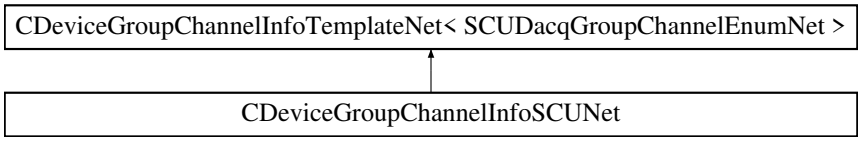
11.17.1 Constructor & Destructor Documentation

11.17.1.1 CDeviceGroupChannelInfoNet() `CDeviceGroupChannelInfoNet (`

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

11.18 CDeviceGroupChannelInfoSCUNet Class Reference

Inheritance diagram for CDeviceGroupChannelInfoSCUNet:



Public Member Functions

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

Additional Inherited Members

11.18.1 Constructor & Destructor Documentation

11.18.1.1 CDeviceGroupChannelInfoSCUNet() `CDeviceGroupChannelInfoSCUNet (`

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

11.19 CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumTemplateNet > Class Template Reference

Public Member Functions

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

Public Attributes

- DacqGroupChannelEnumTemplateNet GroupID
- int NumberOfChannels
- DacqMeaGroupTypeEnumNet GroupType

11.19.1 Constructor & Destructor Documentation

11.19.1.1 CDeviceGroupChannelInfoTemplateNet() `CDeviceGroupChannelInfoTemplateNet (`

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

11.19.2 Member Data Documentation

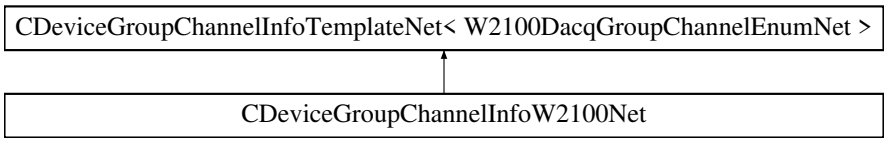
11.19.2.1 GroupID `DacqGroupChannelEnumTemplateNet GroupID`

11.19.2.2 GroupType `DacqMeaGroupTypeEnumNet GroupType`

11.19.2.3 NumberOfChannels `int NumberOfChannels`

11.20 CDeviceGroupChannelInfoW2100Net Class Reference

Inheritance diagram for CDeviceGroupChannelInfoW2100Net:



Public Member Functions

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

Additional Inherited Members

11.20.1 Constructor & Destructor Documentation

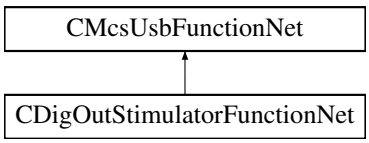
11.20.1.1 CDeviceGroupChannelInfoW2100Net() `CDeviceGroupChannelInfoW2100Net (`

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

11.21 CDigOutStimulatorFunctionNet Class Reference

CDigOutStimulatorFunctionNet is the class of the DigOut stimulator function class.

Inheritance diagram for CDigOutStimulatorFunctionNet:



Public Member Functions

- CDigOutStimulatorFunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ 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

11.21.1 Detailed Description

CDigOutStimulatorFunctionNet is the class of the DigOut stimulator function class.

11.21.2 Constructor & Destructor Documentation

11.21.2.1 CDigOutStimulatorFunctionNet() [1/2]

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

Initializes a new instance of the CDigOutStimulatorFunctionNet class.

11.21.2.2 CDigOutStimulatorFunctionNet() [2/2]

```
CDigOutStimulatorFunctionNet (
    CMcsUsbNet^ mcsusb )
```

11.21.2.3 ~CDigOutStimulatorFunctionNet()

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

11.21.2.4 "!CDigOutStimulatorFunctionNet()

```
!CDigOutStimulatorFunctionNet ( )
```

11.21.3 Member Function Documentation

11.21.3.1 ClearChannel()

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

clear stimulation pattern

Parameters

NrChannel	the channel to clear

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

get repeat whole stimulation pattern

Parameters

NrChannel	channel number

Returns

current value

11.21.3.3 GetNumberOfChannels() `int32_t GetNumberOfChannels ( )`

get the number of channels available on the device

Returns

the number of channels

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

queries start condition of digital out stimulator

Parameters

NrChannel	channel number

Returns

start condition (rising or falling edge)

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

queries stop condition of digital out stimulator

Parameters

NrChannel	channel number

Returns

stop condition (rising or falling edge)

11.21.3.6 PrepareChannelData()

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

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

Parameters

Amplitude	array of amplitudes
Duration	array of durations

Returns

11.21.3.7 SendChannelData()

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

send or append stimulation pattern

Parameters

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

11.21.3.8 SetGlobalRepeat()

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

set repeat whole stimulation pattern

Parameters

NrChannel	channel number
value	new value

11.21.3.9 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)

11.21.3.10 SetStopTriggerSlope() `void SetStopTriggerSlope (`

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

sets stop condition of digital out stimulator

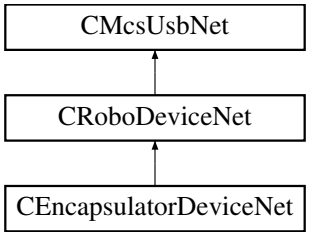
Parameters

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

11.22 CEncapsulatorDeviceNet Class Reference

CEncapsulatorDeviceNet is the to control the MCS HiClamp device

Inheritance diagram for CEncapsulatorDeviceNet:



Public Member Functions

- CEncapsulatorDeviceNet (void)
- CRoboFluidDeviceNet ^ GetRoboFluidDevice ()

Additional Inherited Members

11.22.1 Detailed Description

CEncapsulatorDeviceNet is the to control the MCS HiClamp device

11.22.2 Constructor & Destructor Documentation

11.22.2.1 CEncapsulatorDeviceNet() `CEncapsulatorDeviceNet ( void )`

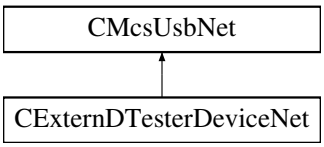
11.22.3 Member Function Documentation

11.22.3.1 GetRoboFluidDevice() `CRoboFluidDeviceNet ^ GetRoboFluidDevice ( )`

11.23 CExternDTesterDeviceNet Class Reference

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

Inheritance diagram for CExternDTesterDeviceNet:



Public Member Functions

- CExternDTesterDeviceNet ()

  Initializes a new instance of the CExternDTesterDeviceNet class.
- virtual ~CExternDTesterDeviceNet ()
- !CExternDTesterDeviceNet ()
- array< uint8_t > ^ Read (int configString_Length)

  Reads the config string from the device.
- String ^ Read2 ()

  Reads the config string from the device.
- void Write (array< uint8_t >^ configString)

  Reads the config string from the device.
- void Write2 (String^ configString)

  Reads the config string from the device.

Additional Inherited Members

11.23.1 Detailed Description

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

11.23.2 Constructor & Destructor Documentation

11.23.2.1 CExternDTesterDeviceNet() `CExternDTesterDeviceNet ( )`

Initializes a new instance of the CExternDTesterDeviceNet class.

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

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

11.23.3 Member Function Documentation

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

Reads the config string from the device.

Parameters

configString_Length	The maximal length of configString.

Returns

The config string.

11.23.3.2 Read2() `String ^ Read2 ( )`

Reads the config string from the device.

Returns

The config string.

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

Reads the config string from the device.

Parameters

configString	The config string.

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

Reads the config string from the device.

Parameters

configString	The config string.

11.24 CFilterCoefficientsNet Class Reference

Classes

- struct s_FilterAttributesNet

Public Member Functions

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

Properties

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

11.24.1 Constructor & Destructor Documentation

11.24.1.1 CFilterCoefficientsNet() [1/4] `CFilterCoefficientsNet ( )`

11.24.1.2 CFilterCoefficientsNet() [2/4] CFilterCoefficientsNet (

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

11.24.1.3 CFilterCoefficientsNet() [3/4] CFilterCoefficientsNet (

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

11.24.1.4 CFilterCoefficientsNet() [4/4] CFilterCoefficientsNet (

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

11.24.1.5 ~CFilterCoefficientsNet() ~CFilterCoefficientsNet ( )

11.24.2 Member Function Documentation

11.24.2.1 GetUintA() uint64_t GetUintA (

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

11.24.2.2 GetUintB() uint64_t GetUintB (

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

11.24.2.3 IsEqual() [1/2] bool IsEqual (

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

11.24.2.4 IsEqual() [2/2]

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

11.24.3 Property Documentation

11.24.3.1 A

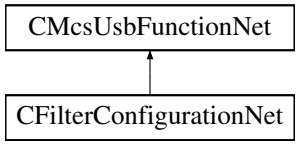
```
array< double>^ A [get]
```

11.24.3.2 B

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

11.25 CFilterConfigurationNet Class Reference

Inheritance diagram for CFilterConfigurationNet:



Public Member Functions

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

Additional Inherited Members

11.25.1 Constructor & Destructor Documentation

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

11.25.2 Member Function Documentation

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

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

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

11.25.2.4 GetHighpassFilterEnable() bool GetHighpassFilterEnable ( )

11.25.2.5 ResetHighpassFilter() void ResetHighpassFilter ( )

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

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

11.25.2.8 SetFilterParameterPermanent()

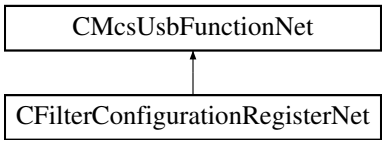
```
void SetFilterParameterPermanent (
    DacqGroupChannelEnumNet GroupID,
    uint32_t FilterNumber )
```

11.25.2.9 SetHighpassFilterEnable()

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

11.26 CFilterConfigurationRegisterNet Class Reference

Inheritance diagram for CFilterConfigurationRegisterNet:



Public Member Functions

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

Additional Inherited Members

11.26.1 Constructor & Destructor Documentation

11.26.1.1 CFilterConfigurationRegisterNet()

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

11.26.2 Member Function Documentation

11.26.2.1 EraseFilterParameterPermanent() [1/2]

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

11.26.2.2 EraseFilterParameterPermanent() [2/2]

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

11.26.2.3 SetFilterParameter() [1/2]

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

11.26.2.4 SetFilterParameter() [2/2]

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

11.26.2.5 SetFilterParameterPermanent() [1/2]

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

11.26.2.6 SetFilterParameterPermanent() [2/2]

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

11.27 CFilterPropertyNet Class Reference

Public Member Functions

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

Properties

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

11.27.1 Constructor & Destructor Documentation

11.27.1.1 CFilterPropertyNet()

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

11.27.1.2 ~CFilterPropertyNet()

```
~CFilterPropertyNet ( )
```

11.27.2 Member Function Documentation

11.27.2.1 ToString()

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

11.27.3 Property Documentation

11.27.3.1 CornerFrequency double CornerFrequency [get]

11.27.3.2 CornerFrequencymHz uint32_t CornerFrequencymHz [get]

11.27.3.3 FilterActive bool FilterActive [get]

11.27.3.4 FilterBand FilterBandEnumNet FilterBand [get]

11.27.3.5 FilterFamily FilterFamilyEnumNet FilterFamily [get]

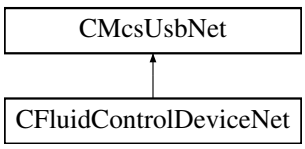
11.27.3.6 FilterType FilterTypeEnumNet FilterType [get]

11.27.3.7 Order uint32_t Order [get]

11.28 CFluidControlDeviceNet Class Reference

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

Inheritance diagram for CFluidControlDeviceNet:



Public Member Functions

- CFluidControlDeviceNet ()

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

  Default destructor.
- void SetValve (unsigned int value)

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

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

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

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

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

  Sets the 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

11.28.1 Detailed Description

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

11.28.2 Constructor & Destructor Documentation

11.28.2.1 CFluidControlDeviceNet() `CFluidControlDeviceNet ( )`

Initialize a new instance of the CFluidControlDeviceNet class.

11.28.2.2 ~CFluidControlDeviceNet() `~CFluidControlDeviceNet ( )`

Default destructor.

11.28.3 Member Function Documentation

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

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

Parameters

channel	Thermocouple channel number.

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

Reads an ADC Value.

Parameters

channel	The ADC channel number to query.

Returns

The current ADC value.

11.28.3.3 GetDigin() `unsigned int GetDigin ( )`

Reads the digital input.

Returns

The bit pattern of the state of the digital inputs.

11.28.3.4 GetDigout() `unsigned int GetDigout ( )`

Gets the state of the digital output.

Returns

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

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

Gets the state of the PWM output.

Returns

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

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

Reads the reference temperature for the Thermocouple.

Parameters

channel	Thermocouple channel number.

Returns

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

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

Gets the state of a valve.

Parameters

valve	number of valve

Returns

state of the valve

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

Gets the calibration constant for the Thermocouple ADC.

Parameters

channel	Thermocouple channel number.

Returns

The calibration constant for the Thermocouple ADC.

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

Reads the proportional constant for the Thermocouple.

Parameters

channel	Thermocouple channel number.

Returns

The proportional constant in Nanovolt per Kelvin.

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

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

Parameters

channel	Thermocouple channel number.

Returns

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

11.28.3.11 GetValve() `unsigned int GetValve ( )`

Gets the state of the valves.

Returns

The current state of the valves as a bit pattern.

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

Define the pattern on the Digital Output.

Parameters

value	bit pattern on the digital output.

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

Sets the duty cycle of the PWM output.

Parameters

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

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

Opens or Closes a valve.

Parameters

valve	number of valve to be changed.

Parameters

onoff	open or close the valve.

11.28.3.15 SetThermocoupleNanovoltPerKelvin()

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

Sets the proportinal constant for the Thermocouple.

Parameters

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

11.28.3.16 SetValve()

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

Open or Close valves.

Parameters

value	bit pattern of valves which should be open.

11.28.4 Property Documentation

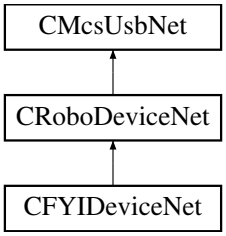
11.28.4.1 McsBus_VoltageMode

```
CMcsBus_VoltageModeNet^ McsBus_VoltageMode [get]
```

11.29 CFYIDeviceNet Class Reference

CFYIDeviceNet is the class to control the MCS FYI device

Inheritance diagram for CFYIDeviceNet:



Public Member Functions

- CFYIDeviceNet (void)

Properties

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

Additional Inherited Members

11.29.1 Detailed Description

CFYIDeviceNet is the class to control the MCS FYI device

11.29.2 Constructor & Destructor Documentation

11.29.2.1 CFYIDeviceNet() CFYIDeviceNet (void)

11.29.3 Property Documentation

11.29.3.1 FYIProgram CRobo_FYIProgram_FunctionNet^ FYIProgram [get]

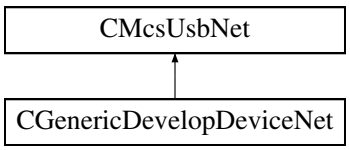
11.29.3.2 FYITemp CRobo_FYITemp_FunctionNet^ FYITemp [get]

11.29.3.3 Sensor CMcsBus_SensorNet^ Sensor [get]

11.30 CGenericDevelopDeviceNet Class Reference

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

Inheritance diagram for CGenericDevelopDeviceNet:



Public Member Functions

- CGenericDevelopDeviceNet (void)

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

  Sets .

Parameters

value	*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)
- 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

11.30.1 Detailed Description

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

11.30.2 Constructor & Destructor Documentation

11.30.2.1 CGenericDevelopDeviceNet() CGenericDevelopDeviceNet (
void)

Initialize a new instance of the CGenericDevelopDeviceNet class.

11.30.2.2 ~CGenericDevelopDeviceNet() ~CGenericDevelopDeviceNet (
void)

11.30.3 Member Function Documentation

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

Close a Pipe to an USB Endpoint.

Parameters

pipeHandle	The endpoint handle.

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

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

Gets an array of type char from the device.

Parameters

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.

11.30.3.4 GetIntBuffer()

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

Gets an array of type int from the device.

Parameters

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.

11.30.3.5 GetShortBuffer()

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

Gets an array of type short from the device.

Parameters

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.

11.30.3.6 GetUByteBuffer()

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

Gets an array of type unsigned char from the device.

Parameters

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.

11.30.3.7 GetUIntBuffer()

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

Gets an array of type unsigned int from the device.

Parameters

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.

11.30.3.8 GetUShortBuffer() `array<unsigned short> ^ GetUShortBuffer (`

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

Gets an array of type unsigned short from the device.

Parameters

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.

11.30.3.9 OpenPipe() `IntPtr OpenPipe (`

```
uint8_t endpointAddress )
```

Open a Pipe to an USB Endpoint.

Parameters

endpointAddress	The Endpoint Number.

Returns

A handle to the endpoint.

11.30.3.10 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.

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

Reset a Pipe to an USB Endpoint.

Parameters

pipeHandle	The endpoint handle.

11.30.3.12 SetBuffer()

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

11.30.3.13 SetByteBuffer()

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

Sends an array of type char to the device.

Parameters

value	The value of the request.

Parameters

Index	The index of the request.

Parameters

buffer	The buffer to send.

11.30.3.14 SetIntBuffer()

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

Sends an array of type int to the device.

Parameters

value	The value of the request.

Parameters

index	The index of the request.

Parameters

buffer	The buffer to send.

11.30.3.15 SetShortBuffer() `void SetShortBuffer (`

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

Sends an array of type short to the device.

Parameters

value	The value of the request.

Parameters

index	The index of the request.

Parameters

buffer	The buffer to send.

11.30.3.16 SetUByteBuffer() `void SetUByteBuffer (`

```
uint16_t value,
```

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

Sends an array of type unsigned char to the device.

Parameters

value	The value of the request.

Parameters

index	The index of the request.

Parameters

buffer	The buffer to send.

11.30.3.17 SetUIntBuffer() `void SetUIntBuffer (`

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

Sends an array of unsigned int to the device.

Parameters

value	The value of the request.

Parameters

index	The index of the request.

Parameters

buffer	The buffer to send.

11.30.3.18 SetUShortBuffer() `void SetUShortBuffer (`

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

Sends an array of type unsigned short to the device.

Parameters

value	The value of the request.

Parameters

index	The index of the request.

Parameters

buffer	The buffer to send.

11.30.3.19 SetValue() `void SetValue (`

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

Sets .

Parameters

value	The value of the request.

Parameters

index	The index of the request.

11.30.3.20 VendorInRequest()

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

11.30.3.21 VendorOutRequest()

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

11.30.3.22 WritePipe()

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

Write data to an USB Endpoint.

Parameters

pipeHandle	The endpoint handle.

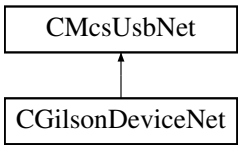
Parameters

buffer	An array of data to write.

11.31 CGilsonDeviceNet Class Reference

CGilsonDeviceNet is the class to control a Gilson device.

Inheritance diagram for CGilsonDeviceNet:



Public Member Functions

- CGilsonDeviceNet (void)

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

Protected Attributes

- CGilsonDevice * m_pGilsonDevice

Additional Inherited Members

11.31.1 Detailed Description

CGilsonDeviceNet is the class to control a Gilson device.

11.31.2 Constructor & Destructor Documentation

11.31.2.1 CGilsonDeviceNet() `CGilsonDeviceNet ( void )`

Initialize a new instance of the CGilsonDeviceNet class.

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

11.31.3 Member Function Documentation

11.31.3.1 ConnectSlave() `void ConnectSlave (`

`byte ID )`

11.31.3.2 GetLastAnswer() `String ^ GetLastAnswer ( )`

11.31.3.3 SendBuffered() `void SendBuffered (`

`String^ command )`

11.31.3.4 SendImmediate() `void SendImmediate (`

`wchar_t command )`

11.31.3.5 SendImmediateGetResponse() `String ^ SendImmediateGetResponse (`

`wchar_t command )`

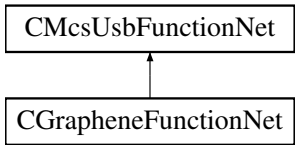
11.31.4 Member Data Documentation

11.31.4.1 m_pGilsonDevice `CGilsonDevice* m_pGilsonDevice [protected]`

11.32 CGrapheneFunctionNet Class Reference

CGrapheneFunctionNet is the class to control 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 SetVgs (int32_t Vgs)

  Sets Vgs
- void SetVgs (uint32_t Headstage, int32_t Vgs)

  Sets Vgs
- void SetVds (int32_t Vds)

  Sets Vds
- void SetVds (uint32_t Headstage, int32_t Vds)

  Sets Vds
- 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 GetVgs ()

  Gets Vgs
- int32_t GetVgs (uint32_t Headstage)

  Gets Vgs
- int32_t GetVds ()

  Gets Vds
- int32_t GetVds (uint32_t Headstage)

  Gets Vds

Additional Inherited Members

11.32.1 Detailed Description

CGrapheneFunctionNet is the class to control Graphene device functions

11.32.2 Constructor & Destructor Documentation

11.32.2.1 CGrapheneFunctionNet() [1/2]

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

Initializes a new instance of the CGrapheneFunctionNet class.

11.32.2.2 CGrapheneFunctionNet() [2/2] CGrapheneFunctionNet (CMcsUsbNet^ *mcsusb*)

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

11.32.2.4 "!CGrapheneFunctionNet() !CGrapheneFunctionNet ( )

11.32.3 Member Function Documentation

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

Gets the resistance of the current to voltage converter

Returns

The resistance in Ohm

11.32.3.2 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

11.32.3.3 GetDACOffset() [1/2] void GetDACOffset ([System::Runtime::InteropServices::Out] int16_t% *offset_vd*, [System::Runtime::InteropServices::Out] int16_t% *offset_vs*)

Gets the DAC offset

Parameters

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

11.32.3.4 GetDACOffset() [2/2]

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

Gets the DAC offset

Parameters

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

11.32.3.5 GetVds() [1/2]

```
int32_t GetVds ( )
```

Gets Vds

Returns

Vds in µV

11.32.3.6 GetVds() [2/2]

```
int32_t GetVds (
    uint32_t Headstage )
```

Gets Vds

Parameters

Headstage	The headstage to query.

Returns

Vds in µV

11.32.3.7 GetVdVs() [1/2]

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

Gets Vd and Vs

Parameters

Vd	Vd in µV
Vs	Vs in µV

11.32.3.8 GetVdVs() [2/2]

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

Gets Vd and Vs

Parameters

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

11.32.3.9 GetVdVsDAC() [1/2]

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

Gets Vd and Vs

Parameters

Vd	Vd in DAC Units
Vs	Vs in DAC Units

11.32.3.10 GetVdVsDAC() [2/2]

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

Gets Vd and Vs

Parameters

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

11.32.3.11 GetVgs() [1/2] int32_t GetVgs ( )

Gets Vgs

Returns

Vgs in µV

11.32.3.12 GetVgs() [2/2] int32_t GetVgs (uint32_t *Headstage*)

Gets Vgs

Parameters

Headstage	The headstage to query.

Returns

Vgs in µV

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

Gets the voltage range

Returns

The voltage range in mV

11.32.3.14 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

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

Gets the reached voltage

Returns

the reached voltage

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

Gets the reached voltage

Parameters

Headstage	The headstage to query.

Returns

The reached voltage

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

Gets the voltage resolution

Returns

The voltage resolution in µV/digit

11.32.3.18 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

11.32.3.19 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

11.32.3.20 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

11.32.3.21 SetVds() [1/2] void SetVds (

```
int32_t Vds )
```

Sets Vds

Parameters

Vds	Vds in µV

11.32.3.22 SetVds() [2/2] void SetVds (

```
uint32_t Headstage,
int32_t Vds )
```

Sets Vds

Parameters

Headstage	The headstage to query.
Vds	Vds in µV

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

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

Sets Vd and Vs

Parameters

Vd	Vd in µV
Vs	Vs in µV

11.32.3.24 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

11.32.3.25 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

11.32.3.26 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

11.32.3.27 SetVgs() [1/2] void SetVgs (

```
int32_t Vgs )
```

Sets Vgs

Parameters

Vgs	Vgs in µV

11.32.3.28 SetVgs() [2/2] void SetVgs (

```
uint32_t Headstage,
int32_t Vgs )
```

Sets Vgs

Parameters

Headstage	The headstage to query.
Vgs	Vgs in µV

11.32.3.29 SetVoltageRange() [1/2] void SetVoltageRange (

```
int32_t range )
```

Sets the voltage range

Parameters

range	The voltage range in mV

11.32.3.30 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

11.32.3.31 SetVoltageResolution() [1/2] void SetVoltageResolution (

```
int32_t resolution )
```

Sets the voltage resolution

Parameters

resolution	The voltage resolution in µV/digit

11.32.3.32 SetVoltageResolution() [2/2] void SetVoltageResolution (

```
uint32_t Headstage,
int32_t resolution )
```

Sets the voltage resolution

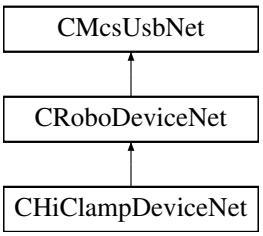
Parameters

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

11.33 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

11.33.1 Detailed Description

CHiClampDeviceNet is the to control the MCS HiClamp device

11.33.2 Constructor & Destructor Documentation

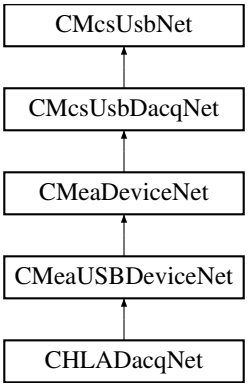
11.33.2.1 CHiClampDeviceNet() CHiClampDeviceNet (void)

11.33.3 Property Documentation

11.33.3.1 RoboDacq CRoboDacqNet^ RoboDacq [get]

11.34 CHLADacqNet Class Reference

Inheritance diagram for CHLADacqNet:



Public Member Functions

- CHLADacqNet (void)

Additional Inherited Members

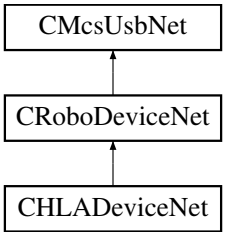
11.34.1 Constructor & Destructor Documentation

11.34.1.1 CHLADacqNet() `CHLADacqNet ( void )`

11.35 CHLADeviceNet Class Reference

CHLADeviceNet is the to control the MCS HLA device

Inheritance diagram for CHLADeviceNet:



Public Member Functions

- CHLADeviceNet (void)

Properties

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

Additional Inherited Members

11.35.1 Detailed Description

CHLADeviceNet is the to control the MCS HLA device

11.35.2 Constructor & Destructor Documentation

11.35.2.1 CHLADeviceNet() CHLADeviceNet (
void)

11.35.3 Property Documentation

11.35.3.1 HLADacq CHLADacqNet^ HLADacq [get]

11.35.3.2 SerialPort CSerialPortNet^ SerialPort [get]

11.36 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.

11.36.1 Detailed Description

Class to provide hardware information about the device.

11.36.2 Constructor & Destructor Documentation

11.36.2.1 CHWInfo() `CHWInfo (`

`CMcsUsbDacqNet^ device )`

11.36.3 Member Function Documentation

11.36.3.1 GetAvailableSampleRates() `virtual uint32_t GetAvailableSampleRates (`

`[System::Runtime::InteropServices::Out] System::Collections::Generic::List< int32_t >^% sampleRates ) [virtual]`

11.36.3.2 GetAvailableVoltageRangesInMicroVolt() `virtual System::Collections::Generic::List<int32_t> ^ GetAvailableVoltageRangesInMicroVolt (`

`int milliGain ) [virtual]`

Gets a List of voltage ranges the device supports.

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

Parameters

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

Returns

List of voltage ranges in µV.

11.36.3.3 GetAvailableVoltageRangesInMicroVoltAndStringsInMilliVolt() `virtual System::Collections::Generic::List<CVoltageRangeInfoNet^> ^ GetAvailableVoltageRangesInMicroVoltAndStringsInMilliVolt (`

`int milliGain ) [virtual]`

Gets a List of voltage ranges the device supports.

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

Parameters

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

Returns

List of voltage ranges in µV.

11.36.3.4 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.

11.36.3.5 GetNumberOfHWDigitalChannels()

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

Get the number of digital channels the device supports.

Parameters

numberOfChannels	Number of digital channels the device supports.

Returns

Error Status. 0 on success.

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

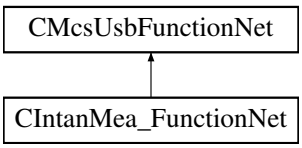
Query if the digital channel replaces an analog channel when enabled (e.g. on MC_Card) or adds a channel link on USB devices.

Returns

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

11.37 CIntanMea_FunctionNet Class Reference

Inheritance diagram for CIntanMea_FunctionNet:



Public Member Functions

- CIntanMea_FunctionNet (CMcsUsbNet^ 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

11.37.1 Constructor & Destructor Documentation

11.37.1.1 CIntanMea_FunctionNet() [1/2]

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

11.37.1.2 CIntanMea_FunctionNet() [2/2] CIntanMea_FunctionNet (CMcsUsbNet^ *mcsusb*)

11.37.2 Member Function Documentation

11.37.2.1 AmplifierSettle() void AmplifierSettle ( )

11.37.2.2 BeginImpedanceCheck() void BeginImpedanceCheck (array< int >^ *config_values*)

11.37.2.3 GetDSPHighPassByIndex() int64_t GetDSPHighPassByIndex (unsigned short *index*)

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

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

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

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

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

11.37.2.9 SetDiagnosticMode()

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

11.37.2.10 SetDSPHighPassByIndex()

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

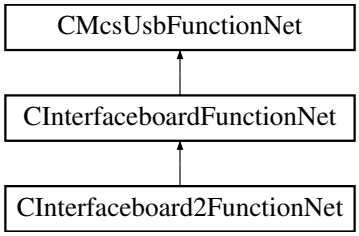
11.37.2.11 SetIntanRegister()

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

11.38 CInterfaceboard2FunctionNet Class Reference

CInterfaceboard2FunctionNet is the class to control the Interfaceboard

Inheritance diagram for CInterfaceboard2FunctionNet:



Public Member Functions

- CInterfaceboard2FunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ 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

11.38.1 Detailed Description

CInterfaceboard2FunctionNet is the class to control the Interfaceboard

11.38.2 Constructor & Destructor Documentation

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

Initializes a new instance of the CInterfaceboard2FunctionNet class.

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

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

11.38.2.4 "!CInterfaceboard2FunctionNet() !CInterfaceboard2FunctionNet ( )

11.38.3 Member Function Documentation

11.38.3.1 GetIoVoltage() IoVoltageEnumNet GetIoVoltage ( )

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

Returns

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

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

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

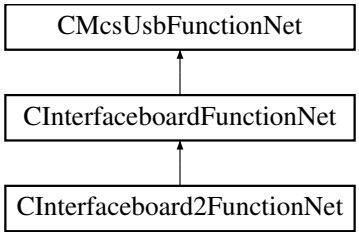
Parameters

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

11.39 CInterfaceboardFunctionNet Class Reference

CInterfaceboardFunctionNet is the class to control the Interfaceboard

Inheritance diagram for CInterfaceboardFunctionNet:



Public Member Functions

- CInterfaceboardFunctionNet (CMcsUsbNet^ 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

11.39.1 Detailed Description

CInterfaceboardFunctionNet is the class to control the Interfaceboard

11.39.2 Constructor & Destructor Documentation

11.39.2.1 CInterfaceboardFunctionNet() [1/2]

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

Initializes a new instance of the CInterfaceboardFunctionNet class.

11.39.2.2 CInterfaceboardFunctionNet() [2/2] CInterfaceboardFunctionNet (CMcsUsbNet^ *mcsusb*)

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

11.39.2.4 "!CInterfaceboardFunctionNet() !CInterfaceboardFunctionNet ( )

11.39.3 Member Function Documentation

11.39.3.1 GetCardinalDacqSamplerate() uint32_t GetCardinalDacqSamplerate ( )

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

Returns

The samplerate in Hz.

11.39.3.2 GetCardinalStgOutputrate() uint32_t GetCardinalStgOutputrate ( )

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

Returns

The output rate in Hz.

11.39.3.3 SetCardinalDacqSamplerate() void SetCardinalDacqSamplerate (uint32_t *samplerate*)

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

Parameters

samplerate	The samplerate in Hz.

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

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

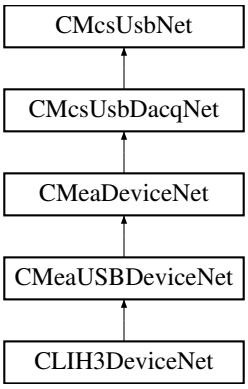
Parameters

outputrate	The output rate in Hz.

11.40 CLIH3DeviceNet Class Reference

CLIH3DeviceNet is the class to access the HEKA LIH3 device.

Inheritance diagram for CLIH3DeviceNet:



Public Member Functions

- CLIH3DeviceNet ()

  Initializes a new instance of the CLIH3DeviceNet class.
- virtual ~CLIH3DeviceNet ()
- !CLIH3DeviceNet ()
- void DummyCommand (uint32_t dummyParameter)

  Dummy command to show how to use the DLL.
- void SetEEpromPage (uint32_t EEpromStartAddress, array< int8_t >^ EEpromData, LIH30_EPC10_Bus_EnumNet epc10bus)

  Writes into EEprom on the EPC10 EEPROM
- array< int8_t > ^ GetEEpromPage (uint32_t EEpromStartAddress, int EEpromData_Length, LIH30_EPC10_Bus_EnumNet epc10bus)

  Reads the requested amount of EEprom byte from the EPC10 EEPROM
- void SetSampleInterval (uint32_t SampleInterval)

  Sets the Sample Interval for the DACQ and Stimulation
- uint32_t GetSampleInterval ()

  Gets the Sample Interval for the DACQ and Stimulation
- void SetAdcOffset (LIH30_ADC_Channel_EnumNet AdcChannel, int32_t Offset)

  Sets the ADC offset of the DACQ for a single channel
- int32_t GetAdcOffset (LIH30_ADC_Channel_EnumNet AdcChannel)

  Gets the ADC offset of the DACQ for a single channel

- void SetAdcOffsetPermanent (LIH30_ADC_Channel_EnumNet AdcChannel)

  Writes the ADC offset of the DACQ for a single channel to permanent EEProm memory
- void ErasePermanentAdcOffset (LIH30_ADC_Channel_EnumNet AdcChannel)

  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

11.40.1 Detailed Description

CLIH3DeviceNet is the class to access the HEKA LIH3 device.

11.40.2 Constructor & Destructor Documentation

11.40.2.1 CLIH3DeviceNet() `CLIH3DeviceNet ( )`

Initializes a new instance of the CLIH3DeviceNet class.

11.40.2.2 ~CLIH3DeviceNet() `virtual ~CLIH3DeviceNet ( ) [virtual]`

11.40.2.3 "!CLIH3DeviceNet() `!CLIH3DeviceNet ( )`

11.40.3 Member Function Documentation

11.40.3.1 DummyCommand() `void DummyCommand ( uint32_t dummyParameter )`

Dummy command to show how to use the DLL.

Parameters

dummyParameter	parameter to send to the device

11.40.3.2 EnableUserTrigger() `void EnableUserTrigger ( bool enable )`

Enables the User Trigger

Parameters

enable	Enable

11.40.3.3 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

11.40.3.4 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

11.40.3.5 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

11.40.3.6 GetAudioOutDacParameter() `void GetAudioOutDacParameter (`

`[System::Runtime::InteropServices::Out] uint32_t%` *Frequency*,
`[System::Runtime::InteropServices::Out] uint32_t%` *Amplification*)

Gets the parameter of the audio DAC output.

Parameters

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

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

Gets the DAC Idle value

Parameters

DacChannel	The DAC channel

Returns

The idle value

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

Gets the offset of a DAC channel.

Parameters

DacChannel	The DAC channel

Returns

The offset for the given channel number

11.40.3.9 GetDacqRunStatus() `uint16_t GetDacqRunStatus ( )`

Gets the data acquisition running status

Returns

The status (1: running / 0: stopped)

11.40.3.10 GetDacUseIdleValue() `bool GetDacUseIdleValue ( uint32_t DacChannel )`

Gets if the DAC Idle value is used after stimulation

Parameters

DacChannel	The DAC channel

Returns

Use idle value

11.40.3.11 GetDigInState() `uint16_t GetDigInState ( )`

Reads from the LIH30 digital input

Returns

The bit mask defining the digital input state

11.40.3.12 GetEEpromPage() `array<int8_t> ^ GetEEpromPage ( uint32_t EEpromStartAddress, int EEpromData_Length, LIH30_EPC10_Bus_EnumNet epc10bus )`

Reads the requested amount of EEprom byte from the EPC10 EEPROM

Parameters

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

11.40.3.13 GetSampleInterval() `uint32_t GetSampleInterval ( )`

Gets the Sample Interval for the DACQ and Stimulation

Returns

Sample Interval configured on the device

11.40.3.14 IsUserTriggerEnabled() `bool IsUserTriggerEnabled ( )`

Is the User Trigger enabled

Returns

Enabled

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

Gets the clipping information

Parameters

epc10bus	The EPC10 bus

Returns

The clipping value

11.40.3.16 ReadUARTData() `String ^ ReadUARTData ( )`

Reads the config string from the device.

Returns

The config string.

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

Send command to the EPC10

Parameters

epc10bus	The EPC10 bus
Command	The command

11.40.3.18 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

11.40.3.19 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

11.40.3.20 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)

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

Sets the DAC Idle value

Parameters

DacChannel	The DAC channel
IdleValue	The idle value

11.40.3.22 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

11.40.3.23 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

11.40.3.24 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

11.40.3.25 SetDigOutState() void SetDigOutState (

uint16_t *DigOutState*)

Writes to the LIH30 digital output

Parameters

DigOutState	The bit mask defining the digital output state

11.40.3.26 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

11.40.3.27 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

11.40.3.28 WriteUARTData() `void WriteUARTData (`

```
String^ commandString )
```

Write the command string to the device.

Parameters

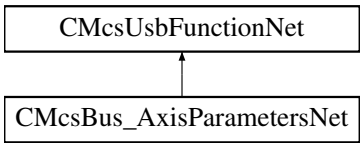
commandString	The config string.

11.40.4 Property Documentation

11.40.4.1 StimulusFunction `CStimulusFunctionNet^ StimulusFunction [get]`

11.41 CMcsBus_AxisParametersNet Class Reference

Inheritance diagram for CMcsBus_AxisParametersNet:



Public Member Functions

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

Additional Inherited Members

11.41.1 Constructor & Destructor Documentation

11.41.1.1 CMcsBus_AxisParametersNet()

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

11.41.1.2 ~CMcsBus_AxisParametersNet()

```
~CMcsBus_AxisParametersNet (
    void  )
```

11.41.2 Member Function Documentation

11.41.2.1 GetAxisParametersSignedEeprom()

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

11.41.2.2 GetAxisParametersUnsignedEeprom()

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

11.41.2.3 SetAxisParametersEeprom() [1/2]

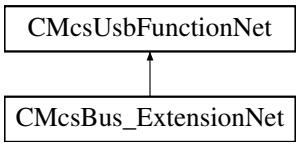
```
void SetAxisParametersEeprom (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis,
    unsigned short index,
    int parameter )
```

11.41.2.4 SetAxisParametersEeprom() [2/2]

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

11.42 CMcsBus_ExtensionNet Class Reference

Inheritance diagram for CMcsBus_ExtensionNet:



Public Member Functions

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

Additional Inherited Members

11.42.1 Constructor & Destructor Documentation

11.42.1.1 CMcsBus_ExtensionNet()

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

11.42.1.2 ~CMcsBus_ExtensionNet()

```
~CMcsBus_ExtensionNet (
    void  )
```

11.42.2 Member Function Documentation

11.42.2.1 GetLEDSwitch() `unsigned short GetLEDSwitch (`

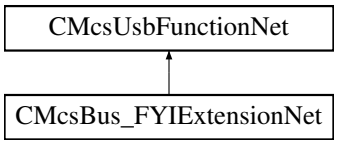
```
unsigned char busnumber,
unsigned char busaddress )
```

11.42.2.2 SetLEDSwitch() `void SetLEDSwitch (`

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

11.43 CMcsBus_FYIExtensionNet Class Reference

Inheritance diagram for CMcsBus_FYIExtensionNet:



Public Member Functions

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

Additional Inherited Members

11.43.1 Constructor & Destructor Documentation

11.43.1.1 CMcsBus_FYIExtensionNet() `CMcsBus_FYIExtensionNet (`

```
CMcsUsbNet^ device )
```

11.43.1.2 ~CMcsBus_FYIExtensionNet() ~CMcsBus_FYIExtensionNet (void)

11.43.2 Member Function Documentation

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

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

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

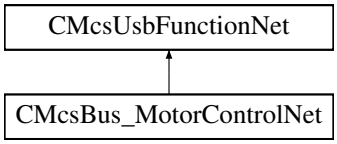
11.43.2.4 SetDIO() void SetDIO (unsigned char *busnumber*, unsigned char *busaddress*, unsigned short *io*)

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

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

11.44 CMcsBus_MotorControlNet Class Reference

Inheritance diagram for CMcsBus_MotorControlNet:



Public Member Functions

- CMcsBus_MotorControlNet (CMcsUsbNet^ 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

11.44.1 Constructor & Destructor Documentation

11.44.1.1 CMcsBus_MotorControlNet() CMcsBus_MotorControlNet (
CMcsUsbNet^ *device*)

11.44.1.2 ~CMcsBus_MotorControlNet() ~CMcsBus_MotorControlNet (
void)

11.44.2 Member Function Documentation

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

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

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

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

11.44.2.5 GetMCBreakCurrent()

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

11.44.2.6 GetMCBreakCurrentEeprom()

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

11.44.2.7 GetMCConfig()

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

11.44.2.8 GetMCConfigEeprom()

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

11.44.2.9 GetMCCurrent()

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

11.44.2.10 GetMCCurrentEeprom()

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

11.44.2.11 GetMCCurrentMode()

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

11.44.2.12 GetMCCurrentModeEeprom()

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

11.44.2.13 GetMCCurrentModeShortCommand()

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

11.44.2.14 GetMCCurrentPosition()

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

11.44.2.15 GetMCCurrentShortCommand()

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

11.44.2.16 GetMCCurrentSpeed()

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

11.44.2.17 GetMCMaxAcceleration()

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

11.44.2.18 GetMCMaxAccelerationEeprom()

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

11.44.2.19 GetMCMaxCurrent()

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

11.44.2.20 GetMCMaxCurrentEeprom()

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

11.44.2.21 GetMCMaxSpeed()

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

11.44.2.22 GetMCMaxSpeedEeprom()

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

11.44.2.23 GetMCMaxTravel()

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

11.44.2.24 GetMCMaxTravelEeprom()

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

11.44.2.25 GetMCMaxTravelShortCommand()

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

11.44.2.26 GetMCMovement()

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

11.44.2.27 GetMCNewPosition()

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

11.44.2.28 GetMCOutputOnOff()

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

11.44.2.29 GetMCPhase()

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

11.44.2.30 GetMCPhaseOffset()

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

11.44.2.31 GetMCReference()

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

11.44.2.32 GetMCReferenceCurrent()

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

11.44.2.33 GetMCReferenceCurrentEeprom()

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

11.44.2.34 GetMCRegulatorGain()

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

11.44.2.35 GetMCRegulatorGainEeprom()

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

11.44.2.36 GetMCScalingFactor()

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

11.44.2.37 GetMCScalingFactorEeprom()

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

11.44.2.38 GetMCSpeed()

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

11.44.2.39 GetMCSpeedEeprom()

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

11.44.2.40 GetMCSpeedShortCommand()

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

11.44.2.41 GetMCSpeedUnitEeprom()

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

11.44.2.42 GetMCStandbyCurrent()

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

11.44.2.43 GetMCStandbyCurrentEeprom()

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

11.44.2.44 GetMCStandbyTime()

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

11.44.2.45 GetMCStandbyTimeEeprom()

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

11.44.2.46 GetSubChannel()

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

11.44.2.47 SetMCAcceleration()

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

11.44.2.48 SetMCAccelerationEeprom()

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

11.44.2.49 SetMCAccelerationShortCommand()

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

11.44.2.50 SetMCAxisRevisionEeprom()

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

11.44.2.51 SetMCBreakCurrent()

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

11.44.2.52 SetMCBreakCurrentEeprom()

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

11.44.2.53 SetMCConfig()

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

11.44.2.54 SetMCConfigEeprom()

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

11.44.2.55 SetMCCurrent()

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

11.44.2.56 SetMCCurrentEeprom()

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

11.44.2.57 SetMCCurrentMode()

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

11.44.2.58 SetMCCurrentModeEeprom()

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

11.44.2.59 SetMCCurrentModeShortCommand()

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

11.44.2.60 SetMCCurrentPosition()

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

11.44.2.61 SetMCCurrentShortCommand()

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

11.44.2.62 SetMCMaxAcceleration()

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

11.44.2.63 SetMCMaxAccelerationEeprom()

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

11.44.2.64 SetMCMaxCurrent()

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

11.44.2.65 SetMCMaxCurrentEeprom()

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

11.44.2.66 SetMCMaxSpeed()

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

11.44.2.67 SetMCMaxSpeedEeprom()

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

11.44.2.68 SetMCMaxTravel()

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

11.44.2.69 SetMCMaxTravelEeprom()

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

11.44.2.70 SetMCMaxTravelShortCommand()

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

11.44.2.71 SetMCNewPosition()

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

11.44.2.72 SetMCOutputOnOff()

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

11.44.2.73 SetMCReference()

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

11.44.2.74 SetMCReferenceCurrent()

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

11.44.2.75 SetMCReferenceCurrentEeprom()

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

11.44.2.76 SetMCRegulatorGain()

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

11.44.2.77 SetMCRegulatorGainEeprom()

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

11.44.2.78 SetMCRotation()

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

11.44.2.79 SetMCScalingFactor()

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

11.44.2.80 SetMCScalingFactorEeprom()

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

11.44.2.81 SetMCSpeed()

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

11.44.2.82 SetMCSpeedEeprom()

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

11.44.2.83 SetMCSpeedShortCommand()

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

11.44.2.84 SetMCSpeedUnitEeprom()

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

11.44.2.85 SetMCStandbyCurrent()

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

11.44.2.86 SetMCStandbyCurrentEeprom()

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

11.44.2.87 SetMCStandbyTime()

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

11.44.2.88 SetMCStandbyTimeEeprom()

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

11.44.2.89 SetSubChannel()

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

11.44.2.90 StartMCMovement()

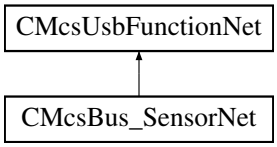
```
void StartMCMovement (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis )
```

11.44.2.91 StopMCMovement()

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

11.45 CMcsBus_SensorNet Class Reference

Inheritance diagram for CMcsBus_SensorNet:



Public Member Functions

- CMcsBus_SensorNet (CMcsUsbNet^ 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

11.45.1 Constructor & Destructor Documentation

11.45.1.1 CMcsBus_SensorNet() CMcsBus_SensorNet (CMcsUsbNet^ *device*)

11.45.1.2 ~CMcsBus_SensorNet() ~CMcsBus_SensorNet (void)

11.45.2 Member Function Documentation

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

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

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

11.45.2.4 CatchAmpGetAdcValueL()

```
int CatchAmpGetAdcValueL (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.45.2.5 CatchAmpGetDacAmplitude()

```
unsigned short CatchAmpGetDacAmplitude (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.45.2.6 CatchAmpGetDacEnable()

```
bool CatchAmpGetDacEnable (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.45.2.7 CatchAmpGetDacOffset()

```
short CatchAmpGetDacOffset (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.45.2.8 CatchAmpGetPwmEnable()

```
bool CatchAmpGetPwmEnable (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.45.2.9 CatchAmpSetDacAmplitude()

```
void CatchAmpSetDacAmplitude (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short dacAmplitude )
```

11.45.2.10 CatchAmpSetDacEnable()

```
void CatchAmpSetDacEnable (
    unsigned char busnumber,
    unsigned char busaddress,
    bool dacEnable )
```

11.45.2.11 CatchAmpSetDacOffset()

```
void CatchAmpSetDacOffset (
    unsigned char busnumber,
    unsigned char busaddress,
    short dacOffset )
```

11.45.2.12 CatchAmpSetPwmEnable()

```
void CatchAmpSetPwmEnable (
    unsigned char busnumber,
    unsigned char busaddress,
    bool pwmEnable )
```

11.45.2.13 Get2AnalogInput()

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

11.45.2.14 Get2DigitalInput()

```
unsigned short Get2DigitalInput (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.45.2.15 Get4ADC()

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

11.45.2.16 Get4ADCAverage()

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

11.45.2.17 Get4ADCCatchampAverageShift()

```
unsigned int Get4ADCCatchampAverageShift (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.45.2.18 Get4ADCMode()

```
PatchServAdcModeEnumNet Get4ADCMode (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.45.2.19 Get4DAC()

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

11.45.2.20 GetADCs()

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

11.45.2.21 GetADCsLoop()

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

11.45.2.22 GetBubbleStatus()

```
unsigned short GetBubbleStatus (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.45.2.23 GetDACs()

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

11.45.2.24 GetDetectionThreshold()

```
unsigned short GetDetectionThreshold (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.45.2.25 GetDetectorValue()

```
unsigned short GetDetectorValue (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.45.2.26 GetLatency()

```
unsigned short GetLatency (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.45.2.27 GetLatencyCounter()

```
unsigned short GetLatencyCounter (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.45.2.28 GetMinimalThreshold()

```
unsigned short GetMinimalThreshold (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.45.2.29 GetMovePump()

```
void GetMovePump (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short index,
    [System::Runtime::InteropServices::Out] unsigned short% speed,
    [System::Runtime::InteropServices::Out] int% position )
```

11.45.2.30 GetPiezoState()

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

11.45.2.31 GetPressure() [1/2]

```
array<int> ^ GetPressure (
    unsigned char busnumber,
    unsigned char busaddress,
    int n )
```

11.45.2.32 GetPressure() [2/2]

```
int GetPressure (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short index )
```

11.45.2.33 GetPressureOffset() [1/2]

```
array<unsigned short> ^ GetPressureOffset (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.45.2.34 GetPressureOffset() [2/2]

```
int GetPressureOffset (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short index )
```

11.45.2.35 GetRegulationTimeouts()

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

11.45.2.36 GetRegulatorFactor()

```
int GetRegulatorFactor (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short index )
```

11.45.2.37 GetRegulatorOnOff()

```
unsigned char GetRegulatorOnOff (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short index )
```

11.45.2.38 GetRegulatorStatus()

```
unsigned int GetRegulatorStatus (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short index )
```

11.45.2.39 GetRotatePump()

```
short GetRotatePump (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short index )
```

11.45.2.40 GetSamplePeriode()

```
unsigned short GetSamplePeriode (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.45.2.41 GetSollPressure()

```
int GetSollPressure (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short index )
```

11.45.2.42 GetSyncState()

```
unsigned short GetSyncState (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.45.2.43 Set4ADCCatchampAverageShift()

```
void Set4ADCCatchampAverageShift (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned int shift )
```

11.45.2.44 Set4ADCMode()

```
void Set4ADCMode (
    unsigned char busnumber,
    unsigned char busaddress,
    PatchServAdcModeEnumNet mode )
```

11.45.2.45 Set4DAC()

```
void Set4DAC (
    unsigned char busnumber,
    unsigned char busaddress,
    array< unsigned short >^ dac )
```

11.45.2.46 SetDACs()

```
void SetDACs (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short index,
    array< unsigned short >^ dac_times_voltages )
```

11.45.2.47 SetDetectionThreshold()

```
void SetDetectionThreshold (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short threshold )
```

11.45.2.48 SetLatency()

```
void SetLatency (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short latency )
```

11.45.2.49 SetMinimalThreshold()

```
void SetMinimalThreshold (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short threshold )
```

11.45.2.50 SetMovePump()

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

11.45.2.51 SetPiezoState()

```
void SetPiezoState (
    unsigned char busnumber,
    unsigned char busaddress,
    int state )
```

11.45.2.52 SetPressureOffset()

```
void SetPressureOffset (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short index )
```

11.45.2.53 SetRegulationTimeouts()

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

11.45.2.54 SetRegulatorFactor()

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

11.45.2.55 SetRegulatorOnOff()

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

11.45.2.56 SetRotatePump()

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

11.45.2.57 SetSamplePeriode()

```
void SetSamplePeriode (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short periode )
```

11.45.2.58 SetSollPressure()

```
void SetSollPressure (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short index,
    int pressure )
```

11.45.2.59 StartSync()

```
void StartSync (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.45.2.60 TactSwitchGetState()

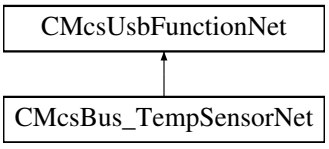
```
int TactSwitchGetState (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.45.2.61 TactSwitchSetDisplay()

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

11.46 CMcsBus_TempSensorNet Class Reference

Inheritance diagram for CMcsBus_TempSensorNet:



Public Member Functions

- CMcsBus_TempSensorNet (CMcsUsbNet^ device)
- ~CMcsBus_TempSensorNet (void)
- short GetTemperatur (unsigned char busnumber, unsigned char busaddress)
- short GetTemperatur (unsigned char busnumber, unsigned char busaddress, short index)
- void SetNanoVoltsPerKelvin (unsigned char busnumber, unsigned char busaddress, int nanovoltsperkelvin)
- int GetNanoVoltsPerKelvin (unsigned char busnumber, unsigned char busaddress)
- short GetThermoVoltage (unsigned char busnumber, unsigned char busaddress, short index)
- short GetThermoTemp (unsigned char busnumber, unsigned char busaddress, short index)
- void SetThermoOffset (unsigned char busnumber, unsigned char busaddress, short index, short offset)
- short GetThermoOffset (unsigned char busnumber, unsigned char busaddress, short index)

Additional Inherited Members

11.46.1 Constructor & Destructor Documentation

11.46.1.1 CMcsBus_TempSensorNet()

```
CMcsBus_TempSensorNet (
    CMcsUsbNet^ device )
```

11.46.1.2 ~CMcsBus_TempSensorNet()

```
~CMcsBus_TempSensorNet (
    void  )
```

11.46.2 Member Function Documentation

11.46.2.1 GetNanoVoltsPerKelvin()

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

11.46.2.2 GetTemperatur() [1/2]

```
short GetTemperatur (
    unsigned char busnumber,
    unsigned char busaddress )
```

11.46.2.3 GetTemperatur() [2/2]

```
short GetTemperatur (
    unsigned char busnumber,
    unsigned char busaddress,
    short index )
```

11.46.2.4 GetThermoOffset()

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

11.46.2.5 GetThermoTemp()

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

11.46.2.6 GetThermoVoltage()

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

11.46.2.7 SetNanoVoltsPerKelvin()

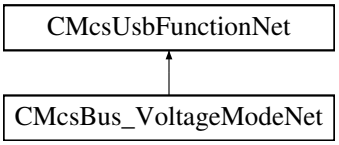
```
void SetNanoVoltsPerKelvin (
    unsigned char busnumber,
    unsigned char busaddress,
    int nanovoltsperkelvin )
```

11.46.2.8 SetThermoOffset()

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

11.47 CMcsBus_VoltageModeNet Class Reference

Inheritance diagram for CMcsBus_VoltageModeNet:



Public Member Functions

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

Additional Inherited Members

11.47.1 Constructor & Destructor Documentation

11.47.1.1 CMcsBus_VoltageModeNet() `CMcsBus_VoltageModeNet ( CMcsUsbNet^ device )`

11.47.1.2 ~CMcsBus_VoltageModeNet() ~CMcsBus_VoltageModeNet (void)

11.47.2 Member Function Documentation

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

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

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

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

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

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

11.47.2.7 GetVMMaxPositiveVoltage()

```
short GetVMMaxPositiveVoltage (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char channel )
```

11.47.2.8 GetVMMaxPositiveVoltageEeprom()

```
short GetVMMaxPositiveVoltageEeprom (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char channel )
```

11.47.2.9 GetVMOutputOnOff()

```
unsigned short GetVMOutputOnOff (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char channel )
```

11.47.2.10 GetVMVoltage()

```
short GetVMVoltage (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char channel )
```

11.47.2.11 SetVMMaxNegativeCurrent()

```
void SetVMMaxNegativeCurrent (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char channel,
    short current )
```

11.47.2.12 SetVMMaxNegativeCurrentEeprom()

```
void SetVMMaxNegativeCurrentEeprom (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char channel,
    short current )
```

11.47.2.13 SetVMMaxNegativeVoltage()

```
void SetVMMaxNegativeVoltage (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char channel,
    short voltage )
```

11.47.2.14 SetVMMaxNegativeVoltageEeprom()

```
void SetVMMaxNegativeVoltageEeprom (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char channel,
    short voltage )
```

11.47.2.15 SetVMMaxPositiveCurrent()

```
void SetVMMaxPositiveCurrent (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char channel,
    short current )
```

11.47.2.16 SetVMMaxPositiveCurrentEeprom()

```
void SetVMMaxPositiveCurrentEeprom (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char channel,
    short current )
```

11.47.2.17 SetVMMaxPositiveVoltage()

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

11.47.2.18 SetVMMaxPositiveVoltageEeprom()

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

11.47.2.19 SetVMOutputOnOff()

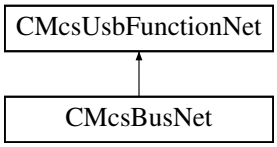
```
void SetVMOutputOnOff (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char channel,
    unsigned short status )
```

11.47.2.20 SetVMVoltage()

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

11.48 CMcsBusNet Class Reference

Inheritance diagram for CMcsBusNet:



Public Member Functions

- CMcsBusNet (CMcsUsbNet^ device)
- virtual ~CMcsBusNet (void)
- void SetCommand (unsigned char command, unsigned char busnumber, unsigned char busaddress, unsigned char axis, unsigned short value)
- void SetCommand (unsigned char command, unsigned char busnumber, unsigned char busaddress, unsigned char axis, short value)
- void SetCommand (unsigned char command, unsigned char busnumber, unsigned char busaddress, unsigned char axis, unsigned int value)
- void SetCommand (unsigned char command, unsigned char busnumber, unsigned char busaddress, unsigned char axis, int value)
- void GetCommand (unsigned char command, unsigned char busnumber, unsigned char busaddress, unsigned char axis, [System::Runtime::InteropServices::Out]unsigned short% value)
- void GetCommand (unsigned char command, unsigned char busnumber, unsigned char busaddress, unsigned char axis, [System::Runtime::InteropServices::Out]short% value)
- void GetCommand (unsigned char command, unsigned char busnumber, unsigned char busaddress, unsigned char axis, [System::Runtime::InteropServices::Out]unsigned int% value)
- void GetCommand (unsigned char command, unsigned char busnumber, unsigned char busaddress, unsigned char axis, [System::Runtime::InteropServices::Out]int% value)
- void SetBusAddressEeprom (unsigned char busnumber, unsigned char busaddress, unsigned short newaddress)
- unsigned short GetBusAddressEeprom (unsigned char busnumber, unsigned char busaddress)
- void SetBusAddress (unsigned char busnumber, unsigned char busaddress, unsigned short newaddress)
- unsigned short GetBusAddress (unsigned char busnumber, unsigned char busaddress)
- void CMcsBusNet::SetModeEeprom (unsigned char busnumber, unsigned char busaddress, unsigned short mode)
- unsigned short CMcsBusNet::GetModeEeprom (unsigned char busnumber, unsigned char busaddress)
- void CMcsBusNet::SetMode (unsigned char busnumber, unsigned char busaddress, unsigned short mode)
- unsigned short CMcsBusNet::GetMode (unsigned char busnumber, unsigned char busaddress)
- void SetHWRevisionEeprom (unsigned char busnumber, unsigned char busaddress, unsigned short revision)
- unsigned short GetHWRevisionEeprom (unsigned char busnumber, unsigned char busaddress)

Additional Inherited Members

11.48.1 Constructor & Destructor Documentation

11.48.1.1 CMcsBusNet() CMcsBusNet (CMcsUsbNet^ *device*)

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

11.48.2 Member Function Documentation

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

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

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

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

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

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

11.48.2.7 GetCommand() [1/4] void GetCommand (

```
unsigned char command,
unsigned char busnumber,
unsigned char busaddress,
unsigned char axis,
[System::Runtime::InteropServices::Out] int% value )
```

11.48.2.8 GetCommand() [2/4] void GetCommand (

```
unsigned char command,
unsigned char busnumber,
unsigned char busaddress,
unsigned char axis,
[System::Runtime::InteropServices::Out] short% value )
```

11.48.2.9 GetCommand() [3/4] void GetCommand (

```
unsigned char command,
unsigned char busnumber,
unsigned char busaddress,
unsigned char axis,
[System::Runtime::InteropServices::Out] unsigned int% value )
```

11.48.2.10 GetCommand() [4/4] void GetCommand (

```
unsigned char command,
unsigned char busnumber,
unsigned char busaddress,
unsigned char axis,
[System::Runtime::InteropServices::Out] unsigned short% value )
```

11.48.2.11 GetHWRevisionEeprom() unsigned short GetHWRevisionEeprom (

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

11.48.2.12 SetBusAddress() void SetBusAddress (

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

11.48.2.13 SetBusAddressEeprom()

```
void SetBusAddressEeprom (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short newaddress )
```

11.48.2.14 SetCommand() [1/4]

```
void SetCommand (
    unsigned char command,
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis,
    int value )
```

11.48.2.15 SetCommand() [2/4]

```
void SetCommand (
    unsigned char command,
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis,
    short value )
```

11.48.2.16 SetCommand() [3/4]

```
void SetCommand (
    unsigned char command,
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis,
    unsigned int value )
```

11.48.2.17 SetCommand() [4/4]

```
void SetCommand (
    unsigned char command,
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned char axis,
    unsigned short value )
```

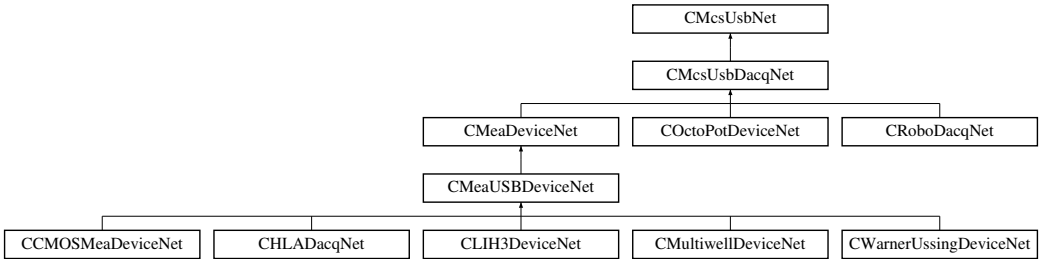
11.48.2.18 SetHWRevisionEeprom()

```
void SetHWRevisionEeprom (
    unsigned char busnumber,
    unsigned char busaddress,
    unsigned short revision )
```

11.49 CMcsUsbDacqNet Class Reference

Base class for data acquisition devices.

Inheritance diagram for CMcsUsbDacqNet:



Classes

- class CHWInfo

  Class to provide hardware information about the device.

Public Member Functions

- CMcsUsbDacqNet ()
- ~CMcsUsbDacqNet ()
- uint32_t GetErrorMessage ([System::Runtime::InteropServices::Out]String^% errorString, [System::Runtime::InteropServices::Out]int% info)
- virtual uint32_t GetVoltageRangeIndex (unsigned int virtualDevice)
- virtual void SetVoltageRangeByIndex (int32_t voltageRangeIndex, unsigned int virtualDevice)

  Sets the voltage range on devices which support multiple voltage ranges.
- virtual void SetVoltageRangeInMicroVolt (int32_t voltageRange, unsigned int virtualDevice)

  Sets the voltage range on devices which support multiple voltage ranges.
- virtual int32_t GetVoltageRangeInMicroVolt (unsigned int virtualDevice)

  Gets the currently selected voltage range on devices which support multiple voltage ranges.
- virtual int32_t GetVoltageRangeInMilliVolt ()

  Gets the currently selected voltage range on devices which support multiple voltage ranges.
- virtual void SetDataMode (DataModeEnumNet dataMode, unsigned int virtualDevice)

  Sets the data mode, can be 16, 24 or 32bit, all signed or unsigned on the MEA2100 device.
- virtual DataModeEnumNet GetDataMode (unsigned int virtualDevice)

  Gets the data mode, can be 16, 24 or 32bit, all signed or unsigned on the MEA2100 device.
- void SetDigitalSource (DigitalTargetEnumNet digitaltarget, int32_t NrChannel, DigitalSourceEnumNet source, int bitnumber_offset)

  Sets the function/source of an digital output bit.
- void SetDigitalSource (DigitalTargetEnumNet digitaltarget, int32_t NrChannel, W2100DigitalSourceEnumNet source, int bitnumber_offset)

  Sets the function/source of an digital output bit.
- void SetDigitalSource (DigitalTargetEnumNet digitaltarget, int32_t NrChannel, SCUDigitalSourceEnumNet source, int bitnumber_offset)

  Sets the function/source of an digital output bit.
- void SetDigitalSource (DigitalTargetEnumNet digitaltarget, int32_t NrChannel, MEA2100_256DigitalSourceEnumNet source, int bitnumber_offset)

  Sets the function/source of an digital output bit.

- template<typename digitalsourceenum >
  void SetDigitalSource (DigitalTargetEnumNet digitaltarget, int32_t NrChannel, DigitalSource< digitalsourceenum >^ source, int bitnumber_offset)

  Sets the function/source of an digital output bit.
- void GetDigitalSource (DigitalTargetEnumNet digitaltarget, int32_t NrChannel, [System::Runtime::InteropServices::Out]DigitalSourceEnumNet% source, [System::Runtime::InteropServices::Out]int% bitnumber_offset)

  Gets the function/source of an digital output bit.
- void GetDigitalSource (DigitalTargetEnumNet digitaltarget, int32_t NrChannel, [System::Runtime::InteropServices::Out]W2100DigitalSourceEnumNet% source, [System::Runtime::InteropServices::Out]int% bitnumber_offset)

  Gets the function/source of an digital output bit.
- void GetDigitalSource (DigitalTargetEnumNet digitaltarget, int32_t NrChannel, [System::Runtime::InteropServices::Out]SCUDigitalSourceEnumNet% source, [System::Runtime::InteropServices::Out]int% bitnumber_offset)

  Gets the function/source of an digital output bit.
- void GetDigitalSource (DigitalTargetEnumNet digitaltarget, int32_t NrChannel, [System::Runtime::InteropServices::Out]MEA2100_256DigitalSourceEnumNet% source, [System::Runtime::InteropServices::Out]int% bitnumber_offset)

  Gets the function/source of an digital output bit.
- template<typename digitalsourceenum >
  void GetDigitalSource (DigitalTargetEnumNet digitaltarget, int32_t NrChannel, [System::Runtime::InteropServices::Out]DigitalSource< digitalsourceenum >^% source, [System::Runtime::InteropServices::Out]int% bitnumber_offset)

  Gets the function/source of an digital output bit.
- virtual AdapterTypeEnumNet GetAdapterType ()

  Gets the adapter which is connected to the MEA2100 device.
- virtual MeaLayoutEnumNet GetMeaLayout ()

  Gets the MEA layout which is connected to the MEA2100 device.
- virtual uint32_t GetAdcDataFormat (uint32_t virtualDevice)

  Gets the ADC data format, 16 means 16 bits, 24 means 24 bits, 32 means 32 bits.
- virtual uint32_t GetAnalogValueUnit (uint32_t virtualDevice, DacqGroupChannelEnumNet group, [System::Runtime::InteropServices::Out] AnalogUnitEnumNet% unit)
- virtual uint32_t GetResolutionPerDigit (uint32_t virtualDevice, DacqGroupChannelEnumNet group, [System::Runtime::InteropServices::Out] int% res, [System::Runtime::InteropServices::Out] int% resUnit)
- virtual uint32_t GetAdcZero (uint32_t virtualDevice, DacqGroupChannelEnumNet group, [System::Runtime::InteropServices::Out] int% adcz)
- virtual uint32_t GetHardwareMinRange (uint32_t virtualDevice, DacqGroupChannelEnumNet group, [System::Runtime::InteropServices::Out] int% r, [System::Runtime::InteropServices::Out] int% rUnit)
- virtual uint32_t GetHardwareMaxRange (uint32_t virtualDevice, DacqGroupChannelEnumNet group, [System::Runtime::InteropServices::Out] int% r, [System::Runtime::InteropServices::Out] int% rUnit)
- virtual uint32_t GetDataFormat (uint32_t virtualDevice, DacqGroupChannelEnumNet group, [System::Runtime::InteropServices::Out] int% numberOfBits)
- virtual uint32_t GetNumberOfDataBits (uint32_t virtualDevice, DacqGroupChannelEnumNet group, [System::Runtime::InteropServices::Out] int% numberOfBits)

  Get the real number of data bits.
- virtual void SetSamplerate (int32_t rate, unsigned int oversample, unsigned int virtualDevice)

  Sets the sampling frequency of the device.
- virtual int32_t GetSamplerate (unsigned int virtualDevice)

  Gets the sampling frequency of the device.
- virtual uint32_t GetMaxSamplingFrequency (int virtualDevice)

  Gets the maximal sampling frequency of the device.
- virtual uint32_t GetMinSamplingFrequencyStepsize ()

  Gets the minimal sampling frequency step size increment value of the device.

- virtual int32_t GetChannelsInBlock (unsigned int virtualDevice)

  Get the number of 16 bit datawords which will be collected per sample frame, use after the device is configured.
- virtual void GetChannelLayout ([System::Runtime::InteropServices::Out]int% AnalogChannels, [System::Runtime::InteropServices::Out]int% DigitalChannels, [System::Runtime::InteropServices::Out]int% ChecksumChannels, [System::Runtime::InteropServices::Out]int% TimestampChannels, [System::Runtime::InteropServices::Out]int% ChannelsInBlock, unsigned int virtualDevice)
- virtual void SendStartDacq ()

  Start sampling.
- virtual void SendStartDacq (int VirtualDacqMap)

  Start sampling.
- virtual void SendStartStgAndDacq (uint32_t trigger_map, int VirtualDacqMap)

  Start sampling together with the STG.
- virtual void SendStopDacq ()

  Stop sampling.
- virtual void SendStopDacq (int VirtualDacqMap)

  Stop sampling.

Parameters

VirtualDacqMap	

- virtual void SendStopStgAndDacq (uint32_t trigger_map, int VirtualDacqMap)

  Stop sampling together with the STG.
- virtual void SendStopStgAndDacqWithOptions (uint32_t trigger_map, int VirtualDacqMap, int options)

  Stop sampling together with the STG and options.
- virtual void StartLoop ()

  Start the data acquisition thread.
- virtual void StartLoop (int32_t timeout)

  Start the data acquisition thread.
- virtual void StartLoop (int32_t timeout, int32_t numSubmittedUsbBuffers, int32_t numUsbBuffers, int32_t packetsInUrb)

  Start the data acquisition thread.
- virtual void StartLoop (int32_t timeout, int32_t numSubmittedUsbBuffers, int32_t numUsbBuffers, int32_t packetsInUrb, uint32_t virtualDevice)

  Start the data acquisition thread.
- virtual void StopLoop ()
- virtual void ClearBuffers ()
- virtual void StartDacq ()

  Start the data acquisition thread and sampling.
- virtual void StartDacq (int32_t timeout)

  Start the data acquisition thread and sampling.
- virtual void StartDacq (int32_t timeout, int32_t numSubmittedUsbBuffers, int32_t numUsbBuffers, int32_t packetsInUrb)

  Start the data acquisition thread and sampling.
- virtual void StartDacq (int32_t timeout, int32_t numSubmittedUsbBuffers, int32_t numUsbBuffers, int32_t packetsInUrb, uint32_t virtualDevice)

  Start the data acquisition thread and sampling.
- virtual void StopDacq ()

  Stop the data acquisition thread and sampling.
- virtual void StopDacq (uint32_t virtualDevice)

  Stop the data acquisition thread and sampling.

- virtual uint32_t SetPoti (uint32_t channel, uint32_t value, bool write_nvram)
- virtual uint32_t GetPoti (uint32_t channel, [System::Runtime::InteropServices::Out]uint32_t% value)
- virtual CFilterPropertyNet ^ GetFilterProperty (DacqGroupChannelEnumNet GroupID, unsigned int index)
- virtual array< CFilterPropertyNet^ > ^ CMcsUsbDacqNet::GetFilterProperties (DacqGroupChannelEnumNet GroupID)
- int GetChannelDataFillSize ()
- virtual void SetSelectedChannels (int nChannels, int queuesize, int threshold, SampleSizeNet samplesize, int ChannelsInBlock)

  Create a FIFO queue per channel. Each channel will have its own FIFO and Callback function.
- virtual void SetSelectedChannels (int nChannels, int queuesize, int threshold, SampleSizeNet samplesize, SampleDstSizeNet sampleDstSize, int ChannelsInBlock)
- virtual void SetSelectedChannels (array< bool >^ selectedChannels, int queuesize, int threshold, SampleSizeNet samplesize, int ChannelsInBlock)

  Create a FIFO queue per channel. Each channel will have its own FIFO and Callback function.
- virtual void SetSelectedChannels (array< bool >^ selectedChannels, int queuesize, int threshold, SampleSizeNet samplesize, SampleDstSizeNet sampleDstSize, int ChannelsInBlock)
- virtual void SetSelectedData (int nChannels, int queuesize, int threshold, SampleSizeNet samplesize, int ChannelsInBlock)

  Create a common FIFO queue for all channels. Use handle = 0 in the ChannelBlock_ReadFrames... functions.
- virtual void SetSelectedData (int nChannels, int queuesize, int threshold, SampleSizeNet samplesize, SampleDstSizeNet sampleDstSize, int ChannelsInBlock)
- virtual void SetSelectedData (array< bool >^ selectedChannels, int queuesize, int threshold, SampleSizeNet samplesize, int ChannelsInBlock)

  Create a common FIFO queue for all channels. Use handle = 0 in the ChannelBlock_ReadFrames... functions.
- virtual void SetSelectedData (array< bool >^ selectedChannels, int queuesize, int threshold, SampleSizeNet samplesize, SampleDstSizeNet sampleDstSize, int ChannelsInBlock)
- virtual int AddSelectedChannelsQueue (int nByteOffset, int nChannelOffset, int nChannels, int queuesize, int threshold, SampleSizeNet samplesize)

  Adds a common FIFO queue for all channels. Data in callback will be a list per channel. Use ChannelBlock_Read↩ FramesDict... 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 >^ selected↩ Channels, 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_Read↩ FramesDict... with handle = 0 to read the data.
- virtual int AddSelectedChannelsQueue (int nByteOffset, int nChannelOffset, array< bool >^ selected↩ Channels, int queuesize, int threshold, SampleSizeNet samplesize, SampleDstSizeNet sampleDstSize)
- 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

11.49.1 Detailed Description

Base class for data acquisition devices.

11.49.2 Constructor & Destructor Documentation

11.49.2.1 CMcsUsbDacqNet() `CMcsUsbDacqNet ( )`

11.49.2.2 ~CMcsUsbDacqNet() `~CMcsUsbDacqNet ( )`

11.49.3 Member Function Documentation

11.49.3.1 AddSelectedChannelsQueue() [1/4]

```
virtual int AddSelectedChannelsQueue (
    int nByteOffset,
    int nChannelOffset,
    array< bool >^ selectedChannels,
    int queuesize,
    int threshold,
    SampleSizeNet samplesize ) [virtual]
```

Adds a common FIFO queue for all channels. Data in callback will be a list per channel. Use ChannelBlock_Read↩ FramesDict... with handle = 0 to read the data.

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

Parameters

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.

11.49.3.2 AddSelectedChannelsQueue() [2/4]

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

11.49.3.3 AddSelectedChannelsQueue() [3/4]

```
virtual int AddSelectedChannelsQueue (
    int nByteOffset,
    int nChannelOffset,
    int nChannels,
    int queuesize,
    int threshold,
    SampleSizeNet samplesize )  [virtual]
```

Adds a common FIFO queue for all channels. Data in callback will be a list per channel. Use ChannelBlock_Read↩ FramesDict... with handle = 0 to read the data.

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

Parameters

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.

11.49.3.4 AddSelectedChannelsQueue() [4/4]

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

11.49.3.5 ChannelBlock_AvailFrames() [1/2]

```
virtual uint32_t ChannelBlock_AvailFrames (
    int handle )  [virtual]
```

Get the number of sample frames already available in the FIFO.

Parameters

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.

11.49.3.6 ChannelBlock_AvailFrames() [2/2]

```
virtual uint32_t ChannelBlock_AvailFrames (
    int handle,
    int queue )  [virtual]
```

11.49.3.7 ChannelBlock_ReadAsFrameArrayI16() [1/2]

```
virtual array<array<int16_t>^> ^ 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.

11.49.3.8 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.

11.49.3.9 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.

11.49.3.10 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.

11.49.3.11 ChannelBlock_ReadAsFrameArrayUI16() [1/2] virtual array<array<uint16_t>^> ^ ChannelBlock_ReadAsFrameArrayUI16 (int *handle*, int *frames*, [System::Runtime::InteropServices::Out] int % *frames_ret*) [virtual]

Read data from a FIFO queue as array of uint16_t data frame arrays

Parameters

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.

11.49.3.12 ChannelBlock_ReadAsFrameArrayUI16() [2/2] virtual array<array<uint16_t>^> ^ ChannelBlock_ReadAsFrameArrayUI16 (int *handle*, int *queue*, int *frames*, [System::Runtime::InteropServices::Out] int % *frames_ret*) [virtual]

Read data from a FIFO queue as array of uint16_t data frame arrays

Parameters

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.

11.49.3.13 ChannelBlock_ReadAsFrameArrayUI32() [1/2] virtual array<array<uint32_t>^> ^ ChannelBlock_ReadAsFrameArrayUI32 (int *handle*, int *frames*, [System::Runtime::InteropServices::Out] int % *frames_ret*) [virtual]

Read data from a FIFO queue as array of uint16_t data frame arrays

Parameters

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.

11.49.3.14 ChannelBlock_ReadAsFrameArrayUI32() [2/2]

```
virtual array<array<uint32_t>^> ^
ChannelBlock_ReadAsFrameArrayUI32 (
            int handle,
            int queue,
            int frames,
            [System::Runtime::InteropServices::Out] int % frames_ret )  [virtual]
```

Read data from a FIFO queue as array of uint16_t data frame arrays

Parameters

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.

11.49.3.15 ChannelBlock_ReadFramesDictI16()

```
virtual System::Collections::Generic::Dictionary<int,
array<int16_t>^> ^ ChannelBlock_ReadFramesDictI16 (
            int handle,
            int frames,
            [System::Runtime::InteropServices::Out] int % frames_ret )  [virtual]
```

Read data from a FIFO queue in int16_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number

Parameters

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.

11.49.3.16 ChannelBlock_ReadFramesDictI32()

```
virtual System::Collections::Generic::Dictionary<int,
array<int32_t>^> ^ ChannelBlock_ReadFramesDictI32 (
            int handle,
            int frames,
            [System::Runtime::InteropServices::Out] int % frames_ret )  [virtual]
```

Read data from a FIFO queue in int32_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number

Parameters

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.

11.49.3.17 ChannelBlock_ReadFramesDictUI16()

```
virtual System::Collections::Generic::Dictionary<int,
array<uint16_t>^> ^ ChannelBlock_ReadFramesDictUI16 (
            int handle,
            int frames,
            [System::Runtime::InteropServices::Out] int % frames_ret )  [virtual]
```

Read data from a FIFO queue in uint16_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number

Parameters

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.

11.49.3.18 ChannelBlock_ReadFramesDictUI32() virtual System::Collections::Generic::Dictionary<int, array<uint32_t>^> ^ ChannelBlock_ReadFramesDictUI32 (

```
int handle,
int frames,
[System::Runtime::InteropServices::Out] int % frames_ret )  [virtual]
```

Read data from a FIFO queue in uint32_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number

Parameters

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.

11.49.3.19 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.

11.49.3.20 ChannelBlock_ReadFramesI16() [2/2]

```
virtual array<int16_t> ^ ChannelBlock_ReadFramesI16 (
            int handle,
            int frames,
            [System::Runtime::InteropServices::Out] int % frames_ret )  [virtual]
```

Read data from a FIFO queue in int16_t data format

Parameters

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.

11.49.3.21 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.

11.49.3.22 ChannelBlock_ReadFramesI32() [2/2]

```
virtual array<int32_t> ^ ChannelBlock_ReadFramesI32 (
            int handle,
            int frames,
            [System::Runtime::InteropServices::Out] int % frames_ret )  [virtual]
```

Read data from a FIFO queue in uint32_t data format

Parameters

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.

11.49.3.23 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.

11.49.3.24 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.

11.49.3.25 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.

11.49.3.26 ChannelBlock_ReadFramesUI32() [2/2]

```
virtual array<uint32_t> ^ ChannelBlock_Read↩
FramesUI32 (
    int handle,
    int frames,
    [System::Runtime::InteropServices::Out] int % frames_ret )  [virtual]
```

Read data from a FIFO queue in uint32_t data format

Parameters

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.

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

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

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

Gets the adapter which is connected to the MEA2100 device.

Returns

AdapterTypeEnumNet which enumerates the possible adapters.

11.49.3.30 GetAdcDataFormat() `virtual uint32_t GetAdcDataFormat ( uint32_t virtualDevice ) [virtual]`

Gets the ADC data format, 16 means 16 bits, 24 means 24 bits, 32 means 32 bits.

Returns

The data format in bits.

11.49.3.31 GetAdcZero() virtual uint32_t GetAdcZero (

```
uint32_t virtualDevice,
DacqGroupChannelEnumNet group,
[System::Runtime::InteropServices::Out] int% adcz ) [virtual]
```

11.49.3.32 GetAnalogValueUnit() virtual uint32_t GetAnalogValueUnit (

```
uint32_t virtualDevice,
DacqGroupChannelEnumNet group,
[System::Runtime::InteropServices::Out] AnalogUnitEnumNet% unit ) [virtual]
```

11.49.3.33 GetChannelDataFillSize() int GetChannelDataFillSize ( )

11.49.3.34 GetChannelLayout() virtual 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]
```

11.49.3.35 GetChannelsInBlock() virtual int32_t GetChannelsInBlock (

```
unsigned int virtualDevice ) [virtual]
```

Get the number of 16 bit datawords which will be collected per sample frame, use after the device is configured.

Returns

Number of 16 bit datawords per sample frame.

11.49.3.36 GetDataFormat() virtual uint32_t GetDataFormat (

```
uint32_t virtualDevice,
DacqGroupChannelEnumNet group,
[System::Runtime::InteropServices::Out] int% numberOfBits ) [virtual]
```

11.49.3.37 GetDataMode() virtual DataModeEnumNet GetDataMode (

```
unsigned int virtualDevice ) [virtual]
```

Gets the data mode, can be 16, 24 or 32bit, all signed or unsigned on the MEA2100 device.

Parameters

virtualDevice	Virtual device to use.

Returns

DataModeEnumNet which enumerates the possible data modes.

11.49.3.38 GetDigitalSource() [1/5]

```
void GetDigitalSource (
    DigitalTargetEnumNet digitaltarget,
    int32_t NrChannel,
    [System::Runtime::InteropServices::Out] DigitalSource< digitalsourceenum >^% source,
    [System::Runtime::InteropServices::Out] int% bitnumber_offset )
```

Gets the function/source of an digital output bit.

This is the templated generic implementation.

Parameters

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.

11.49.3.39 GetDigitalSource() [2/5]

```
void GetDigitalSource (
    DigitalTargetEnumNet digitaltarget,
    int32_t NrChannel,
    [System::Runtime::InteropServices::Out] DigitalSourceEnumNet% source,
    [System::Runtime::InteropServices::Out] int% bitnumber_offset )
```

Gets the function/source of an digital output bit.

This overload is for the MEA2100 device.

Parameters

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.

11.49.3.40 GetDigitalSource() [3/5]

```
void GetDigitalSource (
    DigitalTargetEnumNet digitaltarget,
    int32_t NrChannel,
    [System::Runtime::InteropServices::Out] MEA2100_256DigitalSourceEnumNet% source,
    [System::Runtime::InteropServices::Out] int% bitnumber_offset )
```

Gets the function/source of an digital output bit.

This overload is for the MEA2100-256 device.

Parameters

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.

11.49.3.41 GetDigitalSource() [4/5]

```
void GetDigitalSource (
    DigitalTargetEnumNet digitaltarget,
    int32_t NrChannel,
    [System::Runtime::InteropServices::Out] SCUDigitalSourceEnumNet% source,
    [System::Runtime::InteropServices::Out] int% bitnumber_offset )
```

Gets the function/source of an digital output bit.

This overload is for the SCU device.

Parameters

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.

11.49.3.42 GetDigitalSource() [5/5]

```
void GetDigitalSource (
    DigitalTargetEnumNet digitaltarget,
    int32_t NrChannel,
    [System::Runtime::InteropServices::Out] W2100DigitalSourceEnumNet% source,
    [System::Runtime::InteropServices::Out] int% bitnumber_offset )
```

Gets the function/source of an digital output bit.

This overload is for the W2100 device.

Parameters

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.

11.49.3.43 GetErrorMessage()

```
uint32_t GetErrorMessage (
    [System::Runtime::InteropServices::Out] String^% errorString,
    [System::Runtime::InteropServices::Out] int% info )
```

11.49.3.44 GetFilterProperty()

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

11.49.3.45 GetGroupChannelDataI16()

```
virtual System::Collections::Generic::Dictionary<int,
array<int16_t>^> ^ GetGroupChannelDataI16 (
    DacqGroupChannelEnumNet group,
    int frames,
    [System::Runtime::InteropServices::Out] int % frames_ret )  [virtual]
```

Read data from a FIFO queue in int16_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number

Parameters

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.

11.49.3.46 GetGroupChannelDataI32()

```
virtual System::Collections::Generic::Dictionary<int,
array<int32_t>^> ^ GetGroupChannelDataI32 (
    DacqGroupChannelEnumNet group,
    int frames,
    [System::Runtime::InteropServices::Out] int % frames_ret )  [virtual]
```

Read data from a FIFO queue in int32_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number

Parameters

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.

11.49.3.47 GetGroupChannelDataUI16()

```
virtual System::Collections::Generic::Dictionary<int,
array<uint16_t>^> ^ GetGroupChannelDataUI16 (
            DacqGroupChannelEnumNet group,
            int frames,
            [System::Runtime::InteropServices::Out] int % frames_ret )  [virtual]
```

Read data from a FIFO queue in uint16_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number

Parameters

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.

11.49.3.48 GetGroupChannelDataUI32()

```
virtual System::Collections::Generic::Dictionary<int,
array<uint32_t>^> ^ GetGroupChannelDataUI32 (
            DacqGroupChannelEnumNet group,
            int frames,
            [System::Runtime::InteropServices::Out] int % frames_ret )  [virtual]
```

Read data from a FIFO queue in uint32_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number

Parameters

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.

11.49.3.49 GetHardwareMaxRange()

```
virtual uint32_t GetHardwareMaxRange (
            uint32_t virtualDevice,
            DacqGroupChannelEnumNet group,
            [System::Runtime::InteropServices::Out] int% r,
            [System::Runtime::InteropServices::Out] int% rUnit )  [virtual]
```

11.49.3.50 GetHardwareMinRange()

```
virtual uint32_t GetHardwareMinRange (
            uint32_t virtualDevice,
            DacqGroupChannelEnumNet group,
            [System::Runtime::InteropServices::Out] int% r,
            [System::Runtime::InteropServices::Out] int% rUnit )  [virtual]
```

11.49.3.51 GetMaxSamplingFrequency()

```
virtual uint32_t GetMaxSamplingFrequency (
            int virtualDevice )  [virtual]
```

Gets the maximal sampling frequency of the device.

Returns

Sampling frequency in Hz.

11.49.3.52 GetMeaLayout() virtual MeaLayoutEnumNet GetMeaLayout ( ) [virtual]

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

Returns

MeaLayoutEnumNet which enumerates the MEA types.

11.49.3.53 GetMinSamplingFrequencyStepsize() virtual uint32_t GetMinSamplingFrequencyStepsize ( ) [virtual]

Gets the minimal sampling frequency step size increment value of the device.

Returns

Sampling frequency step size in Hz.

11.49.3.54 GetNumberOfDataBits()

```
virtual uint32_t GetNumberOfDataBits (
    uint32_t virtualDevice,
    DacqGroupChannelEnumNet group,
    [System::Runtime::InteropServices::Out] int% numberOfBits ) [virtual]
```

Get the real number of data bits.

This value may be different from the value returned by GetDataFormat, e.g. in MC_Card the data are shifted 2 bits so the real number is 14 while the data format is 16 bits

11.49.3.55 GetPoti()

```
virtual uint32_t GetPoti (
    uint32_t channel,
    [System::Runtime::InteropServices::Out] uint32_t% value ) [virtual]
```

11.49.3.56 GetResolutionPerDigit()

```
virtual uint32_t GetResolutionPerDigit (
    uint32_t virtualDevice,
    DacqGroupChannelEnumNet group,
    [System::Runtime::InteropServices::Out] int% res,
    [System::Runtime::InteropServices::Out] int% resUnit ) [virtual]
```

11.49.3.57 GetSamplerate()

```
virtual int32_t GetSamplerate (
    unsigned int virtualDevice ) [virtual]
```

Gets the sampling frequency of the device.

Returns

Sampling frequency in Hz.

11.49.3.58 GetVoltageRangeIndex() virtual uint32_t GetVoltageRangeIndex (unsigned int *virtualDevice*) [virtual]

11.49.3.59 GetVoltageRangeInMicroVolt() virtual int32_t GetVoltageRangeInMicroVolt (unsigned int *virtualDevice*) [virtual]

Gets the currently selected voltage range on devices which support multiple voltage ranges.

Returns

The Voltage Range in uV.

11.49.3.60 GetVoltageRangeInMilliVolt() virtual int32_t GetVoltageRangeInMilliVolt ( ) [virtual]

Gets the currently selected voltage range on devices which support multiple voltage ranges.

Returns

The rounded Voltage Range in mV.

11.49.3.61 HWInfo() CHWInfo ^ HWInfo ( )

11.49.3.62 SendStartDacq() [1/2] virtual void SendStartDacq ( ) [virtual]

Start sampling.

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

Start sampling.

Parameters

VirtualDacqMap	

11.49.3.64 SendStartStgAndDacq() virtual void SendStartStgAndDacq (uint32_t *trigger_map*, int *VirtualDacqMap*) [virtual]

Start sampling together with the STG.

Parameters

trigger_map	
VirtualDacqMap	

11.49.3.65 SendStopDacq() [1/2] virtual void SendStopDacq ( ) [virtual]

Stop sampling.

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

Stop sampling.

Parameters

VirtualDacqMap	

11.49.3.67 SendStopStgAndDacq() virtual void SendStopStgAndDacq (uint32_t *trigger_map*, int *VirtualDacqMap*) [virtual]

Stop sampling together with the STG.

Parameters

trigger_map	

11.49.3.68 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	

11.49.3.69 SetDataMode()

```
virtual void SetDataMode (
  DataModeEnumNet dataMode,
  unsigned int virtualDevice )  [virtual]
```

Sets the data mode, can be 16, 24 or 32bit, all signed or unsigned on the MEA2100 device.

Parameters

dataMode	DataModeEnumNet enumerates the possible data modes.
virtualDevice	Virtual device to use.

11.49.3.70 SetDigitalSource() [1/5]

```
void SetDigitalSource (
  DigitalTargetEnumNet digitaltarget,
  int32_t NrChannel,
  DigitalSource< digitalsourceenum >^ source,
  int bitnumber_offset )
```

Sets the function/source of an digital output bit.

This is the templated generic implementation.

Parameters

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.

11.49.3.71 SetDigitalSource() [2/5]

```
void SetDigitalSource (
    DigitalTargetEnumNet digitaltarget,
    int32_t NrChannel,
    DigitalSourceEnumNet source,
    int bitnumber_offset )
```

Sets the function/source of an digital output bit.

This overload is for the MEA2100 device.

Parameters

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.

11.49.3.72 SetDigitalSource() [3/5]

```
void SetDigitalSource (
    DigitalTargetEnumNet digitaltarget,
    int32_t NrChannel,
    MEA2100_256DigitalSourceEnumNet source,
    int bitnumber_offset )
```

Sets the function/source of an digital output bit.

This overload is for the MEA2100-256 device.

Parameters

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.

11.49.3.73 SetDigitalSource() [4/5]

```
void SetDigitalSource (
    DigitalTargetEnumNet digitaltarget,
```

```
int32_t NrChannel,
SCUDigitalSourceEnumNet source,
int bitnumber_offset )
```

Sets the function/source of an digital output bit.

This overload is for the SCU device.

Parameters

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.

11.49.3.74 SetDigitalSource() [5/5]

```
void SetDigitalSource (
DigitalTargetEnumNet digitaltarget,
int32_t NrChannel,
W2100DigitalSourceEnumNet source,
int bitnumber_offset )
```

Sets the function/source of an digital output bit.

This overload is for the W2100 device.

Parameters

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.

11.49.3.75 SetPoti()

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

11.49.3.76 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.

11.49.3.77 SetSelectedChannels() [1/4]

```
virtual void SetSelectedChannels (
    array< bool >^ selectedChannels,
    int queuesize,
    int threshold,
    SampleSizeNet samplesize,
    int ChannelsInBlock )  [virtual]
```

Create a FIFO queue per channel. Each channel will have its own FIFO and Callback function.

When using a 32bit sample size, the number obtained from GetChannelsInBlock must be 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.

11.49.3.78 SetSelectedChannels() [2/4] virtual void SetSelectedChannels (

```
array< bool >^ selectedChannels,
int queuesize,
int threshold,
SampleSizeNet samplesize,
SampleDstSizeNet sampleDstSize,
int ChannelsInBlock )  [virtual]
```

11.49.3.79 SetSelectedChannels() [3/4] virtual void SetSelectedChannels (

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

Create a FIFO queue per channel. Each channel will have its own FIFO and Callback function.

When using a 32bit sample size, the number obtained from GetChannelsInBlock must be 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.

11.49.3.80 SetSelectedChannels() [4/4]

```
virtual void SetSelectedChannels (
    int nChannels,
    int queuesize,
    int threshold,
    SampleSizeNet samplesize,
    SampleDstSizeNet sampleDstSize,
    int ChannelsInBlock ) [virtual]
```

11.49.3.81 SetSelectedChannelsQueue() [1/4]

```
virtual void SetSelectedChannelsQueue (
    array< bool >^ selectedChannels,
    int queuesize,
    int threshold,
    SampleSizeNet samplesize,
    int ChannelsInBlock ) [virtual]
```

Create a common FIFO queue for all channels. Data in callback will be a list per channel. Use ChannelBlock_↩ ReadFramesDict... with handle = 0 to read the data.

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

Parameters

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.

11.49.3.82 SetSelectedChannelsQueue() [2/4] virtual void SetSelectedChannelsQueue (

```
array< bool >^ selectedChannels,
int queuesize,
int threshold,
SampleSizeNet samplesize,
SampleDstSizeNet sampleDstSize,
int ChannelsInBlock )  [virtual]
```

11.49.3.83 SetSelectedChannelsQueue() [3/4] virtual void SetSelectedChannelsQueue (

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

Create a common FIFO queue for all channels. Data in callback will be a list per channel. Use ChannelBlock_↩ ReadFramesDict... with handle = 0 to read the data.

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

Parameters

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.

11.49.3.84 SetSelectedChannelsQueue() [4/4]

```
virtual void SetSelectedChannelsQueue (
    int nChannels,
    int queuesize,
    int threshold,
    SampleSizeNet samplesize,
    SampleDstSizeNet sampleDstSize,
    int ChannelsInBlock )  [virtual]
```

11.49.3.85 SetSelectedData() [1/4]

```
virtual void SetSelectedData (
    array< bool >^ selectedChannels,
    int queuesize,
    int threshold,
    SampleSizeNet samplesize,
    int ChannelsInBlock )  [virtual]
```

Create a common FIFO queue for all channels. Use handle = 0 in the ChannelBlock_ReadFrames... functions.

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

Parameters

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.

11.49.3.86 SetSelectedData() [2/4]

```
virtual void SetSelectedData (
    array< bool >^ selectedChannels,
    int queuesize,
    int threshold,
    SampleSizeNet samplesize,
    SampleDstSizeNet sampleDstSize,
    int ChannelsInBlock )  [virtual]
```

11.49.3.87 SetSelectedData() [3/4]

```
virtual void SetSelectedData (
    int nChannels,
    int queuesize,
    int threshold,
    SampleSizeNet samplesize,
    int ChannelsInBlock )  [virtual]
```

Create a common FIFO queue for all channels. Use handle = 0 in the ChannelBlock_ReadFrames... functions.

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

Parameters

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.

11.49.3.88 SetSelectedData() [4/4]

```
virtual void SetSelectedData (
    int nChannels,
    int queuesize,
    int threshold,
    SampleSizeNet samplesize,
    SampleDstSizeNet sampleDstSize,
    int ChannelsInBlock )  [virtual]
```

11.49.3.89 SetupGroupDacqQueue() [1/2]

```
void SetupGroupDacqQueue (
    int queuesize,
    int threshold )
```

11.49.3.90 SetupGroupDacqQueue() [2/2]

```
void SetupGroupDacqQueue (
    int queuesize,
    int threshold,
    unsigned int virtualDevice )
```

11.49.3.91 SetVoltageRangeByIndex()

```
virtual void SetVoltageRangeByIndex (
    int32_t voltageRangeIndex,
    unsigned int virtualDevice )  [virtual]
```

Sets the voltage range on devices which support multiple voltage ranges.

Parameters

voltageRangeIndex	Voltage Range to use as index, smaller values are larger voltage ranges.

11.49.3.92 SetVoltageRangeInMicroVolt()

```
virtual void SetVoltageRangeInMicroVolt (
    int32_t voltageRange,
    unsigned int virtualDevice ) [virtual]
```

Sets the voltage range on devices which support multiple voltage ranges.

Parameters

voltageRange	Voltage Range to use in µV.

This replaces SetVoltageRange, where the value of the range was in mV!

11.49.3.93 StartDacq() [1/4]

```
virtual void StartDacq ( ) [virtual]
```

Start the data acquisition thread and sampling.

11.49.3.94 StartDacq() [2/4]

```
virtual void StartDacq (
    int32_t timeout ) [virtual]
```

Start the data acquisition thread and sampling.

Parameters

timeout	Timeout in ms.

11.49.3.95 StartDacq() [3/4]

```
virtual void StartDacq (
    int32_t timeout,
    int32_t numSubmittedUsbBuffers,
    int32_t numUsbBuffers,
    int32_t packetsInUrb ) [virtual]
```

Start the data acquisition thread and sampling.

Parameters

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.

11.49.3.96 StartDacq() [4/4]

```
virtual void StartDacq (
    int32_t timeout,
    int32_t numSubmittedUsbBuffers,
    int32_t numUsbBuffers,
    int32_t packetsInUrb,
    uint32_t virtualDevice ) [virtual]
```

Start the data acquisition thread and sampling.

Parameters

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.

11.49.3.97 StartLoop() [1/4] virtual void StartLoop ( ) [virtual]

Start the data acquisition thread.

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

Start the data acquisition thread.

Parameters

timeout	Timeout in ms.

11.49.3.99 StartLoop() [3/4] virtual void StartLoop (int32_t *timeout*, int32_t *numSubmittedUsbBuffers*, int32_t *numUsbBuffers*, int32_t *packetsInUrb*) [virtual]

Start the data acquisition thread.

Parameters

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.

11.49.3.100 StartLoop() [4/4]

```
virtual void StartLoop (
    int32_t timeout,
    int32_t numSubmittedUsbBuffers,
    int32_t numUsbBuffers,
    int32_t packetsInUrb,
    uint32_t virtualDevice ) [virtual]
```

Start the data acquisition thread.

Parameters

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.

11.49.3.101 StopDacq() [1/2] virtual void StopDacq ( ) [virtual]

Stop the data acquisition thread and sampling.

11.49.3.102 StopDacq() [2/2] virtual void StopDacq (uint32_t *virtualDevice*) [virtual]

Stop the data acquisition thread and sampling.

Parameters

virtualDevice	Virtual Device to start.

11.49.3.103 StopLoop() virtual void StopLoop ( ) [virtual]

11.49.4 Member Data Documentation

11.49.4.1 Error_Callback_Aquisition_Stopped const int Error_Callback_Aquisition_Stopped = 0x200 [static]

11.49.4.2 Error_Callback_Data_lost const int Error_Callback_Data_lost = 5 [static]

11.49.4.3 Error_Callback_Frames_Lost const int Error_Callback_Frames_Lost = 4 [static]

11.49.4.4 Error_Callback_Packet_Error `const int Error_Callback_Packet_Error = 1 [static]`

11.49.4.5 Error_Callback_Queue_Full `const int Error_Callback_Queue_Full = 0x100 [static]`

11.49.4.6 Error_Callback_RingQueue_Full `const int Error_Callback_RingQueue_Full = 3 [static]`

11.49.5 Property Documentation

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

The sampling frequency of the device in Hz.

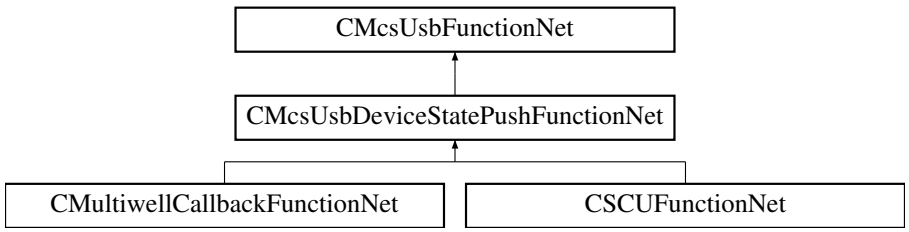
11.49.6 Event Documentation

11.49.6.1 ChannelDataEvent `OnChannelData^ ChannelDataEvent [add], [remove], [raise]`

11.49.6.2 ErrorEvent `OnError^ ErrorEvent [add], [remove], [raise]`

11.50 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

11.50.1 Constructor & Destructor Documentation

11.50.1.1 CMcsUsbDeviceStatePushFunctionNet() CMcsUsbDeviceStatePushFunctionNet (CMcsUsbNet^ *mcsusb*, CMcsUsbFunctionPointerContainer^ *pDevice*) [protected]

11.50.2 Member Function Documentation

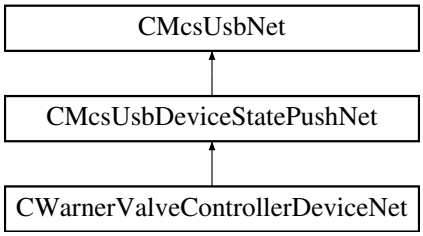
11.50.2.1 TriggerStatus() void TriggerStatus ( )

11.50.3 Event Documentation

11.50.3.1 McsUsbDeviceStateEvent OnMcsUsbDeviceState^ McsUsbDeviceStateEvent [add], [remove], [raise]

11.51 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

11.51.1 Constructor & Destructor Documentation

11.51.1.1 CMcsUsbDeviceStatePushNet() CMcsUsbDeviceStatePushNet (CMcsUsbPointerContainer^ *pDevice*) [protected]

11.51.2 Member Function Documentation

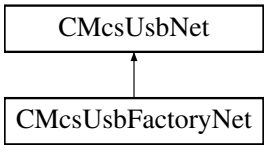
11.51.2.1 TriggerStatus() void TriggerStatus ( )

11.51.3 Event Documentation

11.51.3.1 McsUsbDeviceStateEvent OnMcsUsbDeviceState^ McsUsbDeviceStateEvent [add], [remove], [raise]

11.52 CMcsUsbFactoryNet Class Reference

Inheritance diagram for CMcsUsbFactoryNet:



Public Member Functions

- CMcsUsbFactoryNet ()
- ~CMcsUsbFactoryNet ()
- unsigned int GetNumDestinations ()
- String ^ GetDestinationName (unsigned int index)
- String ^ GetDestinationName (CFirmwareDestinationNet dest)
- void SetDestinationSerialNumber (CFirmwareDestinationNet dest, String^ serialnumber)
- String ^ GetDestinationSerialNumber (CFirmwareDestinationNet dest)
- CFirmwareDestinationNet GetDestination (unsigned int index)
- CFirmwareDestinationNet GetDestination (String^ Key)
- unsigned int GetDestinationTargetAddress (CFirmwareDestinationNet destination)

  Gets the target base address for the destination.
- uint32_t ChangeSerialNumber (String^ serial)
- bool LoadUserFirmware (String^ FirmwareFile, CMcsUsbListEntryNet^ listEntry)

  Send the DSP Firmware to the MEA21 device.
- bool LoadUserFirmware (String^ FirmwareFile, CMcsUsbListEntryNet^ listEntry, uint32_t LockMask)
- bool UpdateFirmware (String^ FirmwareFile, CMcsUsbListEntryNet^ listEntry, CFirmwareDestinationNet Dest, OnUpdateFirmwareStatusChange^ deleg, OnUpdateFirmwareProgress^ progress, bool SkipWait)

  Flashes a firmware file to the device.
- bool UpdateFirmware (String^ FirmwareFile, CMcsUsbListEntryNet^ listEntry, CFirmwareDestinationNet Dest, OnUpdateFirmwareStatusChange^ deleg, OnUpdateFirmwareProgress^ progress, bool SkipWait, unsigned int LockMask)
- bool UpdateFirmware (String^ FirmwareFile, CMcsUsbListEntryNet^ listEntry, CFirmwareDestinationNet dest)

  Flashes a firmware file to the device.
- bool UpdateFirmware (String^ FirmwareFile, CMcsUsbListEntryNet^ listEntry, CFirmwareDestinationNet dest, bool SkipWait)

  Flashes a firmware file to the device.
- bool UpdateFirmware (String^ FirmwareFile, CMcsUsbListEntryNet^ listEntry, CFirmwareDestinationNet dest, bool SkipWait, uint32_t LockMask)
- bool CompareFirmware (String^ FirmwareFile, CMcsUsbListEntryNet^ listEntry, CFirmwareDestinationNet Dest, OnUpdateFirmwareStatusChange^ deleg, OnUpdateFirmwareProgress^ progress, String^ MessagePrefix, unsigned int LockMask, [System::Runtime::InteropServices::Out] String^% ErrorText, [System::Runtime::InteropServices::Out] String^% Protokoll)
- uint32_t Coldstart (CFirmwareDestinationNet dest)
- int32_t GetXilinxFlashOffset (CFirmwareDestinationNet dest)
- uint32_t GetXilinxFlashReadCommand (CFirmwareDestinationNet dest)
- array< uint8_t > ^ DownloadFirmware (CFirmwareDestinationNet Dest, uint32_t Address, uint32_t length)
- bool GetUsercodeFromFlash (unsigned int FPGA, unsigned int Address, [System::Runtime::Interop↩ Services::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

11.52.1 Constructor & Destructor Documentation

11.52.1.1 CMcsUsbFactoryNet() `CMcsUsbFactoryNet ( )`

11.52.1.2 ~CMcsUsbFactoryNet() `~CMcsUsbFactoryNet ( )`

11.52.2 Member Function Documentation

11.52.2.1 ChangeSerialNumber() `uint32_t ChangeSerialNumber ( String^ serial )`

11.52.2.2 Coldstart()

```
uint32_t Coldstart (
    CFirmwareDestinationNet dest )
```

11.52.2.3 CompareFirmware()

```
bool CompareFirmware (
    String^ FirmwareFile,
    CMcsUsbListEntryNet^ listEntry,
    CFirmwareDestinationNet Dest,
    OnUpdateFirmwareStatusChange^ deleg,
    OnUpdateFirmwareProgress^ progress,
    String^ MessagePrefix,
    unsigned int LockMask,
    [System::Runtime::InteropServices::Out] String^% ErrorText,
    [System::Runtime::InteropServices::Out] String^% Protokoll )
```

11.52.2.4 DownloadFirmware()

```
array<uint8_t> ^ DownloadFirmware (
    CFirmwareDestinationNet Dest,
    uint32_t Address,
    uint32_t length )
```

11.52.2.5 FindFirmwareVersionMagicInBuffer()

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

11.52.2.6 GetChecksumFromFX3Image()

```
static uint32_t GetChecksumFromFX3Image (
    String^ FirmwareFile )  [static]
```

11.52.2.7 GetDestination() [1/2]

```
CFirmwareDestinationNet GetDestination (
    String^ Key )
```

11.52.2.8 GetDestination() [2/2]

```
CFirmwareDestinationNet GetDestination (
    unsigned int index )
```

11.52.2.9 GetDestinationDisplayLabel()

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

11.52.2.10 GetDestinationName() [1/2]

```
String ^ GetDestinationName (
    CFirmwareDestinationNet dest )
```

11.52.2.11 GetDestinationName() [2/2]

```
String ^ GetDestinationName (
    unsigned int index )
```

11.52.2.12 GetDestinationSerialNumber()

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

11.52.2.13 GetDestinationTargetAddress()

```
unsigned int GetDestinationTargetAddress (
    CFirmwareDestinationNet destination )
```

Gets the target base address for the destination.

Parameters

destination	The destination to be queried.

Returns

The base address as a 32 bit number, only the lower 16 bit represent the address.

11.52.2.14 GetFirmwareVersionFromFile() [1/2]

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

Retrives version info from a Firmware update file.

11.52.2.15 GetFirmwareVersionFromFile() [2/2]

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

11.52.2.16 GetFirmwareVersionFromHexFile()

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

11.52.2.17 GetNumDestinations() unsigned int GetNumDestinations ( )

11.52.2.18 GetUSBDeviceIDFromFX3Image() static uint32_t GetUSBDeviceIDFromFX3Image (String^ *FirmwareFile*) [static]

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

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

11.52.2.21 GetXilinxFlashOffset() int32_t GetXilinxFlashOffset (CFirmwareDestinationNet *dest*)

11.52.2.22 GetXilinxFlashReadCommand() uint32_t GetXilinxFlashReadCommand (CFirmwareDestinationNet *dest*)

11.52.2.23 LoadUserFirmware() [1/2] bool LoadUserFirmware (String^ *FirmwareFile*, CMcsUsbListEntryNet^ *listEntry*)

Send the DSP Firmware to the MEA21 device.

Parameters

FirmwareFile	Filename of the DSP Firmware (∗.bin) file.

Parameters

listEntry	Device to use for the connection. See CMcsUsbListNet.

11.52.2.24 LoadUserFirmware() [2/2]

```
bool LoadUserFirmware (
    String^ FirmwareFile,
    CMcsUsbListEntryNet^ listEntry,
    uint32_t LockMask )
```

11.52.2.25 ReadBlockFromFlash() [1/2]

```
array<unsigned char> ^ ReadBlockFromFlash (
    unsigned int FPGA,
    unsigned int Address )
```

11.52.2.26 ReadBlockFromFlash() [2/2]

```
void ReadBlockFromFlash (
    unsigned int FPGA,
    unsigned int Address,
    array< unsigned char >^ buffer,
    int position )
```

11.52.2.27 ReadBlockFromIFBGlobalEEprom()

```
array<unsigned char> ^ ReadBlockFromIFBGlobal↩
EEprom (
    unsigned int Address )
```

11.52.2.28 ReadBlockFromNVMEM()

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

11.52.2.29 SetDestinationSerialNumber()

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

11.52.2.30 UpdateFirmware() [1/5]

```
bool UpdateFirmware (
    String^ FirmwareFile,
    CMcsUsbListEntryNet^ listEntry,
    CFirmwareDestinationNet dest )
```

Flashes a firmware file to the device.

Parameters

FirmwareFile	Filename of the Firmware file.

Parameters

listEntry	Device to use for the connection.

11.52.2.31 UpdateFirmware() [2/5] bool UpdateFirmware (

```
String^ FirmwareFile,
CMcsUsbListEntryNet^ listEntry,
CFirmwareDestinationNet dest,
bool SkipWait )
```

Flashes a firmware file to the device.

Parameters

FirmwareFile	Filename of the Firmware file.

Parameters

listEntry	Device to use for the connection.

11.52.2.32 UpdateFirmware() [3/5] bool UpdateFirmware (

```
String^ FirmwareFile,
CMcsUsbListEntryNet^ listEntry,
CFirmwareDestinationNet dest,
bool SkipWait,
uint32_t LockMask )
```

11.52.2.33 UpdateFirmware() [4/5] bool UpdateFirmware (

```
String^ FirmwareFile,
CMcsUsbListEntryNet^ listEntry,
CFirmwareDestinationNet Dest,
```

```
OnUpdateFirmwareStatusChange^ deleg,
OnUpdateFirmwareProgress^ progress,
bool SkipWait )
```

Flashes a firmware file to the device.

Parameters

FirmwareFile	Filename of the Firmware file.

11.52.2.34 UpdateFirmware() [5/5]

```
bool UpdateFirmware (
String^ FirmwareFile,
CMcsUsbListEntryNet^ listEntry,
CFirmwareDestinationNet Dest,
OnUpdateFirmwareStatusChange^ deleg,
OnUpdateFirmwareProgress^ progress,
bool SkipWait,
unsigned int LockMask )
```

11.52.3 Member Data Documentation

11.52.3.1 FX3MCSDataAddress

```
const uint32_t FX3MCSDataAddress = 0x40037E00 [static]
```

11.52.3.2 FX3MCSDataDeviceIdOffset

```
const uint32_t FX3MCSDataDeviceIdOffset = 0x4 [static]
```

11.52.3.3 FX3MCSDataIFB1ImageOffset

```
const uint32_t FX3MCSDataIFB1ImageOffset = 0x2C [static]
```

11.52.3.4 FX3MCSDataIFB2ImageOffset

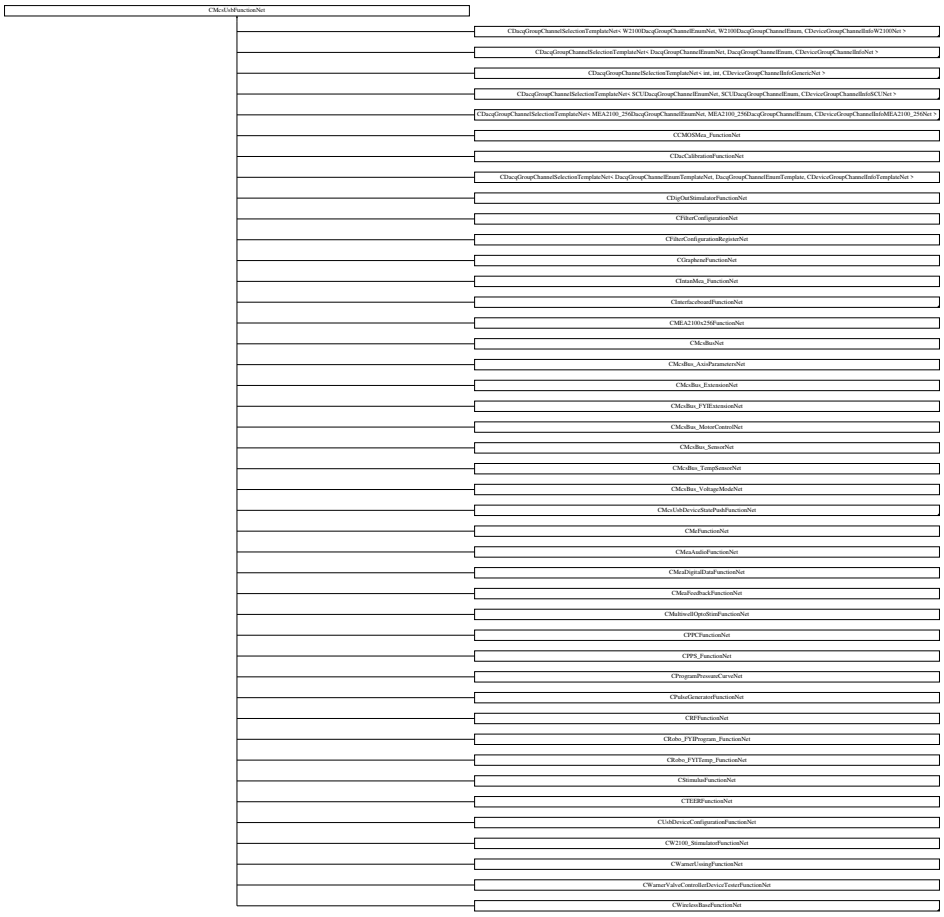
```
const uint32_t FX3MCSDataIFB2ImageOffset = 0xC [static]
```

11.52.3.5 FX3MCSDataVersionOffset

```
const uint32_t FX3MCSDataVersionOffset = 0x8 [static]
```

11.53 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

11.53.1 Constructor & Destructor Documentation

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

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

11.53.1.3 "!CMcsUsbFunctionNet() !CMcsUsbFunctionNet ( )

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

11.53.2 Member Function Documentation

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

11.53.3 Member Data Documentation

11.53.3.1 m_pMcsUsb CMcsUsbNet ^ m_pMcsUsb [protected]

11.53.3.2 m_pMcsUsbFunction CMcsUsbFunction* m_pMcsUsbFunction [protected]

11.54 CMcsUsbFunctionPointerContainer Class Reference

11.55 CMcsUsbListEntryNet Class Reference

McsUsbListEntryNet identifies a connected device.

Public Member Functions

- ~CMcsUsbListEntryNet ()
- virtual bool Equals (Object^ obj) override

  Checks weather two CMcsUsbListEntryNet represent the same USB device.
- void SetStringFormat (String ^ format)

  Specify the text the CMcsUsbListEntryNet.ToString() function should return. The special code N expands to the device name and S expands to the serial number of the device.
- virtual String ^ ToString () override

Static Public Member Functions

- static CMcsUsbListEntryNet ^ GetEntry ()

  Returns one CMcsUsbListEntryNet from the list of USB Devices connected to the computer.
- static CMcsUsbListEntryNet ^ GetEntry (DeviceEnumNet McsUsbDevice)

  Returns one CMcsUsbListEntryNet from the list of USB Devices connected to the computer.
- static CMcsUsbListEntryNet ^ GetEntry (DeviceEnumNet McsUsbDevice, unsigned int index)

  Returns one CMcsUsbListEntryNet from the list of USB Devices connected to the computer.
- static unsigned int GetEntryCount ()

  Returns the number of devices connected to the computer.
- static unsigned int GetEntryCount (DeviceEnumNet McsUsbDevice)

  Returns the number of devices connected to the computer.

Properties

- String^ Manufacturer `[get]`

  The Manufacturer ID of the device represented by this CMcsUsbListEntryNet.
- String^ Product `[get]`

  The Product ID of the device represented by this CMcsUsbListEntryNet.
- String^ DeviceName `[get]`

  The device name of the device represented by this CMcsUsbListEntryNet.
- String^ SerialNumber `[get]`

  The serial number of the device represented by this CMcsUsbListEntryNet.
- String^ HwVersion `[get]`

  The hardware revision of the device represented by this CMcsUsbListEntryNet.
- DeviceIdNet^ DeviceId `[get]`

11.55.1 Detailed Description

McsUsbListEntryNet identifies a connected device.

11.55.2 Constructor & Destructor Documentation

11.55.2.1 ~CMcsUsbListEntryNet() `~CMcsUsbListEntryNet ( )`

11.55.3 Member Function Documentation

11.55.3.1 Equals() `virtual bool Equals ( Object^ obj ) [override], [virtual]`

Checks weather two CMcsUsbListEntryNet represent the same USB device.

Parameters

obj	The CMcsUsbListEntryNet to compare with.

11.55.3.2 GetEntry() [1/3] static CMcsUsbListEntryNet ^ GetEntry ( ) [static]

Returns one CMcsUsbListEntryNet from the list of USB Devices connected to the computer.

Returns

A CMcsUsbListEntryNet to be used to connect to the device.

11.55.3.3 GetEntry() [2/3] static CMcsUsbListEntryNet ^ GetEntry (DeviceEnumNet *McsUsbDevice*) [static]

Returns one CMcsUsbListEntryNet from the list of USB Devices connected to the computer.

Parameters

McsUsbDevice	Specifies the type of devices to look for.

Returns

A CMcsUsbListEntryNet to be used to connect to the device.

11.55.3.4 GetEntry() [3/3] static CMcsUsbListEntryNet ^ GetEntry (DeviceEnumNet *McsUsbDevice*, unsigned int *index*) [static]

Returns one CMcsUsbListEntryNet from the list of USB Devices connected to the computer.

Parameters

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.

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

Returns the number of devices connected to the computer.

Returns

The number of devices.

11.55.3.6 GetEntryCount() [2/2] static unsigned int GetEntryCount (DeviceEnumNet *McsUsbDevice*) [static]

Returns the number of devices connected to the computer.

Parameters

McsUsbDevice	Specifies the type of devices to look for.

Returns

The number of devices.

11.55.3.7 SetStringFormat() void SetStringFormat (String ^ *format*)

Specify the text the CMcsUsbListEntryNet.ToString() function should return. The special code N expands to the device name and S expands to the serial number of the device.

Parameters

format	A String containing the format template. Default is "%N (%S)".

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

11.55.4 Property Documentation

11.55.4.1 DeviceId `DeviceIdNet^ DeviceId [get]`

11.55.4.2 DeviceName `String^ DeviceName [get]`

The device name of the device represented by this CMcsUsbListEntryNet.

11.55.4.3 HwVersion `String^ HwVersion [get]`

The hardware revision of the device represented by this CMcsUsbListEntryNet.

11.55.4.4 Manufacturer `String^ Manufacturer [get]`

The Manufacturer ID of the device represented by this CMcsUsbListEntryNet.

11.55.4.5 Product `String^ Product [get]`

The Product ID of the device represented by this CMcsUsbListEntryNet.

11.55.4.6 SerialNumber `String^ SerialNumber [get]`

The serial number of the device represented by this CMcsUsbListEntryNet.

11.56 CMcsUsbListNet Class Reference

Class to handle a list of connected MCS USB devices.

Public Member Functions

- CMcsUsbListNet (DeviceEnumNet McsUsbDevice)

  Initializes a new instance of CMcsUsbListNet class.
- CMcsUsbListNet (array< DeviceIdNet^ >^ 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

11.56.1 Detailed Description

Class to handle a list of connected MCS USB devices.

11.56.2 Constructor & Destructor Documentation

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

Initializes a new instance of CMcsUsbListNet class.

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

Initializes a new instance of CMcsUsbListNet class.

11.56.2.3 ~CMcsUsbListNet() ~CMcsUsbListNet ( )

Destructor: called by Dispose()

11.56.2.4 "!CMcsUsbListNet() !CMcsUsbListNet ( )

Finalizer: called by GC before collecting

11.56.3 Member Function Documentation

11.56.3.1 GetNumberOfDevices() uint32_t GetNumberOfDevices ( )

Gets the number of devices currently in the list.

Returns

The number of devices currently in the list.

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

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

11.56.3.3 GetUsbListEntry() CMcsUsbListEntryNet ^ GetUsbListEntry (unsigned int *index*)

Returns one CMcsUsbListEntryNet from the list of USB Devices connected to the computer.

Parameters

index	number of the entry to use.

11.56.3.4 IsDeviceTypeOf() `bool IsDeviceTypeOf (`

```
CMcsUsbListEntryNet^ entry,
DeviceEnumNet McsUsbDevice )
```

11.56.3.5 SetStringFormat() `void SetStringFormat (`

```
String ^ format )
```

Specify the text the CMcsUsbListEntryNet.ToString() function should return. The special code N expands to the device name and S expands to the serial number of the device.

Parameters

format	A String containing the format template. Default is "%N (%S)".

11.56.4 Property Documentation

11.56.4.1 Count `uint32_t Count [get]`

Gets the number of devices currently in the list.

11.56.5 Event Documentation

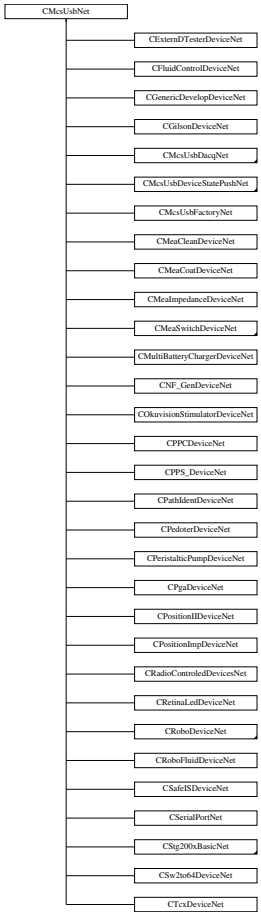
11.56.5.1 DeviceArrival `OnDeviceArrivalRemoval^ DeviceArrival`

11.56.5.2 DeviceRemoval `OnDeviceArrivalRemoval^ DeviceRemoval`

11.57 CMcsUsbNet Class Reference

Base class to handle MCS USB devices. All device classes are derived from this class. Functionality that is provided by all MCS devices is handled by this class.

Inheritance diagram for CMcsUsbNet:



Public Member Functions

- CMcsUsbNet ()

  Initializes a new instance of the base class to handle MCS USB devices.
- CMcsUsbNet (McsBusTypeEnumNet bustype)

  Initializes a new instance of the base class to handle MCS USB devices.
- virtual ~CMcsUsbNet ()
- !CMcsUsbNet ()
- DeviceEnumNet GetDeviceEnum ()
- virtual uint32_t Connect ()

  Opens a connection to the device.
- virtual uint32_t Connect (unsigned int LockMask)

  Opens a connection to the device.
- virtual uint32_t Connect (CMcsUsbListEntryNet^ entry)

  Opens a connection to the device.
- virtual uint32_t Connect (CMcsUsbListEntryNet^ entry, unsigned int LockMask)

  Opens a connection to the device.

- virtual uint32_t GetStatus ([System::Runtime::InteropServices::Out]uint32_t% iStatus)
- virtual bool IsConnected ()

  Check if a device is Connected.
- virtual void Disconnect ()

  Disconnect from a device.
- CMcsUsbListEntryNet ^ GetUsbListEntry ()
- virtual String ^ GetSerialNumber ()

  Query the Serial Number of the device.
- DriverVersionNet ^ GetVersion ()
- DriverVersionNet ^ GetVersion (CFirmwareDestinationNet dest)
- DeviceIdNet ^ GetDeviceId ()
- uint32_t GetIdent ([System::Runtime::InteropServices::Out]String^% Answer)
- void MultibootSelectImage (unsigned int sector)

  Select the multiboot image specified by "sector" (range: 0..2) for IFB FPGA.
- String ^ 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]`

11.57.1 Detailed Description

Base class to handle MCS USB devices. All device classes are derived from this class. Functionality that is provided by all MCS devices is handled by this class.

11.57.2 Constructor & Destructor Documentation

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

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

11.57.2.2 CMcsUsbNet() [2/2] CMcsUsbNet (McsBusTypeEnumNet *bustype*)

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

Parameters

bustype	Type of device to use, either USB or PCI.

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

11.57.2.4 "!CMcsUsbNet() !CMcsUsbNet ( )

11.57.3 Member Function Documentation

11.57.3.1 AddSoftwareKey() void AddSoftwareKey (String^ *key*)

11.57.3.2 AssociateToThis() void AssociateToThis (CMcsUsbNet^ *device*)

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

Opens a connection to the device.

Returns

Error Status. 0 on success.

11.57.3.4 Connect() [2/4] virtual uint32_t Connect (CMcsUsbListEntryNet^ *entry*) [virtual]

Opens a connection to the device.

Parameters

entry	The Device List Entry for the device to be connected.

Returns

Error Status. 0 on success.

11.57.3.5 Connect() [3/4] virtual uint32_t Connect (CMcsUsbListEntryNet^ *entry*, unsigned int *LockMask*) [virtual]

Opens a connection to the device.

Parameters

entry	The Device List Entry for the device to be connected.
LockMask	The Lock Mask for this connection.

Returns

Error Status. 0 on success.

11.57.3.6 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.

11.57.3.7 CyclePort() `uint32_t CyclePort ( )`

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

Disconnect from a device.

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

11.57.3.10 EnableExceptions() `void EnableExceptions ( bool enable )`

Enables or Disables Exceptions for calls to McsUsb Devices. If Exceptions are disabled, the return value of a command can be queries with the GetStatusOfLastCommand call instead.

Parameters

enable	True to enable Exceptions, False to disable.

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

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

11.57.3.13 EraseEepromRegisterPreconfig() [3/3]

```
void EraseEepromRegisterPreconfig (
    uint32_t EEPROMBase,
    uint32_t DMA_reg,
    uint32_t EEPROMSize,
    uint32_t EepromStartAddress )
```

11.57.3.14 GetConfiguration()

```
uint8_t GetConfiguration ( )
```

11.57.3.15 GetDeviceCannotStallOutRequests()

```
bool GetDeviceCannotStallOutRequests ( )
```

11.57.3.16 GetDeviceCapableSpeed()

```
McsUsbSpeedEnumNet GetDeviceCapableSpeed ( )
```

11.57.3.17 GetDeviceEnum()

```
DeviceEnumNet GetDeviceEnum ( )
```

11.57.3.18 GetDeviceId()

```
DeviceIdNet ^ GetDeviceId ( )
```

11.57.3.19 GetDeviceRootHubVendorEnum()

```
UsbVendorIdEnumNet GetDeviceRootHubVendorEnum ( )
```

Gets the Vendor ID of the USB root hub the device is connected to.

Returns

An enum which enumerates the PCI Vendor ID.

11.57.3.20 GetDeviceRootHubVendorID()

```
uint32_t GetDeviceRootHubVendorID ( )
```

Gets the Vendor ID of the USB root hub the device is connected to.

Returns

The PCI Vendor ID, 0x8086 for Intel, 0x1912 for Renesas, 0x1b21 for ASMedia.

11.57.3.21 GetDeviceRootHubVendorName() `String ^ GetDeviceRootHubVendorName ( )`

Gets the Vendor Name of the USB root hub the device is connected to.

Returns

The PCI Vendor Name, either "Intel", "Renesas", "ASMedia" or "unknown".

11.57.3.22 GetDeviceSpeed() `McsUsbSpeedEnumNet GetDeviceSpeed ( )`

Query the Connection Speed of the device.

Returns

0 for Low-Speed, 1 for Full-Speed, 2 for High-Speed and 3 for SuperSpeed.

11.57.3.23 GetErrorText() `static String ^ GetErrorText ( unsigned int Status ) [static]`

Gets the error text string that belongs to a status number.

Parameters

Status	The status number you want the text for.

Returns

The error text string that belongs to the status number.

11.57.3.24 GetFirmwareVersion() `unsigned int GetFirmwareVersion ( CFirmwareDestinationNet destination )`

Gets the firmware version for the destination.

Parameters

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.

11.57.3.25 GetHardwareRevision() `String ^ GetHardwareRevision ( )`

11.57.3.26 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

11.57.3.27 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

11.57.3.28 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

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

11.57.3.30 GetLastUSBError() unsigned int GetLastUSBError ( )

11.57.3.31 GetMea21UsbPort() uint32_t GetMea21UsbPort ( )

Gets the USB port if an IFB that is used by this connection

Returns

number of used port; range: 0..1

11.57.3.32 GetNumConfigurations() uint8_t GetNumConfigurations ( )

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

Query the Serial Number of the device.

Returns

The Serial Number.

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

11.57.3.35 GetSoftwareKeyString() [1/2] String ^ GetSoftwareKeyString (SoftwareKeyProgrammIdsNet::ProgrammIdsNet *ProgrammID*, uint8_t *majorversion*)

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

11.57.3.37 GetStatus() virtual uint32_t GetStatus ([System::Runtime::InteropServices::Out] uint32_t% *iStatus*) [virtual]

11.57.3.38 GetStatusOfLastCommand() uint32_t GetStatusOfLastCommand ( )

Gets the status of the last call to the McsUsb Library.

Returns

The Error Status of the last McsUsb command. 0 on success.

11.57.3.39 GetUsbListEntry() CMcsUsbListEntryNet ^ GetUsbListEntry ( )

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

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

11.57.3.42 HasSoftwareKey() [1/2] bool HasSoftwareKey (SoftwareKeyProgrammIdsNet::ProgrammIdsNet *ProgrammID*, uint8_t *majorversion*)

11.57.3.43 HasSoftwareKey() [2/2] bool HasSoftwareKey (uint8_t *ProgrammID*, uint8_t *majorversion*)

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

Check if a device is Connected.

Returns

true if the device is connected.

11.57.3.45 IsDeviceHighSpeed() `bool IsDeviceHighSpeed ( )`

11.57.3.46 IsDeviceHighSpeedCapable() `bool IsDeviceHighSpeedCapable ( )`

11.57.3.47 IsExceptionsEnabled() `bool IsExceptionsEnabled ( )`

11.57.3.48 MultibootGetCypressImageId() `uint32_t MultibootGetCypressImageId ( unsigned int sector )`

Query the multiboot image id of the device located in specified sector (range: 0..9 0..9) of IFB2 Cypress.

Returns

The magic ident code of the image.

11.57.3.49 MultibootGetImageId() `String ^ MultibootGetImageId ( unsigned int sector )`

Query the multiboot image id of the device located in specified sector (range: 0..2 / 0..9) of IFB1 / IFB2 FPGA.

Returns

The magic ident code of the image.

11.57.3.50 MultibootGetSelectedImage() `uint32_t MultibootGetSelectedImage ( )`

Gets sector index of selected FPGA boot image on IFB

Returns

Sector index of image; range: 0..2

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

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

Returns

Throws exception on error.

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

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

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

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

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

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

11.57.3.58 ReadRegisterTimeSlot()

```
unsigned int ReadRegisterTimeSlot (
    unsigned int reg,
    int TimeSlot )
```

11.57.3.59 RemoveSoftwareKey()

```
void RemoveSoftwareKey (
    unsigned int index )
```

11.57.3.60 RescanHeadstage()

```
void RescanHeadstage (
    uint32_t headstage )
```

rescans and activates a headstage

Parameters

in	*headstage*	the headstage number (0 or 1)

11.57.3.61 SetConfiguration()

```
void SetConfiguration (
    uint8_t config )
```

11.57.3.62 SetSoftwareKey()

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

11.57.3.63 ThrowCUsbExceptionNetOnError()

```
void ThrowCUsbExceptionNetOnError (
    uint32_t status )
```

11.57.3.64 TxnGetSerialNumber()

```
unsigned int TxnGetSerialNumber ( )
```

11.57.3.65 TxnSetSerialNumber()

```
void TxnSetSerialNumber (
    unsigned int number )
```

11.57.3.66 TxnTestMemoryReadAndCheck()

```
unsigned int TxnTestMemoryReadAndCheck (
    unsigned short index )
```

11.57.3.67 TxnTestMemoryWrite()

```
unsigned int TxnTestMemoryWrite (
    unsigned short index )
```

11.57.3.68 ValidKey() [1/2]

```
bool ValidKey (
    String^ key,
    [System::Runtime::InteropServices::Out] String^% serial_number )
```

11.57.3.69 ValidKey() [2/2]

```
bool ValidKey (
    String^ key,
    uint8_t ProgrammID,
    uint8_t majorversion,
    [System::Runtime::InteropServices::Out] String^% serial_number )
```

11.57.3.70 WriteEepromRegisterPreconfig() [1/3]

```
void WriteEepromRegisterPreconfig (
    uint32_t EEPROMBase,
    uint32_t DMA_reg,
    uint32_t DMA_value )
```

11.57.3.71 WriteEepromRegisterPreconfig() [2/3]

```
void WriteEepromRegisterPreconfig (
    uint32_t EEPROMBase,
    uint32_t DMA_reg,
    uint32_t DMA_value,
    uint32_t EEPROMSize )
```

11.57.3.72 WriteEepromRegisterPreconfig() [3/3]

```
void WriteEepromRegisterPreconfig (
    uint32_t EEPROMBase,
    uint32_t DMA_reg,
    uint32_t DMA_value,
    uint32_t EEPROMSize,
    uint32_t EepromStartAddress )
```

11.57.3.73 WriteRegister() [1/2]

```
void WriteRegister (
    unsigned int reg,
    array< unsigned int >^ values )
```

11.57.3.74 WriteRegister() [2/2]

```
void WriteRegister (
    unsigned int reg,
    unsigned int value )
```

11.57.3.75 WriteRegister32()

```
void WriteRegister32 (
    unsigned int adr,
    unsigned int value )
```

11.57.3.76 WriteRegisterArray()

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

11.57.3.77 WriteRegisterTimeSlot() [1/2]

```
void WriteRegisterTimeSlot (
    unsigned int reg,
    array< unsigned int >^ values,
    int TimeSlot )
```

11.57.3.78 WriteRegisterTimeSlot() [2/2]

```
void WriteRegisterTimeSlot (
    unsigned int reg,
    unsigned int value,
    int TimeSlot )
```

11.57.3.79 WriteRegisterValue()

```
void WriteRegisterValue (
    unsigned int reg,
    unsigned int value )
```

11.57.4 Member Data Documentation

11.57.4.1 Status_AlreadyConfigured

```
const uint32_t Status_AlreadyConfigured = (0xE0110001L) [static]
```

11.57.4.2 Status_BadStartFrame

```
const uint32_t Status_BadStartFrame = (0xE0100A00L) [static]
```

11.57.4.3 Status_Btstuff

```
const uint32_t Status_Btstuff = (0xE0100002L) [static]
```

11.57.4.4 Status_BufferOverrun

```
const uint32_t Status_BufferOverrun = (0xE010000CL) [static]
```

11.57.4.5 Status_BufferUnderrun

```
const uint32_t Status_BufferUnderrun = (0xE010000DL) [static]
```

11.57.4.6 Status_Canceled

```
const uint32_t Status_Canceled = (0xE0110000L) [static]
```

11.57.4.7 Status_Canceling

```
const uint32_t Status_Canceling = (0xE0120000L) [static]
```

11.57.4.8 Status_ConnectedPipes

```
const uint32_t Status_ConnectedPipes = (0xE01F000AL) [static]
```

11.57.4.9 Status_ControlNotOwned

```
const uint32_t Status_ControlNotOwned = (0xE0100D00L) [static]
```

11.57.4.10 Status_Crc

```
const uint32_t Status_Crc = (0xE0100001L) [static]
```

11.57.4.11 Status_DataOverrun

```
const uint32_t Status_DataOverrun = (0xE0100008L) [static]
```

11.57.4.12 Status_DataToggleMismatch

```
const uint32_t Status_DataToggleMismatch = (0xE0100003L) [static]
```

11.57.4.13 Status_DataUnderrun

```
const uint32_t Status_DataUnderrun = (0xE0100009L) [static]
```

11.57.4.14 Status_DeviceLocked

```
const uint32_t Status_DeviceLocked = (0xE01F0010L) [static]
```

11.57.4.15 Status_DeviceNotFound

```
const uint32_t Status_DeviceNotFound = (0xE01F0003L) [static]
```

11.57.4.16 Status_DeviceRemoved

```
const uint32_t Status_DeviceRemoved = (0xE01F0008L) [static]
```

11.57.4.17 Status_DevNotResponding

```
const uint32_t Status_DevNotResponding = (0xE0100005L) [static]
```

11.57.4.18 Status_EndpointHalted

```
const uint32_t Status_EndpointHalted = (0xE0100030L) [static]
```

11.57.4.19 Status_ErrorBusy

```
const uint32_t Status_ErrorBusy = (0xE0100400L) [static]
```

11.57.4.20 Status_ErrorShortTransfer

```
const uint32_t Status_ErrorShortTransfer = (0xE0100900L) [static]
```

11.57.4.21 Status_Fifo

```
const uint32_t Status_Fifo = (0xE0100010L) [static]
```

11.57.4.22 Status_FrameControlOwned

```
const uint32_t Status_FrameControlOwned = (0xE0100C00L) [static]
```

11.57.4.23 Status_InternalHcError

```
const uint32_t Status_InternalHcError = (0xE0100800L) [static]
```

11.57.4.24 Status_InvalidDeviceHandle

```
const uint32_t Status_InvalidDeviceHandle = (0xE0100013L) [static]
```

11.57.4.25 Status_InvalidHandle

```
const uint32_t Status_InvalidHandle = (0xE0100012L) [static]
```

11.57.4.26 Status_InvalidParameter

```
const uint32_t Status_InvalidParameter = (0xE0100300L) [static]
```

11.57.4.27 Status_InvalidPipeHandle

```
const uint32_t Status_InvalidPipeHandle = (0xE0100600L) [static]
```

11.57.4.28 Status_InvalidUrbFunction

```
const uint32_t Status_InvalidUrbFunction = (0xE0100200L) [static]
```

11.57.4.29 Status_IoPending

```
const uint32_t Status_IoPending = (0xE01F0006L) [static]
```

11.57.4.30 Status_IoTimeout

```
const uint32_t Status_IoTimeout = (0xE01F0007L) [static]
```

11.57.4.31 Status_IsochRequestFailed

```
const uint32_t Status_IsochRequestFailed = (0xE0100B00L) [static]
```

11.57.4.32 Status_LastUsbErrorMismatch

```
const uint32_t Status_LastUsbErrorMismatch = (0xE01↩
F0022L) [static]
```

11.57.4.33 Status_NoBandwidth

```
const uint32_t Status_NoBandwidth = (0xE0100700L) [static]
```

11.57.4.34 Status_NoMemory

```
const uint32_t Status_NoMemory = (0xE0100100L) [static]
```

11.57.4.35 Status_NoSuchDevice

```
const uint32_t Status_NoSuchDevice = (0xE01F0002L) [static]
```

11.57.4.36 Status_NotAccessed

```
const uint32_t Status_NotAccessed = (0xE010000FL) [static]
```

11.57.4.37 Status_NotSupported

```
const uint32_t Status_NotSupported = (0xE01F0005L) [static]
```

11.57.4.38 Status_PidCheckFailure

```
const uint32_t Status_PidCheckFailure = (0xE0100006L) [static]
```

11.57.4.39 Status_PipeNotLinked

```
const uint32_t Status_PipeNotLinked = (0xE01F0009L) [static]
```

11.57.4.40 Status_RequestFailed

```
const uint32_t Status_RequestFailed = (0xE0100500L) [static]
```

11.57.4.41 Status_RequestMutexFailed

```
const uint32_t Status_RequestMutexFailed = (0xE01F0021L)
[static]
```

11.57.4.42 Status_RequestMutexTimeout

```
const uint32_t Status_RequestMutexTimeout = (0xE01↩
F0020L) [static]
```

11.57.4.43 Status_Stall

```
const uint32_t Status_Stall = (0xE0100004L) [static]
```

11.57.4.44 Status_Unconfigured

```
const uint32_t Status_Unconfigured = (0xE0110002L) [static]
```

11.57.4.45 Status_UnexpectedPid

```
const uint32_t Status_UnexpectedPid = (0xE0100007L) [static]
```

11.57.4.46 WPAError_ScanningIsPending

```
const uint32_t WPAError_ScanningIsPending = ( (0x↩
A0220000L) | 0x0036 ) [static]
```

11.57.5 Property Documentation

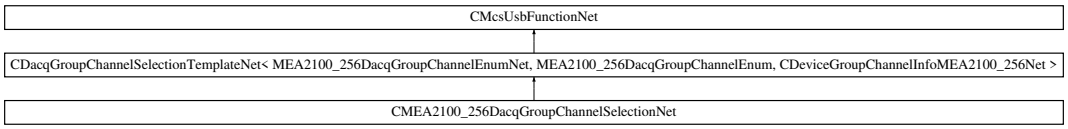
11.57.5.1 SerialNumber

```
virtual String^ SerialNumber [get]
```

11.58 CMcsUsbPointerContainer Class Reference

11.59 CMEA2100_256DacqGroupChannelSelectionNet Class Reference

Inheritance diagram for CMEA2100_256DacqGroupChannelSelectionNet:



Public Member Functions

- CMEA2100_256DacqGroupChannelSelectionNet (CMcsUsbNet^ mcsusb)

Additional Inherited Members

11.59.1 Constructor & Destructor Documentation

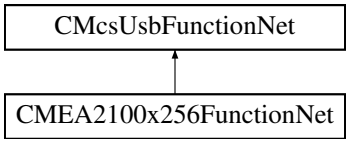
11.59.1.1 CMEA2100_256DacqGroupChannelSelectionNet()

```
CMEA2100_256DacqGroupChannelSelectionNet
(
        CMcsUsbNet^ mcsusb )
```

11.60 CMEA2100x256FunctionNet Class Reference

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

Inheritance diagram for CMEA2100x256FunctionNet:



Public Member Functions

- CMEA2100x256FunctionNet (CMcsUsbNet^ 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

11.60.1 Detailed Description

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

11.60.2 Constructor & Destructor Documentation

11.60.2.1 CMEA2100x256FunctionNet() [1/2]

```
CMEA2100x256FunctionNet (
    CMcsUsbNet^ mcsusb,
    CMcsUsbFunctionPointerContainer^ pMEA2100x256FunctionPointerContainer )
```

Initializes a new instance of the CMEA2100x256FunctionNet class.

11.60.2.2 CMEA2100x256FunctionNet() [2/2] CMEA2100x256FunctionNet (CMcsUsbNet^ *mcsusb*)

11.60.2.3 ~CMEA2100x256FunctionNet() virtual ~CMEA2100x256FunctionNet ( ) [virtual]

11.60.2.4 "!CMEA2100x256FunctionNet() !CMEA2100x256FunctionNet ( )

11.60.3 Member Function Documentation

11.60.3.1 GetLayoutConfiguration() StimulationLayoutConfigurationEnumNet GetLayoutConfiguration ( )

Gets the stimulation layout configuration. Can be single well, 6-well or 9-well. The number of DAC channels available per well is Mcs::Usb::CStg200xBasicNet::GetNumberOfAnalogChannels divided by Mcs::Usb::CStg200xBasicNet::GetNumberOfStimulationSourcesPerElectrode.

Returns

The currently active stimulation layout configuration.

11.60.3.2 SetLayoutConfiguration() void SetLayoutConfiguration (StimulationLayoutConfigurationEnumNet *LayoutConfiguration*)

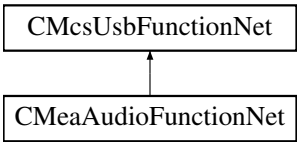
Sets the stimulation layout configuration. Can be single well, 6-well or 9-well. The number of DAC channels available per well is Mcs::Usb::CStg200xBasicNet::GetNumberOfAnalogChannels divided by Mcs::Usb::CStg200xBasicNet::GetNumberOfStimulationSourcesPerElectrode.

Parameters

LayoutConfiguration	The new stimulation layout configuration.

11.61 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

11.61.1 Constructor & Destructor Documentation

11.61.1.1 CMeaAudioFunctionNet() [1/2]

```
CMeaAudioFunctionNet (
    CMcsUsbNet^ mcsusb,
    CMcsUsbFunctionPointerContainer^ meaAudioFunctionPointerContainer )
```

11.61.1.2 CMeaAudioFunctionNet() [2/2]

```
CMeaAudioFunctionNet (
    CMcsUsbNet^ mcsusb )
```

11.61.2 Member Function Documentation

11.61.2.1 GetAudioChannels() [1/2]

```
virtual uint32_t GetAudioChannels (
    [System::Runtime::InteropServices::Out] array< s_setaudionet^ >^% 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.

11.61.2.2 GetAudioChannels() [2/2]

```
virtual uint32_t GetAudioChannels (
    [System::Runtime::InteropServices::Out] array< s_setaudionet^>^% channels,
    unsigned int virtualDevice ) [virtual]
```

Gets the electrode to monitor and amplification for the audio channels.

Parameters

channels	Struct which contains the electrode (channel) and amplification on return.

Parameters

virtualDevice	Virtual device to use.

Returns

Error Status. 0 on success.

11.61.2.3 GetNumberOfAudioChannels()

```
virtual uint32_t GetNumberOfAudioChannels ( ) [virtual]
```

Gets the number of available audio channels.

Returns

The number of audio channels available, 0 when there are none.

11.61.2.4 SetAudioChannels() [1/2]

```
virtual uint32_t SetAudioChannels (
    array< s_setaudionet^>^ channels ) [virtual]
```

Sets the electrode to monitor and amplification for the audio channels.

Parameters

channels	Struct which defines the electrode (channel) and amplification.

Returns

Error Status. 0 on success.

11.61.2.5 SetAudioChannels() [2/2]

```
virtual uint32_t SetAudioChannels (
    array< s_setaudionet^>^ channels,
    unsigned int virtualDevice ) [virtual]
```

Sets the electrode to monitor and amplification for the audio channels.

Parameters

channels	Struct which defines the electrode (channel) and amplification.

Parameters

virtualDevice	Virtual device to use.

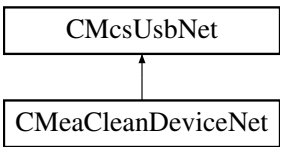
Returns

Error Status. 0 on success.

11.62 CMeaCleanDeviceNet Class Reference

CMeaCleanDeviceNet is the class to access the MEA Clean device.

Inheritance diagram for CMeaCleanDeviceNet:



Public Member Functions

- CMeaCleanDeviceNet ()

  Initializes a new instance of the CMeaCleanDeviceNet class.
- virtual ~CMeaCleanDeviceNet ()
- !CMeaCleanDeviceNet ()
- void Start ()

  Starts a MEA Clean run.
- void Stop ()

  Stops a MEA Clean run.
- void SetSlope (uint32_t voltageSlope)

  Sets the voltage slope.
- void SetCycles (uint32_t cycles)

  Sets the number of cycles.
- void SetMinVoltage (int32_t voltageMin)

  Sets the lower voltage level.
- void SetMaxVoltage (int32_t voltageMax)

  Sets the upper voltage level.
- bool IsRunning ()

  Gets if the MEA Clean device is running.
- uint32_t GetSlope ()

  Gets the voltage slope.
- uint32_t GetCycles ()

  Gets the number of cycles.
- int32_t GetMinVoltage ()

  Gets the lower voltage level.
- int32_t GetMaxVoltage ()

  Gets the upper voltage level
- int32_t GetOutputVoltage ()

  Gets the output voltage.
- int32_t GetCycle ()

  Gets the current cycle.

Additional Inherited Members

11.62.1 Detailed Description

CMeaCleanDeviceNet is the class to access the MEA Clean device.

11.62.2 Constructor & Destructor Documentation

11.62.2.1 CMeaCleanDeviceNet() `CMeaCleanDeviceNet ( )`

Initializes a new instance of the CMeaCleanDeviceNet class.

11.62.2.2 ~CMeaCleanDeviceNet() `virtual ~CMeaCleanDeviceNet ( ) [virtual]`

11.62.2.3 "!CMeaCleanDeviceNet() `!CMeaCleanDeviceNet ( )`

11.62.3 Member Function Documentation

11.62.3.1 GetCycle() `int32_t GetCycle ( )`

Gets the current cycle.

Returns

The cycle number.

11.62.3.2 GetCycles() `uint32_t GetCycles ( )`

Gets the number of cycles.

Returns

The number of cycles to run for.

11.62.3.3 GetMaxVoltage() `int32_t GetMaxVoltage ( )`

Gets the upper voltage level

Returns

The upper voltage level in mV.

11.62.3.4 GetMinVoltage() `int32_t GetMinVoltage ( )`

Gets the lower voltage level.

Returns

The lower voltage level in mV.

11.62.3.5 GetOutputVoltage() `int32_t GetOutputVoltage ( )`

Gets the output voltage.

Returns

The output voltage in mV.

11.62.3.6 GetSlope() `uint32_t GetSlope ( )`

Gets the voltage slope.

Returns

The voltage slope in mV/s.

11.62.3.7 IsRunning() `bool IsRunning ( )`

Gets if the MEA Clean device is running.

Returns

"true" when a run is in progress, otherwise "false".

11.62.3.8 SetCycles() `void SetCycles ( uint32_t cycles )`

Sets the number of cycles.

Parameters

cycles	The number of cycles to run for (0 .. 99).

11.62.3.9 SetMaxVoltage() `void SetMaxVoltage ( int32_t voltageMax )`

Sets the upper voltage level.

Parameters

voltageMax	The upper voltage level in mV (-1.6 .. 1.6 V).

11.62.3.10 SetMinVoltage() `void SetMinVoltage ( int32_t voltageMin )`

Sets the lower voltage level.

Parameters

voltageMin	The lower voltage level in mV (-1.6 .. 1.6 V).

11.62.3.11 SetSlope() `void SetSlope ( uint32_t voltageSlope )`

Sets the voltage slope.

Parameters

voltageSlope	The voltage slope in mV/s (range 0 .. 60 V/s).

11.62.3.12 Start() `void Start ( )`

Starts a MEA Clean run.

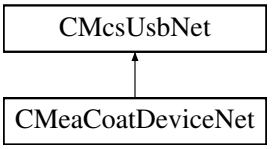
11.62.3.13 Stop() `void Stop ( )`

Stops a MEA Clean run.

11.63 CMeaCoatDeviceNet Class Reference

CMeaCoatDeviceNet is the class to access the MEA Coat device.

Inheritance diagram for CMeaCoatDeviceNet:



Public Member Functions

- CMeaCoatDeviceNet ()

  Initializes a new instance of the CMeaCoatDeviceNet class.
- virtual ~CMeaCoatDeviceNet ()
- !CMeaCoatDeviceNet ()
- void Start ()

  Starts a MEA Coat run.
- void Stop ()

  Stops a MEA Coat run.
- void SetSlope (int32_t currentSlope)

  Sets the current slope.
- void SetDuration (uint32_t duration)

  Sets the duration of a MEA Coat run.
- void SetMaxCurrent (uint32_t currentMax)

  Sets the limit of the current ramp (absolute value).
- void SetOffsetCurrent (int32_t currentOffset)

  Sets the offset of the current.
- bool IsRunning ()

  Gets if the MEA Clean device is running.
- int32_t GetSlope ()

  Gets the current slope.
- uint32_t GetDuration ()

  Gets the duration of a MEA Coat run.
- uint32_t GetMaxCurrent ()

  Gets the limit of the current ramp (absolute value).
- int32_t GetOffsetCurrent ()

  Gets the offset of the current.
- int32_t GetOutputCurrent ()

  Gets the output current.
- int32_t GetTimeInPlateau ()

  Gets the time in the plateau.
- void SetPauseDuration (uint32_t pauseDuration)

  Sets the duration of the pause between MEA Coat pulses.
- uint32_t GetPauseDuration ()

  Gets the duration of the pause between MEA Coat pulses.
- int32_t GetTimeInPause ()

  Gets the time in the pause.
- void SetCycles (uint32_t cycles)

  Sets the number of cycles.
- uint32_t GetCycles ()

  Gets the number of cycles.
- int32_t GetCurrentCycle ()

  Gets the current cycle.

Additional Inherited Members

11.63.1 Detailed Description

CMeaCoatDeviceNet is the class to access the MEA Coat device.

11.63.2 Constructor & Destructor Documentation

11.63.2.1 CMeaCoatDeviceNet() `CMeaCoatDeviceNet ( )`

Initializes a new instance of the CMeaCoatDeviceNet class.

11.63.2.2 ~CMeaCoatDeviceNet() `virtual ~CMeaCoatDeviceNet ( ) [virtual]`

11.63.2.3 "!CMeaCoatDeviceNet() `!CMeaCoatDeviceNet ( )`

11.63.3 Member Function Documentation

11.63.3.1 GetCurrentCycle() `int32_t GetCurrentCycle ( )`

Gets the current cycle.

Returns

The cycle number.

11.63.3.2 GetCycles() `uint32_t GetCycles ( )`

Gets the number of cycles.

Returns

The number of cycles to run for.

11.63.3.3 GetDuration() `uint32_t GetDuration ( )`

Gets the duration of a MEA Coat run.

Returns

The duration in ms.

11.63.3.4 GetMaxCurrent() uint32_t GetMaxCurrent ( )

Gets the limit of the current ramp (absolute value).

Returns

The limit of the current ramp in pA (absolute value).

11.63.3.5 GetOffsetCurrent() int32_t GetOffsetCurrent ( )

Gets the offset of the current.

Returns

The offset of the current in pA.

11.63.3.6 GetOutputCurrent() int32_t GetOutputCurrent ( )

Gets the output current.

Returns

The output current in pA.

11.63.3.7 GetPauseDuration() uint32_t GetPauseDuration ( )

Gets the duration of the pause between MEA Coat pulses.

Returns

The duration in ms.

11.63.3.8 GetSlope() int32_t GetSlope ( )

Gets the current slope.

Returns

The current slope in pA/s.

11.63.3.9 GetTimeInPause() `int32_t GetTimeInPause ( )`

Gets the time in the pause.

Returns

The time in the pause in ms.

11.63.3.10 GetTimeInPlateau() `int32_t GetTimeInPlateau ( )`

Gets the time in the plateau.

Returns

The time in the plateau in ms.

11.63.3.11 IsRunning() `bool IsRunning ( )`

Gets if the MEA Clean device is running.

Returns

"true" when a run is in progress, otherwise "false".

11.63.3.12 SetCycles() `void SetCycles ( uint32_t cycles )`

Sets the number of cycles.

Parameters

cycles	The number of cycles to run for (0 .. 99).

11.63.3.13 SetDuration() `void SetDuration ( uint32_t duration )`

Sets the duration of a MEA Coat run.

Parameters

duration	The duration in ms (range 0 .. 65 s).

11.63.3.14 SetMaxCurrent() `void SetMaxCurrent ( uint32_t currentMax )`

Sets the limit of the current ramp (absolute value).

Parameters

currentMax	The limit of the current ramp in pA (absolute value, 0 .. 18 nA).

11.63.3.15 SetOffsetCurrent() `void SetOffsetCurrent ( int32_t currentOffset )`

Sets the offset of the current.

Parameters

currentOffset	The offset of the current in pA (-10 .. 10 nA).

11.63.3.16 SetPauseDuration() `void SetPauseDuration ( uint32_t pauseDuration )`

Sets the duration of the pause between MEA Coat pulses.

Parameters

pauseDuration	The duration in ms (range 0 .. 65 s).

11.63.3.17 SetSlope() `void SetSlope ( int32_t currentSlope )`

Sets the current slope.

Parameters

currentSlope	The current slope in pA/s (range -65 .. 65 nA/s).

11.63.3.18 Start() `void Start ( )`

Starts a MEA Coat run.

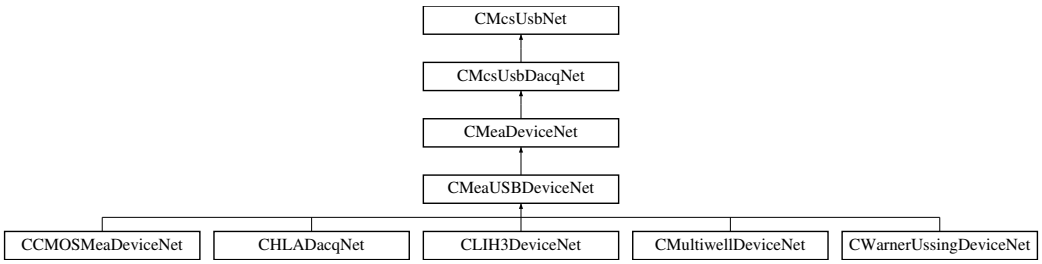
11.63.3.19 Stop() `void Stop ( )`

Stops a MEA Coat run.

11.64 CMeaDeviceNet Class Reference

Base class for MEA data acquisition devices.

Inheritance diagram for CMeaDeviceNet:



Public Member Functions

- CMeaDeviceNet (McsBusTypeEnumNet bustype)

  Initializes a new instance of CMeaDeviceNet class.
- CMeaDeviceNet (McsBusTypeEnumNet bustype, OnChannelData^ 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

11.64.1 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

11.64.2 Constructor & Destructor Documentation

11.64.2.1 CMeaDeviceNet() [1/2]

```
CMeaDeviceNet (
    McsBusTypeEnumNet bustype )
```

Initializes a new instance of CMeaDeviceNet class.

Parameters

bustype	Type of device to use, either USB or PCI.

11.64.2.2 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.

11.64.2.3 ~CMeaDeviceNet()

```
~CMeaDeviceNet ( )
```

11.64.3 Member Function Documentation

11.64.3.1 EnableChecksum()

```
virtual uint32_t EnableChecksum (
    bool enable,
    unsigned int virtualDevice )  [virtual]
```

Enable the checksum data word in the datastream. The checksum is a 32 bit counter and 2x16 bit magic numbers.

Parameters

enable	True to enable, False to disable.
virtualDevice	virtual device to use.

Returns

Error Status. 0 on success.

11.64.3.2 EnableDigitalIn() [1/2] virtual uint32_t EnableDigitalIn (

```
bool enable,
unsigned int virtualDevice ) [virtual]
```

Enable the digital data word in the datastream.

Parameters

enable	True to enable, False to disable.
virtualDevice	virtual device to use.

Returns

Error Status. 0 on success.

11.64.3.3 EnableDigitalIn() [2/2] virtual uint32_t EnableDigitalIn (

```
DigitalDatastreamEnableEnumNet enable,
unsigned int virtualDevice ) [virtual]
```

Enable digital data words in the datastream.

Parameters

enable	True to enable, False to disable.
virtualDevice	virtual device to use.

Returns

Error Status. 0 on success.

11.64.3.4 EnableTimestamp() virtual uint32_t EnableTimestamp (

```
bool enable,
unsigned int virtualDevice ) [virtual]
```

Enable the timestamp data word in the datastream. The timestamp is a 64 bit counter.

Parameters

enable	True to enable, False to disable.
virtualDevice	virtual device to use.

Returns

Error Status. 0 on success.

11.64.3.5 GetAnalogGain() `virtual int32_t GetAnalogGain ( ) [virtual]`

Gets the gain of the analog inputs of the device.

Returns

Gain times 1000, a value of 1000 corresponds to a gain of 1.0.

11.64.3.6 GetEnumerationSpeed() `int32_t GetEnumerationSpeed ( )`

11.64.3.7 GetGain() `virtual int32_t GetGain ( ) [virtual]`

Gets the amplifier gain of the device.

Returns

Gain times 1000, a value of 1000 corresponds to a gain of 1.0.

11.64.3.8 SetDigitalOut() `virtual void SetDigitalOut ( unsigned int digout_value, int pulselength ) [virtual]`

Generate a pulse on the digital output.

Parameters

digout_value	Bitmask to set on the digital out.

Parameters

pulselength	Pulselength in ms.

11.64.3.9 SetNumberOfAnalogChannels()

```
virtual uint32_t SetNumberOfAnalogChannels (
    unsigned int NumberOfChannels_HS1,
    unsigned int NumberOfChannels_HS2,
    unsigned int NumberOfChannels_DSP,
    unsigned int NumberOfChannels_IF,
    unsigned int virtualDevice ) [virtual]
```

Sets the number of analog channels in the datastream for the MEA2100 device.

Parameters

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.

11.64.3.10 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.

11.64.3.11 SetNumberOfChannels() [2/2] virtual uint32_t SetNumberOfChannels (int *NumberOfChannels*, unsigned int *virtualDevice*) [virtual]

Sets the number of analog channels in the datastream.

Parameters

NumberOfChannels	Number of analog channels.
virtualDevice	virtual device to use.

Returns

Error Status. 0 on success.

11.64.3.12 SetTriggerMaskValue() virtual uint32_t SetTriggerMaskValue (unsigned int *mask*, unsigned int *value*, unsigned int *virtualDevice*) [virtual]

Defines a pattern on the digital dataword which will start a trigger when found.

Parameters

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.

11.64.3.13 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.

11.64.4 Property Documentation

11.64.4.1 AnalogGain `virtual int AnalogGain [get]`

The gain of the analog inputs of the device. Value is gain times 1000, a value of 1000 corresponds to a gain of 1.0.

11.64.4.2 Gain `virtual int Gain [get]`

The amplifier gain of the device. Value is gain times 1000, a value of 1000 corresponds to a gain of 1.0.

11.64.4.3 MeaAudioFunctionNet `CMeaAudioFunctionNet^ MeaAudioFunctionNet [get]`

11.64.4.4 MeaDigitalDataFunctionNet `CMeaDigitalDataFunctionNet^ MeaDigitalDataFunctionNet [get]`

11.64.4.5 MeaFeedbackFunctionNet `CMeaFeedbackFunctionNet^ MeaFeedbackFunctionNet [get]`

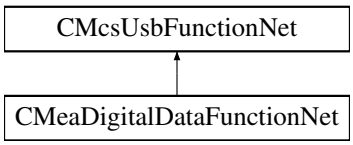
11.64.4.6 MeFunctionNet `CMeFunctionNet^ MeFunctionNet [get]`

11.64.4.7 W2100_FunctionNet `CW2100_FunctionNet^ W2100_FunctionNet [get]`

11.64.4.8 WClassicFunctionNet `CWClassicFunctionNet^ WClassicFunctionNet [get]`

11.65 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

11.65.1 Constructor & Destructor Documentation

11.65.1.1 CMeaDigitalDataFunctionNet() [1/2]

```
CMeaDigitalDataFunctionNet (
    CMcsUsbNet^ mcsusb,
    CMcsUsbFunctionPointerContainer^ meaDigitalFunctionPointerContainer )
```

11.65.1.2 CMeaDigitalDataFunctionNet() [2/2] CMeaDigitalDataFunctionNet (CMcsUsbNet^ *mcsusb*)

11.65.2 Member Function Documentation

11.65.2.1 GetDigitalData() unsigned int GetDigitalData ( )

Get the value of the digital output.

Returns

Value on the digital data register.

11.65.2.2 SetDigitalData() [1/2] void SetDigitalData (unsigned int *bit_number*, bool *value*)

Generate a value on the digital output.

Parameters

bit_number	Bit number to change.

Parameters

value	Bit value.

11.65.2.3 SetDigitalData() [2/2] void SetDigitalData (unsigned int *digital_value*, unsigned int *digital_value_mask*)

Generate a value on the digital output.

Parameters

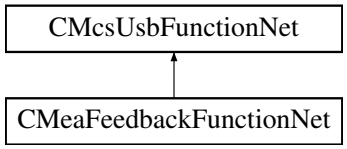
digital_value	Value to set.

Parameters

digital_value_mask	Mask for change.

11.66 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 inmapping)
- 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 FeedbackSetChannelFilter (short channel, char filter)
- void FeedbackSetGlobalChannelFilter (char filter, unsigned short firstchannel, unsigned short lastchannel)
- void FeedbackSetFilterOff ()
- void FeedbackSetNumberOfSpikeDetectors (unsigned short number)
- void FeedbackSetSpikeDetectorThreshold (unsigned short position, unsigned short sourcechannel, unsigned short resultchannel, unsigned short trigger, unsigned short totzeit, int threshold1, int threshold2, short slope)
- void FeedbackSetNumberOfRateCounter (unsigned short number)
- void FeedbackSetRateCounter (unsigned short position, unsigned short sourcechannel, unsigned short resultchannel)
- void FeedbackSetNumberOfRateDetectors (unsigned short number)
- void FeedbackSetRateDetector (unsigned short position, unsigned short resultchannel, unsigned short trigger, unsigned short totzeit, unsigned short pulses, unsigned int duration1, unsigned int duration2)
- void FeedbackSetNumberOfLogics (unsigned short number)
- void FeedbackSetLogic (unsigned short position, array< unsigned short >^ sourcechannel, unsigned short resultchannel, unsigned int lookup)
- void FeedbackSetNumberOfTriggers (unsigned short number)
- void FeedbackSetTrigger (unsigned short position, unsigned short sourcechannel, unsigned short resultchannel, unsigned short trigger, unsigned short totzeit)
- void FeedbackSetAnalogSource (AnalogSourceEnumNet AnalogSource, unsigned int Channels, unsigned int Offset)

Additional Inherited Members

11.66.1 Constructor & Destructor Documentation

11.66.1.1 CMeaFeedbackFunctionNet() [1/2]

```
CMeaFeedbackFunctionNet (
    CMcsUsbNet^ mcsusb,
    CMcsUsbFunctionPointerContainer^ meaFeedbackFunctionNet )
```

11.66.1.2 CMeaFeedbackFunctionNet() [2/2]

```
CMeaFeedbackFunctionNet (
    CMcsUsbNet^ mcsusb )
```

11.66.2 Member Function Documentation

11.66.2.1 FeedbackGetSampleTimerCount()

```
unsigned int FeedbackGetSampleTimerCount (
    [System::Runtime::InteropServices::Out] unsigned int% CurrentCount,
    [System::Runtime::InteropServices::Out] unsigned int% LastKnownCount,
    [System::Runtime::InteropServices::Out] bool% On )
```

11.66.2.2 FeedbackSetAnalogSource()

```
void FeedbackSetAnalogSource (
    AnalogSourceEnumNet AnalogSource,
    unsigned int Channels,
    unsigned int Offset )
```

11.66.2.3 FeedbackSetChannelFilter()

```
void FeedbackSetChannelFilter (
    short channel,
    char filter )
```

11.66.2.4 FeedbackSetDigitalMapping()

```
void FeedbackSetDigitalMapping (
    unsigned short channel,
    unsigned short outmapping,
    unsigned short inmapping )
```

11.66.2.5 FeedbackSetFeedback()

```
void FeedbackSetFeedback (
    unsigned char on,
    unsigned short digoutmask,
    unsigned short diginmask )
```

11.66.2.6 FeedbackSetFilterOff()

```
void FeedbackSetFilterOff ( )
```

11.66.2.7 FeedbackSetFilterParameter()

```
void FeedbackSetFilterParameter (
    unsigned char filter,
    array< short >^ parameters )
```

11.66.2.8 FeedbackSetFilterParameter32()

```
void FeedbackSetFilterParameter32 (
    unsigned char filter,
    array< int >^ parameters )
```

11.66.2.9 FeedbackSetGlobalChannelFilter()

```
void FeedbackSetGlobalChannelFilter (
    char filter,
    unsigned short firstchannel,
    unsigned short lastchannel )
```

11.66.2.10 FeedbackSetIIRFilterParameter()

```
void FeedbackSetIIRFilterParameter (
    unsigned char filter,
    int length,
    array< double >^ parameters )
```

11.66.2.11 FeedbackSetLogic()

```
void FeedbackSetLogic (
    unsigned short position,
    array< unsigned short >^ sourcechannel,
    unsigned short resultchannel,
    unsigned int lookup )
```

11.66.2.12 FeedbackSetMkFilter()

```
void FeedbackSetMkFilter (
    unsigned char filter,
    String^ filtertype,
    double cheb_ribble,
    String^ passtype,
    int order,
    double alpha1,
    double alpha2 )
```

11.66.2.13 FeedbackSetNumberOfLogics()

```
void FeedbackSetNumberOfLogics (
    unsigned short number )
```

11.66.2.14 FeedbackSetNumberOfRateCounter()

```
void FeedbackSetNumberOfRateCounter (
    unsigned short number )
```

11.66.2.15 FeedbackSetNumberOfRateDetectors()

```
void FeedbackSetNumberOfRateDetectors (
    unsigned short number )
```

11.66.2.16 FeedbackSetNumberOfSpikeDetectors()

```
void FeedbackSetNumberOfSpikeDetectors (
    unsigned short number )
```

11.66.2.17 FeedbackSetNumberOfTriggers()

```
void FeedbackSetNumberOfTriggers (
    unsigned short number )
```

11.66.2.18 FeedbackSetRateCounter()

```
void FeedbackSetRateCounter (
    unsigned short position,
    unsigned short sourcechannel,
    unsigned short resultchannel )
```

11.66.2.19 FeedbackSetRateDetector()

```
void FeedbackSetRateDetector (
    unsigned short position,
    unsigned short resultchannel,
    unsigned short trigger,
    unsigned short totzeit,
    unsigned short pulses,
    unsigned int duration1,
    unsigned int duration2 )
```

11.66.2.20 FeedbackSetSpikeDetectorThreshold()

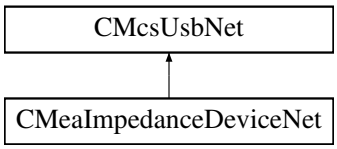
```
void FeedbackSetSpikeDetectorThreshold (
    unsigned short position,
    unsigned short sourcechannel,
    unsigned short resultchannel,
    unsigned short trigger,
    unsigned short totzeit,
    int threshold1,
    int threshold2,
    short slope )
```

11.66.2.21 FeedbackSetTrigger()

```
void FeedbackSetTrigger (
    unsigned short position,
    unsigned short sourcechannel,
    unsigned short resultchannel,
    unsigned short trigger,
    unsigned short totzeit )
```

11.67 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

11.67.1 Constructor & Destructor Documentation

11.67.1.1 CMeaImpedanceDeviceNet()

```
CMeaImpedanceDeviceNet ( )
```

11.67.1.2 ~CMeaImpedanceDeviceNet() ~CMeaImpedanceDeviceNet ( )

11.67.2 Member Function Documentation

11.67.2.1 GetAdapterCode() unsigned short GetAdapterCode ( )

11.67.2.2 GetArraySize() virtual unsigned short GetArraySize ( ) [virtual]

11.67.2.3 GetImpedanceTestFrequency() virtual unsigned int GetImpedanceTestFrequency ( ) [virtual]

11.67.2.4 GetReady() virtual unsigned short GetReady ( ) [virtual]

11.67.2.5 GetResult() virtual array<unsigned short> ^ GetResult ( ) [virtual]

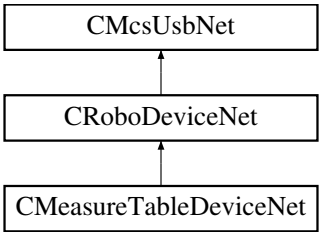
11.67.2.6 SetImpedanceTestFrequency() virtual void SetImpedanceTestFrequency (unsigned int *TestFrequency_Hertz*) [virtual]

11.67.2.7 StartMeasurement() virtual void StartMeasurement (unsigned short *channel*) [virtual]

11.68 CMeasureTableDeviceNet Class Reference

CMeasureTableDeviceNet is the to control the MCS HLA device

Inheritance diagram for CMeasureTableDeviceNet:



Public Member Functions

- CMeasureTableDeviceNet (void)

Properties

- CMcsBus_SensorNet^ Sensor [get]

Additional Inherited Members

11.68.1 Detailed Description

CMeasureTableDeviceNet is the to control the MCS HLA device

11.68.2 Constructor & Destructor Documentation

11.68.2.1 CMeasureTableDeviceNet() CMeasureTableDeviceNet (void)

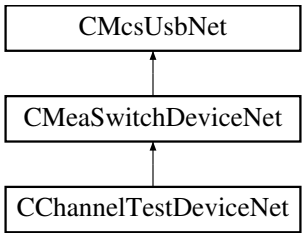
11.68.3 Property Documentation

11.68.3.1 Sensor CMcsBus_SensorNet^ Sensor [get]

11.69 CMeaSwitchDeviceNet Class Reference

The class to control the USB-MEA-Switch.

Inheritance diagram for CMeaSwitchDeviceNet:



Public Member Functions

- CMeaSwitchDeviceNet ()

  Constructor.
- ~CMeaSwitchDeviceNet ()

  Destructor.
- unsigned short GetNumber ()

  Gets the number of boards in the device.
- array< unsigned char > ^ GetPattern ()

  Gets the pattern of the switches that are currently set in the device as char array.
- array< bool > ^ GetPatternBool ()

  Gets the pattern of the switches that are currently set in 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

11.69.1 Detailed Description

The class to control the USB-MEA-Switch.

This class controls the settings of the USB-MEA-Switch. The box has two inputs for signals from a MEA amplifier. Each of the 64 outputs can be connected to one of the MEAs at the same channel.

11.69.2 Constructor & Destructor Documentation

11.69.2.1 CMeaSwitchDeviceNet() `CMeaSwitchDeviceNet ( )`

Constructor.

11.69.2.2 ~CMeaSwitchDeviceNet() `~CMeaSwitchDeviceNet ( )`

Destructor.

11.69.3 Member Function Documentation

11.69.3.1 GetNumber() `unsigned short GetNumber ( )`

Gets the number of boards in the device.

The MEA-Switch are delivered with 64 or 128 channels

11.69.3.2 GetPattern() `array<unsigned char> ^ GetPattern ( )`

Gets the pattern of the switches that are currently set in the device as char array.

11.69.3.3 GetPatternBool() `array<bool> ^ GetPatternBool ( )`

Gets the pattern of the switches that are currently set in he device as bools.

11.69.3.4 SetPattern() `void SetPattern ( array< unsigned char >^ pattern )`

Sets the pattern of switches from a char array.

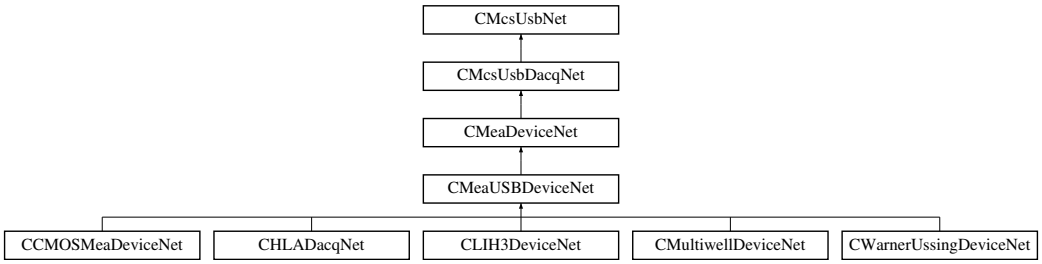
11.69.3.5 SetPatternBool() `void SetPatternBool ( array< bool >^ pattern )`

Sets the pattern of switches from a.

11.70 CMeaUSBDeviceNet Class Reference

Class for data acquisition via ME and MEA USB amplifiers

Inheritance diagram for CMeaUSBDeviceNet:



Public Member Functions

- CMeaUSBDeviceNet (OnChannelData^ channelData, OnError^ error)

  Initializes a new instance of CMeaDeviceNet class.
- CMeaUSBDeviceNet ()

  Initializes a new instance of CMeaDeviceNet class.
- ~CMeaUSBDeviceNet ()

Additional Inherited Members

11.70.1 Detailed Description

Class for data acquisition via ME and MEA USB amplifiers

11.70.2 Constructor & Destructor Documentation

11.70.2.1 CMeaUSBDeviceNet() [1/2] CMeaUSBDeviceNet (OnChannelData^ *channelData*, OnError^ *error*)

Initializes a new instance of CMeaDeviceNet class.

Parameters

channelData	Handler to call when new data is available.

Parameters

error	Handler to call when an error occurs.

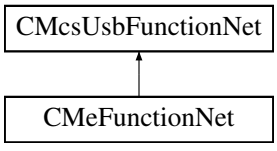
11.70.2.2 CMeaUSBDeviceNet() [2/2] CMeaUSBDeviceNet ( )

Initializes a new instance of CMeaDeviceNet class.

11.70.2.3 ~CMeaUSBDeviceNet() ~CMeaUSBDeviceNet ( )

11.71 CMeFunctionNet Class Reference

Inheritance diagram for CMeFunctionNet:



Public Member Functions

- CMeFunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ meFunctionPointerContainer)

  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

11.71.1 Detailed Description

11.71.2 Constructor & Destructor Documentation

11.71.2.1 CMeFunctionNet() [1/2]

```
CMeFunctionNet (
    CMcsUsbNet^ mcsusb,
    CMcsUsbFunctionPointerContainer^ meFunctionPointerContainer )
```

Initializes a new instance of the CDacCalibrationFunctionNet class.

11.71.2.2 CMeFunctionNet() [2/2]

```
CMeFunctionNet (
    CMcsUsbNet^ mcsusb )
```

11.71.2.3 ~CMeFunctionNet()

```
virtual ~CMeFunctionNet (
    void ) [virtual]
```

11.71.2.4 "!CMeFunctionNet()

```
!CMeFunctionNet (
    void )
```

11.71.3 Member Function Documentation

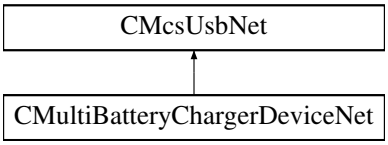
11.71.3.1 SetTransformer()

```
void SetTransformer (
    unsigned int index,
    bool onoff )
```

11.72 CMultiBatteryChargerDeviceNet Class Reference

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

Inheritance diagram for CMultiBatteryChargerDeviceNet:



Public Member Functions

- CMultiBatteryChargerDeviceNet ()

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

11.72.1 Detailed Description

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

11.72.2 Constructor & Destructor Documentation

11.72.2.1 CMultiBatteryChargerDeviceNet() `CMultiBatteryChargerDeviceNet ( )`

Initializes a new instance of the CMultiBatteryChargerDeviceNet class.

11.72.2.2 ~CMultiBatteryChargerDeviceNet() `virtual ~CMultiBatteryChargerDeviceNet ( ) [virtual]`

11.72.2.3 "!CMultiBatteryChargerDeviceNet() `!CMultiBatteryChargerDeviceNet ( )`

11.72.3 Member Function Documentation

11.72.3.1 CapacityTest() `void CapacityTest ( uint32_t NrChannel )`

start capacity test on channel

Parameters

NrChannel	the channel number

11.72.3.2 ChannelReset() `void ChannelReset ( uint32_t NrChannel )`

cancel charging and capacity test functions; check if battery is connected

Parameters

NrChannel	the channel number

11.72.3.3 GetBatteryVoltage() `uint32_t GetBatteryVoltage ( uint32_t NrChannel )`

gets the battery voltage; unit: mV

Parameters

NrChannel	the channel number

Returns

the battery voltage in mV

11.72.3.4 GetChannels() `uint32_t GetChannels ( )`

gets number of channels

Returns

number of channels

11.72.3.5 GetChannelState() `MbcChannelStateEnumNet GetChannelState ( uint32_t NrChannel )`

gets the channel state: IdleNoBattery, IdleChargeFinished, CapacityTestPreCharge, CapacityTestDischarge, StorageCharge, LowCurrentCharge, HighCurrentCharge

Parameters

NrChannel	the channel number

Returns

the current state

11.72.3.6 GetChargeCapacity() `uint32_t GetChargeCapacity ( uint32_t NrChannel )`

gets the charge capacity; unit: µAh

Parameters

NrChannel	the channel number

Returns

the capacity in uAh

11.72.3.7 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

11.72.3.8 GetChargingMode() `MbcChargingModeEnumNet GetChargingMode ( uint32_t NrChannel )`

gets the charging mode: StorageCharge, LowCurrentCharge and HighCurrentCharge

Parameters

NrChannel	the channel number

Returns

the charging mode

11.72.3.9 GetChargingPCoefficient() uint32_t GetChargingPCoefficient ( )

gets the p-coefficient for charging in mA/V / nominal charging current

Returns

the p-coefficient

11.72.3.10 GetDischargeCapacity() uint32_t GetDischargeCapacity (uint32_t *NrChannel*)

gets the discharge capacity; unit: µAh

Parameters

NrChannel	the channel number

Returns

the capacity in uAh

11.72.3.11 GetDischargeCurrent() uint32_t GetDischargeCurrent (uint32_t *NrChannel*)

gets the discharge current; unit: mA

Parameters

NrChannel	the channel number

Returns

the measured discharge current in mA

11.72.3.12 GetDischargeCurrentSetPoint() uint32_t GetDischargeCurrentSetPoint (uint32_t *NrChannel*)

gets the setpoint for the discharge current; unit: mA

Parameters

NrChannel	the channel number

Returns

the discharge current in mA

11.72.3.13 GetFinalDischargeVoltage() `uint32_t GetFinalDischargeVoltage ( uint32_t NrChannel )`

gets the final discharge voltage; unit: mV

Parameters

NrChannel	the channel number

Returns

the battery voltage in mV at the end of discharge

11.72.3.14 GetRatedCapacity() `MbcRatedCapacityEnumNet GetRatedCapacity ( uint32_t NrChannel )`

gets the rated capacity

Parameters

NrChannel	the channel number

Returns

the capacity

11.72.3.15 SetChargingMode() `void SetChargingMode ( uint32_t NrChannel, MbcChargingModeEnumNet NewOperatingMode )`

sets the charging mode: StorageCharge, LowCurrentCharge and HighCurrentCharge

Parameters

NrChannel	the channel number
NewOperatingMode	the charging mode

11.72.3.16 SetChargingPCoefficient() void SetChargingPCoefficient (uint32_t *pCoefficient*)

sets the p-coefficient for charging in mA/V / nominal charging current

Parameters

pCoefficient	the p-coefficient

11.72.3.17 SetDischargeCurrentSetPoint() void SetDischargeCurrentSetPoint (uint32_t *NrChannel*, uint32_t *DischargeCurrent_mA*)

sets the setpoint for the discharge current; unit: mA

Parameters

NrChannel	the channel number
DischargeCurrent_mA	the discharge current in mA

11.72.3.18 SetFinalDischargeVoltage() void SetFinalDischargeVoltage (uint32_t *NrChannel*, uint32_t *FinalDischargeVoltage_mV*)

sets the final discharge voltage; unit: mV

Parameters

NrChannel	the channel number
FinalDischargeVoltage_mV	the battery voltage in mV at the end of discharge

11.72.3.19 SetRatedCapacity() void SetRatedCapacity (uint32_t *NrChannel*, MbcRatedCapacityEnumNet *NewRatedCapacity*)

sets the rated capacity

Parameters

NrChannel	the channel number
NewRatedCapacity	the capacity

11.72.3.20 SetRatedCapacityVolatile() `void SetRatedCapacityVolatile (`

```
uint32_t NrChannel,
MbcRatedCapacityEnumNet NewRatedCapacity )
```

sets the rated capacity (i.e. charge current) without storing it persistently

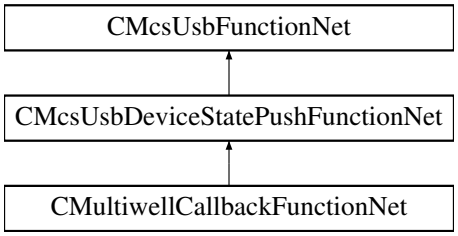
Parameters

NrChannel	the channel number
NewRatedCapacity	the capacity

11.73 CMultiwellCallbackFunctionNet Class Reference

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

Inheritance diagram for CMultiwellCallbackFunctionNet:



Public Member Functions

- delegate void OnGetPlateClampStateByHeadstage (uint32_t Headstage, PlateClampEnumNet plateState)
- CMultiwellCallbackFunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ 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

11.73.1 Detailed Description

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

11.73.2 Constructor & Destructor Documentation

11.73.2.1 CMultiwellCallbackFunctionNet() [1/2]

```
CMultiwellCallbackFunctionNet (
    CMcsUsbNet^ mcsusb,
    CMcsUsbFunctionPointerContainer^ pMultiwellCallbackFunctionPointerContainer )
```

Initializes a new instance of the CMultiwellCallbackFunctionNet class.

11.73.2.2 CMultiwellCallbackFunctionNet() [2/2]

```
CMultiwellCallbackFunctionNet (
    CMcsUsbNet^ mcsusb )
```

11.73.2.3 ~CMultiwellCallbackFunctionNet()

```
virtual ~CMultiwellCallbackFunctionNet ( ) [virtual]
```

11.73.2.4 "!CMultiwellCallbackFunctionNet()

```
!CMultiwellCallbackFunctionNet ( )
```

11.73.3 Member Function Documentation

11.73.3.1 GetPlateClampStateByHeadstage()

```
PlateClampEnumNet GetPlateClampStateByHeadstage (
    uint32_t Headstage )
```

Gets the state of the plate

Parameters

Headstage	The headstage number

Returns

The plate state

11.73.3.2 OnGetPlateClampStateByHeadstage()

```
delegate void OnGetPlateClampStateByHeadstage (
    uint32_t Headstage,
    PlateClampEnumNet plateState )
```

11.73.4 Event Documentation

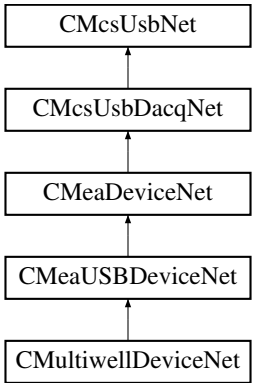
11.73.4.1 GetPlateClampStateByHeadstageEvent `OnGetPlateClampStateByHeadstage^ GetPlateClamp↩StateByHeadstageEvent [add], [remove], [raise]`

Event fires when the plate state for the headstage number has changed

11.74 CMultiwellDeviceNet Class Reference

CMultiwellDeviceNet is the class to access the Multiwell device.

Inheritance diagram for CMultiwellDeviceNet:



Public Member Functions

- CMultiwellDeviceNet ()
  Initializes a new instance of the CMultiwellDeviceNet class.
- virtual ~CMultiwellDeviceNet ()
- !CMultiwellDeviceNet ()
- PlateClampEnumNet GetPlateClampState ()
  Gets the state of the Multiwell plate clamp.
- PlateClampEnumNet GetPlateClampState (uint32_t Headstage)
  Gets the state of the plate
- void OpenPlateClamp ()
  Opens the plate clamp.
- void ClosePlateClamp ()
  Closes the plate clamp.
- void StopPlateClamp ()
  Stops the plate clamp movement.
- uint32_t GetPlateClampLockState ()
  Gets the state of the plate clamp lock.
- void LockPlateClamp ()
  Locks the plate clamp.
- void UnlockPlateClamp ()
  Unlocks the plate clamp.

- MultiwellPlateTypeEnumNet GetPlateType ()

  Gets the plate type.
- MultiwellPlateTypeEnumNet GetPlateType (uint32_t Headstage)

  Gets the plate type.
- void SetPlateType (MultiwellPlateTypeEnumNet plateType)

  Sets the plate type.
- void SetPlateType (uint32_t Headstage, MultiwellPlateTypeEnumNet plateType)

  Sets the plate type.
- void SetPlateMux (uint32_t muxSelection)

  Selects a one quarter of the electrodes on a high density Multiwell plate.
- void SetPlateMux (uint32_t Headstage, uint32_t muxSelection)

  Selects a one quarter of the electrodes on a high density Multiwell plate.
- uint32_t GetPlateMux ()

  Gets the selected quarter of the electrodes on a high density Multiwell plate.
- uint32_t GetPlateMux (uint32_t Headstage)

  Gets the selected quarter of the electrodes on a high density Multiwell plate.
- bool IsPlateTypeValid ()

  Checks whether the plate type is valid, meaning all pins have contact.
- bool IsPlateTypeValid (uint32_t Headstage)

  Checks whether the plate type is valid, meaning all pins have contact.
- void SetPowerMuxPlate (uint32_t Headstage, bool powerOn)

  On the Multiwell Mini device, turn Power to the MUX Plate On or Off.
- bool GetPowerMuxPlate (uint32_t Headstage)

  On the Multiwell Mini device, Query if Power to the MUX Plate is On or Off.

Additional Inherited Members

11.74.1 Detailed Description

CMultiwellDeviceNet is the class to access the Multiwell device.

11.74.2 Constructor & Destructor Documentation

11.74.2.1 CMultiwellDeviceNet() `CMultiwellDeviceNet ( )`

Initializes a new instance of the CMultiwellDeviceNet class.

11.74.2.2 ~CMultiwellDeviceNet() `virtual ~CMultiwellDeviceNet ( ) [virtual]`

11.74.2.3 "!CMultiwellDeviceNet() `!CMultiwellDeviceNet ( )`

11.74.3 Member Function Documentation

11.74.3.1 ClosePlateClamp() `void ClosePlateClamp ( )`

Closes the plate clamp.

11.74.3.2 GetPlateClampLockState() `uint32_t GetPlateClampLockState ( )`

Gets the state of the plate clamp lock.

Returns

the state of the plate lock (unlocked/locked)

11.74.3.3 GetPlateClampState() [1/2] `PlateClampEnumNet GetPlateClampState ( )`

Gets the state of the Multiwell plate clamp.

Returns

the state of the plate clamp (open/closed)

11.74.3.4 GetPlateClampState() [2/2] `PlateClampEnumNet GetPlateClampState ( uint32_t Headstage )`

Gets the state of the plate

Parameters

Headstage	The headstage number

Returns

The plate state

11.74.3.5 GetPlateMux() [1/2] `uint32_t GetPlateMux ( )`

Gets the selected quarter of the electrodes on a high density Multiwell plate.

Returns

the selected quarter

11.74.3.6 GetPlateMux() [2/2] uint32_t GetPlateMux (uint32_t *Headstage*)

Gets the selected quarter of the electrodes on a high density Multiwell plate.

Parameters

Headstage	The headstage to query.

Returns

the selected quarter

11.74.3.7 GetPlateType() [1/2] MultiwellPlateTypeEnumNet GetPlateType ( )

Gets the plate type.

Returns

the plate type

11.74.3.8 GetPlateType() [2/2] MultiwellPlateTypeEnumNet GetPlateType (uint32_t *Headstage*)

Gets the plate type.

Parameters

Headstage	The headstage to query.

Returns

the plate type

11.74.3.9 GetPowerMuxPlate() bool GetPowerMuxPlate (uint32_t *Headstage*)

On the Multiwell Mini device, Query if Power to the MUX Plate is On or Off.

Parameters

Headstage	The headstage to query.

Returns

"true" Power is On, "false" Power is Off

11.74.3.10 IsPlateTypeValid() [1/2] bool IsPlateTypeValid ( )

Checks whether the plate type is valid, meaning all pins have contact.

Returns

"true" when all pins have contact, otherwise "false".

11.74.3.11 IsPlateTypeValid() [2/2] bool IsPlateTypeValid (uint32_t *Headstage*)

Checks whether the plate type is valid, meaning all pins have contact.

Parameters

Headstage	The headstage to query.

Returns

"true" when all pins have contact, otherwise "false".

11.74.3.12 LockPlateClamp() void LockPlateClamp ( )

Locks the plate clamp.

11.74.3.13 OpenPlateClamp() void OpenPlateClamp ( )

Opens the plate clamp.

11.74.3.14 SetPlateMux() [1/2] void SetPlateMux (uint32_t *Headstage*, uint32_t *muxSelection*)

Selects a one quarter of the electrodes on a high density Multiwell plate.

Parameters

Headstage	The headstage to query.
muxSelection	the selected quarter

11.74.3.15 SetPlateMux() [2/2]

```
void SetPlateMux (
    uint32_t muxSelection )
```

Selects a one quarter of the electrodes on a high density Multiwell plate.

Parameters

muxSelection	the selected quarter

11.74.3.16 SetPlateType() [1/2]

```
void SetPlateType (
    MultiwellPlateTypeEnumNet plateType )
```

Sets the plate type.

Parameters

plateType	the plate type

11.74.3.17 SetPlateType() [2/2]

```
void SetPlateType (
    uint32_t Headstage,
    MultiwellPlateTypeEnumNet plateType )
```

Sets the plate type.

Parameters

Headstage	The headstage to query.
plateType	the plate type

11.74.3.18 SetPowerMuxPlate()

```
void SetPowerMuxPlate (
    uint32_t Headstage,
    bool powerOn )
```

On the Multiwell Mini device, turn Power to the MUX Plate On or Off.

Parameters

Headstage	The headstage to query.
powerOn	"true" to turn Power On, "false" to turn Power Off

11.74.3.19 StopPlateClamp() `void StopPlateClamp ( )`

Stops the plate clamp movement.

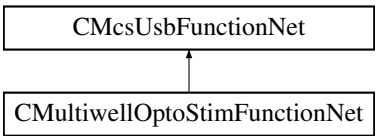
11.74.3.20 UnlockPlateClamp() `void UnlockPlateClamp ( )`

Unlocks the plate clamp.

11.75 CMultiwellOptoStimFunctionNet Class Reference

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

Inheritance diagram for CMultiwellOptoStimFunctionNet:



Public Member Functions

- CMultiwellOptoStimFunctionNet (CMcsUsbNet^ 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

11.75.1 Detailed Description

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

11.75.2 Constructor & Destructor Documentation

11.75.2.1 CMultiwellOptoStimFunctionNet() [1/2]

```
CMultiwellOptoStimFunctionNet (
    CMcsUsbNet^ mcsusb,
    CMcsUsbFunctionPointerContainer^ pMultiwellOptoStimFunctionPointerContainer )
```

Initializes a new instance of the CMultiwellOptoStimFunctionNet class.

11.75.2.2 CMultiwellOptoStimFunctionNet() [2/2]

```
CMultiwellOptoStimFunctionNet (
    CMcsUsbNet^ mcsusb )
```

11.75.2.3 ~CMultiwellOptoStimFunctionNet()

```
virtual ~CMultiwellOptoStimFunctionNet ( ) [virtual]
```

11.75.2.4 "!CMultiwellOptoStimFunctionNet()

```
!CMultiwellOptoStimFunctionNet ( )
```

11.75.3 Member Function Documentation

11.75.3.1 GetAbsMaxCurrentInMicroAmp()

```
uint32_t GetAbsMaxCurrentInMicroAmp (
    uint16_t channel )
```

Parameters

channel	the (analog) channel number

Returns

absolute max. current; unit: uA

11.75.3.2 GetColorRgb() `uint32_t GetColorRgb ( uint16_t channel )`

Parameters

channel	the (analog) channel number

Returns

RGB-value of LED color

11.75.3.3 GetColorStr() `String ^ GetColorStr ( uint16_t channel )`

Parameters

channel	the (analog) channel number

Returns

LED color as string

11.75.3.4 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

11.75.3.5 GetMaxDutyCycleHighCurrent() `uint32_t GetMaxDutyCycleHighCurrent ( uint16_t channel )`

Parameters

channel	the (analog) channel number

Returns

max. duty cycle at max. current; unit: 100∗%

11.75.3.6 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

11.75.3.7 GetWaveLengthInNanometer() uint32_t GetWaveLengthInNanometer (uint16_t *channel*)

Parameters

channel	the (analog) channel number

Returns

wavelength of this channel's LEDs; unit: nm

11.75.3.8 SetAbsMaxCurrentInMicroAmp() void SetAbsMaxCurrentInMicroAmp (uint16_t *channel*, uint32_t *AbsoluteMaxCurrent_uA*)

Parameters

channel	the (analog) channel number
AbsoluteMaxCurrent_uA	absolute max. current; unit: uA

11.75.3.9 SetColorRgb() void SetColorRgb (uint16_t *channel*, uint32_t *ColorRGB*)

Parameters

channel	the (analog) channel number
ColorRGB	RGB-value of LED color

11.75.3.10 SetColorStr() `void SetColorStr (`

```
uint16_t channel,
String^ ColorString )
```

Parameters

channel	the (analog) channel number
ColorString	LED color as string

11.75.3.11 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

11.75.3.12 SetMaxDutyCycleHighCurrent() `void SetMaxDutyCycleHighCurrent (`

```
uint16_t channel,
uint32_t MaxDutyCycleHighCurrent )
```

Parameters

channel	the (analog) channel number
MaxDutyCycleHighCurrent	max. duty cycle at max. current; unit: 100∗%

11.75.3.13 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

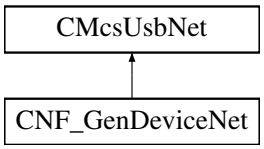
11.75.3.14 SetWaveLengthInNanometer() void SetWaveLengthInNanometer (uint16_t *channel*, uint32_t *WaveLength_nm*)

Parameters

channel	the (analog) channel number
WaveLength_nm	wavelength of this channel's LEDs; unit: nm

11.76 CNF_GenDeviceNet Class Reference

Inheritance diagram for CNF_GenDeviceNet:



Public Member Functions

- CNF_GenDeviceNet (void)
- ~CNF_GenDeviceNet (void)
- void Set_Values (unsigned int frequency, unsigned int amplitude)

Additional Inherited Members

11.76.1 Constructor & Destructor Documentation

11.76.1.1 CNF_GenDeviceNet() CNF_GenDeviceNet (void)

11.76.1.2 ~CNF_GenDeviceNet() ~CNF_GenDeviceNet (void)

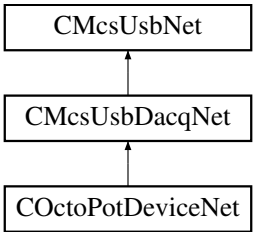
11.76.2 Member Function Documentation

11.76.2.1 Set_Values()

```
void Set_Values (
    unsigned int frequency,
    unsigned int amplitude )
```

11.77 COctoPotDeviceNet Class Reference

Inheritance diagram for COctoPotDeviceNet:



Public Member Functions

- COctoPotDeviceNet (void)
- COctoPotDeviceNet (OnChannelData^ channelData, OnError^ error)
- uint32_t SetOutputRate (uint32_t rate)
- uint32_t SetBathclamp (unsigned int block, bool enable)
- uint32_t SetDacValue (int channel, int value)
- uint32_t SetDacAutoControl (unsigned int channel)
- uint32_t SetPidParameter (unsigned int channel, int const_p, int const_i, int shift_p, int shift_i)
- uint32_t SetRampParameter (unsigned int channel, int start, int min, int max, int slope, int slope2, int pause, unsigned int samples)
- uint32_t RampStart (int channelmap)
- uint32_t SetSineParameter (unsigned int channel, int amplitude)
- uint32_t SineStart (int channelmap)
- uint32_t SetPatternListEntry (unsigned int channel, unsigned int position, unsigned int duration, int value)
- uint32_t PatternListStart (int channelmap)
- uint32_t SetAdcOffset (unsigned int channel, int offset)
- uint32_t SetDacOffset (unsigned int channel, int offset)
- uint32_t ResetAdcOffset (unsigned int channel)
- uint32_t ResetDacOffset (unsigned int channel)
- uint32_t BurnAdcOffset ()
- uint32_t BurnDacOffset ()
- uint32_t GetAdcOffset (unsigned int channel, [System::Runtime::InteropServices::Out] int ^ offset)
- uint32_t GetDacOffset (unsigned int channel, [System::Runtime::InteropServices::Out] int ^ offset)
- uint32_t SetAmplificationSwitch (unsigned int channel, unsigned int state)
- uint32_t SetChannelSwitch (unsigned int channel, unsigned int state)
- uint32_t SetNumberOfChannels (unsigned int NumberOfChannels)
- uint32_t EnableDigitalIn (bool enable)
- uint32_t EnableTimestamp (bool enable)
- uint32_t EnableChecksum (bool enable)

Additional Inherited Members

11.77.1 Constructor & Destructor Documentation

11.77.1.1 COctoPotDeviceNet() [1/2] COctoPotDeviceNet (

```
void )
```

11.77.1.2 COctoPotDeviceNet() [2/2] COctoPotDeviceNet (

```
OnChannelData^ channelData,
OnError^ error )
```

11.77.2 Member Function Documentation

11.77.2.1 BurnAdcOffset() uint32_t BurnAdcOffset ( )

11.77.2.2 BurnDacOffset() uint32_t BurnDacOffset ( )

11.77.2.3 EnableChecksum() uint32_t EnableChecksum (

```
bool enable )
```

11.77.2.4 EnableDigitalIn() uint32_t EnableDigitalIn (

```
bool enable )
```

11.77.2.5 EnableTimestamp() uint32_t EnableTimestamp (

```
bool enable )
```

11.77.2.6 GetAdcOffset() uint32_t GetAdcOffset (

```
unsigned int channel,
[System::Runtime::InteropServices::Out] int ^ offset )
```

11.77.2.7 GetDacOffset() uint32_t GetDacOffset (

```
unsigned int channel,
[System::Runtime::InteropServices::Out] int ^ offset )
```

11.77.2.8 PatternListStart()

```
uint32_t PatternListStart (
    int channelmap )
```

11.77.2.9 RampStart()

```
uint32_t RampStart (
    int channelmap )
```

11.77.2.10 ResetAdcOffset()

```
uint32_t ResetAdcOffset (
    unsigned int channel )
```

11.77.2.11 ResetDacOffset()

```
uint32_t ResetDacOffset (
    unsigned int channel )
```

11.77.2.12 SetAdcOffset()

```
uint32_t SetAdcOffset (
    unsigned int channel,
    int offset )
```

11.77.2.13 SetAmplificationSwitch()

```
uint32_t SetAmplificationSwitch (
    unsigned int channel,
    unsigned int state )
```

11.77.2.14 SetBathclamp()

```
uint32_t SetBathclamp (
    unsigned int block,
    bool enable )
```

11.77.2.15 SetChannelSwitch()

```
uint32_t SetChannelSwitch (
    unsigned int channel,
    unsigned int state )
```

11.77.2.16 SetDacAutoControl()

```
uint32_t SetDacAutoControl (
    unsigned int channel )
```

11.77.2.17 SetDacOffset()

```
uint32_t SetDacOffset (
    unsigned int channel,
    int offset )
```

11.77.2.18 SetDacValue()

```
uint32_t SetDacValue (
    int channel,
    int value )
```

11.77.2.19 SetNumberOfChannels()

```
uint32_t SetNumberOfChannels (
    unsigned int NumberOfChannels )
```

11.77.2.20 SetOutputRate()

```
uint32_t SetOutputRate (
    uint32_t rate )
```

11.77.2.21 SetPatternListEntry()

```
uint32_t SetPatternListEntry (
    unsigned int channel,
    unsigned int position,
    unsigned int duration,
    int value )
```

11.77.2.22 SetPidParameter()

```
uint32_t SetPidParameter (
    unsigned int channel,
    int const_p,
    int const_i,
    int shift_p,
    int shift_i )
```

11.77.2.23 SetRampParameter()

```
uint32_t SetRampParameter (
    unsigned int channel,
    int start,
    int min,
    int max,
    int slope,
    int slope2,
    int pause,
    unsigned int samples )
```

11.77.2.24 SetSineParameter() uint32_t SetSineParameter (

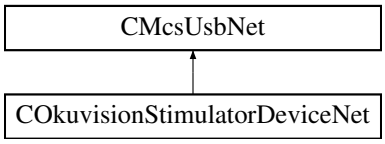
```
unsigned int channel,
int amplitude )
```

11.77.2.25 SineStart() uint32_t SineStart (

```
int channelmap )
```

11.78 COkuvisionStimulatorDeviceNet Class Reference

Inheritance diagram for COkuvisionStimulatorDeviceNet:



Public Member Functions

- COkuvisionStimulatorDeviceNet (void)
- ~COkuvisionStimulatorDeviceNet (void)
- void SetPulseform (int channel, int current, int pulsewidth, int periode, int duration)
- void GetPulseform (int channel, [System::Runtime::InteropServices::Out] int% current, [System::Runtime::InteropServices::Out] int% pulsewidth, [System::Runtime::InteropServices::Out] int% periode, [System::Runtime::InteropServices::Out] int% duration)
- void SetMaxPower (int channel, int power)
- int GetMaxPower (int channel)
- void SetMaxVoltage (int channel, int voltage)
- int GetMaxVoltage (int channel)
- void SetCheckVoltage (int channel, int voltage)
- int GetCheckVoltage (int channel)
- int GetVoltage (int channel)
- void SetDACOffset (int channel, int part, int offset)
- int GetDACOffset (int channel, int part)
- void SetRTC (uint8_t year, uint8_t month, uint8_t day, uint8_t hour, uint8_t minute, uint8_t second)
- void GetRTC ([System::Runtime::InteropServices::Out] uint8_t% year, [System::Runtime::InteropServices::Out] uint8_t% month, [System::Runtime::InteropServices::Out] uint8_t% day, [System::Runtime::InteropServices::Out] uint8_t% hour, [System::Runtime::InteropServices::Out] uint8_t% minute, [System::Runtime::InteropServices::Out] uint8_t% second)
- void SetRTC (DateTime timestamp)
- DateTime GetRTC ()
- void GetStimulatorStatus ([System::Runtime::InteropServices::Out] int% startstop, [System::Runtime::InteropServices::Out] int% last_error, [System::Runtime::InteropServices::Out] int% battery_status)
- void SetCurrentFactor (int channel, int factor)
- int GetCurrentFactor (int channel)

Additional Inherited Members

11.78.1 Constructor & Destructor Documentation

11.78.1.1 COkuvisionStimulatorDeviceNet() COkuvisionStimulatorDeviceNet (void)

11.78.1.2 ~COkuvisionStimulatorDeviceNet() ~COkuvisionStimulatorDeviceNet (void)

11.78.2 Member Function Documentation

11.78.2.1 GetCheckVoltage() int GetCheckVoltage (int *channel*)

11.78.2.2 GetCurrentFactor() int GetCurrentFactor (int *channel*)

11.78.2.3 GetDACOffset() int GetDACOffset (int *channel*, int *part*)

11.78.2.4 GetMaxPower() int GetMaxPower (int *channel*)

11.78.2.5 GetMaxVoltage() int GetMaxVoltage (int *channel*)

11.78.2.6 GetPulseform()

```
void GetPulseform (
    int channel,
    [System::Runtime::InteropServices::Out] int% current,
    [System::Runtime::InteropServices::Out] int% pulsewidth,
    [System::Runtime::InteropServices::Out] int% periode,
    [System::Runtime::InteropServices::Out] int% duration )
```

11.78.2.7 GetRTC() [1/2]

```
DateTime GetRTC ( )
```

11.78.2.8 GetRTC() [2/2]

```
void GetRTC (
    [System::Runtime::InteropServices::Out] uint8_t% year,
    [System::Runtime::InteropServices::Out] uint8_t% month,
    [System::Runtime::InteropServices::Out] uint8_t% day,
    [System::Runtime::InteropServices::Out] uint8_t% hour,
    [System::Runtime::InteropServices::Out] uint8_t% minute,
    [System::Runtime::InteropServices::Out] uint8_t% second )
```

11.78.2.9 GetStimulatorStatus()

```
void GetStimulatorStatus (
    [System::Runtime::InteropServices::Out] int% startstop,
    [System::Runtime::InteropServices::Out] int% last_error,
    [System::Runtime::InteropServices::Out] int% battery_status )
```

11.78.2.10 GetVoltage()

```
int GetVoltage (
    int channel )
```

11.78.2.11 SetCheckVoltage()

```
void SetCheckVoltage (
    int channel,
    int voltage )
```

11.78.2.12 SetCurrentFactor()

```
void SetCurrentFactor (
    int channel,
    int factor )
```

11.78.2.13 SetDACOffset()

```
void SetDACOffset (
    int channel,
    int part,
    int offset )
```

11.78.2.14 SetMaxPower()

```
void SetMaxPower (
    int channel,
    int power )
```

11.78.2.15 SetMaxVoltage()

```
void SetMaxVoltage (
    int channel,
    int voltage )
```

11.78.2.16 SetPulseform()

```
void SetPulseform (
    int channel,
    int current,
    int pulsewidth,
    int periode,
    int duration )
```

11.78.2.17 SetRTC() [1/2]

```
void SetRTC (
    DateTime timestamp )
```

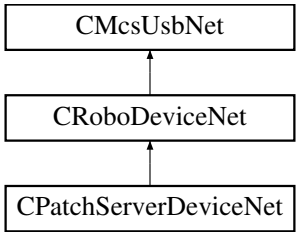
11.78.2.18 SetRTC() [2/2]

```
void SetRTC (
    uint8_t year,
    uint8_t month,
    uint8_t day,
    uint8_t hour,
    uint8_t minute,
    uint8_t second )
```

11.79 CPatchServerDeviceNet Class Reference

CPatchServerDeviceNet is the class to control the MCS PatchServer device

Inheritance diagram for CPatchServerDeviceNet:



Public Member Functions

- CPatchServerDeviceNet (void)

Properties

- CMcsBus_SensorNet^ Sensor [get]

Additional Inherited Members

11.79.1 Detailed Description

CPatchServerDeviceNet is the class to control the MCS PatchServer device

11.79.2 Constructor & Destructor Documentation

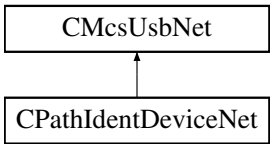
11.79.2.1 CPatchServerDeviceNet() CPatchServerDeviceNet (void)

11.79.3 Property Documentation

11.79.3.1 Sensor CMcsBus_SensorNet^ Sensor [get]

11.80 CPathIdentDeviceNet Class Reference

Inheritance diagram for CPathIdentDeviceNet:



Public Member Functions

- CPathIdentDeviceNet (void)
- ~CPathIdentDeviceNet (void)
- void Set_Values (unsigned int frequency, unsigned int amplitude)
- void Measure ([System::Runtime::InteropServices::Out] unsigned int% phase, [System::Runtime::Interop↩ Services::Out] unsigned int% amplitude)

Additional Inherited Members

11.80.1 Constructor & Destructor Documentation

11.80.1.1 CPathIdentDeviceNet() `CPathIdentDeviceNet (`

```
void )
```

11.80.1.2 ~CPathIdentDeviceNet() `~CPathIdentDeviceNet (`

```
void )
```

11.80.2 Member Function Documentation

11.80.2.1 Measure() `void Measure (`

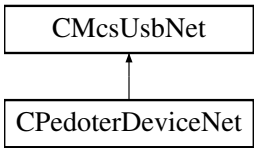
```
[System::Runtime::InteropServices::Out] unsigned int% phase,
[System::Runtime::InteropServices::Out] unsigned int% amplitude )
```

11.80.2.2 Set_Values() `void Set_Values (`

```
unsigned int frequency,
unsigned int amplitude )
```

11.81 CPedoterDeviceNet Class Reference

Inheritance diagram for CPedoterDeviceNet:



Public Member Functions

- CPedoterDeviceNet ()

  Initializes a new instance of the CPedoterDeviceNet class.
- virtual ~CPedoterDeviceNet ()
- !CPedoterDeviceNet ()
- uint32_t GetCommand (uint16_t Argument)

  Get value from the pedoter device
- void SetCommand (uint16_t Argument, uint32_t pData)

  Set value on the pedoter device

Additional Inherited Members

11.81.1 Detailed Description

11.81.2 Constructor & Destructor Documentation

11.81.2.1 CPedoterDeviceNet() `CPedoterDeviceNet ( )`

Initializes a new instance of the CPedoterDeviceNet class.

11.81.2.2 ~CPedoterDeviceNet() `virtual ~CPedoterDeviceNet ( ) [virtual]`

11.81.2.3 "!CPedoterDeviceNet() `!CPedoterDeviceNet ( )`

11.81.3 Member Function Documentation

11.81.3.1 GetCommand() `uint32_t GetCommand ( uint16_t Argument )`

Get value from the pedoter device

Parameters

Argument	argument

Returns

value

11.81.3.2 SetCommand() `void SetCommand ( uint16_t Argument, uint32_t pData )`

Set value on the pedoter device

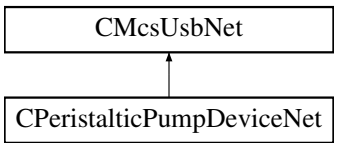
Parameters

Argument	argument
pData	value

11.82 CPeristalticPumpDeviceNet Class Reference

CPeristalticPumpDeviceNet is the class to control a Persistaltic Pump.

Inheritance diagram for CPeristalticPumpDeviceNet:



Public Member Functions

- CPeristalticPumpDeviceNet (void)

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

Properties

- CMcsBus_MotorControlNet^ McsBus_MotorControl `[get]`

Additional Inherited Members

11.82.1 Detailed Description

CPeristalticPumpDeviceNet is the class to control a Persistaltic Pump.

11.82.2 Constructor & Destructor Documentation

11.82.2.1 CPeristalticPumpDeviceNet() `CPeristalticPumpDeviceNet ( void )`

Initialize a new instance of the CPeristalticPumpDeviceNet class.

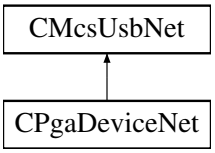
11.82.2.2 ~CPeristalticPumpDeviceNet() ~CPeristalticPumpDeviceNet (void)

11.82.3 Property Documentation

11.82.3.1 McsBus_MotorControl CMcsBus_MotorControlNet^ McsBus_MotorControl [get]

11.83 CPgaDeviceNet Class Reference

Inheritance diagram for CPgaDeviceNet:



Public Member Functions

- CPgaDeviceNet ()
- ~CPgaDeviceNet ()
- uint32_t GetNumFrequencyRanges ([System::Runtime::InteropServices::Out]int% numRanges)
- uint32_t GetFrequencyRange (int rangeIndex, [System::Runtime::InteropServices::Out]int% low, [System::Runtime::InteropServices::Out]int% high, [System::Runtime::InteropServices::Out]int% channels, [System::Runtime::InteropServices::Out]int% gain)
- uint32_t GetNumAmplifications ([System::Runtime::InteropServices::Out]int% number)
- uint32_t GetAmplification (int index, [System::Runtime::InteropServices::Out]int% amplification, [System::Runtime::InteropServices::Out]int% poti1, [System::Runtime::InteropServices::Out]int% poti2)
- uint32_t DefineNumFrequencyRanges (int rnum)
- uint32_t DefineFrequencyRange (int index, int low, int high, int channels, int gain)
- uint32_t DefineNumAmplifications (int number)
- uint32_t DefineAmplification (int index, int amplification, int poti1, int poti2)
- uint32_t SetGain (int channel, int Gain, int poti1, int poti2)
- uint32_t GetGain (int channel, [System::Runtime::InteropServices::Out]int% Gain, [System::Runtime::InteropServices::Out]int% poti1, [System::Runtime::InteropServices::Out]int% poti2)
- uint32_t ApplyGains ()

Additional Inherited Members

11.83.1 Constructor & Destructor Documentation

11.83.1.1 CPgaDeviceNet() CPgaDeviceNet ( )

11.83.1.2 ~CPgaDeviceNet() ~CPgaDeviceNet ( )

11.83.2 Member Function Documentation

11.83.2.1 ApplyGains() uint32_t ApplyGains ( )

11.83.2.2 DefineAmplification() uint32_t DefineAmplification (

```
int index,
int amplification,
int poti1,
int poti2 )
```

11.83.2.3 DefineFrequencyRange() uint32_t DefineFrequencyRange (

```
int index,
int low,
int high,
int channels,
int gain )
```

11.83.2.4 DefineNumAmplifications() uint32_t DefineNumAmplifications (

```
int number )
```

11.83.2.5 DefineNumFrequencyRanges() uint32_t DefineNumFrequencyRanges (

```
int rnum )
```

11.83.2.6 GetAmplification() uint32_t GetAmplification (

```
int index,
[System::Runtime::InteropServices::Out] int% amplification,
[System::Runtime::InteropServices::Out] int% poti1,
[System::Runtime::InteropServices::Out] int% poti2 )
```

11.83.2.7 GetFrequencyRange()

```
uint32_t GetFrequencyRange (
    int rangeIndex,
    [System::Runtime::InteropServices::Out] int% low,
    [System::Runtime::InteropServices::Out] int% high,
    [System::Runtime::InteropServices::Out] int% channels,
    [System::Runtime::InteropServices::Out] int% gain )
```

11.83.2.8 GetGain()

```
uint32_t GetGain (
    int channel,
    [System::Runtime::InteropServices::Out] int% Gain,
    [System::Runtime::InteropServices::Out] int% poti1,
    [System::Runtime::InteropServices::Out] int% poti2 )
```

11.83.2.9 GetNumAmplifications()

```
uint32_t GetNumAmplifications (
    [System::Runtime::InteropServices::Out] int% number )
```

11.83.2.10 GetNumFrequencyRanges()

```
uint32_t GetNumFrequencyRanges (
    [System::Runtime::InteropServices::Out] int% numRanges )
```

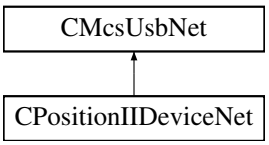
11.83.2.11 SetGain()

```
uint32_t SetGain (
    int channel,
    int Gain,
    int poti1,
    int poti2 )
```

11.84 CPositionIIDeviceNet Class Reference

CPositionIIDeviceNet is the class to control PositionII devices

Inheritance diagram for CPositionIIDeviceNet:



Public Member Functions

- CPositionIIDeviceNet ()

  Initializes a new instance of the CPositionIIDeviceNet class.
- virtual ~CPositionIIDeviceNet ()
- !CPositionIIDeviceNet ()
- uint32_t GetCoilCommunication (uint16_t coil)

  get if the communication to the coil is working
- uint32_t GetOnOff (uint16_t coil)

  get if the coil is switched on/off
- void SwitchOnOff (uint16_t coil, uint32_t on)

  switched the coild on of
- uint32_t GetImplantState (uint16_t coil)

  gets the implantat state
- uint32_t GetImplantCurrentSetpoint (uint16_t coil)

  sets the implant current setpoint
- void SetImplantCurrentSetpoint (uint16_t coil, uint32_t current)

  gets the implant current setpoint
- uint32_t GetPowerStrength (uint16_t coil)

  sets the power for the trigger pulses
- void SetPowerStrength (uint16_t coil, uint32_t power)

  gets the power for the trigger pulses
- uint32_t GetImplantResult (uint16_t coil)

  gets the last result of the implant pulse trigger
- void GetRTC ([System::Runtime::InteropServices::Out]uint8_t% year, [System::Runtime::InteropServices::Out]uint8_t% month, [System::Runtime::InteropServices::Out]uint8_t% day, [System::Runtime::InteropServices::Out]uint8_t% hour, [System::Runtime::InteropServices::Out]uint8_t% minute, [System::Runtime::InteropServices::Out]uint8_t% second)

  Get the RTC
- void SetRTC (uint8_t year, uint8_t month, uint8_t day, uint8_t hour, uint8_t minute, uint8_t second)

  Set the RTC
- uint32_t GetStateDebugData (uint16_t coil)

  get the debug queue state
- void SetStateDebugData (uint16_t coil, uint32_t state)

  clears/starts/stops the debug queue for a certain coil
- void GetDebugData (uint16_t coil, [System::Runtime::InteropServices::Out]uint16_t% index, [System::Runtime::InteropServices::Out]uint16_t% voltage, [System::Runtime::InteropServices::Out]uint16_t% numberofpulses, [System::Runtime::InteropServices::Out]uint16_t% mediantime)

  get the oldest debug entry for a certain coil
- uint32_t GetStateEventData ()

  get the event queue state
- void SetStateEventData (uint32_t state)

  clears/starts/stops the event queue for a certain coil
- void GetEventData ([System::Runtime::InteropServices::Out]uint16_t% index, [System::Runtime::InteropServices::Out]uint8_t% year, [System::Runtime::InteropServices::Out]uint8_t% month, [System::Runtime::InteropServices::Out]uint8_t% day, [System::Runtime::InteropServices::Out]uint8_t% hour, [System::Runtime::InteropServices::Out]uint8_t% minute, [System::Runtime::InteropServices::Out]uint8_t% second, [System::Runtime::InteropServices::Out]uint16_t% coil, [System::Runtime::InteropServices::Out]uint16_t% type, [System::Runtime::InteropServices::Out]uint16_t% value)

  get the oldest event entry

Properties

- CRFFunctionNet^ RFFunction [get]

Additional Inherited Members

11.84.1 Detailed Description

CPositionIIDeviceNet is the class to control PositionII devices

11.84.2 Constructor & Destructor Documentation

11.84.2.1 CPositionIIDeviceNet() CPositionIIDeviceNet ( )

Initializes a new instance of the CPositionIIDeviceNet class.

11.84.2.2 ~CPositionIIDeviceNet() virtual ~CPositionIIDeviceNet ( ) [virtual]

11.84.2.3 "!CPositionIIDeviceNet() !CPositionIIDeviceNet ( )

11.84.3 Member Function Documentation

11.84.3.1 GetCoilCommunication() uint32_t GetCoilCommunication (uint16_t *coil*)

get if the communication to the coil is working

Parameters

coil	the coil

Returns

is communicating

11.84.3.2 GetDebugData() `void GetDebugData (`

```
uint16_t coil,
[System::Runtime::InteropServices::Out] uint16_t% index,
[System::Runtime::InteropServices::Out] uint16_t% voltage,
[System::Runtime::InteropServices::Out] uint16_t% numberofpulses,
[System::Runtime::InteropServices::Out] uint16_t% mediantime )
```

get the oldest debug entry for a certain coil

Parameters

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

11.84.3.3 GetEventData() `void GetEventData (`

```
[System::Runtime::InteropServices::Out] uint16_t% index,
[System::Runtime::InteropServices::Out] uint8_t% year,
[System::Runtime::InteropServices::Out] uint8_t% month,
[System::Runtime::InteropServices::Out] uint8_t% day,
[System::Runtime::InteropServices::Out] uint8_t% hour,
[System::Runtime::InteropServices::Out] uint8_t% minute,
[System::Runtime::InteropServices::Out] uint8_t% second,
[System::Runtime::InteropServices::Out] uint16_t% coil,
[System::Runtime::InteropServices::Out] uint16_t% type,
[System::Runtime::InteropServices::Out] uint16_t% value )
```

get the oldest event entry

Parameters

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

11.84.3.4 GetImplantCurrentSetpoint() `uint32_t GetImplantCurrentSetpoint (`

```
uint16_t coil )
```

sets the implant current setpoint

Parameters

coil	the coil

Returns

the current

11.84.3.5 GetImplantResult() `uint32_t GetImplantResult ( uint16_t coil )`

gets the last result of the implant pulse trigger

Parameters

coil	the coil

Returns

the result

11.84.3.6 GetImplantState() `uint32_t GetImplantState ( uint16_t coil )`

gets the implantat state

Parameters

coil	the coil

Returns

the state

11.84.3.7 GetOnOff() `uint32_t GetOnOff ( uint16_t coil )`

get if the coil is switched on/off

Parameters

coil	the coil

Returns

0 = off, 1 = on

11.84.3.8 GetPowerStrength() uint32_t GetPowerStrength (uint16_t *coil*)

sets the power for the trigger pulses

Parameters

coil	the coil

Returns

the power

11.84.3.9 GetRTC() void GetRTC ([System::Runtime::InteropServices::Out] uint8_t% *year*, [System::Runtime::InteropServices::Out] uint8_t% *month*, [System::Runtime::InteropServices::Out] uint8_t% *day*, [System::Runtime::InteropServices::Out] uint8_t% *hour*, [System::Runtime::InteropServices::Out] uint8_t% *minute*, [System::Runtime::InteropServices::Out] uint8_t% *second*)

Get the RTC

Parameters

year	the year
month	the month
day	the day
hour	the hour
minute	the minute
second	the second

11.84.3.10 GetStateDebugData() uint32_t GetStateDebugData (uint16_t *coil*)

get the debug queue state

Parameters

coil	the coil

Returns

the state

11.84.3.11 GetStateEventData() `uint32_t GetStateEventData ( )`

get the event queue state

Returns

the state

11.84.3.12 SetImplantCurrentSetpoint() `void SetImplantCurrentSetpoint (`

```
uint16_t coil,
uint32_t current )
```

gets the implant current setpoint

Parameters

coil	the coil
current	the current

11.84.3.13 SetPowerStrength() `void SetPowerStrength (`

```
uint16_t coil,
uint32_t power )
```

gets the power for the trigger pulses

Parameters

coil	the coil
power	the power

11.84.3.14 SetRTC() `void SetRTC (`

```
uint8_t year,
uint8_t month,
uint8_t day,
uint8_t hour,
uint8_t minute,
uint8_t second )
```

Set the RTC

Parameters

year	the year
month	the month
day	the day
hour	the hour
minute	the minute
second	the second

11.84.3.15 SetStateDebugData() `void SetStateDebugData (`

```
uint16_t coil,
uint32_t state )
```

clears/starts/stops the debug queue for a certain coil

Parameters

coil	the coil
state	clear/start/stop

11.84.3.16 SetStateEventData() `void SetStateEventData (`

```
uint32_t state )
```

clears/starts/stops the event queue for a certain coil

Parameters

state	clear/start/stop

11.84.3.17 SwitchOnOff() `void SwitchOnOff (`

```
uint16_t coil,
uint32_t on )
```

switched the coild on of

Parameters

coil	the coil
on	0 = off, 1 = on

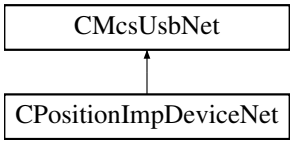
11.84.4 Property Documentation

11.84.4.1 RFFunction `CRFFunctionNet^ RFFunction [get]`

11.85 CPositionImpDeviceNet Class Reference

CPositionImpDeviceNet is the class to access the Position/Imp devices

Inheritance diagram for CPositionImpDeviceNet:



Public Member Functions

- CPositionImpDeviceNet ()

  Initializes a new instance of the CPositionImpDeviceNet class.
- virtual ~CPositionImpDeviceNet ()
- !CPositionImpDeviceNet ()
- void ConnectImp (uint32_t id)

  Connect to a Imp device with a certain ID
- uint32_t ConnectedImp ()

  The ID of the connected Imp device
- int32_t GetRFFrequency ()

  Gets currently used RF frequency
- void SetRFFrequency (int32_t frequency)

  Sets the current RF frequency
- uint32_t GetDeviceList (int32_t index)

  Gets the device list
- void SetDeviceList (int32_t index, uint32_t id)

  Sets the device list
- uint32_t 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

11.85.1 Detailed Description

CPositionImpDeviceNet is the class to access the Position/Imp devices

11.85.2 Constructor & Destructor Documentation

11.85.2.1 CPositionImpDeviceNet() `CPositionImpDeviceNet ( )`

Initializes a new instance of the CPositionImpDeviceNet class.

11.85.2.2 ~CPositionImpDeviceNet() `virtual ~CPositionImpDeviceNet ( ) [virtual]`

11.85.2.3 "!CPositionImpDeviceNet() `!CPositionImpDeviceNet ( )`

11.85.3 Member Function Documentation

11.85.3.1 ConnectedImp() `uint32_t ConnectedImp ( )`

The ID of the connected Imp device

Returns

The ID

11.85.3.2 ConnectImp() `void ConnectImp ( uint32_t id )`

Connect to a Imp device with a certain ID

Parameters

id	The ID

11.85.3.3 GetDeviceList() `uint32_t GetDeviceList ( int32_t index )`

Gets the device list

Parameters

index	the index

Returns

the ID

11.85.3.4 GetImpId() `uint32_t GetImpId ( )`

Gets the ID of the impedance measure device

Returns

the ID

11.85.3.5 GetRFFrequency() `int32_t GetRFFrequency ( )`

Gets currently used RF frequency

Returns

The frequency

11.85.3.6 SetDeviceList() `void SetDeviceList (`

```
int32_t index,
uint32_t id )
```

Sets the device list

Parameters

index	the index
id	the ID

11.85.3.7 SetImpId() `void SetImpId (`

```
uint32_t id )
```

Sets the ID of the impedance measure device

Parameters

id	the ID

11.85.3.8 SetRFFrequency() `void SetRFFrequency (`

`int32_t` *frequency*)

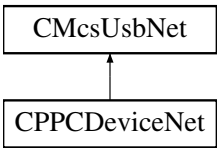
Sets the current RF frequency

Parameters

frequency	The frequency

11.86 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

11.86.1 Constructor & Destructor Documentation

11.86.1.1 CPPCDeviceNet() `CPPCDeviceNet (`

`void`)

11.86.2 Property Documentation

11.86.2.1 McsBus CMcsBusNet^ McsBus [get]

11.86.2.2 McsBus_MotorControl CMcsBus_MotorControlNet^ McsBus_MotorControl [get]

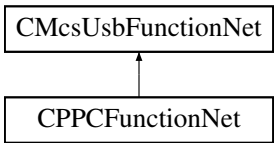
11.86.2.3 McsBus_Sensor CMcsBus_SensorNet^ McsBus_Sensor [get]

11.86.2.4 PPCFunction CPPCFunctionNet^ PPCFunction [get]

11.87 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

11.87.1 Detailed Description

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

11.87.2 Constructor & Destructor Documentation

11.87.2.1 CPPCFunctionNet() [1/2]

```
CPPCFunctionNet (
    CMcsUsbNet^ mcsusb,
    CMcsUsbFunctionPointerContainer^ pPPCFunctionPointerContainer )
```

Initializes a new instance of the CPPCFunctionNet class.

11.87.2.2 CPPCFunctionNet() [2/2] CPPCFunctionNet (CMcsUsbNet^ *mcsusb*)

11.87.2.3 ~CPPCFunctionNet() virtual ~CPPCFunctionNet ( ) [virtual]

11.87.2.4 "!CPPCFunctionNet() !CPPCFunctionNet ( )

11.87.3 Member Function Documentation

11.87.3.1 FirePressurePulse() void FirePressurePulse (int32_t *duration*, int32_t *nextpressure*)

Fire a pressure pulse from the reservoir

Parameters

duration	The pulse duration (valves open)
nextpressure	The next pressure

11.87.3.2 GetAnalogVoltage() uint16_t GetAnalogVoltage (uint16_t *channel*)

Reads the current analog voltage

Parameters

channel	The Channel Number

Returns

The Analog Voltage

11.87.3.3 GetAnalogVoltageRange() void GetAnalogVoltageRange (uint16_t *channel*,

```
[System::Runtime::InteropServices::Out] uint16_t% min_voltage,
[System::Runtime::InteropServices::Out] uint16_t% max_voltage )
```

Reads the Analog Input Voltage Range

Parameters

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

11.87.3.4 GetDigitalIn() `uint16_t GetDigitalIn (`

```
uint16_t channel )
```

Reads the digital input state

Parameters

channel	The Channel Number

Returns

The Digital State

11.87.3.5 GetPressureRange() `void GetPressureRange (`

```
uint16_t channel,
[System::Runtime::InteropServices::Out] int32_t% lower_pressure,
[System::Runtime::InteropServices::Out] int32_t% upper_pressure )
```

Get the pressure range that is used between the analog voltage or the digital states

Parameters

channel	The Channel Number
lower_pressure	The lower border of the pressure range
upper_pressure	The upper border of the pressure range

11.87.3.6 GetPumpModeType() `PP_Pump_Mode_Type_EnumNet GetPumpModeType (`

```
uint16_t channel )
```

Reads the Pump Mode Type.

Parameters

channel	The Channel Number

Returns

The Pump Mode Type.

11.87.3.7 GetPumpSpeedUnit() `int GetPumpSpeedUnit (`

`uint16_t` *channel* `)`

Reads the Pump Speed Unit

Parameters

channel	The Channel Number

Returns

The Speed Unit

11.87.3.8 GetSupplyVoltage() `uint16_t GetSupplyVoltage ( )`

Reads the current supply voltage in mV

Returns

The supply voltage

11.87.3.9 GetValveActive() `int GetValveActive (`

`uint16_t` *valve* `)`

Gets the valve active/inactive state

Parameters

valve	The valve number

Returns

The valve state

11.87.3.10 IsBusy() `void IsBusy (`

`[System::Runtime::InteropServices::Out] int16_t%` *task*`,`
`[System::Runtime::InteropServices::Out] int16_t%` *wait* `)`

Is the PPC busy with a task

Parameters

task	The task state
wait	The wait state

11.87.3.11 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

11.87.3.12 MeasureReservoir() `int32_t MeasureReservoir ( )`

Measures the reservoir pressure

Returns

The pressure

11.87.3.13 SetAnalogVoltageRange() `void SetAnalogVoltageRange (`

```
uint16_t channel,
uint16_t min_voltage,
uint16_t max_voltage )
```

Writes the Analog Input Voltage Range

Parameters

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

11.87.3.14 SetPressureOffset() `void SetPressureOffset ( )`

Sets the pressure offset

11.87.3.15 SetPressureRange() `void SetPressureRange (`

```
uint16_t channel,
int32_t lower_pressure,
int32_t upper_pressure )
```

Get the pressure range that is used between the analog voltage or the digital states

Parameters

channel	The Channel Number
lower_pressure	The lower border of the pressure range
upper_pressure	The upper border of the pressure range

11.87.3.16 SetPumpModeType() `void SetPumpModeType (`

```
uint16_t channel,
PP_Pump_Mode_Type_EnumNet PumpMode )
```

Writes the config string from the device.

Parameters

channel	The Channel Number
PumpMode	The Pump Mode Type.

11.87.3.17 SetPumpSpeedUnit() `void SetPumpSpeedUnit (`

```
uint16_t channel,
int SpeedUnit )
```

Writes the Pump Speed Unit

Parameters

channel	The Channel Number
SpeedUnit	The Speed Unit

11.87.3.18 SetValveActive() `void SetValveActive (`

```
uint16_t valve,
int valveActive )
```

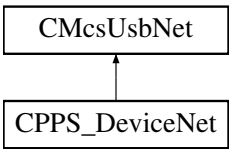
Sets the valve active/inactive state

Parameters

valve	The valve number
valveActive	The valve state

11.88 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

11.88.1 Constructor & Destructor Documentation

11.88.1.1 CPPS_DeviceNet() CPPS_DeviceNet (void)

11.88.2 Property Documentation

11.88.2.1 McsBus CMcsBusNet^ McsBus [get]

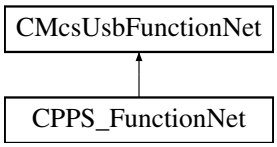
11.88.2.2 McsBus_MotorControl CMcsBus_MotorControlNet^ McsBus_MotorControl [get]

11.88.2.3 McsBus_Sensor CMcsBus_SensorNet^ McsBus_Sensor [get]

11.88.2.4 PPS_Function `CPPS_FunctionNet^ PPS_Function [get]`

11.89 CPPS_FunctionNet Class Reference

Inheritance diagram for CPPS_FunctionNet:



Public Member Functions

- CPPS_FunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ cPPS_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

11.89.1 Constructor & Destructor Documentation

11.89.1.1 CPPS_FunctionNet() [1/2] CPPS_FunctionNet (

```
CMcsUsbNet^ mcsusb,
CMcsUsbFunctionPointerContainer^ cPPS_FunctionPointerContainer )
```

11.89.1.2 CPPS_FunctionNet() [2/2] CPPS_FunctionNet (

```
CMcsUsbNet^ mcsusb )
```

11.89.2 Member Function Documentation

11.89.2.1 GetAnalogVoltage() unsigned short GetAnalogVoltage (

```
unsigned short index )
```

11.89.2.2 GetAnalogVoltages() void GetAnalogVoltages (

```
unsigned short index,
[System::Runtime::InteropServices::Out] unsigned short% minvoltage,
[System::Runtime::InteropServices::Out] unsigned short% maxvoltage )
```

11.89.2.3 GetBubbleState() unsigned short GetBubbleState ( )

11.89.2.4 GetDigitalIn() unsigned short GetDigitalIn (

```
unsigned short index )
```

11.89.2.5 GetPumpCouple() unsigned int GetPumpCouple ( )

11.89.2.6 GetPumpEnableSpeedRatio() unsigned int GetPumpEnableSpeedRatio ( )

11.89.2.7 GetPumpFastOnOff() unsigned int GetPumpFastOnOff (

```
unsigned short index )
```

11.89.2.8 GetPumpFastSpeed()

```
short GetPumpFastSpeed (
    unsigned short index )
```

11.89.2.9 GetPumpFunctionSpeeds()

```
void GetPumpFunctionSpeeds (
    unsigned short index,
    [System::Runtime::InteropServices::Out] short% offspeed,
    [System::Runtime::InteropServices::Out] short% onspeed )
```

11.89.2.10 GetPumpManualOnOff()

```
unsigned int GetPumpManualOnOff (
    unsigned short index )
```

11.89.2.11 GetPumpMaxSpeed()

```
unsigned short GetPumpMaxSpeed (
    unsigned short index )
```

11.89.2.12 GetPumpModeType()

```
PP_Pump_Mode_Type_EnumNet GetPumpModeType (
    unsigned short index )
```

11.89.2.13 GetPumpSpeedRatio()

```
int GetPumpSpeedRatio ( )
```

11.89.2.14 GetPumpSpeedUnit()

```
int GetPumpSpeedUnit (
    unsigned short index )
```

11.89.2.15 GetSupplyVoltage()

```
unsigned short GetSupplyVoltage ( )
```

11.89.2.16 GetUseBubble()

```
unsigned int GetUseBubble (
    unsigned short index )
```

11.89.2.17 SetAnalogVoltages()

```
void SetAnalogVoltages (
    unsigned short index,
    unsigned short minvoltage,
    unsigned short maxvoltage )
```

11.89.2.18 SetPumpCouple()

```
void SetPumpCouple (
    unsigned int i )
```

11.89.2.19 SetPumpEnableSpeedRatio()

```
void SetPumpEnableSpeedRatio (
    unsigned int enable )
```

11.89.2.20 SetPumpFastOnOff()

```
void SetPumpFastOnOff (
    unsigned short index,
    unsigned int onoff )
```

11.89.2.21 SetPumpFastSpeed()

```
void SetPumpFastSpeed (
    unsigned short index,
    short fastspeed )
```

11.89.2.22 SetPumpFunctionSpeeds()

```
void SetPumpFunctionSpeeds (
    unsigned short index,
    short offspeed,
    short onspeed )
```

11.89.2.23 SetPumpManualOnOff()

```
void SetPumpManualOnOff (
    unsigned short index,
    unsigned int onoff )
```

11.89.2.24 SetPumpMaxSpeed()

```
void SetPumpMaxSpeed (
    unsigned short index,
    unsigned short maxspeed )
```

11.89.2.25 SetPumpModeType() void SetPumpModeType (

```
unsigned short index,
PP_Pump_Mode_Type_EnumNet type )
```

11.89.2.26 SetPumpSpeedRatio() void SetPumpSpeedRatio (

```
int ratio )
```

11.89.2.27 SetPumpSpeedUnit() void SetPumpSpeedUnit (

```
unsigned short index,
int speedunit )
```

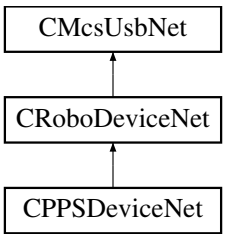
11.89.2.28 SetUseBubble() void SetUseBubble (

```
unsigned short index,
unsigned int usebubble )
```

11.90 CPPSDeviceNet Class Reference

CPPS4plus1DeviceNet is the to control the MCS HLA device

Inheritance diagram for CPPSDeviceNet:



Public Member Functions

- CPPSDeviceNet (void)

Additional Inherited Members

11.90.1 Detailed Description

CPPS4plus1DeviceNet is the to control the MCS HLA device

11.90.2 Constructor & Destructor Documentation

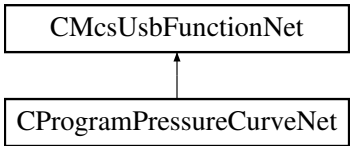
11.90.2.1 CPPSDeviceNet() `CPPSDeviceNet (`

`void )`

11.91 CProgramPressureCurveNet Class Reference

CProgramPressureCurveNet is the class to program pressure curves

Inheritance diagram for CProgramPressureCurveNet:



Public Member Functions

- CProgramPressureCurveNet (CMcsUsbNet^ 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

11.91.1 Detailed Description

CProgramPressureCurveNet is the class to program pressure curves

11.91.2 Constructor & Destructor Documentation

11.91.2.1 CProgramPressureCurveNet() `CProgramPressureCurveNet (`

`CMcsUsbNet^ mcsusb )`

Initializes a new instance of the CPulseGeneratorFunctionNet class.

11.91.2.2 ~CProgramPressureCurveNet() virtual ~CProgramPressureCurveNet (void) [virtual]

11.91.2.3 "!CProgramPressureCurveNet() !CProgramPressureCurveNet (void)

11.91.3 Member Function Documentation

11.91.3.1 GetRepeats() unsigned int GetRepeats (unsigned char *busnumber*, unsigned char *busaddress*, int32_t *channel*)

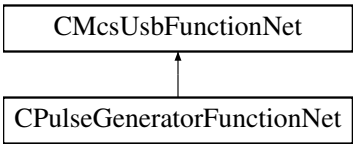
11.91.3.2 Program() void Program (unsigned char *busnumber*, unsigned char *busaddress*, int32_t *channel*, array< int32_t >^ *pressures*, array< int32_t >^ *steps*, array< int16_t >^ *durations*)

11.91.3.3 SetRepeats() void SetRepeats (unsigned char *busnumber*, unsigned char *busaddress*, int32_t *channel*, uint32_t *repeats*)

11.92 CPulseGeneratorFunctionNet Class Reference

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

Inheritance diagram for CPulseGeneratorFunctionNet:



Public Member Functions

- CPulseGeneratorFunctionNet (CMcsUsbNet^ 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

11.92.1 Detailed Description

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

11.92.2 Constructor & Destructor Documentation

11.92.2.1 CPulseGeneratorFunctionNet() [1/2]

```
CPulseGeneratorFunctionNet (
    CMcsUsbNet^ mcsusb,
    CMcsUsbFunctionPointerContainer^ pPulseGeneratorFunctionPointerContainer )
```

Initializes a new instance of the CPulseGeneratorFunctionNet class.

11.92.2.2 CPulseGeneratorFunctionNet() [2/2]

```
CPulseGeneratorFunctionNet (
    CMcsUsbNet^ mcsusb )
```

11.92.2.3 ~CPulseGeneratorFunctionNet()

```
virtual ~CPulseGeneratorFunctionNet ( ) [virtual]
```

11.92.2.4 "!CPulseGeneratorFunctionNet() !CPulseGeneratorFunctionNet ( )

11.92.3 Member Function Documentation

11.92.3.1 GetModeSelect() void GetModeSelect (
int32_t *generator_number*,
[System::Runtime::InteropServices::Out] PulseGenerator_Mode_EnumNet% *mode*,
[System::Runtime::InteropServices::Out] int32_t% *digitalchannel*)

Reads the generator mode

Parameters

generator_number	The generator number
mode	The generator mode
digitalchannel	The digital in channel used as gate

11.92.3.2 GetPeriod() int32_t GetPeriod (
int32_t *generator_number*)

Reads the generator period

Parameters

generator_number	The generator number

Returns

The period

11.92.3.3 GetPulseLength() int32_t GetPulseLength (
int32_t *generator_number*)

Reads the generator pulse length

Parameters

generator_number	The generator number

Returns

The pulse length

11.92.3.4 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

11.92.3.5 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

11.92.3.6 SetPulseLength() `void SetPulseLength (`

```
int32_t generator_number,
int32_t pulselength_in_10us )
```

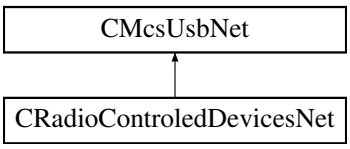
Writes the generator pulse length

Parameters

generator_number	The generator number
pulselength_in_10us	The pulse length

11.93 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

11.93.1 Constructor & Destructor Documentation

11.93.1.1 CRadioControledDevicesNet() [1/2] CRadioControledDevicesNet (void)

11.93.1.2 CRadioControledDevicesNet() [2/2] CRadioControledDevicesNet (CRadioControledDevices * *pRadioControled*) [protected]

11.93.2 Member Function Documentation

11.93.2.1 ConnectDevice() void ConnectDevice (unsigned short *sn*)

11.93.2.2 DisConnectDevice() void DisConnectDevice ( )

11.93.2.3 GetDeviceNames() `array<unsigned short> ^ GetDeviceNames ( )`

11.93.2.4 GetFrequency() `unsigned short GetFrequency ( )`

11.93.2.5 HasRadioControl() `bool HasRadioControl ( )`

11.93.2.6 SetFrequency() `void SetFrequency ( unsigned short frequency )`

11.93.2.7 StillConnected() `bool StillConnected ( )`

11.94 CCMOSMeaDeviceNet::CRegionOfInterestRect Class Reference

Public Member Functions

- CRegionOfInterestRect (int left, int top, int right, int bottom)
- CRegionOfInterestRect ^ DeepCopy ()

Public Attributes

- int m_Left
- int m_Top
- int m_Right
- int m_Bottom

11.94.1 Constructor & Destructor Documentation

11.94.1.1 CRegionOfInterestRect() `CRegionOfInterestRect ( int left, int top, int right, int bottom )`

11.94.2 Member Function Documentation

11.94.2.1 DeepCopy() `CRegionOfInterestRect ^ DeepCopy ( )`

11.94.3 Member Data Documentation

11.94.3.1 m_Bottom `int m_Bottom`

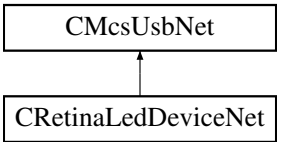
11.94.3.2 m_Left `int m_Left`

11.94.3.3 m_Right `int m_Right`

11.94.3.4 m_Top `int m_Top`

11.95 CRetinaLedDeviceNet Class Reference

Inheritance diagram for CRetinaLedDeviceNet:



Public Member Functions

- CRetinaLedDeviceNet ()
- ~CRetinaLedDeviceNet ()
- unsigned int SetTrigger (int enable)
- unsigned int SetLED (unsigned long long pattern)
- unsigned int SetTablepointer (int position)
- unsigned int GetTablepointer (int % position)
- unsigned int ClearTable ()
- unsigned int AddTableEntry (unsigned long long pattern)
- unsigned int AddLoopEntry (unsigned short repeats, unsigned short steps_back)
- unsigned int SetRepeat (int repeat)
- unsigned int SetLumi (int lumi)
- unsigned int SetPersistency (unsigned int persistency)

Additional Inherited Members

11.95.1 Constructor & Destructor Documentation

11.95.1.1 CRetinaLedDeviceNet() `CRetinaLedDeviceNet ( )`

11.95.1.2 ~CRetinaLedDeviceNet() `~CRetinaLedDeviceNet ( )`

11.95.2 Member Function Documentation

11.95.2.1 AddLoopEntry()

```
unsigned int AddLoopEntry (
    unsigned short repeats,
    unsigned short steps_back )
```

11.95.2.2 AddTableEntry()

```
unsigned int AddTableEntry (
    unsigned long long pattern )
```

11.95.2.3 ClearTable()

```
unsigned int ClearTable ( )
```

11.95.2.4 GetTablepointer()

```
unsigned int GetTablepointer (
    int % position )
```

11.95.2.5 SetLED()

```
unsigned int SetLED (
    unsigned long long pattern )
```

11.95.2.6 SetLumi()

```
unsigned int SetLumi (
    int lumi )
```

11.95.2.7 SetPersistency() unsigned int SetPersistency (unsigned int *persistency*)

11.95.2.8 SetRepeat() unsigned int SetRepeat (int *repeat*)

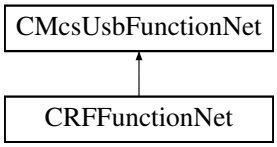
11.95.2.9 SetTablepointer() unsigned int SetTablepointer (int *position*)

11.95.2.10 SetTrigger() unsigned int SetTrigger (int *enable*)

11.96 CRFFunctionNet Class Reference

CRFFunctionNet is the class to control RF devices

Inheritance diagram for CRFFunctionNet:



Public Member Functions

- CRFFunctionNet (CMcsUsbNet^ 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

11.96.1 Detailed Description

CRFFunctionNet is the class to control RF devices

11.96.2 Constructor & Destructor Documentation

11.96.2.1 CRFFunctionNet() [1/2]

```
CRFFunctionNet (
    CMcsUsbNet^ mcsusb,
    CMcsUsbFunctionPointerContainer^ pRFFunctionPointerContainer )
```

Initializes a new instance of the CRFFunctionNet class.

11.96.2.2 CRFFunctionNet() [2/2]

```
CRFFunctionNet (
    CMcsUsbNet^ mcsusb )
```

11.96.2.3 ~CRFFunctionNet()

```
virtual ~CRFFunctionNet ( ) [virtual]
```

11.96.2.4 "!CRFFunctionNet()

```
!CRFFunctionNet ( )
```

11.96.3 Member Function Documentation

11.96.3.1 Connect()

```
void Connect (
    uint32_t sn )
```

connect to a RF device, use 0 to disconnect

Parameters

sn	the serial number

11.96.3.2 GetAvailableDeviceList() `array<uint32_t> ^ GetAvailableDeviceList ( )`

get a list of available devices

Returns

array of devices

11.96.3.3 GetAvailableDeviceListEx() `array<uint32_t> ^ GetAvailableDeviceListEx ( int list_Length )`

get a list of available devices

Parameters

list_Length	The maximal length of list.

Returns

array of devices

11.96.3.4 GetAvailableStateList() `array<uint32_t> ^ GetAvailableStateList ( )`

get a list of the states of the available devices

Returns

array of states

11.96.3.5 GetAvailableStateListEx() `array<uint32_t> ^ GetAvailableStateListEx ( int list_Length )`

get a list of the states of the available devices

Parameters

list_Length	The maximal length of list.

Returns

array of states

11.96.3.6 GetBaseFrequency() `uint32_t GetBaseFrequency (`

`CFirmwareDestinationNet` *destination* `)`

gets the base advertise frequency

Parameters

destination	the destination to query

Returns

the frequency

11.96.3.7 GetConnectedDevice() `uint32_t GetConnectedDevice ( )`

get connect RF device, 0 = no device connected

Returns

the serial number

11.96.3.8 GetState() `uint32_t GetState ( )`

get connection state

Returns

the state

11.96.3.9 GetTestMode() uint32_t GetTestMode ( )

gets test mode

Returns

the mode

11.96.3.10 GetWorkingFrequency() uint32_t GetWorkingFrequency ( )

gets the working frequency

Returns

the frequency

11.96.3.11 SetBaseFrequency() void SetBaseFrequency (CFirmwareDestinationNet *destination*, uint32_t *frequency*)

sets the base advertise frequency

Parameters

destination	the destination to set
frequency	the frequency

11.96.3.12 SetTestMode() void SetTestMode (uint32_t *mode*)

set test mode

Parameters

mode	the mode

11.96.3.13 SetWorkingFrequency() void SetWorkingFrequency (uint32_t *frequency*)

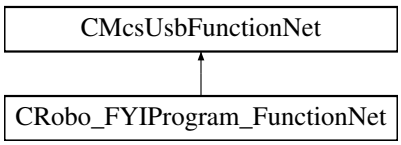
sets the working frequency

Parameters

frequency	the frequency

11.97 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

11.97.1 Constructor & Destructor Documentation

11.97.1.1 CRobo_FYIProgram_FunctionNet() [1/2]

```
CRobo_FYIProgram_FunctionNet (
        CMcsUsbNet^ mcsusb,
        CMcsUsbFunctionPointerContainer^ robo_FYIProgram_FunctionPointerContainer )
```

11.97.1.2 CRobo_FYIProgram_FunctionNet() [2/2]

```
CRobo_FYIProgram_FunctionNet (
        CMcsUsbNet^ mcsusb )
```

11.97.2 Member Function Documentation

11.97.2.1 GetLength()

```
int GetLength (
    unsigned char index )
```

11.97.2.2 GetState()

```
int GetState ( )
```

11.97.2.3 GetValve1()

```
unsigned int GetValve1 (
    unsigned char index )
```

11.97.2.4 GetValve2()

```
unsigned int GetValve2 (
    unsigned char index )
```

11.97.2.5 SetLength()

```
void SetLength (
    unsigned char index,
    int length )
```

11.97.2.6 SetValve1()

```
void SetValve1 (
    unsigned char index,
    unsigned int valve1 )
```

11.97.2.7 SetValve2()

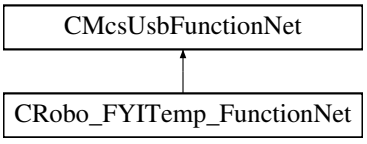
```
void SetValve2 (
    unsigned char index,
    unsigned int valve2 )
```

11.97.2.8 Start()

```
void Start ( )
```

11.98 CRobo_FYITemp_FunctionNet Class Reference

Inheritance diagram for CRobo_FYITemp_FunctionNet:



Public Member Functions

- CRobo_FYITemp_FunctionNet (CMcsUsbNet^ mcsusb)
- void SetRegulatorOnOff (unsigned char index, int onoff)
- int GetRegulatorOnOff (unsigned char index)
- void SetSollTemp (unsigned char index, int temp)
- int GetSollTemp (unsigned char index)
- void SetPCoeff (unsigned char index, int pcoeff)
- int GetPCoeff (unsigned char index)
- void SetICoeff (unsigned char index, int icoeff)
- int GetICoeff (unsigned char index)
- void SetMaxPower (unsigned char index, int power)
- int GetMaxPower (unsigned char index)

Additional Inherited Members

11.98.1 Constructor & Destructor Documentation

11.98.1.1 CRobo_FYITemp_FunctionNet() `CRobo_FYITemp_FunctionNet (`

`CMcsUsbNet^ mcsusb )`

11.98.2 Member Function Documentation

11.98.2.1 GetICoeff() `int GetICoeff (`

`unsigned char index )`

11.98.2.2 GetMaxPower() `int GetMaxPower (`

`unsigned char index )`

11.98.2.3 GetPCoeff() `int GetPCoeff (`

`unsigned char index )`

11.98.2.4 GetRegulatorOnOff() `int GetRegulatorOnOff (`

`unsigned char index )`

11.98.2.5 GetSollTemp()

```
int GetSollTemp (
    unsigned char index )
```

11.98.2.6 SetICoeff()

```
void SetICoeff (
    unsigned char index,
    int icoeff )
```

11.98.2.7 SetMaxPower()

```
void SetMaxPower (
    unsigned char index,
    int power )
```

11.98.2.8 SetPCoeff()

```
void SetPCoeff (
    unsigned char index,
    int pcoeff )
```

11.98.2.9 SetRegulatorOnOff()

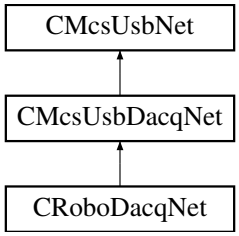
```
void SetRegulatorOnOff (
    unsigned char index,
    int onoff )
```

11.98.2.10 SetSollTemp()

```
void SetSollTemp (
    unsigned char index,
    int temp )
```

11.99 CRoboDacqNet Class Reference

Inheritance diagram for CRoboDacqNet:



Public Member Functions

- CRoboDacqNet (void)
- CRoboDacqNet (CRoboDeviceNet^ robodevice)
- void RunTable ()
- void RunTable (int timeout)
- void StopTable ()
- void StopTable (int timeout)
- void CancelTableLoop ()
- void CancelTableLoopAndStopTable ()
- void 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

11.99.1 Constructor & Destructor Documentation

11.99.1.1 CRoboDacqNet() [1/2] CRoboDacqNet (void)

11.99.1.2 CRoboDacqNet() [2/2] CRoboDacqNet (CRoboDeviceNet^ *robodevice*)

11.99.2 Member Function Documentation

11.99.2.1 CancelTableLoop() void CancelTableLoop ( )

11.99.2.2 CancelTableLoopAndStopTable() void CancelTableLoopAndStopTable ( )

11.99.2.3 ClampAmpRestart() void ClampAmpRestart ( )

11.99.2.4 DoRamp()

```
void DoRamp (
    int startValue,
    int endValue,
    int duration,
    int mode )
```

11.99.2.5 Emu_GetCellCapacity()

```
int Emu_GetCellCapacity ( )
```

11.99.2.6 Emu_GetCellPotential()

```
int Emu_GetCellPotential ( )
```

11.99.2.7 Emu_GetCellResists()

```
int Emu_GetCellResists ( )
```

11.99.2.8 Emu_GetElectrodeResists()

```
int Emu_GetElectrodeResists ( )
```

11.99.2.9 Emu_GetNoise()

```
int Emu_GetNoise ( )
```

11.99.2.10 Emu_SetCellCapacity()

```
void Emu_SetCellCapacity (
    int emuCellCapacity )
```

11.99.2.11 Emu_SetCellPotential()

```
void Emu_SetCellPotential (
    int emuCellPotential )
```

11.99.2.12 Emu_SetCellResists()

```
void Emu_SetCellResists (
    int emuCellResist )
```

11.99.2.13 Emu_SetElectrodeResists()

```
void Emu_SetElectrodeResists (
    int emuElectrodeResist )
```

11.99.2.14 Emu_SetNoise()

```
void Emu_SetNoise (
    int emuNoise )
```

11.99.2.15 GetAllDigout()

```
uint32_t GetAllDigout ( )
```

11.99.2.16 GetCapacityC()

```
int GetCapacityC ( )
```

11.99.2.17 GetCapacityV()

```
int GetCapacityV ( )
```

11.99.2.18 GetCapacityX()

```
int GetCapacityX ( )
```

11.99.2.19 GetClampAmpSerialNumber()

```
unsigned int GetClampAmpSerialNumber ( )
```

11.99.2.20 GetCommand()

```
int GetCommand (
    unsigned char command )
```

11.99.2.21 GetConfigurationBit()

```
bool GetConfigurationBit (
    unsigned short bit )
```

11.99.2.22 GetConfigurationBitAxc()

```
bool GetConfigurationBitAxc ( )
```

11.99.2.23 GetConfigurationBitBlu_Led()

```
bool GetConfigurationBitBlu_Led ( )
```

11.99.2.24 GetConfigurationBitBlu_LedToggleFast()

```
bool GetConfigurationBitBlu_LedToggleFast ( )
```

11.99.2.25 GetConfigurationBitBlu_LedToggleSlow() bool GetConfigurationBitBlu_LedToggleSlow ( )

11.99.2.26 GetConfigurationBitCC_Gen() bool GetConfigurationBitCC_Gen ( )

11.99.2.27 GetConfigurationBitCV_Gen() bool GetConfigurationBitCV_Gen ( )

11.99.2.28 GetConfigurationBitRC_Gen() bool GetConfigurationBitRC_Gen ( )

11.99.2.29 GetConfigurationBitRed_Led() bool GetConfigurationBitRed_Led ( )

11.99.2.30 GetConfigurationBitRed_LedSaturation() bool GetConfigurationBitRed_LedSaturation ( )

11.99.2.31 GetConfigurationBitRed_LedToggleFast() bool GetConfigurationBitRed_LedToggleFast ( )

11.99.2.32 GetConfigurationBitRed_LedToggleSlow() bool GetConfigurationBitRed_LedToggleSlow ( )

11.99.2.33 GetConfigurationBitRelais() bool GetConfigurationBitRelais ( )

11.99.2.34 GetConfigurationBitRV_Gen() bool GetConfigurationBitRV_Gen ( )

11.99.2.35 GetConfigurationBits() unsigned int GetConfigurationBits ( )

11.99.2.36 GetConfigurationBitStream() bool GetConfigurationBitStream ( )

11.99.2.37 GetConfigurationBitSupply() bool GetConfigurationBitSupply ( )

11.99.2.38 GetCrossTalkOffset() int GetCrossTalkOffset ( )

11.99.2.39 GetCrossTalkOptimum() int GetCrossTalkOptimum ( )

11.99.2.40 GetDigout() bool GetDigout (uint16_t *index*)

11.99.2.41 GetDisplayText() String ^ GetDisplayText (int *index*)

11.99.2.42 GetDownsampleFactor() int GetDownsampleFactor (int *index*)

11.99.2.43 GetFilter() int GetFilter ( )

11.99.2.44 GetFilterCoeffs() array<int> ^ GetFilterCoeffs (int *index*)

11.99.2.45 GetIC() int GetIC ( )

11.99.2.46 GetIClamp() int GetIClamp ( )

11.99.2.47 GetICOffset() int GetICOffset ( )

11.99.2.48 GetIGain() int GetIGain ( )

11.99.2.49 GetNIC_MS() int GetNIC_MS ( )

11.99.2.50 GetNUC_MS() int GetNUC_MS ( )

11.99.2.51 GetNUV_MS() int GetNUV_MS ( )

11.99.2.52 GetPGain() int GetPGain ( )

11.99.2.53 GetRecordingNumber() unsigned int GetRecordingNumber ( )

11.99.2.54 GetResistanceC() int GetResistanceC ( )

11.99.2.55 GetResistanceV() int GetResistanceV ( )

11.99.2.56 GetScreen() int GetScreen ( )

11.99.2.57 GetSimulation() unsigned int GetSimulation ( )

11.99.2.58 GetUC() int GetUC ( )

11.99.2.59 GetUClamp() int GetUClamp ( )

11.99.2.60 GetUCOffset() int GetUCOffset ( )

11.99.2.61 GetUpdateDisplay() int GetUpdateDisplay ( )

11.99.2.62 GetUV() int GetUV ( )

11.99.2.63 GetUVOffset() int GetUVOffset ( )

11.99.2.64 GetXGain() int GetXGain ( )

11.99.2.65 RunTable() [1/2] void RunTable ( )

11.99.2.66 RunTable() [2/2] void RunTable (int *timeout*)

11.99.2.67 SetAllDigout() void SetAllDigout (uint32_t *value*)

11.99.2.68 SetCommand() void SetCommand (unsigned char *command*, int *value*)

11.99.2.69 SetConfigurationBit()

```
void SetConfigurationBit (
    unsigned short bit,
    bool value )
```

11.99.2.70 SetConfigurationBitAxc()

```
void SetConfigurationBitAxc (
    bool value )
```

11.99.2.71 SetConfigurationBitBlu_Led()

```
void SetConfigurationBitBlu_Led (
    bool value )
```

11.99.2.72 SetConfigurationBitBlu_LedToggleFast()

```
void SetConfigurationBitBlu_LedToggleFast (
    bool value )
```

11.99.2.73 SetConfigurationBitBlu_LedToggleSlow()

```
void SetConfigurationBitBlu_LedToggleSlow (
    bool value )
```

11.99.2.74 SetConfigurationBitCC_Gen()

```
void SetConfigurationBitCC_Gen (
    bool value )
```

11.99.2.75 SetConfigurationBitCV_Gen()

```
void SetConfigurationBitCV_Gen (
    bool value )
```

11.99.2.76 SetConfigurationBitRC_Gen()

```
void SetConfigurationBitRC_Gen (
    bool value )
```

11.99.2.77 SetConfigurationBitRed_Led()

```
void SetConfigurationBitRed_Led (
    bool value )
```

11.99.2.78 SetConfigurationBitRed_LedSaturation()

```
void SetConfigurationBitRed_LedSaturation (
    bool value )
```

11.99.2.79 SetConfigurationBitRed_LedToggleFast()

```
void SetConfigurationBitRed_LedToggleFast (
    bool value )
```

11.99.2.80 SetConfigurationBitRed_LedToggleSlow()

```
void SetConfigurationBitRed_LedToggleSlow (
    bool value )
```

11.99.2.81 SetConfigurationBitRelais()

```
void SetConfigurationBitRelais (
    bool value )
```

11.99.2.82 SetConfigurationBitRV_Gen()

```
void SetConfigurationBitRV_Gen (
    bool value )
```

11.99.2.83 SetConfigurationBitStream()

```
void SetConfigurationBitStream (
    bool value )
```

11.99.2.84 SetConfigurationBitSupply()

```
void SetConfigurationBitSupply (
    bool value )
```

11.99.2.85 SetCrossTalkOffset()

```
void SetCrossTalkOffset (
    int CrossTalk )
```

11.99.2.86 SetCrossTalkOptimum()

```
void SetCrossTalkOptimum (
    int cxOptimum )
```

11.99.2.87 SetDigout()

```
void SetDigout (
    uint16_t index,
    bool enable )
```

11.99.2.88 SetDisplayText()

```
void SetDisplayText (
    int index,
    String^ displayText )
```

11.99.2.89 SetDownsampleFactor()

```
void SetDownsampleFactor (
    int index,
    int downsample_factor )
```

11.99.2.90 SetFilter()

```
void SetFilter (
    int filter )
```

11.99.2.91 SetFilterCoeffs()

```
void SetFilterCoeffs (
    int index,
    array< int >^ coeffs )
```

11.99.2.92 SetIClamp()

```
void SetIClamp (
    int iClamp )
```

11.99.2.93 SetICOffset()

```
void SetICOffset (
    int ICOffset )
```

11.99.2.94 SetIGain()

```
void SetIGain (
    int iGain )
```

11.99.2.95 SetNoFilterCoeffs()

```
void SetNoFilterCoeffs (
    int index )
```

11.99.2.96 SetPGain()

```
void SetPGain (
    int pGain )
```

11.99.2.97 SetRecordingNumber()

```
void SetRecordingNumber (
    unsigned int recordingNumber )
```

11.99.2.98 SetScreen()

```
void SetScreen (
    int screen )
```

11.99.2.99 SetSimulation()

```
void SetSimulation (
    unsigned int enable )
```

11.99.2.100 SetTriggerMaskValue()

```
void SetTriggerMaskValue (
    unsigned int mask,
    unsigned int value,
    unsigned int virtualDevice )
```

11.99.2.101 SetUClamp()

```
void SetUClamp (
    int uClamp )
```

11.99.2.102 SetUCOffset()

```
void SetUCOffset (
    int UCOffset )
```

11.99.2.103 SetUVOffset()

```
void SetUVOffset (
    int UVOffset )
```

11.99.2.104 SetXGain()

```
void SetXGain (
    int xGain )
```

11.99.2.105 StopTable() [1/2] void StopTable ( )

11.99.2.106 StopTable() [2/2] void StopTable (int *timeout*)

11.99.2.107 Table_Wait() void Table_Wait (unsigned int *tableWait*)

11.99.2.108 TableDefBegin() void TableDefBegin ( )

11.99.2.109 TableDefEnd() void TableDefEnd ( )

11.99.2.110 UpdateDisplay() void UpdateDisplay ( )

11.99.3 Member Data Documentation

11.99.3.1 TriggerMask_Default const unsigned int TriggerMask_Default = 0xFF00 [static]

11.99.3.2 TriggerValue_MoveAbs const unsigned int TriggerValue_MoveAbs = COMMAND_ROBO_MOVEABS [static]

11.99.3.3 TriggerValue_StartQueue const unsigned int TriggerValue_StartQueue = COMMAND_ROBO_↩ QUEUE [static]

11.99.3.4 VirtualDevice_ContinousDacq const unsigned int VirtualDevice_ContinousDacq = 0 [static]

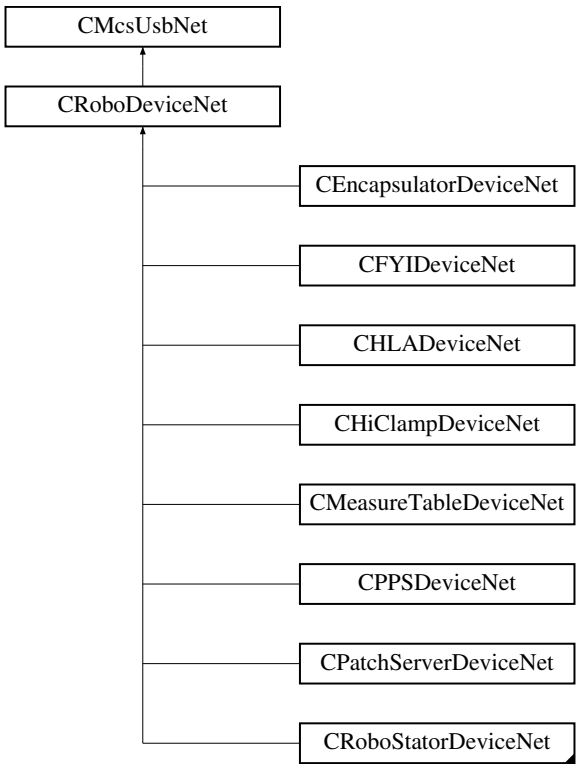
11.99.3.5 VirtualDevice_TableRun

```
const unsigned int VirtualDevice_TableRun = 1  [static]
```

11.100 CRoboDeviceNet Class Reference

CRoboDeviceNet is the base class for all Robo platform based devices

Inheritance diagram for CRoboDeviceNet:



Classes

- class RoboMainLowLevelCommands

Public Member Functions

- CRoboDeviceNet (void)
- ~CRoboDeviceNet (void)
- void SetInMovement ()

  Low level command, sets the internal state to "In Movement"
- bool GetInMovement ()

  Low level command, gets the internal state "In Movement"
- uint32_t GetMovementError ()

  Low level command, gets the error of the last movement end
- void FindReference (unsigned char busaddress, char axes)
- void FindReference (unsigned char busaddress, char axes, int timeout)

  Searches the reference position of the motor
- void MoveAbs (unsigned char busaddress, char axes, int x, int y)

  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

11.100.1 Detailed Description

CRoboDeviceNet is the base class for all Robo platform based devices

11.100.2 Constructor & Destructor Documentation

11.100.2.1 CRoboDeviceNet() `CRoboDeviceNet (`

`void )`

11.100.2.2 ~CRoboDeviceNet() `~CRoboDeviceNet (`

`void )`

11.100.3 Member Function Documentation

11.100.3.1 CancelPoolLoop() `void CancelPoolLoop ( )`

11.100.3.2 CancelPoolLoopAndStopMovement() void CancelPoolLoopAndStopMovement ( )

11.100.3.3 EnableQueue() void EnableQueue (
bool *enable*)

11.100.3.4 FindReference() [1/2] void FindReference (
unsigned char *busaddress*,
char *axes*)

11.100.3.5 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)

11.100.3.6 GetAirpressure() unsigned int GetAirpressure ( )

11.100.3.7 GetAirpressureLimit() void GetAirpressureLimit (
[System::Runtime::InteropServices::Out] unsigned int% *lowerpressure*,
[System::Runtime::InteropServices::Out] unsigned int% *upperpressure*)

11.100.3.8 GetAirValve() unsigned int GetAirValve ( )

11.100.3.9 GetCurrentAirvalve() unsigned int GetCurrentAirvalve ( )

11.100.3.10 GetCurrentAirvalveLimit()

```
void GetCurrentAirvalveLimit (
 [System::Runtime::InteropServices::Out] unsigned int% lowercurrent,
 [System::Runtime::InteropServices::Out] unsigned int% uppercurrent )
```

11.100.3.11 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

11.100.3.12 GetErrorAirpressure()

```
unsigned int GetErrorAirpressure ( )
```

11.100.3.13 GetErrorCurrentAirvalve()

```
unsigned int GetErrorCurrentAirvalve ( )
```

11.100.3.14 GetErrorVoltage12V()

```
unsigned int GetErrorVoltage12V ( )
```

11.100.3.15 GetErrorVoltage5V()

```
unsigned int GetErrorVoltage5V ( )
```

11.100.3.16 GetErrorVoltageAirvalve()

```
unsigned int GetErrorVoltageAirvalve ( )
```

11.100.3.17 GetErrorVoltageRs485A()

```
unsigned int GetErrorVoltageRs485A ( )
```

11.100.3.18 GetErrorVoltageRs485B() unsigned int GetErrorVoltageRs485B ( )

11.100.3.19 GetErrorVoltageValves() unsigned int GetErrorVoltageValves ( )

11.100.3.20 GetInMovement() bool GetInMovement ( )

Low level command, gets the internal state "In Movement"

11.100.3.21 GetMinPressure() int GetMinPressure ( )

11.100.3.22 GetMovementError() uint32_t GetMovementError ( )

Low level command, gets the error of the last movement end

11.100.3.23 GetVoltage12V() unsigned int GetVoltage12V ( )

11.100.3.24 GetVoltage12VLimit() void GetVoltage12VLimit (

```
[System::Runtime::InteropServices::Out] unsigned int% lowervoltage,
[System::Runtime::InteropServices::Out] unsigned int% uppervoltage )
```

11.100.3.25 GetVoltage5V() unsigned int GetVoltage5V ( )

11.100.3.26 GetVoltage5VLimit() void GetVoltage5VLimit (

```
[System::Runtime::InteropServices::Out] unsigned int% lowervoltage,
[System::Runtime::InteropServices::Out] unsigned int% uppervoltage )
```

11.100.3.27 GetVoltageAirvalve() unsigned int GetVoltageAirvalve ( )

11.100.3.28 GetVoltageAirvalveLimit()

```
void GetVoltageAirvalveLimit (
 [System::Runtime::InteropServices::Out] unsigned int% lowervoltage,
 [System::Runtime::InteropServices::Out] unsigned int% uppervoltage )
```

11.100.3.29 GetVoltageRs485A()

```
unsigned int GetVoltageRs485A ( )
```

11.100.3.30 GetVoltageRs485ALimit()

```
void GetVoltageRs485ALimit (
 [System::Runtime::InteropServices::Out] unsigned int% lowervoltage,
 [System::Runtime::InteropServices::Out] unsigned int% uppervoltage )
```

11.100.3.31 GetVoltageRs485B()

```
unsigned int GetVoltageRs485B ( )
```

11.100.3.32 GetVoltageRs485BLimit()

```
void GetVoltageRs485BLimit (
 [System::Runtime::InteropServices::Out] unsigned int% lowervoltage,
 [System::Runtime::InteropServices::Out] unsigned int% uppervoltage )
```

11.100.3.33 GetVoltageValves()

```
unsigned int GetVoltageValves ( )
```

11.100.3.34 GetVoltageValvesLimit()

```
void GetVoltageValvesLimit (
 [System::Runtime::InteropServices::Out] unsigned int% lowervoltage,
 [System::Runtime::InteropServices::Out] unsigned int% uppervoltage )
```

11.100.3.35 IsQueueEnabled()

```
bool IsQueueEnabled ( )
```

11.100.3.36 IsQueueStarted()

```
bool IsQueueStarted ( )
```

11.100.3.37 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

11.100.3.38 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)

11.100.3.39 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

11.100.3.40 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)

11.100.3.41 NullCommand()

```
void NullCommand (
    unsigned int marker )
```

11.100.3.42 SetAirpressureLimit()

```
void SetAirpressureLimit (
    unsigned int lowerpressure,
    unsigned int upperpressure )
```

11.100.3.43 SetAirValve()

```
void SetAirValve (
    unsigned int onoff )
```

11.100.3.44 SetCurrentAirvalveLimit()

```
void SetCurrentAirvalveLimit (
    unsigned int lowercurrent,
    unsigned int uppercurrent )
```

11.100.3.45 SetCurrentAndAir() [1/2]

```
void SetCurrentAndAir (
    unsigned char busaddress,
    char axes,
    unsigned short onoff )
```

11.100.3.46 SetCurrentAndAir() [2/2]

```
void SetCurrentAndAir (
    unsigned char busaddress,
    char axes,
    unsigned short onoff,
    int timeout )
```

11.100.3.47 SetInMovement() void SetInMovement ( )

Low level command, sets the internal state to "In Movement"

11.100.3.48 SetMinPressure() void SetMinPressure (int *pressure*)

11.100.3.49 SetVoltage12VLimit() void SetVoltage12VLimit (unsigned int *lowervoltage*, unsigned int *uppervoltage*)

11.100.3.50 SetVoltage5VLimit() void SetVoltage5VLimit (unsigned int *lowervoltage*, unsigned int *uppervoltage*)

11.100.3.51 SetVoltageAirvalveLimit() void SetVoltageAirvalveLimit (unsigned int *lowervoltage*, unsigned int *uppervoltage*)

11.100.3.52 SetVoltageRs485ALimit() void SetVoltageRs485ALimit (unsigned int *lowervoltage*, unsigned int *uppervoltage*)

11.100.3.53 SetVoltageRs485BLimit() void SetVoltageRs485BLimit (unsigned int *lowervoltage*, unsigned int *uppervoltage*)

11.100.3.54 SetVoltageValvesLimit() void SetVoltageValvesLimit (unsigned int *lowervoltage*, unsigned int *uppervoltage*)

11.100.3.55 StartQueue() void StartQueue (bool *start*)

11.100.3.56 StopMovement() [1/2] void StopMovement (unsigned char *busaddress*, char *axes*)

11.100.3.57 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)

11.100.3.58 WaitTimer() void WaitTimer (uint32_t *waittime*, int *timeout*)

11.100.4 Member Data Documentation

11.100.4.1 Axes_I const char Axes_I = 2 [static]

Bit pattern for i axis for axes argument

11.100.4.2 Axes_X const char Axes_X = 1 [static]

Bit pattern for x axis for axes argument

11.100.4.3 Axes_Y `const char Axes_Y = 2 [static]`

Bit pattern for y axis for axes argument

11.100.4.4 Axes_Z `const char Axes_Z = 1 [static]`

Bit pattern for z axis for axes argument

11.100.4.5 Axis_I `const byte Axis_I = 1 [static]`

Axis number of i for axis argument

11.100.4.6 Axis_X `const byte Axis_X = 0 [static]`

Axis number of x for axis argument

11.100.4.7 Axis_Y `const byte Axis_Y = 1 [static]`

Axis number of y for axis argument

11.100.4.8 Axis_Z `const byte Axis_Z = 0 [static]`

Axis number of z for axis argument

11.100.4.9 McsBus_XY `const byte McsBus_XY = 1 [static]`

McsBus address for the xy-plane

11.100.4.10 McsBus_ZI `const byte McsBus_ZI = 2 [static]`

McsBus address for the z and i axes

11.100.4.11 RoboError_AnotherMaster

```
const uint32_t RoboError_AnotherMaster = ( (0xA0110000L) | 0x0005 ) [static]
```

11.100.4.12 RoboError_Base

```
const uint32_t RoboError_Base = (0xA0110000L) [static]
```

11.100.4.13 RoboError_CannotEscapeEndSwitch

```
const uint32_t RoboError_CannotEscapeEndSwitch = ( (0xA0110000L) | 0x0009 ) [static]
```

11.100.4.14 RoboError_CommandAlreadyInProgress

```
const uint32_t RoboError_CommandAlreadyIn↩
Progress = ( (0xA0110000L) | 0x000A ) [static]
```

11.100.4.15 RoboError_CommandNotPossible

```
const uint32_t RoboError_CommandNotPossible = ( (0xA0110000L) | 0x0014 ) [static]
```

11.100.4.16 RoboError_CommunicationTimeout

```
const uint32_t RoboError_CommunicationTimeout = ( (0xA0110000L) | 0x0004 ) [static]
```

11.100.4.17 RoboError_DacqNotReady

```
const uint32_t RoboError_DacqNotReady = ( (0xA0110000L) | 0x0015 ) [static]
```

11.100.4.18 RoboError_DLLMovementTimeout

```
const uint32_t RoboError_DLLMovementTimeout = ( (0xA0110000L) | 0x1001) [static]
```

11.100.4.19 RoboError_FindReferenceMethod

```
const uint32_t RoboError_FindReferenceMethod = ( (0xA0110000L) | 0x0006 ) [static]
```

11.100.4.20 RoboError_GilsonCommandPending

```
const uint32_t RoboError_GilsonCommandPending = ( (0xA0110000L) | 0x0011 ) [static]
```

11.100.4.21 RoboError_GilsonTimeout

```
const uint32_t RoboError_GilsonTimeout = ( (0xA0110000L) | 0x000F ) [static]
```

11.100.4.22 RoboError_GilsonWrondID

```
const uint32_t RoboError_GilsonWrondID = ( (0xA0110000L) | 0x0010 ) [static]
```

11.100.4.23 RoboError_McsBus_UnknownCommand

```
const uint32_t RoboError_McsBus_Unknown↩
Command = ( (0xA0110000L) | 0x003F) [static]
```

11.100.4.24 RoboError_NoEndSwitch

```
const uint32_t RoboError_NoEndSwitch = ( (0xA0110000L) | 0x0008 ) [static]
```

11.100.4.25 RoboError_NoMoreData

```
const uint32_t RoboError_NoMoreData = ( (0xA0110000L) | 0x0016 ) [static]
```

11.100.4.26 RoboError_NoReference

```
const uint32_t RoboError_NoReference = ( (0xA0110000L) | 0x000B ) [static]
```

11.100.4.27 RoboError_NoSpeedOrAcceleration

```
const uint32_t RoboError_NoSpeedOrAcceleration = ( (0xA0110000L) | 0x0007 ) [static]
```

11.100.4.28 RoboError_OverPressure

```
const uint32_t RoboError_OverPressure = ( (0xA0110000L) | 0x000C ) [static]
```

11.100.4.29 RoboError_ParameterNotAllowed

```
const uint32_t RoboError_ParameterNotAllowed = ( (0xA0110000L) | 0x0012 ) [static]
```

11.100.4.30 RoboError_PeristalticTimeout

```
const uint32_t RoboError_PeristalticTimeout = ( (0x↩
A0110000L) | 0x000E ) [static]
```

11.100.4.31 RoboError_Phase0OutOfRange

```
const uint32_t RoboError_Phase0OutOfRange = ( (0xA0110000L) | 0x000D ) [static]
```

11.100.4.32 RoboError_PollLoopCanceled

```
const uint32_t RoboError_PollLoopCanceled = ( (0xA0110000L) | 0x1002) [static]
```

11.100.4.33 RoboError_PollLoopCanceledAndStopMovement

```
const uint32_t RoboError_PollLoopCanceledAndStopMovement = ( (0xA0110000L) | 0x1003) [static]
```

11.100.4.34 RoboError_Pressure

```
const uint32_t RoboError_Pressure = ( (0xA0110000L) | 0x0002 ) [static]
```

11.100.4.35 RoboError_RangeExceeded

```
const uint32_t RoboError_RangeExceeded = ( (0xA0110000L) | 0x0003 ) [static]
```

11.100.4.36 RoboError_StateChangeNotPossible

```
const uint32_t RoboError_StateChangeNotPossible = ( (0xA0110000L) | 0x0013 ) [static]
```

11.100.4.37 RoboError_Timeout

```
const uint32_t RoboError_Timeout = ( (0xA0110000L) | 0x0001 ) [static]
```

11.100.4.38 RoboError_UnknownCommand

```
const uint32_t RoboError_UnknownCommand = ( (0xA0110000L) ) [static]
```

11.100.5 Property Documentation

11.100.5.1 McsBus

```
CMcsBusNet^ McsBus [get]
```

11.100.5.2 McsBus_MotorControl CMcsBus_MotorControlNet^ McsBus_MotorControl [get]

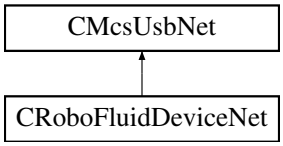
11.100.5.3 RoboMainLowLevelCommand RoboMainLowLevelCommands^ RoboMainLowLevelCommand [get]

11.100.6 Event Documentation

11.100.6.1 RoboStatusEvent RoboStatusEventDelegate^ RoboStatusEvent

11.101 CRoboFluidDeviceNet Class Reference

Inheritance diagram for CRoboFluidDeviceNet:



Public Member Functions

- CRoboFluidDeviceNet (void)
- ~CRoboFluidDeviceNet (void)
- void SetValve (int value)

  Open or Close valves.
- void SetSingleValve (int valve, bool onoff)

  Opens or Closes a valve.
- int GetValve ()

  Query the state of the values.
- bool GetSingleValve (int valve)

  Query the state of a valve.
- void CloseAllValves ()
- void PumpOn (int index, short speed)
- void SetPumpSpeed (int index, short speed)
- void PumpOff (int index)
- short GetPumpSpeed (int index)
- bool IsPumpMotorOn (int index)

Protected Attributes

- CRoboFluidDevice * m_pRoboFluidDevice
- CMcsBus_MotorControlNet ^ m_pMcsBus_MotorControlNet

Properties

- CMcsBus_MotorControlNet^ McsBus_MotorControl [get]

Additional Inherited Members

11.101.1 Constructor & Destructor Documentation

11.101.1.1 CRoboFluidDeviceNet() CRoboFluidDeviceNet (void)

11.101.1.2 ~CRoboFluidDeviceNet() ~CRoboFluidDeviceNet (void)

11.101.2 Member Function Documentation

11.101.2.1 CloseAllValves() void CloseAllValves ( )

11.101.2.2 GetPumpSpeed() short GetPumpSpeed (int *index*)

11.101.2.3 GetSingleValve() bool GetSingleValve (int *valve*)

Query the state of a valve.

Parameters

valve	number of valve /*!

Returns

state of the valve

11.101.2.4 GetValve() int GetValve ( )

Query the state of the values.

Returns

the current state of the valves as a bit pattern.

11.101.2.5 IsPumpMotorOn() bool IsPumpMotorOn (int *index*)

11.101.2.6 PumpOff() void PumpOff (int *index*)

11.101.2.7 PumpOn() void PumpOn (int *index*, short *speed*)

11.101.2.8 SetPumpSpeed() void SetPumpSpeed (int *index*, short *speed*)

11.101.2.9 SetSingleValve() void SetSingleValve (int *valve*, bool *onoff*)

Opens or Closes a valve.

Parameters

valve	number of valve to be changed /∗!
onoff	open or close the valve

11.101.2.10 SetValve() void SetValve (int *value*)

Open or Close valves.

Parameters

value	bit pattern of valves which should be open.

11.101.3 Member Data Documentation

11.101.3.1 m_pMcsBus_MotorControlNet `CMcsBus_MotorControlNet ^ m_pMcsBus_MotorControlNet [protected]`

11.101.3.2 m_pRoboFluidDevice `CRoboFluidDevice* m_pRoboFluidDevice [protected]`

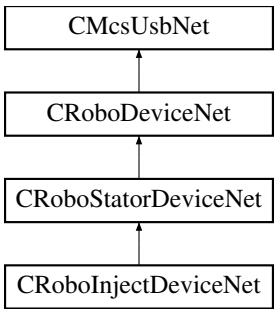
11.101.4 Property Documentation

11.101.4.1 McsBus_MotorControl `CMcsBus_MotorControlNet^ McsBus_MotorControl [get]`

11.102 CRoboInjectDeviceNet Class Reference

CRoboInjectDeviceNet is the to control the MCS RoboInject device

Inheritance diagram for CRoboInjectDeviceNet:



Public Member Functions

- CRoboInjectDeviceNet (void)

Additional Inherited Members

11.102.1 Detailed Description

CRoboInjectDeviceNet is the to control the MCS RoboInject device

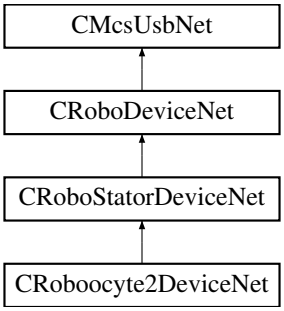
11.102.2 Constructor & Destructor Documentation

11.102.2.1 CRoboInjectDeviceNet() `CRoboInjectDeviceNet ( void )`

11.103 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

11.103.1 Detailed Description

CRoboocyte2DeviceNet is the class to control the MCS Roboocyte2 device

11.103.2 Constructor & Destructor Documentation

11.103.2.1 CRoboocyte2DeviceNet() CRoboocyte2DeviceNet (void)

11.103.3 Member Function Documentation

11.103.3.1 GetAxisLED() bool GetAxisLED ( )

11.103.3.2 GetGilsonDevice() CGilsonDeviceNet ^ GetGilsonDevice ( )

11.103.3.3 GetMcsBus_Extension() CMcsBus_ExtensionNet ^ GetMcsBus_Extension ( )

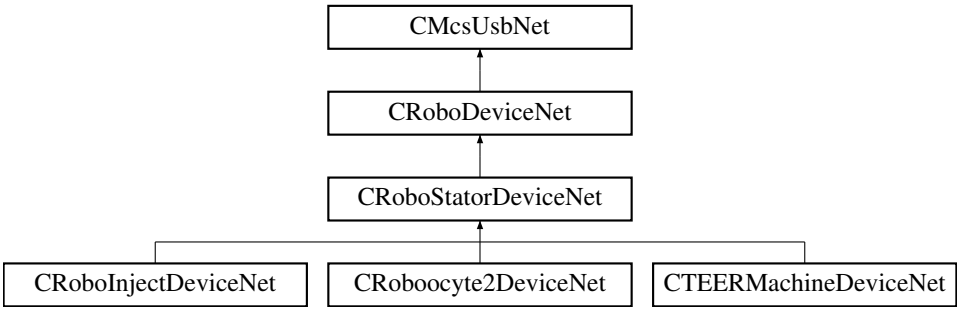
11.103.3.4 GetRoboDacq() CRoboDacqNet ^ GetRoboDacq ( )

11.103.3.5 GetRoboFluidDevice() CRoboFluidDeviceNet ^ GetRoboFluidDevice ( )

11.103.3.6 SetAxisLED() void SetAxisLED (bool *onoff*)

11.104 CRoboStatorDeviceNet Class Reference

Inheritance diagram for CRoboStatorDeviceNet:



Classes

- class RoboMainStatorLowLevelCommands

Public Member Functions

- CRoboStatorDeviceNet (void)
- void FindReferenceXY ()
- void FindReferenceXY (int timeout)
- void FindReferenceZ ()
- void FindReferenceZ (int timeout)
- void FindReferenceI ()
- void FindReferenceI (int timeout)
- unsigned char HasRefXY ()
- unsigned char HasRefZ ()
- unsigned char HasRefI ()
- void MoveAbsXY (int x, int y)
- void MoveAbsXY (int x, int y, int timeout)
- void MoveAbsZ (int z)
- void MoveAbsZ (int z, int timeout)
- void MoveAbsI (int i)
- void MoveAbsI (int i, int timeout)
- void StopMovementXY ()
- void StopMovementXY (int timeout)
- void StopMovementZ ()
- void StopMovementZ (int timeout)
- void StopMovementI ()
- void StopMovementI (int timeout)
- void SetCurrentAndAirXY (unsigned short onoff)
- void SetCurrentAndAirXY (unsigned short onoff, int timeout)
- void GetCurrentPositionXY ([System::Runtime::InteropServices::Out]int% x, [System::Runtime::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

11.104.1 Constructor & Destructor Documentation

11.104.1.1 CRoboStatorDeviceNet() CRoboStatorDeviceNet (void)

11.104.2 Member Function Documentation

11.104.2.1 FindReferenceI() [1/2] void FindReferenceI ( )

11.104.2.2 FindReferenceI() [2/2] void FindReferenceI (int *timeout*)

11.104.2.3 FindReferenceXY() [1/2] void FindReferenceXY ( )

11.104.2.4 FindReferenceXY() [2/2] void FindReferenceXY (int *timeout*)

11.104.2.5 FindReferenceZ() [1/2] void FindReferenceZ ( )

11.104.2.6 FindReferenceZ() [2/2] void FindReferenceZ (int *timeout*)

11.104.2.7 GetCurrentPositionI() void GetCurrentPositionI ([System::Runtime::InteropServices::Out] int% *i*)

11.104.2.8 GetCurrentPositionXY()

```
void GetCurrentPositionXY (
    [System::Runtime::InteropServices::Out] int% x,
    [System::Runtime::InteropServices::Out] int% y )
```

11.104.2.9 GetCurrentPositionZ()

```
void GetCurrentPositionZ (
    [System::Runtime::InteropServices::Out] int% z )
```

11.104.2.10 HasRefI()

```
unsigned char HasRefI ( )
```

11.104.2.11 HasRefXY()

```
unsigned char HasRefXY ( )
```

11.104.2.12 HasRefZ()

```
unsigned char HasRefZ ( )
```

11.104.2.13 MoveAbsI() [1/2]

```
void MoveAbsI (
    int i )
```

11.104.2.14 MoveAbsI() [2/2]

```
void MoveAbsI (
    int i,
    int timeout )
```

11.104.2.15 MoveAbsXY() [1/2]

```
void MoveAbsXY (
    int x,
    int y )
```

11.104.2.16 MoveAbsXY() [2/2]

```
void MoveAbsXY (
    int x,
    int y,
    int timeout )
```

11.104.2.17 MoveAbsZ() [1/2]

```
void MoveAbsZ (
    int z )
```

11.104.2.18 MoveAbsZ() [2/2]

```
void MoveAbsZ (
    int z,
    int timeout )
```

11.104.2.19 SetAccelerationI()

```
void SetAccelerationI (
    int a )
```

11.104.2.20 SetAccelerationNativeI()

```
void SetAccelerationNativeI (
    int a )
```

11.104.2.21 SetAccelerationNativeXY()

```
void SetAccelerationNativeXY (
    int a )
```

11.104.2.22 SetAccelerationNativeZ()

```
void SetAccelerationNativeZ (
    int a )
```

11.104.2.23 SetAccelerationXY()

```
void SetAccelerationXY (
    int a )
```

11.104.2.24 SetAccelerationZ()

```
void SetAccelerationZ (
    int a )
```

11.104.2.25 SetCurrentAndAirXY() [1/2]

```
void SetCurrentAndAirXY (
    unsigned short onoff )
```

11.104.2.26 SetCurrentAndAirXY() [2/2] void SetCurrentAndAirXY (

```
unsigned short onoff,
int timeout )
```

11.104.2.27 SetSpeedI() void SetSpeedI (

```
int v )
```

11.104.2.28 SetSpeedNativeI() void SetSpeedNativeI (

```
int v )
```

11.104.2.29 SetSpeedNativeXY() void SetSpeedNativeXY (

```
int v )
```

11.104.2.30 SetSpeedNativeZ() void SetSpeedNativeZ (

```
int v )
```

11.104.2.31 SetSpeedXY() void SetSpeedXY (

```
int v )
```

11.104.2.32 SetSpeedZ() void SetSpeedZ (

```
int v )
```

11.104.2.33 SetVelocityI() void SetVelocityI (

```
int v )
```

11.104.2.34 SetVelocityXY() void SetVelocityXY (

```
int v )
```

11.104.2.35 SetVelocityZ() void SetVelocityZ (int *v*)

11.104.2.36 StopMovementI() [1/2] void StopMovementI ( )

11.104.2.37 StopMovementI() [2/2] void StopMovementI (int *timeout*)

11.104.2.38 StopMovementXY() [1/2] void StopMovementXY ( )

11.104.2.39 StopMovementXY() [2/2] void StopMovementXY (int *timeout*)

11.104.2.40 StopMovementZ() [1/2] void StopMovementZ ( )

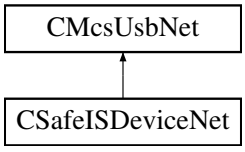
11.104.2.41 StopMovementZ() [2/2] void StopMovementZ (int *timeout*)

11.104.3 Property Documentation

11.104.3.1 RoboMainStatorLowLevelCommand RoboMainStatorLowLevelCommands^ RoboMainStator↩ LowLevelCommand [get]

11.105 CSafeISDeviceNet Class Reference

Inheritance diagram for CSafeISDeviceNet:



Public Member Functions

- CSafeISDeviceNet (void)

  Initializes a new instance of the CSafeISDeviceNet class.
- ~CSafeISDeviceNet (void)

  Releases unmanaged resources and performs other cleanup operations before the CSafeISDeviceNet is reclaimed by garbage collection.
- void SetSwitches (unsigned short switches)

  Sets the switches for all electrodes on the device. Do not use during measurement
- void SetAdcChannels (unsigned char channels)

  Sets the ADC channels you want to be sampled
- void SetAdcSamplePos (array< unsigned short >^ positions)

  Sets the sample position of the ADC.
- void SetDacMode (unsigned char mode)

  Sets the DAC mode.
- void SetDacPulseform (array< short >^ pulseform)

  Sets the DAC pulseform.
- void SetDacPeriode (unsigned int periode)

  Sets the DAC periode.

Properties

- CRoboDeviceNet^ 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

11.105.1 Detailed Description

11.105.2 Constructor & Destructor Documentation

11.105.2.1 CSafeISDeviceNet() `CSafeISDeviceNet ( void )`

Initializes a new instance of the CSafeISDeviceNet class.

11.105.2.2 ~CSafeISDeviceNet() `~CSafeISDeviceNet ( void )`

Releases unmanaged resources and performs other cleanup operations before the CSafeISDeviceNet is reclaimed by garbage collection.

11.105.3 Member Function Documentation

11.105.3.1 SetAdcChannels() `void SetAdcChannels ( unsigned char channels )`

Sets the ADC channels you want to be sampled

Parameters

channels	The bitmap of the 8 channels. Set bit to 1 for the channels you want measure

11.105.3.2 SetAdcSamplePos() `void SetAdcSamplePos ( array< unsigned short >^ positions )`

Sets the sample position of the ADC.

Parameters

positions	The positions in units of 0.1µs.

11.105.3.3 SetDacMode() `void SetDacMode ( unsigned char mode )`

Sets the DAC mode.

Parameters

mode	The mode: 0 = Impedance ; 1 = Amperometry

11.105.3.4 SetDacPeriode() `void SetDacPeriode ( unsigned int periode )`

Sets the DAC periode.

Parameters

periode	The periode in units of 10µs.

11.105.3.5 SetDacPulseform() `void SetDacPulseform ( array< short >^ pulseform )`

Sets the DAC pulseform.

Parameters

pulseform	The pulseform.

11.105.3.6 SetSwitches() `void SetSwitches ( unsigned short switches )`

Sets the switches for all electrodes on the device. Do not use during measurement

Parameters

switches	The switches: See Schematics for the meaning

11.105.4 Property Documentation

11.105.4.1 DacqDevice `CMcsUsbDacqNet^ DacqDevice [get]`

Gets the CMcsUsbDacqNet. Use this to control the data aquisition.

11.105.4.2 FluidControlDevice `CFluidControlDeviceNet^ FluidControlDevice [get]`

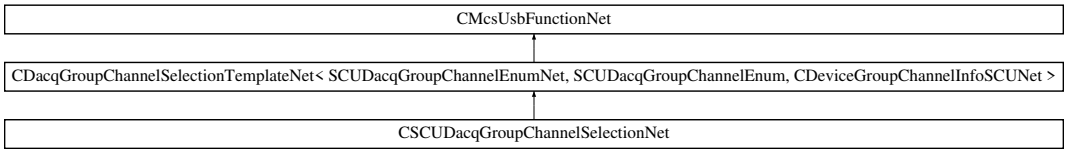
Gets the CFluidControlDeviceNet. Use this to control the valves. Only SetSingleValve is implemented for CSafeISDeviceNet.

11.105.4.3 RoboDevice `CRoboDeviceNet^ RoboDevice [get]`

Gets the CRoboDeviceNet. Use this to control the syringe.

11.106 CSCUDacqGroupChannelSelectionNet Class Reference

Inheritance diagram for CSCUDacqGroupChannelSelectionNet:



Public Member Functions

- CSCUDacqGroupChannelSelectionNet (CMcsUsbNet^ mcsusb)

Additional Inherited Members

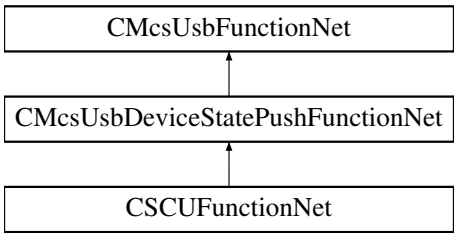
11.106.1 Constructor & Destructor Documentation

11.106.1.1 CSCUDacqGroupChannelSelectionNet() `CSCUDacqGroupChannelSelectionNet ( CMcsUsbNet^ mcsusb )`

11.107 CSCUFunctionNet Class Reference

CSCUFunctionNet is the class to control the SCU device

Inheritance diagram for CSCUFunctionNet:



Public Member Functions

- delegate void OnGetAvailableHeadstages (uint32_t AvailableHeadstages)
- delegate void OnIsHeadstageAvailable (uint32_t Headstage, bool available)
- CSCUFunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ 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.
- 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.
- bool GetHeadstagePowerStateAtStart (uint32_t Headstage)

  Gets the Power Status at SCU Power on of a given headstage.
- void SetHeadstagePowerStateAtStart (uint32_t Headstage, bool Powerstatus)

  Sets the Power Status at SCU Power on of a given headstage.
- bool HasGalvanicIsolation ()

  Has galvanic isolated hardware
- bool HasAnalogOut ()

  Has AnalogOut hardware
- void EnableAnalogOut (bool enable)

  Enables AnalogOut globally
- bool IsAnalogOutEnabled ()

  Is AnalogOut enabled
- void SetAnalogOutDACRange (AnalogOut_DAC_Range_EnumNet range)

  Sets the analog out DAC range
- AnalogOut_DAC_Range_EnumNet GetAnalogOutDACRange ()

  Gets the analog out DAC range
- void SetAnalogOutADCRange (uint32_t range)

  Sets the analog out ADC range

- uint32_t GetAnalogOutADCRange ()

  Gets the analog out ADC range
- void AutomaticAnalogOut (bool automatic)

  Sets automatic source channel selection
- bool IsAutomaticAnalogOut ()

  Is Automatic source channel selection selected
- void SetAnalogOutChannels (uint32_t out_channel, uint32_t source_channel)

  Set the source channel number for a certain output channel
- uint32_t GetAnalogOutChannels (uint32_t out_channel)

  Get the connected source channel number for a certain output channel
- void SetReferenceElectrodeSwitchState (uint32_t Headstage, ReferenceElectrodeSwitchPositionEnumNet NewSwitchPos)

  Sets the position of the switch for the reference electrode
- ReferenceElectrodeSwitchPositionEnumNet GetReferenceElectrodeSwitchState (uint32_t Headstage)

  Gets the position of the switch for the reference electrode
- void SetReferenceElectrodeMode (uint32_t Headstage, ReferenceElectrodeModeEnumNet NewValue)

  Sets the mode for the reference electrode
- ReferenceElectrodeModeEnumNet GetReferenceElectrodeMode (uint32_t Headstage)

  Gets the mode for the reference electrode
- CFilterPropertyNet ^ GetFilterProperty (SCUDacqGroupChannelEnumNet GroupID, uint32_t FilterNumber)

  Gets the filter property
- array< CFilterPropertyNet^ > ^ GetFilterProperties (SCUDacqGroupChannelEnumNet GroupID, int filter↩ Configurations_Length)

  Gets multiple filter properties

Events

- OnGetAvailableHeadstages^ GetAvailableHeadstagesEvent `[add, remove, raise]`

  Event fires when the bitmap of available headstages has changed
- OnIsHeadstageAvailable^ IsHeadstageAvailableEvent `[add, remove, raise]`

  Event fires when 'true' if the headstage is connected for the headstage to query has changed

Additional Inherited Members

11.107.1 Detailed Description

CSCUFunctionNet is the class to control the SCU device

11.107.2 Constructor & Destructor Documentation

11.107.2.1 CSCUFunctionNet() `[1/2]`

```
CSCUFunctionNet (
    CMcsUsbNet^ mcsusb,
    CMcsUsbFunctionPointerContainer^ pSCUFunctionPointerContainer )
```

Initializes a new instance of the CSCUFunctionNet class.

11.107.2.2 CSCUFunctionNet() [2/2] CSCUFunctionNet (CMcsUsbNet^ *mcsusb*)

11.107.2.3 ~CSCUFunctionNet() virtual ~CSCUFunctionNet ( ) [virtual]

11.107.2.4 "!CSCUFunctionNet() !CSCUFunctionNet ( )

11.107.3 Member Function Documentation

11.107.3.1 AutomaticAnalogOut() void AutomaticAnalogOut (bool *automatic*)

Sets automatic source channel selection

Parameters

automatic	Automatic

11.107.3.2 EnableAnalogOut() void EnableAnalogOut (bool *enable*)

Enables AnalogOut globally

Parameters

enable	Enable

11.107.3.3 GetAnalogOutADCRange() uint32_t GetAnalogOutADCRange ( )

Gets the analog out ADC range

Returns

Range

11.107.3.4 GetAnalogOutChannels() `uint32_t GetAnalogOutChannels ( uint32_t out_channel )`

Get the connected source channel number for a certain output channel

Parameters

out_channel	Output channel number

Returns

Source channel number

11.107.3.5 GetAnalogOutDACRange() `AnalogOut_DAC_Range_EnumNet GetAnalogOutDACRange ( )`

Gets the analog out DAC range

Returns

Range

11.107.3.6 GetAvailableHeadstages() `uint32_t GetAvailableHeadstages ( )`

Gets a bitmap of available headstages.

Returns

The bitmap of available headstages.

11.107.3.7 GetFilterProperties() `array<CFilterPropertyNet^> ^ GetFilterProperties ( SCUDacqGroupChannelEnumNet GroupID, int filterConfigurations_Length )`

Gets multiple filter properties

Parameters

GroupID	The group ID
filterConfigurations_Length	The maximal length of filterConfigurations.

Returns

array of filter properties

11.107.3.8 GetFilterProperty() `CFilterPropertyNet ^ GetFilterProperty ( SCUDacqGroupChannelEnumNet GroupID, uint32_t FilterNumber )`

Gets the filter property

Parameters

GroupID	The group ID
FilterNumber	The filter number

Returns

The filter property

11.107.3.9 GetHeadstageAdcBits() `uint32_t GetHeadstageAdcBits ( uint32_t Headstage )`

Gets the Number of ADC bits for a given headstage.

Parameters

Headstage	The headstage to query.

Returns

The number of bits the ADC has for the given headstage.

11.107.3.10 GetHeadstageAdcRangeInMicroVolt() `uint32_t GetHeadstageAdcRangeInMicroVolt ( uint32_t Headstage )`

Gets the ADC Range in uV for a given headstage.

Parameters

Headstage	The headstage to query.

Returns

The ADC Range in uV for the given headstage.

11.107.3.11 GetHeadstageDacBits() uint32_t GetHeadstageDacBits (uint32_t *Headstage*)

Gets the Number of DAC bits for a given headstage.

Parameters

Headstage	The headstage to query.

Returns

The number of bits the DAC has for the given headstage.

11.107.3.12 GetHeadstageDacCurrentRangeInMicroAmpere() uint32_t GetHeadstageDacCurrentRangeInMicroAmpere (uint32_t *Headstage*)

Gets the DAC Current Range in uA for a given headstage.

Parameters

Headstage	The headstage to query.

Returns

The DAC Current Range in uA for the given headstage.

11.107.3.13 GetHeadstageDacCurrentResolutionInNanoAmpere() uint32_t GetHeadstageDacCurrentResolutionInNanoAmpere (uint32_t *Headstage*)

Gets the DAC Current Resolution in nA for a given headstage.

Parameters

Headstage	The headstage to query.

Returns

The DAC Current Resolution in nA for the given headstage.

11.107.3.14 GetHeadstageDacVoltageRangeInMilliVolt()

```
uint32_t GetHeadstageDacVoltageRangeInMilliVolt (
            uint32_t Headstage )
```

Gets the DAC Voltage Range in mV for a given headstage.

Parameters

Headstage	The headstage to query.

Returns

The DAC Voltage Range in mV for the given headstage.

11.107.3.15 GetHeadstageDacVoltageResolutionInMicroVolt()

```
uint32_t GetHeadstageDacVoltageResolutionInMicroVolt (
            uint32_t Headstage )
```

Gets the DAC Voltage Resolution in uV for a given headstage.

Parameters

Headstage	The headstage to query.

Returns

The DAC Voltage Resolution in uV for the given headstage.

11.107.3.16 GetHeadstageGainInPermille()

```
uint32_t GetHeadstageGainInPermille (
            uint32_t Headstage )
```

Gets the gain factor in permille for a given headstage.

Parameters

Headstage	The headstage to query.

Returns

The gain factor in permille for the given headstage.

11.107.3.17 GetHeadstageID() SCU_HeadstageIdEnumNet GetHeadstageID (uint32_t *Headstage*)

Gets the headstage fpga ID.

Parameters

Headstage	The headstage to query.

Returns

The headstage fpga ID.

11.107.3.18 GetHeadstageNumberOfAnalogChannels() uint32_t GetHeadstageNumberOfAnalogChannels (uint32_t *Headstage*)

Gets the number of analog channels for a given headstage.

Parameters

Headstage	The headstage to query.

Returns

The number of analog channels the headstage has.

11.107.3.19 GetHeadstageNumberOfStimulationChannels() uint32_t GetHeadstageNumberOfStimulationChannels (uint32_t *Headstage*)

Gets the number of stimulation channels for a given headstage.

Parameters

Headstage	The headstage to query.

Returns

The number of stimulation channels the headstage has.

11.107.3.20 GetHeadstagePowerStateAtStart() `bool GetHeadstagePowerStateAtStart ( uint32_t Headstage )`

Gets the Power Status at SCU Power on of a given headstage.

Parameters

Headstage	The headstage to query.

Returns

The Power State at startup for the given headstage: bool false -> off, bool true -> on.

11.107.3.21 GetHeadstageSamplerate() `uint32_t GetHeadstageSamplerate ( uint32_t Headstage )`

Gets the Samplerate of a given headstage.

Parameters

Headstage	The headstage to query.

Returns

The samplerate in Hz for the given headstage.

11.107.3.22 GetHeadstageSerialNumber() `String ^ GetHeadstageSerialNumber ( uint32_t Headstage )`

Gets the serial number of a given headstage.

Parameters

Headstage	The headstage to query.

Returns

The serial number of the headstage.

11.107.3.23 GetMaxNumberOfHeadstages() uint32_t GetMaxNumberOfHeadstages ( )

Gets the maximal number of headstages.

Returns

The maximal number of headstages.

11.107.3.24 GetMaxStimulusChannelsPerHeadstage() uint32_t GetMaxStimulusChannelsPerHeadstage ( )

Gets the maximal number of stimulation channels a headstage can have.

Returns

The maximal number of stimulation channels a headstage can have.

11.107.3.25 GetReferenceElectrodeMode() ReferenceElectrodeModeEnumNet GetReferenceElectrodeMode (uint32_t *Headstage*)

Gets the mode for the reference electrode

Parameters

Headstage	The headstage number

Returns

The mode

11.107.3.26 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

11.107.3.27 HasAnalogOut() `bool HasAnalogOut ( )`

Has AnalogOut hardware

Returns

Enabled

11.107.3.28 HasGalvanicIsolation() `bool HasGalvanicIsolation ( )`

Has galvanic isolated hardware

Returns

Enabled

11.107.3.29 HasHSPowerSwitch() `bool HasHSPowerSwitch ( )`

Has SCU HS power switch

Returns

Has Switch

11.107.3.30 IsAnalogOutEnabled() `bool IsAnalogOutEnabled ( )`

Is AnalogOut enabled

Returns

Enabled

11.107.3.31 IsAutomaticAnalogOut() `bool IsAutomaticAnalogOut ( )`

Is Automatic source channel selection selected

Returns

Automatic

11.107.3.32 IsHeadstageAvailable() `bool IsHeadstageAvailable ( uint32_t *Headstage* )`

Checks whether the given headstage is available.

Parameters

Headstage	The headstage to query.

Returns

'true' if the headstage is connected.

11.107.3.33 IsHSPowered() `bool IsHSPowered (`

```
uint32_t Headstage )
```

Is the HS powered

Parameters

Headstage	The headstage to query.

Returns

'true' if the headstage is powered.

11.107.3.34 IsInDacqLegacyMode() `bool IsInDacqLegacyMode ( )`

Is the SCU in legacy mode

Returns

Is Enabled

11.107.3.35 OnGetAvailableHeadstages() `delegate void OnGetAvailableHeadstages (`

```
uint32_t AvailableHeadstages )
```

11.107.3.36 OnIsHeadstageAvailable() `delegate void OnIsHeadstageAvailable (`

```
uint32_t Headstage,
bool available )
```

11.107.3.37 PowerHS() `void PowerHS (`

```
uint32_t Headstage,
bool power )
```

Power the HS

Parameters

Headstage	The headstage to query.
power	'true' if the headstage is powered.

11.107.3.38 SetAnalogOutADCRange() `void SetAnalogOutADCRange (`

```
uint32_t range )
```

Sets the analog out ADC range

Parameters

range	Range

11.107.3.39 SetAnalogOutChannels() `void SetAnalogOutChannels (`

```
uint32_t out_channel,
uint32_t source_channel )
```

Set the source channel number for a certain output channel

Parameters

out_channel	Output channel number
source_channel	Source channel number

11.107.3.40 SetAnalogOutDACRange() `void SetAnalogOutDACRange (`

```
AnalogOut_DAC_Range_EnumNet range )
```

Sets the analog out DAC range

Parameters

range	Range

11.107.3.41 SetDacqLegacyMode() `void SetDacqLegacyMode (`

```
bool enable )
```

Enable the SCU legacy mode

Parameters

enable	Enable

11.107.3.42 SetHeadstagePowerStateAtStart() `void SetHeadstagePowerStateAtStart (`

```
uint32_t Headstage,
bool Powerstatus )
```

Sets the Power Status at SCU Power on of a given headstage.

Parameters

Headstage	The headstage number
Powerstatus	The Power State at startup for the given headstage: bool false -> off, bool true -> on.

11.107.3.43 SetReferenceElectrodeMode() `void SetReferenceElectrodeMode (`

```
uint32_t Headstage,
ReferenceElectrodeModeEnumNet NewValue )
```

Sets the mode for the reference electrode

Parameters

Headstage	The headstage number
NewValue	The mode

11.107.3.44 SetReferenceElectrodeSwitchState() `void SetReferenceElectrodeSwitchState (`

```
uint32_t Headstage,
ReferenceElectrodeSwitchPositionEnumNet NewSwitchPos )
```

Sets the position of the switch for the reference electrode

Parameters

Headstage	The headstage number
NewSwitchPos	The switch position

11.107.4 Event Documentation

11.107.4.1 GetAvailableHeadstagesEvent `OnGetAvailableHeadstages^ GetAvailableHeadstagesEvent [add], [remove], [raise]`

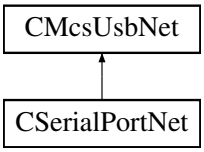
Event fires when the bitmap of available headstages has changed

11.107.4.2 IsHeadstageAvailableEvent `OnIsHeadstageAvailable^ IsHeadstageAvailableEvent [add], [remove], [raise]`

Event fires when 'true' if the headstage is connected for the headstage to query has changed

11.108 CSerialPortNet Class Reference

Inheritance diagram for CSerialPortNet:



Public Member Functions

- CSerialPortNet (void)
- void Send (array< byte >^ buffer)
- void Send (String^ command)
- array< byte > ^ Receive (void)
- array< byte > ^ Receive (int length)
- String ^ ReceiveString (void)
- String ^ ReceiveString (int length)
- int GetBytesAvailable (void)

Additional Inherited Members

11.108.1 Constructor & Destructor Documentation

11.108.1.1 CSerialPortNet() `CSerialPortNet ( void )`

11.108.2 Member Function Documentation

11.108.2.1 GetBytesAvailable()

```
int GetBytesAvailable (
    void  )
```

11.108.2.2 Receive() [1/2]

```
array<byte> ^ Receive (
    int length )
```

11.108.2.3 Receive() [2/2]

```
array<byte> ^ Receive (
    void  )
```

11.108.2.4 ReceiveString() [1/2]

```
String ^ ReceiveString (
    int length )
```

11.108.2.5 ReceiveString() [2/2]

```
String ^ ReceiveString (
    void  )
```

11.108.2.6 Send() [1/2]

```
void Send (
    array< byte >^ buffer )
```

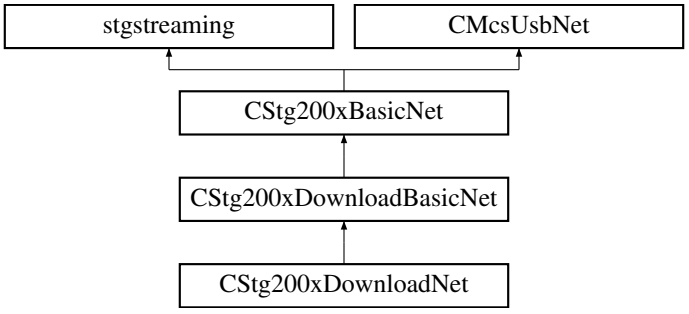
11.108.2.7 Send() [2/2]

```
void Send (
    String^ command )
```

11.109 CStg200xBasicNet Class Reference

Base class for the Stg200x.

Inheritance diagram for CStg200xBasicNet:



Public Member Functions

- virtual ~CStg200xBasicNet ()

  The destructor.
- void SetOutputRate (uint32_t rate)

  Change the output rate of the STG. Valid rates are from 1000 Hz to 50000 Hz.
- uint32_t GetOutputRate ()

  Queries the output rate of the STG. Valid rates are from 1000 Hz to 50000 Hz.
- void SendStart (uint32_t triggermap)

  Start (Trigger) the STG. The startup delay is in the range of a few ms.
- void SendStop (uint32_t triggermap)

  Stop some or all triggers of the STG.
- void SendStop (uint32_t triggermap, int options)

  Stop some or all triggers of the STG.
- void GetStgVersionInfo ([Out]String^% SwVersion, [Out]String^% HwVersion)

  Queries software and hardware version.
- void GetAnalogRanges (int channel, [Out]int% URange, [Out]int% IRange)

  Gets the range of the analog outputs.
- void GetAnalogResolution (int channel, [Out]int% URes, [Out]int% IRes)

  Gets the resolution of the analog outputs.
- virtual int32_t GetDACResolution ()

  Gets number of bits of the DAC resolution.
- virtual int32_t GetVoltageRangeInMicroVolt (uint32_t channel)

  Gets the Voltage Range of the specified channel in Microvolts.
- virtual int32_t GetVoltageResolutionInMicroVolt (uint32_t channel)

  Gets the Voltage Resolution of the specified channel in Microvolts.
- virtual int32_t GetCurrentRangeInNanoAmp (uint32_t channel)

  Gets the Current Range of the specified channel in Nanoamps.
- virtual int32_t GetCurrentResolutionInNanoAmp (uint32_t channel)

  Gets the Current Resolution of the specified channel in Nanoamps.
- void GetStgProgramInfo ([Out]bool% IsProgrammed, [Out]System::Runtime::InteropServices::ComTypes::↩ FILETIME% timestamp, [Out]String^% filename, [Out]Guid% guid)

  Queries Download information from the STG. If download information was stored by the use of SetStgProgramInfo, this function can be used to retrieve it.
- void GetStgProgramInfo ([Out]bool% IsProgrammed, [Out]DateTime% timestamp, [Out]String^% filename, [Out]Guid% guid)

  Queries Download information from the STG. If download information was stored by the use of SetStgProgramInfo, this function can be used to retrieve it.
- void SetStgProgramInfo (DateTime timestamp, String^ filename, Guid guid)

  Store Download information in the STG. This function can be used to store the filename and timestamp of the last download for later query.
- uint32_t GetAvailableMemory ()

  Gets the amount of memory available in the currently selected segment of the STG.
- uint32_t GetTotalMemory ()

  Gets the total amount of memory available on the STG (all segments).
- virtual uint32_t GetNumberOfAnalogChannels ()

  Gets the Number of available analog channels of the device.
- virtual uint32_t GetNumberOfSyncoutChannels ()

  Gets the Number of available syncout channels of the device.
- virtual uint32_t GetNumberOfTriggerInputs ()

  Gets the Number of trigger inputs of the device.
- virtual uint32_t GetNumberOfHWDACPaths ()

Gets the Number of HW Stimulation DACs of the device.

- virtual uint32_t GetNumberOfStimulationSourcesPerElectrode ()

  Gets the number of stimulation sources (DACs) per electrode.
- virtual void SetVoltageMode (unsigned int channel)

  Sets a channel to voltage mode (STG3008-FA and STG400x only).
- virtual void SetCurrentMode (unsigned int channel)

  Sets a channel to current mode (STG3008-FA and STG400x only).
- virtual void SetVoltageMode ()

  Sets all channels to voltage mode (STG3008-FA and STG400x only).
- virtual void SetCurrentMode ()

  Sets all channels to current mode (STG3008-FA and STG400x only).
- virtual uint32_t GetCurrentRangeByIndex (uint32_t channel)

  Gets the currently selected range index for the current output (STG5 only).
- virtual void SetCurrentRangeByIndex (uint32_t channel, uint32_t rangeIndex)

  Sets the range index for the current output (STG5 only).
- virtual void SetMeasurementMode (unsigned int channel)

  Sets a channel to measurement mode (STG3008-FA).
- virtual void SetFAAmplification (unsigned int amplification)
- virtual uint32_t GetFAAmplification ()
- virtual void SetAutocalibrationDisabled (unsigned int channel, bool disable)

  Sets the autocalibration configuration.
- virtual bool GetAutocalibrationDisabled (unsigned int channel)

  Gets the autocalibration configuration.
- virtual void SetElectrodeMode (uint32_t electrode, array< ElectrodeModeEnumNet >^ mode)

  Puts an electrode in either automatic or manual mode.
- virtual void SetElectrodeMode (uint32_t electrode, ElectrodeModeEnumNet mode)

  Puts an electrode in either automatic or manual mode.
- virtual void SetElectrodeMode (uint32_t Scu_HS, uint32_t electrode, array< ElectrodeModeEnumNet >^ mode)

  Puts an electrode in either automatic or manual mode.
- virtual void SetElectrodeMode (uint32_t Scu_HS, uint32_t electrode, ElectrodeModeEnumNet mode)

  Puts an electrode in either automatic or manual mode.
- virtual uint32_t GetElectrodeMode (uint32_t electrode)

  Gets the mode an electrode is in.
- virtual uint32_t GetElectrodeMode (uint32_t Scu_HS, uint32_t electrode)

  Gets the mode an electrode is in.
- virtual void SetElectrodeDacMux (uint32_t electrode, uint32_t 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.

Additional Inherited Members

11.109.1 Detailed Description

Base class for the Stg200x.

From this class all STG related classes are derived: Mcs.Usb.CStg200xDownloadBasicNet Mcs.Usb.CStg200xDownloadNet for Download Mode and Mcs.Usb.CStg200xStreamingNet for Streaming Mode.

CStg200xBasicNet is the base class to control MCS STG device.

11.109.2 Constructor & Destructor Documentation

11.109.2.1 ~CStg200xBasicNet() `virtual ~CStg200xBasicNet ( ) [virtual]`

The destructor.

11.109.3 Member Function Documentation

11.109.3.1 GetAnalogRanges() `void GetAnalogRanges (`

```
int channel,
[Out] int% URange,
[Out] int% IRange )
```

Gets the range of the analog outputs.

Parameters

channel	The channel which is queried.
URange	The Voltage range in mV.
IRange	The Current range in uA.

11.109.3.2 GetAnalogResolution() `void GetAnalogResolution (`

```
int channel,
[Out] int% URes,
[Out] int% IRes )
```

Gets the resolution of the analog outputs.

Parameters

channel	The channel which is queried.
URes	The Voltage resolution in mV.
IRes	The Current resolution in uA.

11.109.3.3 GetAutocalibrationDisabled() `virtual bool GetAutocalibrationDisabled (`

```
unsigned int channel ) [virtual]
```

Gets the autocalibration configuration.

Parameters

channel	The channel number.

Returns

`true` if autocalibration is disabled.

11.109.3.4 GetAvailableMemory() `uint32_t GetAvailableMemory ( )`

Gets the amount of memory available in the currently selected segment of the STG.

Returns

The memory available in the currently selected segment in bytes.

11.109.3.5 GetBlankingEnable() [1/2] `virtual bool GetBlankingEnable ( uint32_t electrode ) [virtual]`

Gets whether an electrode should be blanked while stimulation is in progress.

Parameters

electrode	The electrode number.

Returns

true if blanking is enabled while stimulation is in progress.

11.109.3.6 GetBlankingEnable() [2/2] `virtual bool GetBlankingEnable ( uint32_t Scu_HS, uint32_t electrode ) [virtual]`

Gets whether an electrode should be blanked while stimulation is in progress.

Parameters

Scu_HS	The SCU headstage number.

Parameters

electrode	The electrode number.

Returns

true if blanking is enabled while stimulation is in progress.

11.109.3.7 GetCurrentRangeByIndex() `virtual uint32_t GetCurrentRangeByIndex ( 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.

11.109.3.8 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.

11.109.3.9 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.

11.109.3.10 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.

11.109.3.11 GetDACResolution() `virtual int32_t GetDACResolution ( ) [virtual]`

Gets number of bits of the DAC resolution.

Returns

The DAC resolution in bits.

11.109.3.12 GetDiginValue() `virtual uint32_t GetDiginValue ( ) [virtual]`

Gets the Value on the digital input port.

Returns

The current value on the digital inputs.

11.109.3.13 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.

11.109.3.14 GetDigoutValue() virtual uint32_t GetDigoutValue ( ) [virtual]

Gets the Value on the digital output port.

Returns

The current value on the digital outputs.

11.109.3.15 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.

11.109.3.16 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.

11.109.3.17 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.

11.109.3.18 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.

11.109.3.19 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.

11.109.3.20 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.

11.109.3.21 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.

11.109.3.22 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.

11.109.3.23 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.

11.109.3.24 GetFAAmplification() virtual uint32_t GetFAAmplification ( ) [virtual]

11.109.3.25 GetHeadstage() virtual uint32_t GetHeadstage ( ) [virtual]

11.109.3.26 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".

11.109.3.27 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.

11.109.3.28 GetNumberOfAnalogChannels()

```
virtual uint32_t GetNumberOfAnalogChannels ( ) [virtual]
```

Gets the Number of available analog channels of the device.

Returns

The number of analog channels.

11.109.3.29 GetNumberOfHWDACPaths() `virtual uint32_t GetNumberOfHWDACPaths ( ) [virtual]`

Gets the Number of HW Stimulation DACs of the device.

Returns

The number of independent HW Stimulation outputs.

11.109.3.30 GetNumberOfStimulationElectrodes() `virtual uint32_t GetNumberOfStimulationElectrodes ( ) [virtual]`

11.109.3.31 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.

11.109.3.32 GetNumberOfSyncoutChannels() `virtual uint32_t GetNumberOfSyncoutChannels ( ) [virtual]`

Gets the Number of available syncout channels of the device.

Returns

The number of analog channels.

11.109.3.33 GetNumberOfTriggerInputs() `virtual uint32_t GetNumberOfTriggerInputs ( ) [virtual]`

Gets the Number of trigger inputs of the device.

Returns

The number of trigger inputs.

11.109.3.34 GetOutputRate() `uint32_t GetOutputRate ( )`

Queries the output rate of the STG. Valid rates are from 1000 Hz to 50000 Hz.

Returns

Returns the current output rate in Hz.

11.109.3.35 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.

11.109.3.36 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.

11.109.3.37 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.

11.109.3.38 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.

11.109.3.39 GetTotalMemory() `uint32_t GetTotalMemory ( )`

Gets the total amount of memory available on the STG (all segments).

Returns

The total memory available on the STG in bytes.

11.109.3.40 GetTriggerSource() `virtual TriggerSourceEnumNet GetTriggerSource ( unsigned int triggernum ) [virtual]`

11.109.3.41 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.

11.109.3.42 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.

11.109.3.43 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.

11.109.3.44 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.

11.109.3.45 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.

11.109.3.46 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.

11.109.3.47 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..

11.109.3.48 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.

11.109.3.49 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.

11.109.3.50 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.

11.109.3.51 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.

11.109.3.52 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.

11.109.3.53 SetCurrentMode() [1/2]

```
virtual void SetCurrentMode ( ) [virtual]
```

Sets all channels to current mode (STG3008-FA and STG400x only).

11.109.3.54 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.

11.109.3.55 SetCurrentRangeByIndex() virtual void SetCurrentRangeByIndex (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.

11.109.3.56 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.

11.109.3.57 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.

11.109.3.58 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.

11.109.3.59 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).

11.109.3.60 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).

11.109.3.61 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).

11.109.3.62 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).

11.109.3.63 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.

11.109.3.64 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.

11.109.3.65 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.

11.109.3.66 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.

11.109.3.67 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.

11.109.3.68 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.

11.109.3.69 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.

11.109.3.70 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.

11.109.3.71 SetEnableAmplifierProtectionSwitch() [1/4] virtual void SetEnableAmplifierProtection↩ Switch (

```
uint32_t electrode,
array< bool >^ enable ) [virtual]
```

Defines whether the Amplifier Protection Switch is openend while stimulation is in progress.

Parameters

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.

11.109.3.72 SetEnableAmplifierProtectionSwitch() [2/4] virtual void SetEnableAmplifierProtection↩ Switch (

```
uint32_t electrode,
bool enable ) [virtual]
```

Defines whether the Amplifier Protection Switch is openend while stimulation is in progress.

Parameters

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.

11.109.3.73 SetEnableAmplifierProtectionSwitch() [3/4] virtual void SetEnableAmplifierProtection↩ Switch (

```
uint32_t Scu_HS,
uint32_t electrode,
array< bool >^ enable ) [virtual]
```

Defines whether the Amplifier Protection Switch is openend while stimulation is in progress.

Parameters

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.

11.109.3.74 SetEnableAmplifierProtectionSwitch() [4/4] virtual void SetEnableAmplifierProtection↩ Switch (

```
uint32_t Scu_HS,
uint32_t electrode,
bool enable ) [virtual]
```

Defines whether the Amplifier Protection Switch is openend while stimulation is in progress.

Parameters

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.

11.109.3.75 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.

11.109.3.76 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.

11.109.3.77 SetFAAmplification()

```
virtual void SetFAAmplification (
    unsigned int amplification ) [virtual]
```

11.109.3.78 SetHeadstage()

```
virtual void SetHeadstage (
    unsigned int headstage ) [virtual]
```

11.109.3.79 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".

11.109.3.80 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.

11.109.3.81 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.

11.109.3.82 SetMeasurementMode() `virtual void SetMeasurementMode ( unsigned int channel ) [virtual]`

Sets a channel to measurement mode (STG3008-FA).

Parameters

channel	The channel to change.

11.109.3.83 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.

11.109.3.84 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.

11.109.3.85 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.

11.109.3.86 SetTriggerSource() [1/3] virtual void SetTriggerSource (

```
unsigned int triggernum,
DigitalSource< digitalsourceenum >^ triggersource,
int bitnum_offset ) [virtual]
```

11.109.3.87 SetTriggerSource() [2/3] virtual void SetTriggerSource (

```
unsigned int triggernum,
TriggerSourceEnumNet triggersource ) [virtual]
```

11.109.3.88 SetTriggerSource() [3/3] virtual void SetTriggerSource (

```
unsigned int triggernum,
TriggerSourceEnumNet triggersource,
int bitnum_offset ) [virtual]
```

11.109.3.89 SetVoltageMode() [1/2] virtual void SetVoltageMode ( ) [virtual]

Sets all channels to voltage mode (STG3008-FA and STG400x only).

11.109.3.90 SetVoltageMode() [2/2] virtual void SetVoltageMode (

```
unsigned int channel ) [virtual]
```

Sets a channel to voltage mode (STG3008-FA and STG400x only).

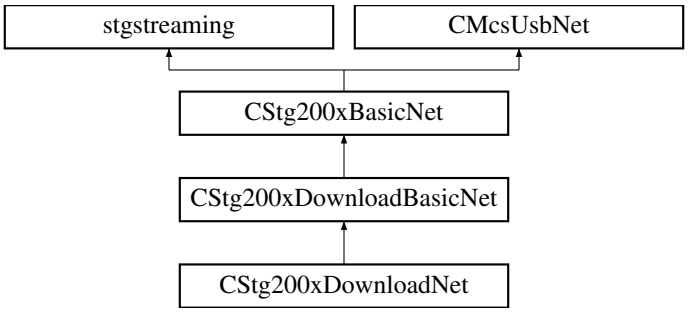
Parameters

channel	The channel to change.

11.110 CStg200xDownloadBasicNet Class Reference

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

Inheritance diagram for CStg200xDownloadBasicNet:



Public Member Functions

- virtual void SetupTrigger (uint32_t first_trigger, array< uint32_t >^ channelmap, array< uint32_t >^ syncoutmap, array< uint32_t >^ repeat)

  Configures the trigger settings for the STG. Note that all memory segments have their own trigger setting.
- virtual void SetupTriggerSingle (uint32_t trigger, uint32_t channelmap, uint32_t syncoutmap, uint32_t repeat)

  Configures the trigger settings for the STG. Note that all memory segments have their own trigger setting.
- void GetTrigger ([Out] array< uint32_t >^% channelmap, [Out] array< uint32_t >^% syncoutmap, [Out] array< uint32_t >^% repeat)

  Queries the trigger settings for the STG. Note that all memory segments have their own trigger setting.
- void GetSweepCount ([Out] array< uint32_t >^% sweeps, [Out] array< uint32_t >^% triggers)

  Get the sweep and trigger count of the STG.
  - *The triggercount tells how many times each trigger was active and is reset to zero on download of new channel data.*
  - *The sweepcount tells how many times each trigger was already repeated. This count is set to zero on trigger and counts up to repeat in CStg200xDownloadBasicNet::SetupTrigger.*
- void ForceStatusEvent ()

  Force a status event.
- void ResetStatus (uint32_t triggermap)

  Reset the status flag.
- uint32_t GetMemoryUsageDAC (uint32_t Channel)

  Queries the memory usage of the current segment and selected analog DAC channel.
- uint32_t GetMemoryUsageSyncout (uint32_t Channel)

  Queries the memory usage of the current segment and selected syncout channel.
- virtual void ClearSyncData (uint32_t channel)

  Delete a SyncOut pattern for a channel from STG memory.
- virtual void SendSyncData (uint32_t channel, array< uint16_t >^ pData, array< uint64_t >^ tData)

  Uploads sync output data to the STG.
  Sends sync output data to a given channel on the STG. The list of datapoints will be sent to the selected sync output channel. Sync output data previously sent to the channel is overwritten.
  Each datapoint is represented by an integer value and can be either 0 or 1.
  The duration is given as a list of 64 bit integers. Durations are given in units of 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

11.110.1 Detailed Description

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

11.110.2 Member Function Documentation

11.110.2.1 ClearChannelData() `virtual void ClearChannelData ( uint32_t channel ) [virtual]`

Delete a stimulus pattern for a channel from STG memory

Parameters

channel	Specifies the channel to clear.

11.110.2.2 ClearSyncData() virtual void ClearSyncData (uint32_t *channel*) [virtual]

Delete a SyncOut pattern for a channel from STG memory.

Parameters

channel	Specifies the syncout channel to clear.

11.110.2.3 DisableAutoReset() virtual void DisableAutoReset ( ) [virtual]

Disable AutoReset of the STG Status.

If autoreset is disabled, the STG status switches to FINISHED after the defined number of sweeps is finished. To switch back to the IDLE status, use CStg200xDownload::ResetStatus()

11.110.2.4 EnableAutoReset() virtual void EnableAutoReset ( ) [virtual]

Enable AutoReset of the STG Status.

This is the default on power up. If autoreset is enabled, the STG status switches to FINISHED only for one poll cycle after this, it switches to IDLE automatically.

11.110.2.5 ForceStatusEvent() void ForceStatusEvent ( )

Force a status event.

Force the DLL to create a PollMessage event and to call the pPollCallback function, even if no new status information is available.

11.110.2.6 GetMemoryUsageDAC() uint32_t GetMemoryUsageDAC (uint32_t *Channel*)

Queries the memory usage of the current segment and selected analog DAC channel.

The currently used memory is reported for the requested channel.

Parameters

Channel	channel for the amount of interested usage.

Returns

Returns the usage in STG memory.

11.110.2.7 GetMemoryUsageSyncout() `uint32_t GetMemoryUsageSyncout (`

```
uint32_t Channel )
```

Queries the memory usage of the current segment and selected syncout channel.

The currently used memory is reported for the requested channel.

Parameters

Channel	channel for the amount of interested usage.

Returns

Returns the usage in STG memory.

11.110.2.8 GetSweepCount() `void GetSweepCount (`

```
[Out] array< uint32_t >^% sweeps,
[Out] array< uint32_t >^% triggers )
```

Get the sweep and trigger count of the STG.

- The triggercount tells how many times each trigger was active and is reset to zero on download of new channel data.
- The sweepcount tells how many times each trigger was already repeated. This count is set to zero on trigger and counts up to repeat in CStg200xDownloadBasicNet::SetupTrigger.

Parameters

sweeps	on return contains the number of sweeps for each trigger.
triggers	on return contains the number of trigger events seen for each trigger.

11.110.2.9 GetTrigger() `void GetTrigger (`

```
[Out] array< uint32_t >^% channelmap,
[Out] array< uint32_t >^% syncoutmap,
[Out] array< uint32_t >^% repeat )
```

Queries the trigger settings for the STG. Note that all memory segments have their own trigger setting.

Parameters

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.

11.110.2.10 ResetStatus() `void ResetStatus ( uint32_t triggermap )`

Reset the status flag.

Parameters

triggermap	bitmap of trigger for which to reset the status.

11.110.2.11 SendChannelData() `virtual void SendChannelData ( uint32_t channel, array< uint16_t >^ pData, array< uint64_t >^ tData ) [virtual]`

Uploads analog data (stimulus patterns) to the STG.

Sends datapoints to a given channel on the STG. The list of datapoints will be sent to the selected channel. Data previously sent to the channel is overwritten.

Each datapoint is represented by an integer value in the range from 0 to 4095 (bit 0 to 11), its sign is taken from bit 12, 0 is for positive amplitude, and 1 for negative amplitude Bits 13 to 15 have to be zero.

The duration is given as a list of 64 bit integers. Durations are given in units of 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.

11.110.2.12 SendSyncData() `virtual void SendSyncData ( uint32_t channel, array< uint16_t >^ pData, array< uint64_t >^ tData ) [virtual]`

Uploads sync output data to the STG.

Sends sync output data to a given channel on the STG. The list of datapoints will be sent to the selected sync output channel. Sync output data previously sent to the channel is overwritten.

Each datapoint is represented by an integer value and can be either 0 or 1.

The duration is given as a list of 64 bit integers. Durations are given in units of 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.

11.110.2.13 SetupRetriggerMode() [1/2]

```
virtual void SetupRetriggerMode (
    int8_t trigger,
    RetriggerActionEnumNet same_trigger,
    RetriggerActionEnumNet other_trigger ) [virtual]
```

Define the action on triggers while the STG is running.

The STG has three options how to handle a successive trigger while a trigger is active.

- stop this trigger (default action)
- restart this trigger
- ignore the signal

Parameters

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.

11.110.2.14 SetupRetriggerMode() [2/2]

```
virtual void SetupRetriggerMode (
    RetriggerActionEnumNet same_trigger,
    RetriggerActionEnumNet other_trigger ) [virtual]
```

Define the action on triggers while the STG is running.

The STG has three options how to handle a successive trigger while a trigger is active.

- stop this trigger (default action)
- restart this trigger
- ignore the signal

Parameters

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.

11.110.2.15 SetupTrigger()

```
virtual void SetupTrigger (
    uint32_t first_trigger,
    array< uint32_t >^ channelmap,
    array< uint32_t >^ syncoutmap,
    array< uint32_t >^ repeat )  [virtual]
```

Configures the trigger settings for the STG. Note that all memory segments have their own trigger setting.

Parameters

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.

11.110.2.16 SetupTriggerSingle()

```
virtual void SetupTriggerSingle (
    uint32_t trigger,
    uint32_t channelmap,
    uint32_t syncoutmap,
    uint32_t repeat )  [virtual]
```

Configures the trigger settings for the STG. Note that all memory segments have their own trigger setting.

Parameters

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.

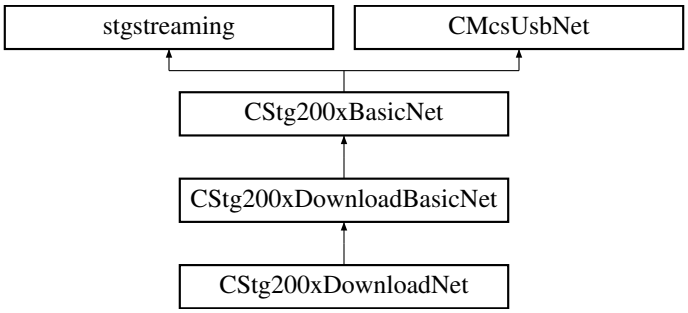
11.110.3 Property Documentation

11.110.3.1 Stimulus `CStimulusFunctionNet^ Stimulus [get]`

11.111 CStg200xDownloadNet Class Reference

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

Inheritance diagram for CStg200xDownloadNet:



Public Member Functions

- CStg200xDownloadNet ()

  Use this constructor if you do not want to use the status callback.
- CStg200xDownloadNet (OnStgPollStatus^ pollStatus)

  Use this constructor if you want to use the status callback.
- ~CStg200xDownloadNet ()

- void PrepareAndSendData (uint32_t channel, array< int32_t >^ amplitude, array< uint64_t >^ duration, STG_DestinationEnumNet destType)

  Prepare and send data to a given channel on the STG. Previous data sent to that channel is erased first.
- void PrepareAndAppendData (uint32_t channel, array< int32_t >^ amplitude, array< uint64_t >^ duration, STG_DestinationEnumNet destType)

  Prepare and append data to a given channel on the STG.
- void ClearChannel_PrepareAndSendData (uint32_t channel, array< int32_t >^ amplitude, array< uint64_t >^ duration, STG_DestinationEnumNet destType, bool doClear)

  Prepare and append data to a given channel on the STG.
- void SendSegmentDefine (array< uint32_t >^ segment_list)

  Defines the segment memory layout of the STG.
- void SendSegmentStart (uint32_t triggermap, uint32_t segment, Stg200xSegmentFlagsEnumNet segmentflags)

  Switchs segment and starts trigger.
- void SendSegmentSelect (uint32_t segment, Stg200xSegmentFlagsEnumNet segmentflags)

  Switchs segment.
- void EnableMultiFileMode (uint32_t submode)

  Enable the Multi-File mode of the STG.
- void DisableMultiFileMode ()

  Disable the Multi-File mode of the STG
- StgStatusNet ^ QueryTriggerstatus ()
- void SetOutputMap (array< uint32_t >^ ChannelLayout)
- int32_t GetModuleTemp (unsigned int channel)
- uint32_t GetModuleCurrent (unsigned int channel)

Events

- OnStgPollStatus^ Stg200xPollStatusEvent `[add, remove, raise]`
- OnMwPollStatus^ MwPollStatusEvent `[add, remove, raise]`

Additional Inherited Members

11.111.1 Detailed Description

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

11.111.2 Constructor & Destructor Documentation

11.111.2.1 CStg200xDownloadNet() [1/2] CStg200xDownloadNet ( )

Use this constructor if you do not want to use the status callback.

11.111.2.2 CStg200xDownloadNet() [2/2] CStg200xDownloadNet (OnStgPollStatus^ *pollStatus*)

Use this constructor if you want to use the status callback.

11.111.2.3 ~CStg200xDownloadNet() ~CStg200xDownloadNet ( )

11.111.3 Member Function Documentation

11.111.3.1 ClearChannel_PrepareAndSendData() void ClearChannel_PrepareAndSendData (uint32_t *channel*, array< int32_t >^ *amplitude*, array< uint64_t >^ *duration*, STG_DestinationEnumNet *destType*, bool *doClear*)

Prepare and append data to a given channel on the STG.

Each datapoint is represented by an signed 32bit integer value. When using voltage stimulation, the values are in multiple of 1 uV, thus the possible range is += 2000 V. When using current stimulation, the values are in multiple of 1 nA, this the possible range is += 2000 mA.

The duration is given as a list of 64 bit integers. Durations are given in units of 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.

11.111.3.2 DisableMultiFileMode() void DisableMultiFileMode ( )

Disable the Multi-File mode of the STG

Switch the STG back to normal mode. In this mode, trigger inputs are assigned to channels, not to segments.

11.111.3.3 EnableMultiFileMode() void EnableMultiFileMode (uint32_t *submode*)

Enable the Multi-File mode of the STG.

In Multi-File mode, the trigger inputs switch between segments. To use this mode, define up to as many segments as trigger inputs are available and fill each segment with a stimulus pattern.

Now a trigger on trigger input 1 switches the STG to the first segment and starts all triggers in this segment. Likewise, a trigger on trigger input 2, 3 and 4 selects the respective segment and start all triggers in this segment So the Multi-File Mode can be used to predefine up to four different stimuli which can be selected without the need for a computer connection.

Parameters

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.

11.111.3.4 GetModuleCurrent() uint32_t GetModuleCurrent (unsigned int *channel*)

11.111.3.5 GetModuleTemp() int32_t GetModuleTemp (unsigned int *channel*)

11.111.3.6 PrepareAndAppendData() void PrepareAndAppendData (uint32_t *channel*, array< int32_t >^ *amplitude*, array< uint64_t >^ *duration*, STG_DestinationEnumNet *destType*)

Prepare and append data to a given channel on the STG.

Each datapoint is represented by an signed 32bit integer value. When using voltage stimulation, the values are in multiple of 1 uV, thus the possible range is += 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.

11.111.3.7 PrepareAndSendData()

```
void PrepareAndSendData (
    uint32_t channel,
    array< int32_t >^ amplitude,
    array< uint64_t >^ duration,
    STG_DestinationEnumNet destType )
```

Prepare and send data to a given channel on the STG. Previous data sent to that channel is erased first.

Each datapoint is represented by an signed 32bit integer value. When using voltage stimulation, the values are in multiple of 1 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.

11.111.3.8 QueryTriggerstatus() `StgStatusNet ^ QueryTriggerstatus ( )`

11.111.3.9 SendSegmentDefine() `void SendSegmentDefine ( array< uint32_t >^ segment_list )`

Defines the segment memory layout of the STG.

On reset, the STG has one segment containing all available memory.

With this command, the STG memory can be devided into several segments. Each segment can be filled with stimulus data.

Parameters

segment_list	The List of memory sizes (one per segment).

11.111.3.10 SendSegmentSelect() `void SendSegmentSelect ( 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.

11.111.3.11 SendSegmentStart()

```
void SendSegmentStart (
    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.

11.111.3.12 SetOutputMap()

```
void SetOutputMap (
    array< uint32_t >^ ChannelLayout )
```

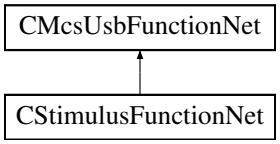
11.111.4 Event Documentation

11.111.4.1 MwPollStatusEvent `OnMwPollStatus^ MwPollStatusEvent [add], [remove], [raise]`

11.111.4.2 Stg200xPollStatusEvent `OnStgPollStatus^ Stg200xPollStatusEvent [add], [remove], [raise]`

11.112 CStimulusFunctionNet Class Reference

Inheritance diagram for CStimulusFunctionNet:



Classes

- class SidebandData
- class StimulusDeviceDataAndUnrolledData

Public Member Functions

- CStimulusFunctionNet (CMcsUsbNet^ 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.
- void SetupTrigger (uint32_t first_trigger, array< uint32_t >^ channelmap, array< uint32_t >^ syncoutmap, array< uint32_t >^ repeat)

  Configures the trigger settings for the STG. Note that all memory segments have their own trigger setting.
- void SetupTriggerSingle (uint32_t trigger, uint32_t channelmap, uint32_t syncoutmap, uint32_t repeat)

  Configures the trigger settings for the STG. Note that all memory segments have their own trigger setting.
- uint32_t GetTotalMemory ()

  Get the total amount of memory available on the STG (all segments).
- uint32_t GetAvailableMemory ()

  Get the amount of memory available in the currently selected segment of the STG.
- int GetNumberOfAnalogChannels ()

  Get the number of STG channels.

Events

- OnStgPollStatus^ PollStatusEvent

Additional Inherited Members

11.112.1 Constructor & Destructor Documentation

11.112.1.1 CStimulusFunctionNet() [1/2] CStimulusFunctionNet (

```
CMcsUsbNet^ mcsusb,
CMcsUsbFunctionPointerContainer^ stimulusFunctionPointerContainer )
```

11.112.1.2 CStimulusFunctionNet() [2/2] CStimulusFunctionNet (

```
CMcsUsbNet^ mcsusb )
```

11.112.2 Member Function Documentation

11.112.2.1 ClearChannel_PrepareAndSendData() void ClearChannel_PrepareAndSendData (

```
uint32_t channel,
array< int32_t >^ amplitude,
array< uint64_t >^ duration,
STG_DestinationEnumNet destType,
bool doClear )
```

11.112.2.2 ClearChannelData() void ClearChannelData (

```
int channel )
```

Delete a Stimulus Pattern from STG memory

Parameters

channel	specifies the channel to clear.

11.112.2.3 ClearMultiplexedData() void ClearMultiplexedData ( )

Clears the Stimulation Memory in the STG device.

11.112.2.4 ClearSyncData() void ClearSyncData (

```
int channel )
```

Delete a Syncout Pattern from STG memory

Parameters

channel	specifies the channel to clear.

11.112.2.5 CreateSideband()

```
SidebandData ^ CreateSideband (
    array< int32_t >^ StimulusActive,
    array< int32_t >^ Syncout,
    array< uint64_t >^ Duration,
    uint32_t Bit0Time,
    uint32_t Bit3Time,
    uint32_t Bit4Time )
```

Creates the Sideband Channel for the MEA2100 device.

Each datapoint is represented by an signed 32bit integer value. A value 0 means that the stimulation is active during that time. A value 1 means that the stimulation is not active during that time.

The duration is given as a list of 64 bit integers. Durations are given in units of 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.

11.112.2.6 ForceStatusEvent() `void ForceStatusEvent ( )`

Force a status event. Force the DLL to create a PollMessage event and to call the pPollCallback function, even if no new status information is available.

11.112.2.7 GetAvailableMemory() `uint32_t GetAvailableMemory ( )`

Get the amount of memory available in the currently selected segment of the STG.

Returns

The total memory available on the STG in bytes.

11.112.2.8 GetCurrentRangeInNanoAmp() `int GetCurrentRangeInNanoAmp ( uint32_t channel )`

Gets the Current Range of the specified channel in Nanoamps.

Parameters

channel	Channel which is queried.

Returns

The Current Range of the specified channel in Nanoamps.

11.112.2.9 GetCurrentResolutionInNanoAmp() `int GetCurrentResolutionInNanoAmp ( uint32_t channel )`

Gets the Current Resolution of the specified channel in Nanoamps.

Parameters

channel	Channel which is queried.

Returns

The Current Resolution of the specified channel in Nanoamps.

11.112.2.10 GetDACResolution() `int GetDACResolution ( )`

Gets number of bits of the DAC resolution.

Returns

The DAC resolution in bits.

11.112.2.11 GetMultiplexedDataChannelsInBlock() `int GetMultiplexedDataChannelsInBlock ( )`

Gets the number of stimulus data channels to send per time slice. Might be greater than the number of configured channels. Fill unused channels with dummy data in SendMultiplexedData

11.112.2.12 GetNumberOfAnalogChannels() `int GetNumberOfAnalogChannels ( )`

Get the number of STG channels.

Returns

The number of STG channels.

11.112.2.13 GetTotalMemory() `uint32_t GetTotalMemory ( )`

Get the total amount of memory available on the STG (all segments).

Returns

The total memory available on the STG in bytes.

11.112.2.14 GetVoltageRangeInMicroVolt() `int GetVoltageRangeInMicroVolt ( uint32_t channel )`

Gets the Voltage Range of the specified channel in Microvolts.

Parameters

channel	Channel which is queried.

Returns

The Voltage Range of the specified channel in Microvolts.

11.112.2.15 GetVoltageResolutionInMicroVolt() `int GetVoltageResolutionInMicroVolt ( uint32_t channel )`

Gets the Voltage Resolution of the specified channel in Microvolts.

Parameters

channel	Channel which is queried.

Returns

The Voltage Resolution of the specified channel in Microvolts.

11.112.2.16 PrepareAndAppendData() `void PrepareAndAppendData ( uint32_t channel, array< int32_t >^ amplitude, array< uint64_t >^ duration, STG_DestinationEnumNet destType )`

Prepare and append data to a given channel on the STG.

Each datapoint is represented by an signed 32bit integer value. When using voltage stimulation, the values are in multiple of 1 uV, thus the possible range is += 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.

11.112.2.17 PrepareAndSendData()

```
void PrepareAndSendData (
    uint32_t channel,
    array< int32_t >^ amplitude,
    array< uint64_t >^ duration,
    STG_DestinationEnumNet destType )
```

Prepare and send data to a given channel on the STG. Previous data sent to that channel is erased first.

Each datapoint is represented by an signed 32bit integer value. When using voltage stimulation, the values are in multiple of 1 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.

11.112.2.18 PrepareData()

```
StimulusDeviceDataAndUnrolledData ^ PrepareData (
    int channel,
    array< int32_t >^ amplitude,
    array< uint64_t >^ duration,
    STG_DestinationEnumNet destType )
```

11.112.2.19 SendMultiplexedData()

```
void SendMultiplexedData (
    array< uint16_t >^ data )
```

Sends stimulus data in multiplexed form. All 16 bits words for the enabled DAC and digital channels are muxed together per time slice.

Parameters

data	Array of data to be sent.

11.112.2.20 SendPreparedData()

```
void SendPreparedData (
    int channel,
    StimulusDeviceDataAndUnrolledData^ device_data_and_unrolled,
    STG_DestinationEnumNet destType )
```

11.112.2.21 SendStart()

```
void SendStart (
    uint32_t triggermap )
```

Start (Trigger) the STG. The startup delay is in the range of a few ms.

Parameters

triggermap	A bitmap of triggers which will be started.

11.112.2.22 SendStop() [1/2] void SendStop (uint32_t *triggermap*)

Stop some or all triggers of the STG.

Parameters

triggermap	A bitmap of triggers which will be stopped.

11.112.2.23 SendStop() [2/2] void SendStop (uint32_t *triggermap*, int *options*)

Stop some or all triggers of the STG.

Parameters

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..

11.112.2.24 SetupTrigger() void SetupTrigger (uint32_t *first_trigger*, array< uint32_t >^ *channelmap*, array< uint32_t >^ *syncoutmap*, array< uint32_t >^ *repeat*)

Configures the trigger settings for the STG. Note that all memory segments have their own trigger setting.

Parameters

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.

11.112.2.25 SetupTriggerSingle() `void SetupTriggerSingle (`

```
uint32_t trigger,
uint32_t channelmap,
uint32_t syncoutmap,
uint32_t repeat )
```

Configures the trigger settings for the STG. Note that all memory segments have their own trigger setting.

Parameters

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.

11.112.2.26 StartPoll() `void StartPoll ( )`

Starts the interrupt fetching thread and delivers events

11.112.2.27 StopPoll() `void StopPoll ( )`

Stops the interrupt fetching thread and delivers events

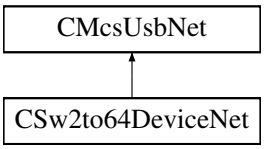
11.112.3 Event Documentation

11.112.3.1 PollStatusEvent `OnStgPollStatus^ PollStatusEvent`

11.113 CSw2to64DeviceNet Class Reference

The class to control the MCS-USB-Sw2to64 device.

Inheritance diagram for CSw2to64DeviceNet:



Public Member Functions

- CSw2to64DeviceNet ()
- ~CSw2to64DeviceNet ()
- unsigned short GetNumber ()

  Gets the number of channels that can be switched in this box.
- array< unsigned char > ^ GetChannels ()

  Gets the current switch positions as char array.
- void SetChannels (array< unsigned char >^ pattern)

  Sets the switch positions from a char array.
- unsigned char GetChannel (unsigned short index)

  Gets one current switch position.
- void SetChannel (unsigned short index, unsigned char pattern)

  Sets one switch position.

Additional Inherited Members

11.113.1 Detailed Description

The class to control the MCS-USB-Sw2to64 device.

This class controls the settings of the MCS-USB-Sw2to64. The box has two inputs for signals. Each of the 64 outputs can be connected to one of the input signals, could be held open or connected ground. Valid switch states are 0, 1, 2 or 3 for each of the settings.

11.113.2 Constructor & Destructor Documentation

11.113.2.1 CSw2to64DeviceNet() `CSw2to64DeviceNet ( )`

11.113.2.2 ~CSw2to64DeviceNet() `~CSw2to64DeviceNet ( )`

11.113.3 Member Function Documentation

11.113.3.1 GetChannel() `unsigned char GetChannel ( unsigned short index )`

Gets one current switch position.

Parameters

in	*index*	number of channel to read the switch position from

Returns

switch position of desired channel

11.113.3.2 GetChannels() `array<unsigned char> ^ GetChannels ( )`

Gets the current switch positions as char array.

Returns

array of char with the size of the number of channels, each char has the setting of a channel

11.113.3.3 GetNumber() `unsigned short GetNumber ( )`

Gets the number of channels that can be switched in this box.

The box can have a different number of channels it can switch. Up to now usually 64 channels are returned

11.113.3.4 SetChannel() `void SetChannel ( unsigned short index, unsigned char pattern )`

Sets one switch position.

Parameters

in	*index*	number of channel to write the switch position to
in	*pattern*	switch position of the channel

11.113.3.5 SetChannels() `void SetChannels ( array< unsigned char >^ pattern )`

Sets the switch positions from a char array.

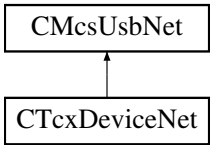
Parameters

in	*pattern*	array of char with the size of the number of channels, each char has the setting of a channel

11.114 CTcxDeviceNet Class Reference

Class to control a Temperature Controller (TCX)

Inheritance diagram for CTcxDeviceNet:



Public Member Functions

- CTcxDeviceNet ()

  Initializes a new instance of CTcxDeviceNet class.
- ~CTcxDeviceNet ()
- unsigned int GetNumControlChannels ()

  Gets the number of channels the device can control/regulate.
- unsigned int GetNumMeasureChannels ()

  Gets the number of channels the device can measure.
- int GetValue (unsigned int channel)

  Gets the temperate of the specified channel in units of 0.1 °C.
- int GetValueHires (unsigned int channel)

  Gets the temperate of the specified channel in units of 0.01 °C.
- int GetHeaterTemp (unsigned int channel)

  Gets the temperate of the specified heater in units of 0.1 °C.
- int GetHeaterLimit (unsigned int device)

  Gets the temperate limit of the specified heater in units of 0.1 °C.
- double GetMaxHeaterPowerMultiwell ()

  queries the max. heater power that the Multiwell temperature controller will apply; unit: W; useful range: 5.2W..7.6W

- void SetMaxHeaterPowerMultiwell (double MaxPowerWatt)

  sets the max. heater power that the Multiwell temperature controller will apply; unit: W; useful range: 5.2W..7.6W
- bool GetHasThermocouple ()

  Gets weather the device supports a thermocouple.
- bool GetEnableHeaterLimit (unsigned int device)
- bool GetEnableThermocouple (unsigned int device)
- TcxSensorTypeEnumNet GetSensorType (unsigned int device)
- String ^ GetUnit (unsigned int channel)
- unsigned int GetBoardTemp ()

  Gets the temperate of the mainboard in units of 0.1 ℃.
- 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

11.114.1 Detailed Description

Class to control a Temperature Controller (TCX)

11.114.2 Constructor & Destructor Documentation

11.114.2.1 CTcxDeviceNet() CTcxDeviceNet ( )

Initializes a new instance of CTcxDeviceNet class.

11.114.2.2 ~CTcxDeviceNet() ~CTcxDeviceNet ( )

11.114.3 Member Function Documentation

11.114.3.1 CalibrateThermocouple() void CalibrateThermocouple (unsigned int *channel*)

11.114.3.2 FactoryReset() void FactoryReset ( )

11.114.3.3 GetBoardTemp() unsigned int GetBoardTemp ( )

Gets the temperate of the mainboard in units of 0.1℃.

11.114.3.4 GetCalibration() int GetCalibration (unsigned int *channel*)

11.114.3.5 GetCalibrationDecp() int GetCalibrationDecp (unsigned int *channel*)

11.114.3.6 GetCalibrationMax() int GetCalibrationMax (unsigned int *channel*)

11.114.3.7 GetCalibrationMin() int GetCalibrationMin (unsigned int *channel*)

11.114.3.8 GetCurrent()

```
int GetCurrent (
    unsigned int channel )
```

11.114.3.9 GetD()

```
int GetD (
    unsigned int device )
```

Gets the D-coefficient of the specified device.

11.114.3.10 GetDDecp()

```
int GetDDecp (
    unsigned int device )
```

11.114.3.11 GetDevice()

```
int GetDevice (
    unsigned int channel )
```

11.114.3.12 GetDeviceType()

```
TcxDeviceTypeEnumNet GetDeviceType ( )
```

11.114.3.13 GetDevname()

```
String ^ GetDevname (
    unsigned int device )
```

11.114.3.14 GetDMax()

```
int GetDMax (
    unsigned int device )
```

11.114.3.15 GetDMin()

```
int GetDMin (
    unsigned int device )
```

11.114.3.16 GetDuty()

```
int GetDuty (
    unsigned int channel )
```

Gets the duty cycle of the heating element.

Parameters

channel	The channel number.

Returns

The duty cycle in percent, the value of 320 ∗ 64 corresponds to 100 %.

11.114.3.17 GetEnableHeaterLimit() `bool GetEnableHeaterLimit (`

`unsigned int` *device* `)`

11.114.3.18 GetEnableThermocouple() `bool GetEnableThermocouple (`

`unsigned int` *device* `)`

11.114.3.19 GetHasThermocouple() `bool GetHasThermocouple ( )`

Gets weather the device supports a thermocouple.

11.114.3.20 GetHeaterLimit() `int GetHeaterLimit (`

`unsigned int` *device* `)`

Gets the temperate limit of the specified heater in units of 0.1 ℃.

11.114.3.21 GetHeaterTemp() `int GetHeaterTemp (`

`unsigned int` *channel* `)`

Gets the temperate of the specified heater in units of 0.1 ℃.

11.114.3.22 GetI() `int GetI (`

`unsigned int` *device* `)`

Gets the I-coefficient of the specified device.

11.114.3.23 GetIDecp() int GetIDecp (unsigned int *device*)

11.114.3.24 GetIMax() int GetIMax (unsigned int *device*)

11.114.3.25 GetIMin() int GetIMin (unsigned int *device*)

11.114.3.26 GetIOut() int GetIOut (unsigned int *channel*)

Gets the current through the heating element.

Parameters

channel	The channel number.

Returns

The current in units of mA.

11.114.3.27 GetMaxHeaterPowerMultiwell() double GetMaxHeaterPowerMultiwell ( )

queries the max. heater power that the Multiwell temperature controller will apply; unit: W; useful range: 5.2W..7.6W

11.114.3.28 GetMaxP() int GetMaxP (unsigned int *device*)

Gets the maximum heater power of the specified device.

11.114.3.29 GetMaxpDecp() int GetMaxpDecp (unsigned int *device*)

11.114.3.30 GetMaxpMax() int GetMaxpMax (unsigned int *device*)

11.114.3.31 GetMaxpMin() int GetMaxpMin (unsigned int *device*)

11.114.3.32 GetNumControlChannels() unsigned int GetNumControlChannels ( )

Gets the number of channels the device can control/regulate.

11.114.3.33 GetNumDevices() unsigned int GetNumDevices ( )

11.114.3.34 GetNumMeasureChannels() unsigned int GetNumMeasureChannels ( )

Gets the number of channels the device can measure.

11.114.3.35 GetOnOff() int GetOnOff (unsigned int *channel*)

Gets if the specified channel is on or off.

11.114.3.36 GetP() int GetP (unsigned int *device*)

Gets the P-coefficient of the specified device.

11.114.3.37 GetPDecp() int GetPDecp (unsigned int *device*)

11.114.3.38 GetPMax() int GetPMax (unsigned int *device*)

11.114.3.39 GetPMin() int GetPMin (unsigned int *device*)

11.114.3.40 GetPOut() int GetPOut (unsigned int *channel*)

Gets the output power of the heating element.

Parameters

channel	The channel number.

Returns

The resistance in units of mW.

11.114.3.41 GetPwrOut() `int GetPwrOut (`

`unsigned int` *channel* `)`

11.114.3.42 GetPwrSet() `int GetPwrSet (`

`unsigned int` *channel* `)`

11.114.3.43 GetRes1() `int GetRes1 (`

`unsigned int` *channel* `)`

11.114.3.44 GetRes2() `int GetRes2 (`

`unsigned int` *channel* `)`

11.114.3.45 GetResS() `int GetResS (`

`unsigned int` *channel* `)`

11.114.3.46 GetResX() `int GetResX (`

`unsigned int` *channel* `)`

11.114.3.47 GetROut() `int GetROut (`

`unsigned int` *channel* `)`

Gets the resistance of the heating element.

Parameters

channel	The channel number.

Returns

The resistance in units of 0.1 Ohm.

11.114.3.48 GetSensorType() TcxSensorTypeEnumNet GetSensorType (unsigned int *device*)

11.114.3.49 GetSetpoint() int GetSetpoint (unsigned int *channel*)

Gets the target temperate of specified channel in units of 0.1°C.

11.114.3.50 GetSetpointDecp() int GetSetpointDecp (unsigned int *channel*)

11.114.3.51 GetSetpointMax() int GetSetpointMax (unsigned int *channel*)

11.114.3.52 GetSetpointMin() int GetSetpointMin (unsigned int *channel*)

11.114.3.53 GetThermocoupleCalibration() int GetThermocoupleCalibration (unsigned int *channel*)

11.114.3.54 GetThermocoupleNanovoltPerKelvin() unsigned int GetThermocoupleNanovoltPerKelvin (unsigned int *channel*)

Gets the proportional constant for the thermocouple.

Parameters

channel	Thermocouple channel number.

Returns

The proportional constant in Nanovolt per Kelvin.

11.114.3.55 GetThermocoupleReferenceTemp() `int GetThermocoupleReferenceTemp (`

`unsigned int` *channel* `)`

11.114.3.56 GetThermocoupleTemp() `int GetThermocoupleTemp (`

`unsigned int` *channel* `)`

11.114.3.57 GetThermocoupleTempAbs() `int GetThermocoupleTempAbs (`

`unsigned int` *channel* `)`

11.114.3.58 GetUnit() `String ^ GetUnit (`

`unsigned int` *channel* `)`

11.114.3.59 GetUOut() `int GetUOut (`

`unsigned int` *channel* `)`

Gets the voltage on the heating element.

Parameters

channel	The channel number.

Returns

The voltage in units of mV.

11.114.3.60 GetValue()

```
int GetValue (
    unsigned int channel )
```

Gets the temperate of the specified channel in units of 0.1℃.

11.114.3.61 GetValueHires()

```
int GetValueHires (
    unsigned int channel )
```

Gets the temperate of the specified channel in units of 0.01℃.

11.114.3.62 GetVolti()

```
unsigned int GetVolti (
    unsigned int channel )
```

11.114.3.63 SetCalibration()

```
void SetCalibration (
    unsigned int channel,
    int calib )
```

11.114.3.64 SetD()

```
void SetD (
    unsigned int device,
    int d_coeff )
```

Sets the D-coefficient of the specified device.

11.114.3.65 SetDevice()

```
void SetDevice (
    unsigned int channel,
    int device )
```

11.114.3.66 SetDeviceType()

```
void SetDeviceType (
    TcxDeviceTypeEnumNet devicetype )
```

11.114.3.67 SetDevname()

```
void SetDevname (
    unsigned int device,
    String^ Devicename )
```

11.114.3.68 SetEnableHeaterLimit()

```
void SetEnableHeaterLimit (
    unsigned int device,
    bool enable )
```

11.114.3.69 SetEnableThermocouple()

```
void SetEnableThermocouple (
    unsigned int device,
    bool enable )
```

11.114.3.70 SetHeaterLimit()

```
void SetHeaterLimit (
    unsigned int device,
    int heater_limit )
```

11.114.3.71 SetI()

```
void SetI (
    unsigned int device,
    int i_coeff )
```

Sets the I-coefficient of the specified device.

11.114.3.72 SetMaxHeaterPowerMultiwell()

```
void SetMaxHeaterPowerMultiwell (
    double MaxPowerWatt )
```

sets the max. heater power that the Multiwell temperature controller will apply; unit: W; useful range: 5.2W..7.6W

11.114.3.73 SetMaxP()

```
void SetMaxP (
    unsigned int device,
    int maxp )
```

Sets the maximum heater power of the specified device.

11.114.3.74 SetOnOff()

```
void SetOnOff (
    unsigned int channel,
    bool on )
```

Switches the specified channel on or off.

Parameters

channel	The channel number.

11.114.3.75 SetP() `void SetP (`

```
unsigned int device,
int p_coeff )
```

Sets the P-coefficient of the specified device.

11.114.3.76 SetSensorType() `void SetSensorType (`

```
unsigned int device,
TcxSensorTypeEnumNet type )
```

11.114.3.77 SetSetpoint() `void SetSetpoint (`

```
unsigned int channel,
int sp )
```

Sets the target temperate of specified channel in units of 0.1 °C.

11.114.3.78 SetThermocoupleNanovoltPerKelvin() `void SetThermocoupleNanovoltPerKelvin (`

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

Sets the proportional constant for the thermocouple.

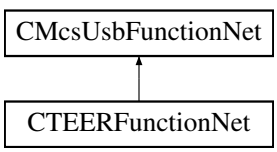
Parameters

channel	Thermocouple channel number.
value	Proportinal constant in Nanovolt per Kelvin.

11.115 CTEERFunctionNet Class Reference

CTEERFunctionNet is the class to control the TEER device

Inheritance diagram for CTEERFunctionNet:



Public Member Functions

- CTEERFunctionNet (CMcsUsbNet^ 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

11.115.1 Detailed Description

CTEERFunctionNet is the class to control the TEER device

11.115.2 Constructor & Destructor Documentation

11.115.2.1 CTEERFunctionNet() [1/2]

```
CTEERFunctionNet (
    CMcsUsbNet^ mcsusb,
    CMcsUsbFunctionPointerContainer^ pTEERFunctionPointerContainer )
```

Initializes a new instance of the CTEERFunctionNet class.

11.115.2.2 CTEERFunctionNet() [2/2] CTEERFunctionNet (CMcsUsbNet^ *mcsusb*)

11.115.2.3 ~CTEERFunctionNet() virtual ~CTEERFunctionNet ( ) [virtual]

11.115.2.4 "!CTEERFunctionNet() !CTEERFunctionNet ( )

11.115.3 Member Function Documentation

11.115.3.1 CancelInternalCalibration() void CancelInternalCalibration ( )

in case the internal calibration "hangs", this will cancel it

11.115.3.2 GetAdapterCode() uint32_t GetAdapterCode ( )

gets the adapter code

Returns

the adapter code

11.115.3.3 GetAdcOffsetU1() int GetAdcOffsetU1 ( )

returns ADC offset of U1 channel (result of internal calibration)

Returns

the ADC offset for U1

11.115.3.4 GetAdcOffsetU2() int GetAdcOffsetU2 ( )

returns ADC offset of U2 channel (result of internal calibration)

Returns

the ADC offset for U2

11.115.3.5 GetAmplitude_nA() `uint32_t GetAmplitude_nA ( )`

gets TEER stimulation amplitude in nA

Returns

current stimulation amplitude in nA

11.115.3.6 GetBytesPerSample() `uint32_t GetBytesPerSample ( )`

private function to be called internally only

Returns

not documented

11.115.3.7 GetClampMode() `TeerClampModeEnumNet GetClampMode ( )`

gets TEER clamp mode (voltage/current)

Returns

current TEER clamp mode

11.115.3.8 GetControllerParams()

```
void GetControllerParams (
    [System::Runtime::InteropServices::Out] uint32_t% P,
    [System::Runtime::InteropServices::Out] uint32_t% I,
    [System::Runtime::InteropServices::Out] uint32_t% D )
```

gets PID controller parameters for voltage clamp mode

Parameters

P	the P value
I	the I value
D	the D value

11.115.3.9 GetCurrentEnable() `bool GetCurrentEnable ( )`

when disabled, no current will flow through chamber

Returns

false when disabled, true when enabled

11.115.3.10 GetDacZero() `int GetDacZero ( )`

returns DAC-offset (result of internal calibration); use to check for plausibility only

Returns

the DAC offset

11.115.3.11 GetFrameErrorCounter() `int GetFrameErrorCounter ( )`

returns number of times (since bootup) sample memory got overwritten

Returns

the number of errors

11.115.3.12 GetLiquidResistance() `int32_t GetLiquidResistance ( )`

gets the resitance of the liquid in ohms

Returns

the resistance in ohms

11.115.3.13 GetMaxChunkSize_Byte() `uint32_t GetMaxChunkSize_Byte ( )`

private function to be called internally only

Returns

not documented

11.115.3.14 GetNumberOfAvailableSamples() `uint32_t GetNumberOfAvailableSamples ( )`

private function to be called internally only

Returns

not documented

11.115.3.15 GetPeriod_us() `uint32_t GetPeriod_us ( )`

gets the period of TEER stimulation in us

Returns

the period in us

11.115.3.16 GetRotaryPositionCode() `uint32_t GetRotaryPositionCode ( )`

gets the rotary position code

Returns

the rotary position code

11.115.3.17 GetSampleBufferChunk() `array<int32_t> ^ GetSampleBufferChunk ( int Buffer_Length )`

private function to query max. 100 bytes of sample buffer; called internally

Parameters

Buffer_Length	The maximal length of Buffer.

Returns

not documented

11.115.3.18 GetSampleRate() `int GetSampleRate ( )`

returns sample rate in Hz

Returns

the sample rate in Hz

11.115.3.19 GetSampleVoltageBuffer_uV() `array<int32_t> ^ GetSampleVoltageBuffer_uV ( int Buffer_Length )`

returns voltage sample buffer (max. 500 values); unit: uV

Parameters

Buffer_Length	The maximal length of Buffer.

Returns

the voltage sample buffer

11.115.3.20 GetScaleFactorU1() `int GetScaleFactorU1 ( )`

returns U1 scale factor times 10^6 (result of internal calibration)

Returns

the U1 scale factor

11.115.3.21 GetScaleFactorU2() `int GetScaleFactorU2 ( )`

returns U2 scale factor times 10^6 (result of internal calibration)

Returns

the U2 scale factor

11.115.3.22 GetUptimeSeconds() `int32_t GetUptimeSeconds ( )`

returns time in seconds since device was powered up

Returns

seconds since power-on

11.115.3.23 GetWaveform() `TeerWaveformEnumNet GetWaveform ( )`

gets TEER stimulation waveform (sine/rect)

Returns

waveform enum

11.115.3.24 IsInternalCalibrationFinished() `bool IsInternalCalibrationFinished ( )`

queries whether internal calibration has finished

Returns

true if calibration has finished

11.115.3.25 IsSamplingFinished() `uint32_t IsSamplingFinished ( )`

returns false iff stimulation/sampling is going on, otherwise true

Returns

true if sampling is finished

11.115.3.26 SetAmplitude_nA() `void SetAmplitude_nA ( uint32_t Amplitude_nA )`

sets TEER stimulation amplitude in nA

Parameters

Amplitude_nA	new stimulation amplitude in nA

11.115.3.27 SetBufferIndex() `void SetBufferIndex ( uint32_t NewBufferIndex )`

pre-selects sample buffer to be tranferred by GetSampleVoltageBuffer_uV()

Parameters

NewBufferIndex	0 - chamber voltage; 1 - compliance voltage

11.115.3.28 SetClampMode() `void SetClampMode (`

`TeerClampModeEnumNet ClampMode )`

sets TEER clamp mode (voltage/current)

Parameters

ClampMode	new TEER clamp mode

11.115.3.29 SetControllerParams() `void SetControllerParams (`

```
uint32_t P,
uint32_t I,
uint32_t D )
```

sets PID controller parameters for voltage clamp mode

Parameters

P	the P value
I	the I value
D	the D value

11.115.3.30 SetCurrentEnable() `void SetCurrentEnable (`

`bool NewCurrentEnable )`

when disabled, no current will flow through chamber

Parameters

NewCurrentEnable	disabled when false, enabled when true

11.115.3.31 SetExternalLED() `void SetExternalLED (`

`uint32_t NewState )`

sets the external LED

Parameters

NewState	state

11.115.3.32 SetLiquidResistance() `void SetLiquidResistance ( int32_t NewLiquidResistance_Ohm )`

sets the resistance of the liquid in ohms

Parameters

NewLiquidResistance_Ohm	the resistance in ohms

11.115.3.33 SetPeriod_us() `void SetPeriod_us ( uint32_t period_us )`

sets the period of TEER stimulation in us

Parameters

period_us	the period in us

11.115.3.34 SetWaveform() `void SetWaveform ( TeerWaveformEnumNet Waveform )`

sets TEER stimulation waveform (sine/rect)

Parameters

Waveform	waveform enum

11.115.3.35 StartInternalCalibration() `void StartInternalCalibration ( )`

starts determination of internal DAC-offset; result is used internally; NON-BLOCKING call

11.115.3.36 StartSampling() `void StartSampling ( uint32_t NumberOfCycles )`

starts TEER stimulation (duration: n cycles) and samples during last cycle

Parameters

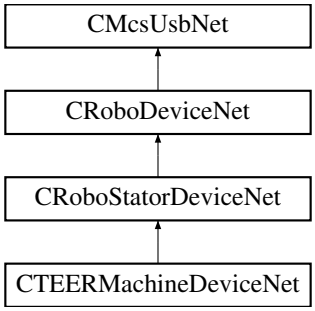
NumberOfCycles	number of cycles (sine or rect) to output (0 - loop forever)

11.115.3.37 StopSampling() `void StopSampling ( )`

stops TEER stimulation and sampling

11.116 CTEERMachineDeviceNet Class Reference

Inheritance diagram for CTEERMachineDeviceNet:



Public Member Functions

- CTEERMachineDeviceNet ()
- ~CTEERMachineDeviceNet ()

Properties

- CTEERFunctionNet^ TEERFunctionNet `[get]`

Additional Inherited Members

11.116.1 Constructor & Destructor Documentation

11.116.1.1 CTEERMachineDeviceNet() `CTEERMachineDeviceNet ( )`

11.116.1.2 ~CTEERMachineDeviceNet() `~CTEERMachineDeviceNet ( )`

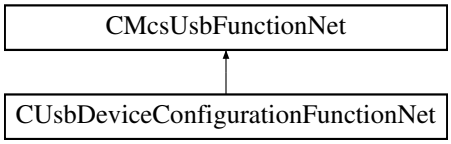
11.116.2 Property Documentation

11.116.2.1 TEERFunctionNet CTEERFunctionNet^ TEERFunctionNet [get]

11.117 CUsbDeviceConfigurationFunctionNet Class Reference

CUsbDeviceConfigurationFunctionNet is the class to configure the USB firmware

Inheritance diagram for CUsbDeviceConfigurationFunctionNet:



Public Member Functions

- CUsbDeviceConfigurationFunctionNet (CMcsUsbNet^ 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

11.117.1 Detailed Description

CUsbDeviceConfigurationFunctionNet is the class to configure the USB firmware

11.117.2 Constructor & Destructor Documentation

11.117.2.1 CUsbDeviceConfigurationFunctionNet() [1/2] CUsbDeviceConfigurationFunctionNet (CMcsUsbNet^ *mcsusb*, CMcsUsbFunctionPointerContainer^ *pUsbDeviceConfigurationFunctionPointerContainer*)

Initializes a new instance of the CUsbDeviceConfigurationFunctionNet class.

11.117.2.2 CUsbDeviceConfigurationFunctionNet() [2/2] CUsbDeviceConfigurationFunctionNet (CMcsUsbNet^ *mcsusb*)

11.117.2.3 ~CUsbDeviceConfigurationFunctionNet() virtual ~CUsbDeviceConfigurationFunctionNet ( ) [virtual]

11.117.2.4 "!CUsbDeviceConfigurationFunctionNet() !CUsbDeviceConfigurationFunctionNet ( )

11.117.3 Member Function Documentation

11.117.3.1 SetDeviceId() void SetDeviceId (ProductIdEnumNet *id*)

sets the USB device name for configurable devices

Parameters

id	

11.117.3.2 SetDeviceName() void SetDeviceName (String^ *name*)

sets the USB device name for configurable devices

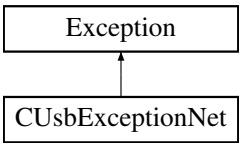
Parameters

name	

11.118 CUsbExceptionNet Class Reference

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

Inheritance diagram for CUsbExceptionNet:



Public Member Functions

- CUsbExceptionNet (uint32_t status)

  Constructor of a CUsbException.
- CUsbExceptionNet (uint32_t status, String^ message)

Properties

- uint32_t Status [get]

11.118.1 Detailed Description

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

11.118.2 Constructor & Destructor Documentation

11.118.2.1 CUsbExceptionNet() [1/2]

```
CUsbExceptionNet (
    uint32_t status )
```

Constructor of a CUsbException.

Parameters

status	the status number

11.118.2.2 CUsbExceptionNet() [2/2]

```
CUsbExceptionNet (
    uint32_t status,
    String^ message )
```

11.118.3 Property Documentation

11.118.3.1 Status

```
uint32_t Status [get]
```

11.119 CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet Class Reference

Public Member Functions

- CVoltageRangeInfoNet (int vr, String^ vrString)

Public Attributes

- int VoltageRangeInMicroVolt
- String ^ VoltageRangeDisplayStringMilliVolt

11.119.1 Constructor & Destructor Documentation

11.119.1.1 CVoltageRangeInfoNet() `CVoltageRangeInfoNet (`

```
int vr,
String^ vrString )
```

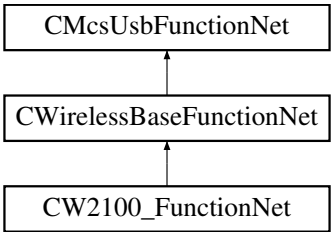
11.119.2 Member Data Documentation

11.119.2.1 VoltageRangeDisplayStringMilliVolt `String ^ VoltageRangeDisplayStringMilliVolt`

11.119.2.2 VoltageRangeInMicroVolt `int VoltageRangeInMicroVolt`

11.120 CW2100_FunctionNet Class Reference

Inheritance diagram for CW2100_FunctionNet:



Classes

- struct AudioChannelsNet

Public Member Functions

- CW2100_FunctionNet (CMcsUsbNet^ 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

11.120.1 Constructor & Destructor Documentation

11.120.1.1 CW2100_FunctionNet() [1/2]

```
CW2100_FunctionNet (
    CMcsUsbNet^ mcsusb,
    CMcsUsbFunctionPointerContainer^ w2100_FunctionPointerContainer )
```

11.120.1.2 CW2100_FunctionNet() [2/2]

```
CW2100_FunctionNet (
    CMcsUsbNet^ mcsusb )
```

11.120.2 Member Function Documentation

11.120.2.1 ClearStimulusParametersCache() [1/2]

```
static void ClearStimulusParametersCache ( )
[static]
```

11.120.2.2 ClearStimulusParametersCache() [2/2]

```
static void ClearStimulusParametersCache (
    unsigned short ID )  [static]
```

11.120.2.3 ClearUserDefinedNameCache() [1/2]

```
static void ClearUserDefinedNameCache ( )  [static]
```

11.120.2.4 ClearUserDefinedNameCache() [2/2]

```
static void ClearUserDefinedNameCache (
    unsigned short ID )  [static]
```

11.120.2.5 DeselectAllHeadstages()

```
void DeselectAllHeadstages ( )
```

11.120.2.6 DeselectHeadstage()

```
void DeselectHeadstage (
    int TimeSlotNr )
```

11.120.2.7 GetAccelGyroCurrentRate()

```
int GetAccelGyroCurrentRate (
    int TimeSlotNr )
```

11.120.2.8 GetAccelGyroDesiredRate()

```
int GetAccelGyroDesiredRate (
    int TimeSlotNr )
```

11.120.2.9 GetAccelGyroEnabled()

```
W2100_Accel_Gyro_Select_EnumNet GetAccelGyroEnabled (
    int TimeSlotNr )
```

11.120.2.10 GetAccelRange()

```
int GetAccelRange (
    int TimeSlotNr )
```

11.120.2.11 GetAnalogOutChannel()

```
unsigned int GetAnalogOutChannel (
    [System::Runtime::InteropServices::Out] int % automatic,
    unsigned short index )
```

11.120.2.12 GetAnalogOutFilter()

```
array<unsigned int> ^ GetAnalogOutFilter (
    [System::Runtime::InteropServices::Out] int % automatic )
```

11.120.2.13 GetAudioChannels()

```
array<AudioChannelsNet^> ^ GetAudioChannels ( )
```

11.120.2.14 GetAvailableHeadstages()

```
array<HeadStageIDType^> ^ GetAvailableHeadstages (
    unsigned int max_length )
```

11.120.2.15 GetBatteryState()

```
BatteryState ^ GetBatteryState (
    int TimeSlotNr )
```

11.120.2.16 GetDacRange()

```
AnalogOut_DAC_Range_EnumNet GetDacRange ( )
```

11.120.2.17 GetFilterProperties()

```
array<CFilterPropertyNet^> ^ GetFilterProperties (
    W2100DacqGroupChannelEnumNet GroupID )
```

11.120.2.18 GetFilterProperty()

```
CFilterPropertyNet ^ GetFilterProperty (
    W2100DacqGroupChannelEnumNet GroupID,
    unsigned int index )
```

11.120.2.19 GetFPGAFirmwareType()

```
unsigned int GetFPGAFirmwareType (
    int TimeSlotNr )
```

11.120.2.20 GetGyroRange()

```
int GetGyroRange (
    int TimeSlotNr )
```

11.120.2.21 GetHeadstageOnOff()

```
unsigned short GetHeadstageOnOff (
    int TimeSlotNr )
```

11.120.2.22 GetHeadstageSamplingActive()

```
bool GetHeadstageSamplingActive (
    int TimeSlotNr )
```

11.120.2.23 GetMultiHeadstageMode()

```
bool GetMultiHeadstageMode ( )
```

11.120.2.24 GetPicFirmwareType()

```
unsigned int GetPicFirmwareType (
    int TimeSlotNr )
```

11.120.2.25 GetSelectedChannels()

```
array<BYTE> ^ GetSelectedChannels (
    int TimeSlotNr )
```

11.120.2.26 GetSelectedHeadstageState()

```
HeadStageIDTypeState ^ GetSelectedHeadstageState (
    int TimeSlotNr )
```

11.120.2.27 GetStimulusParametersCache() [1/2]

```
W2100_StimulusParametersNet ^ GetStimulus↩
ParametersCache (
    unsigned int typeValue )
```

11.120.2.28 GetStimulusParametersCache() [2/2]

```
uint32_t GetStimulusParametersCache (
    unsigned int typeValue,
    [System::Runtime::InteropServices::Out] W2100_StimulusParametersNet^% Stimulus↩
Parameters )
```

11.120.2.29 GetStimulusParametersFromSelectedHS()

```
W2100_StimulusParametersNet ^ GetStimulus↩
ParametersFromSelectedHS (
    int TimeSlotNr )
```

11.120.2.30 GetStiumlusParameters()

```
W2100_StimulusParametersNet ^ GetStiumlusParameters (
    unsigned short ID )
```

11.120.2.31 GetUserDefinedName()

```
System::String ^ GetUserDefinedName (
    unsigned short ID )
```

11.120.2.32 GetUserDefinedNameCache() [1/2]

```
System::String ^ GetUserDefinedNameCache (
    unsigned short ID )
```

11.120.2.33 GetUserDefinedNameCache() [2/2] uint32_t GetUserDefinedNameCache (

```
unsigned short ID,
[System::Runtime::InteropServices::Out] System::String^% Name )
```

11.120.2.34 GetUserDefinedNameFromSelectedHS() System::String ^ GetUserDefinedNameFrom↩ SelectedHS (

```
int TimeSlotNr )
```

11.120.2.35 SelectHeadstage() void SelectHeadstage (

```
unsigned int IDorEntry,
int TimeSlotNr )
```

11.120.2.36 SetAccelGyroDesiredRate() void SetAccelGyroDesiredRate (

```
int rate,
int TimeSlotNr )
```

11.120.2.37 SetAccelGyroEnabled() void SetAccelGyroEnabled (

```
W2100_Accel_Gyro_Select_EnumNet enable,
int TimeSlotNr )
```

11.120.2.38 SetAccelRange() void SetAccelRange (

```
int range,
int TimeSlotNr )
```

11.120.2.39 SetAnalogOutChannel() void SetAnalogOutChannel (

```
int automatic,
unsigned short index,
unsigned int Channel )
```

11.120.2.40 SetAnalogOutFilter() void SetAnalogOutFilter (

```
int automatic,
array< unsigned int >^ Coeffs )
```

11.120.2.41 SetAudioChannels()

```
void SetAudioChannels (
    array< AudioChannelsNet^>^ channels )
```

11.120.2.42 SetDacRange()

```
void SetDacRange (
    AnalogOut_DAC_Range_EnumNet range )
```

11.120.2.43 SetGyroRange()

```
void SetGyroRange (
    int range,
    int TimeSlotNr )
```

11.120.2.44 SetHeadstageOnOff()

```
void SetHeadstageOnOff (
    unsigned short On,
    int TimeSlotNr )
```

11.120.2.45 SetHeadstageSamplingActive()

```
void SetHeadstageSamplingActive (
    bool Active,
    int TimeSlotNr )
```

11.120.2.46 SetHeadstageToSleep()

```
void SetHeadstageToSleep (
    unsigned int Sleep16ms,
    int TimeSlotNr )
```

11.120.2.47 SetMultiHeadstageMode()

```
void SetMultiHeadstageMode (
    bool Mode )
```

11.120.2.48 SetSelectedChannels()

```
void SetSelectedChannels (
    array< BYTE >^ channels,
    int TimeSlotNr )
```

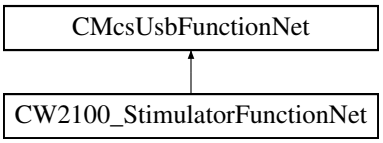
11.120.3 Property Documentation

11.120.3.1 PulseGenerator CPulseGeneratorFunctionNet^ PulseGenerator [get]

11.120.3.2 Stimulator CW2100_StimulatorFunctionNet^ Stimulator [get]

11.121 CW2100_StimulatorFunctionNet Class Reference

Inheritance diagram for CW2100_StimulatorFunctionNet:



Public Member Functions

- CW2100_StimulatorFunctionNet (CMcsUsbNet^ mcsusb)
- void SendStart (uint32_t triggermap)

  Start (Trigger) the STG. The startup delay is in the range of a few ms.
- void SendStop (uint32_t triggermap)

  Stop some or all triggers of the STG.
- CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData ^ PrepareData (int channel, array< int32_t >^ amplitude, array< uint64_t >^ duration, STG_DestinationEnumNet destType, uint32_t repeat)
- CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData ^ PrepareDataSync (int channel, array< int32_t >^ amplitude, array< uint32_t >^ Sync, array< uint64_t >^ duration, STG_DestinationEnumNet destType, uint32_t repeat)
- void SendPreparedData (int channel, CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData^ device_data_and_unrolled, STG_DestinationEnumNet destType)
- void ClearChannelData (int channel)

  Delete a Stimulus Pattern from STG memory
- int GetDACResolution ()

  Gets number of bits of the DAC resolution.
- int GetTimeResolutionInNanoSeconds ()

  Gets number of bits of the DAC resolution.
- int GetVoltageRangeInMicroVolt (uint32_t channel)

  Gets the Voltage Range of the specified channel in Microvolts.
- int GetVoltageResolutionInMicroVolt (uint32_t channel)

  Gets the Voltage Resolution of the specified channel in Microvolts.
- int GetCurrentRangeInNanoAmp (uint32_t channel)

  Gets the Current Range of the specified channel in Nanoamps.
- int GetCurrentResolutionInNanoAmp (uint32_t channel)

  Gets the Current Resolution of the specified channel in Nanoamps.
- uint32_t GetNumberOfAnalogChannels ()
- uint32_t GetNumberOfSyncoutChannels ()
- uint32_t GetNumberOfTriggerInputs ()
- void SelectTimeSlot (int TimeSlotNr)
- int GetTimeSlot ()
- uint32_t GetStimulationPatternMemory ()

- uint32_t GetBoostPreTime ()
- uint32_t GetBoostAlwaysOnMode ()
- void SetDigitalStimulatorTrigger (int TimeSlotNr, DigitalStimulatorTriggerEventEnumNet trigger_event, int trigger_number, W2100DigitalSourceEnumNet digstream_source, int bitnumber_offset)
- void GetDigitalStimulatorTrigger (int TimeSlotNr, DigitalStimulatorTriggerEventEnumNet trigger_event, int trigger_number, [System::Runtime::InteropServices::Out]W2100DigitalSourceEnumNet% digstream_source, [System::Runtime::InteropServices::Out]int% bitnumber_offset)
- void SetDigitalStimulatorTriggerSlope (int TimeSlotNr, DigitalStimulatorTriggerEventEnumNet trigger_event, int trigger_number, DigitalStimulatorTriggerSlopeEnumNet slope)
- DigitalStimulatorTriggerSlopeEnumNet GetDigitalStimulatorTriggerSlope (int TimeSlotNr, 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

11.121.1 Constructor & Destructor Documentation

11.121.1.1 CW2100_StimulatorFunctionNet() `CW2100_StimulatorFunctionNet ( CMcsUsbNet^ mcsusb )`

11.121.2 Member Function Documentation

11.121.2.1 ClearChannelData() `void ClearChannelData ( int channel )`

Delete a Stimulus Pattern from STG memory

Parameters

channel	specifies the channel to clear.

11.121.2.2 GetBoostAlwaysOnMode() uint32_t GetBoostAlwaysOnMode ( )

11.121.2.3 GetBoostPreTime() uint32_t GetBoostPreTime ( )

11.121.2.4 GetCurrentRangeInNanoAmp() int GetCurrentRangeInNanoAmp (uint32_t *channel*)

Gets the Current Range of the specified channel in Nanoamps.

Parameters

channel	Channel which is queried.

Returns

The Current Range of the specified channel in Nanoamps.

11.121.2.5 GetCurrentResolutionInNanoAmp() int GetCurrentResolutionInNanoAmp (uint32_t *channel*)

Gets the Current Resolution of the specified channel in Nanoamps.

Parameters

channel	Channel which is queried.

Returns

The Current Resolution of the specified channel in Nanoamps.

11.121.2.6 GetDACResolution() int GetDACResolution ( )

Gets number of bits of the DAC resolution.

Returns

The DAC resolution in bits.

11.121.2.7 GetDigitalStimulatorTrigger()

```
void GetDigitalStimulatorTrigger (
            int TimeSlotNr,
            DigitalStimulatorTriggerEventEnumNet trigger_event,
            int trigger_number,
            [System::Runtime::InteropServices::Out] W2100DigitalSourceEnumNet% digstream_source,
            [System::Runtime::InteropServices::Out] int% bitnumber_offset )
```

11.121.2.8 GetDigitalStimulatorTriggerSlope()

```
DigitalStimulatorTriggerSlopeEnumNet GetDigitalStimulatorTriggerSlope (
            int TimeSlotNr,
            DigitalStimulatorTriggerEventEnumNet trigger_event,
            int trigger_number )
```

11.121.2.9 GetNumberOfAnalogChannels()

```
uint32_t GetNumberOfAnalogChannels ( )
```

11.121.2.10 GetNumberOfSyncoutChannels()

```
uint32_t GetNumberOfSyncoutChannels ( )
```

11.121.2.11 GetNumberOfTriggerInputs()

```
uint32_t GetNumberOfTriggerInputs ( )
```

11.121.2.12 GetStimulationPatternMemory()

```
uint32_t GetStimulationPatternMemory ( )
```

11.121.2.13 GetTimeResolutionInNanoSeconds()

```
int GetTimeResolutionInNanoSeconds ( )
```

Gets number of bits of the DAC resolution.

Returns

The time resolution in ns.

11.121.2.14 GetTimeSlot()

```
int GetTimeSlot ( )
```

11.121.2.15 GetVoltageRangeInMicroVolt()

```
int GetVoltageRangeInMicroVolt (
            uint32_t channel )
```

Gets the Voltage Range of the specified channel in Microvolts.

Parameters

channel	Channel which is queried.

Returns

The Voltage Range of the specified channel in Microvolts.

11.121.2.16 GetVoltageResolutionInMicroVolt()

```
int GetVoltageResolutionInMicroVolt (
        uint32_t channel )
```

Gets the Voltage Resolution of the specified channel in Microvolts.

Parameters

channel	Channel which is queried.

Returns

The Voltage Resolution of the specified channel in Microvolts.

11.121.2.17 PrepareData()

```
CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData ^ PrepareData (
        int channel,
        array< int32_t >^ amplitude,
        array< uint64_t >^ duration,
        STG_DestinationEnumNet destType,
        uint32_t repeat )
```

11.121.2.18 PrepareDataSync()

```
CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData ^ PrepareDataSync (
        int channel,
        array< int32_t >^ amplitude,
        array< uint32_t >^ Sync,
        array< uint64_t >^ duration,
        STG_DestinationEnumNet destType,
        uint32_t repeat )
```

11.121.2.19 SelectTimeSlot()

```
void SelectTimeSlot (
        int TimeSlotNr )
```

11.121.2.20 SendPreparedData() void SendPreparedData (

```
int channel,
CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData^ device_data_and_unrolled,
STG_DestinationEnumNet destType )
```

11.121.2.21 SendStart() void SendStart (

```
uint32_t triggermap )
```

Start (Trigger) the STG. The startup delay is in the range of a few ms.

Parameters

triggermap	A bitmap of triggers which will be started.

11.121.2.22 SendStop() void SendStop (

```
uint32_t triggermap )
```

Stop some or all triggers of the STG.

Parameters

triggermap	A bitmap of triggers which will be stopped.

11.121.2.23 SetDigitalStimulatorTrigger() void SetDigitalStimulatorTrigger (

```
int TimeSlotNr,
DigitalStimulatorTriggerEventEnumNet trigger_event,
int trigger_number,
W2100DigitalSourceEnumNet digstream_source,
int bitnumber_offset )
```

11.121.2.24 SetDigitalStimulatorTriggerSlope() void SetDigitalStimulatorTriggerSlope (

```
int TimeSlotNr,
DigitalStimulatorTriggerEventEnumNet trigger_event,
int trigger_number,
DigitalStimulatorTriggerSlopeEnumNet slope )
```

11.121.2.25 StartPoll() void StartPoll ( )

11.121.2.26 StopPoll() void StopPoll ( )

11.121.3 Member Data Documentation

11.121.3.1 BOOST_BIT const uint32_t BOOST_BIT = (1 << 0) [static]

11.121.3.2 GND_SWITCH_BIT const uint32_t GND_SWITCH_BIT = (1 << 1) [static]

11.121.3.3 SYNC_BIT0 const uint32_t SYNC_BIT0 = (1 << 2) [static]

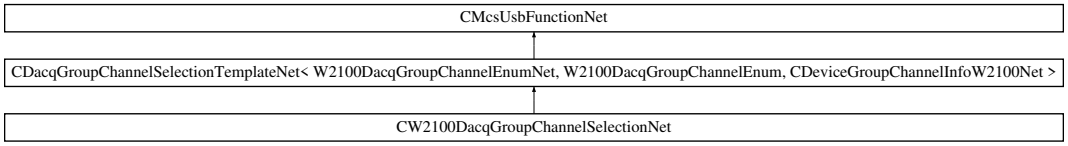
11.121.3.4 SYNC_BIT1 const uint32_t SYNC_BIT1 = (1 << 3) [static]

11.121.4 Event Documentation

11.121.4.1 PollStatusEvent OnStgPollStatus^ PollStatusEvent

11.122 CW2100DacqGroupChannelSelectionNet Class Reference

Inheritance diagram for CW2100DacqGroupChannelSelectionNet:



Public Member Functions

- CW2100DacqGroupChannelSelectionNet (CMcsUsbNet^ mcsusb)

Additional Inherited Members

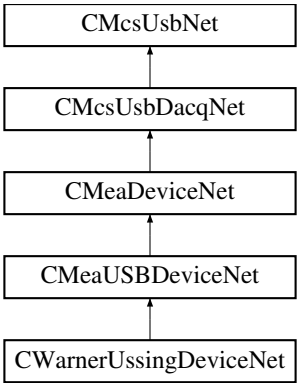
11.122.1 Constructor & Destructor Documentation

11.122.1.1 CW2100DacqGroupChannelSelectionNet() CW2100DacqGroupChannelSelectionNet (CMcsUsbNet^ *mcsusb*)

11.123 CWarnerUssingDeviceNet Class Reference

CWarnerUssingDeviceNet is the class to control the Ussing device

Inheritance diagram for CWarnerUssingDeviceNet:



Public Member Functions

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

Properties

- CWarnerUssingFunctionNet^ WarnerUssingFunction [get]

Additional Inherited Members

11.123.1 Detailed Description

CWarnerUssingDeviceNet is the class to control the Ussing device

11.123.2 Constructor & Destructor Documentation

11.123.2.1 CWarnerUssingDeviceNet() CWarnerUssingDeviceNet ( )

Initializes a new instance of the CWarnerUssingDeviceNet class.

11.123.2.2 ~CWarnerUssingDeviceNet() `virtual ~CWarnerUssingDeviceNet ( ) [virtual]`

11.123.2.3 "!CWarnerUssingDeviceNet() `!CWarnerUssingDeviceNet ( )`

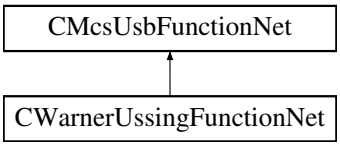
11.123.3 Property Documentation

11.123.3.1 WarnerUssingFunction `CWarnerUssingFunctionNet^ WarnerUssingFunction [get]`

11.124 CWarnerUssingFunctionNet Class Reference

CWarnerUssingFunctionNet is the class to control the Ussing device

Inheritance diagram for CWarnerUssingFunctionNet:



Public Member Functions

- CWarnerUssingFunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ 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

11.124.1 Detailed Description

CWarnerUssingFunctionNet is the class to control the Ussing device

11.124.2 Constructor & Destructor Documentation

11.124.2.1 CWarnerUssingFunctionNet() [1/2]

```
CWarnerUssingFunctionNet (
    CMcsUsbNet^ mcsusb,
    CMcsUsbFunctionPointerContainer^ pWarnerUssingFunctionPointerContainer )
```

Initializes a new instance of the CWarnerUssingFunctionNet class.

11.124.2.2 CWarnerUssingFunctionNet() [2/2]

```
CWarnerUssingFunctionNet (
    CMcsUsbNet^ mcsusb )
```

11.124.2.3 ~CWarnerUssingFunctionNet()

```
virtual ~CWarnerUssingFunctionNet ( ) [virtual]
```

11.124.2.4 "!CWarnerUssingFunctionNet()

```
!CWarnerUssingFunctionNet ( )
```

11.124.3 Member Function Documentation

11.124.3.1 CompensateElectrodeOffset()

```
bool CompensateElectrodeOffset (
    int32_t ChamberId )
```

blocking call to compensate electrode offset of one chamber; returns true when successful

Parameters

ChamberId	index of hardware chamber slot (zero-based)

Returns

true if compensation succeeded

11.124.3.2 GetAvailableChambers() `array<int> ^ GetAvailableChambers ( )`

returns array with (zero-based) ChamberIds of all available chambers

11.124.3.3 GetChannelsCountOfChamber() `int32_t GetChannelsCountOfChamber ( int32_t ChamberId )`

gets number of channels in datastream from chamber amp with given index

Parameters

ChamberId	index of hardware chamber slot (zero-based)

Returns

return value of zero means that amp is not placed

11.124.3.4 GetClampMode() `UssingClampModeEnumNet GetClampMode ( int32_t ChamberId )`

gets clamp mode (voltage, current or open clamp; do not use when device is in internal calibration mode)

Parameters

ChamberId	index of hardware chamber slot (zero-based)

Returns

the current clamp mode

11.124.3.5 GetComplianceVoltageThreshold() int32_t GetComplianceVoltageThreshold (int32_t *ChamberId*)

returns compliance voltage threshold in uV; when Uc is above, current source is overloaded

Parameters

ChamberId	index of hardware chamber slot (zero-based)

Returns

the compliance voltage threshold in uV

11.124.3.6 GetDacPampsPerDigitHighCurrentRange() uint32_t GetDacPampsPerDigitHighCurrentRange (int32_t *ChamberId*)

get the resolution of the high current mode

Parameters

ChamberId	index of hardware chamber slot (zero-based)

Returns

unit: pA/digit in high current mode

11.124.3.7 GetDacPampsPerDigitLowCurrentRange() uint32_t GetDacPampsPerDigitLowCurrentRange (int32_t *ChamberId*)

get the resolution of the low current mode

Parameters

ChamberId	index of hardware chamber slot (zero-based)

Returns

pA/digit in low current mode

11.124.3.8 GetDacZero() `int32_t GetDacZero ( int32_t ChamberId )`

- diagnostic function only - ; gets real zero value of DAC in digits (0 -> neg. current; 32767 -> near zero; 65535 -> pos. current)

Parameters

ChamberId	index of hardware chamber slot (zero-based)

Returns

the zero value of the DAC

11.124.3.9 GetHighCurrentRange() `uint32_t GetHighCurrentRange ( int32_t ChamberId )`

query the range of the high current mode

Parameters

ChamberId	index of hardware chamber slot (zero-based)

Returns

low current range in nA

11.124.3.10 GetIdleModeOffset() `int32_t GetIdleModeOffset ( int32_t ChamberId, UssingClampModeEnumNet ClampMode )`

gets the offset (voltage or current) that will be applied when clamping is DISABLED

Parameters

ChamberId	index of hardware chamber slot (zero-based)
ClampMode	voltage or current clamp stimulation

Returns

unit: nA or uV

11.124.3.11 GetLiquidResistance() `int32_t GetLiquidResistance ( int32_t *ChamberId* )`

gets the resistance of the liquid

Parameters

ChamberId	index of hardware chamber slot (zero-based)

Returns

the liquid restistance in ohm

11.124.3.12 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

11.124.3.13 GetNumberOfAvailableChambers() `int32_t GetNumberOfAvailableChambers ( )`

gets number of actually connected chamber amps

Returns

the number of actually connected chambers

11.124.3.14 GetNumberOfHardwareSlotsForChambers() `int32_t GetNumberOfHardwareSlotsForChambers ( )`

gets number of physical hardware slots for chambers amps

Returns

the number of hardware chamber slots on the backplane

11.124.3.15 GetU1Offset() int32_t GetU1Offset (int32_t *ChamberId*)

- diagnostic function only -

Parameters

ChamberId	index of hardware chamber slot (zero-based)

Returns

U1 offset

11.124.3.16 GetU1Reference() `int32_t GetU1Reference ( int32_t ChamberId )`

- diagnostic function only -

Parameters

ChamberId	index of hardware chamber slot (zero-based)

Returns

U1 reference

11.124.3.17 GetU2Offset() `int32_t GetU2Offset ( int32_t ChamberId )`

- diagnostic function only -

Parameters

ChamberId	index of hardware chamber slot (zero-based)

Returns

U2 offset

11.124.3.18 GetU2Reference() `int32_t GetU2Reference ( int32_t ChamberId )`

- diagnostic function only -

Parameters

ChamberId	index of hardware chamber slot (zero-based)

Returns

U2 reference

11.124.3.19 GetUnitDescription()

```
String ^ GetUnitDescription (
    int32_t ChamberId,
    int32_t ChannelId )
```

gets the description for the unit

Parameters

ChamberId	index of hardware chamber slot (zero-based)
ChannelId	index of channel (zero-based)

Returns

the description of the unix

11.124.3.20 GetUnitExponent()

```
int32_t GetUnitExponent (
    int32_t ChamberId,
    int32_t ChannelId )
```

gets the unit exponent for specified chamber and channel

Parameters

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

11.124.3.21 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

11.124.3.22 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

11.124.3.23 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

11.124.3.24 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

11.124.3.25 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

11.124.3.26 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

11.124.3.27 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

11.124.3.28 IsHighCurrentMode() `bool IsHighCurrentMode (`

`int32_t` *ChamberId*)

Parameters

ChamberId	index of hardware chamber slot (zero-based)

Returns

true if in hight current mode

11.124.3.29 IsInternalCalibrationFinished() `bool IsInternalCalibrationFinished (`

`int32_t` *ChamberId*)

when internal calibration is finished, values for U1,2_offset and U1,2_reference and DAC_offset are available

Parameters

ChamberId	index of hardware chamber slot (zero-based)

Returns

true if finished

11.124.3.30 IsPulseEnabled() `bool IsPulseEnabled (`

`int32_t` *ChamberId*,
`UssingClampModeEnumNet` *ClampMode*)

returns true when pulse of given chamber and current mode (voltage/current clamp) of this chamber is ENABLED

Parameters

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

11.124.3.31 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

11.124.3.32 SetEnablePulse()

```
void SetEnablePulse (
    int32_t ChamberId,
    UssingClampModeEnumNet ClampMode,
    bool Enable )
```

enable pulse of given chamber and mode (voltage/current clamp) of this chamber

Parameters

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

11.124.3.33 SetHighCurrentMode()

```
void SetHighCurrentMode (
    int32_t ChamberId )
```

switch to high-current mode

Parameters

ChamberId	index of hardware chamber slot (zero-based)

11.124.3.34 SetIdleModeOffset() `void SetIdleModeOffset (`

```
int32_t ChamberId,
UssingClampModeEnumNet ClampMode,
int32_t NewIdleOffset )
```

sets the offset (voltage or current) that will be applied when clamping is DISABLED

Parameters

ChamberId	index of hardware chamber slot (zero-based)
ClampMode	voltage or current clamp stimulation
NewIdleOffset	unit: nA or uV

11.124.3.35 SetLiquidResistance() `void SetLiquidResistance (`

```
int32_t ChamberId,
int32_t NewLiquidResistance_Ohm )
```

sets the resistance of the liquid

Parameters

ChamberId	index of hardware chamber slot (zero-based)
NewLiquidResistance_Ohm	the liquid resistiance in ohm

11.124.3.36 SetLowCurrentMode() `void SetLowCurrentMode (`

```
int32_t ChamberId )
```

switch to low-current mode

Parameters

ChamberId	index of hardware chamber slot (zero-based)

11.124.3.37 SetPulse() `void SetPulse (`

```
int32_t ChamberId,
```

```
UssingClampModeEnumNet StgMode,
int32_t NumberOfRepetitions,
array< int >^ Amplitudes,
array< int >^ Durations,
array< int >^ PulseMarker )
```

defines stimulation pulse pattern for voltage or current stimulation; CAUTION: zero-length amplitude will be briefly applied -> choose matching to neighbour to avoid spikes

Parameters

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")

11.124.3.38 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

11.124.3.39 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

11.124.3.40 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

11.124.3.41 WaitForAllChambers() `bool WaitForAllChambers ( )`

blocking call that waits for ALL chambers' boot-up calibration to complete

Returns

returns false when at least one chamber's calibration fails (e.g. timeout...)

11.124.3.42 WaitForChamber() `bool WaitForChamber (`

```
int32_t ChamberId )
```

blocking call that waits for chamber boot-up calibration to complete

Parameters

ChamberId	index of hardware chamber slot (zero-based)

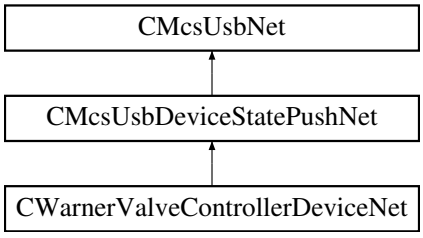
Returns

returns false when calibration fails (e.g. timeout...)

11.125 CWarnerValveControllerDeviceNet Class Reference

CWarnerValveControllerDeviceNet is the class to access the Warner Valve Controller

Inheritance diagram for CWarnerValveControllerDeviceNet:



Public Member Functions

- delegate void OnGetValveActive (uint16_t valve, int valveActive)
- delegate void OnGetValveManualState (uint16_t valve, int32_t valveManualState)
- delegate void OnGetValveManualGroup (uint16_t valve, int32_t valveManualGroup)
- delegate void OnGetValveMode (uint16_t valve, WvcValveModeEnumNet ValveMode)
- delegate void OnGetAnalogThresholdLow (uint16_t valve, int32_t threshold)
- delegate void OnGetAnalogThresholdHigh (uint16_t valve, int32_t threshold)
- delegate void OnGetDigitalPortDirection (uint16_t port, PortDirectionEnumNet direction)
- delegate void OnIsValveDigitalInInverted (uint16_t valve, bool isInverted)
- delegate void OnGetValveDigitalInPort (uint16_t valve, uint32_t digitalInPort)
- delegate void OnIsDigitalOutPortInverted (uint16_t digitalOutPort, bool isInverted)
- delegate void OnGetDigitalOutPortValve (uint16_t digitalOutPort, uint32_t valve)
- delegate void OnIsValveOpen (uint16_t valve, bool valveOpen)
- delegate void OnIsValveOpenInDigitalMode (uint16_t valve, bool valveOpen)
- delegate void OnIsValveOpenInAnalogMode (uint16_t valve, bool valveOpen)
- delegate void OnGetAnalogVoltage (int32_t voltage)
- delegate void OnTableEntryChanged (uint16_t tableNumber)
- delegate void OnGetTableNamebyIndex (uint16_t tableNumber, String^ tableName)
- delegate void OnGetActiveRunningTableNumber (uint32_t tableNumber)
- delegate void OnGetCurrentNumberOfValves (int32_t numberOfValves)
- delegate void OnGetValveBoardRevision (uint32_t revision)
- delegate void OnGetValveLedOn (bool ledon)
- delegate void OnGetDisplayMode (WvcDisplayModeEnumNet DisplayMode)
- CWarnerValveControllerDeviceNet ()

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

11.125.1 Detailed Description

CWarnerValveControllerDeviceNet is the class to access the Warner Valve Controller

11.125.2 Constructor & Destructor Documentation

11.125.2.1 CWarnerValveControllerDeviceNet() `CWarnerValveControllerDeviceNet ( )`

Initializes a new instance of the CWarnerValveControllerDeviceNet class.

11.125.2.2 ~CWarnerValveControllerDeviceNet() `virtual ~CWarnerValveControllerDeviceNet ( ) [virtual]`

11.125.2.3 "!CWarnerValveControllerDeviceNet() `!CWarnerValveControllerDeviceNet ( )`

11.125.3 Member Function Documentation

11.125.3.1 ClearTableName() `void ClearTableName ( )`

Clear the name of current protocol table

11.125.3.2 ClearValveTable() `void ClearValveTable ( uint16_t valve )`

Clear the valve protocol table

Parameters

valve	The valve number

11.125.3.3 GetActiveRunningTableNumber() `uint32_t GetActiveRunningTableNumber ( )`

Gets the number of the table that is active for running

Returns

The table number

11.125.3.4 GetAnalogThresholdHigh() int32_t GetAnalogThresholdHigh (uint16_t *valve*)

Gets the upper threshold for the analog in port per valve

Parameters

valve	The valve number

Returns

The threshold in mV

11.125.3.5 GetAnalogThresholdLow() int32_t GetAnalogThresholdLow (uint16_t *valve*)

Gets the lower threshold for the analog in port per valve

Parameters

valve	The valve number

Returns

The threshold in mV

11.125.3.6 GetAnalogVoltage() int32_t GetAnalogVoltage ( )

Reads the voltage on the analog in port

Returns

The voltage in mV

11.125.3.7 GetCurrentEditTableNumber() uint32_t GetCurrentEditTableNumber ( )

Gets the number of the table that is current for editing

Returns

The table number

11.125.3.8 GetCurrentNumberOfValves() `int32_t GetCurrentNumberOfValves ( )`

Get the current number of valves connected to the system

Returns

The number of valves

11.125.3.9 GetDigitalOutPortValve() `uint32_t GetDigitalOutPortValve ( uint16_t digitalOutPort )`

Gets the number of the valve which is mapped to a digital out port

Parameters

digitalOutPort	The digital out port

Returns

The valve number

11.125.3.10 GetDigitalPortDirection() `PortDirectionEnumNet GetDigitalPortDirection ( uint16_t port )`

Gets the direction of a digital port

Parameters

port	The port number

Returns

the direction

11.125.3.11 GetDisplayMode() `WvcDisplayModeEnumNet GetDisplayMode ( )`

Reads the display mode

Returns

The display mode

11.125.3.12 GetTableName() `String ^ GetTableName ( )`

Get the name of the current protocol table

Returns

The name of the table

11.125.3.13 GetTableNamebyIndex() `String ^ GetTableNamebyIndex ( uint16_t tableNumber )`

Get the name of a protocol table

Parameters

tableNumber	The table number

Returns

The name of the table

11.125.3.14 GetTotalNumberOfDigitalPorts() `int32_t GetTotalNumberOfDigitalPorts ( )`

Get the total number of digital ports in the system

Returns

The number of digital ports

11.125.3.15 GetTotalNumberOfTables() `int32_t GetTotalNumberOfTables ( )`

Get the total number of tables in the system

Returns

The number of tables

11.125.3.16 GetTotalNumberOfValves() `int32_t GetTotalNumberOfValves ( )`

Get the total number of valves in the system

Returns

The number of valves

11.125.3.17 GetTotalTableSize() `int32_t GetTotalTableSize ( )`

Get the total table size in the system

Returns

The table size

11.125.3.18 GetValveActive() `int GetValveActive ( uint16_t valve )`

Gets the valve active/inactive state

Parameters

valve	The valve number

Returns

The valve state

11.125.3.19 GetValveBoardRevision() `uint32_t GetValveBoardRevision ( )`

Gets the revision code of the valve board

Returns

The revision code

11.125.3.20 GetValveBoardRevisionString() `String ^ GetValveBoardRevisionString ( )`

Gets the revision name of the valve board

Returns

The revision name

11.125.3.21 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)

11.125.3.22 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

11.125.3.23 GetValveLedOn() `bool GetValveLedOn ( )`

Gets the LED state of the valve board

Returns

The LED state

11.125.3.24 GetValveManualGroup() `int32_t GetValveManualGroup (`

```
uint16_t valve )
```

Gets the valve manual group

Parameters

valve	The valve number

Returns

The manual valve group

11.125.3.25 GetValveManualState() `int32_t GetValveManualState ( uint16_t valve )`

Gets the valve manual on/off state

Parameters

valve	The valve number

Returns

The manual valve state

11.125.3.26 GetValveMode() `WvcValveModeEnumNet GetValveMode ( uint16_t valve )`

Reads the valve mode

Parameters

valve	The valve number

Returns

The valve mode

11.125.3.27 GetValvesActiveMap() `uint32_t GetValvesActiveMap ( )`

Gets the valves active/inactive states

Returns

The valves states

11.125.3.28 GetValvesManualStateMap() `uint32_t GetValvesManualStateMap ( )`

Gets the valves manual on/off states

Returns

The manual valves states

11.125.3.29 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

11.125.3.30 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

11.125.3.31 IsValveDigitalInInverted() `bool IsValveDigitalInInverted (`

```
uint16_t valve )
```

Is digital in inverted

Parameters

valve	The valve number

Returns

is inverted

11.125.3.32 IsValveOpen() `bool IsValveOpen ( uint16_t valve )`

Is valve open

Parameters

valve	The valve number

Returns

is open

11.125.3.33 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

11.125.3.34 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

11.125.3.35 LoadValveTable() void LoadValveTable ( )

Load the current table from permanent memory

11.125.3.36 OnGetActiveRunningTableNumber() delegate void OnGetActiveRunningTableNumber (uint32_t *tableNumber*)

11.125.3.37 OnGetAnalogThresholdHigh() delegate void OnGetAnalogThresholdHigh (uint16_t *valve*, int32_t *threshold*)

11.125.3.38 OnGetAnalogThresholdLow() delegate void OnGetAnalogThresholdLow (uint16_t *valve*, int32_t *threshold*)

11.125.3.39 OnGetAnalogVoltage() delegate void OnGetAnalogVoltage (int32_t *voltage*)

11.125.3.40 OnGetCurrentNumberOfValves() delegate void OnGetCurrentNumberOfValves (int32_t *numberOfValves*)

11.125.3.41 OnGetDigitalOutPortValve() delegate void OnGetDigitalOutPortValve (uint16_t *digitalOutPort*, uint32_t *valve*)

11.125.3.42 OnGetDigitalPortDirection() delegate void OnGetDigitalPortDirection (uint16_t *port*, PortDirectionEnumNet *direction*)

11.125.3.43 OnGetDisplayMode() delegate void OnGetDisplayMode (WvcDisplayModeEnumNet *DisplayMode*)

11.125.3.44 OnGetTableNamebyIndex() delegate void OnGetTableNamebyIndex (

```
uint16_t tableNumber,
String^ tableName )
```

11.125.3.45 OnGetValveActive() delegate void OnGetValveActive (

```
uint16_t valve,
int valveActive )
```

11.125.3.46 OnGetValveBoardRevision() delegate void OnGetValveBoardRevision (

```
uint32_t revision )
```

11.125.3.47 OnGetValveDigitalInPort() delegate void OnGetValveDigitalInPort (

```
uint16_t valve,
uint32_t digitalInPort )
```

11.125.3.48 OnGetValveLedOn() delegate void OnGetValveLedOn (

```
bool ledon )
```

11.125.3.49 OnGetValveManualGroup() delegate void OnGetValveManualGroup (

```
uint16_t valve,
int32_t valveManualGroup )
```

11.125.3.50 OnGetValveManualState() delegate void OnGetValveManualState (

```
uint16_t valve,
int32_t valveManualState )
```

11.125.3.51 OnGetValveMode() delegate void OnGetValveMode (

```
uint16_t valve,
WvcValveModeEnumNet ValveMode )
```

11.125.3.52 OnIsDigitalOutPortInverted() delegate void OnIsDigitalOutPortInverted (

```
uint16_t digitalOutPort,
bool isInverted )
```

11.125.3.53 OnIsValveDigitalInInverted() delegate void OnIsValveDigitalInInverted (

```
uint16_t valve,
bool isInverted )
```

11.125.3.54 OnIsValveOpen() delegate void OnIsValveOpen (

```
uint16_t valve,
bool valveOpen )
```

11.125.3.55 OnIsValveOpenInAnalogMode() delegate void OnIsValveOpenInAnalogMode (

```
uint16_t valve,
bool valveOpen )
```

11.125.3.56 OnIsValveOpenInDigitalMode() delegate void OnIsValveOpenInDigitalMode (

```
uint16_t valve,
bool valveOpen )
```

11.125.3.57 OnTableEntryChanged() delegate void OnTableEntryChanged (

```
uint16_t tableNumber )
```

11.125.3.58 SetActiveRunningTableNumber() void SetActiveRunningTableNumber (

```
uint32_t tableNumber )
```

Sets the number of the tanle that is active for running

Parameters

tableNumber	The table number

11.125.3.59 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

11.125.3.60 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

11.125.3.61 SetCurrentEditTableNumber() `void SetCurrentEditTableNumber ( uint32_t tableNumber )`

Sets the number of the table that is current for editing

Parameters

tableNumber	The table number

11.125.3.62 SetDefault() `void SetDefault ( )`

Sets the settings of the valve controller to default

11.125.3.63 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

11.125.3.64 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

11.125.3.65 SetDigitalPortDirection() `void SetDigitalPortDirection (`

```
uint16_t port,
PortDirectionEnumNet direction )
```

Sets the direction of a digital port

Parameters

port	The port number
direction	the direction

11.125.3.66 SetDisplayMode() `void SetDisplayMode (`

```
WvcDisplayModeEnumNet DisplayMode,
int32_t lockTimeMs )
```

Writes the display mode

Parameters

DisplayMode	The display mode
lockTimeMs	Locks the display for ms

11.125.3.67 SetTableName() `void SetTableName (`

```
String^ tableName )
```

Set the name of the current protocol table

Parameters

tableName	The name of the table

11.125.3.68 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

11.125.3.69 SetTableStepAll() `void SetTableStepAll (`

```
int32_t steps )
```

Skips the table protocol for all valves by steps

Parameters

steps	Number of steps

11.125.3.70 SetValveActive() `void SetValveActive (`

```
uint16_t valve,
int valveActive )
```

Sets the valve active/inactive state

Parameters

valve	The valve number
valveActive	The valve state

11.125.3.71 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

11.125.3.72 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

11.125.3.73 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

11.125.3.74 SetValveLedOn() `void SetValveLedOn (`

```
bool ledon )
```

Gets the LED state of the valve board

Parameters

ledon	The LED state

11.125.3.75 SetValveManualGroup() `void SetValveManualGroup (`

```
uint16_t valve,
int32_t valveManualGroup )
```

Sets the valve manual group

Parameters

valve	The valve number
valveManualGroup	The manual valve group

11.125.3.76 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

11.125.3.77 SetValveMode() `void SetValveMode (`

```
uint16_t valve,
WvcValveModeEnumNet ValveMode )
```

Writes the valve mode

Parameters

valve	The valve number
ValveMode	The valve mode

11.125.3.78 SetValvesActiveMap() `void SetValvesActiveMap (`

```
uint32_t valvesActive )
```

Sets the valve active/inactive state

Parameters

valvesActive	The valves states

11.125.3.79 SetValvesManualStateMap() `void SetValvesManualStateMap (`

```
uint32_t valveaManualState )
```

Sets the valve manual on/off state

Parameters

valveaManualState	The manual valves states

11.125.3.80 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

11.125.3.81 StoreValveTable() `void StoreValveTable ( )`

Store the current table in permanent memory

11.125.4 Event Documentation

11.125.4.1 GetActiveRunningTableNumberEvent `OnGetActiveRunningTableNumber^ GetActiveRunningTableNumberEvent [add], [remove], [raise]`

Event fires when the table number has changed

11.125.4.2 GetAnalogThresholdHighEvent `OnGetAnalogThresholdHigh^ GetAnalogThresholdHighEvent [add], [remove], [raise]`

Event fires when the threshold in mV for the valve number has changed

11.125.4.3 GetAnalogThresholdLowEvent `OnGetAnalogThresholdLow^ GetAnalogThresholdLowEvent [add], [remove], [raise]`

Event fires when the threshold in mV for the valve number has changed

11.125.4.4 GetAnalogVoltageEvent

OnGetAnalogVoltage^ GetAnalogVoltageEvent [add], [remove], [raise]

Event fires when the voltage in mV has changed

11.125.4.5 GetCurrentNumberOfValvesEvent

OnGetCurrentNumberOfValves^ GetCurrentNumberOf↩ ValvesEvent [add], [remove], [raise]

Event fires when the number of valves has changed

11.125.4.6 GetDigitalOutPortValveEvent

OnGetDigitalOutPortValve^ GetDigitalOutPortValveEvent [add], [remove], [raise]

Event fires when the valve number for the digital out port has changed

11.125.4.7 GetDigitalPortDirectionEvent

OnGetDigitalPortDirection^ GetDigitalPortDirectionEvent [add], [remove], [raise]

Event fires when the direction for the port number has changed

11.125.4.8 GetDisplayModeEvent

OnGetDisplayMode^ GetDisplayModeEvent [add], [remove], [raise]

Event fires when the display mode has changed

11.125.4.9 GetTableNamebyIndexEvent

OnGetTableNamebyIndex^ GetTableNamebyIndexEvent [add], [remove], [raise]

Event fires when the name of the table for the table number has changed

11.125.4.10 GetValveActiveEvent

OnGetValveActive^ GetValveActiveEvent [add], [remove], [raise]

Event fires when the valve state for the valve number has changed

11.125.4.11 GetValveBoardRevisionEvent OnGetValveBoardRevision^ GetValveBoardRevisionEvent [add], [remove], [raise]

Event fires when the revision code has changed

11.125.4.12 GetValveDigitalInPortEvent OnGetValveDigitalInPort^ GetValveDigitalInPortEvent [add], [remove], [raise]

Event fires when the digital in port for the valve number has changed

11.125.4.13 GetValveLedOnEvent OnGetValveLedOn^ GetValveLedOnEvent [add], [remove], [raise]

Event fires when the LED state has changed

11.125.4.14 GetValveManualGroupEvent OnGetValveManualGroup^ GetValveManualGroupEvent [add], [remove], [raise]

Event fires when the manual valve group for the valve number has changed

11.125.4.15 GetValveManualStateEvent OnGetValveManualState^ GetValveManualStateEvent [add], [remove], [raise]

Event fires when the manual valve state for the valve number has changed

11.125.4.16 GetValveModeEvent OnGetValveMode^ GetValveModeEvent [add], [remove], [raise]

Event fires when the valve mode for the valve number has changed

11.125.4.17 IsDigitalOutPortInvertedEvent OnIsDigitalOutPortInverted^ IsDigitalOutPortInverted↩Event [add], [remove], [raise]

Event fires when is inverted for the digital out port has changed

11.125.4.18 IsValveDigitalInInvertedEvent

`OnIsValveDigitalInInverted^ IsValveDigitalInInvertedEvent [add], [remove], [raise]`

Event fires when is inverted for the valve number has changed

11.125.4.19 IsValveOpenEvent

`OnIsValveOpen^ IsValveOpenEvent [add], [remove], [raise]`

Event fires when is open for the valve number has changed

11.125.4.20 IsValveOpenInAnalogModeEvent

`OnIsValveOpenInAnalogMode^ IsValveOpenInAnalogModeEvent [add], [remove], [raise]`

Event fires when is open for the valve number has changed

11.125.4.21 IsValveOpenInDigitalModeEvent

`OnIsValveOpenInDigitalMode^ IsValveOpenInDigitalModeEvent [add], [remove], [raise]`

Event fires when is open for the valve number has changed

11.125.4.22 TableEntryChangedEvent

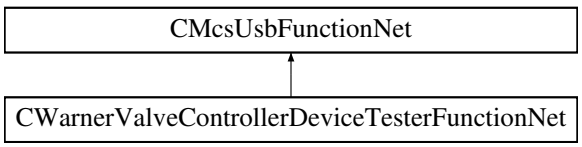
`OnTableEntryChanged^ TableEntryChangedEvent [add], [remove], [raise]`

Event fires when an entry of a table changed

11.126 CWarnerValveControllerDeviceTesterFunctionNet Class Reference

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

Inheritance diagram for CWarnerValveControllerDeviceTesterFunctionNet:



Public Member Functions

- CWarnerValveControllerDeviceTesterFunctionNet (CMcsUsbNet^ 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

11.126.1 Detailed Description

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

11.126.2 Constructor & Destructor Documentation

11.126.2.1 CWarnerValveControllerDeviceTesterFunctionNet() [1/2]

```
CWarnerValveControllerDeviceTesterFunctionNet
(
            CMcsUsbNet^ mcsusb,
            CMcsUsbFunctionPointerContainer^ pWarnerValveControllerDeviceTesterFunction↩
PointerContainer )
```

Initializes a new instance of the CWarnerValveControllerDeviceTesterFunctionNet class.

11.126.2.2 CWarnerValveControllerDeviceTesterFunctionNet() [2/2]

```
CWarnerValveControllerDeviceTesterFunctionNet
(
            CMcsUsbNet^ mcsusb )
```

11.126.2.3 ~CWarnerValveControllerDeviceTesterFunctionNet() virtual ~CWarnerValveControllerDeviceTesterFunctio ( ) [virtual]

11.126.2.4 "!CWarnerValveControllerDeviceTesterFunctionNet() !CWarnerValveControllerDeviceTesterFunctionNet ( )

11.126.3 Member Function Documentation

11.126.3.1 GetIO() uint32_t GetIO ( )

Gets the output from the io ports

Returns

The manual valves states

11.126.3.2 GetSync() uint32_t GetSync ( )

Gets the output from the sync port

Returns

The sync state

11.126.3.3 SetADC() void SetADC (uint32_t *onoff*)

Sets the ADC port of the tester

Parameters

onoff	The port state

11.126.3.4 SetIO() void SetIO (uint32_t *io*)

Sets the input to the io ports

Parameters

io	The manual valves states

11.126.3.5 SetIODirection() `void SetIODirection ( int32_t direction )`

Sets the direction of the IO test ports

Parameters

direction	The 16bit direction map: 1=IN 0=OUT

11.126.3.6 SetTrigger() `void SetTrigger ( uint32_t trigger )`

Sets the input to the trigger port

Parameters

trigger	The trigger state

11.126.3.7 SetTriggerSyncDirection() `void SetTriggerSyncDirection ( uint32_t direction )`

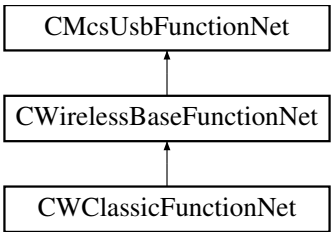
Sets the direction of the trigger/sync test port

Parameters

direction	The direction: 1=IN 0=OUT

11.127 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

11.127.1 Constructor & Destructor Documentation

11.127.1.1 CWClassicFunctionNet() [1/2] CWClassicFunctionNet (

```
CMcsUsbNet^ mcsusb,
CMcsUsbFunctionPointerContainer^ wClassicFuntionPointerContainer )
```

11.127.1.2 CWClassicFunctionNet() [2/2] CWClassicFunctionNet (

```
CMcsUsbNet^ mcsusb )
```

11.127.2 Member Function Documentation

11.127.2.1 GetFilterParametersHeadstage() array<int> ^ GetFilterParametersHeadstage (

```
unsigned short index )
```

11.127.2.2 GetHasChecksum() unsigned int GetHasChecksum (

```
unsigned int Device )
```

11.127.2.3 GetHasRedLedHeadstage() bool GetHasRedLedHeadstage ( )

11.127.2.4 GetHeadstageOnOff() USHORT GetHeadstageOnOff ( )

11.127.2.5 GetResetFilter() unsigned int GetResetFilter (

```
unsigned int Device )
```

11.127.2.6 GetRFConnectionStatus() unsigned int GetRFConnectionStatus ( )

11.127.2.7 GetRFFrequencyHeadstage() unsigned short GetRFFrequencyHeadstage (

```
uint8_t receiver_nb )
```

11.127.2.8 GetRFFrequencyReceiver()

```
unsigned short GetRFFrequencyReceiver (
    uint8_t receiver_nb,
    uint8_t configuration )
```

11.127.2.9 GetRFPower()

```
unsigned short GetRFPower ( )
```

11.127.2.10 GetScanHeadstagesResult()

```
uint8_t GetScanHeadstagesResult (
    int max_wait_for_ms )
```

11.127.2.11 GetSelectedHeadstage()

```
uint8_t GetSelectedHeadstage ( )
```

11.127.2.12 GetSerialNumberHeadstage()

```
unsigned short GetSerialNumberHeadstage ( )
```

11.127.2.13 GetWPADebugMode()

```
unsigned int GetWPADebugMode (
    unsigned int Device )
```

11.127.2.14 GetWPAType()

```
unsigned short GetWPAType (
    unsigned int Device )
```

11.127.2.15 ResetChannelmap()

```
uint32_t ResetChannelmap (
    unsigned int virtualDevice )
```

11.127.2.16 ScanForHeadstages()

```
void ScanForHeadstages ( )
```

11.127.2.17 SetChannelmap()

```
uint32_t SetChannelmap (
    unsigned char position,
    unsigned char channel,
    unsigned int Device )
```

11.127.2.18 SetFilterParametersHeadstage()

```
void SetFilterParametersHeadstage (
    unsigned short index,
    array< int >^ buffer )
```

11.127.2.19 SetHasChecksum()

```
void SetHasChecksum (
    unsigned int has,
    unsigned int Device )
```

11.127.2.20 SetHeadstageOnOff()

```
void SetHeadstageOnOff (
    uint16_t onoff )
```

11.127.2.21 SetHWSelectedChannels()

```
void SetHWSelectedChannels (
    array< bool >^ channels,
    unsigned int Device )
```

11.127.2.22 SetResetFilter()

```
void SetResetFilter (
    unsigned int reset,
    unsigned int Device )
```

11.127.2.23 SetRFFrequencyHeadstage()

```
void SetRFFrequencyHeadstage (
    uint8_t receiver_nb,
    unsigned short frequency )
```

11.127.2.24 SetRFFrequencyReceiver()

```
void SetRFFrequencyReceiver (
    uint8_t receiver_nb,
    uint8_t configuration,
    unsigned short frequency )
```

11.127.2.25 SetRFFrequencyReceiverEeprom()

```
void SetRFFrequencyReceiverEeprom (
    uint8_t receiver_nb,
    uint8_t configuration,
    unsigned short frequency )
```

11.127.2.26 SetRFLostBehaviour()

```
void SetRFLostBehaviour (
    uint8_t stoponfailure,
    unsigned int Device )
```

11.127.2.27 SetRFPower()

```
void SetRFPower (
    unsigned short power )
```

11.127.2.28 SetSelectedHeadstage()

```
void SetSelectedHeadstage (
    uint8_t number )
```

11.127.2.29 SetSerialNumberHeadstage()

```
void SetSerialNumberHeadstage (
    unsigned short number )
```

11.127.2.30 SetWPADebugMode()

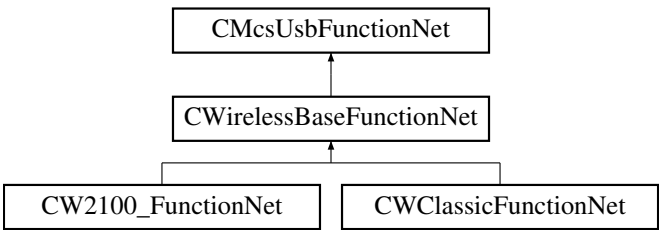
```
void SetWPADebugMode (
    unsigned int mode,
    unsigned int Device )
```

11.127.2.31 SetWPAType()

```
void SetWPAType (
    unsigned short type,
    unsigned int Device )
```

11.128 CWirelessBaseFunctionNet Class Reference

Inheritance diagram for CWirelessBaseFunctionNet:



Public Member Functions

- CWirelessBaseFunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ mcsusbfunction)

Static Public Member Functions

- static String ^ CreateWirelessHeadstageSerialNumberString (unsigned short ID)

Additional Inherited Members

11.128.1 Constructor & Destructor Documentation

11.128.1.1 CWirelessBaseFunctionNet() `CWirelessBaseFunctionNet (`

```
CMcsUsbNet^ mcsusb,
CMcsUsbFunctionPointerContainer^ mcsusbfunction )
```

11.128.2 Member Function Documentation

11.128.2.1 CreateWirelessHeadstageSerialNumberString() `static String ^ CreateWirelessHeadstage↩ SerialNumberString (`

```
unsigned short ID )  [static]
```

11.129 DeviceIdNet Struct Reference

Device Id.

Public Member Functions

- DeviceIdNet ()
- DeviceIdNet (VendorIdEnumNet vendor, ProductIdEnumNet product, int bcd, McsBusTypeEnumNet bustype)
- DeviceIdNet (DeviceIdNet% deviceId)
- DeviceIdNet operator= (DeviceIdNet% deviceId)

Public Attributes

- VendorIdEnumNet IdVendor
- ProductIdEnumNet IdProduct
- int BcdDevice
- McsBusTypeEnumNet BusType

11.129.1 Detailed Description

Device Id.

11.129.2 Constructor & Destructor Documentation

11.129.2.1 DeviceIdNet() [1/3] DeviceIdNet ( )

11.129.2.2 DeviceIdNet() [2/3] DeviceIdNet (

```
VendorIdEnumNet vendor,
ProductIdEnumNet product,
int bcd,
McsBusTypeEnumNet bustype )
```

11.129.2.3 DeviceIdNet() [3/3] DeviceIdNet (

```
DeviceIdNet% deviceId )
```

11.129.3 Member Function Documentation

11.129.3.1 operator=() DeviceIdNet operator= (

```
DeviceIdNet% deviceId )
```

11.129.4 Member Data Documentation

11.129.4.1 BcdDevice int BcdDevice

11.129.4.2 BusType McsBusTypeEnumNet BusType

11.129.4.3 IdProduct ProductIdEnumNet IdProduct

11.129.4.4 IdVendor VendorIdEnumNet IdVendor

11.130 DigitalSource< digitalsourceenum > Class Template Reference

Public Member Functions

- DigitalSource ()
- DigitalSource (digitalsourceenum source)
- int MaxBitNumber ()
- int MaxBitNumber (digitalsourceenum Source)

Static Public Member Functions

- static int MaxBitNumberStatic (digitalsourceenum Source)
- static int size ()

Properties

- digitalsourceenum Source `[get, set]`

11.130.1 Constructor & Destructor Documentation

11.130.1.1 DigitalSource() [1/2] DigitalSource ( )

11.130.1.2 DigitalSource() [2/2] DigitalSource (digitalsourceenum *source*)

11.130.2 Member Function Documentation

11.130.2.1 MaxBitNumber() [1/2] int MaxBitNumber ( )

11.130.2.2 MaxBitNumber() [2/2] int MaxBitNumber (digitalsourceenum *Source*)

11.130.2.3 MaxBitNumberStatic() static int MaxBitNumberStatic (digitalsourceenum *Source*) [static]

11.130.2.4 size() `static int size ( ) [static]`

11.130.3 Property Documentation

11.130.3.1 Source `digitalsourceenum Source [get], [set]`

11.131 DigitalSourceGeneral Class Reference

Public Member Functions

- DigitalSourceGeneral (Type^ 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]`

11.131.1 Constructor & Destructor Documentation

11.131.1.1 DigitalSourceGeneral() [1/2] `DigitalSourceGeneral ( Type^ type )`

11.131.1.2 DigitalSourceGeneral() [2/2] `DigitalSourceGeneral ( Type^ type, int Source )`

11.131.2 Member Function Documentation

11.131.2.1 MaxBitNumber() [1/3] int MaxBitNumber ( )

11.131.2.2 MaxBitNumber() [2/3] int MaxBitNumber (
int *Source*)

11.131.2.3 MaxBitNumber() [3/3] static int MaxBitNumber (
Type^ *type*,
int *Source*) [static]

11.131.2.4 size() static int size (
Type^ *type*) [static]

11.131.3 Property Documentation

11.131.3.1 Source int Source [get], [set]

11.132 DriverVersionNet Class Reference

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

Public Member Functions

- DriverVersionNet ()
  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)

11.132.1 Detailed Description

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

11.132.2 Constructor & Destructor Documentation

11.132.2.1 DriverVersionNet() `DriverVersionNet ( )`

Contructor.

11.132.2.2 ~DriverVersionNet() `~DriverVersionNet ( )`

Destructor.

11.132.3 Member Function Documentation

11.132.3.1 DriverVersionNet::FormatVersion()

```
static String ^ DriverVersionNet::FormatVersion (
    unsigned int v )  [static]
```

11.132.3.2 GetDestinationCode()

```
CFirmwareDestinationNet GetDestinationCode (
    unsigned int index )
```

Get CFirmwareDestinationNet.

Parameters

index	by index of firmware destination

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

Get firmware destination name.

Parameters

dest	by CFirmwareDestionationNet

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

Get firmware destination name.

Parameters

index	by index of firmware destination

11.132.3.5 GetMajor() [1/2] unsigned int GetMajor (CFirmwareDestinationNet *dest*)

Get the major version number of firmware destination.

Parameters

dest	by CFirmwareDestionationNet

11.132.3.6 GetMajor() [2/2] unsigned int GetMajor (unsigned int *index*)

Get the major version number of firmware destination.

Parameters

index	by index of firmware destination

11.132.3.7 GetMinor() [1/2] unsigned int GetMinor (CFirmwareDestinationNet *dest*)

Get the minor version number of firmware destination.

Parameters

dest	by CFirmwareDestionationNet

11.132.3.8 GetMinor() [2/2] unsigned int GetMinor (unsigned int *index*)

Get the minor version number of firmware destination.

Parameters

index	by index of firmware destination

11.132.3.9 GetNumEntries() unsigned int GetNumEntries ( )

Get the number of available firmware destinations.

11.132.3.10 GetSerialNumber() [1/2] String ^ GetSerialNumber (CFirmwareDestinationNet *dest*)

Get the serial number of the destination, when no serial number if found, return an empty string.

Parameters

dest	by CFirmwareDestionationNet

11.132.3.11 GetSerialNumber() [2/2] String ^ GetSerialNumber (unsigned int *index*)

Get the serial number of the destination, when no serial number if found, return an empty string.

Parameters

index	by index of firmware destination

11.132.3.12 GetStatus() [1/2] unsigned int GetStatus (CFirmwareDestinationNet *dest*)

Get status of firmware destination.

Parameters

dest	by CFirmwareDestionationNet

11.132.3.13 GetStatus() [2/2] unsigned int GetStatus (unsigned int *index*)

Get status of firmware destination.

Parameters

index	by index of firmware destination

11.132.3.14 GetVersionInt() [1/2] unsigned int GetVersionInt (CFirmwareDestinationNet *dest*)

Get the version number of firmware destination (major in high word, minor in low word)

Parameters

dest	by CFirmwareDestionationNet

11.132.3.15 GetVersionInt() [2/2] unsigned int GetVersionInt (unsigned int *index*)

Get the version number of firmware destination (major in high word, minor in low word)

Parameters

index	by index of firmware destination

11.132.3.16 GetVersionString() [1/2] String ^ GetVersionString (CFirmwareDestinationNet *dest*)

Get the version as a string in the format Major.Minor.

Parameters

dest	by CFirmwareDestionationNet

11.132.3.17 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

11.133 FirmwareDestinationNames Class Reference

Static Public Attributes

- static String ^ DSP = gcnew String("DSP")
- static String ^ USB = gcnew String("USB")
- static String ^ MCU1 = gcnew String("MCU1")
- static String ^ Bootstrap = gcnew String("Bootstrap")
- static String ^ MCSBUS1 = gcnew String("McsBus1")
- static String ^ MCSBUS2 = gcnew String("McsBus2")
- static String ^ MCSBUS3 = gcnew String("McsBus3")
- static String ^ MCSBUS4 = gcnew String("McsBus4")
- static String ^ MCSBUS5 = gcnew String("McsBus5")
- static String ^ MCSBUS6 = gcnew String("McsBus6")
- static String ^ MCSBUS7 = gcnew String("McsBus7")
- static String ^ MCSBUS8 = gcnew String("McsBus8")
- static String ^ MCSBUS9 = gcnew String("McsBus9")
- static String ^ MCSBUS10 = gcnew String("McsBus10")
- static String ^ MCSBUS11 = gcnew String("McsBus11")
- static String ^ MCSBUS12 = gcnew String("McsBus12")
- static String ^ MCSBUS13 = gcnew String("McsBus13")
- static String ^ BUS1_MCSBUS1 = gcnew String("Bus1McsBus1")
- static String ^ BUS1_MCSBUS2 = gcnew String("Bus1McsBus2")
- static String ^ PIC = gcnew String("PIC")
- static String ^ PIC2 = gcnew String("PIC2")
- static String ^ PIC3 = gcnew String("PIC3")
- static String ^ PIC4 = gcnew String("PIC4")

- static String ^ Altera = gcnew String("Altera")
- static String ^ FPGA2 = gcnew String("FPGA2")
- static String ^ FPGA3 = gcnew String("FPGA3")
- static String ^ FPGA4 = gcnew String("FPGA4")
- static String ^ FPGA5 = gcnew String("FPGA5")
- static String ^ FPGA6 = gcnew String("FPGA6")

11.133.1 Member Data Documentation

11.133.1.1 Altera `String ^ Altera = gcnew String( "Altera" ) [static]`

11.133.1.2 Bootstrap `String ^ Bootstrap = gcnew String( "Bootstrap" ) [static]`

11.133.1.3 BUS1_MCSBUS1 `String ^ BUS1_MCSBUS1 = gcnew String( "Bus1McsBus1" ) [static]`

11.133.1.4 BUS1_MCSBUS2 `String ^ BUS1_MCSBUS2 = gcnew String( "Bus1McsBus2" ) [static]`

11.133.1.5 DSP `String ^ DSP = gcnew String( "DSP" ) [static]`

11.133.1.6 FPGA2 `String ^ FPGA2 = gcnew String( "FPGA2" ) [static]`

11.133.1.7 FPGA3 `String ^ FPGA3 = gcnew String( "FPGA3" ) [static]`

11.133.1.8 FPGA4 `String ^ FPGA4 = gcnew String( "FPGA4" ) [static]`

11.133.1.9 FPGA5 `String ^ FPGA5 = gcnew String( "FPGA5" ) [static]`

11.133.1.10 FPGA6 String ^ FPGA6 = gcnew String("FPGA6") [static]

11.133.1.11 MCSBUS1 String ^ MCSBUS1 = gcnew String("McsBus1") [static]

11.133.1.12 MCSBUS10 String ^ MCSBUS10 = gcnew String("McsBus10") [static]

11.133.1.13 MCSBUS11 String ^ MCSBUS11 = gcnew String("McsBus11") [static]

11.133.1.14 MCSBUS12 String ^ MCSBUS12 = gcnew String("McsBus12") [static]

11.133.1.15 MCSBUS13 String ^ MCSBUS13 = gcnew String("McsBus13") [static]

11.133.1.16 MCSBUS2 String ^ MCSBUS2 = gcnew String("McsBus2") [static]

11.133.1.17 MCSBUS3 String ^ MCSBUS3 = gcnew String("McsBus3") [static]

11.133.1.18 MCSBUS4 String ^ MCSBUS4 = gcnew String("McsBus4") [static]

11.133.1.19 MCSBUS5 String ^ MCSBUS5 = gcnew String("McsBus5") [static]

11.133.1.20 MCSBUS6 String ^ MCSBUS6 = gcnew String("McsBus6") [static]

11.133.1.21 MCSBUS7

```
String ^ MCSBUS7 = gcnew String( "McsBus7" )  [static]
```

11.133.1.22 MCSBUS8

```
String ^ MCSBUS8 = gcnew String( "McsBus8" )  [static]
```

11.133.1.23 MCSBUS9

```
String ^ MCSBUS9 = gcnew String( "McsBus9" )  [static]
```

11.133.1.24 MCU1

```
String ^ MCU1 = gcnew String( "MCU1" )  [static]
```

11.133.1.25 PIC

```
String ^ PIC = gcnew String( "PIC" )  [static]
```

11.133.1.26 PIC2

```
String ^ PIC2 = gcnew String( "PIC2" )  [static]
```

11.133.1.27 PIC3

```
String ^ PIC3 = gcnew String( "PIC3" )  [static]
```

11.133.1.28 PIC4

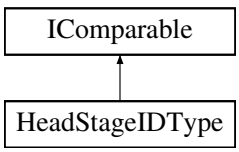
```
String ^ PIC4 = gcnew String( "PIC4" )  [static]
```

11.133.1.29 USB

```
String ^ USB = gcnew String( "USB" )  [static]
```

11.134 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]

11.134.1 Member Enumeration Documentation

11.134.1.1 HeadstageTypeEnum enum HeadstageTypeEnum [strong]

Enumerator

Unknown	
MeasuringOnly	
OpticalStimulation	
ElectricalStimulation	

11.134.2 Constructor & Destructor Documentation

11.134.2.1 HeadStageIDType() HeadStageIDType (

```
unsigned int entry,
CW2100_FunctionNet^ device )
```

11.134.3 Member Function Documentation

11.134.3.1 CompareTo() virtual Int32 CompareTo (

```
Object^ obj )  [virtual]
```

11.134.3.2 Equals() virtual bool Equals (

```
Object^ obj )  [override], [virtual]
```

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

11.134.4 Property Documentation

11.134.4.1 Entry unsigned int Entry [get]

11.134.4.2 HasIMU bool HasIMU [get]

11.134.4.3 HasOptoCurrentMessurement bool HasOptoCurrentMessurement [get]

11.134.4.4 HeadstageType HeadstageTypeEnum HeadstageType [get]

11.134.4.5 ID unsigned short ID [get]

11.134.4.6 NumberOfAnalogChannels `int NumberOfAnalogChannels [get]`

11.134.4.7 NumberOfStimulationChannels `int NumberOfStimulationChannels [get]`

11.134.4.8 SN `System:: String^ SN [get]`

11.134.4.9 StimulusParameters `W2100_StimulusParametersNet^ StimulusParameters [get]`

11.134.4.10 Type `System:: String^ Type [get]`

11.134.4.11 TypeValue `unsigned int TypeValue [get]`

11.134.4.12 UserDefinedName `System:: String^ UserDefinedName [get]`

11.134.4.13 Valid `bool Valid [get]`

11.134.4.14 W16IsW14 `bool W16IsW14 [get]`

11.135 HeadstageIDTypeObject Class Reference

Public Member Functions

- HeadstageIDTypeObject (HeadStageIDType^ 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]

11.135.1 Constructor & Destructor Documentation

11.135.1.1 HeadstageIDTypeObject() HeadstageIDTypeObject (HeadStageIDType^ *idType*)

11.135.2 Member Function Documentation

11.135.2.1 Equals() virtual bool Equals (Object^ *obj*) [override], [virtual]

11.135.2.2 GetHashCode() virtual int GetHashCode ( ) [override], [virtual]

11.135.2.3 ToString() virtual String ^ ToString ( ) [override], [virtual]

11.135.3 Member Data Documentation

11.135.3.1 _AdditionalText String ^ _AdditionalText

11.135.3.2 _IdType HeadStageIDType ^ _IdType

11.135.4 Property Documentation

11.135.4.1 AdditionalText `String^ AdditionalText [get], [set]`

11.135.4.2 IdType `HeadStageIDType^ IdType [get]`

11.136 HeadStageIDTypeState Class Reference

Properties

- unsigned int State `[get]`
- HeadStageIDType^ IdType `[get]`
- bool ControlState `[get]`
- bool DataState `[get]`

11.136.1 Property Documentation

11.136.1.1 ControlState `bool ControlState [get]`

11.136.1.2 DataState `bool DataState [get]`

11.136.1.3 IdType `HeadStageIDType^ IdType [get]`

11.136.1.4 State `unsigned int State [get]`

11.137 mkfilterNet Class Reference

Static Public Member Functions

- static int mkfilter (String^ filtertype, double value, String^ passtype, int order, double alpha1, double alpha2, [System::Runtime::InteropServices::Out] array< double >^% xcoeffs, [System::Runtime::InteropServices::Out] array< double >^% ycoeffs)
- static int mkfilter_MCS (int SamplesPerSecond, double R1, double R2, double C, double Amplification, double Correction, [System::Runtime::InteropServices::Out] array< double >^% xcoeffs, [System::Runtime::InteropServices::Out] array< double >^% ycoeffs)
- static int mkfilter_MCS (int SamplesPerSecond, double R1, double R2, double C, double Correction, [System::Runtime::InteropServices::Out] array< double >^% xcoeffs, [System::Runtime::InteropServices::Out] array< double >^% ycoeffs)
- static int mkfilter_MCS_k (int SamplesPerSecond, double R1, double R2, double C, double Amplification, double Correction, [System::Runtime::InteropServices::Out] array< double >^% coeffs)
- static int mkfilter_MCS_k (int SamplesPerSecond, double R1, double R2, double C, double Correction, [System::Runtime::InteropServices::Out] array< double >^% coeffs)
- static void mkfilter_coef_in_one_set (int n, [System::Runtime::InteropServices::In] array< double >^ xcoeffs, [System::Runtime::InteropServices::In] array< double >^ ycoeffs, [System::Runtime::InteropServices::Out] array< double >^% out_coeffs)
- static void mkfilter_scale_coef_in_one_set (int n, double scale, [System::Runtime::InteropServices::In] array< double >^ xcoeffs, [System::Runtime::InteropServices::In] array< double >^ ycoeffs, [System::Runtime::InteropServices::Out] array< double >^% out_coeffs)
- static void mkfilter_normalize_coeffs_short (short maxvalue, [System::Runtime::InteropServices::In] array< double >^ coeffs, [System::Runtime::InteropServices::Out] array< short >^% out_coeffs)
- static void mkfilter_normalize_coeffs_int (int maxvalue, [System::Runtime::InteropServices::In] array< double >^ coeffs, [System::Runtime::InteropServices::Out] array< int >^% out_coeffs)
- static void mkfilter_normalize_scale_coeffs_int (int maxvalue, [System::Runtime::InteropServices::In] array< double >^ coeffs, [System::Runtime::InteropServices::Out] array< int >^% out_coeffs)
- static double mkfilter_highpass_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)

11.137.1 Member Function Documentation

11.137.1.1 mkfilter()

```
static int mkfilter (
    String^ filtertype,
    double value,
    String^ passtype,
    int order,
    double alpha1,
    double alpha2,
    [System::Runtime::InteropServices::Out] array< double >^% xcoeffs,
    [System::Runtime::InteropServices::Out] array< double >^% ycoeffs ) [static]
```

11.137.1.2 mkfilter_coef_in_one_set()

```
static void mkfilter_coef_in_one_set (
    int n,
    [System::Runtime::InteropServices::In] array< double >^ xcoeffs,
    [System::Runtime::InteropServices::In] array< double >^ ycoeffs,
    [System::Runtime::InteropServices::Out] array< double >^% out_coeffs )  [static]
```

11.137.1.3 mkfilter_highpass_coeff()

```
static double mkfilter_highpass_coeff (
    int SamplesPerSecond,
    double Frequency )  [static]
```

11.137.1.4 mkfilter_highpass_frequency_from_coeff()

```
static double mkfilter_highpass_frequency_from_coeff (
    int SamplesPerSecond,
    double coeff )  [static]
```

11.137.1.5 mkfilter_highpass_frequency_from_k()

```
static double mkfilter_highpass_frequency_from_k (
    int SamplesPerSecond,
    double k )  [static]
```

11.137.1.6 mkfilter_highpass_k()

```
static double mkfilter_highpass_k (
    int SamplesPerSecond,
    double Frequency )  [static]
```

11.137.1.7 mkfilter_MCS() [1/2]

```
static int mkfilter_MCS (
    int SamplesPerSecond,
    double R1,
    double R2,
    double C,
    double Amplification,
    double Correction,
    [System::Runtime::InteropServices::Out] array< double >^% xcoeffs,
    [System::Runtime::InteropServices::Out] array< double >^% ycoeffs )  [static]
```

11.137.1.8 mkfilter_MCS() [2/2]

```
static int mkfilter_MCS (
    int SamplesPerSecond,
    double R1,
    double R2,
    double C,
    double Correction,
    [System::Runtime::InteropServices::Out] array< double >^% xcoeffs,
    [System::Runtime::InteropServices::Out] array< double >^% ycoeffs )  [static]
```

11.137.1.9 mkfilter_MCS_k() [1/2]

```
static int mkfilter_MCS_k (
    int SamplesPerSecond,
    double R1,
    double R2,
    double C,
    double Amplification,
    double Correction,
    [System::Runtime::InteropServices::Out] array< double >^% coeffs )  [static]
```

11.137.1.10 mkfilter_MCS_k() [2/2]

```
static int mkfilter_MCS_k (
    int SamplesPerSecond,
    double R1,
    double R2,
    double C,
    double Correction,
    [System::Runtime::InteropServices::Out] array< double >^% coeffs )  [static]
```

11.137.1.11 mkfilter_normalize_coeffs_int()

```
static void mkfilter_normalize_coeffs_int (
    int maxvalue,
    [System::Runtime::InteropServices::In] array< double >^ coeffs,
    [System::Runtime::InteropServices::Out] array< int >^% out_coeffs )  [static]
```

11.137.1.12 mkfilter_normalize_coeffs_short()

```
static void mkfilter_normalize_coeffs_short (
    short maxvalue,
    [System::Runtime::InteropServices::In] array< double >^ coeffs,
    [System::Runtime::InteropServices::Out] array< short >^% out_coeffs )  [static]
```

11.137.1.13 mkfilter_normalize_scale_coeffs_int()

```
static void mkfilter_normalize_scale_coeffs_int
(
    int maxvalue,
    [System::Runtime::InteropServices::In] array< double >^ coeffs,
    [System::Runtime::InteropServices::Out] array< int >^% out_coeffs )  [static]
```

11.137.1.14 mkfilter_scale_coef_in_one_set()

```
static void mkfilter_scale_coef_in_one_set (
    int n,
    double scale,
    [System::Runtime::InteropServices::In] array< double >^ xcoeffs,
    [System::Runtime::InteropServices::In] array< double >^ ycoeffs,
    [System::Runtime::InteropServices::Out] array< double >^% out_coeffs )  [static]
```

11.138 CRoboDeviceNet::RoboMainLowLevelCommands Class Reference

Public Member Functions

- void SetParameter (unsigned short command, unsigned short index, unsigned int value)
- void SetParameter (unsigned short command, unsigned short index, unsigned int value1, unsigned int value2)
- void SetUserParameter (unsigned short index, unsigned int value)

  Stores persistently 32 bit integer values on RoboMain
- void SetUserParameter (unsigned short index, int value)

  Stores persistently 32 bit integer values on RoboMain
- void GetParameter (unsigned short command, unsigned short index, [System::Runtime::InteropServices::↩ Out]unsigned int% value)
- void GetParameter (unsigned short command, unsigned short index, [System::Runtime::InteropServices::↩ Out]unsigned int% value1, [System::Runtime::InteropServices::Out]unsigned int% value2)
- void GetUserParameter (unsigned short index, [System::Runtime::InteropServices::Out]unsigned int% value)

  Reads 32 bit integer values stored persistently on RoboMain
- void GetUserParameter (unsigned short index, [System::Runtime::InteropServices::Out]int% value)

  Reads 32 bit integer values stored persistently on RoboMain
- void FindReferencePhase0 (unsigned char busaddress, char axes)
- void FindReferencePhase0 (unsigned char busaddress, char axes, int timeout)
- unsigned char HasRef (unsigned char busaddress, char axes)
- void SetHWRevision (unsigned int revision)
- unsigned int GetHWRevision ()
- void SetHWConfig (unsigned int config)
- unsigned int GetHWConfig ()
- void SetMinPressureWaitTime (unsigned int t)
- unsigned int GetMinPressureWaitTime ()
- void SetMinPressure (unsigned int pressure)
- unsigned int GetMinPressure ()
- void SetMaxPressureWaitTime (unsigned int t)
- unsigned int GetMaxPressureWaitTime ()
- void SetMinNoPressureWaitTime (unsigned int t)
- unsigned int GetMinNoPressureWaitTime ()
- void SetMaxNoPressure (unsigned int pressure)
- unsigned int GetMaxNoPressure ()
- void SetMaxNoPressureWaitTime (unsigned int t)
- unsigned int GetMaxNoPressureWaitTime ()
- void SetSearchReferenceMethod (unsigned char busaddress, char axes, unsigned int method)
- unsigned int GetSearchReferenceMethod (unsigned char busaddress, char axes)
- void SetSearchReferenceOffsetPos (unsigned char busaddress, char axes, int offsetpos)
- int GetSearchReferenceOffsetPos (unsigned char busaddress, char axes)
- void SetSearchReferenceFastSpeed (unsigned char busaddress, char axes, unsigned short speed)
- unsigned short GetSearchReferenceFastSpeed (unsigned char busaddress, char axes)
- void SetSearchReferenceFastAccel (unsigned char busaddress, char axes, unsigned short accel)
- unsigned short GetSearchReferenceFastAccel (unsigned char busaddress, char axes)
- void SetSearchReferenceFineSpeed (unsigned char busaddress, char axes, unsigned short speed)
- unsigned short GetSearchReferenceFineSpeed (unsigned char busaddress, char axes)
- void SetSearchReferenceFineAccel (unsigned char busaddress, char axes, unsigned short accel)
- unsigned short GetSearchReferenceFineAccel (unsigned char busaddress, char axes)
- void SetSearchReferenceMoveOut (unsigned char busaddress, char axes, int move)
- int GetSearchReferenceMoveOut (unsigned char busaddress, char axes)
- void SetAxisConfig (unsigned char busaddress, char axes, unsigned int config)
- unsigned int GetAxisConfig (unsigned char busaddress, char axes)
- void GetPhases (unsigned char busaddress, char axes, [System::Runtime::InteropServices::Out] unsigned short% phase0, [System::Runtime::InteropServices::Out] unsigned short% lastphase)

11.138.1 Member Function Documentation

11.138.1.1 FindReferencePhase0() [1/2]

```
void FindReferencePhase0 (
    unsigned char busaddress,
    char axes )
```

11.138.1.2 FindReferencePhase0() [2/2]

```
void FindReferencePhase0 (
    unsigned char busaddress,
    char axes,
    int timeout )
```

11.138.1.3 GetAxisConfig()

```
unsigned int GetAxisConfig (
    unsigned char busaddress,
    char axes )
```

11.138.1.4 GetHWConfig()

```
unsigned int GetHWConfig ( )
```

11.138.1.5 GetHWRevision()

```
unsigned int GetHWRevision ( )
```

11.138.1.6 GetMaxNoPressure()

```
unsigned int GetMaxNoPressure ( )
```

11.138.1.7 GetMaxNoPressureWaitTime()

```
unsigned int GetMaxNoPressureWaitTime ( )
```

11.138.1.8 GetMaxPressureWaitTime()

```
unsigned int GetMaxPressureWaitTime ( )
```

11.138.1.9 GetMinNoPressureWaitTime()

```
unsigned int GetMinNoPressureWaitTime ( )
```

11.138.1.10 GetMinPressure() unsigned int GetMinPressure ( )

11.138.1.11 GetMinPressureWaitTime() unsigned int GetMinPressureWaitTime ( )

11.138.1.12 GetParameter() [1/2] void GetParameter (

```
unsigned short command,
unsigned short index,
[System::Runtime::InteropServices::Out] unsigned int% value )
```

11.138.1.13 GetParameter() [2/2] void GetParameter (

```
unsigned short command,
unsigned short index,
[System::Runtime::InteropServices::Out] unsigned int% value1,
[System::Runtime::InteropServices::Out] unsigned int% value2 )
```

11.138.1.14 GetPhases() void GetPhases (

```
unsigned char busaddress,
char axes,
[System::Runtime::InteropServices::Out] unsigned short% phase0,
[System::Runtime::InteropServices::Out] unsigned short% lastphase )
```

11.138.1.15 GetSearchReferenceFastAccel() unsigned short GetSearchReferenceFastAccel (

```
unsigned char busaddress,
char axes )
```

11.138.1.16 GetSearchReferenceFastSpeed() unsigned short GetSearchReferenceFastSpeed (

```
unsigned char busaddress,
char axes )
```

11.138.1.17 GetSearchReferenceFineAccel() unsigned short GetSearchReferenceFineAccel (

```
unsigned char busaddress,
char axes )
```

11.138.1.18 GetSearchReferenceFineSpeed()

```
unsigned short GetSearchReferenceFineSpeed (
    unsigned char busaddress,
    char axes )
```

11.138.1.19 GetSearchReferenceMethod()

```
unsigned int GetSearchReferenceMethod (
    unsigned char busaddress,
    char axes )
```

11.138.1.20 GetSearchReferenceMoveOut()

```
int GetSearchReferenceMoveOut (
    unsigned char busaddress,
    char axes )
```

11.138.1.21 GetSearchReferenceOffsetPos()

```
int GetSearchReferenceOffsetPos (
    unsigned char busaddress,
    char axes )
```

11.138.1.22 GetUserParameter() [1/2]

```
void GetUserParameter (
    unsigned short index,
    [System::Runtime::InteropServices::Out] int% value )
```

Reads 32 bit integer values stored persistently on RoboMain

intention: provide free persistent user memory space on motor controller

Parameters

index	address offset of parameter; range: 0..15
value	data buffer

11.138.1.23 GetUserParameter() [2/2]

```
void GetUserParameter (
    unsigned short index,
    [System::Runtime::InteropServices::Out] unsigned int% value )
```

Reads 32 bit integer values stored persistently on RoboMain

intention: provide free persistent user memory space on motor controller

Parameters

index	address offset of parameter; range: 0..15
value	data buffer

11.138.1.24 HasRef()

```
unsigned char HasRef (
    unsigned char busaddress,
    char axes )
```

11.138.1.25 SetAxisConfig()

```
void SetAxisConfig (
    unsigned char busaddress,
    char axes,
    unsigned int config )
```

11.138.1.26 SetHWConfig()

```
void SetHWConfig (
    unsigned int config )
```

11.138.1.27 SetHWRevision()

```
void SetHWRevision (
    unsigned int revision )
```

11.138.1.28 SetMaxNoPressure()

```
void SetMaxNoPressure (
    unsigned int pressure )
```

11.138.1.29 SetMaxNoPressureWaitTime()

```
void SetMaxNoPressureWaitTime (
    unsigned int t )
```

11.138.1.30 SetMaxPressureWaitTime()

```
void SetMaxPressureWaitTime (
    unsigned int t )
```

11.138.1.31 SetMinNoPressureWaitTime()

```
void SetMinNoPressureWaitTime (
    unsigned int t )
```

11.138.1.32 SetMinPressure()

```
void SetMinPressure (
    unsigned int pressure )
```

11.138.1.33 SetMinPressureWaitTime()

```
void SetMinPressureWaitTime (
    unsigned int t )
```

11.138.1.34 SetParameter() [1/2]

```
void SetParameter (
    unsigned short command,
    unsigned short index,
    unsigned int value )
```

11.138.1.35 SetParameter() [2/2]

```
void SetParameter (
    unsigned short command,
    unsigned short index,
    unsigned int value1,
    unsigned int value2 )
```

11.138.1.36 SetSearchReferenceFastAccel()

```
void SetSearchReferenceFastAccel (
    unsigned char busaddress,
    char axes,
    unsigned short accel )
```

11.138.1.37 SetSearchReferenceFastSpeed()

```
void SetSearchReferenceFastSpeed (
    unsigned char busaddress,
    char axes,
    unsigned short speed )
```

11.138.1.38 SetSearchReferenceFineAccel()

```
void SetSearchReferenceFineAccel (
    unsigned char busaddress,
    char axes,
    unsigned short accel )
```

11.138.1.39 SetSearchReferenceFineSpeed()

```
void SetSearchReferenceFineSpeed (
    unsigned char busaddress,
    char axes,
    unsigned short speed )
```

11.138.1.40 SetSearchReferenceMethod()

```
void SetSearchReferenceMethod (
    unsigned char busaddress,
    char axes,
    unsigned int method )
```

11.138.1.41 SetSearchReferenceMoveOut()

```
void SetSearchReferenceMoveOut (
    unsigned char busaddress,
    char axes,
    int move )
```

11.138.1.42 SetSearchReferenceOffsetPos()

```
void SetSearchReferenceOffsetPos (
    unsigned char busaddress,
    char axes,
    int offsetpos )
```

11.138.1.43 SetUserParameter() [1/2]

```
void SetUserParameter (
    unsigned short index,
    int value )
```

Stores *persistently* 32 bit integer values on RoboMain

intention: provide free persistent user memory space on RoboMain

Parameters

index	address offset of parameter; range: 0..15
value	data to be stored

11.138.1.44 SetUserParameter() [2/2]

```
void SetUserParameter (
    unsigned short index,
    unsigned int value )
```

Stores *persistently* 32 bit integer values on RoboMain

intention: provide free persistent user memory space on RoboMain

Parameters

index	address offset of parameter; range: 0..15
value	data to be stored

11.139 CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands Class Reference

Public Member Functions

- void FindReferencePhase0XY ()
- void FindReferencePhase0XY (int timeout)

11.139.1 Member Function Documentation

11.139.1.1 FindReferencePhase0XY() [1/2] void FindReferencePhase0XY ( )

11.139.1.2 FindReferencePhase0XY() [2/2] void FindReferencePhase0XY (int *timeout*)

11.140 CFilterCoefficientsNet::s_FilterAttributesNet Struct Reference

Public Member Functions

- s_FilterAttributesNet (s_FilterAttributes attrib)
- s_FilterAttributes ToCpp ()

Public Attributes

- uint32_t PreCommaB
- uint32_t PostCommaB
- uint32_t CommaPositionB
- uint32_t PreCommaA
- uint32_t PostCommaA
- uint32_t CommaPositionA

11.140.1 Constructor & Destructor Documentation

11.140.1.1 s_FilterAttributesNet() s_FilterAttributesNet (s_FilterAttributes *attrib*)

11.140.2 Member Function Documentation

11.140.2.1 ToCpp() s_FilterAttributes ToCpp ( )

11.140.3 Member Data Documentation

11.140.3.1 CommaPositionA uint32_t CommaPositionA

11.140.3.2 CommaPositionB uint32_t CommaPositionB

11.140.3.3 PostCommaA uint32_t PostCommaA

11.140.3.4 PostCommaB uint32_t PostCommaB

11.140.3.5 PreCommaA uint32_t PreCommaA

11.140.3.6 PreCommaB uint32_t PreCommaB

11.141 CMeaAudioFunctionNet::s_setaudionet Struct Reference

Public Attributes

- int channel
- int amplification

11.141.1 Member Data Documentation

11.141.1.1 amplification int amplification

11.141.1.2 channel `int channel`

11.142 CStimulusFunctionNet::SidebandData Class Reference

Public Member Functions

- SidebandData ()
- ~SidebandData ()

  Destructor: called by Dispose()
- !SidebandData ()

  Finalizer: called by GC before collecting

Properties

- array< int32_t >^ Sideband `[get]`
- array< uint64_t >^ Duration `[get]`

11.142.1 Constructor & Destructor Documentation

11.142.1.1 SidebandData() `SidebandData ( )`

11.142.1.2 ~SidebandData() `~SidebandData ( )`

Destructor: called by Dispose()

11.142.1.3 "!SidebandData() `!SidebandData ( )`

Finalizer: called by GC before collecting

11.142.2 Property Documentation

11.142.2.1 Duration `array< uint64_t>^ Duration [get]`

11.142.2.2 Sideband `array< int32_t>^ Sideband [get]`

11.143 StgStatusNet Class Reference

Static Public Member Functions

- static StgStatusNet ^ FromIntPtr (IntPtr stgstatus)
- static StgStatusNet ^ FromPtr (stgstatus_t *stgstatus)

Public Attributes

- array< Stg200xTriggerStatusEnumNet > ^ TiggerStatus
- array< uint32_t > ^ ListOfChangedTriggers

11.143.1 Member Function Documentation

11.143.1.1 FromIntPtr() `static StgStatusNet ^ FromIntPtr ( IntPtr stgstatus ) [static]`

11.143.1.2 FromPtr() `static StgStatusNet ^ FromPtr ( stgstatus_t * stgstatus ) [static]`

11.143.2 Member Data Documentation

11.143.2.1 ListOfChangedTriggers `array<uint32_t> ^ ListOfChangedTriggers`

11.143.2.2 TiggerStatus `array<Stg200xTriggerStatusEnumNet> ^ TiggerStatus`

11.144 CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData Class Reference

Public Member Functions

- StimulusDeviceDataAndUnrolledData ()
- ~StimulusDeviceDataAndUnrolledData ()

  Destructor: called by Dispose()
- !StimulusDeviceDataAndUnrolledData ()

  Finalizer: called by GC before collecting

Properties

- array< uint8_t >^ DeviceData [get]
- int DeviceDataLength [get]
- array< int32_t >^ UnrolledAmplitude [get]
- array< uint32_t >^ UnrolledSync [get]
- array< uint64_t >^ UnrolledDuration [get]

11.144.1 Constructor & Destructor Documentation

11.144.1.1 StimulusDeviceDataAndUnrolledData() StimulusDeviceDataAndUnrolledData ( )

11.144.1.2 ~StimulusDeviceDataAndUnrolledData() ~StimulusDeviceDataAndUnrolledData ( )

Destructor: called by Dispose()

11.144.1.3 "!StimulusDeviceDataAndUnrolledData() !StimulusDeviceDataAndUnrolledData ( )

Finalizer: called by GC before collecting

11.144.2 Property Documentation

11.144.2.1 DeviceData array< uint8_t>^ DeviceData [get]

11.144.2.2 DeviceDataLength int DeviceDataLength [get]

11.144.2.3 UnrolledAmplitude array< int32_t>^ UnrolledAmplitude [get]

11.144.2.4 UnrolledDuration array< uint64_t>^ UnrolledDuration [get]

11.144.2.5 UnrolledSync `array< uint32_t>^ UnrolledSync [get]`

11.145 usbSetupPacket_t Class Reference

Public Attributes

- uint8_t bmRequestType
- uint8_t bRequest
- uint16_t wValue
- uint16_t wIndex
- uint16_t wLength

11.145.1 Member Data Documentation

11.145.1.1 bmRequestType `uint8_t bmRequestType`

11.145.1.2 bRequest `uint8_t bRequest`

11.145.1.3 wIndex `uint16_t wIndex`

11.145.1.4 wLength `uint16_t wLength`

11.145.1.5 wValue `uint16_t wValue`

11.146 W2100_StimulusParametersNet Struct Reference

Public Attributes

- int DACResolution
- int TimeResolutionInNanoSeconds
- int VoltageRangeInMicroVolt
- int VoltageResolutionInMicroVolt
- int CurrentRangeInNanoAmp
- int CurrentResolutionInNanoAmp

11.146.1 Member Data Documentation

11.146.1.1 CurrentRangeInNanoAmp `int CurrentRangeInNanoAmp`

11.146.1.2 CurrentResolutionInNanoAmp `int CurrentResolutionInNanoAmp`

11.146.1.3 DACResolution `int DACResolution`

11.146.1.4 TimeResolutionInNanoSeconds `int TimeResolutionInNanoSeconds`

11.146.1.5 VoltageRangeInMicroVolt `int VoltageRangeInMicroVolt`

11.146.1.6 VoltageResolutionInMicroVolt `int VoltageResolutionInMicroVolt`

Index

!CDacCalibrationFunctionNet
CDacCalibrationFunctionNet, 113
!CDigOutStimulatorFunctionNet
CDigOutStimulatorFunctionNet, 123
!CExternDTesterDeviceNet
CExternDTesterDeviceNet, 128
!CGrapheneFunctionNet
CGrapheneFunctionNet, 168
!CInterfaceboard2FunctionNet
CInterfaceboard2FunctionNet, 185
!CInterfaceboardFunctionNet
CInterfaceboardFunctionNet, 187
!CLIH3DeviceNet
CLIH3DeviceNet, 190
!CMEA2100x256FunctionNet
CMEA2100x256FunctionNet, 331
!CMcsUsbFunctionNet
CMcsUsbFunctionNet, 299
!CMcsUsbListNet
CMcsUsbListNet, 306
!CMcsUsbNet
CMcsUsbNet, 312
!CMeFunctionNet
CMeFunctionNet, 364
!CMeaCleanDeviceNet
CMeaCleanDeviceNet, 336
!CMeaCoatDeviceNet
CMeaCoatDeviceNet, 340
!CMultiBatteryChargerDeviceNet
CMultiBatteryChargerDeviceNet, 366
!CMultiwellCallbackFunctionNet
CMultiwellCallbackFunctionNet, 373
!CMultiwellDeviceNet
CMultiwellDeviceNet, 375
!CMultiwellOptoStimFunctionNet
CMultiwellOptoStimFunctionNet, 381
!CPPCFunctionNet
CPPCFunctionNet, 415
!CPedoterDeviceNet
CPedoterDeviceNet, 396
!CPositionIIDeviceNet
CPositionIIDeviceNet, 402
!CPositionImpDeviceNet
CPositionImpDeviceNet, 410
!CProgramPressureCurveNet
CProgramPressureCurveNet, 429
!CPulseGeneratorFunctionNet
CPulseGeneratorFunctionNet, 430
!CRFFunctionNet
CRFFunctionNet, 438
!CSCUFunctionNet
CSCUFunctionNet, 494
!CTEERFunctionNet
CTEERFunctionNet, 584
!CUsbDeviceConfigurationFunctionNet
CUsbDeviceConfigurationFunctionNet, 594
!CWarnerUssingDeviceNet
CWarnerUssingDeviceNet, 612
!CWarnerUssingFunctionNet
CWarnerUssingFunctionNet, 614
!CWarnerValveControllerDeviceNet
CWarnerValveControllerDeviceNet, 633
!CWarnerValveControllerDeviceTesterFunctionNet
CWarnerValveControllerDeviceTesterFunctionNet, 655
!SidebandData
CStimulusFunctionNet::SidebandData, 693
!StimulusDeviceDataAndUnrolledData
CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, 695
_AdditionalText
HeadstageIDTypeObject, 679
_IdType
HeadstageIDTypeObject, 679
~CCMOSMeaDeviceNet
CCMOSMeaDeviceNet, 108
~CChannelTestDeviceNet
CChannelTestDeviceNet, 96
~CCreateFilterNet
CCreateFilterNet, 111
~CDacCalibrationFunctionNet
CDacCalibrationFunctionNet, 113
~CDigOutStimulatorFunctionNet
CDigOutStimulatorFunctionNet, 123
~CExternDTesterDeviceNet
CExternDTesterDeviceNet, 128
~CFilterCoefficientsNet
CFilterCoefficientsNet, 130
~CFilterPropertyNet
CFilterPropertyNet, 135
~CFluidControlDeviceNet
CFluidControlDeviceNet, 138
~CGenericDevelopDeviceNet
CGenericDevelopDeviceNet, 152
~CGilsonDeviceNet
CGilsonDeviceNet, 164
~CGrapheneFunctionNet
CGrapheneFunctionNet, 168
~CInterfaceboard2FunctionNet
CInterfaceboard2FunctionNet, 185
~CInterfaceboardFunctionNet
CInterfaceboardFunctionNet, 187
~CLIH3DeviceNet
CLIH3DeviceNet, 190
~CMEA2100x256FunctionNet
CMEA2100x256FunctionNet, 331
~CMcsBusNet
CMcsBusNet, 236
~CMcsBus_AxisParametersNet
CMcsBus_AxisParametersNet, 198

~CMcsBus_ExtensionNet
CMcsBus_ExtensionNet, 199
~CMcsBus_FYIExtensionNet
CMcsBus_FYIExtensionNet, 200
~CMcsBus_MotorControlNet
CMcsBus_MotorControlNet, 205
~CMcsBus_SensorNet
CMcsBus_SensorNet, 221
~CMcsBus_TempSensorNet
CMcsBus_TempSensorNet, 230
~CMcsBus_VoltageModeNet
CMcsBus_VoltageModeNet, 232
~CMcsUsbDacqNet
CMcsUsbDacqNet, 246
~CMcsUsbFactoryNet
CMcsUsbFactoryNet, 291
~CMcsUsbFunctionNet
CMcsUsbFunctionNet, 299
~CMcsUsbListEntryNet
CMcsUsbListEntryNet, 300
~CMcsUsbListNet
CMcsUsbListNet, 306
~CMcsUsbNet
CMcsUsbNet, 312
~CMeFunctionNet
CMeFunctionNet, 364
~CMeaCleanDeviceNet
CMeaCleanDeviceNet, 335
~CMeaCoatDeviceNet
CMeaCoatDeviceNet, 340
~CMeaDeviceNet
CMeaDeviceNet, 346
~CMeaImpedanceDeviceNet
CMeaImpedanceDeviceNet, 358
~CMeaSwitchDeviceNet
CMeaSwitchDeviceNet, 361
~CMeaUSBDeviceNet
CMeaUSBDeviceNet, 363
~CMultiBatteryChargerDeviceNet
CMultiBatteryChargerDeviceNet, 366
~CMultiwellCallbackFunctionNet
CMultiwellCallbackFunctionNet, 373
~CMultiwellDeviceNet
CMultiwellDeviceNet, 375
~CMultiwellOptoStimFunctionNet
CMultiwellOptoStimFunctionNet, 381
~CNF_GenDeviceNet
CNF_GenDeviceNet, 385
~COkuvisionStimulatorDeviceNet
COkuvisionStimulatorDeviceNet, 391
~CPPCFunctionNet
CPPCFunctionNet, 415
~CPathIdentDeviceNet
CPathIdentDeviceNet, 395
~CPedoterDeviceNet
CPedoterDeviceNet, 396
~CPeristalticPumpDeviceNet
CPeristalticPumpDeviceNet, 397
~CPgaDeviceNet
CPgaDeviceNet, 398
~CPositionIIDeviceNet
CPositionIIDeviceNet, 402
~CPositionImpDeviceNet
CPositionImpDeviceNet, 410
~CProgramPressureCurveNet
CProgramPressureCurveNet, 428
~CPulseGeneratorFunctionNet
CPulseGeneratorFunctionNet, 430
~CRFFunctionNet
CRFFunctionNet, 438
~CRetinaLedDeviceNet
CRetinaLedDeviceNet, 436
~CRoboDeviceNet
CRoboDeviceNet, 463
~CRoboFluidDeviceNet
CRoboFluidDeviceNet, 477
~CSCUFunctionNet
CSCUFunctionNet, 494
~CSafeISDeviceNet
CSafeISDeviceNet, 488
~CStg200xBasicNet
CStg200xBasicNet, 512
~CStg200xDownloadNet
CStg200xDownloadNet, 550
~CSw2to64DeviceNet
CSw2to64DeviceNet, 567
~CTEERFunctionNet
CTEERFunctionNet, 584
~CTEERMachineDeviceNet
CTEERMachineDeviceNet, 592
~CTcxDeviceNet
CTcxDeviceNet, 571
~CUsbDeviceConfigurationFunctionNet
CUsbDeviceConfigurationFunctionNet, 594
~CWarnerUssingDeviceNet
CWarnerUssingDeviceNet, 611
~CWarnerUssingFunctionNet
CWarnerUssingFunctionNet, 614
~CWarnerValveControllerDeviceNet
CWarnerValveControllerDeviceNet, 633
~CWarnerValveControllerDeviceTesterFunctionNet
CWarnerValveControllerDeviceTesterFunctionNet, 654
~DriverVersionNet
DriverVersionNet, 667
~SidebandData
CStimulusFunctionNet::SidebandData, 693
~StimulusDeviceDataAndUnrolledData
CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, 695

A
CFilterCoefficientsNet, 131
Mcs::Usb, 67
AccelOnly
Mcs::Usb, 87
AdapterTypeEnumNet

Mcs::Usb, 50
AdditionalText
HeadstageIDTypeObject, 680
AddLoopEntry
CRetinaLedDeviceNet, 436
AddSelectedChannelsQueue
CMcsUsbDacqNet, 246–248
AddSoftwareKey
CMcsUsbNet, 312
AddTableEntry
CRetinaLedDeviceNet, 436
ALA_VC3
Mcs::Usb, 73, 87
ALA_VC3_DEVICE
Mcs::Usb, 59
ALTERA
Mcs::Usb, 52
Altera
FirmwareDestinationNames, 673
ALTERA_BASE
Mcs::Usb, 53
ALTERA_BOOTSTRAP
Mcs::Usb, 53
ALTERA_GOLD
Mcs::Usb, 53
ALTERA_TARGET1
Mcs::Usb, 54
ALTERA_TARGET2
Mcs::Usb, 54
ALTERA_TARGET3
Mcs::Usb, 54
AlwaysOn
Mcs::Usb, 76
Ampere
Mcs::Usb, 51, 87
amplification
CMeaAudioFunctionNet::s_setaudionet, 692
CW2100_FunctionNet::AudioChannelsNet, 92
AmplifierSettle
CIntanMea_FunctionNet, 183
AMS_Dongle
Mcs::Usb, 74
Analog
Mcs::Usb, 73, 90
AnalogGain
CMeaDeviceNet, 351
AnalogGroup
Mcs::Usb, 58
AnalogOut_DAC_Range_EnumNet
Mcs::Usb, 51
AnalogSource_HS1
Mcs::Usb, 51
AnalogSource_HS2
Mcs::Usb, 51
AnalogSource_IF
Mcs::Usb, 51
AnalogSourceEnumNet
Mcs::Usb, 51
AnalogUnitEnumNet
Mcs::Usb, 51
Any
Mcs::Usb, 73, 87
ApplyGains
CPgaDeviceNet, 399
AreTransistorVoltagesSet
CCMOSMea_FunctionNet, 99
Armed
Mcs::Usb, 81
ASMedia
Mcs::Usb, 86
AssociateToThis
CMcsUsbNet, 312
AudioTestChannelGroup
Mcs::Usb, 57, 69, 79, 89
AutomaticAnalogOut
CSCUFunctionNet, 494
Aux
Mcs::Usb, 60
AuxIn
Mcs::Usb, 60, 70, 79, 82, 89
AuxPort
Mcs::Usb, 56
Axes_I
CRoboDeviceNet, 471
Axes_X
CRoboDeviceNet, 471
Axes_Y
CRoboDeviceNet, 471
Axes_Z
CRoboDeviceNet, 472
Axis_I
CRoboDeviceNet, 472
Axis_X
CRoboDeviceNet, 472
Axis_Y
CRoboDeviceNet, 472
Axis_Z
CRoboDeviceNet, 472

B
CFilterCoefficientsNet, 131
Mcs::Usb, 67
BatteryState, 92
Charge, 92
ChargeRegionString, 92
ChargeString, 92
Voltage, 93
VoltageString, 93
BcdDevice
DeviceIdNet, 663
BeginImpedanceCheck
CIntanMea_FunctionNet, 183
Bessel
Mcs::Usb, 64
BesselFilterHighPassNet, 93
BesselFilterHighPassNet, 93
BesselFilterLowPassNet, 93

BesselFilterLowPassNet, 94
BMI
Mcs::Usb, 83
bmRequestType
usbSetupPacket_t, 696
BOOST_BIT
CW2100_StimulatorFunctionNet, 610
Bootstrap
FirmwareDestinationNames, 673
Mcs::Usb, 52
BootstrapOtherCypress
Mcs::Usb, 52
Both
Mcs::Usb, 87
Break
Mcs::Usb, 77
bRequest
usbSetupPacket_t, 696
BurnAdcOffset
COctoPotDeviceNet, 387
BurnDacOffset
CDacCalibrationFunctionNet, 113
COctoPotDeviceNet, 387
BUS1_MCSBUS1
FirmwareDestinationNames, 673
BUS1_MCSBUS2
FirmwareDestinationNames, 673
BUS1MCSBUS1
Mcs::Usb, 52
BUS1MCSBUS2
Mcs::Usb, 52
BUS2MCSBUS1
Mcs::Usb, 52
BUS2MCSBUS2
Mcs::Usb, 52
BUSNUMBER1
Mcs::Usb, 52
BUSNUMBER2
Mcs::Usb, 52
BusType
DeviceIdNet, 663
Butterworth
Mcs::Usb, 64
ButterworthFilterHighPassNet, 94
ButterworthFilterHighPassNet, 94
ButterworthFilterLowPassNet, 95
ButterworthFilterLowPassNet, 95

CalibrateThermocouple
CFluidControlDeviceNet, 138
CTcxDeviceNet, 571
Campden_Ci4600EphysVideoDataIntegrator
Mcs::Usb, 73
CancelInternalCalibration
CTEERFunctionNet, 584
CancelPoolLoop
CRoboDeviceNet, 463
CancelPoolLoopAndStopMovement
CRoboDeviceNet, 463
CancelTableLoop
CRoboDacqNet, 448
CancelTableLoopAndStopTable
CRoboDacqNet, 448
CapacityTest
CMultiBatteryChargerDeviceNet, 366
CatchAmp
Mcs::Usb, 72
CatchAmpGetAdcMean
CMcsBus_SensorNet, 221
CatchAmpGetAdcValue
CMcsBus_SensorNet, 221
CatchAmpGetAdcValueH
CMcsBus_SensorNet, 221
CatchAmpGetAdcValueL
CMcsBus_SensorNet, 221
CatchAmpGetDacAmplitude
CMcsBus_SensorNet, 222
CatchAmpGetDacEnable
CMcsBus_SensorNet, 222
CatchAmpGetDacOffset
CMcsBus_SensorNet, 222
CatchAmpGetPwmEnable
CMcsBus_SensorNet, 222
CatchAmpSetDacAmplitude
CMcsBus_SensorNet, 222
CatchAmpSetDacEnable
CMcsBus_SensorNet, 222
CatchAmpSetDacOffset
CMcsBus_SensorNet, 222
CatchAmpSetPwmEnable
CMcsBus_SensorNet, 222
CChannelTestDeviceNet, 95
~CChannelTestDeviceNet, 96
CChannelTestDeviceNet, 96
SetAmplitude, 96
SetAttenuation, 96
SetFrequency, 96
SetWaveform, 96
CCMOSMea_FunctionNet, 96
AreTransistorVoltagesSet, 99
CCMOSMea_FunctionNet, 98, 99
ClearSTGOutput, 99
DetectChipType, 99
EnableChannelsInGroup, 99
GetADCInputOffset, 99
GetBath, 99
GetBathMode, 99
GetEnabledChannelsInGroup, 100
GetGate, 100
GetGNDI, 100
GetGroupADCBits, 100
GetGroupChannelBitmaskBySelect, 100
GetGroupChannelBitmaskHS1NCBathCurrent, 100, 101
GetGroupChannelBitmaskHS1NCCol2Current, 101
GetGroupChannelBitmaskHS1NChipTemp, 101

GetGroupChannelBitmaskHS1Sidebands, 101
GetGroupChannelBitmaskHS1TriggerStatus, 101, 102
GetGroupChannelBitmaskIFDigChannels, 102
GetGroupChannelBitmaskInterfaceADC, 102
GetGroupChannelBitmaskPacketFrameContext, 102
GetGroupChannelBitmaskSTG1DACSignal, 102, 103
GetGroupDCOffset, 103
GetGroupID, 103
GetGroupNumberOfChannels, 103
GetGroupResolutionPerDigit, 103
GetGroupSampleSize, 104
GetGroupType, 104
GetGroupUnit, 104
GetMaxNumOfColumns, 104
GetNeurochipMemoryData, 104
GetNeurochipMemorySize, 105
GetNumberOfSupportedGroups, 105
GetSourceBulk, 105
GetSourceDrain, 105
GetSourceGate, 105
GetStimulusSites, 105
GetVDD3I, 105
GetVDDI, 105
IsChipPowered, 105
IsGateFloating, 105
PowerChip, 106
SetADCInputOffset, 106
SetBath, 106
SetBathMode, 106
SetGate, 106
SetGateFloating, 106
SetGateToVOP, 106
SetNeurochipMemoryData, 106
SetSourceBulk, 106
SetSourceDrain, 107
SetSourceGate, 107
SetStimulusSites, 107
UpdateTransistorVoltages, 107
VOPSTimerSetResetTimes, 107
CCMOSMeaDeviceNet, 107
~CCMOSMeaDeviceNet, 108
CCMOSMeaDeviceNet, 108
CMosMea, 110
GetAvailableBaseSamplerates, 108
GetBaseSamplerate, 109
GetChannelDataI16, 109
GetChannelDataI32, 109
GetChannelDataUI16, 109
GetChannelDataUI32, 109
GetCMOSDataDictionary, 109
GetMaxReadableColumns, 109
SetBaseSamplerate, 109
SetRegionOfInterests, 110
Stimulus, 110
UpdateChannelBlock, 110
CCMOSMeaDeviceNet::CRegionOfInterestRect, 434
CRegionOfInterestRect, 434
DeepCopy, 434
m_Bottom, 435
m_Left, 435
m_Right, 435
m_Top, 435
CCreateFilterNet, 110
~CCreateFilterNet, 111
CCreateFilterNet, 111
CutoffFrequency, 112
FindFilter, 111
GetBiQuad, 112
GetBiQuads, 112
NumCoefSets, 112
Order, 112
SampleRate, 112
Scale, 112
CDacCalibrationFunctionNet, 112
!CDacCalibrationFunctionNet, 113
~CDacCalibrationFunctionNet, 113
BurnDacOffset, 113
CDacCalibrationFunctionNet, 113
GetDacOffset, 114
SetDacOffset, 114
CDacqGroupChannelGenericSelectionNet, 114
CDacqGroupChannelGenericSelectionNet, 115
CDacqGroupChannelSelectionNet, 115
CDacqGroupChannelSelectionNet, 115
CDacqGroupChannelSelectionTemplateNet
CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet >, 116
CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet >, 115
CDacqGroupChannelSelectionTemplateNet, 116
EnableChannelsInGroup, 116
GetDeviceGroupChannelInfos, 116, 117
GetEnabledChannelsInGroup, 117
GetGroupID, 117
GetGroupNumberOfChannels, 117
GetGroupSampleSize, 117
GetGroupType, 118
GetNumberOfSupportedGroups, 118
CDeviceGroupChannelInfoGenericNet, 118
CDeviceGroupChannelInfoGenericNet, 118
CDeviceGroupChannelInfoMEA2100_256Net, 119
CDeviceGroupChannelInfoMEA2100_256Net, 119
CDeviceGroupChannelInfoNet, 119
CDeviceGroupChannelInfoNet, 120
CDeviceGroupChannelInfoSCUNet, 120
CDeviceGroupChannelInfoSCUNet, 120
CDeviceGroupChannelInfoTemplateNet
CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumTemplateNet >, 121

CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumTemplateNet >, 120
CDeviceGroupChannelInfoTemplateNet, 121
GroupID, 121
GroupType, 121
NumberOfChannels, 121
CDeviceGroupChannelInfoW2100Net, 121
CDeviceGroupChannelInfoW2100Net, 122
CDigOutStimulatorFunctionNet, 122
!CDigOutStimulatorFunctionNet, 123
~CDigOutStimulatorFunctionNet, 123
CDigOutStimulatorFunctionNet, 123
ClearChannel, 123
GetGlobalRepeat, 123
GetNumberOfChannels, 124
GetStartTriggerSlope, 124
GetStopTriggerSlope, 124
PrepareChannelData, 125
SendChannelData, 125
SetGlobalRepeat, 125
SetStartTriggerSlope, 126
SetStopTriggerSlope, 126
CEncapsulatorDeviceNet, 126
CEncapsulatorDeviceNet, 127
GetRoboFluidDevice, 127
CExternDTesterDeviceNet, 127
!CExternDTesterDeviceNet, 128
~CExternDTesterDeviceNet, 128
CExternDTesterDeviceNet, 128
Read, 128
Read2, 128
Write, 128
Write2, 129
CFilterCoefficientsNet, 129
~CFilterCoefficientsNet, 130
A, 131
B, 131
CFilterCoefficientsNet, 129, 130
GetUintA, 130
GetUintB, 130
IsEqual, 130
CFilterCoefficientsNet::s_FilterAttributesNet, 691
CommaPositionA, 692
CommaPositionB, 692
PostCommaA, 692
PostCommaB, 692
PreCommaA, 692
PreCommaB, 692
s_FilterAttributesNet, 691
ToCpp, 691
CFilterConfigurationNet, 131
CFilterConfigurationNet, 131
EraseFilterParameterPermanent, 132
GetFilterAttributes, 132
GetHighpassFilterEnable, 132
ResetHighpassFilter, 132
SetFilterParameter, 132
SetFilterParameterPermanent, 132
SetHighpassFilterEnable, 133
CFilterConfigurationRegisterNet, 133
CFilterConfigurationRegisterNet, 133
EraseFilterParameterPermanent, 133, 134
SetFilterParameter, 134
SetFilterParameterPermanent, 134
CFilterPropertyNet, 135
~CFilterPropertyNet, 135
CFilterPropertyNet, 135
CornerFrequency, 135
CornerFrequencymHz, 136
FilterActive, 136
FilterBand, 136
FilterFamily, 136
FilterType, 136
Order, 136
ToString, 135
CFirmwareDestinationNet
Mcs::Usb, 51
CFluidControlDeviceNet, 136
~CFluidControlDeviceNet, 138
CalibrateThermocouple, 138
CFluidControlDeviceNet, 138
GetAdc, 138
GetDigin, 138
GetDigout, 139
GetPWM, 139
GetReferenceTemperature, 139
GetSingleValve, 139
GetThermocoupleCalibration, 140
GetThermocoupleNanovoltPerKelvin, 140
GetThermocoupleTemperature, 140
GetValve, 141
McsBus_VoltageMode, 143
SetDigout, 141
SetPWM, 141
SetSingleValve, 141
SetThermocoupleNanovoltPerKelvin, 143
SetValve, 143
CFYIDeviceNet, 143
CFYIDeviceNet, 144
FYIProgram, 144
FYITemp, 144
Sensor, 144
CGenericDevelopDeviceNet, 145
~CGenericDevelopDeviceNet, 152
CGenericDevelopDeviceNet, 152
ClosePipe, 152
GetBuffer, 152
GetByteBuffer, 152
GetIntBuffer, 153
GetShortBuffer, 154
GetUByteBuffer, 154
GetUIntBuffer, 156
GetUShortBuffer, 157
OpenPipe, 157
ReadPipe, 158
ResetPipe, 158

SetBuffer, 159
SetByteBuffer, 159
SetIntBuffer, 159
SetShortBuffer, 160
SetUByteBuffer, 160
SetUIntBuffer, 161
SetUShortBuffer, 162
SetValue, 162
VendorInRequest, 163
VendorOutRequest, 163
WritePipe, 163
CGilsonDeviceNet, 163
~CGilsonDeviceNet, 164
CGilsonDeviceNet, 164
ConnectSlave, 164
GetLastAnswer, 165
m_pGilsonDevice, 165
SendBuffered, 165
SendImmediate, 165
SendImmediateGetResponse, 165
CGrapheneFunctionNet, 165
!CGrapheneFunctionNet, 168
~CGrapheneFunctionNet, 168
CGrapheneFunctionNet, 167
GetCur2VolResistance, 168
GetDACOffset, 168, 169
GetVds, 169
GetVdVs, 169, 170
GetVdVsDAC, 170
GetVgs, 171
GetVoltageRange, 171
GetVoltageReached, 172
GetVoltageResolution, 172
SetDACOffset, 173
SetVds, 173
SetVdVs, 174
SetVdVsDAC, 174, 175
SetVgs, 175
SetVoltageRange, 175, 176
SetVoltageResolution, 176
ChangeSerialNumber
CMcsUsbFactoryNet, 291
channel
CMeaAudioFunctionNet::s_setaudionet, 692
CW2100_FunctionNet::AudioChannelsNet, 92
ChannelBlock_AvailFrames
CMcsUsbDacqNet, 249
ChannelBlock_ReadAsFrameArrayI16
CMcsUsbDacqNet, 249, 250
ChannelBlock_ReadAsFrameArrayI32
CMcsUsbDacqNet, 251
ChannelBlock_ReadAsFrameArrayUI16
CMcsUsbDacqNet, 252, 253
ChannelBlock_ReadAsFrameArrayUI32
CMcsUsbDacqNet, 253, 254
ChannelBlock_ReadFramesDictI16
CMcsUsbDacqNet, 255
ChannelBlock_ReadFramesDictI32
CMcsUsbDacqNet, 255
ChannelBlock_ReadFramesDictUI16
CMcsUsbDacqNet, 256
ChannelBlock_ReadFramesDictUI32
CMcsUsbDacqNet, 256
ChannelBlock_ReadFramesI16
CMcsUsbDacqNet, 257, 258
ChannelBlock_ReadFramesI32
CMcsUsbDacqNet, 258, 259
ChannelBlock_ReadFramesUI16
CMcsUsbDacqNet, 259, 260
ChannelBlock_ReadFramesUI32
CMcsUsbDacqNet, 261
channeldata_current
Mcs::Usb, 82
channeldata_current_always_boost
Mcs::Usb, 82
channeldata_current_always_boost_own_sync
Mcs::Usb, 82
channeldata_current_own_boost_gnd_sync
Mcs::Usb, 82
channeldata_current_own_sync
Mcs::Usb, 82
channeldata_positive_current
Mcs::Usb, 82
channeldata_positive_current_own_boost_gnd_sync
Mcs::Usb, 82
channeldata_positive_current_own_sync
Mcs::Usb, 82
channeldata_positive_voltage
Mcs::Usb, 82
channeldata_voltage
Mcs::Usb, 82
ChannelDataEvent
CMcsUsbDacqNet, 287
ChannelPIC
Mcs::Usb, 52
ChannelReset
CMultiBatteryChargerDeviceNet, 367
ChannelTest
Mcs::Usb, 75
Charge
BatteryState, 92
ChargeRegionString
BatteryState, 92
ChargeString
BatteryState, 92
ChecksumAndPacketCounter
Mcs::Usb, 56
CHiClampDeviceNet, 176
CHiClampDeviceNet, 177
RoboDacq, 177
CHLADacqNet, 177
CHLADacqNet, 178
CHLADeviceNet, 178
CHLADeviceNet, 178
HLADacq, 179
SerialPort, 179

CHWInfo
CMcsUsbDacqNet::CHWInfo, 180
Ci4600Intan
Mcs::Usb, 50
CIntanMea_FunctionNet, 182
AmplifierSettle, 183
BeginImpedanceCheck, 183
CIntanMea_FunctionNet, 182
GetDSPHighPassByIndex, 183
GetImpedanceResult, 183
GetIntanRegister, 183
GetLowerFrequencyByIndex, 183
GetUpperFrequencyByIndex, 183
SetBandwidthByIndex, 183
SetDiagnosticMode, 183
SetDSPHighPassByIndex, 184
SetIntanRegister, 184
CInterfaceboard2FunctionNet, 184
!CInterfaceboard2FunctionNet, 185
~CInterfaceboard2FunctionNet, 185
CInterfaceboard2FunctionNet, 185
GetIoVoltage, 185
SetIoVoltage, 185
CInterfaceboardFunctionNet, 186
!CInterfaceboardFunctionNet, 187
~CInterfaceboardFunctionNet, 187
CInterfaceboardFunctionNet, 186
GetCardinalDacqSamplerate, 187
GetCardinalStgOutputrate, 187
SetCardinalDacqSamplerate, 187
SetCardinalStgOutputrate, 187
ClampAmpRestart
CRoboDacqNet, 448
ClampModeCurrent
Mcs::Usb, 84
ClampModeInternalCalibration
Mcs::Usb, 84
ClampModeOpen
Mcs::Usb, 84
ClampModeVoltage
Mcs::Usb, 83
ClearBuffers
CMcsUsbDacqNet, 262
ClearChannel
CDigOutStimulatorFunctionNet, 123
ClearChannel_PrepareAndSendData
CStg200xDownloadNet, 550
CStimulusFunctionNet, 557
ClearChannelData
CStg200xDownloadBasicNet, 542
CStimulusFunctionNet, 557
CW2100_StimulatorFunctionNet, 605
ClearMultiplexedData
CStimulusFunctionNet, 557
ClearSTGOutput
CCMOSMea_FunctionNet, 99
ClearStimulusParametersCache
CW2100_FunctionNet, 598
ClearSyncData
CStg200xDownloadBasicNet, 543
CStimulusFunctionNet, 557
ClearTable
CRetinaLedDeviceNet, 436
ClearTableName
CWarnerValveControllerDeviceNet, 633
ClearUserDefinedNameCache
CW2100_FunctionNet, 598
ClearValveTable
CWarnerValveControllerDeviceNet, 633
CLIH3DeviceNet, 188
!CLIH3DeviceNet, 190
~CLIH3DeviceNet, 190
CLIH3DeviceNet, 190
DummyCommand, 190
EnableUserTrigger, 190
ErasePermanentAdcOffset, 191
ErasePermanentDacOffset, 191
GetAdcOffset, 191
GetAudioOutDacParameter, 191
GetDacIdleValue, 192
GetDacOffset, 192
GetDacqRunStatus, 192
GetDacUseIdleValue, 192
GetDigInState, 193
GetEEpromPage, 193
GetSampleInterval, 193
IsUserTriggerEnabled, 193
ReadClipping, 194
ReadUARTData, 194
SendCommand, 194
SetAdcOffset, 194
SetAdcOffsetPermanent, 195
SetAudioOutDacParameter, 195
SetDacIdleValue, 195
SetDacOffset, 195
SetDacOffsetPermanent, 196
SetDacUseIdleValue, 196
SetDigOutState, 196
SetEEpromPage, 196
SetSampleInterval, 197
StimulusFunction, 197
WriteUARTData, 197
Close
Mcs::Usb, 72
CloseAllValves
CRoboFluidDeviceNet, 477
ClosePipe
CGenericDevelopDeviceNet, 152
ClosePlateClamp
CMultiwellDeviceNet, 376
CMcsBus_AxisParametersNet, 197
~CMcsBus_AxisParametersNet, 198
CMcsBus_AxisParametersNet, 198
GetAxisParametersSignedEeprom, 198
GetAxisParametersUnsignedEeprom, 198
SetAxisParametersEeprom, 198, 199

CMcsBus_ExtensionNet, 199
~CMcsBus_ExtensionNet, 199
CMcsBus_ExtensionNet, 199
GetLEDSwitch, 200
SetLEDSwitch, 200
CMcsBus_FYIExtensionNet, 200
~CMcsBus_FYIExtensionNet, 200
CMcsBus_FYIExtensionNet, 200
GetDIO, 201
GetSingleHeater, 201
GetValves, 201
SetDIO, 201
SetSingleHeater, 201
SetValves, 201
CMcsBus_MotorControlNet, 202
~CMcsBus_MotorControlNet, 205
CMcsBus_MotorControlNet, 205
GetMCAcceleration, 205
GetMCAccelerationEeprom, 205
GetMCAccelerationShortCommand, 205
GetMCAxisRevisionEeprom, 205
GetMCBreakCurrent, 205
GetMCBreakCurrentEeprom, 206
GetMCConfig, 206
GetMCConfigEeprom, 206
GetMCCurrent, 206
GetMCCurrentEeprom, 206
GetMCCurrentMode, 206
GetMCCurrentModeEeprom, 206
GetMCCurrentModeShortCommand, 207
GetMCCurrentPosition, 207
GetMCCurrentShortCommand, 207
GetMCCurrentSpeed, 207
GetMCMaxAcceleration, 207
GetMCMaxAccelerationEeprom, 207
GetMCMaxCurrent, 207
GetMCMaxCurrentEeprom, 208
GetMCMaxSpeed, 208
GetMCMaxSpeedEeprom, 208
GetMCMaxTravel, 208
GetMCMaxTravelEeprom, 208
GetMCMaxTravelShortCommand, 208
GetMCMovement, 208
GetMCNewPosition, 209
GetMCOutputOnOff, 209
GetMCPhase, 209
GetMCPhaseOffset, 209
GetMCReference, 209
GetMCReferenceCurrent, 209
GetMCReferenceCurrentEeprom, 209
GetMCRegulatorGain, 210
GetMCRegulatorGainEeprom, 210
GetMCScalingFactor, 210
GetMCScalingFactorEeprom, 210
GetMCSpeed, 210
GetMCSpeedEeprom, 210
GetMCSpeedShortCommand, 210
GetMCSpeedUnitEeprom, 211
GetMCStandbyCurrent, 211
GetMCStandbyCurrentEeprom, 211
GetMCStandbyTime, 211
GetMCStandbyTimeEeprom, 211
GetSubChannel, 211
SetMCAcceleration, 211
SetMCAccelerationEeprom, 212
SetMCAccelerationShortCommand, 212
SetMCAxisRevisionEeprom, 212
SetMCBreakCurrent, 212
SetMCBreakCurrentEeprom, 212
SetMCConfig, 212
SetMCConfigEeprom, 213
SetMCCurrent, 213
SetMCCurrentEeprom, 213
SetMCCurrentMode, 213
SetMCCurrentModeEeprom, 213
SetMCCurrentModeShortCommand, 213
SetMCCurrentPosition, 214
SetMCCurrentShortCommand, 214
SetMCMaxAcceleration, 214
SetMCMaxAccelerationEeprom, 214
SetMCMaxCurrent, 214
SetMCMaxCurrentEeprom, 214
SetMCMaxSpeed, 215
SetMCMaxSpeedEeprom, 215
SetMCMaxTravel, 215
SetMCMaxTravelEeprom, 215
SetMCMaxTravelShortCommand, 215
SetMCNewPosition, 215
SetMCOutputOnOff, 216
SetMCReference, 216
SetMCReferenceCurrent, 216
SetMCReferenceCurrentEeprom, 216
SetMCRegulatorGain, 216
SetMCRegulatorGainEeprom, 216
SetMCRotation, 217
SetMCScalingFactor, 217
SetMCScalingFactorEeprom, 217
SetMCSpeed, 217
SetMCSpeedEeprom, 217
SetMCSpeedShortCommand, 217
SetMCSpeedUnitEeprom, 218
SetMCStandbyCurrent, 218
SetMCStandbyCurrentEeprom, 218
SetMCStandbyTime, 218
SetMCStandbyTimeEeprom, 218
SetSubChannel, 218
StartMCMovement, 219
StopMCMovement, 219
CMcsBus_SensorNet, 219
~CMcsBus_SensorNet, 221
CatchAmpGetAdcMean, 221
CatchAmpGetAdcValue, 221
CatchAmpGetAdcValueH, 221
CatchAmpGetAdcValueL, 221
CatchAmpGetDacAmplitude, 222
CatchAmpGetDacEnable, 222

CatchAmpGetDacOffset, 222
CatchAmpGetPwmEnable, 222
CatchAmpSetDacAmplitude, 222
CatchAmpSetDacEnable, 222
CatchAmpSetDacOffset, 222
CatchAmpSetPwmEnable, 222
CMcsBus_SensorNet, 221
Get2AnalogInput, 223
Get2DigitalInput, 223
Get4ADC, 223
Get4ADCAverage, 223
Get4ADCCatchampAverageShift, 223
Get4ADCMode, 223
Get4DAC, 223
GetADCs, 223
GetADCsLoop, 224
GetBubbleStatus, 224
GetDACs, 224
GetDetectionThreshold, 224
GetDetectorValue, 224
GetLatency, 224
GetLatencyCounter, 224
GetMinimalThreshold, 224
GetMovePump, 225
GetPiezoState, 225
GetPressure, 225
GetPressureOffset, 225
GetRegulationTimeouts, 225
GetRegulatorFactor, 226
GetRegulatorOnOff, 226
GetRegulatorStatus, 226
GetRotatePump, 226
GetSamplePeriode, 226
GetSollPressure, 226
GetSyncState, 226
Set4ADCCatchampAverageShift, 227
Set4ADCMode, 227
Set4DAC, 227
SetDACs, 227
SetDetectionThreshold, 227
SetLatency, 227
SetMinimalThreshold, 227
SetMovePump, 228
SetPiezoState, 228
SetPressureOffset, 228
SetRegulationTimeouts, 228
SetRegulatorFactor, 228
SetRegulatorOnOff, 228
SetRotatePump, 228
SetSamplePeriode, 229
SetSollPressure, 229
StartSync, 229
TactSwitchGetState, 229
TactSwitchSetDisplay, 229
CMcsBus_TempSensorNet, 229
~CMcsBus_TempSensorNet, 230
CMcsBus_TempSensorNet, 230
GetNanoVoltsPerKelvin, 230
GetTemperatur, 230
GetThermoOffset, 230
GetThermoTemp, 231
GetThermoVoltage, 231
SetNanoVoltsPerKelvin, 231
SetThermoOffset, 231
CMcsBus_VoltageModeNet, 231
~CMcsBus_VoltageModeNet, 232
CMcsBus_VoltageModeNet, 232
GetVMMaxNegativeCurrent, 233
GetVMMaxNegativeCurrentEeprom, 233
GetVMMaxNegativeVoltage, 233
GetVMMaxNegativeVoltageEeprom, 233
GetVMMaxPositiveCurrent, 233
GetVMMaxPositiveCurrentEeprom, 233
GetVMMaxPositiveVoltage, 233
GetVMMaxPositiveVoltageEeprom, 234
GetVMOutputOnOff, 234
GetVMVoltage, 234
SetVMMaxNegativeCurrent, 234
SetVMMaxNegativeCurrentEeprom, 234
SetVMMaxNegativeVoltage, 234
SetVMMaxNegativeVoltageEeprom, 234
SetVMMaxPositiveCurrent, 235
SetVMMaxPositiveCurrentEeprom, 235
SetVMMaxPositiveVoltage, 235
SetVMMaxPositiveVoltageEeprom, 235
SetVMOutputOnOff, 235
SetVMVoltage, 235
CMcsBusNet, 236
~CMcsBusNet, 236
CMcsBusNet, 236
CMcsBusNet::GetMode, 237
CMcsBusNet::GetModeEeprom, 237
CMcsBusNet::SetMode, 237
CMcsBusNet::SetModeEeprom, 237
GetBusAddress, 237
GetBusAddressEeprom, 237
GetCommand, 237, 238
GetHWRevisionEeprom, 238
SetBusAddress, 238
SetBusAddressEeprom, 238
SetCommand, 239
SetHWRevisionEeprom, 239
CMcsBusNet::GetMode
CMcsBusNet, 237
CMcsBusNet::GetModeEeprom
CMcsBusNet, 237
CMcsBusNet::SetMode
CMcsBusNet, 237
CMcsBusNet::SetModeEeprom
CMcsBusNet, 237
CMcsUsbDacqNet, 240
~CMcsUsbDacqNet, 246
AddSelectedChannelsQueue, 246–248
ChannelBlock_AvailFrames, 249
ChannelBlock_ReadAsFrameArrayI16, 249, 250
ChannelBlock_ReadAsFrameArrayI32, 251

ChannelBlock_ReadAsFrameArrayUI16, 252, 253
ChannelBlock_ReadAsFrameArrayUI32, 253, 254
ChannelBlock_ReadFramesDictI16, 255
ChannelBlock_ReadFramesDictI32, 255
ChannelBlock_ReadFramesDictUI16, 256
ChannelBlock_ReadFramesDictUI32, 256
ChannelBlock_ReadFramesI16, 257, 258
ChannelBlock_ReadFramesI32, 258, 259
ChannelBlock_ReadFramesUI16, 259, 260
ChannelBlock_ReadFramesUI32, 261
ChannelDataEvent, 287
ClearBuffers, 262
CMcsUsbDacqNet, 246
CMcsUsbDacqNet::GetFilterProperties, 262
Error_Callback_Aquisition_Stopped, 286
Error_Callback_Data_lost, 286
Error_Callback_Frames_Lost, 286
Error_Callback_Packet_Error, 286
Error_Callback_Queue_Full, 287
Error_Callback_RingQueue_Full, 287
ErrorEvent, 287
GetAdapterType, 262
GetAdcDataFormat, 262
GetAdcZero, 262
GetAnalogValueUnit, 263
GetChannelDataFillSize, 263
GetChannelLayout, 263
GetChannelsInBlock, 263
GetDataFormat, 263
GetDataMode, 263
GetDigitalSource, 264, 265
GetErrorMessage, 266
GetFilterProperty, 266
GetGroupChannelDataI16, 266
GetGroupChannelDataI32, 266
GetGroupChannelDataUI16, 267
GetGroupChannelDataUI32, 267
GetHardwareMaxRange, 268
GetHardwareMinRange, 268
GetMaxSamplingFrequency, 268
GetMeaLayout, 268
GetMinSamplingFrequencyStepsize, 269
GetNumberOfDataBits, 269
GetPoti, 269
GetResolutionPerDigit, 269
GetSamplerate, 269
GetVoltageRangeIndex, 269
GetVoltageRangeInMicroVolt, 270
GetVoltageRangeInMilliVolt, 270
HWInfo, 270
Samplerate, 287
SendStartDacq, 270
SendStartStgAndDacq, 270
SendStopDacq, 271
SendStopStgAndDacq, 271
SendStopStgAndDacqWithOptions, 271
SetDataMode, 272
SetDigitalSource, 272–274
SetPoti, 274
SetSamplerate, 274
SetSelectedChannels, 275–277
SetSelectedChannelsQueue, 277–279
SetSelectedData, 279–281
SetupGroupDacqQueue, 281
SetVoltageRangeByIndex, 281
SetVoltageRangeInMicroVolt, 282
StartDacq, 282, 283
StartLoop, 284, 285
StopDacq, 286
StopLoop, 286
CMcsUsbDacqNet::CHWInfo, 179
CHWInfo, 180
GetAvailableSampleRates, 180
GetAvailableVoltageRangesInMicroVolt, 180
GetAvailableVoltageRangesInMicroVoltAnd-StringsInMilliVolt, 180
GetNumberOfHWADCChannels, 181
GetNumberOfHWDigitalChannels, 181
IsDigitalChannelDedicated, 181
CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet, 595
CVoltageRangeInfoNet, 596
VoltageRangeDisplayStringMilliVolt, 596
VoltageRangeInMicroVolt, 596
CMcsUsbDacqNet::GetFilterProperties
CMcsUsbDacqNet, 262
CMcsUsbDeviceStatePushFunctionNet, 287
CMcsUsbDeviceStatePushFunctionNet, 288
McsUsbDeviceStateEvent, 288
TriggerStatus, 288
CMcsUsbDeviceStatePushNet, 288
CMcsUsbDeviceStatePushNet, 289
McsUsbDeviceStateEvent, 289
TriggerStatus, 289
CMcsUsbFactoryNet, 289
∼CMcsUsbFactoryNet, 291
ChangeSerialNumber, 291
CMcsUsbFactoryNet, 291
Coldstart, 291
CompareFirmware, 292
DownloadFirmware, 292
FindFirmwareVersionMagicInBuffer, 292
FX3MCSDataAddress, 297
FX3MCSDataDeviceIdOffset, 297
FX3MCSDataIFB1ImageOffset, 297
FX3MCSDataIFB2ImageOffset, 297
FX3MCSDataVersionOffset, 297
GetChecksumFromFX3Image, 292
GetDestination, 292
GetDestinationDisplayLabel, 292
GetDestinationName, 292, 293
GetDestinationSerialNumber, 293
GetDestinationTargetAddress, 293
GetFirmwareVersionFromFile, 293
GetFirmwareVersionFromHexFile, 293
GetNumDestinations, 293

GetUSBDeviceIDFromFX3Image, 294
GetUsercodeFromBitFile, 294
GetUsercodeFromFlash, 294
GetXilinxFlashOffset, 294
GetXilinxFlashReadCommand, 294
LoadUserFirmware, 294, 295
ReadBlockFromFlash, 295
ReadBlockFromIFBGlobalEEprom, 295
ReadBlockFromNVMEM, 295
SetDestinationSerialNumber, 295
UpdateFirmware, 295–297
CMcsUsbFunctionNet, 298
!CMcsUsbFunctionNet, 299
~CMcsUsbFunctionNet, 299
CMcsUsbFunctionNet, 298, 299
m_pMcsUsb, 299
m_pMcsUsbFunction, 299
ThrowCUsbExceptionNetOnError, 299
CMcsUsbFunctionPointerContainer, 299
CMcsUsbListEntryNet, 299
~CMcsUsbListEntryNet, 300
DeviceId, 304
DeviceName, 304
Equals, 301
GetEntry, 302
GetEntryCount, 303
HwVersion, 304
Manufacturer, 304
Product, 304
SerialNumber, 304
SetStringFormat, 303
ToString, 304
CMcsUsbListNet, 305
!CMcsUsbListNet, 306
~CMcsUsbListNet, 306
CMcsUsbListNet, 305
Count, 307
DeviceArrival, 307
DeviceRemoval, 307
GetNumberOfDevices, 306
GetUsbListEntries, 306
GetUsbListEntry, 306
IsDeviceTypeOf, 307
SetStringFormat, 307
CMcsUsbNet, 308
!CMcsUsbNet, 312
~CMcsUsbNet, 312
AddSoftwareKey, 312
AssociateToThis, 312
CMcsUsbNet, 312
Connect, 312, 313
CyclePort, 314
Disconnect, 314
EmptyKey, 314
EnableExceptions, 314
EraseEepromRegisterPreconfig, 314
GetConfiguration, 315
GetDeviceCannotStallOutRequests, 315
GetDeviceCapableSpeed, 315
GetDeviceEnum, 315
GetDeviceId, 315
GetDeviceRootHubVendorEnum, 315
GetDeviceRootHubVendorID, 315
GetDeviceRootHubVendorName, 315
GetDeviceSpeed, 316
GetErrorText, 316
GetFirmwareVersion, 316
GetHardwareRevision, 316
GetHeadstageActive, 317
GetHeadstageID, 317
GetHeadstagePresent, 317
GetIdent, 317
GetLastUSBError, 318
GetMea21UsbPort, 318
GetNumConfigurations, 318
GetSerialNumber, 318
GetSoftwareKey, 318
GetSoftwareKeyString, 318
GetStatus, 318
GetStatusOfLastCommand, 319
GetUsbListEntry, 319
GetVersion, 319
HasSoftwareKey, 319
IsConnected, 319
IsDeviceHighSpeed, 319
IsDeviceHighSpeedCapable, 320
IsExceptionsEnabled, 320
MultibootGetCypressImageId, 320
MultibootGetImageId, 320
MultibootGetSelectedImage, 320
MultibootSelectImage, 320
ReadEepromRegisterPreconfig, 321
ReadRegister, 321
ReadRegister32, 321
ReadRegisterTimeSlot, 321
RemoveSoftwareKey, 322
RescanHeadstage, 322
SerialNumber, 329
SetConfiguration, 322
SetSoftwareKey, 322
Status_AlreadyConfigured, 324
Status_BadStartFrame, 325
Status_Btstuff, 325
Status_BufferOverrun, 325
Status_BufferUnderrun, 325
Status_Canceled, 325
Status_Canceling, 325
Status_ConnectedPipes, 325
Status_ControlNotOwned, 325
Status_Crc, 325
Status_DataOverrun, 325
Status_DataToggleMismatch, 325
Status_DataUnderrun, 326
Status_DeviceLocked, 326
Status_DeviceNotFound, 326
Status_DeviceRemoved, 326

Status_DevNotResponding, 326
Status_EndpointHalted, 326
Status_ErrorBusy, 326
Status_ErrorShortTransfer, 326
Status_Fifo, 326
Status_FrameControlOwned, 326
Status_InternalHcError, 326
Status_InvalidDeviceHandle, 327
Status_InvalidHandle, 327
Status_InvalidParameter, 327
Status_InvalidPipeHandle, 327
Status_InvalidUrbFunction, 327
Status_IoPending, 327
Status_IoTimeout, 327
Status_IsochRequestFailed, 327
Status_LastUsbErrorMismatch, 327
Status_NoBandwidth, 327
Status_NoMemory, 328
Status_NoSuchDevice, 328
Status_NotAccessed, 328
Status_NotSupported, 328
Status_PidCheckFailure, 328
Status_PipeNotLinked, 328
Status_RequestFailed, 328
Status_RequestMutexFailed, 328
Status_RequestMutexTimeout, 328
Status_Stall, 328
Status_Unconfigured, 328
Status_UnexpectedPid, 329
ThrowCUsbExceptionNetOnError, 322
TxnGetSerialNumber, 322
TxnSetSerialNumber, 322
TxnTestMemoryReadAndCheck, 322
TxnTestMemoryWrite, 323
ValidKey, 323
WPAError_ScanningIsPending, 329
WriteEepromRegisterPreconfig, 323
WriteRegister, 323, 324
WriteRegister32, 324
WriteRegisterArray, 324
WriteRegisterTimeSlot, 324
WriteRegisterValue, 324
CMcsUsbPointerContainer, 329
CMEA2100_256DacqGroupChannelSelectionNet, 329
CMEA2100_256DacqGroupChannelSelectionNet, 329
CMEA2100x256FunctionNet, 330
!CMEA2100x256FunctionNet, 331
~CMEA2100x256FunctionNet, 331
CMEA2100x256FunctionNet, 330
GetLayoutConfiguration, 331
SetLayoutConfiguration, 331
CMeaAudioFunctionNet, 331
CMeaAudioFunctionNet, 332
GetAudioChannels, 332, 333
GetNumberOfAudioChannels, 333
SetAudioChannels, 333, 334
CMeaAudioFunctionNet::s_setaudionet, 692
amplification, 692
channel, 692
CMeaCleanDeviceNet, 334
!CMeaCleanDeviceNet, 336
~CMeaCleanDeviceNet, 335
CMeaCleanDeviceNet, 335
GetCycle, 336
GetCycles, 336
GetMaxVoltage, 336
GetMinVoltage, 336
GetOutputVoltage, 336
GetSlope, 337
IsRunning, 337
SetCycles, 337
SetMaxVoltage, 337
SetMinVoltage, 338
SetSlope, 338
Start, 338
Stop, 338
CMeaCoatDeviceNet, 338
!CMeaCoatDeviceNet, 340
~CMeaCoatDeviceNet, 340
CMeaCoatDeviceNet, 340
GetCurrentCycle, 340
GetCycles, 340
GetDuration, 340
GetMaxCurrent, 340
GetOffsetCurrent, 341
GetOutputCurrent, 341
GetPauseDuration, 341
GetSlope, 341
GetTimeInPause, 341
GetTimeInPlateau, 342
IsRunning, 342
SetCycles, 342
SetDuration, 342
SetMaxCurrent, 343
SetOffsetCurrent, 343
SetPauseDuration, 343
SetSlope, 343
Start, 343
Stop, 344
CMeaDeviceNet, 344
~CMeaDeviceNet, 346
AnalogGain, 351
CMeaDeviceNet, 345
EnableChecksum, 346
EnableDigitalIn, 346, 347
EnableTimestamp, 347
Gain, 351
GetAnalogGain, 348
GetEnumerationSpeed, 348
GetGain, 348
MeaAudioFunctionNet, 351
MeaDigitalDataFunctionNet, 351
MeaFeedbackFunctionNet, 351
MeFunctionNet, 352
SetDigitalOut, 348

SetNumberOfAnalogChannels, 348
SetNumberOfChannels, 349, 350
SetTriggerMaskValue, 350
SetTriggerPeriod, 351
W2100_FunctionNet, 352
WClassicFunctionNet, 352
CMeaDigitalDataFunctionNet, 352
CMeaDigitalDataFunctionNet, 352
GetDigitalData, 353
SetDigitalData, 353
CMeaFeedbackFunctionNet, 354
CMeaFeedbackFunctionNet, 355
FeedbackGetSampleTimerCount, 355
FeedbackSetAnalogSource, 355
FeedbackSetChannelFilter, 355
FeedbackSetDigitalMapping, 355
FeedbackSetFeedback, 355
FeedbackSetFilterOff, 356
FeedbackSetFilterParameter, 356
FeedbackSetFilterParameter32, 356
FeedbackSetGlobalChannelFilter, 356
FeedbackSetIIRFilterParameter, 356
FeedbackSetLogic, 356
FeedbackSetMkFilter, 356
FeedbackSetNumberOfLogics, 357
FeedbackSetNumberOfRateCounter, 357
FeedbackSetNumberOfRateDetectors, 357
FeedbackSetNumberOfSpikeDetectors, 357
FeedbackSetNumberOfTriggers, 357
FeedbackSetRateCounter, 357
FeedbackSetRateDetector, 357
FeedbackSetSpikeDetectorThreshold, 357
FeedbackSetTrigger, 358
CMeaImpedanceDeviceNet, 358
~CMeaImpedanceDeviceNet, 358
CMeaImpedanceDeviceNet, 358
GetAdapterCode, 359
GetArraySize, 359
GetImpedanceTestFrequency, 359
GetReady, 359
GetResult, 359
SetImpedanceTestFrequency, 359
StartMeasurement, 359
CMeasureTableDeviceNet, 359
CMeasureTableDeviceNet, 360
Sensor, 360
CMeaSwitchDeviceNet, 360
~CMeaSwitchDeviceNet, 361
CMeaSwitchDeviceNet, 361
GetNumber, 361
GetPattern, 361
GetPatternBool, 362
SetPattern, 362
SetPatternBool, 362
CMeaUSBDeviceNet, 362
~CMeaUSBDeviceNet, 363
CMeaUSBDeviceNet, 363
CMeFunctionNet, 363
!CMeFunctionNet, 364
~CMeFunctionNet, 364
CMeFunctionNet, 364
SetTransformer, 364
CMosMea
CCMOSMeaDeviceNet, 110
CmosMea
Mcs::Usb, 65
CMOSMeaBathModeEnumNet
Mcs::Usb, 54
CmosMeaHeadstage
Mcs::Usb, 64
CMOSMeaHeadstage1NCBathCurrentEnumNet
Mcs::Usb, 54
CMOSMeaHeadstage1NCCol2CurrentEnumNet
Mcs::Usb, 54
CMOSMeaHeadstage1NChipTempEnumNet
Mcs::Usb, 55
CMOSMeaHS1SidebandEnumNet
Mcs::Usb, 55
CMOSMeaHS1TriggerStatusEnumNet
Mcs::Usb, 55
CmosmeaIFB2
Mcs::Usb, 65
CMOSMeaIFDigChannelEnumNet
Mcs::Usb, 55
CMOSMeaInterfaceADCEnumNet
Mcs::Usb, 56
CmosMeaInterfaceboard
Mcs::Usb, 64
CMOSMeaPacketFrameContextGroupEnumNet
Mcs::Usb, 56
CMOSMeaSTG1DACSignalEnumNet
Mcs::Usb, 56
CMOSMeaValueUnitEnumNet
Mcs::Usb, 57
CMultiBatteryChargerDeviceNet, 365
!CMultiBatteryChargerDeviceNet, 366
~CMultiBatteryChargerDeviceNet, 366
CapacityTest, 366
ChannelReset, 367
CMultiBatteryChargerDeviceNet, 366
GetBatteryVoltage, 367
GetChannels, 367
GetChannelState, 367
GetChargeCapacity, 368
GetChargeCurrent, 368
GetChargingMode, 368
GetChargingPCoefficient, 368
GetDischargeCapacity, 369
GetDischargeCurrent, 369
GetDischargeCurrentSetPoint, 369
GetFinalDischargeVoltage, 370
GetRatedCapacity, 370
SetChargingMode, 370
SetChargingPCoefficient, 370
SetDischargeCurrentSetPoint, 371
SetFinalDischargeVoltage, 371

SetRatedCapacity, 371
SetRatedCapacityVolatile, 371
CMultiwellCallbackFunctionNet, 372
!CMultiwellCallbackFunctionNet, 373
~CMultiwellCallbackFunctionNet, 373
CMultiwellCallbackFunctionNet, 373
GetPlateClampStateByHeadstage, 373
GetPlateClampStateByHeadstageEvent, 374
OnGetPlateClampStateByHeadstage, 373
CMultiwellDeviceNet, 374
!CMultiwellDeviceNet, 375
~CMultiwellDeviceNet, 375
ClosePlateClamp, 376
CMultiwellDeviceNet, 375
GetPlateClampLockState, 376
GetPlateClampState, 376
GetPlateMux, 376, 377
GetPlateType, 377
GetPowerMuxPlate, 377
IsPlateTypeValid, 378
LockPlateClamp, 378
OpenPlateClamp, 378
SetPlateMux, 378, 379
SetPlateType, 379
SetPowerMuxPlate, 379
StopPlateClamp, 380
UnlockPlateClamp, 380
CMultiwellOptoStimFunctionNet, 380
!CMultiwellOptoStimFunctionNet, 381
~CMultiwellOptoStimFunctionNet, 381
CMultiwellOptoStimFunctionNet, 381
GetAbsMaxCurrentInMicroAmp, 381
GetColorRgb, 381
GetColorStr, 382
GetMaxDurationHighCurrentInMicroSec, 382
GetMaxDutyCycleHighCurrent, 382
GetPermanentCurrentInMicroAmp, 383
GetWaveLengthInNanometer, 383
SetAbsMaxCurrentInMicroAmp, 383
SetColorRgb, 383
SetColorStr, 384
SetMaxDurationHighCurrentInMicroSec, 384
SetMaxDutyCycleHighCurrent, 384
SetPermanentCurrentInMicroAmp, 384
SetWaveLengthInNanometer, 385
CNF_GenDeviceNet, 385
~CNF_GenDeviceNet, 385
CNF_GenDeviceNet, 385
Set_Values, 385
COctoPotDeviceNet, 386
BurnAdcOffset, 387
BurnDacOffset, 387
COctoPotDeviceNet, 386, 387
EnableChecksum, 387
EnableDigitalIn, 387
EnableTimestamp, 387
GetAdcOffset, 387
GetDacOffset, 387
PatternListStart, 387
RampStart, 388
ResetAdcOffset, 388
ResetDacOffset, 388
SetAdcOffset, 388
SetAmplificationSwitch, 388
SetBathclamp, 388
SetChannelSwitch, 388
SetDacAutoControl, 388
SetDacOffset, 388
SetDacValue, 389
SetNumberOfChannels, 389
SetOutputRate, 389
SetPatternListEntry, 389
SetPidParameter, 389
SetRampParameter, 389
SetSineParameter, 389
SineStart, 390
COkuvisionStimulatorDeviceNet, 390
~COkuvisionStimulatorDeviceNet, 391
COkuvisionStimulatorDeviceNet, 391
GetCheckVoltage, 391
GetCurrentFactor, 391
GetDACOffset, 391
GetMaxPower, 391
GetMaxVoltage, 391
GetPulseform, 391
GetRTC, 392
GetStimulatorStatus, 392
GetVoltage, 392
SetCheckVoltage, 392
SetCurrentFactor, 392
SetDACOffset, 392
SetMaxPower, 393
SetMaxVoltage, 393
SetPulseform, 393
SetRTC, 393
Coldstart
CMcsUsbFactoryNet, 291
CommaPositionA
CFilterCoefficientsNet::s_FilterAttributesNet, 692
Mcs::Usb, 63
CommaPositionB
CFilterCoefficientsNet::s_FilterAttributesNet, 692
Mcs::Usb, 63
CompareFirmware
CMcsUsbFactoryNet, 292
CompareTo
HeadStageIDType, 677
CompensateElectrodeOffset
CWarnerUssingFunctionNet, 614
Connect
CMcsUsbNet, 312, 313
CRFFunctionNet, 438
ConnectDevice
CRadioControledDevicesNet, 433
ConnectedImp
CPositionImpDeviceNet, 410

ConnectImp
CPositionImpDeviceNet, 410
ConnectSlave
CGilsonDeviceNet, 164
ControlState
HeadStageIDTypeState, 680
CornerFrequency
CFilterPropertyNet, 135
CornerFrequencymHz
CFilterPropertyNet, 136
Count
CMcsUsbListNet, 307
CPatchServerDeviceNet, 393
CPatchServerDeviceNet, 394
Sensor, 394
CPathIdentDeviceNet, 394
∼CPathIdentDeviceNet, 395
CPathIdentDeviceNet, 395
Measure, 395
Set_Values, 395
CPedoterDeviceNet, 395
!CPedoterDeviceNet, 396
∼CPedoterDeviceNet, 396
CPedoterDeviceNet, 396
GetCommand, 396
SetCommand, 396
CPeristalticPumpDeviceNet, 397
∼CPeristalticPumpDeviceNet, 397
CPeristalticPumpDeviceNet, 397
McsBus_MotorControl, 398
CPgaDeviceNet, 398
∼CPgaDeviceNet, 398
ApplyGains, 399
CPgaDeviceNet, 398
DefineAmplification, 399
DefineFrequencyRange, 399
DefineNumAmplifications, 399
DefineNumFrequencyRanges, 399
GetAmplification, 399
GetFrequencyRange, 399
GetGain, 400
GetNumAmplifications, 400
GetNumFrequencyRanges, 400
SetGain, 400
CPositionIIDeviceNet, 400
!CPositionIIDeviceNet, 402
∼CPositionIIDeviceNet, 402
CPositionIIDeviceNet, 402
GetCoilCommunication, 402
GetDebugData, 402
GetEventData, 403
GetImplantCurrentSetpoint, 403
GetImplantResult, 404
GetImplantState, 404
GetOnOff, 404
GetPowerStrength, 405
GetRTC, 405
GetStateDebugData, 405
GetStateEventData, 406
RFFunction, 409
SetImplantCurrentSetpoint, 406
SetPowerStrength, 406
SetRTC, 406
SetStateDebugData, 408
SetStateEventData, 408
SwitchOnOff, 408
CPositionImpDeviceNet, 409
!CPositionImpDeviceNet, 410
∼CPositionImpDeviceNet, 410
ConnectedImp, 410
ConnectImp, 410
CPositionImpDeviceNet, 410
GetDeviceList, 410
GetImpId, 411
GetRFFrequency, 411
SetDeviceList, 411
SetImpId, 411
SetRFFrequency, 412
CPPCDeviceNet, 412
CPPCDeviceNet, 412
McsBus, 413
McsBus_MotorControl, 413
McsBus_Sensor, 413
PPCFunction, 413
CPPCFunctionNet, 413
!CPPCFunctionNet, 415
∼CPPCFunctionNet, 415
CPPCFunctionNet, 414
FirePressurePulse, 415
GetAnalogVoltage, 415
GetAnalogVoltageRange, 415
GetDigitalIn, 417
GetPressureRange, 417
GetPumpModeType, 417
GetPumpSpeedUnit, 418
GetSupplyVoltage, 418
GetValveActive, 418
IsBusy, 418
LoadPressure, 420
MeasureReservoir, 420
SetAnalogVoltageRange, 420
SetPressureOffset, 420
SetPressureRange, 420
SetPumpModeType, 421
SetPumpSpeedUnit, 421
SetValveActive, 421
CPPS_DeviceNet, 422
CPPS_DeviceNet, 422
McsBus, 422
McsBus_MotorControl, 422
McsBus_Sensor, 422
PPS_Function, 422
CPPS_FunctionNet, 423
CPPS_FunctionNet, 423, 424
GetAnalogVoltage, 424
GetAnalogVoltages, 424

GetBubbleState, 424
GetDigitalIn, 424
GetPumpCouple, 424
GetPumpEnableSpeedRatio, 424
GetPumpFastOnOff, 424
GetPumpFastSpeed, 424
GetPumpFunctionSpeeds, 425
GetPumpManualOnOff, 425
GetPumpMaxSpeed, 425
GetPumpModeType, 425
GetPumpSpeedRatio, 425
GetPumpSpeedUnit, 425
GetSupplyVoltage, 425
GetUseBubble, 425
SetAnalogVoltages, 425
SetPumpCouple, 426
SetPumpEnableSpeedRatio, 426
SetPumpFastOnOff, 426
SetPumpFastSpeed, 426
SetPumpFunctionSpeeds, 426
SetPumpManualOnOff, 426
SetPumpMaxSpeed, 426
SetPumpModeType, 426
SetPumpSpeedRatio, 427
SetPumpSpeedUnit, 427
SetUseBubble, 427
CPPSDeviceNet, 427
CPPSDeviceNet, 428
CProgramPressureCurveNet, 428
!CProgramPressureCurveNet, 429
~CProgramPressureCurveNet, 428
CProgramPressureCurveNet, 428
GetRepeats, 429
Program, 429
SetRepeats, 429
CPulseGeneratorFunctionNet, 429
!CPulseGeneratorFunctionNet, 430
~CPulseGeneratorFunctionNet, 430
CPulseGeneratorFunctionNet, 430
GetModeSelect, 431
GetPeriod, 431
GetPulseLength, 431
SetModeSelect, 432
SetPeriod, 432
SetPulseLength, 432
CRadioControledDevicesNet, 432
ConnectDevice, 433
CRadioControledDevicesNet, 433
DisConnectDevice, 433
GetDeviceNames, 433
GetFrequency, 434
HasRadioControl, 434
SetFrequency, 434
StillConnected, 434
CreateSideband
CStimulusFunctionNet, 558
CreateWirelessHeadstageSerialNumberString
CWirelessBaseFunctionNet, 662
CRegionOfInterestRect
CCMOSMeaDeviceNet::CRegionOfInterestRect, 434
CRetinaLedDeviceNet, 435
~CRetinaLedDeviceNet, 436
AddLoopEntry, 436
AddTableEntry, 436
ClearTable, 436
CRetinaLedDeviceNet, 436
GetTablepointer, 436
SetLED, 436
SetLumi, 436
SetPersistency, 436
SetRepeat, 437
SetTablepointer, 437
SetTrigger, 437
CRFFunctionNet, 437
!CRFFunctionNet, 438
~CRFFunctionNet, 438
Connect, 438
CRFFunctionNet, 438
GetAvailableDeviceList, 439
GetAvailableDeviceListEx, 439
GetAvailableStateList, 439
GetAvailableStateListEx, 439
GetBaseFrequency, 440
GetConnectedDevice, 440
GetState, 440
GetTestMode, 440
GetWorkingFrequency, 441
SetBaseFrequency, 441
SetTestMode, 441
SetWorkingFrequency, 441
CRobo_FYIProgram_FunctionNet, 442
CRobo_FYIProgram_FunctionNet, 442
GetLength, 442
GetState, 443
GetValve1, 443
GetValve2, 443
SetLength, 443
SetValve1, 443
SetValve2, 443
Start, 443
CRobo_FYITemp_FunctionNet, 443
CRobo_FYITemp_FunctionNet, 444
GetICoeff, 444
GetMaxPower, 444
GetPCoeff, 444
GetRegulatorOnOff, 444
GetSollTemp, 444
SetICoeff, 445
SetMaxPower, 445
SetPCoeff, 445
SetRegulatorOnOff, 445
SetSollTemp, 445
CRoboDacqNet, 445
CancelTableLoop, 448
CancelTableLoopAndStopTable, 448

ClampAmpRestart, 448
CRoboDacqNet, 448
DoRamp, 448
Emu_GetCellCapacity, 449
Emu_GetCellPotential, 449
Emu_GetCellResists, 449
Emu_GetElectrodeResists, 449
Emu_GetNoise, 449
Emu_SetCellCapacity, 449
Emu_SetCellPotential, 449
Emu_SetCellResists, 449
Emu_SetElectrodeResists, 449
Emu_SetNoise, 449
GetAllDigout, 450
GetCapacityC, 450
GetCapacityV, 450
GetCapacityX, 450
GetClampAmpSerialNumber, 450
GetCommand, 450
GetConfigurationBit, 450
GetConfigurationBitAxc, 450
GetConfigurationBitBlu_Led, 450
GetConfigurationBitBlu_LedToggleFast, 450
GetConfigurationBitBlu_LedToggleSlow, 450
GetConfigurationBitCC_Gen, 451
GetConfigurationBitCV_Gen, 451
GetConfigurationBitRC_Gen, 451
GetConfigurationBitRed_Led, 451
GetConfigurationBitRed_LedSaturation, 451
GetConfigurationBitRed_LedToggleFast, 451
GetConfigurationBitRed_LedToggleSlow, 451
GetConfigurationBitRelais, 451
GetConfigurationBitRV_Gen, 451
GetConfigurationBits, 451
GetConfigurationBitStream, 451
GetConfigurationBitSupply, 452
GetCrossTalkOffset, 452
GetCrossTalkOptimum, 452
GetDigout, 452
GetDisplayText, 452
GetDownsampleFactor, 452
GetFilter, 452
GetFilterCoeffs, 452
GetIC, 452
GetIClamp, 452
GetICOffset, 452
GetIGain, 453
GetNIC_MS, 453
GetNUC_MS, 453
GetNUV_MS, 453
GetPGain, 453
GetRecordingNumber, 453
GetResistanceC, 453
GetResistanceV, 453
GetScreen, 453
GetSimulation, 453
GetUC, 453
GetUClamp, 454
GetUCOffset, 454
GetUpdateDisplay, 454
GetUV, 454
GetUVOffset, 454
GetXGain, 454
RunTable, 454
SetAllDigout, 454
SetCommand, 454
SetConfigurationBit, 454
SetConfigurationBitAxc, 455
SetConfigurationBitBlu_Led, 455
SetConfigurationBitBlu_LedToggleFast, 455
SetConfigurationBitBlu_LedToggleSlow, 455
SetConfigurationBitCC_Gen, 455
SetConfigurationBitCV_Gen, 455
SetConfigurationBitRC_Gen, 455
SetConfigurationBitRed_Led, 455
SetConfigurationBitRed_LedSaturation, 455
SetConfigurationBitRed_LedToggleFast, 456
SetConfigurationBitRed_LedToggleSlow, 456
SetConfigurationBitRelais, 456
SetConfigurationBitRV_Gen, 456
SetConfigurationBitStream, 456
SetConfigurationBitSupply, 456
SetCrossTalkOffset, 456
SetCrossTalkOptimum, 456
SetDigout, 456
SetDisplayText, 457
SetDownsampleFactor, 457
SetFilter, 457
SetFilterCoeffs, 457
SetIClamp, 457
SetICOffset, 457
SetIGain, 457
SetNoFilterCoeffs, 457
SetPGain, 457
SetRecordingNumber, 458
SetScreen, 458
SetSimulation, 458
SetTriggerMaskValue, 458
SetUClamp, 458
SetUCOffset, 458
SetUVOffset, 458
SetXGain, 458
StopTable, 458, 459
Table_Wait, 459
TableDefBegin, 459
TableDefEnd, 459
TriggerMask_Default, 459
TriggerValue_MoveAbs, 459
TriggerValue_StartQueue, 459
UpdateDisplay, 459
VirtualDevice_ContinousDacq, 459
VirtualDevice_TableRun, 459
CRoboDeviceNet, 460
~CRoboDeviceNet, 463
Axes_I, 471
Axes_X, 471

Axes_Y, 471
Axes_Z, 472
Axis_I, 472
Axis_X, 472
Axis_Y, 472
Axis_Z, 472
CancelPoolLoop, 463
CancelPoolLoopAndStopMovement, 463
CRoboDeviceNet, 463
EnableQueue, 464
FindReference, 464
GetAirpressure, 464
GetAirpressureLimit, 464
GetAirValve, 464
GetCurrentAirvalve, 464
GetCurrentAirvalveLimit, 464
GetCurrentPosition, 465
GetErrorAirpressure, 465
GetErrorCurrentAirvalve, 465
GetErrorVoltage12V, 465
GetErrorVoltage5V, 465
GetErrorVoltageAirvalve, 465
GetErrorVoltageRs485A, 465
GetErrorVoltageRs485B, 465
GetErrorVoltageValves, 466
GetInMovement, 466
GetMinPressure, 466
GetMovementError, 466
GetVoltage12V, 466
GetVoltage12VLimit, 466
GetVoltage5V, 466
GetVoltage5VLimit, 466
GetVoltageAirvalve, 466
GetVoltageAirvalveLimit, 466
GetVoltageRs485A, 467
GetVoltageRs485ALimit, 467
GetVoltageRs485B, 467
GetVoltageRs485BLimit, 467
GetVoltageValves, 467
GetVoltageValvesLimit, 467
IsQueueEnabled, 467
IsQueueStarted, 467
McsBus, 475
McsBus_MotorControl, 475
McsBus_XY, 472
McsBus_ZI, 472
MoveAbs, 467, 468
NullCommand, 469
RoboError_AnotherMaster, 472
RoboError_Base, 473
RoboError_CannotEscapeEndSwitch, 473
RoboError_CommandAlreadyInProgress, 473
RoboError_CommandNotPossible, 473
RoboError_CommunicationTimeout, 473
RoboError_DacqNotReady, 473
RoboError_DLLMovementTimeout, 473
RoboError_FindReferenceMethod, 473
RoboError_GilsonCommandPending, 473
RoboError_GilsonTimeout, 473
RoboError_GilsonWrondID, 474
RoboError_McsBus_UnknownCommand, 474
RoboError_NoEndSwitch, 474
RoboError_NoMoreData, 474
RoboError_NoReference, 474
RoboError_NoSpeedOrAcceleration, 474
RoboError_OverPressure, 474
RoboError_ParameterNotAllowed, 474
RoboError_PeristalticTimeout, 474
RoboError_Phase0OutOfRange, 474
RoboError_PollLoopCanceled, 475
RoboError_PollLoopCanceledAndStopMovement, 475
RoboError_Pressure, 475
RoboError_RangeExceeded, 475
RoboError_StateChangeNotPossible, 475
RoboError_Timeout, 475
RoboError_UnknownCommand, 475
RoboMainLowLevelCommand, 476
RoboStatusEvent, 476
SetAirpressureLimit, 469
SetAirValve, 469
SetCurrentAirvalveLimit, 469
SetCurrentAndAir, 469
SetInMovement, 469
SetMinPressure, 470
SetVoltage12VLimit, 470
SetVoltage5VLimit, 470
SetVoltageAirvalveLimit, 470
SetVoltageRs485ALimit, 470
SetVoltageRs485BLimit, 470
SetVoltageValvesLimit, 470
StartQueue, 470
StopMovement, 471
WaitTimer, 471
CRoboDeviceNet::RoboMainLowLevelCommands, 684
FindReferencePhase0, 685
GetAxisConfig, 685
GetHWConfig, 685
GetHWRevision, 685
GetMaxNoPressure, 685
GetMaxNoPressureWaitTime, 685
GetMaxPressureWaitTime, 685
GetMinNoPressureWaitTime, 685
GetMinPressure, 685
GetMinPressureWaitTime, 686
GetParameter, 686
GetPhases, 686
GetSearchReferenceFastAccel, 686
GetSearchReferenceFastSpeed, 686
GetSearchReferenceFineAccel, 686
GetSearchReferenceFineSpeed, 686
GetSearchReferenceMethod, 687
GetSearchReferenceMoveOut, 687
GetSearchReferenceOffsetPos, 687
GetUserParameter, 687
HasRef, 688

SetAxisConfig, 688
SetHWConfig, 688
SetHWRevision, 688
SetMaxNoPressure, 688
SetMaxNoPressureWaitTime, 688
SetMaxPressureWaitTime, 688
SetMinNoPressureWaitTime, 688
SetMinPressure, 688
SetMinPressureWaitTime, 688
SetParameter, 689
SetSearchReferenceFastAccel, 689
SetSearchReferenceFastSpeed, 689
SetSearchReferenceFineAccel, 689
SetSearchReferenceFineSpeed, 689
SetSearchReferenceMethod, 689
SetSearchReferenceMoveOut, 690
SetSearchReferenceOffsetPos, 690
SetUserParameter, 690
CRoboFluidDeviceNet, 476
~CRoboFluidDeviceNet, 477
CloseAllValves, 477
CRoboFluidDeviceNet, 477
GetPumpSpeed, 477
GetSingleValve, 477
GetValve, 477
IsPumpMotorOn, 478
m_pMcsBus_MotorControlNet, 479
m_pRoboFluidDevice, 479
McsBus_MotorControl, 479
PumpOff, 478
PumpOn, 478
SetPumpSpeed, 478
SetSingleValve, 478
SetValve, 478
CRoboInjectDeviceNet, 479
CRoboInjectDeviceNet, 480
CRoboocyte2DeviceNet, 480
CRoboocyte2DeviceNet, 481
GetAxisLED, 481
GetGilsonDevice, 481
GetMcsBus_Extension, 481
GetRoboDacq, 481
GetRoboFluidDevice, 481
SetAxisLED, 481
CRoboStatorDeviceNet, 481
CRoboStatorDeviceNet, 483
FindReferenceI, 483
FindReferenceXY, 483
FindReferenceZ, 483
GetCurrentPositionI, 483
GetCurrentPositionXY, 483
GetCurrentPositionZ, 484
HasRefI, 484
HasRefXY, 484
HasRefZ, 484
MoveAbsI, 484
MoveAbsXY, 484
MoveAbsZ, 484, 485
RoboMainStatorLowLevelCommand, 487
SetAccelerationI, 485
SetAccelerationNativeI, 485
SetAccelerationNativeXY, 485
SetAccelerationNativeZ, 485
SetAccelerationXY, 485
SetAccelerationZ, 485
SetCurrentAndAirXY, 485
SetSpeedI, 486
SetSpeedNativeI, 486
SetSpeedNativeXY, 486
SetSpeedNativeZ, 486
SetSpeedXY, 486
SetSpeedZ, 486
SetVelocityI, 486
SetVelocityXY, 486
SetVelocityZ, 486
StopMovementI, 487
StopMovementXY, 487
StopMovementZ, 487
CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands, 691
FindReferencePhase0XY, 691
CSafeISDeviceNet, 487
~CSafeISDeviceNet, 488
CSafeISDeviceNet, 488
DacqDevice, 490
FluidControlDevice, 490
RoboDevice, 490
SetAdcChannels, 489
SetAdcSamplePos, 489
SetDacMode, 489
SetDacPeriode, 489
SetDacPulseform, 489
SetSwitches, 490
csCapacityTestDischarge
Mcs::Usb, 67
csCapacityTestPrecharge
Mcs::Usb, 67
csCharge
Mcs::Usb, 67
CSCUDacqGroupChannelSelectionNet, 490
CSCUDacqGroupChannelSelectionNet, 491
CSCUFunctionNet, 491
!CSCUFunctionNet, 494
~CSCUFunctionNet, 494
AutomaticAnalogOut, 494
CSCUFunctionNet, 493
EnableAnalogOut, 494
GetAnalogOutADCRange, 494
GetAnalogOutChannels, 494
GetAnalogOutDACRange, 495
GetAvailableHeadstages, 495
GetAvailableHeadstagesEvent, 505
GetFilterProperties, 495
GetFilterProperty, 496
GetHeadstageAdcBits, 496
GetHeadstageAdcRangeInMicroVolt, 496

GetHeadstageDacBits, 497
GetHeadstageDacCurrentRangeInMicroAmpere, 497
GetHeadstageDacCurrentResolutionInNanoAmpere, 497
GetHeadstageDacVoltageRangeInMilliVolt, 498
GetHeadstageDacVoltageResolutionInMicroVolt, 498
GetHeadstageGainInPermille, 498
GetHeadstageID, 499
GetHeadstageNumberOfAnalogChannels, 499
GetHeadstageNumberOfStimulationChannels, 499
GetHeadstagePowerStateAtStart, 500
GetHeadstageSamplerate, 500
GetHeadstageSerialNumber, 500
GetMaxNumberOfHeadstages, 501
GetMaxStimulusChannelsPerHeadstage, 501
GetReferenceElectrodeMode, 501
GetReferenceElectrodeSwitchState, 501
HasAnalogOut, 502
HasGalvanicIsolation, 502
HasHSPowerSwitch, 502
IsAnalogOutEnabled, 502
IsAutomaticAnalogOut, 502
IsHeadstageAvailable, 502
IsHeadstageAvailableEvent, 506
IsHSPowered, 503
IsInDacqLegacyMode, 503
OnGetAvailableHeadstages, 503
OnIsHeadstageAvailable, 503
PowerHS, 503
SetAnalogOutADCRange, 504
SetAnalogOutChannels, 504
SetAnalogOutDACRange, 504
SetDacqLegacyMode, 504
SetHeadstagePowerStateAtStart, 505
SetReferenceElectrodeMode, 505
SetReferenceElectrodeSwitchState, 505
csDischarge
Mcs::Usb, 67
CSerialPortNet, 506
CSerialPortNet, 506
GetBytesAvailable, 506
Receive, 507
ReceiveString, 507
Send, 507
csError
Mcs::Usb, 67
csIdleChargeFinished
Mcs::Usb, 67
csIdleNoBattery
Mcs::Usb, 67
csRefreshBattery
Mcs::Usb, 67
CStg200xBasicNet, 507
~CStg200xBasicNet, 512
GetAnalogRanges, 512
GetAnalogResolution, 512
GetAutocalibrationDisabled, 512
GetAvailableMemory, 514
GetBlankingEnable, 514
GetCurrentRangeByIndex, 515
GetCurrentRangeInNanoAmp, 515
GetCurrentResolutionInNanoAmp, 515
GetDacAmplificationFactor, 516
GetDACResolution, 516
GetDiginValue, 516
GetDigoutMode, 516
GetDigoutValue, 516
GetElectrodeDacMux, 517
GetElectrodeEnable, 517, 518
GetElectrodeMode, 518, 519
GetEnableAmplifierProtectionSwitch, 519
GetExternalElectrodeEnable, 520
GetFAAmplification, 520
GetHeadstage, 520
GetListmodeIndexRange, 520
GetListmodeTriggerSource, 521
GetNumberOfAnalogChannels, 521
GetNumberOfHWDACPaths, 521
GetNumberOfStimulationElectrodes, 522
GetNumberOfStimulationSourcesPerElectrode, 522
GetNumberOfSyncoutChannels, 522
GetNumberOfTriggerInputs, 522
GetOutputRate, 522
GetStgProgramInfo, 522, 523
GetStgVersionInfo, 523
GetSyncoutMap, 523
GetTotalMemory, 524
GetTriggerSource, 524
GetVoltageRangeInMicroVolt, 524
GetVoltageResolutionInMicroVolt, 524
ListModeSendStart, 525
ListModeSendStop, 525
SendStart, 525
SendStop, 525
SetAutocalibrationDisabled, 526
SetBlankingEnable, 526, 527
SetCurrentMode, 527
SetCurrentRangeByIndex, 528
SetDacAmplificationFactor, 528
SetDigoutMode, 528
SetDigoutValue, 528
SetElectrodeDacMux, 529, 530
SetElectrodeEnable, 531, 532
SetElectrodeMode, 533, 534
SetEnableAmplifierProtectionSwitch, 535, 536
SetExternalElectrodeEnable, 536, 537
SetFAAmplification, 537
SetHeadstage, 537
SetListmodeIndexRange, 537
SetListmodeTriggerSource, 538
SetMeasurementMode, 539
SetOutputRate, 539
SetStgProgramInfo, 539

SetSyncoutMap, 540
SetTriggerSource, 540
SetVoltageMode, 540
CStg200xDownloadBasicNet, 541
ClearChannelData, 542
ClearSyncData, 543
DisableAutoReset, 543
EnableAutoReset, 543
ForceStatusEvent, 543
GetMemoryUsageDAC, 543
GetMemoryUsageSyncout, 543
GetSweepCount, 544
GetTrigger, 544
ResetStatus, 545
SendChannelData, 545
SendSyncData, 545
SetupRetriggerMode, 546
SetupTrigger, 547
SetupTriggerSingle, 547
Stimulus, 548
CStg200xDownloadNet, 548
∼CStg200xDownloadNet, 550
ClearChannel_PrepareAndSendData, 550
CStg200xDownloadNet, 549
DisableMultiFileMode, 551
EnableMultiFileMode, 551
GetModuleCurrent, 551
GetModuleTemp, 551
MwPollStatusEvent, 554
PrepareAndAppendData, 551
PrepareAndSendData, 552
QueryTriggerstatus, 553
SendSegmentDefine, 553
SendSegmentSelect, 553
SendSegmentStart, 554
SetOutputMap, 554
Stg200xPollStatusEvent, 555
CStimulusFunctionNet, 555
ClearChannel_PrepareAndSendData, 557
ClearChannelData, 557
ClearMultiplexedData, 557
ClearSyncData, 557
CreateSideband, 558
CStimulusFunctionNet, 556, 557
ForceStatusEvent, 558
GetAvailableMemory, 559
GetCurrentRangeInNanoAmp, 559
GetCurrentResolutionInNanoAmp, 559
GetDACResolution, 559
GetMultiplexedDataChannelsInBlock, 560
GetNumberOfAnalogChannels, 560
GetTotalMemory, 560
GetVoltageRangeInMicroVolt, 560
GetVoltageResolutionInMicroVolt, 561
PollStatusEvent, 566
PrepareAndAppendData, 561
PrepareAndSendData, 562
PrepareData, 563
SendMultiplexedData, 563
SendPreparedData, 563
SendStart, 563
SendStop, 564
SetupTrigger, 564
SetupTriggerSingle, 565
StartPoll, 565
StopPoll, 565
CStimulusFunctionNet::SidebandData, 693
!SidebandData, 693
∼SidebandData, 693
Duration, 693
Sideband, 693
SidebandData, 693
CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, 694
!StimulusDeviceDataAndUnrolledData, 695
∼StimulusDeviceDataAndUnrolledData, 695
DeviceData, 695
DeviceDataLength, 695
StimulusDeviceDataAndUnrolledData, 695
UnrolledAmplitude, 695
UnrolledDuration, 695
UnrolledSync, 695
CSw2to64DeviceNet, 566
∼CSw2to64DeviceNet, 567
CSw2to64DeviceNet, 567
GetChannel, 567
GetChannels, 567
GetNumber, 567
SetChannel, 567
SetChannels, 568
CTcxDeviceNet, 568
∼CTcxDeviceNet, 571
CalibrateThermocouple, 571
CTcxDeviceNet, 570
FactoryReset, 571
GetBoardTemp, 571
GetCalibration, 571
GetCalibrationDecp, 571
GetCalibrationMax, 571
GetCalibrationMin, 571
GetCurrent, 571
GetD, 572
GetDDecp, 572
GetDevice, 572
GetDeviceType, 572
GetDevname, 572
GetDMax, 572
GetDMin, 572
GetDuty, 572
GetEnableHeaterLimit, 573
GetEnableThermocouple, 573
GetHasThermocouple, 573
GetHeaterLimit, 573
GetHeaterTemp, 573
GetI, 573
GetIDecp, 573

GetIMax, 574
GetIMin, 574
GetIOut, 574
GetMaxHeaterPowerMultiwell, 574
GetMaxP, 574
GetMaxpDecp, 574
GetMaxpMax, 574
GetMaxpMin, 575
GetNumControlChannels, 575
GetNumDevices, 575
GetNumMeasureChannels, 575
GetOnOff, 575
GetP, 575
GetPDecp, 575
GetPMax, 575
GetPMin, 575
GetPOut, 575
GetPwrOut, 576
GetPwrSet, 576
GetRes1, 576
GetRes2, 576
GetResS, 576
GetResX, 576
GetROut, 576
GetSensorType, 577
GetSetpoint, 577
GetSetpointDecp, 577
GetSetpointMax, 577
GetSetpointMin, 577
GetThermocoupleCalibration, 577
GetThermocoupleNanovoltPerKelvin, 577
GetThermocoupleReferenceTemp, 578
GetThermocoupleTemp, 578
GetThermocoupleTempAbs, 578
GetUnit, 578
GetUOut, 578
GetValue, 578
GetValueHires, 579
GetVolti, 579
SetCalibration, 579
SetD, 579
SetDevice, 579
SetDeviceType, 579
SetDevname, 579
SetEnableHeaterLimit, 579
SetEnableThermocouple, 580
SetHeaterLimit, 580
SetI, 580
SetMaxHeaterPowerMultiwell, 580
SetMaxP, 580
SetOnOff, 580
SetP, 581
SetSensorType, 581
SetSetpoint, 581
SetThermocoupleNanovoltPerKelvin, 581
CTEERFunctionNet, 581
!CTEERFunctionNet, 584
~CTEERFunctionNet, 584
CancelInternalCalibration, 584
CTEERFunctionNet, 583
GetAdapterCode, 584
GetAdcOffsetU1, 584
GetAdcOffsetU2, 584
GetAmplitude_nA, 584
GetBytesPerSample, 585
GetClampMode, 585
GetControllerParams, 585
GetCurrentEnable, 585
GetDacZero, 586
GetFrameErrorCounter, 586
GetLiquidResistance, 586
GetMaxChunkSize_Byte, 586
GetNumberOfAvailableSamples, 586
GetPeriod_us, 587
GetRotaryPositionCode, 587
GetSampleBufferChunk, 587
GetSampleRate, 587
GetSampleVoltageBuffer_uV, 588
GetScaleFactorU1, 588
GetScaleFactorU2, 588
GetUptimeSeconds, 588
GetWaveform, 588
IsInternalCalibrationFinished, 589
IsSamplingFinished, 589
SetAmplitude_nA, 589
SetBufferIndex, 589
SetClampMode, 590
SetControllerParams, 590
SetCurrentEnable, 590
SetExternalLED, 590
SetLiquidResistance, 591
SetPeriod_us, 591
SetWaveform, 591
StartInternalCalibration, 591
StartSampling, 591
StopSampling, 592
CTEERMachineDeviceNet, 592
~CTEERMachineDeviceNet, 592
CTEERMachineDeviceNet, 592
TEERFunctionNet, 593
CurrentClamp
Mcs::Usb, 87
CurrentMeasure
Mcs::Usb, 54
CurrentRangeInNanoAmp
W2100_StimulusParametersNet, 697
CurrentResolutionInNanoAmp
W2100_StimulusParametersNet, 697
CUsbDeviceConfigurationFunctionNet, 593
!CUsbDeviceConfigurationFunctionNet, 594
~CUsbDeviceConfigurationFunctionNet, 594
CUsbDeviceConfigurationFunctionNet, 593
SetDeviceId, 594
SetDeviceName, 594
CUsbExceptionNet, 594
CUsbExceptionNet, 595

Status, 595
CutoffFrequency
CCreateFilterNet, 112
CVoltageRangeInfoNet
CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet, 596
CW2100_FunctionNet, 596
ClearStimulusParametersCache, 598
ClearUserDefinedNameCache, 598
CW2100_FunctionNet, 598
DeselectAllHeadstages, 599
DeselectHeadstage, 599
GetAccelGyroCurrentRate, 599
GetAccelGyroDesiredRate, 599
GetAccelGyroEnabled, 599
GetAccelRange, 599
GetAnalogOutChannel, 599
GetAnalogOutFilter, 599
GetAudioChannels, 599
GetAvailableHeadstages, 599
GetBatteryState, 600
GetDacRange, 600
GetFilterProperties, 600
GetFilterProperty, 600
GetFPGAFirmwareType, 600
GetGyroRange, 600
GetHeadstageOnOff, 600
GetHeadstageSamplingActive, 600
GetMultiHeadstageMode, 600
GetPicFirmwareType, 600
GetSelectedChannels, 601
GetSelectedHeadstageState, 601
GetStimulusParametersCache, 601
GetStimulusParametersFromSelectedHS, 601
GetStiumlusParameters, 601
GetUserDefinedName, 601
GetUserDefinedNameCache, 601
GetUserDefinedNameFromSelectedHS, 602
PulseGenerator, 603
SelectHeadstage, 602
SetAccelGyroDesiredRate, 602
SetAccelGyroEnabled, 602
SetAccelRange, 602
SetAnalogOutChannel, 602
SetAnalogOutFilter, 602
SetAudioChannels, 602
SetDacRange, 603
SetGyroRange, 603
SetHeadstageOnOff, 603
SetHeadstageSamplingActive, 603
SetHeadstageToSleep, 603
SetMultiHeadstageMode, 603
SetSelectedChannels, 603
Stimulator, 604
CW2100_FunctionNet::AudioChannelsNet, 92
amplification, 92
channel, 92
dacqgroup, 92
CW2100_StimulatorFunctionNet, 604
BOOST_BIT, 610
ClearChannelData, 605
CW2100_StimulatorFunctionNet, 605
GetBoostAlwaysOnMode, 606
GetBoostPreTime, 606
GetCurrentRangeInNanoAmp, 606
GetCurrentResolutionInNanoAmp, 606
GetDACResolution, 606
GetDigitalStimulatorTrigger, 606
GetDigitalStimulatorTriggerSlope, 607
GetNumberOfAnalogChannels, 607
GetNumberOfSyncoutChannels, 607
GetNumberOfTriggerInputs, 607
GetStimulationPatternMemory, 607
GetTimeResolutionInNanoSeconds, 607
GetTimeSlot, 607
GetVoltageRangeInMicroVolt, 607
GetVoltageResolutionInMicroVolt, 608
GND_SWITCH_BIT, 610
PollStatusEvent, 610
PrepareData, 608
PrepareDataSync, 608
SelectTimeSlot, 608
SendPreparedData, 608
SendStart, 609
SendStop, 609
SetDigitalStimulatorTrigger, 609
SetDigitalStimulatorTriggerSlope, 609
StartPoll, 609
StopPoll, 609
SYNC_BIT0, 610
SYNC_BIT1, 610
CW2100DacqGroupChannelSelectionNet, 610
CW2100DacqGroupChannelSelectionNet, 610
CWarnerUssingDeviceNet, 611
!CWarnerUssingDeviceNet, 612
~CWarnerUssingDeviceNet, 611
CWarnerUssingDeviceNet, 611
WarnerUssingFunction, 612
CWarnerUssingFunctionNet, 612
!CWarnerUssingFunctionNet, 614
~CWarnerUssingFunctionNet, 614
CompensateElectrodeOffset, 614
CWarnerUssingFunctionNet, 614
GetAvailableChambers, 615
GetChannelsCountOfChamber, 615
GetClampMode, 615
GetComplianceVoltageThreshold, 615
GetDacPampsPerDigitHighCurrentRange, 616
GetDacPampsPerDigitLowCurrentRange, 616
GetDacZero, 616
GetHighCurrentRange, 617
GetIdleModeOffset, 617
GetLiquidResistance, 617
GetLowCurrentRange, 618
GetNumberOfAvailableChambers, 618
GetNumberOfHardwareSlotsForChambers, 618

GetU1Offset, 618
GetU1Reference, 620
GetU2Offset, 620
GetU2Reference, 620
GetUnitDescription, 621
GetUnitExponent, 621
GetUnitName, 621
GetUnitsPerDigit, 622
GetUptimeSeconds, 622
GetVoltageClampControllerParam_D, 622
GetVoltageClampControllerParam_I, 623
GetVoltageClampControllerParam_P, 623
IsChamberAvailable, 623
IsHighCurrentMode, 624
IsInternalCalibrationFinished, 624
IsPulseEnabled, 624
SetClampMode, 625
SetEnablePulse, 625
SetHighCurrentMode, 625
SetIdleModeOffset, 626
SetLiquidResistance, 626
SetLowCurrentMode, 626
SetPulse, 626
SetVoltageClampControllerParam_D, 627
SetVoltageClampControllerParam_I, 627
SetVoltageClampControllerParam_P, 627
WaitForAllChambers, 628
WaitForChamber, 628
CWarnerValveControllerDeviceNet, 628
!CWarnerValveControllerDeviceNet, 633
~CWarnerValveControllerDeviceNet, 633
ClearTableName, 633
ClearValveTable, 633
CWarnerValveControllerDeviceNet, 633
GetActiveRunningTableNumber, 633
GetActiveRunningTableNumberEvent, 650
GetAnalogThresholdHigh, 633
GetAnalogThresholdHighEvent, 650
GetAnalogThresholdLow, 634
GetAnalogThresholdLowEvent, 650
GetAnalogVoltage, 634
GetAnalogVoltageEvent, 650
GetCurrentEditTableNumber, 634
GetCurrentNumberOfValves, 634
GetCurrentNumberOfValvesEvent, 651
GetDigitalOutPortValve, 635
GetDigitalOutPortValveEvent, 651
GetDigitalPortDirection, 635
GetDigitalPortDirectionEvent, 651
GetDisplayMode, 635
GetDisplayModeEvent, 651
GetTableName, 635
GetTableNamebyIndex, 636
GetTableNamebyIndexEvent, 651
GetTotalNumberOfDigitalPorts, 636
GetTotalNumberOfTables, 636
GetTotalNumberOfValves, 636
GetTotalTableSize, 637
GetValveActive, 637
GetValveActiveEvent, 651
GetValveBoardRevision, 637
GetValveBoardRevisionEvent, 651
GetValveBoardRevisionString, 637
GetValveCurrent, 637
GetValveDigitalInPort, 638
GetValveDigitalInPortEvent, 652
GetValveLedOn, 638
GetValveLedOnEvent, 652
GetValveManualGroup, 638
GetValveManualGroupEvent, 652
GetValveManualState, 639
GetValveManualStateEvent, 652
GetValveMode, 639
GetValveModeEvent, 652
GetValvesActiveMap, 639
GetValvesManualStateMap, 639
GetValveTableEntry, 640
IsDigitalOutPortInverted, 640
IsDigitalOutPortInvertedEvent, 652
IsValveDigitalInInverted, 640
IsValveDigitalInInvertedEvent, 652
IsValveOpen, 641
IsValveOpenEvent, 653
IsValveOpenInAnalogMode, 641
IsValveOpenInAnalogModeEvent, 653
IsValveOpenInDigitalMode, 641
IsValveOpenInDigitalModeEvent, 653
LoadValveTable, 641
OnGetActiveRunningTableNumber, 642
OnGetAnalogThresholdHigh, 642
OnGetAnalogThresholdLow, 642
OnGetAnalogVoltage, 642
OnGetCurrentNumberOfValves, 642
OnGetDigitalOutPortValve, 642
OnGetDigitalPortDirection, 642
OnGetDisplayMode, 642
OnGetTableNamebyIndex, 642
OnGetValveActive, 643
OnGetValveBoardRevision, 643
OnGetValveDigitalInPort, 643
OnGetValveLedOn, 643
OnGetValveManualGroup, 643
OnGetValveManualState, 643
OnGetValveMode, 643
OnIsDigitalOutPortInverted, 643
OnIsValveDigitalInInverted, 644
OnIsValveOpen, 644
OnIsValveOpenInAnalogMode, 644
OnIsValveOpenInDigitalMode, 644
OnTableEntryChanged, 644
SetActiveRunningTableNumber, 644
SetAnalogThresholdHigh, 644
SetAnalogThresholdLow, 645
SetCurrentEditTableNumber, 645
SetDefault, 645
SetDigitalOutPortInvert, 645

SetDigitalOutPortValve, 646
SetDigitalPortDirection, 646
SetDisplayMode, 646
SetTableName, 646
SetTableStep, 647
SetTableStepAll, 647
SetValveActive, 647
SetValveCurrent, 647
SetValveDigitalInInvert, 648
SetValveDigitalInPort, 648
SetValveLedOn, 648
SetValveManualGroup, 648
SetValveManualState, 649
SetValveMode, 649
SetValvesActiveMap, 649
SetValvesManualStateMap, 649
SetValveTableEntry, 649
StoreValveTable, 650
TableEntryChangedEvent, 653
CWarnerValveControllerDeviceTesterFunctionNet, 653
!CWarnerValveControllerDeviceTesterFunctionNet, 655
~CWarnerValveControllerDeviceTesterFunctionNet, 654
CWarnerValveControllerDeviceTesterFunctionNet, 654
GetIO, 655
GetSync, 655
SetADC, 655
SetIO, 655
SetIODirection, 656
SetTrigger, 656
SetTriggerSyncDirection, 656
CWClassicFunctionNet, 656
CWClassicFunctionNet, 657, 658
GetFilterParametersHeadstage, 658
GetHasChecksum, 658
GetHasRedLedHeadstage, 658
GetHeadstageOnOff, 658
GetResetFilter, 658
GetRFConnectionStatus, 658
GetRFFrequencyHeadstage, 658
GetRFFrequencyReceiver, 658
GetRFPower, 659
GetScanHeadstagesResult, 659
GetSelectedHeadstage, 659
GetSerialNumberHeadstage, 659
GetWPADebugMode, 659
GetWPAType, 659
ResetChannelmap, 659
ScanForHeadstages, 659
SetChannelmap, 659
SetFilterParametersHeadstage, 659
SetHasChecksum, 660
SetHeadstageOnOff, 660
SetHWSelectedChannels, 660
SetResetFilter, 660
SetRFFrequencyHeadstage, 660
SetRFFrequencyReceiver, 660
SetRFFrequencyReceiverEeprom, 660
SetRFLostBehaviour, 660
SetRFPower, 661
SetSelectedHeadstage, 661
SetSerialNumberHeadstage, 661
SetWPADebugMode, 661
SetWPAType, 661
CWirelessBaseFunctionNet, 661
CreateWirelessHeadstageSerialNumberString, 662
CWirelessBaseFunctionNet, 662
CyclePort
CMcsUsbNet, 314
Cypress
Mcs::Usb, 87
Cypress_FX1
Mcs::Usb, 73
Cypress_FX2
Mcs::Usb, 73
Cypress_FX3
Mcs::Usb, 73

DAC1Channel
Mcs::Usb, 56
DAC2Channel
Mcs::Usb, 56
DAC3Channel
Mcs::Usb, 56
DAC4Channel
Mcs::Usb, 56
DACQ1DigitalGroup
Mcs::Usb, 57
DacqDevice
CSafeISDeviceNet, 490
dacqgroup
CW2100_FunctionNet::AudioChannelsNet, 92
DacqGroupChannelEnumNet
Mcs::Usb, 57
DacqMeaGroupTypeEnumNet
Mcs::Usb, 57
DacqTrigger
Mcs::Usb, 62
DACResolution
W2100_StimulusParametersNet, 697
DataModeEnumNet
Mcs::Usb, 58
DataState
HeadStageIDTypeState, 680
DeepCopy
CCMOSMeaDeviceNet::CRegionOfInterestRect, 434
DefineAmplification
CPgaDeviceNet, 399
DefineFrequencyRange
CPgaDeviceNet, 399
DefineNumAmplifications
CPgaDeviceNet, 399
DefineNumFrequencyRanges

CPgaDeviceNet, 399
DeselectAllHeadstages
CW2100_FunctionNet, 599
DeselectHeadstage
CW2100_FunctionNet, 599
DEST_FX3_TARGET_MASK
Mcs::Usb, 54
DEST_TARGET1
Mcs::Usb, 53
DEST_TARGET10
Mcs::Usb, 53
DEST_TARGET11
Mcs::Usb, 53
DEST_TARGET12
Mcs::Usb, 53
DEST_TARGET13
Mcs::Usb, 54
DEST_TARGET14
Mcs::Usb, 54
DEST_TARGET15
Mcs::Usb, 54
DEST_TARGET2
Mcs::Usb, 53
DEST_TARGET3
Mcs::Usb, 53
DEST_TARGET4
Mcs::Usb, 53
DEST_TARGET5
Mcs::Usb, 53
DEST_TARGET6
Mcs::Usb, 53
DEST_TARGET7
Mcs::Usb, 53
DEST_TARGET8
Mcs::Usb, 53
DEST_TARGET9
Mcs::Usb, 53
DEST_TARGET_MASK
Mcs::Usb, 54
DetectChipType
CCMOSMea_FunctionNet, 99
DEVICE_NOT_FOUND
Mcs::Usb, 63
DeviceArrival
CMcsUsbListNet, 307
DeviceData
CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, 695
DeviceDataLength
CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, 695
DeviceEnumNet
Mcs::Usb, 58
DeviceHasNoHeadstage
Mcs::Usb, 65, 78
DeviceId
CMcsUsbListEntryNet, 304
DeviceIdNet, 662
BcdDevice, 663
BusType, 663
DeviceIdNet, 663
IdProduct, 663
IdVendor, 663
operator=, 663
DeviceName
CMcsUsbListEntryNet, 304
DeviceNotConnected
Mcs::Usb, 64, 65, 78
DeviceRemoval
CMcsUsbListNet, 307
DeviceRunStatus
Mcs::Usb, 61, 70, 79, 83, 89
DigDataFromReceiver
Mcs::Usb, 89
Digital
Mcs::Usb, 73, 90
DigitalData
Mcs::Usb, 61, 70, 79, 83, 89
DigitalDatastreamEnableEnumNet
Mcs::Usb, 59
DigitalGroup
Mcs::Usb, 58
DigitalIn
Mcs::Usb, 59, 60, 70, 79, 82, 89
DigitalInOfOutPort
Mcs::Usb, 60, 70, 79, 82, 89
DigitalInPort
Mcs::Usb, 55
DigitalInReserverd
Mcs::Usb, 59
DigitalMux
Mcs::Usb, 55
DigitalOut
Mcs::Usb, 59
DigitalOutReg
Mcs::Usb, 55
DigitalOutReserved
Mcs::Usb, 60
DigitalOutStimulator
Mcs::Usb, 61, 70, 79, 83, 89
DigitalPulse
Mcs::Usb, 60, 70, 79, 82, 89
DigitalReg
Mcs::Usb, 56
DigitalSource
DigitalSource< digitalsourceenum >, 664
DigitalSource< digitalsourceenum >, 664
DigitalSource, 664
MaxBitNumber, 664
MaxBitNumberStatic, 664
size, 664
Source, 665
DigitalSourceEnumNet
Mcs::Usb, 60
DigitalSourceGeneral, 665
DigitalSourceGeneral, 665

MaxBitNumber, 665, 666
size, 666
Source, 666
DigitalStimulatorTriggerEventEnumNet
Mcs::Usb, 61
DigitalStimulatorTriggerSlopeEnumNet
Mcs::Usb, 61
DigitalTargetEnumNet
Mcs::Usb, 61
Digout
Mcs::Usb, 62
DigOutStim
Mcs::Usb, 60
DigOutStimulatorStartTrigger
Mcs::Usb, 62
DigOutStimulatorStopTrigger
Mcs::Usb, 62
Digstream
Mcs::Usb, 62
DigStreamFromReceiver
Mcs::Usb, 89
DigStreamToReceiver
Mcs::Usb, 62
Dilutor
Mcs::Usb, 76
DisableAutoReset
CStg200xDownloadBasicNet, 543
DisableMultiFileMode
CStg200xDownloadNet, 551
Disconnect
CMcsUsbNet, 314
DisConnectDevice
CRadioControledDevicesNet, 433
DongleS
Mcs::Usb, 74
DoRamp
CRoboDacqNet, 448
Dotriapot
Mcs::Usb, 74
DoubleToInt
Mcs::Usb, 63
DownloadFirmware
CMcsUsbFactoryNet, 292
DownloadOnly
Mcs::Usb, 81
DriverVersionNet, 666
~DriverVersionNet, 667
DriverVersionNet, 667
DriverVersionNet::FormatVersion, 667
GetDestinationCode, 668
GetDestinationName, 669
GetMajor, 669
GetMinor, 670
GetNumEntries, 670
GetSerialNumber, 670
GetStatus, 671
GetVersionInt, 671
GetVersionString, 672
DriverVersionNet::FormatVersion
DriverVersionNet, 667
DSP
FirmwareDestinationNames, 673
Mcs::Usb, 51
DSPAnalogGroup
Mcs::Usb, 69, 79
DSPDataGroup
Mcs::Usb, 57, 88
DSPDigitalGroup
Mcs::Usb, 69, 79
DummyCommand
CLIH3DeviceNet, 190
Duration
CStimulusFunctionNet::SidebandData, 693

eCube
Mcs::Usb, 75, 78
eCubeHeadstage
Mcs::Usb, 65
ElectricalStimulation
HeadStageIDType, 676
ElectrodeDacMuxEnumNet
Mcs::Usb, 62
ElectrodeModeEnumNet
Mcs::Usb, 62
ElectrodeOffset
Mcs::Usb, 87
emAutomatic
Mcs::Usb, 62
emManual
Mcs::Usb, 62
EmptyKey
CMcsUsbNet, 314
Emu_GetCellCapacity
CRoboDacqNet, 449
Emu_GetCellPotential
CRoboDacqNet, 449
Emu_GetCellResists
CRoboDacqNet, 449
Emu_GetElectrodeResists
CRoboDacqNet, 449
Emu_GetNoise
CRoboDacqNet, 449
Emu_SetCellCapacity
CRoboDacqNet, 449
Emu_SetCellPotential
CRoboDacqNet, 449
Emu_SetCellResists
CRoboDacqNet, 449
Emu_SetElectrodeResists
CRoboDacqNet, 449
Emu_SetNoise
CRoboDacqNet, 449
EnableAnalogOut
CSCUFunctionNet, 494
EnableAutoReset
CStg200xDownloadBasicNet, 543
EnableChannelsInGroup

CCMOSMea_FunctionNet, 99
CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet >, 116
EnableChecksum
CMeaDeviceNet, 346
COctoPotDeviceNet, 387
EnableDigitalIn
CMeaDeviceNet, 346, 347
COctoPotDeviceNet, 387
EnableExceptions
CMcsUsbNet, 314
EnableMultiFileMode
CStg200xDownloadNet, 551
EnableQueue
CRoboDeviceNet, 464
EnableTimestamp
CMeaDeviceNet, 347
COctoPotDeviceNet, 387
EnableUserTrigger
CLIH3DeviceNet, 190
Encapsulator
Mcs::Usb, 76
enCMosMeaChipType
Mcs::Usb, 62
EnSTG200x_STATUS
Mcs::Usb, 63
Entry
HeadStageIDType, 677
EOFAndCRC
Mcs::Usb, 56
Equals
CMcsUsbListEntryNet, 301
HeadStageIDType, 677
HeadstageIDTypeObject, 679
EraseEepromRegisterPreconfig
CMcsUsbNet, 314
EraseFilterParameterPermanent
CFilterConfigurationNet, 132
CFilterConfigurationRegisterNet, 133, 134
ErasePermanentAdcOffset
CLIH3DeviceNet, 191
ErasePermanentDacOffset
CLIH3DeviceNet, 191
Error_Callback_Aquisition_Stopped
CMcsUsbDacqNet, 286
Error_Callback_Data_lost
CMcsUsbDacqNet, 286
Error_Callback_Frames_Lost
CMcsUsbDacqNet, 286
Error_Callback_Packet_Error
CMcsUsbDacqNet, 286
Error_Callback_Queue_Full
CMcsUsbDacqNet, 287
Error_Callback_RingQueue_Full
CMcsUsbDacqNet, 287
ErrorEvent
CMcsUsbDacqNet, 287
ExternBCTester
Mcs::Usb, 74
ExternDTester
Mcs::Usb, 74
ExternSTester
Mcs::Usb, 74

FactoryReset
CTcxDeviceNet, 571
Falling
Mcs::Usb, 61
FCB
Mcs::Usb, 74
FCX
Mcs::Usb, 74
Feedback
Mcs::Usb, 60, 70, 79, 82, 89
FeedbackGetSampleTimerCount
CMeaFeedbackFunctionNet, 355
FeedbackHigh
Mcs::Usb, 60
FeedbackLow
Mcs::Usb, 60
FeedbackReg
Mcs::Usb, 56
FeedbackSetAnalogSource
CMeaFeedbackFunctionNet, 355
FeedbackSetChannelFilter
CMeaFeedbackFunctionNet, 355
FeedbackSetDigitalMapping
CMeaFeedbackFunctionNet, 355
FeedbackSetFeedback
CMeaFeedbackFunctionNet, 355
FeedbackSetFilterOff
CMeaFeedbackFunctionNet, 356
FeedbackSetFilterParameter
CMeaFeedbackFunctionNet, 356
FeedbackSetFilterParameter32
CMeaFeedbackFunctionNet, 356
FeedbackSetGlobalChannelFilter
CMeaFeedbackFunctionNet, 356
FeedbackSetIIRFilterParameter
CMeaFeedbackFunctionNet, 356
FeedbackSetLogic
CMeaFeedbackFunctionNet, 356
FeedbackSetMkFilter
CMeaFeedbackFunctionNet, 356
FeedbackSetNumberOfLogics
CMeaFeedbackFunctionNet, 357
FeedbackSetNumberOfRateCounter
CMeaFeedbackFunctionNet, 357
FeedbackSetNumberOfRateDetectors
CMeaFeedbackFunctionNet, 357
FeedbackSetNumberOfSpikeDetectors
CMeaFeedbackFunctionNet, 357
FeedbackSetNumberOfTriggers
CMeaFeedbackFunctionNet, 357
FeedbackSetRateCounter

CMeaFeedbackFunctionNet, 357
FeedbackSetRateDetector
CMeaFeedbackFunctionNet, 357
FeedbackSetSpikeDetectorThreshold
CMeaFeedbackFunctionNet, 357
FeedbackSetTrigger
CMeaFeedbackFunctionNet, 358
FilterActive
CFilterPropertyNet, 136
FilterAttributeEnumNet
Mcs::Usb, 63
FilterBand
CFilterPropertyNet, 136
FilterBandEnumNet
Mcs::Usb, 63
FilterCalculationDirectionEnumNet
Mcs::Usb, 63
FilterFamily
CFilterPropertyNet, 136
FilterFamilyEnumNet
Mcs::Usb, 63
FilterType
CFilterPropertyNet, 136
FilterTypeEnumNet
Mcs::Usb, 64
FindFilter
CCreateFilterNet, 111
FindFirmwareVersionMagicInBuffer
CMcsUsbFactoryNet, 292
FindReference
CRoboDeviceNet, 464
FindReferenceI
CRoboStatorDeviceNet, 483
FindReferencePhase0
CRoboDeviceNet::RoboMainLowLevelCommands, 685
FindReferencePhase0XY
CRoboStatorDeviceNet::RoboMainStatorLowLevelCommands, 691
FindReferenceXY
CRoboStatorDeviceNet, 483
FindReferenceZ
CRoboStatorDeviceNet, 483
Finished
Mcs::Usb, 81
FirePressurePulse
CPPCFunctionNet, 415
FirmwareDestinationNames, 672
Altera, 673
Bootstrap, 673
BUS1_MCSBUS1, 673
BUS1_MCSBUS2, 673
DSP, 673
FPGA2, 673
FPGA3, 673
FPGA4, 673
FPGA5, 673
FPGA6, 673
MCSBUS1, 674
MCSBUS10, 674
MCSBUS11, 674
MCSBUS12, 674
MCSBUS13, 674
MCSBUS2, 674
MCSBUS3, 674
MCSBUS4, 674
MCSBUS5, 674
MCSBUS6, 674
MCSBUS7, 674
MCSBUS8, 675
MCSBUS9, 675
MCU1, 675
PIC, 675
PIC2, 675
PIC3, 675
PIC4, 675
USB, 675
FluidControlDevice
CSafeISDeviceNet, 490
ForceStatusEvent
CStg200xDownloadBasicNet, 543
CStimulusFunctionNet, 558
FPGA10
Mcs::Usb, 52
FPGA10_BASE
Mcs::Usb, 53
FPGA10_GOLD
Mcs::Usb, 53
FPGA11
Mcs::Usb, 52
FPGA11_BASE
Mcs::Usb, 53
FPGA11_GOLD
Mcs::Usb, 53
FPGA12
Mcs::Usb, 52
FPGA12_BASE
Mcs::Usb, 53
FPGA12_GOLD
Mcs::Usb, 53
FPGA13
Mcs::Usb, 52
FPGA13_BASE
Mcs::Usb, 53
FPGA13_GOLD
Mcs::Usb, 53
FPGA14
Mcs::Usb, 52
FPGA14_BASE
Mcs::Usb, 53
FPGA14_GOLD
Mcs::Usb, 53
FPGA15
Mcs::Usb, 52
FPGA15_BASE
Mcs::Usb, 53

FPGA15_GOLD
Mcs::Usb, 53
FPGA16
Mcs::Usb, 53
FPGA16_BASE
Mcs::Usb, 53
FPGA16_GOLD
Mcs::Usb, 53
FPGA2
FirmwareDestinationNames, 673
Mcs::Usb, 52
FPGA2_BASE
Mcs::Usb, 53
FPGA2_GOLD
Mcs::Usb, 53
FPGA3
FirmwareDestinationNames, 673
Mcs::Usb, 52
FPGA3_BASE
Mcs::Usb, 53
FPGA3_GOLD
Mcs::Usb, 53
FPGA4
FirmwareDestinationNames, 673
Mcs::Usb, 52
FPGA4_BASE
Mcs::Usb, 53
FPGA4_GOLD
Mcs::Usb, 53
FPGA5
FirmwareDestinationNames, 673
Mcs::Usb, 52
FPGA5_BASE
Mcs::Usb, 53
FPGA5_GOLD
Mcs::Usb, 53
FPGA6
FirmwareDestinationNames, 673
Mcs::Usb, 52
FPGA6_BASE
Mcs::Usb, 53
FPGA6_GOLD
Mcs::Usb, 53
FPGA7
Mcs::Usb, 52
FPGA7_BASE
Mcs::Usb, 53
FPGA7_GOLD
Mcs::Usb, 53
FPGA8
Mcs::Usb, 52
FPGA8_BASE
Mcs::Usb, 53
FPGA8_GOLD
Mcs::Usb, 53
FPGA9
Mcs::Usb, 52
FPGA9_BASE
Mcs::Usb, 53
FPGA9_GOLD
Mcs::Usb, 53
FPGA_BASE
Mcs::Usb, 53
FPGA_BOOTSTRAP
Mcs::Usb, 53
FPGA_GOLD
Mcs::Usb, 53
FPGA_NORMAL
Mcs::Usb, 51
FpgaIdEnumNet
Mcs::Usb, 64
FrameContextGroup
Mcs::Usb, 58
FromIntPtr
StgStatusNet, 694
FromPtr
StgStatusNet, 694
FullCharge
Mcs::Usb, 67
FullSpeed
Mcs::Usb, 69
FunkDongleS
Mcs::Usb, 74
FX3MCSDataAddress
CMcsUsbFactoryNet, 297
FX3MCSDataDeviceIdOffset
CMcsUsbFactoryNet, 297
FX3MCSDataIFB1ImageOffset
CMcsUsbFactoryNet, 297
FX3MCSDataIFB2ImageOffset
CMcsUsbFactoryNet, 297
FX3MCSDataVersionOffset
CMcsUsbFactoryNet, 297
FYIProgram
CFYIDeviceNet, 144
FYITemp
CFYIDeviceNet, 144

Gain
CMeaDeviceNet, 351
Gated_High_Active
Mcs::Usb, 76
Gated_Low_Active
Mcs::Usb, 76
GE2100
Mcs::Usb, 75
Generic
Mcs::Usb, 74
Get2AnalogInput
CMcsBus_SensorNet, 223
Get2DigitalInput
CMcsBus_SensorNet, 223
Get4ADC
CMcsBus_SensorNet, 223
Get4ADCAverage
CMcsBus_SensorNet, 223
Get4ADCCatchampAverageShift

CMcsBus_SensorNet, 223
Get4ADCMode
CMcsBus_SensorNet, 223
Get4DAC
CMcsBus_SensorNet, 223
GetAbsMaxCurrentInMicroAmp
CMultiwellOptoStimFunctionNet, 381
GetAccelGyroCurrentRate
CW2100_FunctionNet, 599
GetAccelGyroDesiredRate
CW2100_FunctionNet, 599
GetAccelGyroEnabled
CW2100_FunctionNet, 599
GetAccelRange
CW2100_FunctionNet, 599
GetActiveRunningTableNumber
CWarnerValveControllerDeviceNet, 633
GetActiveRunningTableNumberEvent
CWarnerValveControllerDeviceNet, 650
GetAdapterCode
CMeaImpedanceDeviceNet, 359
CTEERFunctionNet, 584
GetAdapterType
CMcsUsbDacqNet, 262
GetAdc
CFluidControlDeviceNet, 138
GetAdcDataFormat
CMcsUsbDacqNet, 262
GetADCInputOffset
CCMOSMea_FunctionNet, 99
GetAdcOffset
CLIH3DeviceNet, 191
COctoPotDeviceNet, 387
GetAdcOffsetU1
CTEERFunctionNet, 584
GetAdcOffsetU2
CTEERFunctionNet, 584
GetADCs
CMcsBus_SensorNet, 223
GetADCsLoop
CMcsBus_SensorNet, 224
GetAdcZero
CMcsUsbDacqNet, 262
GetAirpressure
CRoboDeviceNet, 464
GetAirpressureLimit
CRoboDeviceNet, 464
GetAirValve
CRoboDeviceNet, 464
GetAllDigout
CRoboDacqNet, 450
GetAmplification
CPgaDeviceNet, 399
GetAmplitude_nA
CTEERFunctionNet, 584
GetAnalogGain
CMeaDeviceNet, 348
GetAnalogOutADCRange
CSCUFunctionNet, 494
GetAnalogOutChannel
CW2100_FunctionNet, 599
GetAnalogOutChannels
CSCUFunctionNet, 494
GetAnalogOutDACRange
CSCUFunctionNet, 495
GetAnalogOutFilter
CW2100_FunctionNet, 599
GetAnalogRanges
CStg200xBasicNet, 512
GetAnalogResolution
CStg200xBasicNet, 512
GetAnalogThresholdHigh
CWarnerValveControllerDeviceNet, 633
GetAnalogThresholdHighEvent
CWarnerValveControllerDeviceNet, 650
GetAnalogThresholdLow
CWarnerValveControllerDeviceNet, 634
GetAnalogThresholdLowEvent
CWarnerValveControllerDeviceNet, 650
GetAnalogValueUnit
CMcsUsbDacqNet, 263
GetAnalogVoltage
CPPCFunctionNet, 415
CPPS_FunctionNet, 424
CWarnerValveControllerDeviceNet, 634
GetAnalogVoltageEvent
CWarnerValveControllerDeviceNet, 650
GetAnalogVoltageRange
CPPCFunctionNet, 415
GetAnalogVoltages
CPPS_FunctionNet, 424
GetArraySize
CMeaImpedanceDeviceNet, 359
GetAudioChannels
CMeaAudioFunctionNet, 332, 333
CW2100_FunctionNet, 599
GetAudioOutDacParameter
CLIH3DeviceNet, 191
GetAutocalibrationDisabled
CStg200xBasicNet, 512
GetAvailableBaseSamplerates
CCMOSMeaDeviceNet, 108
GetAvailableChambers
CWarnerUssingFunctionNet, 615
GetAvailableDeviceList
CRFFunctionNet, 439
GetAvailableDeviceListEx
CRFFunctionNet, 439
GetAvailableHeadstages
CSCUFunctionNet, 495
CW2100_FunctionNet, 599
GetAvailableHeadstagesEvent
CSCUFunctionNet, 505
GetAvailableMemory
CStg200xBasicNet, 514
CStimulusFunctionNet, 559

GetAvailableSampleRates
CMcsUsbDacqNet::CHWInfo, 180
GetAvailableStateList
CRFFunctionNet, 439
GetAvailableStateListEx
CRFFunctionNet, 439
GetAvailableVoltageRangesInMicroVolt
CMcsUsbDacqNet::CHWInfo, 180
GetAvailableVoltageRangesInMicroVoltAndStringsInMilliVolt
CMcsUsbDacqNet::CHWInfo, 180
GetAxisConfig
CRoboDeviceNet::RoboMainLowLevelCommands, 685
GetAxisLED
CRoboocyte2DeviceNet, 481
GetAxisParametersSignedEeprom
CMcsBus_AxisParametersNet, 198
GetAxisParametersUnsignedEeprom
CMcsBus_AxisParametersNet, 198
GetBaseFrequency
CRFFunctionNet, 440
GetBaseSamplerate
CCMOSMeaDeviceNet, 109
GetBath
CCMOSMea_FunctionNet, 99
GetBathMode
CCMOSMea_FunctionNet, 99
GetBatteryState
CW2100_FunctionNet, 600
GetBatteryVoltage
CMultiBatteryChargerDeviceNet, 367
GetBiQuad
CCreateFilterNet, 112
GetBiQuads
CCreateFilterNet, 112
GetBlankingEnable
CStg200xBasicNet, 514
GetBoardTemp
CTcxDeviceNet, 571
GetBoostAlwaysOnMode
CW2100_StimulatorFunctionNet, 606
GetBoostPreTime
CW2100_StimulatorFunctionNet, 606
GetBubbleState
CPPS_FunctionNet, 424
GetBubbleStatus
CMcsBus_SensorNet, 224
GetBuffer
CGenericDevelopDeviceNet, 152
GetBusAddress
CMcsBusNet, 237
GetBusAddressEeprom
CMcsBusNet, 237
GetByteBuffer
CGenericDevelopDeviceNet, 152
GetBytesAvailable
CSerialPortNet, 506
GetBytesPerSample
CTEERFunctionNet, 585
GetCalibration
CTcxDeviceNet, 571
GetCalibrationDecp
CTcxDeviceNet, 571
GetCalibrationMax
CTcxDeviceNet, 571
GetCalibrationMin
CTcxDeviceNet, 571
GetCapacityC
CRoboDacqNet, 450
GetCapacityV
CRoboDacqNet, 450
GetCapacityX
CRoboDacqNet, 450
GetCardinalDacqSamplerate
CInterfaceboardFunctionNet, 187
GetCardinalStgOutputrate
CInterfaceboardFunctionNet, 187
GetChannel
CSw2to64DeviceNet, 567
GetChannelDataFillSize
CMcsUsbDacqNet, 263
GetChannelDataI16
CCMOSMeaDeviceNet, 109
GetChannelDataI32
CCMOSMeaDeviceNet, 109
GetChannelDataUI16
CCMOSMeaDeviceNet, 109
GetChannelDataUI32
CCMOSMeaDeviceNet, 109
GetChannelLayout
CMcsUsbDacqNet, 263
GetChannels
CMultiBatteryChargerDeviceNet, 367
CSw2to64DeviceNet, 567
GetChannelsCountOfChamber
CWarnerUssingFunctionNet, 615
GetChannelsInBlock
CMcsUsbDacqNet, 263
GetChannelState
CMultiBatteryChargerDeviceNet, 367
GetChargeCapacity
CMultiBatteryChargerDeviceNet, 368
GetChargeCurrent
CMultiBatteryChargerDeviceNet, 368
GetChargingMode
CMultiBatteryChargerDeviceNet, 368
GetChargingPCoefficient
CMultiBatteryChargerDeviceNet, 368
GetChecksumFromFX3Image
CMcsUsbFactoryNet, 292
GetCheckVoltage
COkuvisionStimulatorDeviceNet, 391
GetClampAmpSerialNumber
CRoboDacqNet, 450
GetClampMode
CTEERFunctionNet, 585

CWarnerUssingFunctionNet, 615
GetCMOSDataDictionary
CCMOSMeaDeviceNet, 109
GetCoilCommunication
CPositionIIDeviceNet, 402
GetColorRgb
CMultiwellOptoStimFunctionNet, 381
GetColorStr
CMultiwellOptoStimFunctionNet, 382
GetCommand
CMcsBusNet, 237, 238
CPedoterDeviceNet, 396
CRoboDacqNet, 450
GetComplianceVoltageThreshold
CWarnerUssingFunctionNet, 615
GetConfiguration
CMcsUsbNet, 315
GetConfigurationBit
CRoboDacqNet, 450
GetConfigurationBitAxc
CRoboDacqNet, 450
GetConfigurationBitBlu_Led
CRoboDacqNet, 450
GetConfigurationBitBlu_LedToggleFast
CRoboDacqNet, 450
GetConfigurationBitBlu_LedToggleSlow
CRoboDacqNet, 450
GetConfigurationBitCC_Gen
CRoboDacqNet, 451
GetConfigurationBitCV_Gen
CRoboDacqNet, 451
GetConfigurationBitRC_Gen
CRoboDacqNet, 451
GetConfigurationBitRed_Led
CRoboDacqNet, 451
GetConfigurationBitRed_LedSaturation
CRoboDacqNet, 451
GetConfigurationBitRed_LedToggleFast
CRoboDacqNet, 451
GetConfigurationBitRed_LedToggleSlow
CRoboDacqNet, 451
GetConfigurationBitRelais
CRoboDacqNet, 451
GetConfigurationBitRV_Gen
CRoboDacqNet, 451
GetConfigurationBits
CRoboDacqNet, 451
GetConfigurationBitStream
CRoboDacqNet, 451
GetConfigurationBitSupply
CRoboDacqNet, 452
GetConnectedDevice
CRFFunctionNet, 440
GetControllerParams
CTEERFunctionNet, 585
GetCrossTalkOffset
CRoboDacqNet, 452
GetCrossTalkOptimum
CRoboDacqNet, 452
GetCur2VolResistance
CGrapheneFunctionNet, 168
GetCurrent
CTcxDeviceNet, 571
GetCurrentAirvalve
CRoboDeviceNet, 464
GetCurrentAirvalveLimit
CRoboDeviceNet, 464
GetCurrentCycle
CMeaCoatDeviceNet, 340
GetCurrentEditTableNumber
CWarnerValveControllerDeviceNet, 634
GetCurrentEnable
CTEERFunctionNet, 585
GetCurrentFactor
COkuvisionStimulatorDeviceNet, 391
GetCurrentNumberOfValves
CWarnerValveControllerDeviceNet, 634
GetCurrentNumberOfValvesEvent
CWarnerValveControllerDeviceNet, 651
GetCurrentPosition
CRoboDeviceNet, 465
GetCurrentPositionI
CRoboStatorDeviceNet, 483
GetCurrentPositionXY
CRoboStatorDeviceNet, 483
GetCurrentPositionZ
CRoboStatorDeviceNet, 484
GetCurrentRangeByIndex
CStg200xBasicNet, 515
GetCurrentRangeInNanoAmp
CStg200xBasicNet, 515
CStimulusFunctionNet, 559
CW2100_StimulatorFunctionNet, 606
GetCurrentResolutionInNanoAmp
CStg200xBasicNet, 515
CStimulusFunctionNet, 559
CW2100_StimulatorFunctionNet, 606
GetCycle
CMeaCleanDeviceNet, 336
GetCycles
CMeaCleanDeviceNet, 336
CMeaCoatDeviceNet, 340
GetD
CTcxDeviceNet, 572
GetDacAmplificationFactor
CStg200xBasicNet, 516
GetDacIdleValue
CLIH3DeviceNet, 192
GetDACOffset
CGrapheneFunctionNet, 168, 169
COkuvisionStimulatorDeviceNet, 391
GetDacOffset
CDacCalibrationFunctionNet, 114
CLIH3DeviceNet, 192
COctoPotDeviceNet, 387
GetDacPampsPerDigitHighCurrentRange

CWarnerUssingFunctionNet, 616
GetDacPampsPerDigitLowCurrentRange
CWarnerUssingFunctionNet, 616
GetDacqRunStatus
CLIH3DeviceNet, 192
GetDacRange
CW2100_FunctionNet, 600
GetDACResolution
CStg200xBasicNet, 516
CStimulusFunctionNet, 559
CW2100_StimulatorFunctionNet, 606
GetDACs
CMcsBus_SensorNet, 224
GetDacUseIdleValue
CLIH3DeviceNet, 192
GetDacZero
CTEERFunctionNet, 586
CWarnerUssingFunctionNet, 616
GetDataFormat
CMcsUsbDacqNet, 263
GetDataMode
CMcsUsbDacqNet, 263
GetDDecp
CTcxDeviceNet, 572
GetDebugData
CPositionIIDeviceNet, 402
GetDestination
CMcsUsbFactoryNet, 292
GetDestinationCode
DriverVersionNet, 668
GetDestinationDisplayLabel
CMcsUsbFactoryNet, 292
GetDestinationName
CMcsUsbFactoryNet, 292, 293
DriverVersionNet, 669
GetDestinationSerialNumber
CMcsUsbFactoryNet, 293
GetDestinationTargetAddress
CMcsUsbFactoryNet, 293
GetDetectionThreshold
CMcsBus_SensorNet, 224
GetDetectorValue
CMcsBus_SensorNet, 224
GetDevice
CTcxDeviceNet, 572
GetDeviceCannotStallOutRequests
CMcsUsbNet, 315
GetDeviceCapableSpeed
CMcsUsbNet, 315
GetDeviceEnum
CMcsUsbNet, 315
GetDeviceGroupChannelInfos
CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet >, 116, 117
GetDeviceId
CMcsUsbNet, 315
GetDeviceList
CPositionImpDeviceNet, 410
GetDeviceNames
CRadioControledDevicesNet, 433
GetDeviceRootHubVendorEnum
CMcsUsbNet, 315
GetDeviceRootHubVendorID
CMcsUsbNet, 315
GetDeviceRootHubVendorName
CMcsUsbNet, 315
GetDeviceSpeed
CMcsUsbNet, 316
GetDeviceType
CTcxDeviceNet, 572
GetDevname
CTcxDeviceNet, 572
GetDigin
CFluidControlDeviceNet, 138
GetDigInState
CLIH3DeviceNet, 193
GetDiginValue
CStg200xBasicNet, 516
GetDigitalData
CMeaDigitalDataFunctionNet, 353
GetDigitalIn
CPPCFunctionNet, 417
CPPS_FunctionNet, 424
GetDigitalOutPortValve
CWarnerValveControllerDeviceNet, 635
GetDigitalOutPortValveEvent
CWarnerValveControllerDeviceNet, 651
GetDigitalPortDirection
CWarnerValveControllerDeviceNet, 635
GetDigitalPortDirectionEvent
CWarnerValveControllerDeviceNet, 651
GetDigitalSource
CMcsUsbDacqNet, 264, 265
GetDigitalStimulatorTrigger
CW2100_StimulatorFunctionNet, 606
GetDigitalStimulatorTriggerSlope
CW2100_StimulatorFunctionNet, 607
GetDigout
CFluidControlDeviceNet, 139
CRoboDacqNet, 452
GetDigoutMode
CStg200xBasicNet, 516
GetDigoutValue
CStg200xBasicNet, 516
GetDIO
CMcsBus_FYIExtensionNet, 201
GetDischargeCapacity
CMultiBatteryChargerDeviceNet, 369
GetDischargeCurrent
CMultiBatteryChargerDeviceNet, 369
GetDischargeCurrentSetPoint
CMultiBatteryChargerDeviceNet, 369
GetDisplayMode
CWarnerValveControllerDeviceNet, 635

GetDisplayModeEvent
CWarnerValveControllerDeviceNet, 651
GetDisplayText
CRoboDacqNet, 452
GetDMax
CTcxDeviceNet, 572
GetDMin
CTcxDeviceNet, 572
GetDownsampleFactor
CRoboDacqNet, 452
GetDSPHighPassByIndex
CIntanMea_FunctionNet, 183
GetDuration
CMeaCoatDeviceNet, 340
GetDuty
CTcxDeviceNet, 572
GetEEpromPage
CLIH3DeviceNet, 193
GetElectrodeDacMux
CStg200xBasicNet, 517
GetElectrodeEnable
CStg200xBasicNet, 517, 518
GetElectrodeMode
CStg200xBasicNet, 518, 519
GetEnableAmplifierProtectionSwitch
CStg200xBasicNet, 519
GetEnabledChannelsInGroup
CCMOSMea_FunctionNet, 100
CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet >, 117
GetEnableHeaterLimit
CTcxDeviceNet, 573
GetEnableThermocouple
CTcxDeviceNet, 573
GetEntry
CMcsUsbListEntryNet, 302
GetEntryCount
CMcsUsbListEntryNet, 303
GetEnumerationSpeed
CMeaDeviceNet, 348
GetErrorAirpressure
CRoboDeviceNet, 465
GetErrorCurrentAirvalve
CRoboDeviceNet, 465
GetErrorMessage
CMcsUsbDacqNet, 266
GetErrorText
CMcsUsbNet, 316
GetErrorVoltage12V
CRoboDeviceNet, 465
GetErrorVoltage5V
CRoboDeviceNet, 465
GetErrorVoltageAirvalve
CRoboDeviceNet, 465
GetErrorVoltageRs485A
CRoboDeviceNet, 465
GetErrorVoltageRs485B
CRoboDeviceNet, 465
GetErrorVoltageValves
CRoboDeviceNet, 466
GetEventData
CPositionIIDeviceNet, 403
GetExternalElectrodeEnable
CStg200xBasicNet, 520
GetFAAmplification
CStg200xBasicNet, 520
GetFilter
CRoboDacqNet, 452
GetFilterAttributes
CFilterConfigurationNet, 132
GetFilterCoeffs
CRoboDacqNet, 452
GetFilterParametersHeadstage
CWClassicFunctionNet, 658
GetFilterProperties
CSCUFunctionNet, 495
CW2100_FunctionNet, 600
GetFilterProperty
CMcsUsbDacqNet, 266
CSCUFunctionNet, 496
CW2100_FunctionNet, 600
GetFinalDischargeVoltage
CMultiBatteryChargerDeviceNet, 370
GetFirmwareVersion
CMcsUsbNet, 316
GetFirmwareVersionFromFile
CMcsUsbFactoryNet, 293
GetFirmwareVersionFromHexFile
CMcsUsbFactoryNet, 293
GetFPGAFirmwareType
CW2100_FunctionNet, 600
GetFrameErrorCounter
CTEERFunctionNet, 586
GetFrequency
CRadioControledDevicesNet, 434
GetFrequencyRange
CPgaDeviceNet, 399
GetGain
CMeaDeviceNet, 348
CPgaDeviceNet, 400
GetGate
CCMOSMea_FunctionNet, 100
GetGilsonDevice
CRoboocyte2DeviceNet, 481
GetGlobalRepeat
CDigOutStimulatorFunctionNet, 123
GetGNDI
CCMOSMea_FunctionNet, 100
GetGroupADCBits
CCMOSMea_FunctionNet, 100
GetGroupChannelBitmaskBySelect
CCMOSMea_FunctionNet, 100
GetGroupChannelBitmaskHS1NCBathCurrent
CCMOSMea_FunctionNet, 100, 101

GetGroupChannelBitmaskHS1NCCol2Current
CCMOSMea_FunctionNet, 101
GetGroupChannelBitmaskHS1NChipTemp
CCMOSMea_FunctionNet, 101
GetGroupChannelBitmaskHS1Sidebands
CCMOSMea_FunctionNet, 101
GetGroupChannelBitmaskHS1TriggerStatus
CCMOSMea_FunctionNet, 101, 102
GetGroupChannelBitmaskIFDigChannels
CCMOSMea_FunctionNet, 102
GetGroupChannelBitmaskInterfaceADC
CCMOSMea_FunctionNet, 102
GetGroupChannelBitmaskPacketFrameContext
CCMOSMea_FunctionNet, 102
GetGroupChannelBitmaskSTG1DACSignal
CCMOSMea_FunctionNet, 102, 103
GetGroupChannelDataI16
CMcsUsbDacqNet, 266
GetGroupChannelDataI32
CMcsUsbDacqNet, 266
GetGroupChannelDataUI16
CMcsUsbDacqNet, 267
GetGroupChannelDataUI32
CMcsUsbDacqNet, 267
GetGroupDCOffset
CCMOSMea_FunctionNet, 103
GetGroupID
CCMOSMea_FunctionNet, 103
CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet >, 117
GetGroupNumberOfChannels
CCMOSMea_FunctionNet, 103
CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet >, 117
GetGroupResolutionPerDigit
CCMOSMea_FunctionNet, 103
GetGroupSampleSize
CCMOSMea_FunctionNet, 104
CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet >, 117
GetGroupType
CCMOSMea_FunctionNet, 104
CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet >, 118
GetGroupUnit
CCMOSMea_FunctionNet, 104
GetGyroRange
CW2100_FunctionNet, 600
GetHardwareMaxRange
CMcsUsbDacqNet, 268
GetHardwareMinRange
CMcsUsbDacqNet, 268
GetHardwareRevision
CMcsUsbNet, 316
GetHasChecksum
CWClassicFunctionNet, 658
GetHashCode
HeadstageIDTypeObject, 679
GetHasRedLedHeadstage
CWClassicFunctionNet, 658
GetHasThermocouple
CTcxDeviceNet, 573
GetHeadstage
CStg200xBasicNet, 520
GetHeadstageActive
CMcsUsbNet, 317
GetHeadstageAdcBits
CSCUFunctionNet, 496
GetHeadstageAdcRangeInMicroVolt
CSCUFunctionNet, 496
GetHeadstageDacBits
CSCUFunctionNet, 497
GetHeadstageDacCurrentRangeInMicroAmpere
CSCUFunctionNet, 497
GetHeadstageDacCurrentResolutionInNanoAmpere
CSCUFunctionNet, 497
GetHeadstageDacVoltageRangeInMilliVolt
CSCUFunctionNet, 498
GetHeadstageDacVoltageResolutionInMicroVolt
CSCUFunctionNet, 498
GetHeadstageGainInPermille
CSCUFunctionNet, 498
GetHeadstageID
CMcsUsbNet, 317
CSCUFunctionNet, 499
GetHeadstageNumberOfAnalogChannels
CSCUFunctionNet, 499
GetHeadstageNumberOfStimulationChannels
CSCUFunctionNet, 499
GetHeadstageOnOff
CW2100_FunctionNet, 600
CWClassicFunctionNet, 658
GetHeadstagePowerStateAtStart
CSCUFunctionNet, 500
GetHeadstagePresent
CMcsUsbNet, 317
GetHeadstageSamplerate
CSCUFunctionNet, 500
GetHeadstageSamplingActive
CW2100_FunctionNet, 600
GetHeadstageSerialNumber
CSCUFunctionNet, 500
GetHeaterLimit
CTcxDeviceNet, 573
GetHeaterTemp
CTcxDeviceNet, 573
GetHighCurrentRange
CWarnerUssingFunctionNet, 617

GetHighpassFilterEnable
CFilterConfigurationNet, 132
GetHWConfig
CRoboDeviceNet::RoboMainLowLevelCommands, 685
GetHWRevision
CRoboDeviceNet::RoboMainLowLevelCommands, 685
GetHWRevisionEeprom
CMcsBusNet, 238
GetI
CTcxDeviceNet, 573
GetIC
CRoboDacqNet, 452
GetIClamp
CRoboDacqNet, 452
GetICoeff
CRobo_FYITemp_FunctionNet, 444
GetICOffset
CRoboDacqNet, 452
GetIDecp
CTcxDeviceNet, 573
GetIdent
CMcsUsbNet, 317
GetIdleModeOffset
CWarnerUssingFunctionNet, 617
GetIGain
CRoboDacqNet, 453
GetIMax
CTcxDeviceNet, 574
GetIMin
CTcxDeviceNet, 574
GetImpedanceResult
CIntanMea_FunctionNet, 183
GetImpedanceTestFrequency
CMeaImpedanceDeviceNet, 359
GetImpId
CPositionImpDeviceNet, 411
GetImplantCurrentSetpoint
CPositionIIDeviceNet, 403
GetImplantResult
CPositionIIDeviceNet, 404
GetImplantState
CPositionIIDeviceNet, 404
GetInMovement
CRoboDeviceNet, 466
GetIntanRegister
CIntanMea_FunctionNet, 183
GetIntBuffer
CGenericDevelopDeviceNet, 153
GetIO
CWarnerValveControllerDeviceTesterFunctionNet, 655
GetIOut
CTcxDeviceNet, 574
GetIoVoltage
CInterfaceboard2FunctionNet, 185
GetLastAnswer
CGilsonDeviceNet, 165
GetLastUSBError
CMcsUsbNet, 318
GetLatency
CMcsBus_SensorNet, 224
GetLatencyCounter
CMcsBus_SensorNet, 224
GetLayoutConfiguration
CMEA2100x256FunctionNet, 331
GetLEDSwitch
CMcsBus_ExtensionNet, 200
GetLength
CRobo_FYIProgram_FunctionNet, 442
GetLiquidResistance
CTEERFunctionNet, 586
CWarnerUssingFunctionNet, 617
GetListmodeIndexRange
CStg200xBasicNet, 520
GetListmodeTriggerSource
CStg200xBasicNet, 521
GetLowCurrentRange
CWarnerUssingFunctionNet, 618
GetLowerFrequencyByIndex
CIntanMea_FunctionNet, 183
GetMajor
DriverVersionNet, 669
GetMaxChunkSize_Byte
CTEERFunctionNet, 586
GetMaxCurrent
CMeaCoatDeviceNet, 340
GetMaxDurationHighCurrentInMicroSec
CMultiwellOptoStimFunctionNet, 382
GetMaxDutyCycleHighCurrent
CMultiwellOptoStimFunctionNet, 382
GetMaxHeaterPowerMultiwell
CTcxDeviceNet, 574
GetMaxNoPressure
CRoboDeviceNet::RoboMainLowLevelCommands, 685
GetMaxNoPressureWaitTime
CRoboDeviceNet::RoboMainLowLevelCommands, 685
GetMaxNumberOfHeadstages
CSCUFunctionNet, 501
GetMaxNumOfColumns
CCMOSMea_FunctionNet, 104
GetMaxP
CTcxDeviceNet, 574
GetMaxpDecp
CTcxDeviceNet, 574
GetMaxpMax
CTcxDeviceNet, 574
GetMaxpMin
CTcxDeviceNet, 575
GetMaxPower
COkuvisionStimulatorDeviceNet, 391
CRobo_FYITemp_FunctionNet, 444
GetMaxPressureWaitTime

CRoboDeviceNet::RoboMainLowLevelCommands, 685
GetMaxReadableColumns
CCMOSMeaDeviceNet, 109
GetMaxSamplingFrequency
CMcsUsbDacqNet, 268
GetMaxStimulusChannelsPerHeadstage
CSCUFunctionNet, 501
GetMaxVoltage
CMeaCleanDeviceNet, 336
COkuvisionStimulatorDeviceNet, 391
GetMCAcceleration
CMcsBus_MotorControlNet, 205
GetMCAccelerationEeprom
CMcsBus_MotorControlNet, 205
GetMCAccelerationShortCommand
CMcsBus_MotorControlNet, 205
GetMCAxisRevisionEeprom
CMcsBus_MotorControlNet, 205
GetMCBreakCurrent
CMcsBus_MotorControlNet, 205
GetMCBreakCurrentEeprom
CMcsBus_MotorControlNet, 206
GetMCConfig
CMcsBus_MotorControlNet, 206
GetMCConfigEeprom
CMcsBus_MotorControlNet, 206
GetMCCurrent
CMcsBus_MotorControlNet, 206
GetMCCurrentEeprom
CMcsBus_MotorControlNet, 206
GetMCCurrentMode
CMcsBus_MotorControlNet, 206
GetMCCurrentModeEeprom
CMcsBus_MotorControlNet, 206
GetMCCurrentModeShortCommand
CMcsBus_MotorControlNet, 207
GetMCCurrentPosition
CMcsBus_MotorControlNet, 207
GetMCCurrentShortCommand
CMcsBus_MotorControlNet, 207
GetMCCurrentSpeed
CMcsBus_MotorControlNet, 207
GetMCMaxAcceleration
CMcsBus_MotorControlNet, 207
GetMCMaxAccelerationEeprom
CMcsBus_MotorControlNet, 207
GetMCMaxCurrent
CMcsBus_MotorControlNet, 207
GetMCMaxCurrentEeprom
CMcsBus_MotorControlNet, 208
GetMCMaxSpeed
CMcsBus_MotorControlNet, 208
GetMCMaxSpeedEeprom
CMcsBus_MotorControlNet, 208
GetMCMaxTravel
CMcsBus_MotorControlNet, 208
GetMCMaxTravelEeprom
CMcsBus_MotorControlNet, 208
GetMCMaxTravelShortCommand
CMcsBus_MotorControlNet, 208
GetMCMovement
CMcsBus_MotorControlNet, 208
GetMCNewPosition
CMcsBus_MotorControlNet, 209
GetMCOutputOnOff
CMcsBus_MotorControlNet, 209
GetMCPhase
CMcsBus_MotorControlNet, 209
GetMCPhaseOffset
CMcsBus_MotorControlNet, 209
GetMCReference
CMcsBus_MotorControlNet, 209
GetMCReferenceCurrent
CMcsBus_MotorControlNet, 209
GetMCReferenceCurrentEeprom
CMcsBus_MotorControlNet, 209
GetMCRegulatorGain
CMcsBus_MotorControlNet, 210
GetMCRegulatorGainEeprom
CMcsBus_MotorControlNet, 210
GetMcsBus_Extension
CRoboocyte2DeviceNet, 481
GetMCScalingFactor
CMcsBus_MotorControlNet, 210
GetMCScalingFactorEeprom
CMcsBus_MotorControlNet, 210
GetMCSpeed
CMcsBus_MotorControlNet, 210
GetMCSpeedEeprom
CMcsBus_MotorControlNet, 210
GetMCSpeedShortCommand
CMcsBus_MotorControlNet, 210
GetMCSpeedUnitEeprom
CMcsBus_MotorControlNet, 211
GetMCStandbyCurrent
CMcsBus_MotorControlNet, 211
GetMCStandbyCurrentEeprom
CMcsBus_MotorControlNet, 211
GetMCStandbyTime
CMcsBus_MotorControlNet, 211
GetMCStandbyTimeEeprom
CMcsBus_MotorControlNet, 211
GetMea21UsbPort
CMcsUsbNet, 318
GetMeaLayout
CMcsUsbDacqNet, 268
GetMemoryUsageDAC
CStg200xDownloadBasicNet, 543
GetMemoryUsageSyncout
CStg200xDownloadBasicNet, 543
GetMinimalThreshold
CMcsBus_SensorNet, 224
GetMinNoPressureWaitTime
CRoboDeviceNet::RoboMainLowLevelCommands, 685

GetMinor
DriverVersionNet, 670
GetMinPressure
CRoboDeviceNet, 466
CRoboDeviceNet::RoboMainLowLevelCommands, 685
GetMinPressureWaitTime
CRoboDeviceNet::RoboMainLowLevelCommands, 686
GetMinSamplingFrequencyStepsize
CMcsUsbDacqNet, 269
GetMinVoltage
CMeaCleanDeviceNet, 336
GetModeSelect
CPulseGeneratorFunctionNet, 431
GetModuleCurrent
CStg200xDownloadNet, 551
GetModuleTemp
CStg200xDownloadNet, 551
GetMovementError
CRoboDeviceNet, 466
GetMovePump
CMcsBus_SensorNet, 225
GetMultiHeadstageMode
CW2100_FunctionNet, 600
GetMultiplexedDataChannelsInBlock
CStimulusFunctionNet, 560
GetNanoVoltsPerKelvin
CMcsBus_TempSensorNet, 230
GetNeurochipMemoryData
CCMOSMea_FunctionNet, 104
GetNeurochipMemorySize
CCMOSMea_FunctionNet, 105
GetNIC_MS
CRoboDacqNet, 453
GetNUC_MS
CRoboDacqNet, 453
GetNumAmplifications
CPgaDeviceNet, 400
GetNumber
CMeaSwitchDeviceNet, 361
CSw2to64DeviceNet, 567
GetNumberOfAnalogChannels
CStg200xBasicNet, 521
CStimulusFunctionNet, 560
CW2100_StimulatorFunctionNet, 607
GetNumberOfAudioChannels
CMeaAudioFunctionNet, 333
GetNumberOfAvailableChambers
CWarnerUssingFunctionNet, 618
GetNumberOfAvailableSamples
CTEERFunctionNet, 586
GetNumberOfChannels
CDigOutStimulatorFunctionNet, 124
GetNumberOfDataBits
CMcsUsbDacqNet, 269
GetNumberOfDevices
CMcsUsbListNet, 306
GetNumberOfHardwareSlotsForChambers
CWarnerUssingFunctionNet, 618
GetNumberOfHWADCChannels
CMcsUsbDacqNet::CHWInfo, 181
GetNumberOfHWDACPaths
CStg200xBasicNet, 521
GetNumberOfHWDigitalChannels
CMcsUsbDacqNet::CHWInfo, 181
GetNumberOfStimulationElectrodes
CStg200xBasicNet, 522
GetNumberOfStimulationSourcesPerElectrode
CStg200xBasicNet, 522
GetNumberOfSupportedGroups
CCMOSMea_FunctionNet, 105
CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet >, 118
GetNumberOfSyncoutChannels
CStg200xBasicNet, 522
CW2100_StimulatorFunctionNet, 607
GetNumberOfTriggerInputs
CStg200xBasicNet, 522
CW2100_StimulatorFunctionNet, 607
GetNumConfigurations
CMcsUsbNet, 318
GetNumControlChannels
CTcxDeviceNet, 575
GetNumDestinations
CMcsUsbFactoryNet, 293
GetNumDevices
CTcxDeviceNet, 575
GetNumEntries
DriverVersionNet, 670
GetNumFrequencyRanges
CPgaDeviceNet, 400
GetNumMeasureChannels
CTcxDeviceNet, 575
GetNUV_MS
CRoboDacqNet, 453
GetOffsetCurrent
CMeaCoatDeviceNet, 341
GetOnOff
CPositionIIDeviceNet, 404
CTcxDeviceNet, 575
GetOutputCurrent
CMeaCoatDeviceNet, 341
GetOutputRate
CStg200xBasicNet, 522
GetOutputVoltage
CMeaCleanDeviceNet, 336
GetP
CTcxDeviceNet, 575
GetParameter
CRoboDeviceNet::RoboMainLowLevelCommands, 686
GetPattern
CMeaSwitchDeviceNet, 361

GetPatternBool
CMeaSwitchDeviceNet, 362
GetPauseDuration
CMeaCoatDeviceNet, 341
GetPCoeff
CRobo_FYITemp_FunctionNet, 444
GetPDecp
CTcxDeviceNet, 575
GetPeriod
CPulseGeneratorFunctionNet, 431
GetPeriod_us
CTEERFunctionNet, 587
GetPermanentCurrentInMicroAmp
CMultiwellOptoStimFunctionNet, 383
GetPGain
CRoboDacqNet, 453
GetPhases
CRoboDeviceNet::RoboMainLowLevelCommands, 686
GetPicFirmwareType
CW2100_FunctionNet, 600
GetPiezoState
CMcsBus_SensorNet, 225
GetPlateClampLockState
CMultiwellDeviceNet, 376
GetPlateClampState
CMultiwellDeviceNet, 376
GetPlateClampStateByHeadstage
CMultiwellCallbackFunctionNet, 373
GetPlateClampStateByHeadstageEvent
CMultiwellCallbackFunctionNet, 374
GetPlateMux
CMultiwellDeviceNet, 376, 377
GetPlateType
CMultiwellDeviceNet, 377
GetPMax
CTcxDeviceNet, 575
GetPMin
CTcxDeviceNet, 575
GetPoti
CMcsUsbDacqNet, 269
GetPOut
CTcxDeviceNet, 575
GetPowerMuxPlate
CMultiwellDeviceNet, 377
GetPowerStrength
CPositionIIDeviceNet, 405
GetPressure
CMcsBus_SensorNet, 225
GetPressureOffset
CMcsBus_SensorNet, 225
GetPressureRange
CPPCFunctionNet, 417
GetPulseform
COkuvisionStimulatorDeviceNet, 391
GetPulseLength
CPulseGeneratorFunctionNet, 431
GetPumpCouple
CPPS_FunctionNet, 424
GetPumpEnableSpeedRatio
CPPS_FunctionNet, 424
GetPumpFastOnOff
CPPS_FunctionNet, 424
GetPumpFastSpeed
CPPS_FunctionNet, 424
GetPumpFunctionSpeeds
CPPS_FunctionNet, 425
GetPumpManualOnOff
CPPS_FunctionNet, 425
GetPumpMaxSpeed
CPPS_FunctionNet, 425
GetPumpModeType
CPPCFunctionNet, 417
CPPS_FunctionNet, 425
GetPumpSpeed
CRoboFluidDeviceNet, 477
GetPumpSpeedRatio
CPPS_FunctionNet, 425
GetPumpSpeedUnit
CPPCFunctionNet, 418
CPPS_FunctionNet, 425
GetPWM
CFluidControlDeviceNet, 139
GetPwrOut
CTcxDeviceNet, 576
GetPwrSet
CTcxDeviceNet, 576
GetRatedCapacity
CMultiBatteryChargerDeviceNet, 370
GetReady
CMeaImpedanceDeviceNet, 359
GetRecordingNumber
CRoboDacqNet, 453
GetReferenceElectrodeMode
CSCUFunctionNet, 501
GetReferenceElectrodeSwitchState
CSCUFunctionNet, 501
GetReferenceTemperature
CFluidControlDeviceNet, 139
GetRegulationTimeouts
CMcsBus_SensorNet, 225
GetRegulatorFactor
CMcsBus_SensorNet, 226
GetRegulatorOnOff
CMcsBus_SensorNet, 226
CRobo_FYITemp_FunctionNet, 444
GetRegulatorStatus
CMcsBus_SensorNet, 226
GetRepeats
CProgramPressureCurveNet, 429
GetRes1
CTcxDeviceNet, 576
GetRes2
CTcxDeviceNet, 576
GetResetFilter
CWClassicFunctionNet, 658

GetResistanceC
CRoboDacqNet, 453
GetResistanceV
CRoboDacqNet, 453
GetResolutionPerDigit
CMcsUsbDacqNet, 269
GetResS
CTcxDeviceNet, 576
GetResult
CMeaImpedanceDeviceNet, 359
GetResX
CTcxDeviceNet, 576
GetRFConnectionStatus
CWClassicFunctionNet, 658
GetRFFrequency
CPositionImpDeviceNet, 411
GetRFFrequencyHeadstage
CWClassicFunctionNet, 658
GetRFFrequencyReceiver
CWClassicFunctionNet, 658
GetRFPower
CWClassicFunctionNet, 659
GetRoboDacq
CRoboocyte2DeviceNet, 481
GetRoboFluidDevice
CEncapsulatorDeviceNet, 127
CRoboocyte2DeviceNet, 481
GetRotaryPositionCode
CTEERFunctionNet, 587
GetRotatePump
CMcsBus_SensorNet, 226
GetROut
CTcxDeviceNet, 576
GetRTC
COkuvisionStimulatorDeviceNet, 392
CPositionIIDeviceNet, 405
GetSampleBufferChunk
CTEERFunctionNet, 587
GetSampleInterval
CLIH3DeviceNet, 193
GetSamplePeriode
CMcsBus_SensorNet, 226
GetSampleRate
CTEERFunctionNet, 587
GetSamplerate
CMcsUsbDacqNet, 269
GetSampleVoltageBuffer_uV
CTEERFunctionNet, 588
GetScaleFactorU1
CTEERFunctionNet, 588
GetScaleFactorU2
CTEERFunctionNet, 588
GetScanHeadstagesResult
CWClassicFunctionNet, 659
GetScreen
CRoboDacqNet, 453
GetSearchReferenceFastAccel
CRoboDeviceNet::RoboMainLowLevelCommands, 686
GetSearchReferenceFastSpeed
CRoboDeviceNet::RoboMainLowLevelCommands, 686
GetSearchReferenceFineAccel
CRoboDeviceNet::RoboMainLowLevelCommands, 686
GetSearchReferenceFineSpeed
CRoboDeviceNet::RoboMainLowLevelCommands, 686
GetSearchReferenceMethod
CRoboDeviceNet::RoboMainLowLevelCommands, 687
GetSearchReferenceMoveOut
CRoboDeviceNet::RoboMainLowLevelCommands, 687
GetSearchReferenceOffsetPos
CRoboDeviceNet::RoboMainLowLevelCommands, 687
GetSelectedChannels
CW2100_FunctionNet, 601
GetSelectedHeadstage
CWClassicFunctionNet, 659
GetSelectedHeadstageState
CW2100_FunctionNet, 601
GetSensorType
CTcxDeviceNet, 577
GetSerialNumber
CMcsUsbNet, 318
DriverVersionNet, 670
GetSerialNumberHeadstage
CWClassicFunctionNet, 659
GetSetpoint
CTcxDeviceNet, 577
GetSetpointDecp
CTcxDeviceNet, 577
GetSetpointMax
CTcxDeviceNet, 577
GetSetpointMin
CTcxDeviceNet, 577
GetShortBuffer
CGenericDevelopDeviceNet, 154
GetSimulation
CRoboDacqNet, 453
GetSingleHeater
CMcsBus_FYIExtensionNet, 201
GetSingleValve
CFluidControlDeviceNet, 139
CRoboFluidDeviceNet, 477
GetSlope
CMeaCleanDeviceNet, 337
CMeaCoatDeviceNet, 341
GetSoftwareKey
CMcsUsbNet, 318
GetSoftwareKeyString
CMcsUsbNet, 318
GetSollPressure

CMcsBus_SensorNet, 226
GetSollTemp
CRobo_FYITemp_FunctionNet, 444
GetSourceBulk
CCMOSMea_FunctionNet, 105
GetSourceDrain
CCMOSMea_FunctionNet, 105
GetSourceGate
CCMOSMea_FunctionNet, 105
GetStartTriggerSlope
CDigOutStimulatorFunctionNet, 124
GetState
CRFFunctionNet, 440
CRobo_FYIProgram_FunctionNet, 443
GetStateDebugData
CPositionIIDeviceNet, 405
GetStateEventData
CPositionIIDeviceNet, 406
GetStatus
CMcsUsbNet, 318
DriverVersionNet, 671
GetStatusOfLastCommand
CMcsUsbNet, 319
GetStgProgramInfo
CStg200xBasicNet, 522, 523
GetStgVersionInfo
CStg200xBasicNet, 523
GetStimulationPatternMemory
CW2100_StimulatorFunctionNet, 607
GetStimulatorStatus
COkuvisionStimulatorDeviceNet, 392
GetStimulusParametersCache
CW2100_FunctionNet, 601
GetStimulusParametersFromSelectedHS
CW2100_FunctionNet, 601
GetStimulusSites
CCMOSMea_FunctionNet, 105
GetStiumlusParameters
CW2100_FunctionNet, 601
GetStopTriggerSlope
CDigOutStimulatorFunctionNet, 124
GetSubChannel
CMcsBus_MotorControlNet, 211
GetSupplyVoltage
CPPCFunctionNet, 418
CPPS_FunctionNet, 425
GetSweepCount
CStg200xDownloadBasicNet, 544
GetSync
CWarnerValveControllerDeviceTesterFunctionNet, 655
GetSyncoutMap
CStg200xBasicNet, 523
GetSyncState
CMcsBus_SensorNet, 226
GetTableName
CWarnerValveControllerDeviceNet, 635
GetTableNamebyIndex
CWarnerValveControllerDeviceNet, 636
GetTableNamebyIndexEvent
CWarnerValveControllerDeviceNet, 651
GetTablepointer
CRetinaLedDeviceNet, 436
GetTemperatur
CMcsBus_TempSensorNet, 230
GetTestMode
CRFFunctionNet, 440
GetThermocoupleCalibration
CFluidControlDeviceNet, 140
CTcxDeviceNet, 577
GetThermocoupleNanovoltPerKelvin
CFluidControlDeviceNet, 140
CTcxDeviceNet, 577
GetThermocoupleReferenceTemp
CTcxDeviceNet, 578
GetThermocoupleTemp
CTcxDeviceNet, 578
GetThermocoupleTempAbs
CTcxDeviceNet, 578
GetThermocoupleTemperature
CFluidControlDeviceNet, 140
GetThermoOffset
CMcsBus_TempSensorNet, 230
GetThermoTemp
CMcsBus_TempSensorNet, 231
GetThermoVoltage
CMcsBus_TempSensorNet, 231
GetTimeInPause
CMeaCoatDeviceNet, 341
GetTimeInPlateau
CMeaCoatDeviceNet, 342
GetTimeResolutionInNanoSeconds
CW2100_StimulatorFunctionNet, 607
GetTimeSlot
CW2100_StimulatorFunctionNet, 607
GetTotalMemory
CStg200xBasicNet, 524
CStimulusFunctionNet, 560
GetTotalNumberOfDigitalPorts
CWarnerValveControllerDeviceNet, 636
GetTotalNumberOfTables
CWarnerValveControllerDeviceNet, 636
GetTotalNumberOfValves
CWarnerValveControllerDeviceNet, 636
GetTotalTableSize
CWarnerValveControllerDeviceNet, 637
GetTrigger
CStg200xDownloadBasicNet, 544
GetTriggerSource
CStg200xBasicNet, 524
GetU1Offset
CWarnerUssingFunctionNet, 618
GetU1Reference
CWarnerUssingFunctionNet, 620
GetU2Offset
CWarnerUssingFunctionNet, 620

GetU2Reference
CWarnerUssingFunctionNet, 620
GetUByteBuffer
CGenericDevelopDeviceNet, 154
GetUC
CRoboDacqNet, 453
GetUClamp
CRoboDacqNet, 454
GetUCOffset
CRoboDacqNet, 454
GetUintA
CFilterCoefficientsNet, 130
GetUintB
CFilterCoefficientsNet, 130
GetUIntBuffer
CGenericDevelopDeviceNet, 156
GetUnit
CTcxDeviceNet, 578
GetUnitDescription
CWarnerUssingFunctionNet, 621
GetUnitExponent
CWarnerUssingFunctionNet, 621
GetUnitName
CWarnerUssingFunctionNet, 621
GetUnitsPerDigit
CWarnerUssingFunctionNet, 622
GetUOut
CTcxDeviceNet, 578
GetUpdateDisplay
CRoboDacqNet, 454
GetUpperFrequencyByIndex
CIntanMea_FunctionNet, 183
GetUptimeSeconds
CTEERFunctionNet, 588
CWarnerUssingFunctionNet, 622
GetUSBDeviceIDFromFX3Image
CMcsUsbFactoryNet, 294
GetUsbListEntries
CMcsUsbListNet, 306
GetUsbListEntry
CMcsUsbListNet, 306
CMcsUsbNet, 319
GetUseBubble
CPPS_FunctionNet, 425
GetUsercodeFromBitFile
CMcsUsbFactoryNet, 294
GetUsercodeFromFlash
CMcsUsbFactoryNet, 294
GetUserDefinedName
CW2100_FunctionNet, 601
GetUserDefinedNameCache
CW2100_FunctionNet, 601
GetUserDefinedNameFromSelectedHS
CW2100_FunctionNet, 602
GetUserParameter
CRoboDeviceNet::RoboMainLowLevelCommands, 687
GetUShortBuffer
CGenericDevelopDeviceNet, 157
GetUV
CRoboDacqNet, 454
GetUVOffset
CRoboDacqNet, 454
GetValue
CTcxDeviceNet, 578
GetValueHires
CTcxDeviceNet, 579
GetValve
CFluidControlDeviceNet, 141
CRoboFluidDeviceNet, 477
GetValve1
CRobo_FYIProgram_FunctionNet, 443
GetValve2
CRobo_FYIProgram_FunctionNet, 443
GetValveActive
CPPCFunctionNet, 418
CWarnerValveControllerDeviceNet, 637
GetValveActiveEvent
CWarnerValveControllerDeviceNet, 651
GetValveBoardRevision
CWarnerValveControllerDeviceNet, 637
GetValveBoardRevisionEvent
CWarnerValveControllerDeviceNet, 651
GetValveBoardRevisionString
CWarnerValveControllerDeviceNet, 637
GetValveCurrent
CWarnerValveControllerDeviceNet, 637
GetValveDigitalInPort
CWarnerValveControllerDeviceNet, 638
GetValveDigitalInPortEvent
CWarnerValveControllerDeviceNet, 652
GetValveLedOn
CWarnerValveControllerDeviceNet, 638
GetValveLedOnEvent
CWarnerValveControllerDeviceNet, 652
GetValveManualGroup
CWarnerValveControllerDeviceNet, 638
GetValveManualGroupEvent
CWarnerValveControllerDeviceNet, 652
GetValveManualState
CWarnerValveControllerDeviceNet, 639
GetValveManualStateEvent
CWarnerValveControllerDeviceNet, 652
GetValveMode
CWarnerValveControllerDeviceNet, 639
GetValveModeEvent
CWarnerValveControllerDeviceNet, 652
GetValves
CMcsBus_FYIExtensionNet, 201
GetValvesActiveMap
CWarnerValveControllerDeviceNet, 639
GetValvesManualStateMap
CWarnerValveControllerDeviceNet, 639
GetValveTableEntry
CWarnerValveControllerDeviceNet, 640
GetVDD3I

CCMOSMea_FunctionNet, 105
GetVDDI
CCMOSMea_FunctionNet, 105
GetVds
CGrapheneFunctionNet, 169
GetVdVs
CGrapheneFunctionNet, 169, 170
GetVdVsDAC
CGrapheneFunctionNet, 170
GetVersion
CMcsUsbNet, 319
GetVersionInt
DriverVersionNet, 671
GetVersionString
DriverVersionNet, 672
GetVgs
CGrapheneFunctionNet, 171
GetVMMaxNegativeCurrent
CMcsBus_VoltageModeNet, 233
GetVMMaxNegativeCurrentEeprom
CMcsBus_VoltageModeNet, 233
GetVMMaxNegativeVoltage
CMcsBus_VoltageModeNet, 233
GetVMMaxNegativeVoltageEeprom
CMcsBus_VoltageModeNet, 233
GetVMMaxPositiveCurrent
CMcsBus_VoltageModeNet, 233
GetVMMaxPositiveCurrentEeprom
CMcsBus_VoltageModeNet, 233
GetVMMaxPositiveVoltage
CMcsBus_VoltageModeNet, 233
GetVMMaxPositiveVoltageEeprom
CMcsBus_VoltageModeNet, 234
GetVMOutputOnOff
CMcsBus_VoltageModeNet, 234
GetVMVoltage
CMcsBus_VoltageModeNet, 234
GetVoltage
COkuvisionStimulatorDeviceNet, 392
GetVoltage12V
CRoboDeviceNet, 466
GetVoltage12VLimit
CRoboDeviceNet, 466
GetVoltage5V
CRoboDeviceNet, 466
GetVoltage5VLimit
CRoboDeviceNet, 466
GetVoltageAirvalve
CRoboDeviceNet, 466
GetVoltageAirvalveLimit
CRoboDeviceNet, 466
GetVoltageClampControllerParam_D
CWarnerUssingFunctionNet, 622
GetVoltageClampControllerParam_I
CWarnerUssingFunctionNet, 623
GetVoltageClampControllerParam_P
CWarnerUssingFunctionNet, 623
GetVoltageRange
CGrapheneFunctionNet, 171
GetVoltageRangeIndex
CMcsUsbDacqNet, 269
GetVoltageRangeInMicroVolt
CMcsUsbDacqNet, 270
CStg200xBasicNet, 524
CStimulusFunctionNet, 560
CW2100_StimulatorFunctionNet, 607
GetVoltageRangeInMilliVolt
CMcsUsbDacqNet, 270
GetVoltageReached
CGrapheneFunctionNet, 172
GetVoltageResolution
CGrapheneFunctionNet, 172
GetVoltageResolutionInMicroVolt
CStg200xBasicNet, 524
CStimulusFunctionNet, 561
CW2100_StimulatorFunctionNet, 608
GetVoltageRs485A
CRoboDeviceNet, 467
GetVoltageRs485ALimit
CRoboDeviceNet, 467
GetVoltageRs485B
CRoboDeviceNet, 467
GetVoltageRs485BLimit
CRoboDeviceNet, 467
GetVoltageValves
CRoboDeviceNet, 467
GetVoltageValvesLimit
CRoboDeviceNet, 467
GetVolti
CTcxDeviceNet, 579
GetWaveform
CTEERFunctionNet, 588
GetWaveLengthInNanometer
CMultiwellOptoStimFunctionNet, 383
GetWorkingFrequency
CRFFunctionNet, 441
GetWPADebugMode
CWClassicFunctionNet, 659
GetWPAType
CWClassicFunctionNet, 659
GetXGain
CRoboDacqNet, 454
GetXilinxFlashOffset
CMcsUsbFactoryNet, 294
GetXilinxFlashReadCommand
CMcsUsbFactoryNet, 294
GND_SWITCH_BIT
CW2100_StimulatorFunctionNet, 610
Graphene_FlagShip_Core_2
Mcs::Usb, 75
GrapheneFlagshipCore2
Mcs::Usb, 78
GrapheneFlagshipCore2Headstage
Mcs::Usb, 65
GrapheneProjectTestDevice
Mcs::Usb, 76

Ground
Mcs::Usb, 54, 62
GroupID
CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumTemplateNet >, 121
GroupType
CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumTemplateNet >, 121
GyroOnly
Mcs::Usb, 87

Hardware
Mcs::Usb, 64
HasAnalogOut
CSCUFunctionNet, 502
HasGalvanicIsolation
CSCUFunctionNet, 502
HasHSPowerSwitch
CSCUFunctionNet, 502
HasIMU
HeadStageIDType, 677
HasOptoCurrentMessurement
HeadStageIDType, 677
HasRadioControl
CRadioControledDevicesNet, 434
HasRef
CRoboDeviceNet::RoboMainLowLevelCommands, 688
HasRefI
CRoboStatorDeviceNet, 484
HasRefXY
CRoboStatorDeviceNet, 484
HasRefZ
CRoboStatorDeviceNet, 484
HasSoftwareKey
CMcsUsbNet, 319
Headstage1NCBathCurrentGroup
Mcs::Usb, 57
Headstage1NCCol2CurrentGroup
Mcs::Usb, 57
Headstage1NChipTempGroup
Mcs::Usb, 57
HeadstageElectrodeGroup
Mcs::Usb, 57
HeadstageIdEnumNet
Mcs::Usb, 65
HeadStageIDType, 675
CompareTo, 677
ElectricalStimulation, 676
Entry, 677
Equals, 677
HasIMU, 677
HasOptoCurrentMessurement, 677
HeadStageIDType, 676
HeadstageType, 677
HeadstageTypeEnum, 676
ID, 677
MeasuringOnly, 676
NumberOfAnalogChannels, 677
NumberOfStimulationChannels, 678
OpticalStimulation, 676
SN, 678
StimulusParameters, 678
ToString, 677
Type, 678
TypeValue, 678
Unknown, 676
UserDefinedName, 678
Valid, 678
W16IsW14, 678
HeadstageIDTypeObject, 678
_AdditionalText, 679
_IdType, 679
AdditionalText, 680
Equals, 679
GetHashCode, 679
HeadstageIDTypeObject, 679
IdType, 680
ToString, 679
HeadStageIDTypeState, 680
ControlState, 680
DataState, 680
IdType, 680
State, 680
HeadstageType
HeadStageIDType, 677
HeadstageTypeEnum
HeadStageIDType, 676
HEKA_LIH3_DEVICE
Mcs::Usb, 59
HekaEPC10Double
Mcs::Usb, 74
HekaEPC10Quadro
Mcs::Usb, 74
HekaEPC10Single
Mcs::Usb, 74
HekaEPC10Triple
Mcs::Usb, 74
HekaEPCLite
Mcs::Usb, 74
HekaITEV100
Mcs::Usb, 74
HekaLIH30
Mcs::Usb, 73
HekaLIH406
Mcs::Usb, 74
HekaLIH816
Mcs::Usb, 74
HekaPG610
Mcs::Usb, 74
HekaPG611
Mcs::Usb, 74
HekaPG612
Mcs::Usb, 74
HekaPG618
Mcs::Usb, 74
HekaPG690

Mcs::Usb, 74
HiClamp
Mcs::Usb, 76
HiClamp4Uart
Mcs::Usb, 76
Highpass
Mcs::Usb, 63
HighSpeed
Mcs::Usb, 69
HLA
Mcs::Usb, 74
HLADacq
CHLADeviceNet, 179
Hs1Digital
Mcs::Usb, 60
HS1DigitalData1
Mcs::Usb, 83
HS1ElectrodeGroup
Mcs::Usb, 69
HS1Sideband1
Mcs::Usb, 60, 70, 80
HS1Sideband10
Mcs::Usb, 70, 80
HS1Sideband11
Mcs::Usb, 70, 80
HS1Sideband12
Mcs::Usb, 70, 80
HS1Sideband13
Mcs::Usb, 70
HS1Sideband14
Mcs::Usb, 70
HS1Sideband15
Mcs::Usb, 70
HS1Sideband16
Mcs::Usb, 70
HS1Sideband17
Mcs::Usb, 70
HS1Sideband18
Mcs::Usb, 70
HS1Sideband2
Mcs::Usb, 60, 70, 80
HS1Sideband3
Mcs::Usb, 60, 70, 80
HS1Sideband4
Mcs::Usb, 60, 70, 80
HS1Sideband5
Mcs::Usb, 60, 70, 80
HS1Sideband6
Mcs::Usb, 60, 70, 80
HS1Sideband7
Mcs::Usb, 70, 80
HS1Sideband8
Mcs::Usb, 70, 80
HS1Sideband9
Mcs::Usb, 70, 80
Hs1SidebandHigh
Mcs::Usb, 60
Hs1SidebandLow
Mcs::Usb, 60
Hs1Trigger
Mcs::Usb, 60
HS1Trigger10Status
Mcs::Usb, 70, 80
HS1Trigger11Status
Mcs::Usb, 70, 80
HS1Trigger12Status
Mcs::Usb, 70, 80
HS1Trigger13Status
Mcs::Usb, 70
HS1Trigger14Status
Mcs::Usb, 70
HS1Trigger15Status
Mcs::Usb, 70
HS1Trigger16Status
Mcs::Usb, 70
HS1Trigger17Status
Mcs::Usb, 70
HS1Trigger18Status
Mcs::Usb, 70
HS1Trigger1Status
Mcs::Usb, 60, 70, 79
HS1Trigger2Status
Mcs::Usb, 60, 70, 80
HS1Trigger3Status
Mcs::Usb, 60, 70, 80
HS1Trigger4Status
Mcs::Usb, 60, 70, 80
HS1Trigger5Status
Mcs::Usb, 60, 70, 80
HS1Trigger6Status
Mcs::Usb, 60, 70, 80
HS1Trigger7Status
Mcs::Usb, 70, 80
HS1Trigger8Status
Mcs::Usb, 70, 80
HS1Trigger9Status
Mcs::Usb, 70, 80
Hs2Digital
Mcs::Usb, 60
HS2DigitalData1
Mcs::Usb, 83
HS2ElectrodeGroup
Mcs::Usb, 69
HS2Sideband1
Mcs::Usb, 61, 71, 80
HS2Sideband10
Mcs::Usb, 71, 80
HS2Sideband11
Mcs::Usb, 71, 80
HS2Sideband12
Mcs::Usb, 71, 80
HS2Sideband13
Mcs::Usb, 71
HS2Sideband14
Mcs::Usb, 71
HS2Sideband15

Mcs::Usb, 71
HS2Sideband16
Mcs::Usb, 71
HS2Sideband17
Mcs::Usb, 71
HS2Sideband18
Mcs::Usb, 71
HS2Sideband2
Mcs::Usb, 61, 71, 80
HS2Sideband3
Mcs::Usb, 61, 71, 80
HS2Sideband4
Mcs::Usb, 61, 71, 80
HS2Sideband5
Mcs::Usb, 61, 71, 80
HS2Sideband6
Mcs::Usb, 61, 71, 80
HS2Sideband7
Mcs::Usb, 71, 80
HS2Sideband8
Mcs::Usb, 71, 80
HS2Sideband9
Mcs::Usb, 71, 80
Hs2SidebandHigh
Mcs::Usb, 60
Hs2SidebandLow
Mcs::Usb, 60
Hs2Trigger
Mcs::Usb, 60
HS2Trigger10Status
Mcs::Usb, 71, 80
HS2Trigger11Status
Mcs::Usb, 71, 80
HS2Trigger12Status
Mcs::Usb, 71, 80
HS2Trigger13Status
Mcs::Usb, 71
HS2Trigger14Status
Mcs::Usb, 71
HS2Trigger15Status
Mcs::Usb, 71
HS2Trigger16Status
Mcs::Usb, 71
HS2Trigger17Status
Mcs::Usb, 71
HS2Trigger18Status
Mcs::Usb, 71
HS2Trigger1Status
Mcs::Usb, 60, 71, 80
HS2Trigger2Status
Mcs::Usb, 60, 71, 80
HS2Trigger3Status
Mcs::Usb, 61, 71, 80
HS2Trigger4Status
Mcs::Usb, 61, 71, 80
HS2Trigger5Status
Mcs::Usb, 61, 71, 80
HS2Trigger6Status
Mcs::Usb, 61, 71, 80
HS2Trigger7Status
Mcs::Usb, 71, 80
HS2Trigger8Status
Mcs::Usb, 71, 80
HS2Trigger9Status
Mcs::Usb, 71, 80
HWInfo
CMcsUsbDacqNet, 270
HwVersion
CMcsUsbListEntryNet, 304

ID
HeadStageIDType, 677
Idle
Mcs::Usb, 81
IdProduct
DeviceIdNet, 663
IdType
HeadstageIDTypeObject, 680
HeadStageIDTypeState, 680
IdVendor
DeviceIdNet, 663
IFB2GoldenInterfaceboard
Mcs::Usb, 65
IFB30GoldenInterfaceboard
Mcs::Usb, 65
IFChannel1
Mcs::Usb, 56
IFChannel2
Mcs::Usb, 56
IFChannel3
Mcs::Usb, 56
IFChannel4
Mcs::Usb, 56
IFChannel5
Mcs::Usb, 56
IFChannel6
Mcs::Usb, 56
IFChannel7
Mcs::Usb, 56
IFChannel8
Mcs::Usb, 56
IFDigChannelsGroup
Mcs::Usb, 57, 69, 79, 89
IM16KRC
Mcs::Usb, 76
IM16S16KRA
Mcs::Usb, 76
IM16S8KRA
Mcs::Usb, 76
IM64KRB
Mcs::Usb, 76
IM64KRC
Mcs::Usb, 76
Input
Mcs::Usb, 73
Intel
Mcs::Usb, 86

InterfaceADCGroup
Mcs::Usb, 57, 69, 79, 88
InterfaceBoard2
Mcs::Usb, 65
IntToDouble
Mcs::Usb, 63
InvitroSignalCollectorUnit
Mcs::Usb, 65
InvivoSignalCollectorUnit
Mcs::Usb, 65
IoVoltageEnumNet
Mcs::Usb, 65
IS32KRA
Mcs::Usb, 76
IsAnalogOutEnabled
CSCUFunctionNet, 502
IsAutomaticAnalogOut
CSCUFunctionNet, 502
IsBusy
CPPCFunctionNet, 418
IsChamberAvailable
CWarnerUssingFunctionNet, 623
IsChipPowered
CCMOSMea_FunctionNet, 105
IsConnected
CMcsUsbNet, 319
IsDeviceHighSpeed
CMcsUsbNet, 319
IsDeviceHighSpeedCapable
CMcsUsbNet, 320
IsDeviceTypeOf
CMcsUsbListNet, 307
IsDigitalChannelDedicated
CMcsUsbDacqNet::CHWInfo, 181
IsDigitalOutPortInverted
CWarnerValveControllerDeviceNet, 640
IsDigitalOutPortInvertedEvent
CWarnerValveControllerDeviceNet, 652
IsEqual
CFilterCoefficientsNet, 130
IsExceptionsEnabled
CMcsUsbNet, 320
IsGateFloating
CCMOSMea_FunctionNet, 105
IsHeadstageAvailable
CSCUFunctionNet, 502
IsHeadstageAvailableEvent
CSCUFunctionNet, 506
IsHighCurrentMode
CWarnerUssingFunctionNet, 624
IsHSPowered
CSCUFunctionNet, 503
IsInDacqLegacyMode
CSCUFunctionNet, 503
IsInternalCalibrationFinished
CTEERFunctionNet, 589
CWarnerUssingFunctionNet, 624
IsPlateTypeValid
CMultiwellDeviceNet, 378
IsPulseEnabled
CWarnerUssingFunctionNet, 624
IsPumpMotorOn
CRoboFluidDeviceNet, 478
IsQueueEnabled
CRoboDeviceNet, 467
IsQueueStarted
CRoboDeviceNet, 467
IsRunning
CMeaCleanDeviceNet, 337
CMeaCoatDeviceNet, 342
IsSamplingFinished
CTEERFunctionNet, 589
IsUserTriggerEnabled
CLIH3DeviceNet, 193
IsValveDigitalInInverted
CWarnerValveControllerDeviceNet, 640
IsValveDigitalInInvertedEvent
CWarnerValveControllerDeviceNet, 652
IsValveOpen
CWarnerValveControllerDeviceNet, 641
IsValveOpenEvent
CWarnerValveControllerDeviceNet, 653
IsValveOpenInAnalogMode
CWarnerValveControllerDeviceNet, 641
IsValveOpenInAnalogModeEvent
CWarnerValveControllerDeviceNet, 653
IsValveOpenInDigitalMode
CWarnerValveControllerDeviceNet, 641
IsValveOpenInDigitalModeEvent
CWarnerValveControllerDeviceNet, 653

Kelvin
Mcs::Usb, 51

LastPosition
Mcs::Usb, 61, 71, 80, 83, 89
LIH30_ADC_Channel_EnumNet
Mcs::Usb, 66
LIH30_DAC_Channel_EnumNet
Mcs::Usb, 66
LIH30_EPC10_Bus_EnumNet
Mcs::Usb, 66
LIH30ADCCtrl
Mcs::Usb, 65
LIH30ADCModulesGroup
Mcs::Usb, 57
LIH30Interfaceboard
Mcs::Usb, 65
LIH30TestADCGroup
Mcs::Usb, 57
LIH30UserADCGroup
Mcs::Usb, 57
ListModeSendStart
CStg200xBasicNet, 525
ListModeSendStop
CStg200xBasicNet, 525
ListOfChangedTriggers

StgStatusNet, 694
LoadPressure
CPPCFunctionNet, 420
LoadUserFirmware
CMcsUsbFactoryNet, 294, 295
LoadValveTable
CWarnerValveControllerDeviceNet, 641
Lock
Mcs::Usb, 73
LockPlateClamp
CMultiwellDeviceNet, 378
Lowpass
Mcs::Usb, 63
LowSpeed
Mcs::Usb, 69

m_Bottom
CCMOSMeaDeviceNet::CRegionOfInterestRect, 435
m_Left
CCMOSMeaDeviceNet::CRegionOfInterestRect, 435
m_pGilsonDevice
CGilsonDeviceNet, 165
m_pMcsBus_MotorControlNet
CRoboFluidDeviceNet, 479
m_pMcsUsb
CMcsUsbFunctionNet, 299
m_pMcsUsbFunction
CMcsUsbFunctionNet, 299
m_pRoboFluidDevice
CRoboFluidDeviceNet, 479
m_Right
CCMOSMeaDeviceNet::CRegionOfInterestRect, 435
m_Top
CCMOSMeaDeviceNet::CRegionOfInterestRect, 435
Manual
Mcs::Usb, 73, 81, 90
Manufacturer
CMcsUsbListEntryNet, 304
MaxBitNumber
DigitalSource< digitalsourceenum >, 664
DigitalSourceGeneral, 665, 666
MaxBitNumberStatic
DigitalSource< digitalsourceenum >, 664
MBC08
Mcs::Usb, 75
MbcChannelStateEnumNet
Mcs::Usb, 67
MbcChargingModeEnumNet
Mcs::Usb, 67
MbcRatedCapacityEnumNet
Mcs::Usb, 67
MC_Card
Mcs::Usb, 73
MCS
Mcs::Usb, 87
Mcs, 22
Mcs::Usb, 22
A, 67
AccelOnly, 87
AdapterTypeEnumNet, 50
ALA_VC3, 73, 87
ALA_VC3_DEVICE, 59
ALTERA, 52
ALTERA_BASE, 53
ALTERA_BOOTSTRAP, 53
ALTERA_GOLD, 53
ALTERA_TARGET1, 54
ALTERA_TARGET2, 54
ALTERA_TARGET3, 54
AlwaysOn, 76
Ampere, 51, 87
AMS_Dongle, 74
Analog, 73, 90
AnalogGroup, 58
AnalogOut_DAC_Range_EnumNet, 51
AnalogSource_HS1, 51
AnalogSource_HS2, 51
AnalogSource_IF, 51
AnalogSourceEnumNet, 51
AnalogUnitEnumNet, 51
Any, 73, 87
Armed, 81
ASMedia, 86
AudioTestChannelGroup, 57, 69, 79, 89
Aux, 60
AuxIn, 60, 70, 79, 82, 89
AuxPort, 56
B, 67
Bessel, 64
BMI, 83
Bootstrap, 52
BootstrapOtherCypress, 52
Both, 87
Break, 77
BUS1MCSBUS1, 52
BUS1MCSBUS2, 52
BUS2MCSBUS1, 52
BUS2MCSBUS2, 52
BUSNUMBER1, 52
BUSNUMBER2, 52
Butterworth, 64
Campden_Ci4600EphysVideoDataIntegrator, 73
CatchAmp, 72
CFirmwareDestinationNet, 51
channeldata_current, 82
channeldata_current_always_boost, 82
channeldata_current_always_boost_own_sync, 82
channeldata_current_own_boost_gnd_sync, 82
channeldata_current_own_sync, 82
channeldata_positive_current, 82
channeldata_positive_current_own_boost_gnd_sync, 82
channeldata_positive_current_own_sync, 82

channeldata_positive_voltage, 82
channeldata_voltage, 82
ChannelPIC, 52
ChannelTest, 75
ChecksumAndPacketCounter, 56
Ci4600Intan, 50
ClampModeCurrent, 84
ClampModeInternalCalibration, 84
ClampModeOpen, 84
ClampModeVoltage, 83
Close, 72
CmosMea, 65
CMOSMeaBathModeEnumNet, 54
CmosMeaHeadstage, 64
CMOSMeaHeadstage1NCBathCurrentEnumNet, 54
CMOSMeaHeadstage1NCCol2CurrentEnumNet, 54
CMOSMeaHeadstage1NChipTempEnumNet, 55
CMOSMeaHS1SidebandEnumNet, 55
CMOSMeaHS1TriggerStatusEnumNet, 55
CmosmeaIFB2, 65
CMOSMeaIFDigChannelEnumNet, 55
CMOSMeaInterfaceADCEnumNet, 56
CmosMeaInterfaceboard, 64
CMOSMeaPacketFrameContextGroupEnumNet, 56
CMOSMeaSTG1DACSignalEnumNet, 56
CMOSMeaValueUnitEnumNet, 57
CommaPositionA, 63
CommaPositionB, 63
csCapacityTestDischarge, 67
csCapacityTestPrecharge, 67
csCharge, 67
csDischarge, 67
csError, 67
csIdleChargeFinished, 67
csIdleNoBattery, 67
csRefreshBattery, 67
CurrentClamp, 87
CurrentMeasure, 54
Cypress, 87
Cypress_FX1, 73
Cypress_FX2, 73
Cypress_FX3, 73
DAC1Channel, 56
DAC2Channel, 56
DAC3Channel, 56
DAC4Channel, 56
DACQ1DigitalGroup, 57
DacqGroupChannelEnumNet, 57
DacqMeaGroupTypeEnumNet, 57
DacqTrigger, 62
DataModeEnumNet, 58
DEST_FX3_TARGET_MASK, 54
DEST_TARGET1, 53
DEST_TARGET10, 53
DEST_TARGET11, 53
DEST_TARGET12, 53
DEST_TARGET13, 54
DEST_TARGET14, 54
DEST_TARGET15, 54
DEST_TARGET2, 53
DEST_TARGET3, 53
DEST_TARGET4, 53
DEST_TARGET5, 53
DEST_TARGET6, 53
DEST_TARGET7, 53
DEST_TARGET8, 53
DEST_TARGET9, 53
DEST_TARGET_MASK, 54
DEVICE_NOT_FOUND, 63
DeviceEnumNet, 58
DeviceHasNoHeadstage, 65, 78
DeviceNotConnected, 64, 65, 78
DeviceRunStatus, 61, 70, 79, 83, 89
DigDataFromReceiver, 89
Digital, 73, 90
DigitalData, 61, 70, 79, 83, 89
DigitalDatastreamEnableEnumNet, 59
DigitalGroup, 58
DigitalIn, 59, 60, 70, 79, 82, 89
DigitalInOfOutPort, 60, 70, 79, 82, 89
DigitalInPort, 55
DigitalInReserverd, 59
DigitalMux, 55
DigitalOut, 59
DigitalOutReg, 55
DigitalOutReserved, 60
DigitalOutStimulator, 61, 70, 79, 83, 89
DigitalPulse, 60, 70, 79, 82, 89
DigitalReg, 56
DigitalSourceEnumNet, 60
DigitalStimulatorTriggerEventEnumNet, 61
DigitalStimulatorTriggerSlopeEnumNet, 61
DigitalTargetEnumNet, 61
Digout, 62
DigOutStim, 60
DigOutStimulatorStartTrigger, 62
DigOutStimulatorStopTrigger, 62
Digstream, 62
DigStreamFromReceiver, 89
DigStreamToReceiver, 62
Dilutor, 76
DongleS, 74
Dotriapot, 74
DoubleToInt, 63
DownloadOnly, 81
DSP, 51
DSPAnalogGroup, 69, 79
DSPDataGroup, 57, 88
DSPDigitalGroup, 69, 79
eCube, 75, 78
eCubeHeadstage, 65
ElectrodeDacMuxEnumNet, 62
ElectrodeModeEnumNet, 62

ElectrodeOffset, 87
emAutomatic, 62
emManual, 62
Encapsulator, 76
enCMosMeaChipType, 62
EnSTG200x_STATUS, 63
EOFAndCRC, 56
ExternBCTester, 74
ExternDTester, 74
ExternSTester, 74
Falling, 61
FCB, 74
FCX, 74
Feedback, 60, 70, 79, 82, 89
FeedbackHigh, 60
FeedbackLow, 60
FeedbackReg, 56
FilterAttributeEnumNet, 63
FilterBandEnumNet, 63
FilterCalculationDirectionEnumNet, 63
FilterFamilyEnumNet, 63
FilterTypeEnumNet, 64
Finished, 81
FPGA10, 52
FPGA10_BASE, 53
FPGA10_GOLD, 53
FPGA11, 52
FPGA11_BASE, 53
FPGA11_GOLD, 53
FPGA12, 52
FPGA12_BASE, 53
FPGA12_GOLD, 53
FPGA13, 52
FPGA13_BASE, 53
FPGA13_GOLD, 53
FPGA14, 52
FPGA14_BASE, 53
FPGA14_GOLD, 53
FPGA15, 52
FPGA15_BASE, 53
FPGA15_GOLD, 53
FPGA16, 53
FPGA16_BASE, 53
FPGA16_GOLD, 53
FPGA2, 52
FPGA2_BASE, 53
FPGA2_GOLD, 53
FPGA3, 52
FPGA3_BASE, 53
FPGA3_GOLD, 53
FPGA4, 52
FPGA4_BASE, 53
FPGA4_GOLD, 53
FPGA5, 52
FPGA5_BASE, 53
FPGA5_GOLD, 53
FPGA6, 52
FPGA6_BASE, 53
FPGA6_GOLD, 53
FPGA7, 52
FPGA7_BASE, 53
FPGA7_GOLD, 53
FPGA8, 52
FPGA8_BASE, 53
FPGA8_GOLD, 53
FPGA9, 52
FPGA9_BASE, 53
FPGA9_GOLD, 53
FPGA_BASE, 53
FPGA_BOOTSTRAP, 53
FPGA_GOLD, 53
FPGA_NORMAL, 51
FpgaIdEnumNet, 64
FrameContextGroup, 58
FullCharge, 67
FullSpeed, 69
FunkDongleS, 74
Gated_High_Active, 76
Gated_Low_Active, 76
GE2100, 75
Generic, 74
Graphene_FlagShip_Core_2, 75
GrapheneFlagshipCore2, 78
GrapheneFlagshipCore2Headstage, 65
GrapheneProjectTestDevice, 76
Ground, 54, 62
GyroOnly, 87
Hardware, 64
Headstage1NCBathCurrentGroup, 57
Headstage1NCCol2CurrentGroup, 57
Headstage1NChipTempGroup, 57
HeadstageElectrodeGroup, 57
HeadstageIdEnumNet, 65
HEKA_LIH3_DEVICE, 59
HekaEPC10Double, 74
HekaEPC10Quadro, 74
HekaEPC10Single, 74
HekaEPC10Triple, 74
HekaEPCLite, 74
HekaITEV100, 74
HekaLIH30, 73
HekaLIH406, 74
HekaLIH816, 74
HekaPG610, 74
HekaPG611, 74
HekaPG612, 74
HekaPG618, 74
HekaPG690, 74
HiClamp, 76
HiClamp4Uart, 76
Highpass, 63
HighSpeed, 69
HLA, 74
Hs1Digital, 60
HS1DigitalData1, 83
HS1ElectrodeGroup, 69

HS1Sideband1, 60, 70, 80
HS1Sideband10, 70, 80
HS1Sideband11, 70, 80
HS1Sideband12, 70, 80
HS1Sideband13, 70
HS1Sideband14, 70
HS1Sideband15, 70
HS1Sideband16, 70
HS1Sideband17, 70
HS1Sideband18, 70
HS1Sideband2, 60, 70, 80
HS1Sideband3, 60, 70, 80
HS1Sideband4, 60, 70, 80
HS1Sideband5, 60, 70, 80
HS1Sideband6, 60, 70, 80
HS1Sideband7, 70, 80
HS1Sideband8, 70, 80
HS1Sideband9, 70, 80
Hs1SidebandHigh, 60
Hs1SidebandLow, 60
Hs1Trigger, 60
HS1Trigger10Status, 70, 80
HS1Trigger11Status, 70, 80
HS1Trigger12Status, 70, 80
HS1Trigger13Status, 70
HS1Trigger14Status, 70
HS1Trigger15Status, 70
HS1Trigger16Status, 70
HS1Trigger17Status, 70
HS1Trigger18Status, 70
HS1Trigger1Status, 60, 70, 79
HS1Trigger2Status, 60, 70, 80
HS1Trigger3Status, 60, 70, 80
HS1Trigger4Status, 60, 70, 80
HS1Trigger5Status, 60, 70, 80
HS1Trigger6Status, 60, 70, 80
HS1Trigger7Status, 70, 80
HS1Trigger8Status, 70, 80
HS1Trigger9Status, 70, 80
Hs2Digital, 60
HS2DigitalData1, 83
HS2ElectrodeGroup, 69
HS2Sideband1, 61, 71, 80
HS2Sideband10, 71, 80
HS2Sideband11, 71, 80
HS2Sideband12, 71, 80
HS2Sideband13, 71
HS2Sideband14, 71
HS2Sideband15, 71
HS2Sideband16, 71
HS2Sideband17, 71
HS2Sideband18, 71
HS2Sideband2, 61, 71, 80
HS2Sideband3, 61, 71, 80
HS2Sideband4, 61, 71, 80
HS2Sideband5, 61, 71, 80
HS2Sideband6, 61, 71, 80
HS2Sideband7, 71, 80
HS2Sideband8, 71, 80
HS2Sideband9, 71, 80
Hs2SidebandHigh, 60
Hs2SidebandLow, 60
Hs2Trigger, 60
HS2Trigger10Status, 71, 80
HS2Trigger11Status, 71, 80
HS2Trigger12Status, 71, 80
HS2Trigger13Status, 71
HS2Trigger14Status, 71
HS2Trigger15Status, 71
HS2Trigger16Status, 71
HS2Trigger17Status, 71
HS2Trigger18Status, 71
HS2Trigger1Status, 60, 71, 80
HS2Trigger2Status, 60, 71, 80
HS2Trigger3Status, 61, 71, 80
HS2Trigger4Status, 61, 71, 80
HS2Trigger5Status, 61, 71, 80
HS2Trigger6Status, 61, 71, 80
HS2Trigger7Status, 71, 80
HS2Trigger8Status, 71, 80
HS2Trigger9Status, 71, 80
Idle, 81
IFB2GoldenInterfaceboard, 65
IFB30GoldenInterfaceboard, 65
IFChannel1, 56
IFChannel2, 56
IFChannel3, 56
IFChannel4, 56
IFChannel5, 56
IFChannel6, 56
IFChannel7, 56
IFChannel8, 56
IFDigChannelsGroup, 57, 69, 79, 89
IM16KRC, 76
IM16S16KRA, 76
IM16S8KRA, 76
IM64KRB, 76
IM64KRC, 76
Input, 73
Intel, 86
InterfaceADCGroup, 57, 69, 79, 88
InterfaceBoard2, 65
IntToDouble, 63
InvitroSignalCollectorUnit, 65
InvivoSignalCollectorUnit, 65
IoVoltageEnumNet, 65
IS32KRA, 76
Kelvin, 51
LastPosition, 61, 71, 80, 83, 89
LIH30_ADC_Channel_EnumNet, 66
LIH30_DAC_Channel_EnumNet, 66
LIH30_EPC10_Bus_EnumNet, 66
LIH30ADCCtrl, 65
LIH30ADCModulesGroup, 57
LIH30Interfaceboard, 65
LIH30TestADCGroup, 57

LIH30UserADCGroup, 57
Lock, 73
Lowpass, 63
LowSpeed, 69
Manual, 73, 81, 90
MBC08, 75
MbcChannelStateEnumNet, 67
MbcChargingModeEnumNet, 67
MbcRatedCapacityEnumNet, 67
MC_Card, 73
MCS, 87
MCS_ANY_BUS, 69
MCS_CHANNELTEST_DEVICE, 58
MCS_DEVICE_ANY, 58
MCS_DEVICE_USB, 58
MCS_DEVICE_USB_CYPRESS, 59
MCS_ENCAPSULATOR_DEVICE, 59
MCS_EXTERN_BC_TESTER_DEVICE, 59
MCS_EXTERN_D_TESTER_DEVICE, 59
MCS_FCX_DEVICE, 58
MCS_FYI_DEVICE, 59
MCS_GENERIC_DEVELOPMENT_DEVICE, 58
MCS_HICLAMP_DEVICE, 59
MCS_HLA_DEVICE, 59
MCS_MBC08_DEVICE, 59
MCS_MC_STIMULUS_DEVICE, 58
MCS_MCCARD_DEVICE, 58
MCS_MEA_CLEAN_DEVICE, 59
MCS_MEA_COAT_DEVICE, 59
MCS_MEA_DEVICE, 58
MCS_MEA_IMPEDANCE_DEVICE, 58
MCS_MEA_SWITCH_DEVICE, 58
MCS_MEASURETABLE_DEVICE, 59
MCS_MEAUSB_DEVICE, 58
MCS_NF_GEN_DEVICE, 59
MCS_OCTOPOT_DEVICE, 58
MCS_OKUVISION_STIMULATOR_DEVICE, 59
MCS_PATCHSERVER_DEVICE, 59
MCS_PATHIDENT_DEVICE, 59
MCS_PCI_BUS, 69
MCS_PCX_DEVICE, 58
MCS_PEDOTER_DEVICE, 59
MCS_PERISTALTIC_PUMP_DEVICE, 59
MCS_PGA_DEVICE, 58
MCS_PPC_DEVICE, 59
MCS_PPS5_DEVICE, 59
MCS_PPS_DEVICE, 59
MCS_RETINA_AMS_DONGLE, 59
MCS_RETINA_LED_DEVICE, 58
MCS_ROBO_DEVICE, 59
MCS_ROBOINJECT_DEVICE, 59
MCS_ROBOOCYTE2_DEVICE, 59
MCS_SAFEIS_DEVICE, 59
MCS_SMARTIMPLANT_DEVICE, 59
MCS_SOFTWARE_DONGLE_DEVICE, 59
MCS_STG_DEVICE, 58
MCS_SW2TO64_DEVICE, 58
MCS_TCX_DEVICE, 58
MCS_TERSENS_DEVICE, 58
MCS_UNDEFINED_BUS, 69
MCS_USB_BUS, 69
MCSBUS0, 52
MCSBUS1, 51
MCSBUS10, 52
MCSBUS11, 52
MCSBUS12, 52
MCSBUS13, 52
MCSBUS14, 52
MCSBUS15, 52
MCSBUS2, 51
MCSBUS3, 52
MCSBUS4, 52
MCSBUS5, 52
MCSBUS6, 52
MCSBUS7, 52
MCSBUS8, 52
MCSBUS9, 52
McsBusTypeEnumNet, 67
McsUsbSpeedEnumNet, 69
MCU1, 51
ME128, 75
ME16, 75
ME2100, 75
Me2100_32PICiCE40, 78
Me2100_32PICiCE40Headstage, 64
Me2100_32Xilinx, 78
Me2100_32XilinxHeadstage, 64
Me2100Graphene16_32, 78
Me2100Graphene16_32Headstage, 65
Me2100Interfaceboard, 64
Me2100InvitroSignalCollectorUnit, 64
Me2100InvivoSignalCollectorUnit, 64
Me2100UPA32, 78
Me2100UPA32Headstage, 64
ME256, 75
ME32, 75
ME64, 75
MEA1060, 75
MEA120, 50
MEA2100, 75
Mea2100, 65
MEA2100_256, 75
Mea2100_256, 65
MEA2100_256DacqGroupChannelEnumNet, 69
MEA2100_256DigitalSourceEnumNet, 69
Mea2100_256Headstage, 64
Mea2100_256Interfaceboard, 64
MEA2100_32, 75
MEA2100_Lite, 75
Mea2100_Lite, 65
MEA2100_Mini, 75
MEA2100_Mini_Usb_develop, 75
MEA2100BetaScreen, 75
Mea2100BetaScreen, 78
Mea2100BetaScreenHeadstage, 64
Mea2100Headstage, 64

Mea2100Interfaceboard, 64
Mea2100LiteHeadstage, 65
Mea2100Mini120, 78
Mea2100Mini120Headstage, 64
Mea2100Mini60ECP5, 78
Mea2100Mini60ECP5Headstage, 65
Mea2100Mini60PICiCE40, 78
Mea2100Mini60PICiCE40Headstage, 64
Mea2100MultiwellIFB2, 64
Mea2100STG, 64
MEA252, 50
MEA256, 75
MEA2x32, 50
MEA2x60, 50
MEA32, 50
MEA60, 50
MEA_2_252_2, 50
MEA_2_252_2_6Well, 50
MEA_2_252_2_9Well, 50
MEA_2_252_2_Test, 50
MEA_Clean, 75
MEA_Coat, 75
MEA_Impedance, 75
MEA_Sanofi, 75
MEA_Switch, 75
MEA_Switch_2_1, 75
MEA_Switch_4_2, 75
MeaLayoutEnumNet, 71
MeasureTable, 76
MicroAmpere, 57
MilliDegreeCelsius, 57
mIMEA60, 71
mIUnknown, 71
ModulA_ADC0, 66
ModulA_ADC1, 66
ModulA_ADC2, 66
ModulA_ADC3, 66
ModulA_DAC0, 66
ModulA_DAC1, 66
ModulB_ADC0, 66
ModulB_ADC1, 66
ModulB_ADC2, 66
ModulB_ADC3, 66
ModulB_DAC0, 66
ModulB_DAC1, 66
ModulC_ADC0, 66
ModulC_ADC1, 66
ModulC_ADC2, 66
ModulC_ADC3, 66
ModulC_DAC0, 66
ModulC_DAC1, 66
ModulD_ADC0, 66
ModulD_ADC1, 66
ModulD_ADC2, 66
ModulD_ADC3, 66
ModulD_DAC0, 66
ModulD_DAC1, 66
Monitor, 81
Movement, 77
Multiboot, 75
Multiwell, 65, 75
Multiwell96, 50
Multiwell_ICC, 75
Multiwell_MEA_Mini, 75
MultiwellHeadstage, 64
MultiwellInterfaceboard, 64
MultiwellMini, 78
MultiwellMiniHeadstage, 64
MultiwellOptoStim, 74
MultiwellPlateTypeEnumNet, 72
Mux, 59
MuxOtherDevice, 59
Nanion, 83
NanoAmpere, 57
NanoVolt, 57
NCBathCurrent, 54
NCCol2Current, 54
NChipTemperature, 55
Neptun, 76
NeuroChip, 75
NeurochipConfig, 76
NF_Gen, 75
NineWell, 82
nMos16LV, 62
nMos32LV, 62
nMos36LN, 62
nMos64LN, 62
No_Plate, 72
None, 50, 59, 73, 81, 86, 87
Normal, 72
NOT_CONNECTED, 63
NotApplicable, 50
NoUnit, 57
NTC10K, 83
Octopot, 74
Off, 76, 77, 87
off, 77
OK, 63
Okuvision_Stimulator, 74
OnChannelData, 90
OnDeviceArrivalRemoval, 90
One, 60, 70, 79, 82, 89
OnError, 90
OnMcsUsbDeviceState, 90
OnMcsUsbDeviceStateCallback, 91
OnMwPollStatus, 91
OnStg200xDataHandler, 91
OnStg200xErrorHandler, 91
OnStgPollStatus, 91
OnUpdateFirmwareProgress, 91
OnUpdateFirmwareStatusChange, 91
Open, 72
OpenClamp, 87
Output, 73
PacketFrameContextGroup, 57, 69, 79, 89
PatchServAdcModeEnumNet, 72

PatchServer, 76
PathIdent, 75
PC, 90
PCI, 87
PCX, 74
PeriodicPulse, 60
PeristalticPump, 75
PGA, 74
PIC, 52
PIC10, 52
PIC11, 52
PIC12, 52
PIC2, 52
PIC3, 52
PIC4, 52
PIC5, 52
PIC6, 52
PIC7, 52
PIC8, 52
PIC9, 52
PicoAmpere, 57
Plate_24W030MGA, 72
Plate_24W300_30_1152GBA, 72
Plate_24W300_30GBA, 72
Plate_24W300_30GBB, 72
Plate_24W300_30GMA, 72
Plate_24W700_100FMA, 72
Plate_24W700_100FMB, 72
Plate_24W700_100FMC, 72
Plate_24W700_100PBA, 72
Plate_72W500_100FMA, 72
Plate_72W500_100PMA, 72
Plate_96W300_80_1152FMA, 72
Plate_96W400_80_1152FMB, 72
Plate_96W700_100FMA, 72
Plate_96W700_100FMB, 72
Plate_96W700_100GBA, 72
Plate_96W700_100GBB, 72
Plate_96W700_100GBC, 72
Plate_96W700_100GBD, 72
Plate_96W700_100GMA, 72
Plate_Dummy, 72
Plate_Dummy_126, 72
PlateClampEnumNet, 72
PlateClampLockEnumNet, 73
PlusMinus10Volts, 51
PlusMinus2Comma5Volts, 51
PlusMinus5Volts, 51
PortDirectionEnumNet, 73
Pos900, 76
PositionBase, 76
PositionIIBase, 76
PositionIICentralUnit, 76
PositionImp, 76
PostCommaA, 63
PostCommaB, 63
PP_Pump_Mode_Type_EnumNet, 73
PPC, 75
PPS2, 75
PPS4plus1, 75
PPS5, 75
PPS5_DIG, 75
PreCommaA, 63
PreCommaB, 63
ProductIdEnumNet, 73
PT100, 83
PT1000, 83
PulseGenerator, 61, 70, 79, 83, 89
PulseGenerator_Mode_EnumNet, 76
raGate, 77
raIgnore, 77
raRestart, 77
raSingle, 77
raStop, 77
rawdata, 82
RC, 64
rc100mAh, 67
rc200mAh, 67
rc300mAh, 67
rc30mAh, 67
rcGreater300mAh, 67
Rectangle, 84
Ref16, 77
Ref24, 77
Ref32, 77
Ref8, 77
Reference, 77
ReferenceElectrodeModeEnumNet, 76
ReferenceElectrodeSwitchPositionEnumNet, 77
RegisterHigh, 60
RegisterLow, 60
Regular, 83
Renesas, 86
Reserved1, 83
Reserved2, 83
Reserved3, 83
Reserved4, 83
Reserved5, 83
Retina_LED, 74
RetriggerActionEnumNet, 77
Rising, 61
RoboCurrentModeEnumNet, 77
RoboInject, 76
Robooycte2, 76
RoboStatusEventDelegate, 91
Running, 81
SafeIS, 76
SampleDstSize16, 78
SampleDstSize32, 78
SampleDstSizeNet, 78
SampleSize16Signed, 78
SampleSize16Unsigned, 78
SampleSize24Signed, 78
SampleSize24Unsigned, 78
SampleSize32Signed, 78
SampleSize32Unsigned, 78

SampleSize64Signed, 78
SampleSize64Unsigned, 78
SampleSizeNet, 78
SBSVector1, 55
SBSVector2, 55
SBSVector3, 55
SBSVector4, 55
SCU1ElectrodeGroupHS1, 79
SCU1ElectrodeGroupHS2, 79
SCU1ElectrodeGroupHS3, 79
SCU1ElectrodeGroupHS4, 79
SCU2ElectrodeGroupHS1, 79
SCU2ElectrodeGroupHS2, 79
SCU2ElectrodeGroupHS3, 79
SCU2ElectrodeGroupHS4, 79
SCU_HeadstageIdEnumNet, 78
SCUDacqGroupChannelEnumNet, 79
SCUDigitalSourceEnumNet, 79
Settings, 90
Signed_16bit, 58
Signed_24bit, 58
Signed_32bit, 58
Sine, 84
SingleWell, 82
SixWell, 82
SmartImplant, 76
SOFAndCTRLword, 56
Software, 64
SoftwareDongle, 75
Standby, 77, 87
Start, 61
State, 87
STG, 74
Stg1, 62
STG1DACSignalGroup, 57, 69, 79
STG1SidebandsGroup, 57, 69, 79
STG1TriggerStatusGroup, 57, 69, 79
Stg2, 62
Stg200xDigoutModeEnumNet, 81
Stg200xSegmentFlagsEnumNet, 81
Stg200xTriggerStatusEnumNet, 81
STG2DACSignalGroup, 69, 79
STG2SidebandsGroup, 69, 79
STG2TriggerStatusGroup, 69, 79
Stg3, 62
STG3008_FA, 74
STG4002, 74
STG4002_opto, 74
STG4004, 74
STG4004_opto, 74
STG4008, 74
STG4008_opto, 74
STG400x, 74
STG400x_opto, 74
STG5, 74
STG_DestinationEnumNet, 82
StgListModeTrigger, 62
StgTrigger, 62
Stimulation, 54
StimulationLayoutConfigurationEnumNet, 82
Stop, 61, 72
StorageCharge, 67
SubtractFromAll, 77
SubtractFromAllOther, 77
SubtractFromReferenceElectrodeOnly, 77
SubtractionOff, 77
SuperSpeed, 69
Sw2to64, 75
SYNCOUT1, 81
SYNCOUT2, 81
SYNCOUT3, 81
SYNCOUT4, 81
SYNCOUT5, 81
SYNCOUT6, 81
SYNCOUT7, 81
SYNCOUT8, 81
syncoutdata, 82
SyncStart, 81
Table, 90
TBSI_127, 50
TBSI_15, 50
TBSI_31, 50
TBSI_5, 50
TBSI_63, 50
TBSI_Dacq, 75
TBSI_DACQDigitalSourceEnumNet, 82
TBSI_Reserved, 50
TbsiDacq, 65
TbsiDacqHeadstage, 64
TbsiDacqInterfaceboard, 64
TC01, 74
TC02, 74
TCX, 74
TcxDeviceTypeEnumNet, 83
TcxSensorTypeEnumNet, 83
TeerClampModeEnumNet, 83
TeerWaveformEnumNet, 84
Tersens, 74
Test_ADC_EPC10, 66
Test_DAC_EPC10, 66
Timestamp, 56
TouchTest, 90
Triggerbox_AMS, 74
Triggerbox_AMS3, 74
Triggerbox_IMS, 74
Triggerbox_R5, 74
TriggerOnly, 81
TriggerSourceEnumNet, 84
TriggerStatus1, 55
TriggerStatus2, 55
TriggerStatus3, 55
TriggerStatus4, 55
tsAuxIn1, 85
tsAuxIn2, 85
tsDACQCy1Dev1Runs, 86
tsDACQCy1Dev2Runs, 86

tsDACQCy2Dev1Runs, 86
tsDACQCy2Dev2Runs, 86
tsDigitalIn1, 84
tsDigitalIn10, 84
tsDigitalIn11, 84
tsDigitalIn12, 84
tsDigitalIn13, 84
tsDigitalIn14, 84
tsDigitalIn15, 84
tsDigitalIn16, 84
tsDigitalIn17, 84
tsDigitalIn18, 84
tsDigitalIn19, 84
tsDigitalIn2, 84
tsDigitalIn20, 84
tsDigitalIn21, 84
tsDigitalIn22, 84
tsDigitalIn23, 84
tsDigitalIn24, 84
tsDigitalIn25, 85
tsDigitalIn26, 85
tsDigitalIn27, 85
tsDigitalIn28, 85
tsDigitalIn29, 85
tsDigitalIn3, 84
tsDigitalIn30, 85
tsDigitalIn31, 85
tsDigitalIn32, 85
tsDigitalIn4, 84
tsDigitalIn5, 84
tsDigitalIn6, 84
tsDigitalIn7, 84
tsDigitalIn8, 84
tsDigitalIn9, 84
tsDigitalPuse0, 85
tsDigitalPuse1, 85
tsDigitalPuse10, 86
tsDigitalPuse11, 86
tsDigitalPuse12, 86
tsDigitalPuse13, 86
tsDigitalPuse14, 86
tsDigitalPuse15, 86
tsDigitalPuse16, 86
tsDigitalPuse17, 86
tsDigitalPuse18, 86
tsDigitalPuse19, 86
tsDigitalPuse2, 85
tsDigitalPuse20, 86
tsDigitalPuse21, 86
tsDigitalPuse22, 86
tsDigitalPuse23, 86
tsDigitalPuse24, 86
tsDigitalPuse25, 86
tsDigitalPuse26, 86
tsDigitalPuse27, 86
tsDigitalPuse28, 86
tsDigitalPuse29, 86
tsDigitalPuse3, 85
tsDigitalPuse30, 86
tsDigitalPuse31, 86
tsDigitalPuse4, 85
tsDigitalPuse5, 85
tsDigitalPuse6, 85
tsDigitalPuse7, 85
tsDigitalPuse8, 86
tsDigitalPuse9, 86
tsFeedback1, 85
tsFeedback10, 85
tsFeedback11, 85
tsFeedback12, 85
tsFeedback13, 85
tsFeedback14, 85
tsFeedback15, 85
tsFeedback16, 85
tsFeedback17, 85
tsFeedback18, 85
tsFeedback19, 85
tsFeedback2, 85
tsFeedback20, 85
tsFeedback21, 85
tsFeedback22, 85
tsFeedback23, 85
tsFeedback24, 85
tsFeedback25, 85
tsFeedback26, 85
tsFeedback27, 85
tsFeedback28, 85
tsFeedback29, 85
tsFeedback3, 85
tsFeedback30, 85
tsFeedback31, 85
tsFeedback32, 85
tsFeedback4, 85
tsFeedback5, 85
tsFeedback6, 85
tsFeedback7, 85
tsFeedback8, 85
tsFeedback9, 85
tsNone, 84
tsSidebandBit8, 86
tsTriggered, 86
Unknown, 50, 51, 63, 64, 83, 86
unknown, 62
UnknownDest, 54
UnknownSpeed, 69
Unlock, 73
Unsigned_16bit, 58
Unsigned_24bit, 58
Unsigned_32bit, 58
UpdateTrigger, 81
USB, 51
USB_TARGET1, 54
USB_TARGET2, 54
USB_TARGET3, 54
UsbTest, 75
UsbVendorIdEnumNet, 86

User_ADC_0, 66
User_ADC_1, 66
User_ADC_2, 66
User_ADC_3, 66
User_ADC_4, 66
User_DAC_0, 66
User_DAC_1, 66
User_DAC_2, 66
UssingChamber, 65
UssingClampModeEnumNet, 86
UssingRail, 65
UssingUnitEnumNet, 87
VendorIdEnumNet, 87
Volt, 51, 87
Voltage_3V3, 65
Voltage_5V0, 65
VoltageClamp, 87
W2100, 75
W2100_Accel_Gyro_Select_EnumNet, 87
W2100DacqGroupChannelEnumNet, 87
W2100DigitalSourceEnumNet, 89
W2100IFB2, 65
W2100Interfaceboard, 64
W2100WirelessReceiver, 64, 65
W2100WirelessReceiverAnalog, 64, 65
Warner, 83
Warner_TEER_Machine, 76
Warner_Ussing, 76
WARNER_USSING_DEVICE, 59
Warner_Valve_Control, 76
WARNER_VALVE_CONTROL_DEVICE, 59
Whole_Cell_Patch, 75
WholeCellPatch, 78
WholeCellPatchHeadstage, 65
WirelessHeadStageAccDataRE1HS1, 88
WirelessHeadStageAccDataRE1HS2, 88
WirelessHeadStageAccDataRE1HS3, 88
WirelessHeadStageAccDataRE1HS4, 88
WirelessHeadStageAccDataRE2HS1, 88
WirelessHeadStageAccDataRE2HS2, 88
WirelessHeadStageAccDataRE2HS3, 88
WirelessHeadStageAccDataRE2HS4, 88
WirelessHeadStageAnalogRE1HS1, 88
WirelessHeadStageAnalogRE1HS2, 88
WirelessHeadStageAnalogRE1HS3, 88
WirelessHeadStageAnalogRE1HS4, 88
WirelessHeadStageAnalogRE2HS1, 88
WirelessHeadStageAnalogRE2HS2, 88
WirelessHeadStageAnalogRE2HS3, 88
WirelessHeadStageAnalogRE2HS4, 88
WirelessHeadStageGyroDataRE1HS1, 88
WirelessHeadStageGyroDataRE1HS2, 88
WirelessHeadStageGyroDataRE1HS3, 88
WirelessHeadStageGyroDataRE1HS4, 88
WirelessHeadStageGyroDataRE2HS1, 88
WirelessHeadStageGyroDataRE2HS2, 88
WirelessHeadStageGyroDataRE2HS3, 88
WirelessHeadStageGyroDataRE2HS4, 88
WirelessHeadStageOptoStimCurrentRE1HS1, 88
WirelessHeadStageOptoStimCurrentRE1HS2, 88
WirelessHeadStageOptoStimCurrentRE1HS3, 88
WirelessHeadStageOptoStimCurrentRE1HS4, 88
WirelessHeadStageOptoStimCurrentRE2HS1, 88
WirelessHeadStageOptoStimCurrentRE2HS2, 89
WirelessHeadStageOptoStimCurrentRE2HS3, 89
WirelessHeadStageOptoStimCurrentRE2HS4, 89
WirelessHeadStageReservedARE1HS1, 88
WirelessHeadStageReservedARE1HS2, 88
WirelessHeadStageReservedARE1HS3, 88
WirelessHeadStageReservedARE1HS4, 88
WirelessHeadStageReservedARE2HS1, 89
WirelessHeadStageReservedARE2HS2, 89
WirelessHeadStageReservedARE2HS3, 89
WirelessHeadStageReservedARE2HS4, 89
WirelessHeadStageReservedBRE1HS1, 89
WirelessHeadStageReservedBRE1HS2, 89
WirelessHeadStageReservedBRE1HS3, 89
WirelessHeadStageReservedBRE1HS4, 89
WirelessHeadStageReservedBRE2HS1, 89
WirelessHeadStageReservedBRE2HS2, 89
WirelessHeadStageReservedBRE2HS3, 89
WirelessHeadStageReservedBRE2HS4, 89
WirelessHeadStageReservedCRE1HS1, 89
WirelessHeadStageReservedCRE1HS2, 89
WirelessHeadStageReservedCRE1HS3, 89
WirelessHeadStageReservedCRE1HS4, 89
WirelessHeadStageReservedCRE2HS1, 89
WirelessHeadStageReservedCRE2HS2, 89
WirelessHeadStageReservedCRE2HS3, 89
WirelessHeadStageReservedCRE2HS4, 89
WirelessHeadStageStatusRE1HS1, 88
WirelessHeadStageStatusRE1HS2, 88
WirelessHeadStageStatusRE1HS3, 88
WirelessHeadStageStatusRE1HS4, 88
WirelessHeadStageStatusRE2HS1, 88
WirelessHeadStageStatusRE2HS2, 88
WirelessHeadStageStatusRE2HS3, 88
WirelessHeadStageStatusRE2HS4, 88
WirelessTestAdapter, 50
Work, 90
WPA16, 75
WPA32, 75
WPA4, 75
WPA8, 75
WvcDisplayModeEnumNet, 90
WvcValveModeEnumNet, 90
Zero, 60, 70, 79, 82, 89
MCS_ANY_BUS
Mcs::Usb, 69
MCS_CHANNELTEST_DEVICE
Mcs::Usb, 58
MCS_DEVICE_ANY
Mcs::Usb, 58
MCS_DEVICE_USB
Mcs::Usb, 58
MCS_DEVICE_USB_CYPRESS

Mcs::Usb, 59
MCS_ENCAPSULATOR_DEVICE
Mcs::Usb, 59
MCS_EXTERN_BC_TESTER_DEVICE
Mcs::Usb, 59
MCS_EXTERN_D_TESTER_DEVICE
Mcs::Usb, 59
MCS_FCX_DEVICE
Mcs::Usb, 58
MCS_FYI_DEVICE
Mcs::Usb, 59
MCS_GENERIC_DEVELOPMENT_DEVICE
Mcs::Usb, 58
MCS_HICLAMP_DEVICE
Mcs::Usb, 59
MCS_HLA_DEVICE
Mcs::Usb, 59
MCS_MBC08_DEVICE
Mcs::Usb, 59
MCS_MC_STIMULUS_DEVICE
Mcs::Usb, 58
MCS_MCCARD_DEVICE
Mcs::Usb, 58
MCS_MEA_CLEAN_DEVICE
Mcs::Usb, 59
MCS_MEA_COAT_DEVICE
Mcs::Usb, 59
MCS_MEA_DEVICE
Mcs::Usb, 58
MCS_MEA_IMPEDANCE_DEVICE
Mcs::Usb, 58
MCS_MEA_SWITCH_DEVICE
Mcs::Usb, 58
MCS_MEASURETABLE_DEVICE
Mcs::Usb, 59
MCS_MEAUSB_DEVICE
Mcs::Usb, 58
MCS_NF_GEN_DEVICE
Mcs::Usb, 59
MCS_OCTOPOT_DEVICE
Mcs::Usb, 58
MCS_OKUVISION_STIMULATOR_DEVICE
Mcs::Usb, 59
MCS_PATCHSERVER_DEVICE
Mcs::Usb, 59
MCS_PATHIDENT_DEVICE
Mcs::Usb, 59
MCS_PCI_BUS
Mcs::Usb, 69
MCS_PCX_DEVICE
Mcs::Usb, 58
MCS_PEDOTER_DEVICE
Mcs::Usb, 59
MCS_PERISTALTIC_PUMP_DEVICE
Mcs::Usb, 59
MCS_PGA_DEVICE
Mcs::Usb, 58
MCS_PPC_DEVICE
Mcs::Usb, 59
MCS_PPS5_DEVICE
Mcs::Usb, 59
MCS_PPS_DEVICE
Mcs::Usb, 59
MCS_RETINA_AMS_DONGLE
Mcs::Usb, 59
MCS_RETINA_LED_DEVICE
Mcs::Usb, 58
MCS_ROBO_DEVICE
Mcs::Usb, 59
MCS_ROBOINJECT_DEVICE
Mcs::Usb, 59
MCS_ROBOOCYTE2_DEVICE
Mcs::Usb, 59
MCS_SAFEIS_DEVICE
Mcs::Usb, 59
MCS_SMARTIMPLANT_DEVICE
Mcs::Usb, 59
MCS_SOFTWARE_DONGLE_DEVICE
Mcs::Usb, 59
MCS_STG_DEVICE
Mcs::Usb, 58
MCS_SW2TO64_DEVICE
Mcs::Usb, 58
MCS_TCX_DEVICE
Mcs::Usb, 58
MCS_TERSENS_DEVICE
Mcs::Usb, 58
MCS_UNDEFINED_BUS
Mcs::Usb, 69
MCS_USB_BUS
Mcs::Usb, 69
McsBus
CPPCDeviceNet, 413
CPPS_DeviceNet, 422
CRoboDeviceNet, 475
MCSBUS0
Mcs::Usb, 52
MCSBUS1
FirmwareDestinationNames, 674
Mcs::Usb, 51
MCSBUS10
FirmwareDestinationNames, 674
Mcs::Usb, 52
MCSBUS11
FirmwareDestinationNames, 674
Mcs::Usb, 52
MCSBUS12
FirmwareDestinationNames, 674
Mcs::Usb, 52
MCSBUS13
FirmwareDestinationNames, 674
Mcs::Usb, 52
MCSBUS14
Mcs::Usb, 52
MCSBUS15
Mcs::Usb, 52

MCSBUS2
FirmwareDestinationNames, 674
Mcs::Usb, 51
MCSBUS3
FirmwareDestinationNames, 674
Mcs::Usb, 52
MCSBUS4
FirmwareDestinationNames, 674
Mcs::Usb, 52
MCSBUS5
FirmwareDestinationNames, 674
Mcs::Usb, 52
MCSBUS6
FirmwareDestinationNames, 674
Mcs::Usb, 52
MCSBUS7
FirmwareDestinationNames, 674
Mcs::Usb, 52
MCSBUS8
FirmwareDestinationNames, 675
Mcs::Usb, 52
MCSBUS9
FirmwareDestinationNames, 675
Mcs::Usb, 52
McsBus_MotorControl
CPeristalticPumpDeviceNet, 398
CPPCDeviceNet, 413
CPPS_DeviceNet, 422
CRoboDeviceNet, 475
CRoboFluidDeviceNet, 479
McsBus_Sensor
CPPCDeviceNet, 413
CPPS_DeviceNet, 422
McsBus_VoltageMode
CFluidControlDeviceNet, 143
McsBus_XY
CRoboDeviceNet, 472
McsBus_ZI
CRoboDeviceNet, 472
McsBusTypeEnumNet
Mcs::Usb, 67
McsUsbDeviceStateEvent
CMcsUsbDeviceStatePushFunctionNet, 288
CMcsUsbDeviceStatePushNet, 289
McsUsbSpeedEnumNet
Mcs::Usb, 69
MCU1
FirmwareDestinationNames, 675
Mcs::Usb, 51
ME128
Mcs::Usb, 75
ME16
Mcs::Usb, 75
ME2100
Mcs::Usb, 75
Me2100_32PICiCE40
Mcs::Usb, 78
Me2100_32PICiCE40Headstage
Mcs::Usb, 64
Me2100_32Xilinx
Mcs::Usb, 78
Me2100_32XilinxHeadstage
Mcs::Usb, 64
Me2100Graphene16_32
Mcs::Usb, 78
Me2100Graphene16_32Headstage
Mcs::Usb, 65
Me2100Interfaceboard
Mcs::Usb, 64
Me2100InvitroSignalCollectorUnit
Mcs::Usb, 64
Me2100InvivoSignalCollectorUnit
Mcs::Usb, 64
Me2100UPA32
Mcs::Usb, 78
Me2100UPA32Headstage
Mcs::Usb, 64
ME256
Mcs::Usb, 75
ME32
Mcs::Usb, 75
ME64
Mcs::Usb, 75
MEA1060
Mcs::Usb, 75
MEA120
Mcs::Usb, 50
MEA2100
Mcs::Usb, 75
Mea2100
Mcs::Usb, 65
MEA2100_256
Mcs::Usb, 75
Mea2100_256
Mcs::Usb, 65
MEA2100_256DacqGroupChannelEnumNet
Mcs::Usb, 69
MEA2100_256DigitalSourceEnumNet
Mcs::Usb, 69
Mea2100_256Headstage
Mcs::Usb, 64
Mea2100_256Interfaceboard
Mcs::Usb, 64
MEA2100_32
Mcs::Usb, 75
MEA2100_Lite
Mcs::Usb, 75
Mea2100_Lite
Mcs::Usb, 65
MEA2100_Mini
Mcs::Usb, 75
MEA2100_Mini_Usb_develop
Mcs::Usb, 75
MEA2100BetaScreen
Mcs::Usb, 75
Mea2100BetaScreen

Mcs::Usb, 78
Mea2100BetaScreenHeadstage
Mcs::Usb, 64
Mea2100Headstage
Mcs::Usb, 64
Mea2100Interfaceboard
Mcs::Usb, 64
Mea2100LiteHeadstage
Mcs::Usb, 65
Mea2100Mini120
Mcs::Usb, 78
Mea2100Mini120Headstage
Mcs::Usb, 64
Mea2100Mini60ECP5
Mcs::Usb, 78
Mea2100Mini60ECP5Headstage
Mcs::Usb, 65
Mea2100Mini60PICiCE40
Mcs::Usb, 78
Mea2100Mini60PICiCE40Headstage
Mcs::Usb, 64
Mea2100MultiwellIFB2
Mcs::Usb, 64
Mea2100STG
Mcs::Usb, 64
MEA252
Mcs::Usb, 50
MEA256
Mcs::Usb, 75
MEA2x32
Mcs::Usb, 50
MEA2x60
Mcs::Usb, 50
MEA32
Mcs::Usb, 50
MEA60
Mcs::Usb, 50
MEA_2_252_2
Mcs::Usb, 50
MEA_2_252_2_6Well
Mcs::Usb, 50
MEA_2_252_2_9Well
Mcs::Usb, 50
MEA_2_252_2_Test
Mcs::Usb, 50
MEA_Clean
Mcs::Usb, 75
MEA_Coat
Mcs::Usb, 75
MEA_Impedance
Mcs::Usb, 75
MEA_Sanofi
Mcs::Usb, 75
MEA_Switch
Mcs::Usb, 75
MEA_Switch_2_1
Mcs::Usb, 75
MEA_Switch_4_2
Mcs::Usb, 75
MeaAudioFunctionNet
CMeaDeviceNet, 351
MeaDigitalDataFunctionNet
CMeaDeviceNet, 351
MeaFeedbackFunctionNet
CMeaDeviceNet, 351
MeaLayoutEnumNet
Mcs::Usb, 71
Measure
CPathIdentDeviceNet, 395
MeasureReservoir
CPPCFunctionNet, 420
MeasureTable
Mcs::Usb, 76
MeasuringOnly
HeadStageIDType, 676
MeFunctionNet
CMeaDeviceNet, 352
MicroAmpere
Mcs::Usb, 57
MilliDegreeCelsius
Mcs::Usb, 57
mkfilter
mkfilterNet, 681
mkfilter_coef_in_one_set
mkfilterNet, 681
mkfilter_highpass_coeff
mkfilterNet, 682
mkfilter_highpass_frequency_from_coeff
mkfilterNet, 682
mkfilter_highpass_frequency_from_k
mkfilterNet, 682
mkfilter_highpass_k
mkfilterNet, 682
mkfilter_MCS
mkfilterNet, 682
mkfilter_MCS_k
mkfilterNet, 682, 683
mkfilter_normalize_coeffs_int
mkfilterNet, 683
mkfilter_normalize_coeffs_short
mkfilterNet, 683
mkfilter_normalize_scale_coeffs_int
mkfilterNet, 683
mkfilter_scale_coef_in_one_set
mkfilterNet, 683
mkfilterNet, 681
mkfilter, 681
mkfilter_coef_in_one_set, 681
mkfilter_highpass_coeff, 682
mkfilter_highpass_frequency_from_coeff, 682
mkfilter_highpass_frequency_from_k, 682
mkfilter_highpass_k, 682
mkfilter_MCS, 682
mkfilter_MCS_k, 682, 683
mkfilter_normalize_coeffs_int, 683
mkfilter_normalize_coeffs_short, 683

mkfilter_normalize_scale_coeffs_int, 683
mkfilter_scale_coef_in_one_set, 683
mlMEA60
Mcs::Usb, 71
mlUnknown
Mcs::Usb, 71
ModulA_ADC0
Mcs::Usb, 66
ModulA_ADC1
Mcs::Usb, 66
ModulA_ADC2
Mcs::Usb, 66
ModulA_ADC3
Mcs::Usb, 66
ModulA_DAC0
Mcs::Usb, 66
ModulA_DAC1
Mcs::Usb, 66
ModulB_ADC0
Mcs::Usb, 66
ModulB_ADC1
Mcs::Usb, 66
ModulB_ADC2
Mcs::Usb, 66
ModulB_ADC3
Mcs::Usb, 66
ModulB_DAC0
Mcs::Usb, 66
ModulB_DAC1
Mcs::Usb, 66
ModulC_ADC0
Mcs::Usb, 66
ModulC_ADC1
Mcs::Usb, 66
ModulC_ADC2
Mcs::Usb, 66
ModulC_ADC3
Mcs::Usb, 66
ModulC_DAC0
Mcs::Usb, 66
ModulC_DAC1
Mcs::Usb, 66
ModulD_ADC0
Mcs::Usb, 66
ModulD_ADC1
Mcs::Usb, 66
ModulD_ADC2
Mcs::Usb, 66
ModulD_ADC3
Mcs::Usb, 66
ModulD_DAC0
Mcs::Usb, 66
ModulD_DAC1
Mcs::Usb, 66
Monitor
Mcs::Usb, 81
MoveAbs
CRoboDeviceNet, 467, 468
MoveAbsI
CRoboStatorDeviceNet, 484
MoveAbsXY
CRoboStatorDeviceNet, 484
MoveAbsZ
CRoboStatorDeviceNet, 484, 485
Movement
Mcs::Usb, 77
Multiboot
Mcs::Usb, 75
MultibootGetCypressImageId
CMcsUsbNet, 320
MultibootGetImageId
CMcsUsbNet, 320
MultibootGetSelectedImage
CMcsUsbNet, 320
MultibootSelectImage
CMcsUsbNet, 320
Multiwell
Mcs::Usb, 65, 75
Multiwell96
Mcs::Usb, 50
Multiwell_ICC
Mcs::Usb, 75
Multiwell_MEA_Mini
Mcs::Usb, 75
MultiwellHeadstage
Mcs::Usb, 64
MultiwellInterfaceboard
Mcs::Usb, 64
MultiwellMini
Mcs::Usb, 78
MultiwellMiniHeadstage
Mcs::Usb, 64
MultiwellOptoStim
Mcs::Usb, 74
MultiwellPlateTypeEnumNet
Mcs::Usb, 72
Mux
Mcs::Usb, 59
MuxOtherDevice
Mcs::Usb, 59
MwPollStatusEvent
CStg200xDownloadNet, 554

Nanion
Mcs::Usb, 83
NanoAmpere
Mcs::Usb, 57
NanoVolt
Mcs::Usb, 57
NCBathCurrent
Mcs::Usb, 54
NCCol2Current
Mcs::Usb, 54
NChipTemperature
Mcs::Usb, 55
Neptun
Mcs::Usb, 76

NeuroChip
Mcs::Usb, 75
NeurochipConfig
Mcs::Usb, 76
NF_Gen
Mcs::Usb, 75
NineWell
Mcs::Usb, 82
nMos16LV
Mcs::Usb, 62
nMos32LV
Mcs::Usb, 62
nMos36LN
Mcs::Usb, 62
nMos64LN
Mcs::Usb, 62
No_Plate
Mcs::Usb, 72
None
Mcs::Usb, 50, 59, 73, 81, 86, 87
Normal
Mcs::Usb, 72
NOT_CONNECTED
Mcs::Usb, 63
NotApplicable
Mcs::Usb, 50
NoUnit
Mcs::Usb, 57
NTC10K
Mcs::Usb, 83
NullCommand
CRoboDeviceNet, 469
NumberOfAnalogChannels
HeadStageIDType, 677
NumberOfChannels
CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumTemplateNet >, 121
NumberOfStimulationChannels
HeadStageIDType, 678
NumCoefSets
CCreateFilterNet, 112

Octopot
Mcs::Usb, 74
Off
Mcs::Usb, 76, 77, 87
off
Mcs::Usb, 77
OK
Mcs::Usb, 63
Okuvision_Stimulator
Mcs::Usb, 74
OnChannelData
Mcs::Usb, 90
OnDeviceArrivalRemoval
Mcs::Usb, 90
One
Mcs::Usb, 60, 70, 79, 82, 89
OnError
Mcs::Usb, 90
OnGetActiveRunningTableNumber
CWarnerValveControllerDeviceNet, 642
OnGetAnalogThresholdHigh
CWarnerValveControllerDeviceNet, 642
OnGetAnalogThresholdLow
CWarnerValveControllerDeviceNet, 642
OnGetAnalogVoltage
CWarnerValveControllerDeviceNet, 642
OnGetAvailableHeadstages
CSCUFunctionNet, 503
OnGetCurrentNumberOfValves
CWarnerValveControllerDeviceNet, 642
OnGetDigitalOutPortValve
CWarnerValveControllerDeviceNet, 642
OnGetDigitalPortDirection
CWarnerValveControllerDeviceNet, 642
OnGetDisplayMode
CWarnerValveControllerDeviceNet, 642
OnGetPlateClampStateByHeadstage
CMultiwellCallbackFunctionNet, 373
OnGetTableNamebyIndex
CWarnerValveControllerDeviceNet, 642
OnGetValveActive
CWarnerValveControllerDeviceNet, 643
OnGetValveBoardRevision
CWarnerValveControllerDeviceNet, 643
OnGetValveDigitalInPort
CWarnerValveControllerDeviceNet, 643
OnGetValveLedOn
CWarnerValveControllerDeviceNet, 643
OnGetValveManualGroup
CWarnerValveControllerDeviceNet, 643
OnGetValveManualState
CWarnerValveControllerDeviceNet, 643
OnGetValveMode
CWarnerValveControllerDeviceNet, 643
OnIsDigitalOutPortInverted
CWarnerValveControllerDeviceNet, 643
OnIsHeadstageAvailable
CSCUFunctionNet, 503
OnIsValveDigitalInInverted
CWarnerValveControllerDeviceNet, 644
OnIsValveOpen
CWarnerValveControllerDeviceNet, 644
OnIsValveOpenInAnalogMode
CWarnerValveControllerDeviceNet, 644
OnIsValveOpenInDigitalMode
CWarnerValveControllerDeviceNet, 644
OnMcsUsbDeviceState
Mcs::Usb, 90
OnMcsUsbDeviceStateCallback
Mcs::Usb, 91
OnMwPollStatus
Mcs::Usb, 91
OnStg200xDataHandler
Mcs::Usb, 91
OnStg200xErrorHandler

Mcs::Usb, 91
OnStgPollStatus
Mcs::Usb, 91
OnTableEntryChanged
CWarnerValveControllerDeviceNet, 644
OnUpdateFirmwareProgress
Mcs::Usb, 91
OnUpdateFirmwareStatusChange
Mcs::Usb, 91
Open
Mcs::Usb, 72
OpenClamp
Mcs::Usb, 87
OpenPipe
CGenericDevelopDeviceNet, 157
OpenPlateClamp
CMultiwellDeviceNet, 378
operator=
DeviceIdNet, 663
OpticalStimulation
HeadStageIDType, 676
Order
CCreateFilterNet, 112
CFilterPropertyNet, 136
Output
Mcs::Usb, 73

PacketFrameContextGroup
Mcs::Usb, 57, 69, 79, 89
PatchServAdcModeEnumNet
Mcs::Usb, 72
PatchServer
Mcs::Usb, 76
PathIdent
Mcs::Usb, 75
PatternListStart
COctoPotDeviceNet, 387
PC
Mcs::Usb, 90
PCI
Mcs::Usb, 87
PCX
Mcs::Usb, 74
PeriodicPulse
Mcs::Usb, 60
PeristalticPump
Mcs::Usb, 75
PGA
Mcs::Usb, 74
PIC
FirmwareDestinationNames, 675
Mcs::Usb, 52
PIC10
Mcs::Usb, 52
PIC11
Mcs::Usb, 52
PIC12
Mcs::Usb, 52
PIC2
FirmwareDestinationNames, 675
Mcs::Usb, 52
PIC3
FirmwareDestinationNames, 675
Mcs::Usb, 52
PIC4
FirmwareDestinationNames, 675
Mcs::Usb, 52
PIC5
Mcs::Usb, 52
PIC6
Mcs::Usb, 52
PIC7
Mcs::Usb, 52
PIC8
Mcs::Usb, 52
PIC9
Mcs::Usb, 52
PicoAmpere
Mcs::Usb, 57
Plate_24W030MGA
Mcs::Usb, 72
Plate_24W300_30_1152GBA
Mcs::Usb, 72
Plate_24W300_30GBA
Mcs::Usb, 72
Plate_24W300_30GBB
Mcs::Usb, 72
Plate_24W300_30GMA
Mcs::Usb, 72
Plate_24W700_100FMA
Mcs::Usb, 72
Plate_24W700_100FMB
Mcs::Usb, 72
Plate_24W700_100FMC
Mcs::Usb, 72
Plate_24W700_100PBA
Mcs::Usb, 72
Plate_72W500_100FMA
Mcs::Usb, 72
Plate_72W500_100PMA
Mcs::Usb, 72
Plate_96W300_80_1152FMA
Mcs::Usb, 72
Plate_96W400_80_1152FMB
Mcs::Usb, 72
Plate_96W700_100FMA
Mcs::Usb, 72
Plate_96W700_100FMB
Mcs::Usb, 72
Plate_96W700_100GBA
Mcs::Usb, 72
Plate_96W700_100GBB
Mcs::Usb, 72
Plate_96W700_100GBC
Mcs::Usb, 72
Plate_96W700_100GBD
Mcs::Usb, 72

Plate_96W700_100GMA
Mcs::Usb, 72
Plate_Dummy
Mcs::Usb, 72
Plate_Dummy_126
Mcs::Usb, 72
PlateClampEnumNet
Mcs::Usb, 72
PlateClampLockEnumNet
Mcs::Usb, 73
PlusMinus10Volts
Mcs::Usb, 51
PlusMinus2Comma5Volts
Mcs::Usb, 51
PlusMinus5Volts
Mcs::Usb, 51
PollStatusEvent
CStimulusFunctionNet, 566
CW2100_StimulatorFunctionNet, 610
PortDirectionEnumNet
Mcs::Usb, 73
Pos900
Mcs::Usb, 76
PositionBase
Mcs::Usb, 76
PositionIIBase
Mcs::Usb, 76
PositionIICentralUnit
Mcs::Usb, 76
PositionImp
Mcs::Usb, 76
PostCommaA
CFilterCoefficientsNet::s_FilterAttributesNet, 692
Mcs::Usb, 63
PostCommaB
CFilterCoefficientsNet::s_FilterAttributesNet, 692
Mcs::Usb, 63
PowerChip
CCMOSMea_FunctionNet, 106
PowerHS
CSCUFunctionNet, 503
PP_Pump_Mode_Type_EnumNet
Mcs::Usb, 73
PPC
Mcs::Usb, 75
PPCFunction
CPPCDeviceNet, 413
PPS2
Mcs::Usb, 75
PPS4plus1
Mcs::Usb, 75
PPS5
Mcs::Usb, 75
PPS5_DIG
Mcs::Usb, 75
PPS_Function
CPPS_DeviceNet, 422
PreCommaA
CFilterCoefficientsNet::s_FilterAttributesNet, 692
Mcs::Usb, 63
PreCommaB
CFilterCoefficientsNet::s_FilterAttributesNet, 692
Mcs::Usb, 63
PrepareAndAppendData
CStg200xDownloadNet, 551
CStimulusFunctionNet, 561
PrepareAndSendData
CStg200xDownloadNet, 552
CStimulusFunctionNet, 562
PrepareChannelData
CDigOutStimulatorFunctionNet, 125
PrepareData
CStimulusFunctionNet, 563
CW2100_StimulatorFunctionNet, 608
PrepareDataSync
CW2100_StimulatorFunctionNet, 608
Product
CMcsUsbListEntryNet, 304
ProductIdEnumNet
Mcs::Usb, 73
Program
CProgramPressureCurveNet, 429
PT100
Mcs::Usb, 83
PT1000
Mcs::Usb, 83
PulseGenerator
CW2100_FunctionNet, 603
Mcs::Usb, 61, 70, 79, 83, 89
PulseGenerator_Mode_EnumNet
Mcs::Usb, 76
PumpOff
CRoboFluidDeviceNet, 478
PumpOn
CRoboFluidDeviceNet, 478

QueryTriggerstatus
CStg200xDownloadNet, 553

raGate
Mcs::Usb, 77
raIgnore
Mcs::Usb, 77
RampStart
COctoPotDeviceNet, 388
raRestart
Mcs::Usb, 77
raSingle
Mcs::Usb, 77
raStop
Mcs::Usb, 77
rawdata
Mcs::Usb, 82
RC
Mcs::Usb, 64
rc100mAh
Mcs::Usb, 67

rc200mAh
Mcs::Usb, 67
rc300mAh
Mcs::Usb, 67
rc30mAh
Mcs::Usb, 67
rcGreater300mAh
Mcs::Usb, 67
Read
CExternDTesterDeviceNet, 128
Read2
CExternDTesterDeviceNet, 128
ReadBlockFromFlash
CMcsUsbFactoryNet, 295
ReadBlockFromIFBGlobalEEprom
CMcsUsbFactoryNet, 295
ReadBlockFromNVMEM
CMcsUsbFactoryNet, 295
ReadClipping
CLIH3DeviceNet, 194
ReadEepromRegisterPreconfig
CMcsUsbNet, 321
ReadPipe
CGenericDevelopDeviceNet, 158
ReadRegister
CMcsUsbNet, 321
ReadRegister32
CMcsUsbNet, 321
ReadRegisterTimeSlot
CMcsUsbNet, 321
ReadUARTData
CLIH3DeviceNet, 194
Receive
CSerialPortNet, 507
ReceiveString
CSerialPortNet, 507
Rectangle
Mcs::Usb, 84
Ref16
Mcs::Usb, 77
Ref24
Mcs::Usb, 77
Ref32
Mcs::Usb, 77
Ref8
Mcs::Usb, 77
Reference
Mcs::Usb, 77
ReferenceElectrodeModeEnumNet
Mcs::Usb, 76
ReferenceElectrodeSwitchPositionEnumNet
Mcs::Usb, 77
RegisterHigh
Mcs::Usb, 60
RegisterLow
Mcs::Usb, 60
Regular
Mcs::Usb, 83
RemoveSoftwareKey
CMcsUsbNet, 322
Renesas
Mcs::Usb, 86
RescanHeadstage
CMcsUsbNet, 322
Reserved1
Mcs::Usb, 83
Reserved2
Mcs::Usb, 83
Reserved3
Mcs::Usb, 83
Reserved4
Mcs::Usb, 83
Reserved5
Mcs::Usb, 83
ResetAdcOffset
COctoPotDeviceNet, 388
ResetChannelmap
CWClassicFunctionNet, 659
ResetDacOffset
COctoPotDeviceNet, 388
ResetHighpassFilter
CFilterConfigurationNet, 132
ResetPipe
CGenericDevelopDeviceNet, 158
ResetStatus
CStg200xDownloadBasicNet, 545
Retina_LED
Mcs::Usb, 74
RetriggerActionEnumNet
Mcs::Usb, 77
RFFunction
CPositionIIDeviceNet, 409
Rising
Mcs::Usb, 61
RoboCurrentModeEnumNet
Mcs::Usb, 77
RoboDacq
CHiClampDeviceNet, 177
RoboDevice
CSafeISDeviceNet, 490
RoboError_AnotherMaster
CRoboDeviceNet, 472
RoboError_Base
CRoboDeviceNet, 473
RoboError_CannotEscapeEndSwitch
CRoboDeviceNet, 473
RoboError_CommandAlreadyInProgress
CRoboDeviceNet, 473
RoboError_CommandNotPossible
CRoboDeviceNet, 473
RoboError_CommunicationTimeout
CRoboDeviceNet, 473
RoboError_DacqNotReady
CRoboDeviceNet, 473
RoboError_DLLMovementTimeout
CRoboDeviceNet, 473

RoboError_FindReferenceMethod
CRoboDeviceNet, 473
RoboError_GilsonCommandPending
CRoboDeviceNet, 473
RoboError_GilsonTimeout
CRoboDeviceNet, 473
RoboError_GilsonWrondID
CRoboDeviceNet, 474
RoboError_McsBus_UnknownCommand
CRoboDeviceNet, 474
RoboError_NoEndSwitch
CRoboDeviceNet, 474
RoboError_NoMoreData
CRoboDeviceNet, 474
RoboError_NoReference
CRoboDeviceNet, 474
RoboError_NoSpeedOrAcceleration
CRoboDeviceNet, 474
RoboError_OverPressure
CRoboDeviceNet, 474
RoboError_ParameterNotAllowed
CRoboDeviceNet, 474
RoboError_PeristalticTimeout
CRoboDeviceNet, 474
RoboError_Phase0OutOfRange
CRoboDeviceNet, 474
RoboError_PollLoopCanceled
CRoboDeviceNet, 475
RoboError_PollLoopCanceledAndStopMovement
CRoboDeviceNet, 475
RoboError_Pressure
CRoboDeviceNet, 475
RoboError_RangeExceeded
CRoboDeviceNet, 475
RoboError_StateChangeNotPossible
CRoboDeviceNet, 475
RoboError_Timeout
CRoboDeviceNet, 475
RoboError_UnknownCommand
CRoboDeviceNet, 475
RoboInject
Mcs::Usb, 76
RoboMainLowLevelCommand
CRoboDeviceNet, 476
RoboMainStatorLowLevelCommand
CRoboStatorDeviceNet, 487
Robooycte2
Mcs::Usb, 76
RoboStatusEvent
CRoboDeviceNet, 476
RoboStatusEventDelegate
Mcs::Usb, 91
Running
Mcs::Usb, 81
RunTable
CRoboDacqNet, 454

s_FilterAttributesNet
CFilterCoefficientsNet::s_FilterAttributesNet, 691
SafeIS
Mcs::Usb, 76
SampleDstSize16
Mcs::Usb, 78
SampleDstSize32
Mcs::Usb, 78
SampleDstSizeNet
Mcs::Usb, 78
SampleRate
CCreateFilterNet, 112
Samplerate
CMcsUsbDacqNet, 287
SampleSize16Signed
Mcs::Usb, 78
SampleSize16Unsigned
Mcs::Usb, 78
SampleSize24Signed
Mcs::Usb, 78
SampleSize24Unsigned
Mcs::Usb, 78
SampleSize32Signed
Mcs::Usb, 78
SampleSize32Unsigned
Mcs::Usb, 78
SampleSize64Signed
Mcs::Usb, 78
SampleSize64Unsigned
Mcs::Usb, 78
SampleSizeNet
Mcs::Usb, 78
SBSVector1
Mcs::Usb, 55
SBSVector2
Mcs::Usb, 55
SBSVector3
Mcs::Usb, 55
SBSVector4
Mcs::Usb, 55
Scale
CCreateFilterNet, 112
ScanForHeadstages
CWClassicFunctionNet, 659
SCU1ElectrodeGroupHS1
Mcs::Usb, 79
SCU1ElectrodeGroupHS2
Mcs::Usb, 79
SCU1ElectrodeGroupHS3
Mcs::Usb, 79
SCU1ElectrodeGroupHS4
Mcs::Usb, 79
SCU2ElectrodeGroupHS1
Mcs::Usb, 79
SCU2ElectrodeGroupHS2
Mcs::Usb, 79
SCU2ElectrodeGroupHS3
Mcs::Usb, 79
SCU2ElectrodeGroupHS4
Mcs::Usb, 79

SCU_HeadstageIdEnumNet
Mcs::Usb, 78
SCUDacqGroupChannelEnumNet
Mcs::Usb, 79
SCUDigitalSourceEnumNet
Mcs::Usb, 79
SelectHeadstage
CW2100_FunctionNet, 602
SelectTimeSlot
CW2100_StimulatorFunctionNet, 608
Send
CSerialPortNet, 507
SendBuffered
CGilsonDeviceNet, 165
SendChannelData
CDigOutStimulatorFunctionNet, 125
CStg200xDownloadBasicNet, 545
SendCommand
CLIH3DeviceNet, 194
SendImmediate
CGilsonDeviceNet, 165
SendImmediateGetResponse
CGilsonDeviceNet, 165
SendMultiplexedData
CStimulusFunctionNet, 563
SendPreparedData
CStimulusFunctionNet, 563
CW2100_StimulatorFunctionNet, 608
SendSegmentDefine
CStg200xDownloadNet, 553
SendSegmentSelect
CStg200xDownloadNet, 553
SendSegmentStart
CStg200xDownloadNet, 554
SendStart
CStg200xBasicNet, 525
CStimulusFunctionNet, 563
CW2100_StimulatorFunctionNet, 609
SendStartDacq
CMcsUsbDacqNet, 270
SendStartStgAndDacq
CMcsUsbDacqNet, 270
SendStop
CStg200xBasicNet, 525
CStimulusFunctionNet, 564
CW2100_StimulatorFunctionNet, 609
SendStopDacq
CMcsUsbDacqNet, 271
SendStopStgAndDacq
CMcsUsbDacqNet, 271
SendStopStgAndDacqWithOptions
CMcsUsbDacqNet, 271
SendSyncData
CStg200xDownloadBasicNet, 545
Sensor
CFYIDeviceNet, 144
CMeasureTableDeviceNet, 360
CPatchServerDeviceNet, 394
SerialNumber
CMcsUsbListEntryNet, 304
CMcsUsbNet, 329
SerialPort
CHLADeviceNet, 179
Set4ADCCatchampAverageShift
CMcsBus_SensorNet, 227
Set4ADCMode
CMcsBus_SensorNet, 227
Set4DAC
CMcsBus_SensorNet, 227
Set_Values
CNF_GenDeviceNet, 385
CPathIdentDeviceNet, 395
SetAbsMaxCurrentInMicroAmp
CMultiwellOptoStimFunctionNet, 383
SetAccelerationI
CRoboStatorDeviceNet, 485
SetAccelerationNativeI
CRoboStatorDeviceNet, 485
SetAccelerationNativeXY
CRoboStatorDeviceNet, 485
SetAccelerationNativeZ
CRoboStatorDeviceNet, 485
SetAccelerationXY
CRoboStatorDeviceNet, 485
SetAccelerationZ
CRoboStatorDeviceNet, 485
SetAccelGyroDesiredRate
CW2100_FunctionNet, 602
SetAccelGyroEnabled
CW2100_FunctionNet, 602
SetAccelRange
CW2100_FunctionNet, 602
SetActiveRunningTableNumber
CWarnerValveControllerDeviceNet, 644
SetADC
CWarnerValveControllerDeviceTesterFunctionNet, 655
SetAdcChannels
CSafeISDeviceNet, 489
SetADCInputOffset
CCMOSMea_FunctionNet, 106
SetAdcOffset
CLIH3DeviceNet, 194
COctoPotDeviceNet, 388
SetAdcOffsetPermanent
CLIH3DeviceNet, 195
SetAdcSamplePos
CSafeISDeviceNet, 489
SetAirpressureLimit
CRoboDeviceNet, 469
SetAirValve
CRoboDeviceNet, 469
SetAllDigout
CRoboDacqNet, 454
SetAmplificationSwitch
COctoPotDeviceNet, 388

SetAmplitude
CChannelTestDeviceNet, 96
SetAmplitude_nA
CTEERFunctionNet, 589
SetAnalogOutADCRange
CSCUFunctionNet, 504
SetAnalogOutChannel
CW2100_FunctionNet, 602
SetAnalogOutChannels
CSCUFunctionNet, 504
SetAnalogOutDACRange
CSCUFunctionNet, 504
SetAnalogOutFilter
CW2100_FunctionNet, 602
SetAnalogThresholdHigh
CWarnerValveControllerDeviceNet, 644
SetAnalogThresholdLow
CWarnerValveControllerDeviceNet, 645
SetAnalogVoltageRange
CPPCFunctionNet, 420
SetAnalogVoltages
CPPS_FunctionNet, 425
SetAttenuation
CChannelTestDeviceNet, 96
SetAudioChannels
CMeaAudioFunctionNet, 333, 334
CW2100_FunctionNet, 602
SetAudioOutDacParameter
CLIH3DeviceNet, 195
SetAutocalibrationDisabled
CStg200xBasicNet, 526
SetAxisConfig
CRoboDeviceNet::RoboMainLowLevelCommands, 688
SetAxisLED
CRoboocyte2DeviceNet, 481
SetAxisParametersEeprom
CMcsBus_AxisParametersNet, 198, 199
SetBandwidthByIndex
CIntanMea_FunctionNet, 183
SetBaseFrequency
CRFFunctionNet, 441
SetBaseSamplerate
CCMOSMeaDeviceNet, 109
SetBath
CCMOSMea_FunctionNet, 106
SetBathclamp
COctoPotDeviceNet, 388
SetBathMode
CCMOSMea_FunctionNet, 106
SetBlankingEnable
CStg200xBasicNet, 526, 527
SetBuffer
CGenericDevelopDeviceNet, 159
SetBufferIndex
CTEERFunctionNet, 589
SetBusAddress
CMcsBusNet, 238
SetBusAddressEeprom
CMcsBusNet, 238
SetByteBuffer
CGenericDevelopDeviceNet, 159
SetCalibration
CTcxDeviceNet, 579
SetCardinalDacqSamplerate
CInterfaceboardFunctionNet, 187
SetCardinalStgOutputrate
CInterfaceboardFunctionNet, 187
SetChannel
CSw2to64DeviceNet, 567
SetChannelmap
CWClassicFunctionNet, 659
SetChannels
CSw2to64DeviceNet, 568
SetChannelSwitch
COctoPotDeviceNet, 388
SetChargingMode
CMultiBatteryChargerDeviceNet, 370
SetChargingPCoefficient
CMultiBatteryChargerDeviceNet, 370
SetCheckVoltage
COkuvisionStimulatorDeviceNet, 392
SetClampMode
CTEERFunctionNet, 590
CWarnerUssingFunctionNet, 625
SetColorRgb
CMultiwellOptoStimFunctionNet, 383
SetColorStr
CMultiwellOptoStimFunctionNet, 384
SetCommand
CMcsBusNet, 239
CPedoterDeviceNet, 396
CRoboDacqNet, 454
SetConfiguration
CMcsUsbNet, 322
SetConfigurationBit
CRoboDacqNet, 454
SetConfigurationBitAxc
CRoboDacqNet, 455
SetConfigurationBitBlu_Led
CRoboDacqNet, 455
SetConfigurationBitBlu_LedToggleFast
CRoboDacqNet, 455
SetConfigurationBitBlu_LedToggleSlow
CRoboDacqNet, 455
SetConfigurationBitCC_Gen
CRoboDacqNet, 455
SetConfigurationBitCV_Gen
CRoboDacqNet, 455
SetConfigurationBitRC_Gen
CRoboDacqNet, 455
SetConfigurationBitRed_Led
CRoboDacqNet, 455
SetConfigurationBitRed_LedSaturation
CRoboDacqNet, 455
SetConfigurationBitRed_LedToggleFast

CRoboDacqNet, 456
SetConfigurationBitRed_LedToggleSlow
CRoboDacqNet, 456
SetConfigurationBitRelais
CRoboDacqNet, 456
SetConfigurationBitRV_Gen
CRoboDacqNet, 456
SetConfigurationBitStream
CRoboDacqNet, 456
SetConfigurationBitSupply
CRoboDacqNet, 456
SetControllerParams
CTEERFunctionNet, 590
SetCrossTalkOffset
CRoboDacqNet, 456
SetCrossTalkOptimum
CRoboDacqNet, 456
SetCurrentAirvalveLimit
CRoboDeviceNet, 469
SetCurrentAndAir
CRoboDeviceNet, 469
SetCurrentAndAirXY
CRoboStatorDeviceNet, 485
SetCurrentEditTableNumber
CWarnerValveControllerDeviceNet, 645
SetCurrentEnable
CTEERFunctionNet, 590
SetCurrentFactor
COkuvisionStimulatorDeviceNet, 392
SetCurrentMode
CStg200xBasicNet, 527
SetCurrentRangeByIndex
CStg200xBasicNet, 528
SetCycles
CMeaCleanDeviceNet, 337
CMeaCoatDeviceNet, 342
SetD
CTcxDeviceNet, 579
SetDacAmplificationFactor
CStg200xBasicNet, 528
SetDacAutoControl
COctoPotDeviceNet, 388
SetDacIdleValue
CLIH3DeviceNet, 195
SetDacMode
CSafeISDeviceNet, 489
SetDACOffset
CGrapheneFunctionNet, 173
COkuvisionStimulatorDeviceNet, 392
SetDacOffset
CDacCalibrationFunctionNet, 114
CLIH3DeviceNet, 195
COctoPotDeviceNet, 388
SetDacOffsetPermanent
CLIH3DeviceNet, 196
SetDacPeriode
CSafeISDeviceNet, 489
SetDacPulseform
CSafeISDeviceNet, 489
SetDacqLegacyMode
CSCUFunctionNet, 504
SetDacRange
CW2100_FunctionNet, 603
SetDACs
CMcsBus_SensorNet, 227
SetDacUseIdleValue
CLIH3DeviceNet, 196
SetDacValue
COctoPotDeviceNet, 389
SetDataMode
CMcsUsbDacqNet, 272
SetDefault
CWarnerValveControllerDeviceNet, 645
SetDestinationSerialNumber
CMcsUsbFactoryNet, 295
SetDetectionThreshold
CMcsBus_SensorNet, 227
SetDevice
CTcxDeviceNet, 579
SetDeviceId
CUsbDeviceConfigurationFunctionNet, 594
SetDeviceList
CPositionImpDeviceNet, 411
SetDeviceName
CUsbDeviceConfigurationFunctionNet, 594
SetDeviceType
CTcxDeviceNet, 579
SetDevname
CTcxDeviceNet, 579
SetDiagnosticMode
CIntanMea_FunctionNet, 183
SetDigitalData
CMeaDigitalDataFunctionNet, 353
SetDigitalOut
CMeaDeviceNet, 348
SetDigitalOutPortInvert
CWarnerValveControllerDeviceNet, 645
SetDigitalOutPortValve
CWarnerValveControllerDeviceNet, 646
SetDigitalPortDirection
CWarnerValveControllerDeviceNet, 646
SetDigitalSource
CMcsUsbDacqNet, 272–274
SetDigitalStimulatorTrigger
CW2100_StimulatorFunctionNet, 609
SetDigitalStimulatorTriggerSlope
CW2100_StimulatorFunctionNet, 609
SetDigout
CFluidControlDeviceNet, 141
CRoboDacqNet, 456
SetDigoutMode
CStg200xBasicNet, 528
SetDigOutState
CLIH3DeviceNet, 196
SetDigoutValue
CStg200xBasicNet, 528

SetDIO
CMcsBus_FYIExtensionNet, 201
SetDischargeCurrentSetPoint
CMultiBatteryChargerDeviceNet, 371
SetDisplayMode
CWarnerValveControllerDeviceNet, 646
SetDisplayText
CRoboDacqNet, 457
SetDownsampleFactor
CRoboDacqNet, 457
SetDSPHighPassByIndex
CIntanMea_FunctionNet, 184
SetDuration
CMeaCoatDeviceNet, 342
SetEEpromPage
CLIH3DeviceNet, 196
SetElectrodeDacMux
CStg200xBasicNet, 529, 530
SetElectrodeEnable
CStg200xBasicNet, 531, 532
SetElectrodeMode
CStg200xBasicNet, 533, 534
SetEnableAmplifierProtectionSwitch
CStg200xBasicNet, 535, 536
SetEnableHeaterLimit
CTcxDeviceNet, 579
SetEnablePulse
CWarnerUssingFunctionNet, 625
SetEnableThermocouple
CTcxDeviceNet, 580
SetExternalElectrodeEnable
CStg200xBasicNet, 536, 537
SetExternalLED
CTEERFunctionNet, 590
SetFAAmplification
CStg200xBasicNet, 537
SetFilter
CRoboDacqNet, 457
SetFilterCoeffs
CRoboDacqNet, 457
SetFilterParameter
CFilterConfigurationNet, 132
CFilterConfigurationRegisterNet, 134
SetFilterParameterPermanent
CFilterConfigurationNet, 132
CFilterConfigurationRegisterNet, 134
SetFilterParametersHeadstage
CWClassicFunctionNet, 659
SetFinalDischargeVoltage
CMultiBatteryChargerDeviceNet, 371
SetFrequency
CChannelTestDeviceNet, 96
CRadioControledDevicesNet, 434
SetGain
CPgaDeviceNet, 400
SetGate
CCMOSMea_FunctionNet, 106
SetGateFloating
CCMOSMea_FunctionNet, 106
SetGateToVOP
CCMOSMea_FunctionNet, 106
SetGlobalRepeat
CDigOutStimulatorFunctionNet, 125
SetGyroRange
CW2100_FunctionNet, 603
SetHasChecksum
CWClassicFunctionNet, 660
SetHeadstage
CStg200xBasicNet, 537
SetHeadstageOnOff
CW2100_FunctionNet, 603
CWClassicFunctionNet, 660
SetHeadstagePowerStateAtStart
CSCUFunctionNet, 505
SetHeadstageSamplingActive
CW2100_FunctionNet, 603
SetHeadstageToSleep
CW2100_FunctionNet, 603
SetHeaterLimit
CTcxDeviceNet, 580
SetHighCurrentMode
CWarnerUssingFunctionNet, 625
SetHighpassFilterEnable
CFilterConfigurationNet, 133
SetHWConfig
CRoboDeviceNet::RoboMainLowLevelCommands, 688
SetHWRevision
CRoboDeviceNet::RoboMainLowLevelCommands, 688
SetHWRevisionEeprom
CMcsBusNet, 239
SetHWSelectedChannels
CWClassicFunctionNet, 660
SetI
CTcxDeviceNet, 580
SetIClamp
CRoboDacqNet, 457
SetICoeff
CRobo_FYITemp_FunctionNet, 445
SetICOffset
CRoboDacqNet, 457
SetIdleModeOffset
CWarnerUssingFunctionNet, 626
SetIGain
CRoboDacqNet, 457
SetImpedanceTestFrequency
CMeaImpedanceDeviceNet, 359
SetImpId
CPositionImpDeviceNet, 411
SetImplantCurrentSetpoint
CPositionIIDeviceNet, 406
SetInMovement
CRoboDeviceNet, 469
SetIntanRegister
CIntanMea_FunctionNet, 184

SetIntBuffer
CGenericDevelopDeviceNet, 159
SetIO
CWarnerValveControllerDeviceTesterFunctionNet, 655
SetIODirection
CWarnerValveControllerDeviceTesterFunctionNet, 656
SetIoVoltage
CInterfaceboard2FunctionNet, 185
SetLatency
CMcsBus_SensorNet, 227
SetLayoutConfiguration
CMEA2100x256FunctionNet, 331
SetLED
CRetinaLedDeviceNet, 436
SetLEDSwitch
CMcsBus_ExtensionNet, 200
SetLength
CRobo_FYIProgram_FunctionNet, 443
SetLiquidResistance
CTEERFunctionNet, 591
CWarnerUssingFunctionNet, 626
SetListmodeIndexRange
CStg200xBasicNet, 537
SetListmodeTriggerSource
CStg200xBasicNet, 538
SetLowCurrentMode
CWarnerUssingFunctionNet, 626
SetLumi
CRetinaLedDeviceNet, 436
SetMaxCurrent
CMeaCoatDeviceNet, 343
SetMaxDurationHighCurrentInMicroSec
CMultiwellOptoStimFunctionNet, 384
SetMaxDutyCycleHighCurrent
CMultiwellOptoStimFunctionNet, 384
SetMaxHeaterPowerMultiwell
CTcxDeviceNet, 580
SetMaxNoPressure
CRoboDeviceNet::RoboMainLowLevelCommands, 688
SetMaxNoPressureWaitTime
CRoboDeviceNet::RoboMainLowLevelCommands, 688
SetMaxP
CTcxDeviceNet, 580
SetMaxPower
COkuvisionStimulatorDeviceNet, 393
CRobo_FYITemp_FunctionNet, 445
SetMaxPressureWaitTime
CRoboDeviceNet::RoboMainLowLevelCommands, 688
SetMaxVoltage
CMeaCleanDeviceNet, 337
COkuvisionStimulatorDeviceNet, 393
SetMCAcceleration
CMcsBus_MotorControlNet, 211
SetMCAccelerationEeprom
CMcsBus_MotorControlNet, 212
SetMCAccelerationShortCommand
CMcsBus_MotorControlNet, 212
SetMCAxisRevisionEeprom
CMcsBus_MotorControlNet, 212
SetMCBreakCurrent
CMcsBus_MotorControlNet, 212
SetMCBreakCurrentEeprom
CMcsBus_MotorControlNet, 212
SetMCConfig
CMcsBus_MotorControlNet, 212
SetMCConfigEeprom
CMcsBus_MotorControlNet, 213
SetMCCurrent
CMcsBus_MotorControlNet, 213
SetMCCurrentEeprom
CMcsBus_MotorControlNet, 213
SetMCCurrentMode
CMcsBus_MotorControlNet, 213
SetMCCurrentModeEeprom
CMcsBus_MotorControlNet, 213
SetMCCurrentModeShortCommand
CMcsBus_MotorControlNet, 213
SetMCCurrentPosition
CMcsBus_MotorControlNet, 214
SetMCCurrentShortCommand
CMcsBus_MotorControlNet, 214
SetMCMaxAcceleration
CMcsBus_MotorControlNet, 214
SetMCMaxAccelerationEeprom
CMcsBus_MotorControlNet, 214
SetMCMaxCurrent
CMcsBus_MotorControlNet, 214
SetMCMaxCurrentEeprom
CMcsBus_MotorControlNet, 214
SetMCMaxSpeed
CMcsBus_MotorControlNet, 215
SetMCMaxSpeedEeprom
CMcsBus_MotorControlNet, 215
SetMCMaxTravel
CMcsBus_MotorControlNet, 215
SetMCMaxTravelEeprom
CMcsBus_MotorControlNet, 215
SetMCMaxTravelShortCommand
CMcsBus_MotorControlNet, 215
SetMCNewPosition
CMcsBus_MotorControlNet, 215
SetMCOutputOnOff
CMcsBus_MotorControlNet, 216
SetMCReference
CMcsBus_MotorControlNet, 216
SetMCReferenceCurrent
CMcsBus_MotorControlNet, 216
SetMCReferenceCurrentEeprom
CMcsBus_MotorControlNet, 216
SetMCRegulatorGain
CMcsBus_MotorControlNet, 216

SetMCRegulatorGainEeprom
CMcsBus_MotorControlNet, 216
SetMCRotation
CMcsBus_MotorControlNet, 217
SetMCScalingFactor
CMcsBus_MotorControlNet, 217
SetMCScalingFactorEeprom
CMcsBus_MotorControlNet, 217
SetMCSpeed
CMcsBus_MotorControlNet, 217
SetMCSpeedEeprom
CMcsBus_MotorControlNet, 217
SetMCSpeedShortCommand
CMcsBus_MotorControlNet, 217
SetMCSpeedUnitEeprom
CMcsBus_MotorControlNet, 218
SetMCStandbyCurrent
CMcsBus_MotorControlNet, 218
SetMCStandbyCurrentEeprom
CMcsBus_MotorControlNet, 218
SetMCStandbyTime
CMcsBus_MotorControlNet, 218
SetMCStandbyTimeEeprom
CMcsBus_MotorControlNet, 218
SetMeasurementMode
CStg200xBasicNet, 539
SetMinimalThreshold
CMcsBus_SensorNet, 227
SetMinNoPressureWaitTime
CRoboDeviceNet::RoboMainLowLevelCommands, 688
SetMinPressure
CRoboDeviceNet, 470
CRoboDeviceNet::RoboMainLowLevelCommands, 688
SetMinPressureWaitTime
CRoboDeviceNet::RoboMainLowLevelCommands, 688
SetMinVoltage
CMeaCleanDeviceNet, 338
SetModeSelect
CPulseGeneratorFunctionNet, 432
SetMovePump
CMcsBus_SensorNet, 228
SetMultiHeadstageMode
CW2100_FunctionNet, 603
SetNanoVoltsPerKelvin
CMcsBus_TempSensorNet, 231
SetNeurochipMemoryData
CCMOSMea_FunctionNet, 106
SetNoFilterCoeffs
CRoboDacqNet, 457
SetNumberOfAnalogChannels
CMeaDeviceNet, 348
SetNumberOfChannels
CMeaDeviceNet, 349, 350
COctoPotDeviceNet, 389
SetOffsetCurrent
CMeaCoatDeviceNet, 343
SetOnOff
CTcxDeviceNet, 580
SetOutputMap
CStg200xDownloadNet, 554
SetOutputRate
COctoPotDeviceNet, 389
CStg200xBasicNet, 539
SetP
CTcxDeviceNet, 581
SetParameter
CRoboDeviceNet::RoboMainLowLevelCommands, 689
SetPattern
CMeaSwitchDeviceNet, 362
SetPatternBool
CMeaSwitchDeviceNet, 362
SetPatternListEntry
COctoPotDeviceNet, 389
SetPauseDuration
CMeaCoatDeviceNet, 343
SetPCoeff
CRobo_FYITemp_FunctionNet, 445
SetPeriod
CPulseGeneratorFunctionNet, 432
SetPeriod_us
CTEERFunctionNet, 591
SetPermanentCurrentInMicroAmp
CMultiwellOptoStimFunctionNet, 384
SetPersistency
CRetinaLedDeviceNet, 436
SetPGain
CRoboDacqNet, 457
SetPidParameter
COctoPotDeviceNet, 389
SetPiezoState
CMcsBus_SensorNet, 228
SetPlateMux
CMultiwellDeviceNet, 378, 379
SetPlateType
CMultiwellDeviceNet, 379
SetPoti
CMcsUsbDacqNet, 274
SetPowerMuxPlate
CMultiwellDeviceNet, 379
SetPowerStrength
CPositionIIDeviceNet, 406
SetPressureOffset
CMcsBus_SensorNet, 228
CPPCFunctionNet, 420
SetPressureRange
CPPCFunctionNet, 420
SetPulse
CWarnerUssingFunctionNet, 626
SetPulseform
COkuvisionStimulatorDeviceNet, 393
SetPulseLength
CPulseGeneratorFunctionNet, 432

SetPumpCouple
CPPS_FunctionNet, 426
SetPumpEnableSpeedRatio
CPPS_FunctionNet, 426
SetPumpFastOnOff
CPPS_FunctionNet, 426
SetPumpFastSpeed
CPPS_FunctionNet, 426
SetPumpFunctionSpeeds
CPPS_FunctionNet, 426
SetPumpManualOnOff
CPPS_FunctionNet, 426
SetPumpMaxSpeed
CPPS_FunctionNet, 426
SetPumpModeType
CPPCFunctionNet, 421
CPPS_FunctionNet, 426
SetPumpSpeed
CRoboFluidDeviceNet, 478
SetPumpSpeedRatio
CPPS_FunctionNet, 427
SetPumpSpeedUnit
CPPCFunctionNet, 421
CPPS_FunctionNet, 427
SetPWM
CFluidControlDeviceNet, 141
SetRampParameter
COctoPotDeviceNet, 389
SetRatedCapacity
CMultiBatteryChargerDeviceNet, 371
SetRatedCapacityVolatile
CMultiBatteryChargerDeviceNet, 371
SetRecordingNumber
CRoboDacqNet, 458
SetReferenceElectrodeMode
CSCUFunctionNet, 505
SetReferenceElectrodeSwitchState
CSCUFunctionNet, 505
SetRegionOfInterests
CCMOSMeaDeviceNet, 110
SetRegulationTimeouts
CMcsBus_SensorNet, 228
SetRegulatorFactor
CMcsBus_SensorNet, 228
SetRegulatorOnOff
CMcsBus_SensorNet, 228
CRobo_FYITemp_FunctionNet, 445
SetRepeat
CRetinaLedDeviceNet, 437
SetRepeats
CProgramPressureCurveNet, 429
SetResetFilter
CWClassicFunctionNet, 660
SetRFFrequency
CPositionImpDeviceNet, 412
SetRFFrequencyHeadstage
CWClassicFunctionNet, 660
SetRFFrequencyReceiver
CWClassicFunctionNet, 660
SetRFFrequencyReceiverEeprom
CWClassicFunctionNet, 660
SetRFLostBehaviour
CWClassicFunctionNet, 660
SetRFPower
CWClassicFunctionNet, 661
SetRotatePump
CMcsBus_SensorNet, 228
SetRTC
COkuvisionStimulatorDeviceNet, 393
CPositionIIDeviceNet, 406
SetSampleInterval
CLIH3DeviceNet, 197
SetSamplePeriode
CMcsBus_SensorNet, 229
SetSamplerate
CMcsUsbDacqNet, 274
SetScreen
CRoboDacqNet, 458
SetSearchReferenceFastAccel
CRoboDeviceNet::RoboMainLowLevelCommands, 689
SetSearchReferenceFastSpeed
CRoboDeviceNet::RoboMainLowLevelCommands, 689
SetSearchReferenceFineAccel
CRoboDeviceNet::RoboMainLowLevelCommands, 689
SetSearchReferenceFineSpeed
CRoboDeviceNet::RoboMainLowLevelCommands, 689
SetSearchReferenceMethod
CRoboDeviceNet::RoboMainLowLevelCommands, 689
SetSearchReferenceMoveOut
CRoboDeviceNet::RoboMainLowLevelCommands, 690
SetSearchReferenceOffsetPos
CRoboDeviceNet::RoboMainLowLevelCommands, 690
SetSelectedChannels
CMcsUsbDacqNet, 275–277
CW2100_FunctionNet, 603
SetSelectedChannelsQueue
CMcsUsbDacqNet, 277–279
SetSelectedData
CMcsUsbDacqNet, 279–281
SetSelectedHeadstage
CWClassicFunctionNet, 661
SetSensorType
CTcxDeviceNet, 581
SetSerialNumberHeadstage
CWClassicFunctionNet, 661
SetSetpoint
CTcxDeviceNet, 581
SetShortBuffer
CGenericDevelopDeviceNet, 160

SetSimulation
CRoboDacqNet, 458
SetSineParameter
COctoPotDeviceNet, 389
SetSingleHeater
CMcsBus_FYIExtensionNet, 201
SetSingleValve
CFluidControlDeviceNet, 141
CRoboFluidDeviceNet, 478
SetSlope
CMeaCleanDeviceNet, 338
CMeaCoatDeviceNet, 343
SetSoftwareKey
CMcsUsbNet, 322
SetSollPressure
CMcsBus_SensorNet, 229
SetSollTemp
CRobo_FYITemp_FunctionNet, 445
SetSourceBulk
CCMOSMea_FunctionNet, 106
SetSourceDrain
CCMOSMea_FunctionNet, 107
SetSourceGate
CCMOSMea_FunctionNet, 107
SetSpeedI
CRoboStatorDeviceNet, 486
SetSpeedNativeI
CRoboStatorDeviceNet, 486
SetSpeedNativeXY
CRoboStatorDeviceNet, 486
SetSpeedNativeZ
CRoboStatorDeviceNet, 486
SetSpeedXY
CRoboStatorDeviceNet, 486
SetSpeedZ
CRoboStatorDeviceNet, 486
SetStartTriggerSlope
CDigOutStimulatorFunctionNet, 126
SetStateDebugData
CPositionIIDeviceNet, 408
SetStateEventData
CPositionIIDeviceNet, 408
SetStgProgramInfo
CStg200xBasicNet, 539
SetStimulusSites
CCMOSMea_FunctionNet, 107
SetStopTriggerSlope
CDigOutStimulatorFunctionNet, 126
SetStringFormat
CMcsUsbListEntryNet, 303
CMcsUsbListNet, 307
SetSubChannel
CMcsBus_MotorControlNet, 218
SetSwitches
CSafeISDeviceNet, 490
SetSyncoutMap
CStg200xBasicNet, 540
SetTableName
CWarnerValveControllerDeviceNet, 646
SetTablepointer
CRetinaLedDeviceNet, 437
SetTableStep
CWarnerValveControllerDeviceNet, 647
SetTableStepAll
CWarnerValveControllerDeviceNet, 647
SetTestMode
CRFFunctionNet, 441
SetThermocoupleNanovoltPerKelvin
CFluidControlDeviceNet, 143
CTcxDeviceNet, 581
SetThermoOffset
CMcsBus_TempSensorNet, 231
Settings
Mcs::Usb, 90
SetTransformer
CMeFunctionNet, 364
SetTrigger
CRetinaLedDeviceNet, 437
CWarnerValveControllerDeviceTesterFunctionNet, 656
SetTriggerMaskValue
CMeaDeviceNet, 350
CRoboDacqNet, 458
SetTriggerPeriod
CMeaDeviceNet, 351
SetTriggerSource
CStg200xBasicNet, 540
SetTriggerSyncDirection
CWarnerValveControllerDeviceTesterFunctionNet, 656
SetUByteBuffer
CGenericDevelopDeviceNet, 160
SetUClamp
CRoboDacqNet, 458
SetUCOffset
CRoboDacqNet, 458
SetUIntBuffer
CGenericDevelopDeviceNet, 161
SetupGroupDacqQueue
CMcsUsbDacqNet, 281
SetupRetriggerMode
CStg200xDownloadBasicNet, 546
SetupTrigger
CStg200xDownloadBasicNet, 547
CStimulusFunctionNet, 564
SetupTriggerSingle
CStg200xDownloadBasicNet, 547
CStimulusFunctionNet, 565
SetUseBubble
CPPS_FunctionNet, 427
SetUserParameter
CRoboDeviceNet::RoboMainLowLevelCommands, 690
SetUShortBuffer
CGenericDevelopDeviceNet, 162
SetUVOffset

CRoboDacqNet, 458
SetValue
CGenericDevelopDeviceNet, 162
SetValve
CFluidControlDeviceNet, 143
CRoboFluidDeviceNet, 478
SetValve1
CRobo_FYIProgram_FunctionNet, 443
SetValve2
CRobo_FYIProgram_FunctionNet, 443
SetValveActive
CPPCFunctionNet, 421
CWarnerValveControllerDeviceNet, 647
SetValveCurrent
CWarnerValveControllerDeviceNet, 647
SetValveDigitalInInvert
CWarnerValveControllerDeviceNet, 648
SetValveDigitalInPort
CWarnerValveControllerDeviceNet, 648
SetValveLedOn
CWarnerValveControllerDeviceNet, 648
SetValveManualGroup
CWarnerValveControllerDeviceNet, 648
SetValveManualState
CWarnerValveControllerDeviceNet, 649
SetValveMode
CWarnerValveControllerDeviceNet, 649
SetValves
CMcsBus_FYIExtensionNet, 201
SetValvesActiveMap
CWarnerValveControllerDeviceNet, 649
SetValvesManualStateMap
CWarnerValveControllerDeviceNet, 649
SetValveTableEntry
CWarnerValveControllerDeviceNet, 649
SetVds
CGrapheneFunctionNet, 173
SetVdVs
CGrapheneFunctionNet, 174
SetVdVsDAC
CGrapheneFunctionNet, 174, 175
SetVelocityI
CRoboStatorDeviceNet, 486
SetVelocityXY
CRoboStatorDeviceNet, 486
SetVelocityZ
CRoboStatorDeviceNet, 486
SetVgs
CGrapheneFunctionNet, 175
SetVMMaxNegativeCurrent
CMcsBus_VoltageModeNet, 234
SetVMMaxNegativeCurrentEeprom
CMcsBus_VoltageModeNet, 234
SetVMMaxNegativeVoltage
CMcsBus_VoltageModeNet, 234
SetVMMaxNegativeVoltageEeprom
CMcsBus_VoltageModeNet, 234
SetVMMaxPositiveCurrent
CMcsBus_VoltageModeNet, 235
SetVMMaxPositiveCurrentEeprom
CMcsBus_VoltageModeNet, 235
SetVMMaxPositiveVoltage
CMcsBus_VoltageModeNet, 235
SetVMMaxPositiveVoltageEeprom
CMcsBus_VoltageModeNet, 235
SetVMOutputOnOff
CMcsBus_VoltageModeNet, 235
SetVMVoltage
CMcsBus_VoltageModeNet, 235
SetVoltage12VLimit
CRoboDeviceNet, 470
SetVoltage5VLimit
CRoboDeviceNet, 470
SetVoltageAirvalveLimit
CRoboDeviceNet, 470
SetVoltageClampControllerParam_D
CWarnerUssingFunctionNet, 627
SetVoltageClampControllerParam_I
CWarnerUssingFunctionNet, 627
SetVoltageClampControllerParam_P
CWarnerUssingFunctionNet, 627
SetVoltageMode
CStg200xBasicNet, 540
SetVoltageRange
CGrapheneFunctionNet, 175, 176
SetVoltageRangeByIndex
CMcsUsbDacqNet, 281
SetVoltageRangeInMicroVolt
CMcsUsbDacqNet, 282
SetVoltageResolution
CGrapheneFunctionNet, 176
SetVoltageRs485ALimit
CRoboDeviceNet, 470
SetVoltageRs485BLimit
CRoboDeviceNet, 470
SetVoltageValvesLimit
CRoboDeviceNet, 470
SetWaveform
CChannelTestDeviceNet, 96
CTEERFunctionNet, 591
SetWaveLengthInNanometer
CMultiwellOptoStimFunctionNet, 385
SetWorkingFrequency
CRFFunctionNet, 441
SetWPADebugMode
CWClassicFunctionNet, 661
SetWPAType
CWClassicFunctionNet, 661
SetXGain
CRoboDacqNet, 458
Sideband
CStimulusFunctionNet::SidebandData, 693
SidebandData
CStimulusFunctionNet::SidebandData, 693
Signed_16bit
Mcs::Usb, 58

Signed_24bit
Mcs::Usb, 58
Signed_32bit
Mcs::Usb, 58
Sine
Mcs::Usb, 84
SineStart
COctoPotDeviceNet, 390
SingleWell
Mcs::Usb, 82
SixWell
Mcs::Usb, 82
size
DigitalSource< digitalsourceenum >, 664
DigitalSourceGeneral, 666
SmartImplant
Mcs::Usb, 76
SN
HeadStageIDType, 678
SOFAndCTRLword
Mcs::Usb, 56
Software
Mcs::Usb, 64
SoftwareDongle
Mcs::Usb, 75
Source
DigitalSource< digitalsourceenum >, 665
DigitalSourceGeneral, 666
Standby
Mcs::Usb, 77, 87
Start
CMeaCleanDeviceNet, 338
CMeaCoatDeviceNet, 343
CRobo_FYIProgram_FunctionNet, 443
Mcs::Usb, 61
StartDacq
CMcsUsbDacqNet, 282, 283
StartInternalCalibration
CTEERFunctionNet, 591
StartLoop
CMcsUsbDacqNet, 284, 285
StartMCMovement
CMcsBus_MotorControlNet, 219
StartMeasurement
CMeaImpedanceDeviceNet, 359
StartPoll
CStimulusFunctionNet, 565
CW2100_StimulatorFunctionNet, 609
StartQueue
CRoboDeviceNet, 470
StartSampling
CTEERFunctionNet, 591
StartSync
CMcsBus_SensorNet, 229
State
HeadStageIDTypeState, 680
Mcs::Usb, 87
Status
CUsbExceptionNet, 595
Status_AlreadyConfigured
CMcsUsbNet, 324
Status_BadStartFrame
CMcsUsbNet, 325
Status_Btstuff
CMcsUsbNet, 325
Status_BufferOverrun
CMcsUsbNet, 325
Status_BufferUnderrun
CMcsUsbNet, 325
Status_Canceled
CMcsUsbNet, 325
Status_Canceling
CMcsUsbNet, 325
Status_ConnectedPipes
CMcsUsbNet, 325
Status_ControlNotOwned
CMcsUsbNet, 325
Status_Crc
CMcsUsbNet, 325
Status_DataOverrun
CMcsUsbNet, 325
Status_DataToggleMismatch
CMcsUsbNet, 325
Status_DataUnderrun
CMcsUsbNet, 326
Status_DeviceLocked
CMcsUsbNet, 326
Status_DeviceNotFound
CMcsUsbNet, 326
Status_DeviceRemoved
CMcsUsbNet, 326
Status_DevNotResponding
CMcsUsbNet, 326
Status_EndpointHalted
CMcsUsbNet, 326
Status_ErrorBusy
CMcsUsbNet, 326
Status_ErrorShortTransfer
CMcsUsbNet, 326
Status_Fifo
CMcsUsbNet, 326
Status_FrameControlOwned
CMcsUsbNet, 326
Status_InternalHcError
CMcsUsbNet, 326
Status_InvalidDeviceHandle
CMcsUsbNet, 327
Status_InvalidHandle
CMcsUsbNet, 327
Status_InvalidParameter
CMcsUsbNet, 327
Status_InvalidPipeHandle
CMcsUsbNet, 327
Status_InvalidUrbFunction
CMcsUsbNet, 327
Status_IoPending

CMcsUsbNet, 327
Status_IoTimeout
CMcsUsbNet, 327
Status_IsochRequestFailed
CMcsUsbNet, 327
Status_LastUsbErrorMismatch
CMcsUsbNet, 327
Status_NoBandwidth
CMcsUsbNet, 327
Status_NoMemory
CMcsUsbNet, 328
Status_NoSuchDevice
CMcsUsbNet, 328
Status_NotAccessed
CMcsUsbNet, 328
Status_NotSupported
CMcsUsbNet, 328
Status_PidCheckFailure
CMcsUsbNet, 328
Status_PipeNotLinked
CMcsUsbNet, 328
Status_RequestFailed
CMcsUsbNet, 328
Status_RequestMutexFailed
CMcsUsbNet, 328
Status_RequestMutexTimeout
CMcsUsbNet, 328
Status_Stall
CMcsUsbNet, 328
Status_Unconfigured
CMcsUsbNet, 328
Status_UnexpectedPid
CMcsUsbNet, 329
STG
Mcs::Usb, 74
Stg1
Mcs::Usb, 62
STG1DACSignalGroup
Mcs::Usb, 57, 69, 79
STG1SidebandsGroup
Mcs::Usb, 57, 69, 79
STG1TriggerStatusGroup
Mcs::Usb, 57, 69, 79
Stg2
Mcs::Usb, 62
Stg200xDigoutModeEnumNet
Mcs::Usb, 81
Stg200xPollStatusEvent
CStg200xDownloadNet, 555
Stg200xSegmentFlagsEnumNet
Mcs::Usb, 81
Stg200xTriggerStatusEnumNet
Mcs::Usb, 81
STG2DACSignalGroup
Mcs::Usb, 69, 79
STG2SidebandsGroup
Mcs::Usb, 69, 79
STG2TriggerStatusGroup
Mcs::Usb, 69, 79
Stg3
Mcs::Usb, 62
STG3008_FA
Mcs::Usb, 74
STG4002
Mcs::Usb, 74
STG4002_opto
Mcs::Usb, 74
STG4004
Mcs::Usb, 74
STG4004_opto
Mcs::Usb, 74
STG4008
Mcs::Usb, 74
STG4008_opto
Mcs::Usb, 74
STG400x
Mcs::Usb, 74
STG400x_opto
Mcs::Usb, 74
STG5
Mcs::Usb, 74
STG_DestinationEnumNet
Mcs::Usb, 82
StgListModeTrigger
Mcs::Usb, 62
StgStatusNet, 694
FromIntPtr, 694
FromPtr, 694
ListOfChangedTriggers, 694
TiggerStatus, 694
StgTrigger
Mcs::Usb, 62
StillConnected
CRadioControledDevicesNet, 434
Stimulation
Mcs::Usb, 54
StimulationLayoutConfigurationEnumNet
Mcs::Usb, 82
Stimulator
CW2100_FunctionNet, 604
Stimulus
CCMOSMeaDeviceNet, 110
CStg200xDownloadBasicNet, 548
StimulusDeviceDataAndUnrolledData
CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, 695
StimulusFunction
CLIH3DeviceNet, 197
StimulusParameters
HeadStageIDType, 678
Stop
CMeaCleanDeviceNet, 338
CMeaCoatDeviceNet, 344
Mcs::Usb, 61, 72
StopDacq
CMcsUsbDacqNet, 286

StopLoop
CMcsUsbDacqNet, 286
StopMCMovement
CMcsBus_MotorControlNet, 219
StopMovement
CRoboDeviceNet, 471
StopMovementI
CRoboStatorDeviceNet, 487
StopMovementXY
CRoboStatorDeviceNet, 487
StopMovementZ
CRoboStatorDeviceNet, 487
StopPlateClamp
CMultiwellDeviceNet, 380
StopPoll
CStimulusFunctionNet, 565
CW2100_StimulatorFunctionNet, 609
StopSampling
CTEERFunctionNet, 592
StopTable
CRoboDacqNet, 458, 459
StorageCharge
Mcs::Usb, 67
StoreValveTable
CWarnerValveControllerDeviceNet, 650
SubtractFromAll
Mcs::Usb, 77
SubtractFromAllOther
Mcs::Usb, 77
SubtractFromReferenceElectrodeOnly
Mcs::Usb, 77
SubtractionOff
Mcs::Usb, 77
SuperSpeed
Mcs::Usb, 69
Sw2to64
Mcs::Usb, 75
SwitchOnOff
CPositionIIDeviceNet, 408
SYNC_BIT0
CW2100_StimulatorFunctionNet, 610
SYNC_BIT1
CW2100_StimulatorFunctionNet, 610
SYNCOUT1
Mcs::Usb, 81
SYNCOUT2
Mcs::Usb, 81
SYNCOUT3
Mcs::Usb, 81
SYNCOUT4
Mcs::Usb, 81
SYNCOUT5
Mcs::Usb, 81
SYNCOUT6
Mcs::Usb, 81
SYNCOUT7
Mcs::Usb, 81
SYNCOUT8
Mcs::Usb, 81
syncoutdata
Mcs::Usb, 82
SyncStart
Mcs::Usb, 81

Table
Mcs::Usb, 90
Table_Wait
CRoboDacqNet, 459
TableDefBegin
CRoboDacqNet, 459
TableDefEnd
CRoboDacqNet, 459
TableEntryChangedEvent
CWarnerValveControllerDeviceNet, 653
TactSwitchGetState
CMcsBus_SensorNet, 229
TactSwitchSetDisplay
CMcsBus_SensorNet, 229
TBSI_127
Mcs::Usb, 50
TBSI_15
Mcs::Usb, 50
TBSI_31
Mcs::Usb, 50
TBSI_5
Mcs::Usb, 50
TBSI_63
Mcs::Usb, 50
TBSI_Dacq
Mcs::Usb, 75
TBSI_DACQDigitalSourceEnumNet
Mcs::Usb, 82
TBSI_Reserved
Mcs::Usb, 50
TbsiDacq
Mcs::Usb, 65
TbsiDacqHeadstage
Mcs::Usb, 64
TbsiDacqInterfaceboard
Mcs::Usb, 64
TC01
Mcs::Usb, 74
TC02
Mcs::Usb, 74
TCX
Mcs::Usb, 74
TcxDeviceTypeEnumNet
Mcs::Usb, 83
TcxSensorTypeEnumNet
Mcs::Usb, 83
TeerClampModeEnumNet
Mcs::Usb, 83
TEERFunctionNet
CTEERMachineDeviceNet, 593
TeerWaveformEnumNet
Mcs::Usb, 84
Tersens

Mcs::Usb, 74
Test_ADC_EPC10
Mcs::Usb, 66
Test_DAC_EPC10
Mcs::Usb, 66
ThrowCUsbExceptionNetOnError
CMcsUsbFunctionNet, 299
CMcsUsbNet, 322
TiggerStatus
StgStatusNet, 694
TimeResolutionInNanoSeconds
W2100_StimulusParametersNet, 697
Timestamp
Mcs::Usb, 56
ToCpp
CFilterCoefficientsNet::s_FilterAttributesNet, 691
ToString
CFilterPropertyNet, 135
CMcsUsbListEntryNet, 304
HeadStageIDType, 677
HeadstageIDTypeObject, 679
TouchTest
Mcs::Usb, 90
Triggerbox_AMS
Mcs::Usb, 74
Triggerbox_AMS3
Mcs::Usb, 74
Triggerbox_IMS
Mcs::Usb, 74
Triggerbox_R5
Mcs::Usb, 74
TriggerMask_Default
CRoboDacqNet, 459
TriggerOnly
Mcs::Usb, 81
TriggerSourceEnumNet
Mcs::Usb, 84
TriggerStatus
CMcsUsbDeviceStatePushFunctionNet, 288
CMcsUsbDeviceStatePushNet, 289
TriggerStatus1
Mcs::Usb, 55
TriggerStatus2
Mcs::Usb, 55
TriggerStatus3
Mcs::Usb, 55
TriggerStatus4
Mcs::Usb, 55
TriggerValue_MoveAbs
CRoboDacqNet, 459
TriggerValue_StartQueue
CRoboDacqNet, 459
tsAuxIn1
Mcs::Usb, 85
tsAuxIn2
Mcs::Usb, 85
tsDACQCy1Dev1Runs
Mcs::Usb, 86
tsDACQCy1Dev2Runs
Mcs::Usb, 86
tsDACQCy2Dev1Runs
Mcs::Usb, 86
tsDACQCy2Dev2Runs
Mcs::Usb, 86
tsDigitalIn1
Mcs::Usb, 84
tsDigitalIn10
Mcs::Usb, 84
tsDigitalIn11
Mcs::Usb, 84
tsDigitalIn12
Mcs::Usb, 84
tsDigitalIn13
Mcs::Usb, 84
tsDigitalIn14
Mcs::Usb, 84
tsDigitalIn15
Mcs::Usb, 84
tsDigitalIn16
Mcs::Usb, 84
tsDigitalIn17
Mcs::Usb, 84
tsDigitalIn18
Mcs::Usb, 84
tsDigitalIn19
Mcs::Usb, 84
tsDigitalIn2
Mcs::Usb, 84
tsDigitalIn20
Mcs::Usb, 84
tsDigitalIn21
Mcs::Usb, 84
tsDigitalIn22
Mcs::Usb, 84
tsDigitalIn23
Mcs::Usb, 84
tsDigitalIn24
Mcs::Usb, 84
tsDigitalIn25
Mcs::Usb, 85
tsDigitalIn26
Mcs::Usb, 85
tsDigitalIn27
Mcs::Usb, 85
tsDigitalIn28
Mcs::Usb, 85
tsDigitalIn29
Mcs::Usb, 85
tsDigitalIn3
Mcs::Usb, 84
tsDigitalIn30
Mcs::Usb, 85
tsDigitalIn31
Mcs::Usb, 85
tsDigitalIn32
Mcs::Usb, 85

tsDigitalIn4
Mcs::Usb, 84
tsDigitalIn5
Mcs::Usb, 84
tsDigitalIn6
Mcs::Usb, 84
tsDigitalIn7
Mcs::Usb, 84
tsDigitalIn8
Mcs::Usb, 84
tsDigitalIn9
Mcs::Usb, 84
tsDigitalPuse0
Mcs::Usb, 85
tsDigitalPuse1
Mcs::Usb, 85
tsDigitalPuse10
Mcs::Usb, 86
tsDigitalPuse11
Mcs::Usb, 86
tsDigitalPuse12
Mcs::Usb, 86
tsDigitalPuse13
Mcs::Usb, 86
tsDigitalPuse14
Mcs::Usb, 86
tsDigitalPuse15
Mcs::Usb, 86
tsDigitalPuse16
Mcs::Usb, 86
tsDigitalPuse17
Mcs::Usb, 86
tsDigitalPuse18
Mcs::Usb, 86
tsDigitalPuse19
Mcs::Usb, 86
tsDigitalPuse2
Mcs::Usb, 85
tsDigitalPuse20
Mcs::Usb, 86
tsDigitalPuse21
Mcs::Usb, 86
tsDigitalPuse22
Mcs::Usb, 86
tsDigitalPuse23
Mcs::Usb, 86
tsDigitalPuse24
Mcs::Usb, 86
tsDigitalPuse25
Mcs::Usb, 86
tsDigitalPuse26
Mcs::Usb, 86
tsDigitalPuse27
Mcs::Usb, 86
tsDigitalPuse28
Mcs::Usb, 86
tsDigitalPuse29
Mcs::Usb, 86
tsDigitalPuse3
Mcs::Usb, 85
tsDigitalPuse30
Mcs::Usb, 86
tsDigitalPuse31
Mcs::Usb, 86
tsDigitalPuse4
Mcs::Usb, 85
tsDigitalPuse5
Mcs::Usb, 85
tsDigitalPuse6
Mcs::Usb, 85
tsDigitalPuse7
Mcs::Usb, 85
tsDigitalPuse8
Mcs::Usb, 86
tsDigitalPuse9
Mcs::Usb, 86
tsFeedback1
Mcs::Usb, 85
tsFeedback10
Mcs::Usb, 85
tsFeedback11
Mcs::Usb, 85
tsFeedback12
Mcs::Usb, 85
tsFeedback13
Mcs::Usb, 85
tsFeedback14
Mcs::Usb, 85
tsFeedback15
Mcs::Usb, 85
tsFeedback16
Mcs::Usb, 85
tsFeedback17
Mcs::Usb, 85
tsFeedback18
Mcs::Usb, 85
tsFeedback19
Mcs::Usb, 85
tsFeedback2
Mcs::Usb, 85
tsFeedback20
Mcs::Usb, 85
tsFeedback21
Mcs::Usb, 85
tsFeedback22
Mcs::Usb, 85
tsFeedback23
Mcs::Usb, 85
tsFeedback24
Mcs::Usb, 85
tsFeedback25
Mcs::Usb, 85
tsFeedback26
Mcs::Usb, 85
tsFeedback27
Mcs::Usb, 85

tsFeedback28
Mcs::Usb, 85
tsFeedback29
Mcs::Usb, 85
tsFeedback3
Mcs::Usb, 85
tsFeedback30
Mcs::Usb, 85
tsFeedback31
Mcs::Usb, 85
tsFeedback32
Mcs::Usb, 85
tsFeedback4
Mcs::Usb, 85
tsFeedback5
Mcs::Usb, 85
tsFeedback6
Mcs::Usb, 85
tsFeedback7
Mcs::Usb, 85
tsFeedback8
Mcs::Usb, 85
tsFeedback9
Mcs::Usb, 85
tsNone
Mcs::Usb, 84
tsSidebandBit8
Mcs::Usb, 86
tsTriggered
Mcs::Usb, 86
TxnGetSerialNumber
CMcsUsbNet, 322
TxnSetSerialNumber
CMcsUsbNet, 322
TxnTestMemoryReadAndCheck
CMcsUsbNet, 322
TxnTestMemoryWrite
CMcsUsbNet, 323
Type
HeadStageIDType, 678
TypeValue
HeadStageIDType, 678

Unknown
HeadStageIDType, 676
Mcs::Usb, 50, 51, 63, 64, 83, 86
unknown
Mcs::Usb, 62
UnknownDest
Mcs::Usb, 54
UnknownSpeed
Mcs::Usb, 69
Unlock
Mcs::Usb, 73
UnlockPlateClamp
CMultiwellDeviceNet, 380
UnrolledAmplitude
CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, 695
UnrolledDuration
CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, 695
UnrolledSync
CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData, 695
Unsigned_16bit
Mcs::Usb, 58
Unsigned_24bit
Mcs::Usb, 58
Unsigned_32bit
Mcs::Usb, 58
UpdateChannelBlock
CCMOSMeaDeviceNet, 110
UpdateDisplay
CRoboDacqNet, 459
UpdateFirmware
CMcsUsbFactoryNet, 295–297
UpdateTransistorVoltages
CCMOSMea_FunctionNet, 107
UpdateTrigger
Mcs::Usb, 81
USB
FirmwareDestinationNames, 675
Mcs::Usb, 51
USB_TARGET1
Mcs::Usb, 54
USB_TARGET2
Mcs::Usb, 54
USB_TARGET3
Mcs::Usb, 54
usbSetupPacket_t, 696
bmRequestType, 696
bRequest, 696
wIndex, 696
wLength, 696
wValue, 696
UsbTest
Mcs::Usb, 75
UsbVendorIdEnumNet
Mcs::Usb, 86
User_ADC_0
Mcs::Usb, 66
User_ADC_1
Mcs::Usb, 66
User_ADC_2
Mcs::Usb, 66
User_ADC_3
Mcs::Usb, 66
User_ADC_4
Mcs::Usb, 66
User_DAC_0
Mcs::Usb, 66
User_DAC_1
Mcs::Usb, 66
User_DAC_2
Mcs::Usb, 66
UserDefinedName

HeadStageIDType, 678
UssingChamber
Mcs::Usb, 65
UssingClampModeEnumNet
Mcs::Usb, 86
UssingRail
Mcs::Usb, 65
UssingUnitEnumNet
Mcs::Usb, 87

Valid
HeadStageIDType, 678
ValidKey
CMcsUsbNet, 323
VendorIdEnumNet
Mcs::Usb, 87
VendorInRequest
CGenericDevelopDeviceNet, 163
VendorOutRequest
CGenericDevelopDeviceNet, 163
VirtualDevice_ContinousDacq
CRoboDacqNet, 459
VirtualDevice_TableRun
CRoboDacqNet, 459
Volt
Mcs::Usb, 51, 87
Voltage
BatteryState, 93
Voltage_3V3
Mcs::Usb, 65
Voltage_5V0
Mcs::Usb, 65
VoltageClamp
Mcs::Usb, 87
VoltageRangeDisplayStringMilliVolt
CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet, 596
VoltageRangeInMicroVolt
CMcsUsbDacqNet::CHWInfo::CVoltageRangeInfoNet, 596
W2100_StimulusParametersNet, 697
VoltageResolutionInMicroVolt
W2100_StimulusParametersNet, 697
VoltageString
BatteryState, 93
VOPSTimerSetResetTimes
CCMOSMea_FunctionNet, 107

W16IsW14
HeadStageIDType, 678
W2100
Mcs::Usb, 75
W2100_Accel_Gyro_Select_EnumNet
Mcs::Usb, 87
W2100_FunctionNet
CMeaDeviceNet, 352
W2100_StimulusParametersNet, 696
CurrentRangeInNanoAmp, 697
CurrentResolutionInNanoAmp, 697
DACResolution, 697
TimeResolutionInNanoSeconds, 697
VoltageRangeInMicroVolt, 697
VoltageResolutionInMicroVolt, 697
W2100DacqGroupChannelEnumNet
Mcs::Usb, 87
W2100DigitalSourceEnumNet
Mcs::Usb, 89
W2100IFB2
Mcs::Usb, 65
W2100Interfaceboard
Mcs::Usb, 64
W2100WirelessReceiver
Mcs::Usb, 64, 65
W2100WirelessReceiverAnalog
Mcs::Usb, 64, 65
WaitForAllChambers
CWarnerUssingFunctionNet, 628
WaitForChamber
CWarnerUssingFunctionNet, 628
WaitTimer
CRoboDeviceNet, 471
Warner
Mcs::Usb, 83
Warner_TEER_Machine
Mcs::Usb, 76
Warner_Ussing
Mcs::Usb, 76
WARNER_USSING_DEVICE
Mcs::Usb, 59
Warner_Valve_Control
Mcs::Usb, 76
WARNER_VALVE_CONTROL_DEVICE
Mcs::Usb, 59
WarnerUssingFunction
CWarnerUssingDeviceNet, 612
WClassicFunctionNet
CMeaDeviceNet, 352
Whole_Cell_Patch
Mcs::Usb, 75
WholeCellPatch
Mcs::Usb, 78
WholeCellPatchHeadstage
Mcs::Usb, 65
wIndex
usbSetupPacket_t, 696
WirelessHeadStageAccDataRE1HS1
Mcs::Usb, 88
WirelessHeadStageAccDataRE1HS2
Mcs::Usb, 88
WirelessHeadStageAccDataRE1HS3
Mcs::Usb, 88
WirelessHeadStageAccDataRE1HS4
Mcs::Usb, 88
WirelessHeadStageAccDataRE2HS1
Mcs::Usb, 88
WirelessHeadStageAccDataRE2HS2
Mcs::Usb, 88

WirelessHeadStageAccDataRE2HS3
Mcs::Usb, 88
WirelessHeadStageAccDataRE2HS4
Mcs::Usb, 88
WirelessHeadStageAnalogRE1HS1
Mcs::Usb, 88
WirelessHeadStageAnalogRE1HS2
Mcs::Usb, 88
WirelessHeadStageAnalogRE1HS3
Mcs::Usb, 88
WirelessHeadStageAnalogRE1HS4
Mcs::Usb, 88
WirelessHeadStageAnalogRE2HS1
Mcs::Usb, 88
WirelessHeadStageAnalogRE2HS2
Mcs::Usb, 88
WirelessHeadStageAnalogRE2HS3
Mcs::Usb, 88
WirelessHeadStageAnalogRE2HS4
Mcs::Usb, 88
WirelessHeadStageGyroDataRE1HS1
Mcs::Usb, 88
WirelessHeadStageGyroDataRE1HS2
Mcs::Usb, 88
WirelessHeadStageGyroDataRE1HS3
Mcs::Usb, 88
WirelessHeadStageGyroDataRE1HS4
Mcs::Usb, 88
WirelessHeadStageGyroDataRE2HS1
Mcs::Usb, 88
WirelessHeadStageGyroDataRE2HS2
Mcs::Usb, 88
WirelessHeadStageGyroDataRE2HS3
Mcs::Usb, 88
WirelessHeadStageGyroDataRE2HS4
Mcs::Usb, 88
WirelessHeadStageOptoStimCurrentRE1HS1
Mcs::Usb, 88
WirelessHeadStageOptoStimCurrentRE1HS2
Mcs::Usb, 88
WirelessHeadStageOptoStimCurrentRE1HS3
Mcs::Usb, 88
WirelessHeadStageOptoStimCurrentRE1HS4
Mcs::Usb, 88
WirelessHeadStageOptoStimCurrentRE2HS1
Mcs::Usb, 88
WirelessHeadStageOptoStimCurrentRE2HS2
Mcs::Usb, 89
WirelessHeadStageOptoStimCurrentRE2HS3
Mcs::Usb, 89
WirelessHeadStageOptoStimCurrentRE2HS4
Mcs::Usb, 89
WirelessHeadStageReservedARE1HS1
Mcs::Usb, 88
WirelessHeadStageReservedARE1HS2
Mcs::Usb, 88
WirelessHeadStageReservedARE1HS3
Mcs::Usb, 88
WirelessHeadStageReservedARE1HS4
Mcs::Usb, 88
WirelessHeadStageReservedARE2HS1
Mcs::Usb, 89
WirelessHeadStageReservedARE2HS2
Mcs::Usb, 89
WirelessHeadStageReservedARE2HS3
Mcs::Usb, 89
WirelessHeadStageReservedARE2HS4
Mcs::Usb, 89
WirelessHeadStageReservedBRE1HS1
Mcs::Usb, 89
WirelessHeadStageReservedBRE1HS2
Mcs::Usb, 89
WirelessHeadStageReservedBRE1HS3
Mcs::Usb, 89
WirelessHeadStageReservedBRE1HS4
Mcs::Usb, 89
WirelessHeadStageReservedBRE2HS1
Mcs::Usb, 89
WirelessHeadStageReservedBRE2HS2
Mcs::Usb, 89
WirelessHeadStageReservedBRE2HS3
Mcs::Usb, 89
WirelessHeadStageReservedBRE2HS4
Mcs::Usb, 89
WirelessHeadStageReservedCRE1HS1
Mcs::Usb, 89
WirelessHeadStageReservedCRE1HS2
Mcs::Usb, 89
WirelessHeadStageReservedCRE1HS3
Mcs::Usb, 89
WirelessHeadStageReservedCRE1HS4
Mcs::Usb, 89
WirelessHeadStageReservedCRE2HS1
Mcs::Usb, 89
WirelessHeadStageReservedCRE2HS2
Mcs::Usb, 89
WirelessHeadStageReservedCRE2HS3
Mcs::Usb, 89
WirelessHeadStageReservedCRE2HS4
Mcs::Usb, 89
WirelessHeadStageStatusRE1HS1
Mcs::Usb, 88
WirelessHeadStageStatusRE1HS2
Mcs::Usb, 88
WirelessHeadStageStatusRE1HS3
Mcs::Usb, 88
WirelessHeadStageStatusRE1HS4
Mcs::Usb, 88
WirelessHeadStageStatusRE2HS1
Mcs::Usb, 88
WirelessHeadStageStatusRE2HS2
Mcs::Usb, 88
WirelessHeadStageStatusRE2HS3
Mcs::Usb, 88
WirelessHeadStageStatusRE2HS4
Mcs::Usb, 88

WirelessTestAdapter
Mcs::Usb, 50
wLength
usbSetupPacket_t, 696
Work
Mcs::Usb, 90
WPA16
Mcs::Usb, 75
WPA32
Mcs::Usb, 75
WPA4
Mcs::Usb, 75
WPA8
Mcs::Usb, 75
WPAError_ScanningIsPending
CMcsUsbNet, 329
Write
CExternDTesterDeviceNet, 128
Write2
CExternDTesterDeviceNet, 129
WriteEepromRegisterPreconfig
CMcsUsbNet, 323
WritePipe
CGenericDevelopDeviceNet, 163
WriteRegister
CMcsUsbNet, 323, 324
WriteRegister32
CMcsUsbNet, 324
WriteRegisterArray
CMcsUsbNet, 324
WriteRegisterTimeSlot
CMcsUsbNet, 324
WriteRegisterValue
CMcsUsbNet, 324
WriteUARTData
CLIH3DeviceNet, 197
wValue
usbSetupPacket_t, 696
WvcDisplayModeEnumNet
Mcs::Usb, 90
WvcValveModeEnumNet
Mcs::Usb, 90

Zero
Mcs::Usb, 60, 70, 79, 82, 89