multichannel systems

McsUsbNet.dll

Version 5.1.13

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 1.8.18

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.CIntanMea_FunctionNet
- Mcs.Usb.CInterfaceboardFunctionNet
- Mcs.Usb.CMcsBus_MotorControlNet
- Mcs.Usb.CMcsBus_VoltageModeNet
- Mcs.Usb.CMcsBus_AxisParametersNet
- Mcs.Usb.CMcsBus_SensorNet
- Mcs.Usb.CMcsBus_TempSensorNet
- Mcs.Usb.CMcsBus_ExtensionNet
- Mcs.Usb.CMcsBus_FYIExtensionNet
- Mcs.Usb.CMcsUsbDeviceStatePushFunctionNet
- Mcs.Usb.CMEA2100x256FunctionNet
- Mcs.Usb.CMeaAudioFunctionNet
- Mcs.Usb.CMeaDigitalDataFunctionNet

- Mcs.Usb.CMeaFeedbackFunctionNet
- Mcs.Usb.CMeFunctionNet
- Mcs.Usb.CMultiwellCallbackFunctionNet
- Mcs.Usb.CMultiwellOptoStimFunctionNet
- Mcs.Usb.CPPCFunctionNet
- Mcs.Usb.CPPS_FunctionNet
- Mcs.Usb.CPPS_FunctionNet
- Mcs.Usb.CPulseGeneratorFunctionNet
- Mcs.Usb.CRFFunctionNet
- Mcs.Usb.CRobo_FYITemp_FunctionNet
- Mcs.Usb.CRobo_FYIProgram_FunctionNet
- Mcs.Usb.CRobo_FYITemp_FunctionNet
- Mcs.Usb.CRobo_FYIProgram_FunctionNet
- Mcs.Usb.CSCUFunctionNet
- Mcs.Usb.CStimulusFunctionNet
- Mcs.Usb.CTEERFunctionNet
- Mcs.Usb.CW2100_FunctionNet
- Mcs.Usb.CW2100_StimulatorFunctionNet
- Mcs.Usb.CWarnerUssingFunctionNet
- Mcs.Usb.CWarnerValveControllerDeviceTesterFunctionNet
- Mcs.Usb.CWClassicFunctionNet
- Mcs.Usb.CWirelessBaseFunctionNet

4 Data ACQuisition (DACQ) Devices

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

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

The class Mcs.Usb.CMeaDeviceNet is the base class for DACQ devices.

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

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

For connecting to a DACQ device see Connecting to an MCS device.

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

```
int hwchannels;
device.HWInfo().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:

8 Hierarchical Index

8.1 Class Hierarchy

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

9 Class Index

9.1 Class List

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

10 Namespace Documentation

10.1 Mcs Namespace Reference

Namespaces

- Usb

10.2 Mcs::Usb Namespace Reference

Classes

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

- class CCreateFilterNet
- class CDacCalibrationFunctionNet
- class CDacqGroupChannelGenericSelectionNet
- class CDacqGroupChannelSelectionNet
- class CDacqGroupChannelSelectionTemplateNet
- class CDeviceGroupChannelInfoGenericNet
- class CDeviceGroupChannelInfoMEA2100_256Net
- class CDeviceGroupChannelInfoNet
- class CDeviceGroupChannelInfoSCUNet
- class CDeviceGroupChannelInfoTemplateNet
- class CDeviceGroupChannelInfoW2100Net
- class CDigOutStimulatorFunctionNet

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

  CEncapsulatorDeviceNet is the to control the MCS HiClamp device
- class CExternDTesterDeviceNet

  CExternDTesterDeviceNet is the class to access the ExternD Tester (Handheld Device Tester D)
- class CFilterCoefficientsNet
- class CFilterConfigurationNet
- class CFilterConfigurationRegisterNet
- class CFilterPropertyNet
- class CFluidControlDeviceNet

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

  CFYIDeviceNet is the class to control the MCS FYI device
- class CGenericDevelopDeviceNet

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

  CGilsonDeviceNet is the class to control a Gilson device.
- class CGrapheneFunctionNet

  CGrapheneFunctionNet is the class to control the TEER device
- class CHiClampDeviceNet

  CHiClampDeviceNet is the to control the MCS HiClamp device
- class CHLADacqNet
- class CHLADeviceNet

  CHLADeviceNet is the to control the MCS HLA device
- class CIntanMea_FunctionNet
- class CInterfaceboard2FunctionNet

  CInterfaceboard2FunctionNet is the class to control the Interfaceboard
- class CInterfaceboardFunctionNet

  CInterfaceboardFunctionNet is the class to control the Interfaceboard
- class CLIH3DeviceNet

  CLIH3DeviceNet is the class to access the HEKA LIH3 device.
- class CMcsBus_AxisParametersNet
- class CMcsBus_ExtensionNet
- class CMcsBus_FYIExtensionNet
- class CMcsBus_MotorControlNet
- class CMcsBus_SensorNet
- class CMcsBus_TempSensorNet
- class CMcsBus_VoltageModeNet
- class CMcsBusNet
- class CMcsUsbDacqNet

Base class for data acquisition devices.
- class CMcsUsbDeviceStatePushFunctionNet
- class CMcsUsbDeviceStatePushNet
- class CMcsUsbFactoryNet
- class CMcsUsbFunctionNet
- class CMcsUsbFunctionPointerContainer
- class CMcsUsbListEntryNet

  McsUsbListEntryNet identifies a connected device.
- class CMcsUsbListNet

  Class to handle a list of connected MCS USB devices.
- class CMcsUsbNet

  Base class to handle MCS USB devices. All device classes are derived from this class. Functionality that is provided by all MCS devices is handled by this class.
- class CMcsUsbPointerContainer
- class CMEA2100_256DacqGroupChannelSelectionNet
- class CMEA2100x256FunctionNet

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

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

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

  Base class for MEA data acquisition devices.
- class CMeaDigitalDataFunctionNet
- class CMeaFeedbackFunctionNet
- class CMeaImpedanceDeviceNet
- class CMeasureTableDeviceNet

  CMeasureTableDeviceNet is the to control the MCS HLA device
- class CMeaSwitchDeviceNet

  The class to control the USB-MEA-Switch.
- class CMeaUSBDeviceNet

  Class for data acquisition via ME and MEA USB amplifiers
- class CMeFunctionNet
- class CMultiBatteryChargerDeviceNet

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

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

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

  CMultiwellOptoStimFunctionNet is the class to access the optical properties of the Multiwell Optostim device
- class CNF_GenDeviceNet
- class COctoPotDeviceNet
- class COkuvisionStimulatorDeviceNet
- class CPatchServerDeviceNet

  CPatchServerDeviceNet is the class to control the MCS PatchServer device
- class CPathIdentDeviceNet
- class CPedoterDeviceNet
- class CPeristalticPumpDeviceNet

  CPeristalticPumpDeviceNet is the class to control a Persistaltic Pump.

- class CPgaDeviceNet
- class CPositionIIDeviceNet

  CPositionIIDeviceNet is the class to control PositionII devices
- class CPositionImpDeviceNet

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

  CPPCFunctionNet is the class to access the PPC (high precision Patch Peristalic patch Pump
- class CPPS_DeviceNet
- class CPPS_FunctionNet
- class CPPSDeviceNet

  CPPS4plus1DeviceNet is the to control the MCS HLA device
- class CProgramPressureCurveNet

  CProgramPressureCurveNet is the class to program pressure curves
- class CPulseGeneratorFunctionNet

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

  CRFFunctionNet is the class to control RF devices
- class CRobo_FYIProgram_FunctionNet
- class CRobo_FYITemp_FunctionNet
- class CRoboDacqNet
- class CRoboDeviceNet

  CRoboDeviceNet is the base class for all Robo platform based devices
- class CRoboFluidDeviceNet
- class CRoboInjectDeviceNet

  CRoboInjectDeviceNet is the to control the MCS RoboInject device
- class CRoboocyte2DeviceNet

  CRoboocyte2DeviceNet is the class to control the MCS Roboocyte2 device
- class CRoboStatorDeviceNet
- class CSafeISDeviceNet
- class CSCUDacqGroupChannelSelectionNet
- class CSCUFunctionNet

  CSCUFunctionNet is the class to control the SCU device
- class CSerialPortNet
- class CStg200xBasicNet

  Base class for the Stg200x.
- class CStg200xDownloadBasicNet

  CStg200xDownloadBasicNet is the base class to control the download mode of the MCS STG device.
- class CStg200xDownloadNet

  Main class for the STG download mode This class implements the STG download mode interface.
- class CStimulusFunctionNet
- class CSw2to64DeviceNet

  The class to control the MCS-USB-Sw2to64 device.
- class CTcxDeviceNet

  Class to control a Temperature Controller (TCX)
- class CTEERFunctionNet

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

  CUsbDeviceConfigurationFunctionNet is the class to configure the USB firmware

- class CUsbExceptionNet

  Exception class that is thrown in case of an USB error.
- class CW2100_FunctionNet
- class CW2100_StimulatorFunctionNet
- class CW2100DacqGroupChannelSelectionNet
- class CWarnerUssingDeviceNet

  CWarnerUssingDeviceNet is the class to control the Ussing device
- class CWarnerUssingFunctionNet

  CWarnerUssingFunctionNet is the class to control the Ussing device
- class CWarnerValveControllerDeviceNet

  CWarnerValveControllerDeviceNet is the class to access the Warner Valve Controller
- class CWarnerValveControllerDeviceTesterFunctionNet

  CWarnerValveControllerDeviceTesterFunctionNet is the class to access the functions for the Warner Valve Controller Device Tester
- class CWClassicFunctionNet
- class CWirelessBaseFunctionNet
- struct DeviceIdNet

  Device Id.
- class DigitalSource
- class DigitalSourceGeneral
- class DriverVersionNet

  Class gives firmware versions of the device's firmware destinations.
- class FirmwareDestinationNames
- class HeadStageIDType
- class HeadstageIDTypeObject
- class HeadStageIDTypeState
- class mkfilterNet
- class StgStatusNet
- class usbSetupPacket_t
- struct W2100_StimulusParametersNet

Enumerations

- enum enCMosMeaChipType {
  unknown = 0,
  nMos16LV = 1,
  nMos32LV = 3,
  nMos36LN = 6,
  nMos64LN = 7 }
- enum EnSTG200x_STATUS {
  OK,
  NOT_CONNECTED,
  DEVICE_NOT_FOUND }

Functions

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

  Delegate to show a device arrival or removal.

- public delegate void OnStgPollStatus (unsigned int status, StgStatusNet^ 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 enCMosMeaChipType `enum enCMosMeaChipType [strong]`

Enumerator

unknown	
nMos16LV	
nMos32LV	
nMos36LN	
nMos64LN	

10.2.1.2 EnSTG200x_STATUS `enum EnSTG200x_STATUS [strong]`

Enumerator

OK	
NOT_CONNECTED	
DEVICE_NOT_FOUND	

10.2.2 Function Documentation

10.2.2.1 OnChannelData()

```
public delegate void Mcs::Usb::OnChannelData (
        CMcsUsbDacqNet^ 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]`

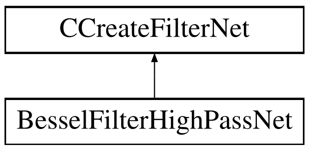
11.2.1.3 ChargeString `System:: String^ ChargeString [get]`

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

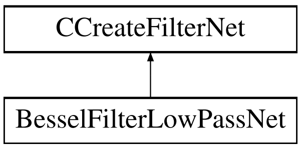
11.3.1 Constructor & Destructor Documentation

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

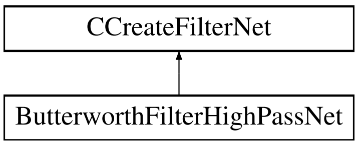
11.4.1 Constructor & Destructor Documentation

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

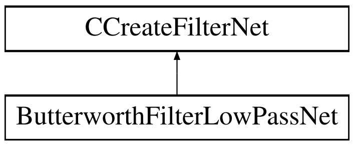
11.5.1 Constructor & Destructor Documentation

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

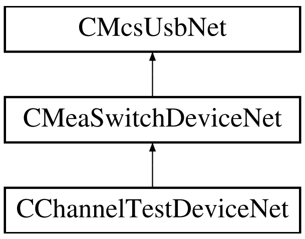
11.6.1 Constructor & Destructor Documentation

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()

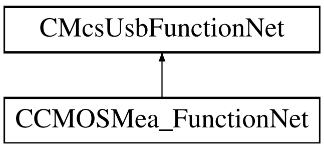
```
void SetAttenuation (
        unsigned int Attenuation )
```

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 Channel↩ Number)
- uint32_t GetGroupChannelBitmaskBySelect (DacqGroupChannelEnumNet GroupID, uint32_t Channel↩ Number, uint32_t virtualDevice)
- CMOSMeaInterfaceADCEnumNet GetGroupChannelBitmaskInterfaceADC (uint32_t ChannelNumber)
- CMOSMeaInterfaceADCEnumNet GetGroupChannelBitmaskInterfaceADC (uint32_t ChannelNumber, uint32_t virtualDevice)
- CMOSMeaIFDigChannelEnumNet GetGroupChannelBitmaskIFDigChannels (uint32_t ChannelNumber)
- CMOSMeaIFDigChannelEnumNet GetGroupChannelBitmaskIFDigChannels (uint32_t ChannelNumber, uint32_t virtualDevice)
- CMOSMeaHeadstage1NCBathCurrentEnumNet GetGroupChannelBitmaskHS1NCBathCurrent (uint32_t ChannelNumber)
- CMOSMeaHeadstage1NCBathCurrentEnumNet GetGroupChannelBitmaskHS1NCBathCurrent (uint32_t ChannelNumber, uint32_t virtualDevice)
- CMOSMeaHeadstage1NCCol2CurrentEnumNet GetGroupChannelBitmaskHS1NCCol2Current (uint32_t ChannelNumber)
- CMOSMeaHeadstage1NCCol2CurrentEnumNet GetGroupChannelBitmaskHS1NCCol2Current (uint32_t ChannelNumber, uint32_t virtualDevice)
- CMOSMeaHeadstage1NChipTempEnumNet GetGroupChannelBitmaskHS1NChipTemp (uint32_t Channel↩ Number)
- CMOSMeaHeadstage1NChipTempEnumNet GetGroupChannelBitmaskHS1NChipTemp (uint32_t Channel↩ Number, uint32_t virtualDevice)

- CMOSMeaSTG1DACSignalEnumNet GetGroupChannelBitmaskSTG1DACSignal (uint32_t Channel↩ Number)
- CMOSMeaSTG1DACSignalEnumNet GetGroupChannelBitmaskSTG1DACSignal (uint32_t Channel↩ Number, uint32_t virtualDevice)
- CMOSMeaHS1SidebandEnumNet GetGroupChannelBitmaskHS1Sidebands (uint32_t ChannelNumber)
- CMOSMeaHS1SidebandEnumNet GetGroupChannelBitmaskHS1Sidebands (uint32_t ChannelNumber, uint32_t virtualDevice)
- CMOSMeaHS1TriggerStatusEnumNet GetGroupChannelBitmaskHS1TriggerStatus (uint32_t Channel↩ Number)
- CMOSMeaHS1TriggerStatusEnumNet GetGroupChannelBitmaskHS1TriggerStatus (uint32_t Channel↩ Number, uint32_t virtualDevice)
- CMOSMeaPacketFrameContextGroupEnumNet GetGroupChannelBitmaskPacketFrameContext (uint32_↩ t ChannelNumber)
- CMOSMeaPacketFrameContextGroupEnumNet GetGroupChannelBitmaskPacketFrameContext (uint32_↩ t ChannelNumber, uint32_t virtualDevice)

Additional Inherited Members

11.8.1 Constructor & Destructor Documentation

11.8.1.1 CCMOSMea_FunctionNet() [1/2]

```
CCMOSMea_FunctionNet (
        CMcsUsbNet^ 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]

```
CMOSMeaHeadstage1NCBathCurrentEnum←
Net GetGroupChannelBitmaskHS1NCBathCurrent (
            uint32_t ChannelNumber )
```

11.8.2.18 GetGroupChannelBitmaskHS1NCBathCurrent() [2/2]

```
CMOSMeaHeadstage1NCBathCurrentEnum←
Net GetGroupChannelBitmaskHS1NCBathCurrent (
            uint32_t ChannelNumber,
            uint32_t virtualDevice )
```

11.8.2.19 GetGroupChannelBitmaskHS1NCCol2Current() [1/2]

```
CMOSMeaHeadstage1NCCol2CurrentEnum←
Net GetGroupChannelBitmaskHS1NCCol2Current (
            uint32_t ChannelNumber )
```

11.8.2.20 GetGroupChannelBitmaskHS1NCCol2Current() [2/2]

```
CMOSMeaHeadstage1NCCol2CurrentEnum←
Net GetGroupChannelBitmaskHS1NCCol2Current (
            uint32_t ChannelNumber,
            uint32_t virtualDevice )
```

11.8.2.21 GetGroupChannelBitmaskHS1NChipTemp() [1/2]

```
CMOSMeaHeadstage1NChipTempEnumNet Get←
GroupChannelBitmaskHS1NChipTemp (
            uint32_t ChannelNumber )
```

11.8.2.22 GetGroupChannelBitmaskHS1NChipTemp() [2/2] `CMOSMeaHeadstage1NChipTempEnumNet Get↩ GroupChannelBitmaskHS1NChipTemp (`

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

11.8.2.23 GetGroupChannelBitmaskHS1Sidebands() [1/2] `CMOSMeaHS1SidebandEnumNet GetGroup↩ ChannelBitmaskHS1Sidebands (`

```
            uint32_t ChannelNumber )
```

11.8.2.24 GetGroupChannelBitmaskHS1Sidebands() [2/2] `CMOSMeaHS1SidebandEnumNet GetGroup↩ ChannelBitmaskHS1Sidebands (`

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

11.8.2.25 GetGroupChannelBitmaskHS1TriggerStatus() [1/2] `CMOSMeaHS1TriggerStatusEnumNet Get↩ GroupChannelBitmaskHS1TriggerStatus (`

```
            uint32_t ChannelNumber )
```

11.8.2.26 GetGroupChannelBitmaskHS1TriggerStatus() [2/2] `CMOSMeaHS1TriggerStatusEnumNet Get↩ GroupChannelBitmaskHS1TriggerStatus (`

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

11.8.2.27 GetGroupChannelBitmaskIFDigChannels() [1/2] `CMOSMeaIFDigChannelEnumNet GetGroup↩ ChannelBitmaskIFDigChannels (`

```
            uint32_t ChannelNumber )
```

11.8.2.28 GetGroupChannelBitmaskIFDigChannels() [2/2] `CMOSMeaIFDigChannelEnumNet GetGroup↩ ChannelBitmaskIFDigChannels (`

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

11.8.2.29 GetGroupChannelBitmaskInterfaceADC() [1/2] `CMOSMeaInterfaceADCEnumNet GetGroup↩ ChannelBitmaskInterfaceADC (`

```
            uint32_t ChannelNumber )
```

11.8.2.30 GetGroupChannelBitmaskInterfaceADC() [2/2]

```
CMOSMeaInterfaceADCEnumNet GetGroup←↩
ChannelBitmaskInterfaceADC (
            uint32_t ChannelNumber,
            uint32_t virtualDevice )
```

11.8.2.31 GetGroupChannelBitmaskPacketFrameContext() [1/2]

```
CMOSMeaPacketFrameContextGroup←↩
EnumNet GetGroupChannelBitmaskPacketFrameContext (
            uint32_t ChannelNumber )
```

11.8.2.32 GetGroupChannelBitmaskPacketFrameContext() [2/2]

```
CMOSMeaPacketFrameContextGroup←↩
EnumNet GetGroupChannelBitmaskPacketFrameContext (
            uint32_t ChannelNumber,
            uint32_t virtualDevice )
```

11.8.2.33 GetGroupChannelBitmaskSTG1DACSignal() [1/2]

```
CMOSMeaSTG1DACSignalEnumNet GetGroup←↩
ChannelBitmaskSTG1DACSignal (
            uint32_t ChannelNumber )
```

11.8.2.34 GetGroupChannelBitmaskSTG1DACSignal() [2/2]

```
CMOSMeaSTG1DACSignalEnumNet GetGroup←↩
ChannelBitmaskSTG1DACSignal (
            uint32_t ChannelNumber,
            uint32_t virtualDevice )
```

11.8.2.35 GetGroupDCOffset() [1/2]

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

11.8.2.36 GetGroupDCOffset() [2/2]

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

11.8.2.37 GetGroupID() [1/2]

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

11.8.2.38 GetGroupID() [2/2]

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

11.8.2.39 GetGroupNumberOfChannels() [1/2]

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

11.8.2.40 GetGroupNumberOfChannels() [2/2]

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

11.8.2.41 GetGroupResolutionPerDigit() [1/2]

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

11.8.2.42 GetGroupResolutionPerDigit() [2/2]

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

11.8.2.43 GetGroupSampleSize() [1/2]

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

11.8.2.44 GetGroupSampleSize() [2/2]

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

11.8.2.45 GetGroupType() [1/2]

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

11.8.2.46 GetGroupType() [2/2]

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

11.8.2.47 GetGroupUnit() [1/2]

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

11.8.2.48 GetGroupUnit() [2/2]

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

11.8.2.49 GetMaxNumOfColumns()

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

11.8.2.50 GetNeurochipMemoryData() [1/2]

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

11.8.2.51 GetNeurochipMemoryData() [2/2]

```
array<uint32_t> ^ GetNeurochipMemoryData (
    uint16_t MemAddress,
    uint32_t 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]

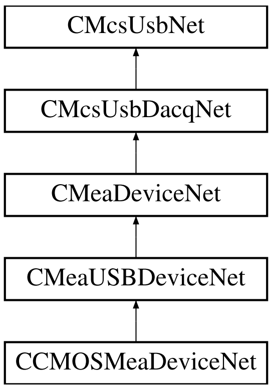
```
void VOPSTimerSetResetTimes (
        uint32_t ResetTime,
        uint32_t IntervalTime )
```

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 ()
- 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 SetBaseSamplerate()

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

11.9.2.9 SetRegionOfInterests()

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

11.9.2.10 UpdateChannelBlock()

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

11.9.3 Property Documentation

11.9.3.1 CMosMea

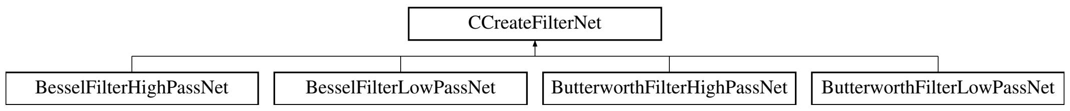
```
CCMOSMea_FunctionNet^ CMosMea [get]
```

11.9.3.2 Stimulus

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

11.10 CCreateFilterNet Class Reference

Inheritance diagram for CCreateFilterNet:



Public Member Functions

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

Static Public Member Functions

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

Protected Member Functions

- CCreateFilterNet (int numCoefSets, CCreateFilter *pCreateFilter)

Properties

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

11.10.1 Constructor & Destructor Documentation

11.10.1.1 CCreateFilterNet() [1/2]

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

11.10.1.2 ∼CCreateFilterNet() ∼CCreateFilterNet ( )

11.10.1.3 CCreateFilterNet() [2/2] CCreateFilterNet (

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

11.10.2 Member Function Documentation

11.10.2.1 FindFilter() [1/2] static int FindFilter (

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

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

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

11.10.2.3 GetBiQuad() CFilterCoefficientsNet ^ GetBiQuad (

```
int index )
```

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

11.10.3 Property Documentation

11.10.3.1 CutoffFrequency double CutoffFrequency [get]

11.10.3.2 NumCoefSets int NumCoefSets [get]

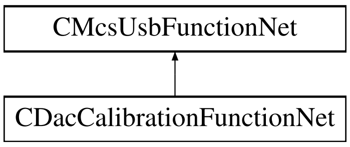
11.10.3.3 Order `int Order [get]`

11.10.3.4 SampleRate `double SampleRate [get]`

11.10.3.5 Scale `double Scale [get]`

11.11 CDacCalibrationFunctionNet Class Reference

Inheritance diagram for CDacCalibrationFunctionNet:



Public Member Functions

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

  Initializes a new instance of the CDacCalibrationFunctionNet class.
- CDacCalibrationFunctionNet (CMcsUsbNet^ mcsusb)
- virtual ∼CDacCalibrationFunctionNet ()
- !CDacCalibrationFunctionNet ()
- void SetDacOffset (uint16_t dacChannel, uint32_t offset)

  Sets the offset of a DAC channel.
- uint32_t GetDacOffset (uint16_t dacChannel)

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

  Writes the offset of a DAC channel to permanent memory.

Additional Inherited Members

11.11.1 Detailed Description

11.11.2 Constructor & Destructor Documentation

11.11.2.1 CDacCalibrationFunctionNet() [1/2]

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

Initializes a new instance of the CDacCalibrationFunctionNet class.

11.11.2.2 CDacCalibrationFunctionNet() [2/2]

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

11.11.2.3 ∼CDacCalibrationFunctionNet()

```
virtual ∼CDacCalibrationFunctionNet ( ) [virtual]
```

11.11.2.4 "!CDacCalibrationFunctionNet()

```
!CDacCalibrationFunctionNet ( )
```

11.11.3 Member Function Documentation

11.11.3.1 BurnDacOffset()

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

Writes the offset of a DAC channel to permanent memory.

Parameters

dacChannel	The DAC channel number.

11.11.3.2 GetDacOffset()

```
uint32_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,
        uint32_t offset )
```

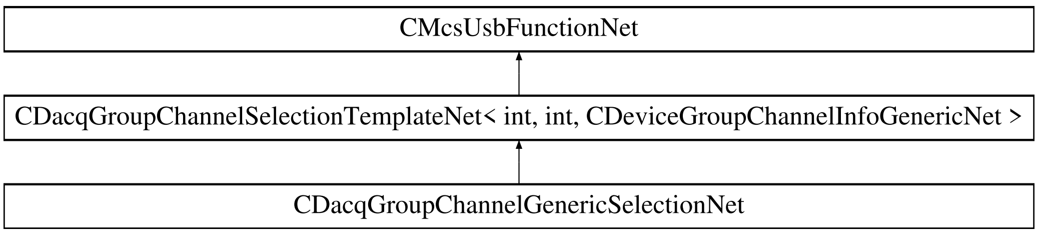
Sets the offset of a DAC channel.

Parameters

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

11.12 CDacqGroupChannelGenericSelectionNet Class Reference

Inheritance diagram for CDacqGroupChannelGenericSelectionNet:


CMcsUsbFunctionNet
CDacqGroupChannelSelectionTemplateNet< int, int, CDeviceGroupChannelInfoGenericNet >
CDacqGroupChannelGenericSelectionNet

Public Member Functions

- CDacqGroupChannelGenericSelectionNet (CMcsUsbNet^ mcsusb)

Additional Inherited Members

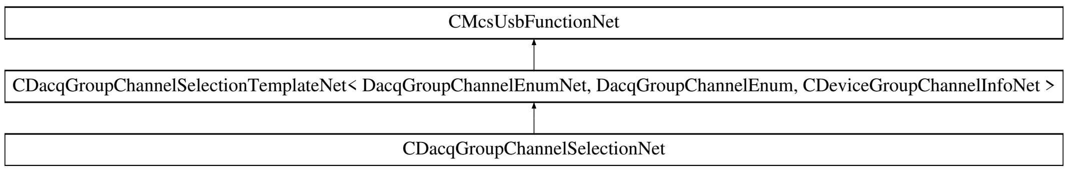
11.12.1 Constructor & Destructor Documentation

11.12.1.1 CDacqGroupChannelGenericSelectionNet()

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

11.13 CDacqGroupChannelSelectionNet Class Reference

Inheritance diagram for CDacqGroupChannelSelectionNet:


CMcsUsbFunctionNet
CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumNet, DacqGroupChannelEnum, CDeviceGroupChannelInfoNet >
CDacqGroupChannelSelectionNet

Public Member Functions

- CDacqGroupChannelSelectionNet (CMcsUsbNet^ mcsusb)

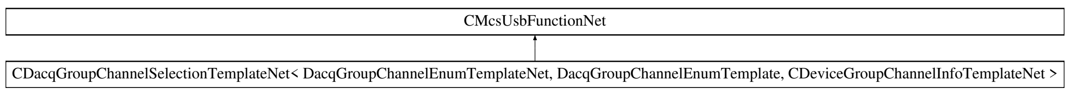
Additional Inherited Members

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 >:


CMcsUsbFunctionNet
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^> ^ Get←`

```
DeviceGroupChannelInfos ( )
```

11.14.2.4 GetDeviceGroupChannelInfos() `[2/2]` `List<CDeviceGroupChannelInfoTemplateNet^> ^ Get←`

```
DeviceGroupChannelInfos (
        uint32_t virtualDevice )
```

11.14.2.5 GetEnabledChannelsInGroup() `[1/2]` `List<bool> ^ GetEnabledChannelsInGroup (`

```
DacqGroupChannelEnumTemplateNet GroupID )
```

11.14.2.6 GetEnabledChannelsInGroup() `[2/2]` `List<bool> ^ GetEnabledChannelsInGroup (`

```
DacqGroupChannelEnumTemplateNet GroupID,
uint32_t virtualDevice )
```

11.14.2.7 GetGroupID() [1/2]

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

11.14.2.8 GetGroupID() [2/2]

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

11.14.2.9 GetGroupNumberOfChannels() [1/2]

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

11.14.2.10 GetGroupNumberOfChannels() [2/2]

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

11.14.2.11 GetGroupSampleSize() [1/2]

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

11.14.2.12 GetGroupSampleSize() [2/2]

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

11.14.2.13 GetGroupType() [1/2]

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

11.14.2.14 GetGroupType() [2/2]

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

11.14.2.15 GetNumberOfSupportedGroups() [1/2]

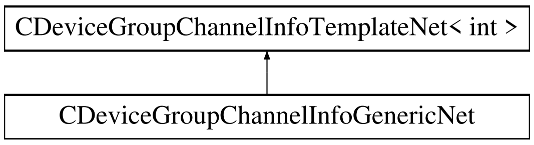
```
uint32_t GetNumberOfSupportedGroups ( )
```

11.14.2.16 GetNumberOfSupportedGroups() [2/2]

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

11.15 CDeviceGroupChannelInfoGenericNet Class Reference

Inheritance diagram for CDeviceGroupChannelInfoGenericNet:


CDeviceGroupChannelInfoTemplateNet< int >
CDeviceGroupChannelInfoGenericNet

Public Member Functions

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

Additional Inherited Members

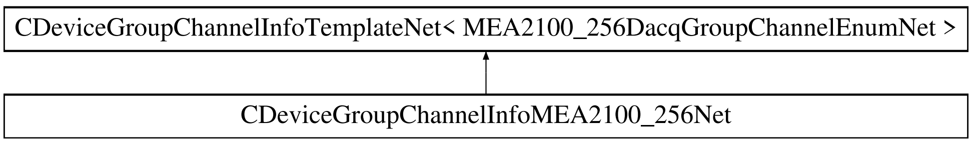
11.15.1 Constructor & Destructor Documentation

11.15.1.1 CDeviceGroupChannelInfoGenericNet()

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

11.16 CDeviceGroupChannelInfoMEA2100_256Net Class Reference

Inheritance diagram for CDeviceGroupChannelInfoMEA2100_256Net:


CDeviceGroupChannelInfoTemplateNet< MEA2100_256DacqGroupChannelEnumNet >
CDeviceGroupChannelInfoMEA2100_256Net

Public Member Functions

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

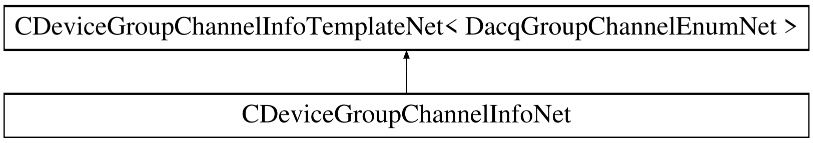
Additional Inherited Members

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:


CDeviceGroupChannelInfoTemplateNet< DacqGroupChannelEnumNet >
CDeviceGroupChannelInfoNet

Public Member Functions

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

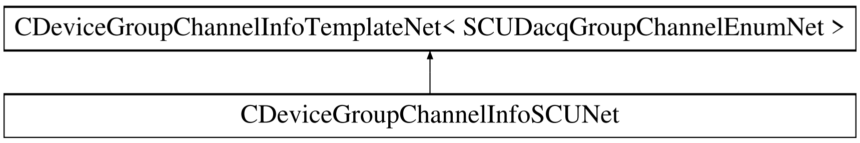
Additional Inherited Members

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:


CDeviceGroupChannelInfoTemplateNet< SCUDacqGroupChannelEnumNet >
CDeviceGroupChannelInfoSCUNet

Public Member Functions

- CDeviceGroupChannelInfoSCUNet (SCUDacqGroupChannelEnumNet id, int channels, DacqMeaGroup↩
TypeEnumNet 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, DacqMea↩
GroupTypeEnumNet 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

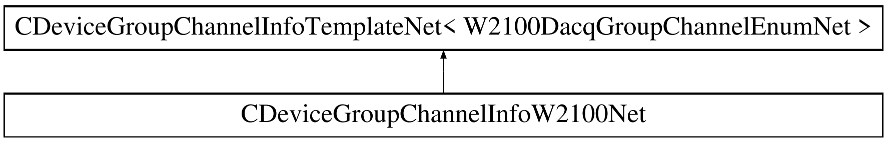
11.19.2.1 GroupID `DacqGroupChannelEnumTemplateNet GroupID`

11.19.2.2 GroupType `DacqMeaGroupTypeEnumNet GroupType`

11.19.2.3 NumberOfChannels `int NumberOfChannels`

11.20 CDeviceGroupChannelInfoW2100Net Class Reference

Inheritance diagram for CDeviceGroupChannelInfoW2100Net:


CDeviceGroupChannelInfoTemplateNet< W2100DacqGroupChannelEnumNet >
CDeviceGroupChannelInfoW2100Net

Public Member Functions

- CDeviceGroupChannelInfoW2100Net (W2100DacqGroupChannelEnumNet id, int channels, DacqMea↩ GroupTypeEnumNet type)

Additional Inherited Members

11.20.1 Constructor & Destructor Documentation

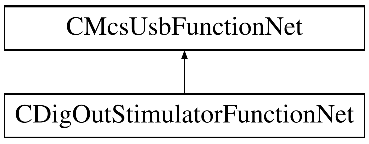
11.20.1.1 CDeviceGroupChannelInfoW2100Net() `CDeviceGroupChannelInfoW2100Net (`

```
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 ^ Prepare↩
ChannelData (
            array< int32_t >^ Amplitude,
            array< uint64_t >^ Duration )
```

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

Parameters

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

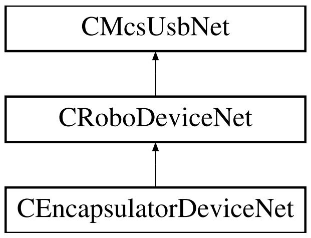
Parameters

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 )`

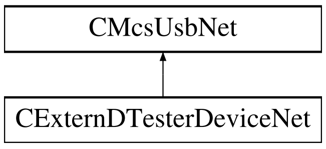
11.22.3 Member Function Documentation

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

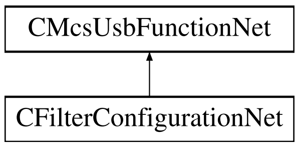
```
array< double>^ A [get]
```

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, FilterAttribute↩ EnumNet 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()

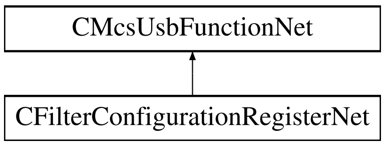
```
void SetFilterParameterPermanent (
    DacqGroupChannelEnumNet GroupID,
    uint32_t FilterNumber )
```

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 Filter↩ InfoRegBase, 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 E↩ EPROMBase, 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, Filter←FamilyEnumNet 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]
```

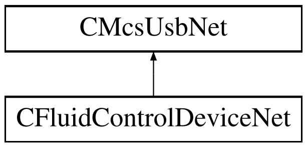
11.27.3.6 FilterType `FilterTypeEnumNet FilterType [get]`

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 ℃.

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.

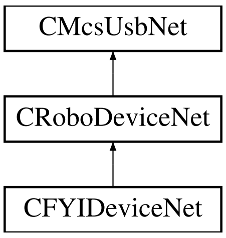
11.28.4 Property Documentation

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]`

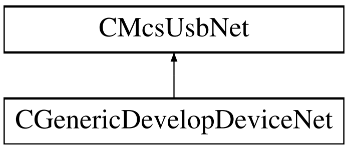
11.29.3.2 FYITemp `CRobo_FYITemp_FunctionNet^ FYITemp [get]`

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.

- 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 WritePipe()

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

Write data to an USB Endpoint.

Parameters

pipeHandle	The endpoint handle.

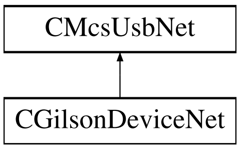
Parameters

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 )`

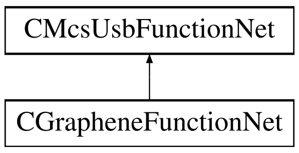
11.31.4 Member Data Documentation

11.31.4.1 m_pGilsonDevice `CGilsonDevice* m_pGilsonDevice [protected]`

11.32 CGrapheneFunctionNet Class Reference

CGrapheneFunctionNet is the class to control the TEER device

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

Additional Inherited Members

11.32.1 Detailed Description

CGrapheneFunctionNet is the class to control the TEER device

11.32.2 Constructor & Destructor Documentation

11.32.2.1 CGrapheneFunctionNet() [1/2]

```
CGrapheneFunctionNet (
        CMcsUsbNet^ 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 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
offset_vs	Vs offset

11.32.3.2 GetDACOffset() `[2/2]`

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

Gets the DAC offset

Parameters

Headstage	The headstage to query.
offset_vd	Vd offset
offset_vs	Vs offset

11.32.3.3 GetVdVs() `[1/2]`

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

Gets Vd and Vs

Parameters

Vd	Vd
Vs	Vs

11.32.3.4 GetVdVs() `[2/2]`

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

Gets Vd and Vs

Parameters

Headstage	The headstage to query.
Vd	Vd
Vs	Vs

11.32.3.5 GetVdVsDAC() `[1/2]`

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

Gets Vd and Vs

Parameters

Vd	Vd
Vs	Vs

11.32.3.6 GetVdVsDAC() `[2/2]`

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

Gets Vd and Vs

Parameters

Headstage	The headstage to query.
Vd	Vd
Vs	Vs

11.32.3.7 GetVoltageRange() `[1/2]`

```
int32_t GetVoltageRange ( )
```

Gets the voltage range

Returns

The voltage range

11.32.3.8 GetVoltageRange() `[2/2]`

```
int32_t GetVoltageRange (
        uint32_t Headstage )
```

Gets the voltage range

Parameters

Headstage	The headstage to query.

Returns

The voltage range

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

Gets the reached voltage

Returns

the reached voltage

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

Gets the reached voltage

Parameters

Headstage	The headstage to query.

Returns

The reached voltage

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

Gets the voltage resolution

Returns

The voltage resolution

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

Gets the voltage resolution

Parameters

Headstage	The headstage to query.

Returns

The voltage resolution

11.32.3.13 SetDACOffset() [1/2]

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

Sets the DAC offset

Parameters

offset_vd	Vd
offset_vs	Vs

11.32.3.14 SetDACOffset() [2/2]

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

Set the DAC offset

Parameters

Headstage	The headstage to query.
offset_vd	Vd
offset_vs	Vs

11.32.3.15 SetVds() [1/2]

```
void SetVds (
    int32_t Vds )
```

Sets Vds

Parameters

Vds	Vds

11.32.3.16 SetVds() [2/2]

```
void SetVds (
    uint32_t Headstage,
    int32_t Vds )
```

Sets Vds

Parameters

Headstage	The headstage to query.
Vds	Vds

11.32.3.17 SetVdVs() [1/2]

```
void SetVdVs (
        int32_t Vd,
        int32_t Vs )
```

Sets Vd and Vs

Parameters

Vd	Vd
Vs	Vs

11.32.3.18 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
Vs	Vs

11.32.3.19 SetVdVsDAC() [1/2]

```
void SetVdVsDAC (
        int16_t Vd,
        int16_t Vs )
```

Sets Vd and Vs

Parameters

Vd	Vd
Vs	Vs

11.32.3.20 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
Vs	Vs

11.32.3.21 SetVgs() `[1/2]`

```
void SetVgs (
    int32_t Vgs )
```

Sets Vgs

Parameters

Vgs	Vgs

11.32.3.22 SetVgs() `[2/2]`

```
void SetVgs (
    uint32_t Headstage,
    int32_t Vgs )
```

Sets Vgs

Parameters

Headstage	The headstage to query.
Vgs	Vgs

11.32.3.23 SetVoltageRange() `[1/2]`

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

Sets the voltage range

Parameters

range	The voltage range

11.32.3.24 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

11.32.3.25 SetVoltageResolution() `[1/2]`

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

Sets the voltage resolution

Parameters

resolution	The voltage resolution

11.32.3.26 SetVoltageResolution() `[2/2]`

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

Sets the voltage resolution

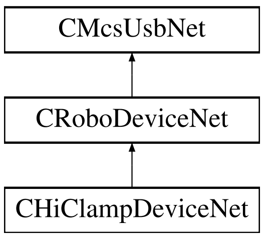
Parameters

Headstage	The headstage to query.
resolution	The voltage resolution

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

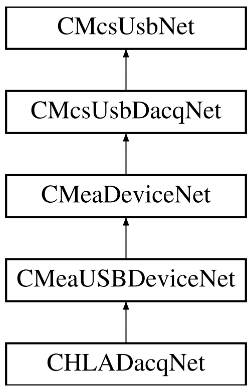
11.33.2.1 CHiClampDeviceNet() `CHiClampDeviceNet ( void )`

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

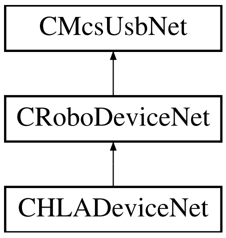
11.34.1 Constructor & Destructor Documentation

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 uint32_t GetAvailableVoltageRangesInMicroVolt ([System::Runtime::InteropServices::Out]System::↵Collections::Generic::List< int32_t >^% voltageRanges)
- virtual uint32_t GetAvailableVoltageRangesInMicroVoltAndStringsInMilliVolt ([System::Runtime::Interop↵Services::Out]System::Collections::Generic::List< CVoltageRangeInfoNet^ >^% voltageRanges)

11.36.1 Detailed Description

Class to provide hardware information about the device.

11.36.2 Constructor & Destructor Documentation

11.36.2.1 CHWInfo() `CHWInfo ( CMcsUsbDacqNet^ device )`

11.36.3 Member Function Documentation

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

11.36.3.2 GetAvailableVoltageRangesInMicroVolt() `virtual uint32_t GetAvailableVoltageRangesInMicroVolt ( [System::Runtime::InteropServices::Out] System::Collections::Generic::List< int32_t >^% voltageRanges ) [virtual]`

11.36.3.3 GetAvailableVoltageRangesInMicroVoltAndStringsInMilliVolt() `virtual uint32_t GetAvailableVoltageRangesInMicroVoltAndStringsInMilliVolt ( [System::Runtime::InteropServices::Out] System::Collections::Generic::List< CVoltageRangeInfoNet voltageRanges ) [virtual]`

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]`

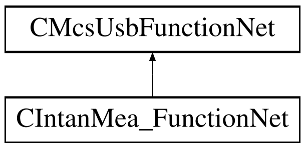
Query if the digital channel replaces an analog channel when enabled (e.g. on MC_Card) or adds a channel link on USB devices.

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 )`

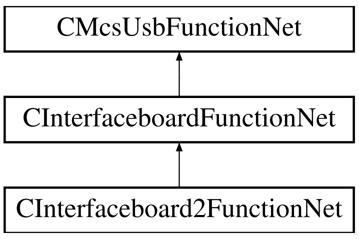
11.37.2.10 SetDSPHighPassByIndex() `void SetDSPHighPassByIndex ( int index )`

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.

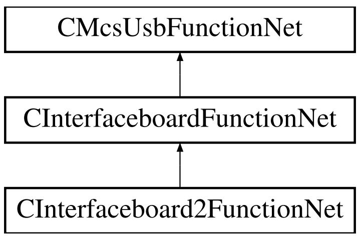
Parameters

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

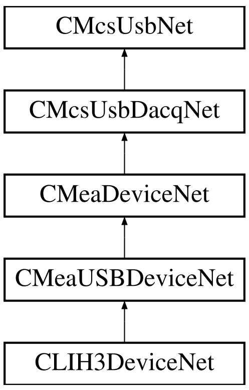
Parameters

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_EP←C10_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.

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 SendCommand()

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

Send command to the EPC10

Parameters

epc10bus	The EPC10 bus
Command	The command

11.40.3.17 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

11.40.3.18 SetAdcOffsetPermanent()

```
void SetAdcOffsetPermanent (
    LIH30_ADC_Channel_EnumNet AdcChannel )
```

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

Parameters

AdcChannel	The ADC channel

11.40.3.19 SetAudioOutDacParameter()

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

Sets the parameter of the audio DAC output.

Parameters

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

11.40.3.20 SetDacIdleValue()

```
void SetDacIdleValue (
    uint32_t DacChannel,
    int32_t IdleValue )
```

Sets the DAC Idle value

Parameters

DacChannel	The DAC channel
IdleValue	The idle value

11.40.3.21 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

11.40.3.22 SetDacOffsetPermanent()

```
void SetDacOffsetPermanent (
    LIH30_DAC_Channel_EnumNet DacChannel )
```

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

Parameters

DacChannel	The DAC channel

11.40.3.23 SetDacUseIdleValue()

```
void SetDacUseIdleValue (
    uint32_t DacChannel,
    bool UseIdle )
```

Sets if the DAC Idle value is used after stimulation

Parameters

DacChannel	The DAC channel
UseIdle	Use idle value

11.40.3.24 SetDigOutState()

```
void SetDigOutState (
    uint16_t DigOutState )
```

Writes to the LIH30 digital output

Parameters

DigOutState	The bit mask defining the digital output state

11.40.3.25 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

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

Sets the Sample Interval for the DACQ and Stimulation

Parameters

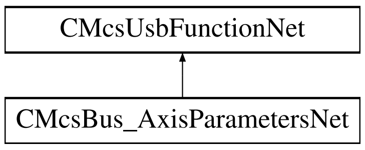
SampleInterval	between the samples, Sample interval is available from 1 to 4194303

11.40.4 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 (`

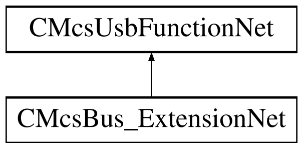
```
unsigned char busnumber,
unsigned char busaddress,
unsigned char axis,
unsigned short index,
int parameter )
```

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()

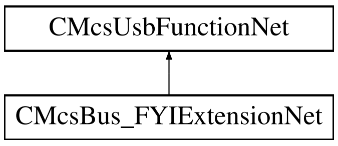
```
unsigned short GetLEDSwitch (
    unsigned char busnumber,
    unsigned char busaddress )
```

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()

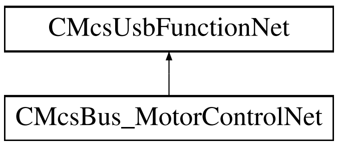
```
void SetSingleHeater (
        unsigned char busnumber,
        unsigned char busaddress,
        short index,
        unsigned short power )
```

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, Robo↩ CurrentModeEnumNet 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 GetMCCurrentModeShort↩ Command (`

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

11.44.2.14 GetMCCurrentPosition() `int GetMCCurrentPosition (`

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

11.44.2.15 GetMCCurrentShortCommand() `short GetMCCurrentShortCommand (`

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

11.44.2.16 GetMCCurrentSpeed() `short GetMCCurrentSpeed (`

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

11.44.2.17 GetMCMaxAcceleration() `unsigned short GetMCMaxAcceleration (`

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

11.44.2.18 GetMCMaxAccelerationEeprom() `unsigned short GetMCMaxAccelerationEeprom (`

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

11.44.2.19 GetMCMaxCurrent() `short GetMCMaxCurrent (`

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

11.44.2.20 GetMCMaxCurrentEeprom() `short GetMCMaxCurrentEeprom (`

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

11.44.2.21 GetMCMaxSpeed() `unsigned short GetMCMaxSpeed (`

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

11.44.2.22 GetMCMaxSpeedEeprom() `unsigned short GetMCMaxSpeedEeprom (`

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

11.44.2.23 GetMCMaxTravel()

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

11.44.2.24 GetMCMaxTravelEeprom()

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

11.44.2.25 GetMCMaxTravelShortCommand()

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

11.44.2.26 GetMCMovement()

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

11.44.2.27 GetMCNewPosition()

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

11.44.2.28 GetMCOutputOnOff()

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

11.44.2.29 GetMCPhase()

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

11.44.2.30 GetMCPhaseOffset() `unsigned short GetMCPhaseOffset (`

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

11.44.2.31 GetMCReference() `unsigned char GetMCReference (`

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

11.44.2.32 GetMCReferenceCurrent() `short GetMCReferenceCurrent (`

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

11.44.2.33 GetMCReferenceCurrentEeprom() `short GetMCReferenceCurrentEeprom (`

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

11.44.2.34 GetMCRegulatorGain() `short GetMCRegulatorGain (`

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

11.44.2.35 GetMCRegulatorGainEeprom() `short GetMCRegulatorGainEeprom (`

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

11.44.2.36 GetMCScalingFactor() `int GetMCScalingFactor (`

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

11.44.2.37 GetMCScalingFactorEeprom() `int GetMCScalingFactorEeprom (`

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

11.44.2.38 GetMCSpeed() `short GetMCSpeed (`

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

11.44.2.39 GetMCSpeedEeprom() `unsigned short GetMCSpeedEeprom (`

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

11.44.2.40 GetMCSpeedShortCommand() `short GetMCSpeedShortCommand (`

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

11.44.2.41 GetMCSpeedUnitEeprom() `int32_t GetMCSpeedUnitEeprom (`

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

11.44.2.42 GetMCStandbyCurrent() `short GetMCStandbyCurrent (`

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

11.44.2.43 GetMCStandbyCurrentEeprom() `short GetMCStandbyCurrentEeprom (`

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

11.44.2.44 GetMCStandbyTime()

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

11.44.2.45 GetMCStandbyTimeEeprom()

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

11.44.2.46 GetSubChannel()

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

11.44.2.47 SetMCAcceleration()

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

11.44.2.48 SetMCAccelerationEeprom()

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

11.44.2.49 SetMCAccelerationShortCommand()

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

11.44.2.50 SetMCAxisRevisionEeprom()

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

11.44.2.51 SetMCBreakCurrent()

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

11.44.2.52 SetMCBreakCurrentEeprom()

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

11.44.2.53 SetMCConfig()

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

11.44.2.54 SetMCConfigEeprom()

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

11.44.2.55 SetMCCurrent()

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

11.44.2.56 SetMCCurrentEeprom()

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

11.44.2.57 SetMCCurrentMode()

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

11.44.2.58 SetMCCurrentModeEeprom()

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

11.44.2.59 SetMCCurrentModeShortCommand()

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

11.44.2.60 SetMCCurrentPosition()

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

11.44.2.61 SetMCCurrentShortCommand()

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

11.44.2.62 SetMCMaxAcceleration()

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

11.44.2.63 SetMCMaxAccelerationEeprom() `void SetMCMaxAccelerationEeprom (`

```
unsigned char busnumber,
unsigned char busaddress,
unsigned char axis,
unsigned short acceleration )
```

11.44.2.64 SetMCMaxCurrent() `void SetMCMaxCurrent (`

```
unsigned char busnumber,
unsigned char busaddress,
unsigned char axis,
short current )
```

11.44.2.65 SetMCMaxCurrentEeprom() `void SetMCMaxCurrentEeprom (`

```
unsigned char busnumber,
unsigned char busaddress,
unsigned char axis,
short current )
```

11.44.2.66 SetMCMaxSpeed() `void SetMCMaxSpeed (`

```
unsigned char busnumber,
unsigned char busaddress,
unsigned char axis,
unsigned short speed )
```

11.44.2.67 SetMCMaxSpeedEeprom() `void SetMCMaxSpeedEeprom (`

```
unsigned char busnumber,
unsigned char busaddress,
unsigned char axis,
unsigned short speed )
```

11.44.2.68 SetMCMaxTravel() `void SetMCMaxTravel (`

```
unsigned char busnumber,
unsigned char busaddress,
unsigned char axis,
int travel )
```

11.44.2.69 SetMCMaxTravelEeprom() `void SetMCMaxTravelEeprom (`

```
unsigned char busnumber,
unsigned char busaddress,
unsigned char axis,
int travel )
```

11.44.2.70 SetMCMaxTravelShortCommand() `void SetMCMaxTravelShortCommand (`

```
unsigned char busnumber,
unsigned char busaddress,
unsigned char axis,
int travel )
```

11.44.2.71 SetMCNewPosition() `void SetMCNewPosition (`

```
unsigned char busnumber,
unsigned char busaddress,
unsigned char axis,
int position )
```

11.44.2.72 SetMCOutputOnOff() `void SetMCOutputOnOff (`

```
unsigned char busnumber,
unsigned char busaddress,
unsigned char axis,
bool OnOff_status )
```

11.44.2.73 SetMCReference() `void SetMCReference (`

```
unsigned char busnumber,
unsigned char busaddress,
unsigned char axis,
unsigned char switch_enable,
unsigned char switch_polarity )
```

11.44.2.74 SetMCReferenceCurrent() `void SetMCReferenceCurrent (`

```
unsigned char busnumber,
unsigned char busaddress,
unsigned char axis,
short current )
```

11.44.2.75 SetMCReferenceCurrentEeprom()

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

11.44.2.76 SetMCRegulatorGain()

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

11.44.2.77 SetMCRegulatorGainEeprom()

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

11.44.2.78 SetMCRotation()

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

11.44.2.79 SetMCScalingFactor()

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

11.44.2.80 SetMCScalingFactorEeprom()

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

11.44.2.81 SetMCSpeed()

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

11.44.2.82 SetMCSpeedEeprom()

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

11.44.2.83 SetMCSpeedShortCommand()

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

11.44.2.84 SetMCSpeedUnitEeprom()

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

11.44.2.85 SetMCStandbyCurrent()

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

11.44.2.86 SetMCStandbyCurrentEeprom()

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

11.44.2.87 SetMCStandbyTime() `void SetMCStandbyTime (`

```
unsigned char busnumber,
unsigned char busaddress,
unsigned char axis,
short t )
```

11.44.2.88 SetMCStandbyTimeEeprom() `void SetMCStandbyTimeEeprom (`

```
unsigned char busnumber,
unsigned char busaddress,
unsigned char axis,
short t )
```

11.44.2.89 SetSubChannel() `void SetSubChannel (`

```
unsigned char busnumber,
unsigned char busaddress,
unsigned char axis,
unsigned short subchannel )
```

11.44.2.90 StartMCMovement() `void StartMCMovement (`

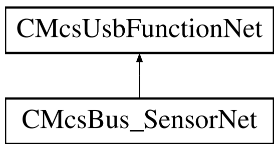
```
unsigned char busnumber,
unsigned char busaddress,
unsigned char axis )
```

11.44.2.91 StopMCMovement() `void StopMCMovement (`

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

11.45 CMcsBus_SensorNet Class Reference

Inheritance diagram for CMcsBus_SensorNet:



Public Member Functions

- CMcsBus_SensorNet (CMcsUsbNet^ 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::Interop←Services::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()

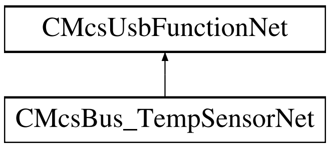
```
int TactSwitchGetState (
    unsigned char busnumber,
    unsigned char busaddress )
```

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()

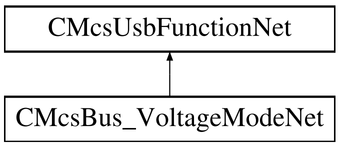
```
void SetNanoVoltsPerKelvin (
    unsigned char busnumber,
    unsigned char busaddress,
    int nanovoltsperkelvin )
```

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()

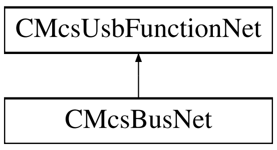
```
void SetVMOutputOnOff (
        unsigned char busnumber,
        unsigned char busaddress,
        unsigned char channel,
        unsigned short status )
```

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 )
```

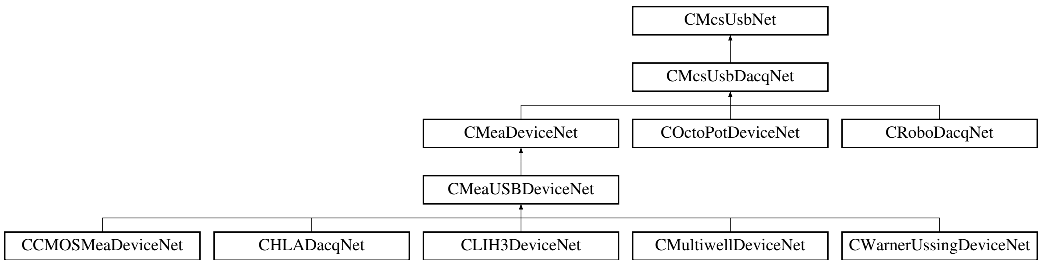
11.48.2.18 SetHWRevisionEeprom()

```
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 ()
- 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_256DigitalSource↩ EnumNet 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::Interop↩ Services::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::Interop↩ Services::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::Interop↩ Services::Out]DigitalSource< digitalsourceenum >^% source, [System::Runtime::InteropServices::Out]int% bitnumber_offset)

Gets the function/source of an digital output bit.

- virtual AdapterTypeEnumNet GetAdapterType ()

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

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

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

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

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

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

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

  Gets the minimal sampling frequency step size increment value of the device.
- virtual int32_t GetChannelsInBlock (unsigned int virtualDevice)

  Get the number of 16 bit datawords which will be collected per sample frame, use after the device is configured.
- virtual uint32_t GetChannelLayout ([System::Runtime::InteropServices::Out]int% AnalogChannels, [System::Runtime::InteropServices::Out]int% DigitalChannels, [System::Runtime::InteropServices::Out]int% ChecksumChannels, [System::Runtime::InteropServices::Out]int% TimestampChannels, [System::↩ Runtime::InteropServices::Out]int% ChannelsInBlock, unsigned int virtualDevice)
- virtual void SendStartDacq ()

  Start sampling.
- virtual void SendStartDacq (int VirtualDacqMap)

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

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

  Stop sampling.
- virtual void SendStopDacq (int VirtualDacqMap)

  Stop sampling.

Parameters

VirtualDacqMap	

- virtual void SendStopStgAndDacq (uint32_t trigger_map, int VirtualDacqMap)

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

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

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

  Start the data acquisition thread.
- virtual void StartLoop (int32_t timeout, int32_t numSubmittedUsbBuffers, int32_t numUsbBuffers, int32_↩ t packetsInUrb)

  Start the data acquisition thread.
- virtual void StartLoop (int32_t timeout, int32_t numSubmittedUsbBuffers, int32_t numUsbBuffers, int32_↩ t packetsInUrb, uint32_t virtualDevice)

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

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

  Start the data acquisition thread and sampling.
- virtual void StartDacq (int32_t timeout, int32_t numSubmittedUsbBuffers, int32_t numUsbBuffers, int32_↩ t packetsInUrb)

  Start the data acquisition thread and sampling.
- virtual void StartDacq (int32_t timeout, int32_t numSubmittedUsbBuffers, int32_t numUsbBuffers, int32_↩ t packetsInUrb, uint32_t virtualDevice)

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

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

  Stop the data acquisition thread and sampling.
- virtual uint32_t SetPoti (uint32_t channel, uint32_t value, bool write_nvram)
- virtual uint32_t GetPoti (uint32_t channel, [System::Runtime::InteropServices::Out]uint32_t% value)
- virtual CFilterPropertyNet ^ GetFilterProperty (DacqGroupChannelEnumNet GroupID, unsigned int index)
- virtual array< CFilterPropertyNet^> ^ CMcsUsbDacqNet::GetFilterProperties (DacqGroupChannelEnum↩ Net 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, Sample↩ SizeNet 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, Sample↩ SizeNet 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 sample-size, 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 sample-size, SampleDstSizeNet sampleDstSize, int ChannelsInBlock)
- virtual void SetSelectedChannelsQueue (array< bool >^ selectedChannels, int queuesize, int threshold, SampleSizeNet samplesize, int ChannelsInBlock)

  Create a common FIFO queue for all channels. Data in callback will be a list per channel. Use ChannelBlock_↩ ReadFramesDict... with handle = 0 to read the data.
- virtual void SetSelectedChannelsQueue (array< bool >^ selectedChannels, int queuesize, int threshold, SampleSizeNet samplesize, SampleDstSizeNet sampleDstSize, int ChannelsInBlock)
- virtual uint32_t ChannelBlock_AvailFrames (int handle)

  Get the number of sample frames already available in the FIFO.
- virtual uint32_t ChannelBlock_AvailFrames (int handle, int queue)
- virtual array< uint16_t > ^ ChannelBlock_ReadFramesUI16 (int handle, int frames, [System::Runtime::↩ InteropServices::Out]int % frames_ret)

  Read data from a FIFO queue in uint16_t data format
- virtual uint32_t ChannelBlock_ReadFramesUI16 (int handle, array< uint16_t >^ buffer, int frames_pos, int frames, [System::Runtime::InteropServices::Out]int % frames_ret)

  Read data from a FIFO queue in uint16_t data format
- virtual array< int16_t > ^ ChannelBlock_ReadFramesI16 (int handle, int frames, [System::Runtime::↩ InteropServices::Out]int % frames_ret)

  Read data from a FIFO queue in int16_t data format
- virtual uint32_t ChannelBlock_ReadFramesI16 (int handle, array< int16_t >^ buffer, int frames_pos, int frames, [System::Runtime::InteropServices::Out]int % frames_ret)

  Read data from a FIFO queue in int16_t data format
- virtual array< uint32_t > ^ ChannelBlock_ReadFramesUI32 (int handle, int frames, [System::Runtime::↩ InteropServices::Out]int % frames_ret)

  Read data from a FIFO queue in uint32_t data format
- virtual uint32_t ChannelBlock_ReadFramesUI32 (int handle, array< uint32_t >^ buffer, int frames_pos, int frames, [System::Runtime::InteropServices::Out]int % frames_ret)

  Read data from a FIFO queue in uint32_t data format
- virtual array< int32_t > ^ ChannelBlock_ReadFramesI32 (int handle, int frames, [System::Runtime::↩ InteropServices::Out]int % frames_ret)

  Read data from a FIFO queue in uint32_t data format
- virtual uint32_t ChannelBlock_ReadFramesI32 (int handle, array< int32_t >^ buffer, int frames_pos, int frames, [System::Runtime::InteropServices::Out]int % frames_ret)

  Read data from a FIFO queue in 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
- OnError^ ErrorEvent

Additional Inherited Members

11.49.1 Detailed Description

Base class for data acquisition devices.

11.49.2 Constructor & Destructor Documentation

11.49.2.1 CMcsUsbDacqNet() `CMcsUsbDacqNet ( )`

11.49.2.2 ∼CMcsUsbDacqNet() `∼CMcsUsbDacqNet ( )`

11.49.3 Member Function Documentation

11.49.3.1 AddSelectedChannelsQueue() [1/4]

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

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

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

Parameters

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>^> ^ Channel↩
Block_ReadAsFrameArrayI16 (
            int handle,
            int frames,
            [System::Runtime::InteropServices::Out] int % frames_ret )  [virtual]
```

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

Parameters

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>^> ^ Channel↩
Block_ReadAsFrameArrayI16 (
            int handle,
            int queue,
            int frames,
            [System::Runtime::InteropServices::Out] int % frames_ret )  [virtual]
```

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

Parameters

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>^> ^ Channel↩ Block_ReadAsFrameArrayI32 (

```
            int handle,
            int frames,
            [System::Runtime::InteropServices::Out] int % frames_ret )  [virtual]
```

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

Parameters

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>^> ^ Channel↩
Block_ReadAsFrameArrayI32 (
            int handle,
            int queue,
            int frames,
            [System::Runtime::InteropServices::Out] int % frames_ret )  [virtual]
```

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

Parameters

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 uint32_t ChannelBlock_ReadFramesI16 (
            int handle,
            array< int16_t >^ buffer,
            int frames_pos,
            int frames,
            [System::Runtime::InteropServices::Out] int % frames_ret )  [virtual]
```

Read data from a FIFO queue in int16_t data format

Parameters

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.

Returns

Error Status. 0 on success.

11.49.3.20 ChannelBlock_ReadFramesI16() [2/2]

```
virtual array<int16_t> ^ ChannelBlock_Read↩
FramesI16 (
            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 uint32_t ChannelBlock_ReadFramesI32 (
        int handle,
        array< int32_t >^ buffer,
        int frames_pos,
        int frames,
        [System::Runtime::InteropServices::Out] int % frames_ret )  [virtual]
```

Read data from a FIFO queue in uint32_t data format

Parameters

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.

Returns

Error Status. 0 on success.

11.49.3.22 ChannelBlock_ReadFramesI32() [2/2]

```
virtual array<int32_t> ^ ChannelBlock_Read↵
FramesI32 (
            int handle,
            int frames,
            [System::Runtime::InteropServices::Out] int % frames_ret )  [virtual]
```

Read data from a FIFO queue in uint32_t data format

Parameters

handle	Handle of the FIFO queue. Either zero when the SetSelectedData call was used or the channel number.

Parameters

frames	Number of sample frames to read.

Parameters

frames_ret	Number of sample frames which were read, might be smaller than frames.

11.49.3.23 ChannelBlock_ReadFramesUI16() [1/2]

```
virtual uint32_t ChannelBlock_ReadFramesUI16 (
            int handle,
            array< uint16_t >^ buffer,
            int frames_pos,
            int frames,
            [System::Runtime::InteropServices::Out] int % frames_ret )  [virtual]
```

Read data from a FIFO queue in uint16_t data format

Parameters

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.

Returns

Error Status. 0 on success.

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 uint32_t ChannelBlock_ReadFramesUI32 (
            int handle,
            array< uint32_t >^ buffer,
            int frames_pos,
            int frames,
            [System::Runtime::InteropServices::Out] int % frames_ret )  [virtual]
```

Read data from a FIFO queue in uint32_t data format

Parameters

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.

Returns

Error Status. 0 on success.

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^> ^ CMcs↩ UsbDacqNet::GetFilterProperties ( DacqGroupChannelEnumNet GroupID ) [virtual]`

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

Gets the adapter which is connected to the MEA2100 device.

Returns

AdapterTypeEnumNet which enumerates the possible adapters.

11.49.3.30 GetAdcDataFormat() `virtual uint32_t GetAdcDataFormat ( uint32_t virtualDevice ) [virtual]`

Gets the ADC data format, 16 means 16 bits, 24 means 24 bits, 32 means 32 bits.

Returns

The data format in bits.

11.49.3.31 GetAdcZero()

```
virtual uint32_t GetAdcZero (
    uint32_t virtualDevice,
    DacqGroupChannelEnumNet group,
    [System::Runtime::InteropServices::Out] int% adcz ) [virtual]
```

11.49.3.32 GetAnalogValueUnit()

```
virtual uint32_t GetAnalogValueUnit (
    uint32_t virtualDevice,
    DacqGroupChannelEnumNet group,
    [System::Runtime::InteropServices::Out] AnalogUnitEnumNet% unit ) [virtual]
```

11.49.3.33 GetChannelDataFillSize()

```
int GetChannelDataFillSize ( )
```

11.49.3.34 GetChannelLayout()

```
virtual uint32_t GetChannelLayout (
    [System::Runtime::InteropServices::Out] int% AnalogChannels,
    [System::Runtime::InteropServices::Out] int% DigitalChannels,
    [System::Runtime::InteropServices::Out] int% ChecksumChannels,
    [System::Runtime::InteropServices::Out] int% TimestampChannels,
    [System::Runtime::InteropServices::Out] int% ChannelsInBlock,
    unsigned int virtualDevice ) [virtual]
```

11.49.3.35 GetChannelsInBlock()

```
virtual int32_t GetChannelsInBlock (
    unsigned int virtualDevice ) [virtual]
```

Get the number of 16 bit datawords which will be collected per sample frame, use after the device is configured.

Returns

Number of 16 bit datawords per sample frame.

11.49.3.36 GetDataFormat()

```
virtual uint32_t GetDataFormat (
    uint32_t virtualDevice,
    DacqGroupChannelEnumNet group,
    [System::Runtime::InteropServices::Out] int% numberOfBits ) [virtual]
```

11.49.3.37 GetDataMode()

```
virtual DataModeEnumNet GetDataMode (
    unsigned int virtualDevice ) [virtual]
```

Gets the data mode, can be 16, 24 or 32bit, all signed or unsigned on the MEA2100 device.

Parameters

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 GetFilterProperty()

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

11.49.3.44 GetGroupChannelDataI16()

```
virtual System::Collections::Generic::Dictionary<int,
array<int16_t>^> ^ GetGroupChannelDataI16 (
            DacqGroupChannelEnumNet group,
            int frames,
            [System::Runtime::InteropServices::Out] int % frames_ret ) [virtual]
```

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

Parameters

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.

11.49.3.45 GetGroupChannelDataI32()

```
virtual System::Collections::Generic::Dictionary<int,
array<int32_t>^> ^ GetGroupChannelDataI32 (
            DacqGroupChannelEnumNet group,
            int frames,
            [System::Runtime::InteropServices::Out] int % frames_ret ) [virtual]
```

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

Parameters

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.

11.49.3.46 GetGroupChannelDataUI16()

```
virtual System::Collections::Generic::Dictionary<int,
array<uint16_t>^> ^ GetGroupChannelDataUI16 (
            DacqGroupChannelEnumNet group,
            int frames,
            [System::Runtime::InteropServices::Out] int % frames_ret ) [virtual]
```

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

Parameters

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.

11.49.3.47 GetGroupChannelDataUI32()

```
virtual System::Collections::Generic::Dictionary<int,
array<uint32_t>^> ^ GetGroupChannelDataUI32 (
            DacqGroupChannelEnumNet group,
int frames,
[System::Runtime::InteropServices::Out] int % frames_ret )  [virtual]
```

Read data from a FIFO queue in uint32_t data format, that contains subqueues, each populates an entry in the dictionary by hardware channel number

Parameters

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.

11.49.3.48 GetHardwareMaxRange()

```
virtual uint32_t GetHardwareMaxRange (
uint32_t virtualDevice,
DacqGroupChannelEnumNet group,
[System::Runtime::InteropServices::Out] int% r,
[System::Runtime::InteropServices::Out] int% rUnit )  [virtual]
```

11.49.3.49 GetHardwareMinRange()

```
virtual uint32_t GetHardwareMinRange (
uint32_t virtualDevice,
DacqGroupChannelEnumNet group,
[System::Runtime::InteropServices::Out] int% r,
[System::Runtime::InteropServices::Out] int% rUnit )  [virtual]
```

11.49.3.50 GetMaxSamplingFrequency()

```
virtual uint32_t GetMaxSamplingFrequency (
int virtualDevice )  [virtual]
```

Gets the maximal sampling frequency of the device.

Returns

Sampling frequency in Hz.

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

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

Returns

MeaLayoutEnumNet which enumerates the MEA types.

11.49.3.52 GetMinSamplingFrequencyStepsize() `virtual uint32_t GetMinSamplingFrequencyStepsize ( ) [virtual]`

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

Returns

Sampling frequency step size in Hz.

11.49.3.53 GetNumberOfDataBits()

```
virtual uint32_t GetNumberOfDataBits (
            uint32_t virtualDevice,
            DacqGroupChannelEnumNet group,
            [System::Runtime::InteropServices::Out] int% numberOfBits ) [virtual]
```

Get the real number of data bits.

This value may be different from the value returned by GetDataFormat, e.g. in MC_Card the data are shifted 2 bits so the real number is 14 while the data format is 16 bits

11.49.3.54 GetPoti()

```
virtual uint32_t GetPoti (
            uint32_t channel,
            [System::Runtime::InteropServices::Out] uint32_t% value ) [virtual]
```

11.49.3.55 GetResolutionPerDigit()

```
virtual uint32_t GetResolutionPerDigit (
            uint32_t virtualDevice,
            DacqGroupChannelEnumNet group,
            [System::Runtime::InteropServices::Out] int% res,
            [System::Runtime::InteropServices::Out] int% resUnit ) [virtual]
```

11.49.3.56 GetSamplerate()

```
virtual int32_t GetSamplerate (
            unsigned int virtualDevice ) [virtual]
```

Gets the sampling frequency of the device.

Returns

Sampling frequency in Hz.

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

11.49.3.58 GetVoltageRangeInMicroVolt() `virtual int32_t GetVoltageRangeInMicroVolt ( unsigned int virtualDevice ) [virtual]`

Gets the currently selected voltage range on devices which support multiple voltage ranges.

Returns

The Voltage Range in uV.

11.49.3.59 GetVoltageRangeInMilliVolt() `virtual int32_t GetVoltageRangeInMilliVolt ( ) [virtual]`

Gets the currently selected voltage range on devices which support multiple voltage ranges.

Returns

The rounded Voltage Range in mV.

11.49.3.60 HWInfo() `CHWInfo ^ HWInfo ( )`

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

Start sampling.

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

Start sampling.

Parameters

VirtualDacqMap	

11.49.3.63 SendStartStgAndDacq()

```
virtual void SendStartStgAndDacq (
            uint32_t trigger_map,
            int VirtualDacqMap )  [virtual]
```

Start sampling together with the STG.

Parameters

trigger_map	
VirtualDacqMap	

11.49.3.64 SendStopDacq() [1/2]

```
virtual void SendStopDacq ( )  [virtual]
```

Stop sampling.

11.49.3.65 SendStopDacq() [2/2]

```
virtual void SendStopDacq (
            int VirtualDacqMap )  [virtual]
```

Stop sampling.

Parameters

VirtualDacqMap	

11.49.3.66 SendStopStgAndDacq()

```
virtual void SendStopStgAndDacq (
            uint32_t trigger_map,
            int VirtualDacqMap )  [virtual]
```

Stop sampling together with the STG.

Parameters

trigger_map	

11.49.3.67 SendStopStgAndDacqWithOptions()

```
virtual void SendStopStgAndDacqWithOptions (
            uint32_t trigger_map,
            int VirtualDacqMap,
            int options )  [virtual]
```

Stop sampling together with the STG and options.

Parameters

trigger_map	

Parameters

options	

Parameters

VirtualDacqMap	

11.49.3.68 SetDataMode() `virtual void SetDataMode (`

```
            DataModeEnumNet dataMode,
            unsigned int virtualDevice )  [virtual]
```

Sets the data mode, can be 16, 24 or 32bit, all signed or unsigned on the MEA2100 device.

Parameters

dataMode	DataModeEnumNet enumerates the possible data modes.
virtualDevice	Virtual device to use.

11.49.3.69 SetDigitalSource() `[1/5]` `void SetDigitalSource (`

```
            DigitalTargetEnumNet digitaltarget,
            int32_t NrChannel,
            DigitalSource< digitalsourceenum >^ source,
            int bitnumber_offset )
```

Sets the function/source of an digital output bit.

This is the templated generic implementation.

Parameters

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.

11.49.3.70 SetDigitalSource() [2/5]

```
void SetDigitalSource (
    DigitalTargetEnumNet digitaltarget,
    int32_t NrChannel,
    DigitalSourceEnumNet source,
    int bitnumber_offset )
```

Sets the function/source of an digital output bit.

This overload is for the MEA2100 device.

Parameters

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() [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.

11.49.3.72 SetDigitalSource() [4/5]

```
void SetDigitalSource (
    DigitalTargetEnumNet digitaltarget,
int32_t NrChannel,
SCUDigitalSourceEnumNet source,
int bitnumber_offset )
```

Sets the function/source of an digital output bit.

This overload is for the SCU device.

Parameters

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() [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.

11.49.3.74 SetPoti()

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

11.49.3.75 SetSamplerate()

```
virtual void SetSamplerate (
    int32_t rate,
    unsigned int oversample,
    unsigned int virtualDevice ) [virtual]
```

Sets the sampling frequency of the device.

Parameters

rate	Sampling frequency in Hz.

11.49.3.76 SetSelectedChannels() [1/4]

```
virtual void SetSelectedChannels (
            array< bool >^ selectedChannels,
            int queuesize,
            int threshold,
            SampleSizeNet samplesize,
            int ChannelsInBlock )  [virtual]
```

Create a FIFO queue per channel. Each channel will have its own FIFO and Callback function.

When using a 32bit sample size, the number obtained from GetChannelsInBlock must be 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.

11.49.3.77 SetSelectedChannels() [2/4]

```
virtual void SetSelectedChannels (
    array< bool >^ selectedChannels,
    int queuesize,
    int threshold,
    SampleSizeNet samplesize,
    SampleDstSizeNet sampleDstSize,
    int ChannelsInBlock ) [virtual]
```

11.49.3.78 SetSelectedChannels() [3/4]

```
virtual void SetSelectedChannels (
    int nChannels,
    int queuesize,
    int threshold,
    SampleSizeNet samplesize,
    int ChannelsInBlock ) [virtual]
```

Create a FIFO queue per channel. Each channel will have its own FIFO and Callback function.

When using a 32bit sample size, the number obtained from GetChannelsInBlock must be 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.

11.49.3.79 SetSelectedChannels() [4/4]

```
virtual void SetSelectedChannels (
    int nChannels,
    int queuesize,
    int threshold,
    SampleSizeNet samplesize,
    SampleDstSizeNet sampleDstSize,
    int ChannelsInBlock ) [virtual]
```

11.49.3.80 SetSelectedChannelsQueue() [1/4]

```
virtual void SetSelectedChannelsQueue (
    array< bool >^ selectedChannels,
    int queuesize,
    int threshold,
    SampleSizeNet samplesize,
    int ChannelsInBlock ) [virtual]
```

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

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

Parameters

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.

11.49.3.81 SetSelectedChannelsQueue() [2/4]

```
virtual void SetSelectedChannelsQueue (
        array< bool >^ selectedChannels,
        int queuesize,
        int threshold,
        SampleSizeNet samplesize,
        SampleDstSizeNet sampleDstSize,
        int ChannelsInBlock )  [virtual]
```

11.49.3.82 SetSelectedChannelsQueue() [3/4]

```
virtual void SetSelectedChannelsQueue (
        int nChannels,
        int queuesize,
        int threshold,
        SampleSizeNet samplesize,
        int ChannelsInBlock )  [virtual]
```

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

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

Parameters

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.

11.49.3.83 SetSelectedChannelsQueue() [4/4]

```
virtual void SetSelectedChannelsQueue (
        int nChannels,
        int queuesize,
        int threshold,
        SampleSizeNet samplesize,
        SampleDstSizeNet sampleDstSize,
        int ChannelsInBlock ) [virtual]
```

11.49.3.84 SetSelectedData() [1/4]

```
virtual void SetSelectedData (
        array< bool >^ selectedChannels,
        int queuesize,
        int threshold,
        SampleSizeNet samplesize,
        int ChannelsInBlock ) [virtual]
```

Create a common FIFO queue for all channels. Use handle = 0 in the ChannelBlock_ReadFrames... functions.

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

Parameters

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.

11.49.3.85 SetSelectedData() [2/4]

```
virtual void SetSelectedData (
    array< bool >^ selectedChannels,
    int queuesize,
    int threshold,
    SampleSizeNet samplesize,
    SampleDstSizeNet sampleDstSize,
    int ChannelsInBlock ) [virtual]
```

11.49.3.86 SetSelectedData() [3/4]

```
virtual void SetSelectedData (
    int nChannels,
    int queuesize,
    int threshold,
    SampleSizeNet samplesize,
    int ChannelsInBlock ) [virtual]
```

Create a common FIFO queue for all channels. Use handle = 0 in the ChannelBlock_ReadFrames... functions.

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

Parameters

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.

11.49.3.87 SetSelectedData() [4/4]

```
virtual void SetSelectedData (
    int nChannels,
    int queuesize,
    int threshold,
    SampleSizeNet samplesize,
    SampleDstSizeNet sampleDstSize,
    int ChannelsInBlock )  [virtual]
```

11.49.3.88 SetupGroupDacqQueue() [1/2]

```
void SetupGroupDacqQueue (
    int queuesize,
    int threshold )
```

11.49.3.89 SetupGroupDacqQueue() [2/2]

```
void SetupGroupDacqQueue (
    int queuesize,
    int threshold,
    unsigned int virtualDevice )
```

11.49.3.90 SetVoltageRangeByIndex()

```
virtual void SetVoltageRangeByIndex (
    int32_t voltageRangeIndex,
    unsigned int virtualDevice )  [virtual]
```

Sets the voltage range on devices which support multiple voltage ranges.

Parameters

voltageRangeIndex	Voltage Range to use as index, smaller values are larger voltage ranges.

11.49.3.91 SetVoltageRangeInMicroVolt()

```
virtual void SetVoltageRangeInMicroVolt (
            int32_t voltageRange,
            unsigned int virtualDevice )  [virtual]
```

Sets the voltage range on devices which support multiple voltage ranges.

Parameters

voltageRange	Voltage Range to use in µV.

This replaces SetVoltageRange, where the value of the range was in mV!

11.49.3.92 StartDacq() [1/4]

```
virtual void StartDacq ( )  [virtual]
```

Start the data acquisition thread and sampling.

11.49.3.93 StartDacq() [2/4]

```
virtual void StartDacq (
            int32_t timeout )  [virtual]
```

Start the data acquisition thread and sampling.

Parameters

timeout	Timeout in ms.

11.49.3.94 StartDacq() [3/4]

```
virtual void StartDacq (
            int32_t timeout,
            int32_t numSubmittedUsbBuffers,
            int32_t numUsbBuffers,
            int32_t packetsInUrb )  [virtual]
```

Start the data acquisition thread and sampling.

Parameters

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.

11.49.3.95 StartDacq() [4/4]

```
virtual void StartDacq (
        int32_t timeout,
        int32_t numSubmittedUsbBuffers,
        int32_t numUsbBuffers,
        int32_t packetsInUrb,
        uint32_t virtualDevice )  [virtual]
```

Start the data acquisition thread and sampling.

Parameters

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.

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

Start the data acquisition thread.

11.49.3.97 StartLoop() [2/4]

```
virtual void StartLoop (
        int32_t timeout ) [virtual]
```

Start the data acquisition thread.

Parameters

timeout	Timeout in ms.

11.49.3.98 StartLoop() [3/4]

```
virtual void StartLoop (
        int32_t timeout,
        int32_t numSubmittedUsbBuffers,
        int32_t numUsbBuffers,
        int32_t packetsInUrb ) [virtual]
```

Start the data acquisition thread.

Parameters

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.

11.49.3.99 StartLoop() [4/4]

```
virtual void StartLoop (
            int32_t timeout,
            int32_t numSubmittedUsbBuffers,
            int32_t numUsbBuffers,
            int32_t packetsInUrb,
            uint32_t virtualDevice )  [virtual]
```

Start the data acquisition thread.

Parameters

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.

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

Stop the data acquisition thread and sampling.

11.49.3.101 StopDacq() `[2/2]` `virtual void StopDacq ( uint32_t virtualDevice ) [virtual]`

Stop the data acquisition thread and sampling.

Parameters

virtualDevice	Virtual Device to start.

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

11.49.4 Member Data Documentation

11.49.4.1 Error_Callback_Aquisition_Stopped `const int Error_Callback_Aquisition_Stopped = 0x200 [static]`

11.49.4.2 Error_Callback_Data_lost `const int Error_Callback_Data_lost = 5 [static]`

11.49.4.3 Error_Callback_Frames_Lost `const int Error_Callback_Frames_Lost = 4 [static]`

11.49.4.4 Error_Callback_Packet_Error `const int Error_Callback_Packet_Error = 1 [static]`

11.49.4.5 Error_Callback_Queue_Full `const int Error_Callback_Queue_Full = 0x100 [static]`

11.49.4.6 Error_Callback_RingQueue_Full `const int Error_Callback_RingQueue_Full = 3 [static]`

11.49.5 Property Documentation

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

The sampling frequency of the device in Hz.

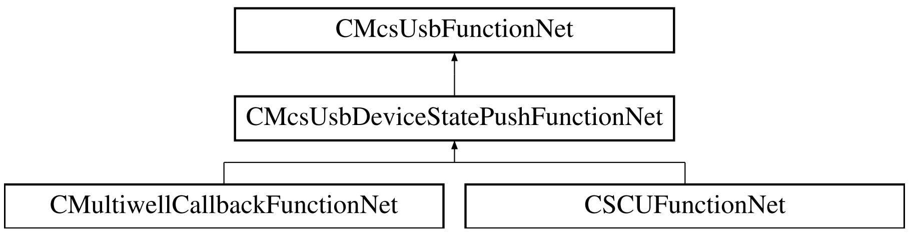
11.49.6 Event Documentation

11.49.6.1 ChannelDataEvent `OnChannelData^ ChannelDataEvent`

11.49.6.2 ErrorEvent `OnError^ ErrorEvent`

11.50 CMcsUsbDeviceStatePushFunctionNet Class Reference

Inheritance diagram for CMcsUsbDeviceStatePushFunctionNet:


CMcsUsbFunctionNet
CMcsUsbDeviceStatePushFunctionNet
CMultiwellCallbackFunctionNet
CSCUFunctionNet

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

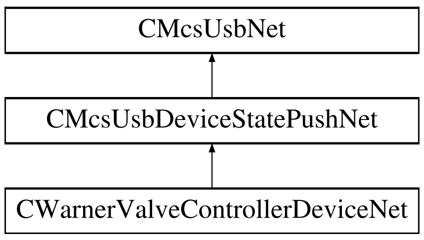
11.50.2.1 TriggerStatus() `void TriggerStatus ( )`

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

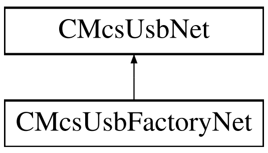
11.51.2.1 TriggerStatus() `void TriggerStatus ( )`

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, CFirmwareDestination↩ Net 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> ^ ReadBlockFromIFBGlobalE↩
Eprom (
    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

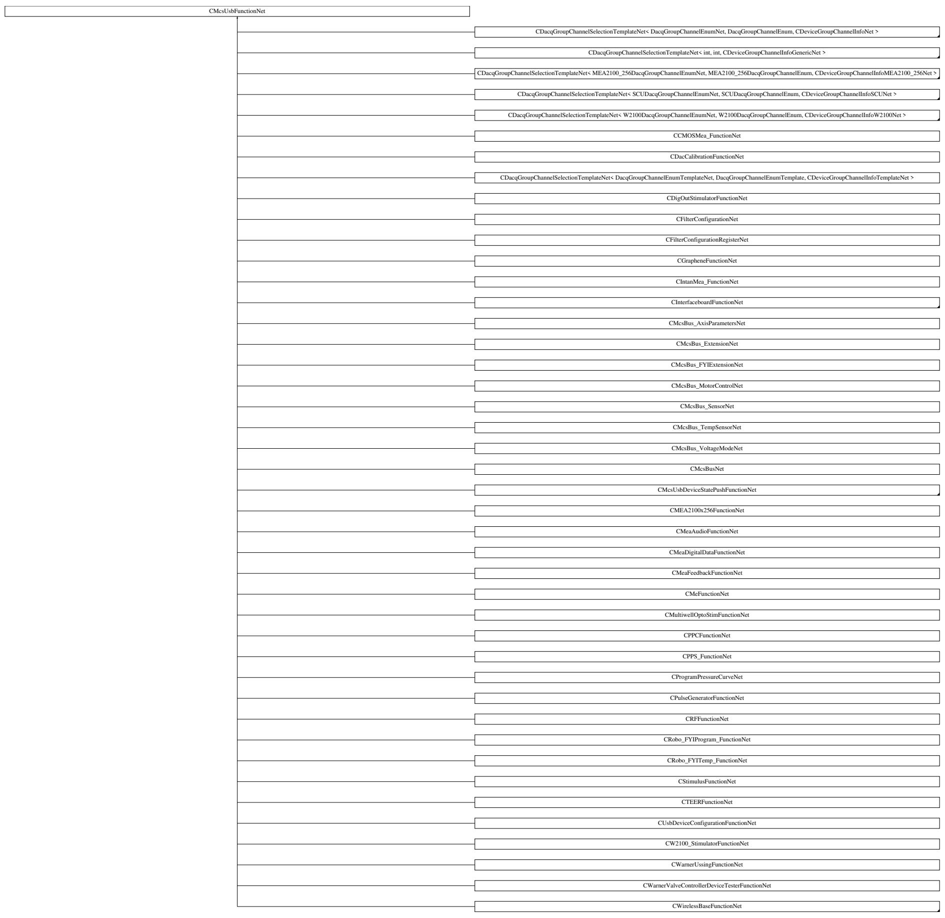
```
const uint32_t FX3MCSDataIFB2ImageOffset = 0xC  [static]
```

11.52.3.5 FX3MCSDataVersionOffset

```
const uint32_t FX3MCSDataVersionOffset = 0x8  [static]
```

11.53 CMcsUsbFunctionNet Class Reference

Inheritance diagram for CMcsUsbFunctionNet:


CMcsUsbFunctionNet
CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumNet, DacqGroupChannelEnum, CDeviceGroupChannelInfoNet >
CDacqGroupChannelSelectionTemplateNet< int, int, CDeviceGroupChannelInfoGenericNet >
CDacqGroupChannelSelectionTemplateNet< MEA2100_256DacqGroupChannelEnumNet, MEA2100_256DacqGroupChannelEnum, CDeviceGroupChannelInfoMEA2100_256Net >
CDacqGroupChannelSelectionTemplateNet< SCUDacqGroupChannelEnumNet, SCUDacqGroupChannelEnum, CDeviceGroupChannelInfoSCUNet >
CDacqGroupChannelSelectionTemplateNet< W2100DacqGroupChannelEnumNet, W2100DacqGroupChannelEnum, CDeviceGroupChannelInfoW2100Net >
CCMOSMea_FunctionNet
CDacCalibrationFunctionNet
CDacqGroupChannelSelectionTemplateNet< DacqGroupChannelEnumTemplateNet, DacqGroupChannelEnumTemplate, CDeviceGroupChannelInfoTemplateNet >
CDigOutStimulatorFunctionNet
CFilterConfigurationNet
CFilterConfigurationRegisterNet
CGrapheneFunctionNet
CIntanMea_FunctionNet
CInterfaceboardFunctionNet
CMcsBus_AxisParametersNet
CMcsBus_ExtensionNet
CMcsBus_FYIExtensionNet
CMcsBus_MotorControlNet
CMcsBus_SensorNet
CMcsBus_TempSensorNet
CMcsBus_VoltageModeNet
CMcsBusNet
CMcsUsbDeviceStatePushFunctionNet
CMEA2100x256FunctionNet
CMeaAudioFunctionNet
CMeaDigitalDataFunctionNet
CMeaFeedbackFunctionNet
CMeFunctionNet
CMultiwellOptoStimFunctionNet
CPPCFunctionNet
CPPS_FunctionNet
CProgramPressureCurveNet
CPulseGeneratorFunctionNet
CRFFunctionNet
CRobo_FYIProgram_FunctionNet
CRobo_FYITemp_FunctionNet
CStimulusFunctionNet
CTEERFunctionNet
CUsbDeviceConfigurationFunctionNet
CW2100_StimulatorFunctionNet
CWarnerUsingFunctionNet
CWarnerValveControllerDeviceTesterFunctionNet
CWirelessBaseFunctionNet

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]`

```
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
```

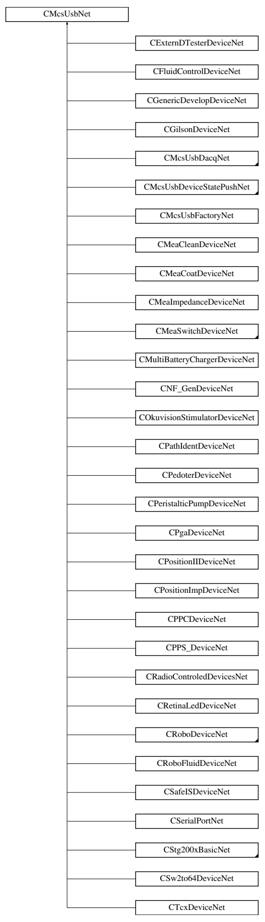
11.56.5.2 DeviceRemoval

```
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]`

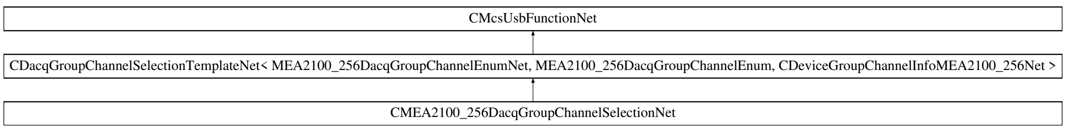
11.57.5 Property Documentation

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:


CMcsUsbFunctionNet
CDacqGroupChannelSelectionTemplateNet< MEA2100_256DacqGroupChannelEnumNet, MEA2100_256DacqGroupChannelEnum, CDeviceGroupChannelInfoMEA2100_256Net >
CMEA2100_256DacqGroupChannelSelectionNet

Public Member Functions

- CMEA2100_256DacqGroupChannelSelectionNet (CMcsUsbNet^ mcsusb)

Additional Inherited Members

11.59.1 Constructor & Destructor Documentation

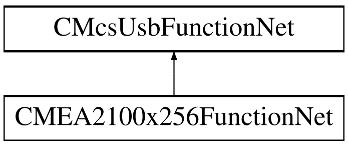
11.59.1.1 CMEA2100_256DacqGroupChannelSelectionNet() `CMEA2100_256DacqGroupChannelSelectionNet (`

```
            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^ pME↵A2100x256FunctionPointerContainer)

  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 D↵AC channels available per well is Mcs::Usb::CStg200xBasicNet::GetNumberOfAnalogChannels divided by Mcs::Usb::CStg200xBasicNet::GetNumberOfStimulationSourcesPerElectrode.
- void SetLayoutConfiguration (StimulationLayoutConfigurationEnumNet LayoutConfiguration)

  Sets the stimulation layout configuration. Can be single well, 6-well or 9-well. The number of D↵AC channels available per well is Mcs::Usb::CStg200xBasicNet::GetNumberOfAnalogChannels divided by Mcs::Usb::CStg200xBasicNet::GetNumberOfStimulationSourcesPerElectrode.

Additional Inherited Members

11.60.1 Detailed Description

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

11.60.2 Constructor & Destructor Documentation

11.60.2.1 CMEA2100x256FunctionNet() `[1/2]` `CMEA2100x256FunctionNet (`

```
CMcsUsbNet^ 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 D←AC channels available per well is Mcs::Usb::CStg200xBasicNet::GetNumberOfAnalogChannels divided by Mcs::Usb::CStg200xBasicNet::GetNumberOfStimulationSourcesPerElectrode.

Returns

The currently active stimulation layout configuration.

11.60.3.2 SetLayoutConfiguration()

`void SetLayoutConfiguration ( StimulationLayoutConfigurationEnumNet LayoutConfiguration )`

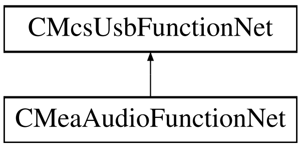
Sets the stimulation layout configuration. Can be single well, 6-well or 9-well. The number of D←AC channels available per well is Mcs::Usb::CStg200xBasicNet::GetNumberOfAnalogChannels divided by Mcs::Usb::CStg200xBasicNet::GetNumberOfStimulationSourcesPerElectrode.

Parameters

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.

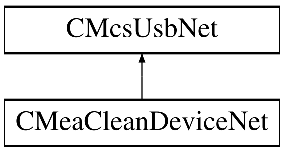
Returns

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.

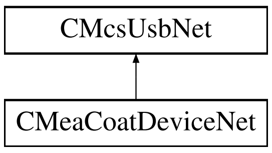
11.62.3.13 Stop() `void Stop ( )`

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.

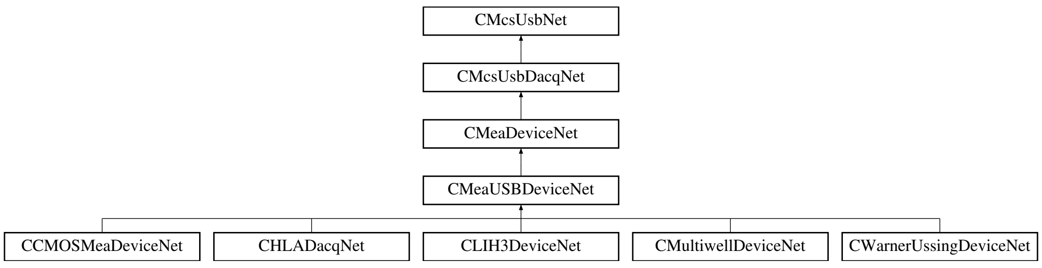
11.63.3.19 Stop() `void Stop ( )`

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]`

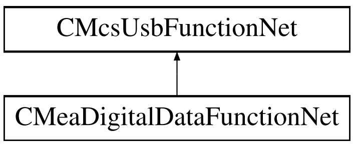
11.64.4.6 MeFunctionNet `CMeFunctionNet^ MeFunctionNet [get]`

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

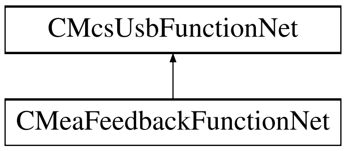
digital_value	Value to set.

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()

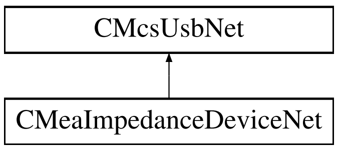
```
void FeedbackSetSpikeDetectorThreshold (
        unsigned short position,
        unsigned short sourcechannel,
        unsigned short resultchannel,
        unsigned short trigger,
        unsigned short totzeit,
        int threshold1,
        int threshold2,
        short slope )
```

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]`

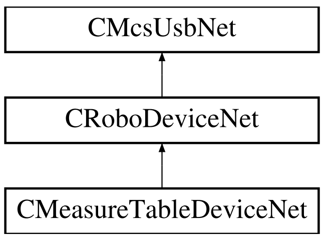
11.67.2.6 SetImpedanceTestFrequency() `virtual void SetImpedanceTestFrequency ( unsigned int TestFrequency_Hertz ) [virtual]`

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 )`

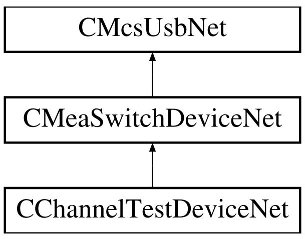
11.68.3 Property Documentation

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.

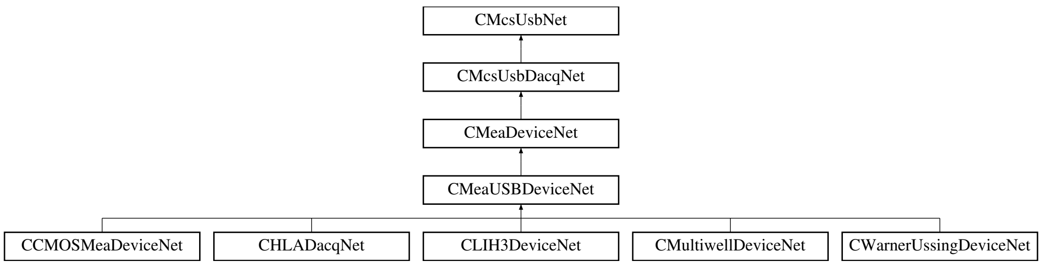
11.69.3.5 SetPatternBool() `void SetPatternBool ( array< bool >^ pattern )`

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 ( )
```

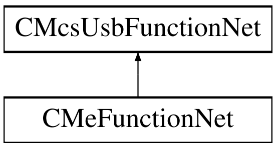
Initializes a new instance of CMeaDeviceNet class.

11.70.2.3 ∼CMeaUSBDeviceNet()

```
~CMeaUSBDeviceNet ( )
```

11.71 CMeFunctionNet Class Reference

Inheritance diagram for CMeFunctionNet:



Public Member Functions

- CMeFunctionNet (CMcsUsbNet^ mcsusb, CMcsUsbFunctionPointerContainer^ meFunctionPointer↩ Container)

  Initializes a new instance of the CDacCalibrationFunctionNet class.
- CMeFunctionNet (CMcsUsbNet^ mcsusb)
- virtual ∼CMeFunctionNet (void)
- !CMeFunctionNet (void)
- void SetTransformer (unsigned int index, bool onoff)

Additional Inherited Members

11.71.1 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

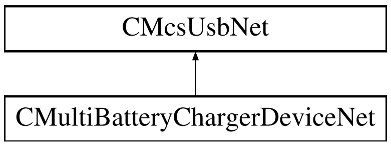
11.71.3.1 SetTransformer()

```
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: ?Ah
- uint32_t GetChargeCapacity (uint32_t NrChannel)

  gets the charge capacity; unit: ?Ah
- uint32_t GetBatteryVoltage (uint32_t NrChannel)

  gets the battery voltage; unit: mV
- uint32_t GetChannels ()

  gets number of channels
- void SetRatedCapacityVolatile (uint32_t NrChannel, MbcRatedCapacityEnumNet NewRatedCapacity)

  sets the rated capacity (i.e. charge current) without storing it persistently
- void SetChargingMode (uint32_t NrChannel, MbcChargingModeEnumNet NewOperatingMode)

  sets the charging mode: StorageCharge, LowCurrentCharge and HighCurrentCharge
- MbcChargingModeEnumNet GetChargingMode (uint32_t NrChannel)

  gets the charging mode: StorageCharge, LowCurrentCharge and HighCurrentCharge
- MbcChannelStateEnumNet GetChannelState (uint32_t NrChannel)

  gets the channel state: IdleNoBattery, IdleChargeFinished, CapacityTestPreCharge, CapacityTestDischarge, StorageCharge, LowCurrentCharge, HighCurrentCharge
- void CapacityTest (uint32_t NrChannel)

  start capacity test on channel
- void ChannelReset (uint32_t NrChannel)

  cancel charging and capacity test functions; check if battery is connected

- void SetChargingPCoefficient (uint32_t pCoefficient)

  sets the p-coefficient for charging in mA/V / nominal charging current
- uint32_t GetChargingPCoefficient ()

  gets the p-coefficient for charging in mA/V / nominal charging current
- void SetRatedCapacity (uint32_t NrChannel, MbcRatedCapacityEnumNet NewRatedCapacity)

  sets the rated capacity
- MbcRatedCapacityEnumNet GetRatedCapacity (uint32_t NrChannel)

  gets the rated capacity

Additional Inherited Members

11.72.1 Detailed Description

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

11.72.2 Constructor & Destructor Documentation

11.72.2.1 CMultiBatteryChargerDeviceNet() `CMultiBatteryChargerDeviceNet ( )`

Initializes a new instance of the CMultiBatteryChargerDeviceNet class.

11.72.2.2 ∼CMultiBatteryChargerDeviceNet() `virtual ∼CMultiBatteryChargerDeviceNet ( ) [virtual]`

11.72.2.3 "!CMultiBatteryChargerDeviceNet() `!CMultiBatteryChargerDeviceNet ( )`

11.72.3 Member Function Documentation

11.72.3.1 CapacityTest()

```
void CapacityTest (
        uint32_t NrChannel )
```

start capacity test on channel

Parameters

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: ?Ah

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: ?Ah

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

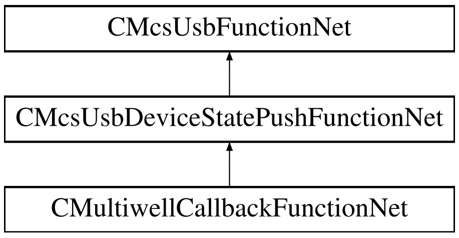
Parameters

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:


CMcsUsbFunctionNet
CMcsUsbDeviceStatePushFunctionNet
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

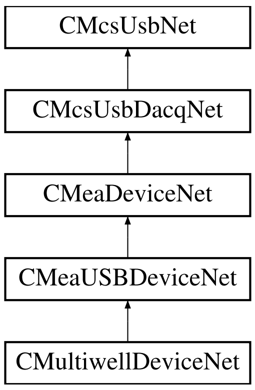
11.73.4.1 GetPlateClampStateByHeadstageEvent `OnGetPlateClampStateByHeadstage^ GetPlateClamp↩ StateByHeadstageEvent [add], [remove], [raise]`

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.

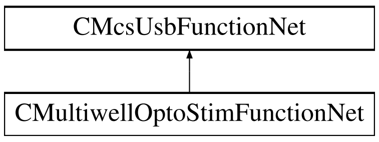
11.74.3.20 UnlockPlateClamp() `void UnlockPlateClamp ( )`

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

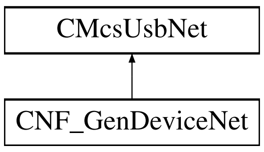
11.75.3.14 SetWaveLengthInNanometer() `void SetWaveLengthInNanometer ( uint16_t channel, uint32_t WaveLength_nm )`

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 )`

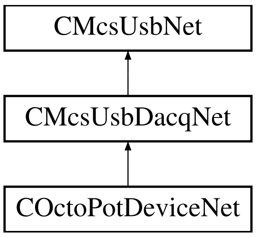
11.76.2 Member Function Documentation

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()

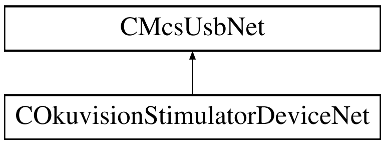
```
uint32_t SetSineParameter (
            unsigned int channel,
            int amplitude )
```

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 )
```

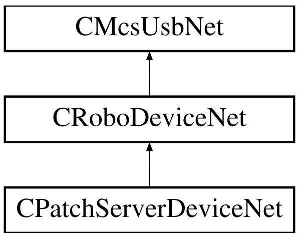
11.78.2.18 SetRTC() [2/2]

```
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

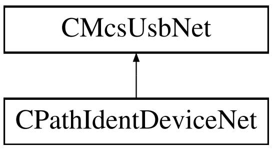
11.79.2.1 CPatchServerDeviceNet() `CPatchServerDeviceNet ( void )`

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()

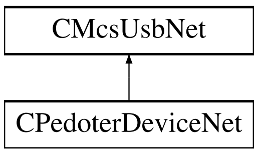
```
void Measure (
    [System::Runtime::InteropServices::Out] unsigned int% phase,
    [System::Runtime::InteropServices::Out] unsigned int% amplitude )
```

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

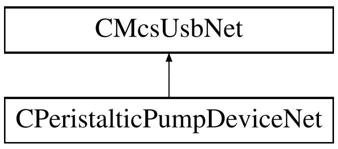
Parameters

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.

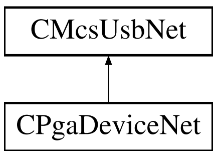
11.82.2.2 ∼CPeristalticPumpDeviceNet() ∼CPeristalticPumpDeviceNet (void)

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 )
```

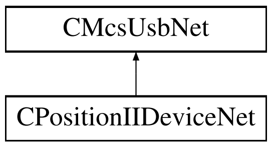
11.83.2.11 SetGain()

```
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::Interop↩ Services::Out]uint8_t% hour, [System::Runtime::InteropServices::Out]uint8_t% minute, [System::Runtime↩ ::InteropServices::Out]uint8_t% second)

  Get the RTC
- void SetRTC (uint8_t year, uint8_t month, uint8_t day, uint8_t hour, uint8_t minute, uint8_t second)

  Set the RTC
- uint32_t GetStateDebugData (uint16_t coil)

  get the debug queue state
- void SetStateDebugData (uint16_t coil, uint32_t state)

  clears/starts/stops the debug queue for a certain coil
- void GetDebugData (uint16_t coil, [System::↩ Runtime::InteropServices::Out]uint16_t% index, [System::Runtime::InteropServices::Out]uint16_t% voltage, [System::Runtime::InteropServices::Out]uint16_t% numberofpulses, [System::Runtime::InteropServices::Out]uint16_t% mediantime)

  get the oldest debug entry for a certain coil
- uint32_t GetStateEventData ()

  get the event queue state
- void SetStateEventData (uint32_t state)

  clears/starts/stops the event queue for a certain coil
- void GetEventData ([System::Runtime::InteropServices::Out]uint16_t% index, [System::Runtime::Interop↩ Services::Out]uint8_t% year, [System::Runtime::InteropServices::Out]uint8_t% month, [System::Runtime↩ ::InteropServices::Out]uint8_t% day, [System::Runtime::InteropServices::Out]uint8_t% hour, [System::↩ Runtime::InteropServices::Out]uint8_t% minute, [System::Runtime::InteropServices::Out]uint8_t% second, [System::Runtime::InteropServices::Out]uint16_t% coil, [System::Runtime::InteropServices::Out]uint16_t% type, [System::Runtime::InteropServices::Out]uint16_t% value)

  get the oldest event entry

Properties

- CRFFunctionNet^ RFFunction `[get]`

Additional Inherited Members

11.84.1 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

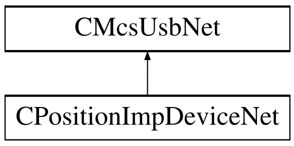
11.84.4 Property Documentation

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 )`

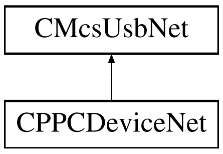
Sets the current RF frequency

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]`

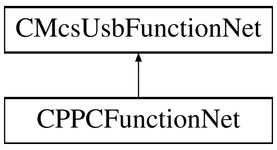
11.86.2.3 McsBus_Sensor `CMcsBus_SensorNet^ McsBus_Sensor [get]`

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 )
```

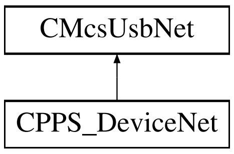
Sets the valve active/inactive state

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]

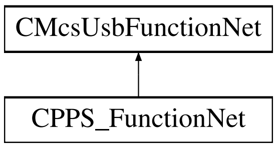
11.88.2.2 McsBus_MotorControl CMcsBus_MotorControlNet^ McsBus_MotorControl [get]

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% off-speed, [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 )
```

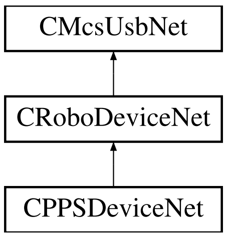
11.89.2.28 SetUseBubble()

```
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

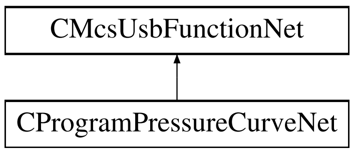
11.90.2.1 CPPSDeviceNet() CPPSDeviceNet (

```
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 )
```

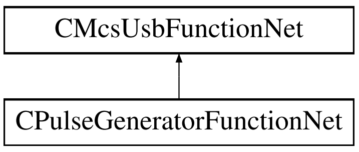
11.91.3.3 SetRepeats()

```
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 )
```

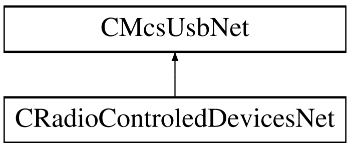
Writes the generator pulse length

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`

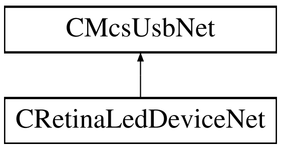
11.94.3.2 m_Left `int m_Left`

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 )`

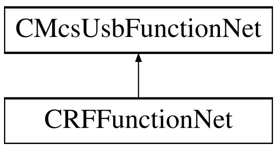
11.95.2.9 SetTablepointer() `unsigned int SetTablepointer ( int position )`

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 )
```

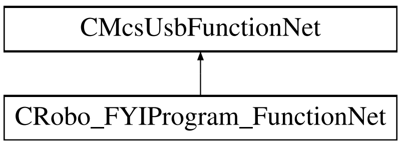
sets the working frequency

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_FY←IProgram_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 )`

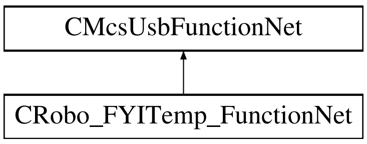
11.97.2.6 SetValve1() `void SetValve1 ( unsigned char index, unsigned int valve1 )`

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 (`

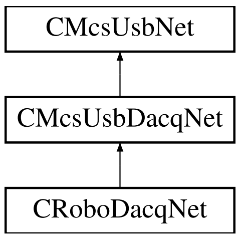
```
unsigned char index,
int onoff )
```

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 SetConfigurationBit (unsigned short bit, bool value)
- void SetConfigurationBitSupply (bool value)
- void SetConfigurationBitRelais (bool value)
- void SetConfigurationBitStream (bool value)
- void SetConfigurationBitAxc (bool value)
- void SetConfigurationBitCC_Gen (bool value)
- void SetConfigurationBitCV_Gen (bool value)
- void SetConfigurationBitRC_Gen (bool value)
- void SetConfigurationBitRV_Gen (bool value)
- void SetConfigurationBitBlu_Led (bool value)
- void SetConfigurationBitRed_Led (bool value)
- void SetConfigurationBitBlu_LedToggleSlow (bool value)
- void SetConfigurationBitRed_LedToggleSlow (bool value)
- void SetConfigurationBitBlu_LedToggleFast (bool value)
- void SetConfigurationBitRed_LedToggleFast (bool value)
- void SetConfigurationBitRed_LedSaturation (bool value)
- void SetSimulation (unsigned int enable)
- void SetUClamp (int uClamp)
- void SetIClamp (int iClamp)
- void SetPGain (int pGain)
- void SetIGain (int iGain)
- void SetFilter (int filter)
- void SetUVOffset (int UVOffset)
- void SetUCOffset (int UCOffset)
- void SetICOffset (int ICOffset)
- void SetCrossTalkOffset (int CrossTalk)
- void SetXGain (int xGain)
- void SetCrossTalkOptimum (int cxOptimum)
- void SetRecordingNumber (unsigned int recordingNumber)
- void ClampAmpRestart ()
- void DoRamp (int startValue, int endValue, int duration, int mode)
- unsigned int GetClampAmpSerialNumber ()
- unsigned int GetConfigurationBits ()
- bool GetConfigurationBit (unsigned short bit)
- bool GetConfigurationBitSupply ()
- bool GetConfigurationBitRelais ()
- bool GetConfigurationBitStream ()
- bool GetConfigurationBitAxc ()
- bool GetConfigurationBitCC_Gen ()
- bool GetConfigurationBitCV_Gen ()
- bool GetConfigurationBitRC_Gen ()
- bool GetConfigurationBitRV_Gen ()
- bool GetConfigurationBitBlu_Led ()
- bool GetConfigurationBitRed_Led ()
- bool GetConfigurationBitBlu_LedToggleSlow ()

- bool GetConfigurationBitRed_LedToggleSlow ()
- bool GetConfigurationBitBlu_LedToggleFast ()
- bool GetConfigurationBitRed_LedToggleFast ()
- bool GetConfigurationBitRed_LedSaturation ()
- unsigned int GetSimulation ()
- int GetUClamp ()
- int GetIClamp ()
- int GetPGain ()
- int GetIGain ()
- int GetFilter ()
- int GetUVOffset ()
- int GetUCOffset ()
- int GetICOffset ()
- int GetCrossTalkOffset ()
- int GetXGain ()
- int GetCrossTalkOptimum ()
- unsigned int GetRecordingNumber ()
- int GetResistanceC ()
- int GetResistanceV ()
- int GetCapacityC ()
- int GetCapacityV ()
- int GetCapacityX ()
- int GetUV ()
- int GetUC ()
- int GetIC ()
- int GetNUV_MS ()
- int GetNUC_MS ()
- int GetNIC_MS ()
- void SetAllDigout (uint32_t value)
- uint32_t GetAllDigout ()
- void SetCommand (unsigned char command, int value)
- int GetCommand (unsigned char command)
- void SetDigout (uint16_t index, bool enable)
- bool GetDigout (uint16_t index)
- void TableDefBegin ()
- void TableDefEnd ()
- void Table_Wait (unsigned int tableWait)
- void SetDownsampleFactor (int index, int downsample_factor)
- void SetFilterCoeffs (int index, array< int >^ coeffs)
- void SetNoFilterCoeffs (int index)
- int GetDownsampleFactor (int index)
- array< int > ^ GetFilterCoeffs (int index)
- void Emu_SetElectrodeResists (int emuElectrodeResist)
- void Emu_SetCellResists (int emuCellResist)
- void Emu_SetCellCapacity (int emuCellCapacity)
- void Emu_SetCellPotential (int emuCellPotential)
- void Emu_SetNoise (int emuNoise)
- int Emu_GetElectrodeResists ()
- int Emu_GetCellResists ()
- int Emu_GetCellCapacity ()
- int Emu_GetCellPotential ()
- int Emu_GetNoise ()
- void SetDisplayText (int index, String^ displayText)
- void SetScreen (int screen)
- void UpdateDisplay ()
- String ^ GetDisplayText (int index)
- int GetScreen ()
- int GetUpdateDisplay ()

Additional Inherited Members

11.99.1 Constructor & Destructor Documentation

11.99.1.1 CRoboDacqNet() [1/2]

```
CRoboDacqNet (
    void )
```

11.99.1.2 CRoboDacqNet() [2/2]

```
CRoboDacqNet (
    CRoboDeviceNet^ robodevice )
```

11.99.2 Member Function Documentation

11.99.2.1 CancelTableLoop()

```
void CancelTableLoop ( )
```

11.99.2.2 CancelTableLoopAndStopTable()

```
void CancelTableLoopAndStopTable ( )
```

11.99.2.3 ClampAmpRestart()

```
void ClampAmpRestart ( )
```

11.99.2.4 DoRamp()

```
void DoRamp (
    int startValue,
    int endValue,
    int duration,
    int mode )
```

11.99.2.5 Emu_GetCellCapacity()

```
int Emu_GetCellCapacity ( )
```

11.99.2.6 Emu_GetCellPotential()

```
int Emu_GetCellPotential ( )
```

11.99.2.7 Emu_GetCellResists() `int Emu_GetCellResists ( )`

11.99.2.8 Emu_GetElectrodeResists() `int Emu_GetElectrodeResists ( )`

11.99.2.9 Emu_GetNoise() `int Emu_GetNoise ( )`

11.99.2.10 Emu_SetCellCapacity() `void Emu_SetCellCapacity (`

`int` *`emuCellCapacity`* `)`

11.99.2.11 Emu_SetCellPotential() `void Emu_SetCellPotential (`

`int` *`emuCellPotential`* `)`

11.99.2.12 Emu_SetCellResists() `void Emu_SetCellResists (`

`int` *`emuCellResist`* `)`

11.99.2.13 Emu_SetElectrodeResists() `void Emu_SetElectrodeResists (`

`int` *`emuElectrodeResist`* `)`

11.99.2.14 Emu_SetNoise() `void Emu_SetNoise (`

`int` *`emuNoise`* `)`

11.99.2.15 GetAllDigout() `uint32_t GetAllDigout ( )`

11.99.2.16 GetCapacityC() `int GetCapacityC ( )`

11.99.2.17 GetCapacityV() `int GetCapacityV ( )`

11.99.2.18 GetCapacityX() `int GetCapacityX ( )`

11.99.2.19 GetClampAmpSerialNumber() `unsigned int GetClampAmpSerialNumber ( )`

11.99.2.20 GetCommand() `int GetCommand (`

```
unsigned char command )
```

11.99.2.21 GetConfigurationBit() `bool GetConfigurationBit (`

```
unsigned short bit )
```

11.99.2.22 GetConfigurationBitAxc() `bool GetConfigurationBitAxc ( )`

11.99.2.23 GetConfigurationBitBlu_Led() `bool GetConfigurationBitBlu_Led ( )`

11.99.2.24 GetConfigurationBitBlu_LedToggleFast() `bool GetConfigurationBitBlu_LedToggleFast ( )`

11.99.2.25 GetConfigurationBitBlu_LedToggleSlow() `bool GetConfigurationBitBlu_LedToggleSlow ( )`

11.99.2.26 GetConfigurationBitCC_Gen() `bool GetConfigurationBitCC_Gen ( )`

11.99.2.27 GetConfigurationBitCV_Gen() `bool GetConfigurationBitCV_Gen ( )`

11.99.2.28 GetConfigurationBitRC_Gen() `bool GetConfigurationBitRC_Gen ( )`

11.99.2.29 GetConfigurationBitRed_Led() `bool GetConfigurationBitRed_Led ( )`

11.99.2.30 GetConfigurationBitRed_LedSaturation() `bool GetConfigurationBitRed_LedSaturation ( )`

11.99.2.31 GetConfigurationBitRed_LedToggleFast() `bool GetConfigurationBitRed_LedToggleFast ( )`

11.99.2.32 GetConfigurationBitRed_LedToggleSlow() `bool GetConfigurationBitRed_LedToggleSlow ( )`

11.99.2.33 GetConfigurationBitRelais() `bool GetConfigurationBitRelais ( )`

11.99.2.34 GetConfigurationBitRV_Gen() `bool GetConfigurationBitRV_Gen ( )`

11.99.2.35 GetConfigurationBits() `unsigned int GetConfigurationBits ( )`

11.99.2.36 GetConfigurationBitStream() `bool GetConfigurationBitStream ( )`

11.99.2.37 GetConfigurationBitSupply() `bool GetConfigurationBitSupply ( )`

11.99.2.38 GetCrossTalkOffset() `int GetCrossTalkOffset ( )`

11.99.2.39 GetCrossTalkOptimum() `int GetCrossTalkOptimum ( )`

11.99.2.40 GetDigout() `bool GetDigout (`
`uint16_t` *`index`* `)`

11.99.2.41 GetDisplayText() `String ^ GetDisplayText (`
`int` *`index`* `)`

11.99.2.42 GetDownsampleFactor() `int GetDownsampleFactor (`
`int` *`index`* `)`

11.99.2.43 GetFilter() `int GetFilter ( )`

11.99.2.44 GetFilterCoeffs() `array<int> ^ GetFilterCoeffs (`
`int` *`index`* `)`

11.99.2.45 GetIC() `int GetIC ( )`

11.99.2.46 GetIClamp() `int GetIClamp ( )`

11.99.2.47 GetICOffset() `int GetICOffset ( )`

11.99.2.48 GetIGain() `int GetIGain ( )`

11.99.2.49 GetNIC_MS() `int GetNIC_MS ( )`

11.99.2.50 GetNUC_MS() `int GetNUC_MS ( )`

11.99.2.51 GetNUV_MS() `int GetNUV_MS ( )`

11.99.2.52 GetPGain() `int GetPGain ( )`

11.99.2.53 GetRecordingNumber() `unsigned int GetRecordingNumber ( )`

11.99.2.54 GetResistanceC() `int GetResistanceC ( )`

11.99.2.55 GetResistanceV() `int GetResistanceV ( )`

11.99.2.56 GetScreen() `int GetScreen ( )`

11.99.2.57 GetSimulation() `unsigned int GetSimulation ( )`

11.99.2.58 GetUC() `int GetUC ( )`

11.99.2.59 GetUClamp() `int GetUClamp ( )`

11.99.2.60 GetUCOffset() `int GetUCOffset ( )`

11.99.2.61 GetUpdateDisplay() `int GetUpdateDisplay ( )`

11.99.2.62 GetUV() `int GetUV ( )`

11.99.2.63 GetUVOffset() `int GetUVOffset ( )`

11.99.2.64 GetXGain() `int GetXGain ( )`

11.99.2.65 RunTable() [1/2] `void RunTable ( )`

11.99.2.66 RunTable() [2/2] `void RunTable (`

```
int timeout )
```

11.99.2.67 SetAllDigout() `void SetAllDigout (`

```
uint32_t value )
```

11.99.2.68 SetCommand() `void SetCommand (`

```
unsigned char command,
int value )
```

11.99.2.69 SetConfigurationBit() `void SetConfigurationBit (`

```
unsigned short bit,
bool value )
```

11.99.2.70 SetConfigurationBitAxc() `void SetConfigurationBitAxc (`

```
bool value )
```

11.99.2.71 SetConfigurationBitBlu_Led()

```
void SetConfigurationBitBlu_Led (
        bool value )
```

11.99.2.72 SetConfigurationBitBlu_LedToggleFast()

```
void SetConfigurationBitBlu_LedToggleFast (
        bool value )
```

11.99.2.73 SetConfigurationBitBlu_LedToggleSlow()

```
void SetConfigurationBitBlu_LedToggleSlow (
        bool value )
```

11.99.2.74 SetConfigurationBitCC_Gen()

```
void SetConfigurationBitCC_Gen (
        bool value )
```

11.99.2.75 SetConfigurationBitCV_Gen()

```
void SetConfigurationBitCV_Gen (
        bool value )
```

11.99.2.76 SetConfigurationBitRC_Gen()

```
void SetConfigurationBitRC_Gen (
        bool value )
```

11.99.2.77 SetConfigurationBitRed_Led()

```
void SetConfigurationBitRed_Led (
        bool value )
```

11.99.2.78 SetConfigurationBitRed_LedSaturation()

```
void SetConfigurationBitRed_LedSaturation (
        bool value )
```

11.99.2.79 SetConfigurationBitRed_LedToggleFast()

```
void SetConfigurationBitRed_LedToggleFast (
        bool value )
```

11.99.2.80 SetConfigurationBitRed_LedToggleSlow()

```
void SetConfigurationBitRed_LedToggleSlow (
        bool value )
```

11.99.2.81 SetConfigurationBitRelais()

```
void SetConfigurationBitRelais (
    bool value )
```

11.99.2.82 SetConfigurationBitRV_Gen()

```
void SetConfigurationBitRV_Gen (
    bool value )
```

11.99.2.83 SetConfigurationBitStream()

```
void SetConfigurationBitStream (
    bool value )
```

11.99.2.84 SetConfigurationBitSupply()

```
void SetConfigurationBitSupply (
    bool value )
```

11.99.2.85 SetCrossTalkOffset()

```
void SetCrossTalkOffset (
    int CrossTalk )
```

11.99.2.86 SetCrossTalkOptimum()

```
void SetCrossTalkOptimum (
    int cxOptimum )
```

11.99.2.87 SetDigout()

```
void SetDigout (
    uint16_t index,
    bool enable )
```

11.99.2.88 SetDisplayText()

```
void SetDisplayText (
    int index,
    String^ displayText )
```

11.99.2.89 SetDownsampleFactor()

```
void SetDownsampleFactor (
    int index,
    int downsample_factor )
```

11.99.2.90 SetFilter() `void SetFilter ( int filter )`

11.99.2.91 SetFilterCoeffs() `void SetFilterCoeffs ( int index, array< int >^ coeffs )`

11.99.2.92 SetIClamp() `void SetIClamp ( int iClamp )`

11.99.2.93 SetICOffset() `void SetICOffset ( int ICOffset )`

11.99.2.94 SetIGain() `void SetIGain ( int iGain )`

11.99.2.95 SetNoFilterCoeffs() `void SetNoFilterCoeffs ( int index )`

11.99.2.96 SetPGain() `void SetPGain ( int pGain )`

11.99.2.97 SetRecordingNumber() `void SetRecordingNumber ( unsigned int recordingNumber )`

11.99.2.98 SetScreen() `void SetScreen ( int screen )`

11.99.2.99 SetSimulation() `void SetSimulation ( unsigned int enable )`

11.99.2.100 SetUClamp() `void SetUClamp ( int uClamp )`

11.99.2.101 SetUCOffset() `void SetUCOffset ( int UCOffset )`

11.99.2.102 SetUVOffset() `void SetUVOffset ( int UVOffset )`

11.99.2.103 SetXGain() `void SetXGain ( int xGain )`

11.99.2.104 StopTable() [1/2] `void StopTable ( )`

11.99.2.105 StopTable() [2/2] `void StopTable ( int timeout )`

11.99.2.106 Table_Wait() `void Table_Wait ( unsigned int tableWait )`

11.99.2.107 TableDefBegin() `void TableDefBegin ( )`

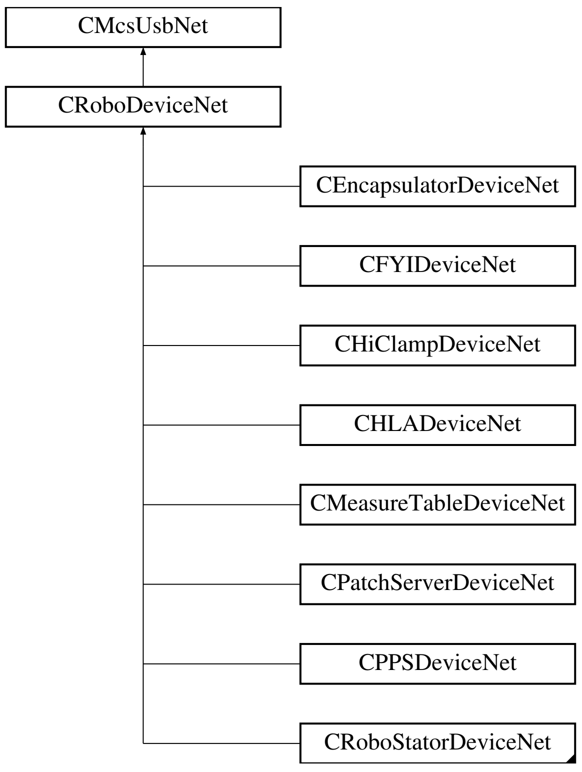
11.99.2.108 TableDefEnd() `void TableDefEnd ( )`

11.99.2.109 UpdateDisplay() `void UpdateDisplay ( )`

11.100 CRoboDeviceNet Class Reference

CRoboDeviceNet is the base class for all Robo platform based devices

Inheritance diagram for CRoboDeviceNet:



Classes

- class RoboMainLowLevelCommands

Public Member Functions

- CRoboDeviceNet (void)
- ∼CRoboDeviceNet (void)
- void SetInMovement ()

    Low level command, sets the internal state to "In Movement"
- bool GetInMovement ()

    Low level command, gets the internal state "In Movement"
- uint32_t GetMovementError ()

    Low level command, gets the error of the last movement end
- void FindReference (unsigned char busaddress, char axes)
- void FindReference (unsigned char busaddress, char axes, int timeout)

    Searches the reference position of the motor
- void MoveAbs (unsigned char busaddress, char axes, int x, int y)
- void MoveAbs (unsigned char busaddress, char axes, int x, int y, int timeout)

Moves the motor to the new absolute position

- void StopMovement (unsigned char busaddress, char axes)
- void StopMovement (unsigned char busaddress, char axes, int timeout)

Stops the current movement

- void SetCurrentAndAir (unsigned char busaddress, char axes, unsigned short onoff)
- void SetCurrentAndAir (unsigned char busaddress, char axes, unsigned short onoff, int timeout)
- void CancelPoolLoop ()
- void CancelPoolLoopAndStopMovement ()
- void GetCurrentPosition (unsigned char busaddress, char axes, [System::Runtime::InteropServices::Out]int% x, [System::Runtime::InteropServices::Out]int% y)

Gets the current position of motors

- void SetAirValve (unsigned int onoff)
- unsigned int GetAirValve ()
- unsigned int GetVoltageValves ()
- unsigned int GetVoltageRs485A ()
- unsigned int GetVoltageRs485B ()
- unsigned int GetVoltageAirvalve ()
- unsigned int GetCurrentAirvalve ()
- unsigned int GetVoltage12V ()
- unsigned int GetAirpressure ()
- unsigned int GetVoltage5V ()
- unsigned int GetErrorVoltageValves ()
- unsigned int GetErrorVoltageRs485A ()
- unsigned int GetErrorVoltageRs485B ()
- unsigned int GetErrorVoltageAirvalve ()
- unsigned int GetErrorCurrentAirvalve ()
- unsigned int GetErrorVoltage12V ()
- unsigned int GetErrorAirpressure ()
- unsigned int GetErrorVoltage5V ()
- void SetVoltageValvesLimit (unsigned int lowervoltage, unsigned int uppervoltage)
- void SetVoltageRs485ALimit (unsigned int lowervoltage, unsigned int uppervoltage)
- void SetVoltageRs485BLimit (unsigned int lowervoltage, unsigned int uppervoltage)
- void SetVoltageAirvalveLimit (unsigned int lowervoltage, unsigned int uppervoltage)
- void SetCurrentAirvalveLimit (unsigned int lowercurrent, unsigned int uppercurrent)
- void SetVoltage12VLimit (unsigned int lowervoltage, unsigned int uppervoltage)
- void SetAirpressureLimit (unsigned int lowerpressure, unsigned int upperpressure)
- void SetVoltage5VLimit (unsigned int lowervoltage, unsigned int uppervoltage)
- void GetVoltageValvesLimit ([System::Runtime::InteropServices::Out] unsigned int% lowervoltage, [System↩ ::Runtime::InteropServices::Out]unsigned int% uppervoltage)
- void GetVoltageRs485ALimit ([System::Runtime::InteropServices::Out]unsigned int% lowervoltage, [System::Runtime::InteropServices::Out]unsigned int% uppervoltage)
- void GetVoltageRs485BLimit ([System::Runtime::InteropServices::Out]unsigned int% lowervoltage, [System::Runtime::InteropServices::Out]unsigned int% uppervoltage)
- void GetVoltageAirvalveLimit ([System::Runtime::InteropServices::Out]unsigned int% lowervoltage, [System::Runtime::InteropServices::Out]unsigned int% uppervoltage)
- void GetCurrentAirvalveLimit ([System::Runtime::InteropServices::Out]unsigned int% lowercurrent, [System::Runtime::InteropServices::Out]unsigned int% uppercurrent)
- void GetVoltage12VLimit ([System::Runtime::InteropServices::Out]unsigned int% lowervoltage, [System::↩ Runtime::InteropServices::Out]unsigned int% uppervoltage)
- void GetAirpressureLimit ([System::Runtime::InteropServices::Out]unsigned int% lowerpressure, [System↩ ::Runtime::InteropServices::Out]unsigned int% upperpressure)
- void GetVoltage5VLimit ([System::Runtime::InteropServices::Out]unsigned int% lowervoltage, [System::↩ Runtime::InteropServices::Out]unsigned int% uppervoltage)
- void SetMinPressure (int pressure)
- int GetMinPressure ()

Static Public Attributes

- static const uint32_t RoboError_Base = (0xA0110000L)
- static const uint32_t RoboError_UnknownCommand = ((0xA0110000L))
- static const uint32_t RoboError_Timeout = ((0xA0110000L) | 0x0001)
- static const uint32_t RoboError_Pressure = ((0xA0110000L) | 0x0002)
- static const uint32_t RoboError_RangeExceeded = ((0xA0110000L) | 0x0003)
- static const uint32_t RoboError_CommunicationTimeout = ((0xA0110000L) | 0x0004)
- static const uint32_t RoboError_AnotherMaster = ((0xA0110000L) | 0x0005)
- static const uint32_t RoboError_FindReferenceMethod = ((0xA0110000L) | 0x0006)
- static const uint32_t RoboError_NoSpeedOrAcceleration = ((0xA0110000L) | 0x0007)
- static const uint32_t RoboError_NoEndSwitch = ((0xA0110000L) | 0x0008)
- static const uint32_t RoboError_CannotEscapeEndSwitch = ((0xA0110000L) | 0x0009)
- static const uint32_t RoboError_CommandAlreadyInProgress = ((0xA0110000L) | 0x000A)
- static const uint32_t RoboError_NoReference = ((0xA0110000L) | 0x000B)
- static const uint32_t RoboError_OverPressure = ((0xA0110000L) | 0x000C)
- static const uint32_t RoboError_Phase0OutOfRange = ((0xA0110000L) | 0x000D)
- static const uint32_t RoboError_PeristalticTimeout = ((0xA0110000L) | 0x000E)
- static const uint32_t RoboError_GilsonTimeout = ((0xA0110000L) | 0x000F)
- static const uint32_t RoboError_GilsonWrondID = ((0xA0110000L) | 0x0010)
- static const uint32_t RoboError_GilsonCommandPending = ((0xA0110000L) | 0x0011)
- static const uint32_t RoboError_ParameterNotAllowed = ((0xA0110000L) | 0x0012)
- static const uint32_t RoboError_StateChangeNotPossible = ((0xA0110000L) | 0x0013)
- static const uint32_t RoboError_CommandNotPossible = ((0xA0110000L) | 0x0014)
- static const uint32_t RoboError_DacqNotReady = ((0xA0110000L) | 0x0015)
- static const uint32_t RoboError_NoMoreData = ((0xA0110000L) | 0x0016)
- static const uint32_t RoboError_McsBus_UnknownCommand = ((0xA0110000L) | 0x003F)
- static const uint32_t RoboError_DLLMovementTimeout = ((0xA0110000L) | 0x1001)
- static const uint32_t RoboError_PollLoopCanceled = ((0xA0110000L) | 0x1002)
- static const uint32_t RoboError_PollLoopCanceledAndStopMovement = ((0xA0110000L) | 0x1003)
- static const byte McsBus_XY = 1

  McsBus address for the xy-plane
- static const byte McsBus_ZI = 2

  McsBus address for the z and i axes
- static const byte Axis_X = 0

  Axis number of x for axis argument
- static const byte Axis_Y = 1

  Axis number of y for axis argument
- static const byte Axis_Z = 0

  Axis number of z for axis argument
- static const byte Axis_I = 1

  Axis number of i for axis argument
- static const char Axes_X = 1

  Bit pattern for x axis for axes argument
- static const char Axes_Y = 2

  Bit pattern for y axis for axes argument
- static const char Axes_Z = 1

  Bit pattern for z axis for axes argument
- static const char Axes_I = 2

  Bit pattern for i axis for axes argument

Properties

- CMcsBusNet^ 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 FindReference() [1/2]

```
void FindReference (
    unsigned char busaddress,
    char axes )
```

11.100.3.4 FindReference() [2/2]

```
void FindReference (
    unsigned char busaddress,
    char axes,
    int timeout )
```

Searches the reference position of the motor

Parameters

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)

11.100.3.5 GetAirpressure() `unsigned int GetAirpressure ( )`

11.100.3.6 GetAirpressureLimit()

```
void GetAirpressureLimit (
    [System::Runtime::InteropServices::Out] unsigned int% lowerpressure,
    [System::Runtime::InteropServices::Out] unsigned int% upperpressure )
```

11.100.3.7 GetAirValve() `unsigned int GetAirValve ( )`

11.100.3.8 GetCurrentAirvalve() `unsigned int GetCurrentAirvalve ( )`

11.100.3.9 GetCurrentAirvalveLimit()

```
void GetCurrentAirvalveLimit (
    [System::Runtime::InteropServices::Out] unsigned int% lowercurrent,
    [System::Runtime::InteropServices::Out] unsigned int% uppercurrent )
```

11.100.3.10 GetCurrentPosition()

```
void GetCurrentPosition (
    unsigned char busaddress,
    char axes,
    [System::Runtime::InteropServices::Out] int% x,
    [System::Runtime::InteropServices::Out] int% y )
```

Gets the current position of motors

Parameters

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

11.100.3.11 GetErrorAirpressure() `unsigned int GetErrorAirpressure ( )`

11.100.3.12 GetErrorCurrentAirvalve() `unsigned int GetErrorCurrentAirvalve ( )`

11.100.3.13 GetErrorVoltage12V() `unsigned int GetErrorVoltage12V ( )`

11.100.3.14 GetErrorVoltage5V() `unsigned int GetErrorVoltage5V ( )`

11.100.3.15 GetErrorVoltageAirvalve() `unsigned int GetErrorVoltageAirvalve ( )`

11.100.3.16 GetErrorVoltageRs485A() `unsigned int GetErrorVoltageRs485A ( )`

11.100.3.17 GetErrorVoltageRs485B() `unsigned int GetErrorVoltageRs485B ( )`

11.100.3.18 GetErrorVoltageValves() `unsigned int GetErrorVoltageValves ( )`

11.100.3.19 GetInMovement() `bool GetInMovement ( )`

Low level command, gets the internal state "In Movement"

11.100.3.20 GetMinPressure() `int GetMinPressure ( )`

11.100.3.21 GetMovementError() `uint32_t GetMovementError ( )`

Low level command, gets the error of the last movement end

11.100.3.22 GetVoltage12V() `unsigned int GetVoltage12V ( )`

11.100.3.23 GetVoltage12VLimit()

```
void GetVoltage12VLimit (
    [System::Runtime::InteropServices::Out] unsigned int% lowervoltage,
    [System::Runtime::InteropServices::Out] unsigned int% uppervoltage )
```

11.100.3.24 GetVoltage5V() `unsigned int GetVoltage5V ( )`

11.100.3.25 GetVoltage5VLimit()

```
void GetVoltage5VLimit (
    [System::Runtime::InteropServices::Out] unsigned int% lowervoltage,
    [System::Runtime::InteropServices::Out] unsigned int% uppervoltage )
```

11.100.3.26 GetVoltageAirvalve() `unsigned int GetVoltageAirvalve ( )`

11.100.3.27 GetVoltageAirvalveLimit()

```
void GetVoltageAirvalveLimit (
    [System::Runtime::InteropServices::Out] unsigned int% lowervoltage,
    [System::Runtime::InteropServices::Out] unsigned int% uppervoltage )
```

11.100.3.28 GetVoltageRs485A() `unsigned int GetVoltageRs485A ( )`

11.100.3.29 GetVoltageRs485ALimit()

```
void GetVoltageRs485ALimit (
    [System::Runtime::InteropServices::Out] unsigned int% lowervoltage,
    [System::Runtime::InteropServices::Out] unsigned int% uppervoltage )
```

11.100.3.30 GetVoltageRs485B()

```
unsigned int GetVoltageRs485B ( )
```

11.100.3.31 GetVoltageRs485BLimit()

```
void GetVoltageRs485BLimit (
    [System::Runtime::InteropServices::Out] unsigned int% lowervoltage,
    [System::Runtime::InteropServices::Out] unsigned int% uppervoltage )
```

11.100.3.32 GetVoltageValves()

```
unsigned int GetVoltageValves ( )
```

11.100.3.33 GetVoltageValvesLimit()

```
void GetVoltageValvesLimit (
    [System::Runtime::InteropServices::Out] unsigned int% lowervoltage,
    [System::Runtime::InteropServices::Out] unsigned int% uppervoltage )
```

11.100.3.34 MoveAbs() [1/2]

```
void MoveAbs (
    unsigned char busaddress,
    char axes,
    int x,
    int y )
```

11.100.3.35 MoveAbs() [2/2]

```
void MoveAbs (
    unsigned char busaddress,
    char axes,
    int x,
    int y,
    int timeout )
```

Moves the motor to the new absolute position

Parameters

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)

11.100.3.36 SetAirpressureLimit()

```
void SetAirpressureLimit (
unsigned int lowerpressure,
unsigned int upperpressure )
```

11.100.3.37 SetAirValve()

```
void SetAirValve (
    unsigned int onoff )
```

11.100.3.38 SetCurrentAirvalveLimit()

```
void SetCurrentAirvalveLimit (
    unsigned int lowercurrent,
    unsigned int uppercurrent )
```

11.100.3.39 SetCurrentAndAir() [1/2]

```
void SetCurrentAndAir (
    unsigned char busaddress,
    char axes,
    unsigned short onoff )
```

11.100.3.40 SetCurrentAndAir() [2/2]

```
void SetCurrentAndAir (
    unsigned char busaddress,
    char axes,
    unsigned short onoff,
    int timeout )
```

11.100.3.41 SetInMovement()

```
void SetInMovement ( )
```

Low level command, sets the internal state to "In Movement"

11.100.3.42 SetMinPressure()

```
void SetMinPressure (
    int pressure )
```

11.100.3.43 SetVoltage12VLimit()

```
void SetVoltage12VLimit (
    unsigned int lowervoltage,
    unsigned int uppervoltage )
```

11.100.3.44 SetVoltage5VLimit()

```
void SetVoltage5VLimit (
    unsigned int lowervoltage,
    unsigned int uppervoltage )
```

11.100.3.45 SetVoltageAirvalveLimit()

```
void SetVoltageAirvalveLimit (
    unsigned int lowervoltage,
    unsigned int uppervoltage )
```

11.100.3.46 SetVoltageRs485ALimit()

```
void SetVoltageRs485ALimit (
    unsigned int lowervoltage,
    unsigned int uppervoltage )
```

11.100.3.47 SetVoltageRs485BLimit()

```
void SetVoltageRs485BLimit (
    unsigned int lowervoltage,
    unsigned int uppervoltage )
```

11.100.3.48 SetVoltageValvesLimit()

```
void SetVoltageValvesLimit (
    unsigned int lowervoltage,
    unsigned int uppervoltage )
```

11.100.3.49 StopMovement() [1/2]

```
void StopMovement (
    unsigned char busaddress,
    char axes )
```

11.100.3.50 StopMovement() [2/2]

```
void StopMovement (
    unsigned char busaddress,
    char axes,
    int timeout )
```

Stops the current movement

Parameters

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)

11.100.4 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 = ( (0x↩ A0110000L) | 0x000D ) [static]`

11.100.4.32 RoboError_PollLoopCanceled `const uint32_t RoboError_PollLoopCanceled = ( (0x↩ A0110000L) | 0x1002) [static]`

11.100.4.33 RoboError_PollLoopCanceledAndStopMovement `const uint32_t RoboError_PollLoop↩ CanceledAndStopMovement = ( (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 = ( (0x← A0110000L) ) [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]`

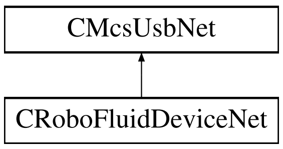
11.100.5.3 RoboMainLowLevelCommand `RoboMainLowLevelCommands^ RoboMainLowLevelCommand [get]`

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

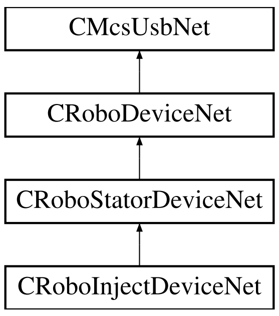
11.101.4.1 McsBus_MotorControl

`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

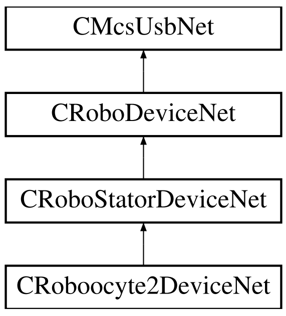
11.102.2 Constructor & Destructor Documentation

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 ( )
```

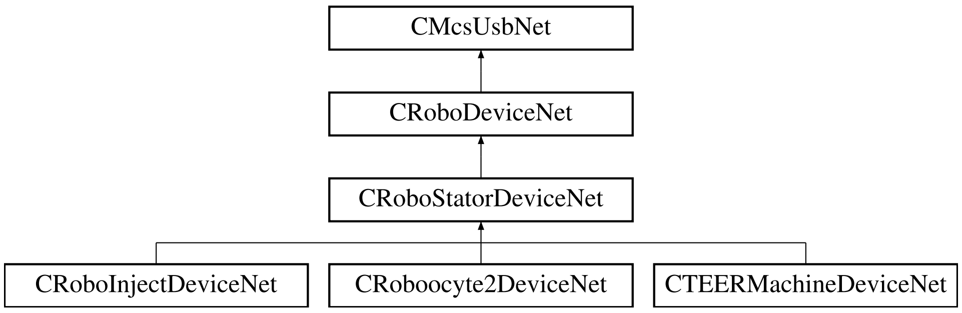
11.103.3.5 GetRoboFluidDevice() `CRoboFluidDeviceNet ^ GetRoboFluidDevice ( )`

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 ( )`

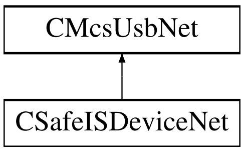
11.104.2.41 StopMovementZ() [2/2] `void StopMovementZ ( int timeout )`

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]`

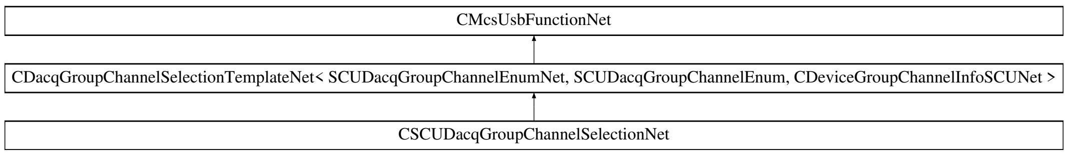
Gets the CFluidControlDeviceNet. Use this to control the valves. Only SetSingleValve is implemented for CSafeISDeviceNet.

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:


CMcsUsbFunctionNet
CDacqGroupChannelSelectionTemplateNet< SCUDacqGroupChannelEnumNet, SCUDacqGroupChannelEnum, CDeviceGroupChannelInfoSCUNet >
CSCUDacqGroupChannelSelectionNet

Public Member Functions

- CSCUDacqGroupChannelSelectionNet (CMcsUsbNet^ mcsusb)

Additional Inherited Members

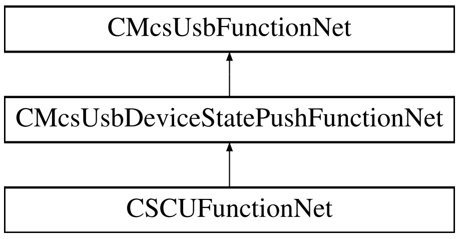
11.106.1 Constructor & Destructor Documentation

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 GetHeadstageDacCurrentRange↩ InMicroAmpere ( 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 GetHeadstageDacCurrent↩
ResolutionInNanoAmpere (
            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 GetHeadstageDacVoltageRangeIn↩
MilliVolt (
            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 GetHeadstageDacVoltage↩
ResolutionInMicroVolt (
            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 GetHeadstageNumberOfStimulation↩
Channels (
            uint32_t Headstage )
```

Gets the number of stimulation channels for a given headstage.

Parameters

Headstage	The headstage to query.

Returns

The number of stimulation channels the headstage has.

11.107.3.20 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 GetReferenceElectrode↵ Mode ( uint32_t Headstage )`

Gets the mode for the reference electrode

Parameters

Headstage	The headstage number

Returns

The mode

11.107.3.26 GetReferenceElectrodeSwitchState() `ReferenceElectrodeSwitchPositionEnumNet Get↩ ReferenceElectrodeSwitchState ( uint32_t Headstage )`

Gets the position of the switch for the reference electrode

Parameters

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]`

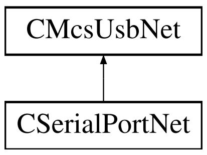
Event fires when the bitmap of available headstages has changed

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 )
```

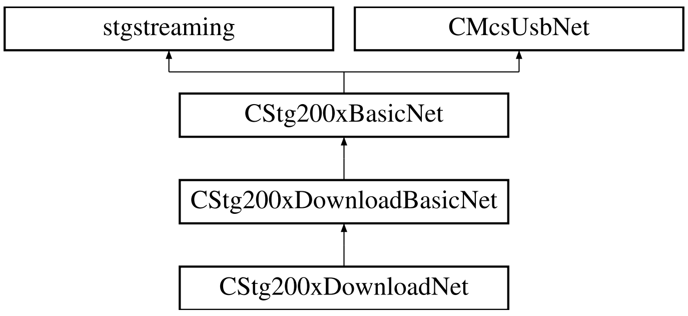
11.108.2.7 Send() [2/2]

```
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 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< ElectrodeDacMux↩ EnumNet >^ 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, Electrode↩ DacMuxEnumNet 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 sideband, unsigned int startIndex, unsigned int endIndex, unsigned int mode)
- virtual void GetListmodeIndexRange (unsigned int sideband, unsigned int &startIndex, unsigned int &end↩ Index, unsigned int &mode)
- virtual void SetListmodeTriggerSource (unsigned int sideband, TriggerSourceEnumNet triggersource, int bitnumOffset)
- virtual void SetListmodeTriggerSource (unsigned int sideband, TriggerSourceEnumNet triggersource)
- virtual TriggerSourceEnumNet GetListmodeTriggerSource (unsigned int sideband)
- virtual void ListModeSendStart (unsigned int sidebandMask)
- virtual void ListModeSendStop (unsigned int sidebandMask)
- virtual void SetHeadstage (unsigned int headstage)
- virtual uint32_t GetHeadstage ()
- virtual void SetDacAmplificationFactor (uint32_t DacNumber, double Factor)

  Set the amplification factor for a DAC.
- virtual double GetDacAmplificationFactor (uint32_t DacNumber)

  Get the amplification factor for a DAC.
- virtual void SetDigoutMode (Stg200xDigoutModeEnumNet digoutMode)

Sets the operation mode of the digital 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 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.

11.109.3.8 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.

11.109.3.9 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.

11.109.3.10 GetDACResolution() `virtual int32_t GetDACResolution ( ) [virtual]`

Gets number of bits of the DAC resolution.

Returns

The DAC resolution in bits.

11.109.3.11 GetDiginValue() `virtual uint32_t GetDiginValue ( ) [virtual]`

Gets the Value on the digital input port.

Returns

The current value on the digital inputs.

11.109.3.12 GetDigoutMode() `virtual Stg200xDigoutModeEnumNet GetDigoutMode ( ) [virtual]`

Gets the operation mode of the digital outport port, can be Monitor, Manual or SyncOut

Returns

The current operation mode.

11.109.3.13 GetDigoutValue() `virtual uint32_t GetDigoutValue ( ) [virtual]`

Gets the Value on the digital output port.

Returns

The current value on the digital outputs.

11.109.3.14 GetElectrodeDacMux() [1/2]

```
virtual ElectrodeDacMuxEnumNet GetElectrodeDacMux (
    uint32_t electrode,
    uint32_t listmodeIndex ) [virtual]
```

Gets the DAC which is used for an electrode.

Parameters

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.

11.109.3.15 GetElectrodeDacMux() [2/2]

```
virtual ElectrodeDacMuxEnumNet GetElectrodeDacMux (
    uint32_t Scu_HS,
    uint32_t electrode,
    uint32_t listmodeIndex ) [virtual]
```

Gets the DAC which is used for an electrode.

Parameters

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.

11.109.3.16 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.

11.109.3.17 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.

11.109.3.18 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.

11.109.3.19 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.

11.109.3.20 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.

11.109.3.21 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.

11.109.3.22 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.

11.109.3.23 GetFAAmplification()

```
virtual uint32_t GetFAAmplification ( ) [virtual]
```

11.109.3.24 GetHeadstage()

```
virtual uint32_t GetHeadstage ( ) [virtual]
```

11.109.3.25 GetListmodeIndexRange()

```
virtual void GetListmodeIndexRange (
        unsigned int sideband,
        unsigned int & startIndex,
        unsigned int & endIndex,
        unsigned int & mode ) [virtual]
```

11.109.3.26 GetListmodeTriggerSource()

```
virtual TriggerSourceEnumNet GetListmodeTriggerSource (
        unsigned int sideband ) [virtual]
```

11.109.3.27 GetNumberOfAnalogChannels()

```
virtual uint32_t GetNumberOfAnalogChannels ( ) [virtual]
```

Gets the Number of available analog channels of the device.

Returns

The number of analog channels.

11.109.3.28 GetNumberOfHWDACPaths()

```
virtual uint32_t GetNumberOfHWDACPaths ( ) [virtual]
```

Gets the Number of HW Stimulation DACs of the device.

Returns

The number of independent HW Stimulation outputs.

11.109.3.29 GetNumberOfStimulationElectrodes() `virtual uint32_t GetNumberOfStimulationElectrodes ( ) [virtual]`

11.109.3.30 GetNumberOfStimulationSourcesPerElectrode() `virtual uint32_t GetNumberOfStimulation↩ SourcesPerElectrode ( ) [virtual]`

Gets the number of stimulation sources (DACs) per electrode.

Returns

The number of stimulation sources (DACs) per electrode.

11.109.3.31 GetNumberOfSyncoutChannels() `virtual uint32_t GetNumberOfSyncoutChannels ( ) [virtual]`

Gets the Number of available syncout channels of the device.

Returns

The number of analog channels.

11.109.3.32 GetNumberOfTriggerInputs() `virtual uint32_t GetNumberOfTriggerInputs ( ) [virtual]`

Gets the Number of trigger inputs of the device.

Returns

The number of trigger inputs.

11.109.3.33 GetOutputRate() `uint32_t GetOutputRate ( )`

Queries the output rate of the STG. Valid rates are from 1000 Hz to 50000 Hz.

Returns

Returns the current output rate in Hz.

11.109.3.34 GetStgProgramInfo() [1/2]

```
void GetStgProgramInfo (
        [Out] bool% IsProgrammed,
        [Out] DateTime% timestamp,
        [Out] String^% filename,
        [Out] Guid% guid )
```

Queries Download information from the STG. If download information was stored by the use of SetStgProgramInfo, this function can be used to retrieve it.

Parameters

IsProgrammed	Flag wether download information is valid.
timestamp	The timestamp of last download.
filename	The filename of the downlaoded waveform.
guid	A GUID.

11.109.3.35 GetStgProgramInfo() [2/2]

```
void GetStgProgramInfo (
    [Out] bool% IsProgrammed,
    [Out] System::Runtime::InteropServices::ComTypes::FILETIME% timestamp,
    [Out] String^% filename,
    [Out] Guid% guid )
```

Queries Download information from the STG. If download information was stored by the use of SetStgProgramInfo, this function can be used to retrieve it.

Parameters

IsProgrammed	Flag wether download information is valid.
timestamp	The timestamp of last download.
filename	The filename of the downlaoded waveform.

11.109.3.36 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.

11.109.3.37 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.

11.109.3.38 GetTotalMemory() `uint32_t GetTotalMemory ( )`

Gets the total amount of memory available on the STG (all segments).

Returns

The total memory available on the STG in bytes.

11.109.3.39 GetTriggerSource() `virtual TriggerSourceEnumNet GetTriggerSource ( unsigned int triggernum ) [virtual]`

11.109.3.40 GetVoltageRangeInMicroVolt() `virtual int32_t GetVoltageRangeInMicroVolt ( uint32_t channel ) [virtual]`

Gets the Voltage Range of the specified channel in Microvolts.

Parameters

channel	Channel which is queried.

Returns

The Voltage Range of the specified channel in Microvolts.

11.109.3.41 GetVoltageResolutionInMicroVolt() `virtual int32_t GetVoltageResolutionInMicroVolt ( uint32_t channel ) [virtual]`

Gets the Voltage Resolution of the specified channel in Microvolts.

Parameters

channel	Channel which is queried.

Returns

The Voltage Resolution of the specified channel in Microvolts.

11.109.3.42 ListModeSendStart() `virtual void ListModeSendStart ( unsigned int sidebandMask ) [virtual]`

11.109.3.43 ListModeSendStop() `virtual void ListModeSendStop ( unsigned int sidebandMask ) [virtual]`

11.109.3.44 SendStart() `void SendStart ( uint32_t triggermap )`

Start (Trigger) the STG. The startup delay is in the range of a few ms.

Parameters

triggermap	A bitmap of triggers which will be started.

11.109.3.45 SendStop() [1/2] `void SendStop ( uint32_t triggermap )`

Stop some or all triggers of the STG.

Parameters

triggermap	A bitmap of triggers which will be stopped.

11.109.3.46 SendStop() [2/2] `void SendStop ( uint32_t triggermap, int options )`

Stop some or all triggers of the STG.

Parameters

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..

11.109.3.47 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.

11.109.3.48 SetBlankingEnable() [1/4]

```
virtual void SetBlankingEnable (
    uint32_t electrode,
    array< bool >^ enable ) [virtual]
```

Defines whether an electrode should be blanked while stimulation is in progress.

Parameters

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.

11.109.3.49 SetBlankingEnable() [2/4]

```
virtual void SetBlankingEnable (
    uint32_t electrode,
    bool enable ) [virtual]
```

Defines whether an electrode should be blanked while stimulation is in progress.

Parameters

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() [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.

11.109.3.51 SetBlankingEnable() [4/4]

```
virtual void SetBlankingEnable (
    uint32_t Scu_HS,
    uint32_t electrode,
    bool enable ) [virtual]
```

Defines whether an electrode should be blanked while stimulation is in progress.

Parameters

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 SetCurrentMode() [1/2]

```
virtual void SetCurrentMode ( ) [virtual]
```

Sets all channels to current mode (STG3008-FA and STG400x only).

11.109.3.53 SetCurrentMode() [2/2]

```
virtual void SetCurrentMode (
    unsigned int channel ) [virtual]
```

Sets a channel to current mode (STG3008-FA and STG400x only).

Parameters

channel	The channel to change.

11.109.3.54 SetDacAmplificationFactor()

```
virtual void SetDacAmplificationFactor (
        uint32_t DacNumber,
        double Factor ) [virtual]
```

Set the amplification factor for a DAC.

Parameters

DacNumber	The number of the DAC.
Factor	the amplifcation factor for that DAC, range is from -1.99999 to +1.99999.

11.109.3.55 SetDigoutMode()

```
virtual void SetDigoutMode (
        Stg200xDigoutModeEnumNet digoutMode ) [virtual]
```

Sets the operation mode of the digital outport port, can be Monitor, Manual or SyncOut

Parameters

digoutMode	The new operation mode.

11.109.3.56 SetDigoutValue()

```
virtual void SetDigoutValue (
        uint32_t digoutValue ) [virtual]
```

Sets the Value on the digital output port when in manual mode.

Parameters

digoutValue	The new value on the digital outputs.

11.109.3.57 SetElectrodeDacMux() [1/4]

```
virtual void SetElectrodeDacMux (
        uint32_t electrode,
        uint32_t listmodeIndex,
        array< ElectrodeDacMuxEnumNet >^ dacMux ) [virtual]
```

Defines the DAC to use for an electrode.

Parameters

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).

11.109.3.58 SetElectrodeDacMux() [2/4]

```
virtual void SetElectrodeDacMux (
        uint32_t electrode,
        uint32_t listmodeIndex,
        ElectrodeDacMuxEnumNet dacMux )  [virtual]
```

Defines the DAC to use for an electrode.

Parameters

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).

11.109.3.59 SetElectrodeDacMux() [3/4]

```
virtual void SetElectrodeDacMux (
        uint32_t Scu_HS,
        uint32_t electrode,
        uint32_t listmodeIndex,
        array< ElectrodeDacMuxEnumNet >^ dacMux )  [virtual]
```

Defines the DAC to use for an electrode.

Parameters

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).

11.109.3.60 SetElectrodeDacMux() [4/4]

```
virtual void SetElectrodeDacMux (
    uint32_t Scu_HS,
    uint32_t electrode,
    uint32_t listmodeIndex,
    ElectrodeDacMuxEnumNet dacMux )  [virtual]
```

Defines the DAC to use for an electrode.

Parameters

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).

11.109.3.61 SetElectrodeEnable() [1/4]

```
virtual void SetElectrodeEnable (
    uint32_t electrode,
uint32_t listmodeIndex,
array< bool >^ enable )  [virtual]
```

Enables or disables the stimulation switch for an electrode.

Parameters

electrode	The electrode number.

Parameters

listmodeIndex	The index for listmode.
enable	1 to enable the electrode, 0 to disable.

11.109.3.62 SetElectrodeEnable() [2/4]

```
virtual void SetElectrodeEnable (
uint32_t electrode,
uint32_t listmodeIndex,
bool enable )  [virtual]
```

Enables or disables the stimulation switch for an electrode.

Parameters

electrode	The electrode number.

Parameters

listmodeIndex	The index for listmode.
enable	1 to enable the electrode, 0 to disable.

11.109.3.63 SetElectrodeEnable() [3/4]

```
virtual void SetElectrodeEnable (
uint32_t Scu_HS,
uint32_t electrode,
uint32_t listmodeIndex,
array< bool >^ enable )  [virtual]
```

Enables or disables the stimulation switch for an electrode.

Parameters

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.

11.109.3.64 SetElectrodeEnable() [4/4]

```
virtual void SetElectrodeEnable (
        uint32_t Scu_HS,
        uint32_t electrode,
        uint32_t listmodeIndex,
        bool enable )  [virtual]
```

Enables or disables the stimulation switch for an electrode.

Parameters

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.

11.109.3.65 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.

11.109.3.66 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.

11.109.3.67 SetElectrodeMode() [3/4]

```
virtual void SetElectrodeMode (
    uint32_t Scu_HS,
    uint32_t electrode,
    array< ElectrodeModeEnumNet >^ mode ) [virtual]
```

Puts an electrode in either automatic or manual mode.

Parameters

Scu_HS	The SCU headstage number.

Parameters

electrode	The electrode number.

Returns

0 for automatic and 3 for manual mode.

11.109.3.68 SetElectrodeMode() [4/4]

```
virtual void SetElectrodeMode (
    uint32_t Scu_HS,
    uint32_t electrode,
    ElectrodeModeEnumNet mode ) [virtual]
```

Puts an electrode in either automatic or manual mode.

Parameters

Scu_HS	The SCU headstage number.

Parameters

electrode	The electrode number.

Parameters

mode	0 for automatic and 3 for manual mode.

11.109.3.69 SetEnableAmplifierProtectionSwitch() [1/4]

```
virtual void SetEnableAmplifierProtection←
Switch (
            uint32_t electrode,
            array< bool >^ enable )  [virtual]
```

Defines whether the Amplifier Protection Switch is openend while stimulation is in progress.

Parameters

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.

11.109.3.70 SetEnableAmplifierProtectionSwitch() [2/4]

```
virtual void SetEnableAmplifierProtection←
Switch (
            uint32_t electrode,
            bool enable )  [virtual]
```

Defines whether the Amplifier Protection Switch is openend while stimulation is in progress.

Parameters

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.

11.109.3.71 SetEnableAmplifierProtectionSwitch() [3/4]

```
virtual void SetEnableAmplifierProtection←
Switch (
            uint32_t Scu_HS,
            uint32_t electrode,
            array< bool >^ enable )  [virtual]
```

Defines whether the Amplifier Protection Switch is openend while stimulation is in progress.

Parameters

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.

11.109.3.72 SetEnableAmplifierProtectionSwitch() [4/4]

```
virtual void SetEnableAmplifierProtection←
Switch (
uint32_t Scu_HS,
uint32_t electrode,
bool enable ) [virtual]
```

Defines whether the Amplifier Protection Switch is openend while stimulation is in progress.

Parameters

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.

11.109.3.73 SetExternalElectrodeEnable() [1/2]

```
virtual void SetExternalElectrodeEnable (
uint32_t electrode,
uint32_t listmodeIndex,
array< bool >^ enable ) [virtual]
```

Enables or disables the stimulation switch for an external electrode.

Parameters

electrode	The electrode number.

Parameters

listmodeIndex	The index for listmode.
enable	1 to enable the electrode, 0 to disable.

11.109.3.74 SetExternalElectrodeEnable() [2/2]

```
virtual void SetExternalElectrodeEnable (
uint32_t electrode,
uint32_t listmodeIndex,
bool enable ) [virtual]
```

Enables or disables the stimulation switch for an external electrode.

Parameters

electrode	The electrode number.

Parameters

listmodeIndex	The index for listmode.
enable	1 to enable the electrode, 0 to disable.

11.109.3.75 SetFAAmplification()

```
virtual void SetFAAmplification (
    unsigned int amplification )  [virtual]
```

11.109.3.76 SetHeadstage()

```
virtual void SetHeadstage (
    unsigned int headstage )  [virtual]
```

11.109.3.77 SetListmodeIndexRange()

```
virtual void SetListmodeIndexRange (
    unsigned int sideband,
    unsigned int startIndex,
    unsigned int endIndex,
    unsigned int mode )  [virtual]
```

11.109.3.78 SetListmodeTriggerSource() [1/2]

```
virtual void SetListmodeTriggerSource (
    unsigned int sideband,
    TriggerSourceEnumNet triggersource )  [virtual]
```

11.109.3.79 SetListmodeTriggerSource() [2/2]

```
virtual void SetListmodeTriggerSource (
    unsigned int sideband,
    TriggerSourceEnumNet triggersource,
    int bitnumOffset )  [virtual]
```

11.109.3.80 SetMeasurementMode()

```
virtual void SetMeasurementMode (
    unsigned int channel )  [virtual]
```

Sets a channel to measurement mode (STG3008-FA).

Parameters

channel	The channel to change.

11.109.3.81 SetOutputRate()

```
void SetOutputRate (
        uint32_t rate )
```

Change the output rate of the STG. Valid rates are from 1000 Hz to 50000 Hz.

Parameters

rate	The new output rate in Hz.

11.109.3.82 SetStgProgramInfo()

```
void SetStgProgramInfo (
        DateTime timestamp,
        String^ filename,
        Guid guid )
```

Store Download information in the STG. This function can be used to store the filename and timestamp of the last download for later query.

Parameters

timestamp	The timestamp of last download.
filename	The filename of the downlaoded waveform.

11.109.3.83 SetSyncoutMap()

```
virtual void SetSyncoutMap (
        uint32_t channel,
        uint32_t syncoutMap ) [virtual]
```

Sets the mapping between external syncout outputs and internal syncout channels.

Parameters

channel	The external syncout output channel number (zero based).
syncoutMap	A bitmap of internal syncout channels to map to channel.

11.109.3.84 SetTriggerSource() [1/3]

```
virtual void SetTriggerSource (
        unsigned int triggernum,
        DigitalSource< digitalsourceenum >^ triggersource,
        int bitnum_offset )  [virtual]
```

11.109.3.85 SetTriggerSource() [2/3]

```
virtual void SetTriggerSource (
    unsigned int triggernum,
    TriggerSourceEnumNet triggersource )  [virtual]
```

11.109.3.86 SetTriggerSource() [3/3]

```
virtual void SetTriggerSource (
    unsigned int triggernum,
    TriggerSourceEnumNet triggersource,
    int bitnum_offset )  [virtual]
```

11.109.3.87 SetVoltageMode() [1/2]

```
virtual void SetVoltageMode ( )  [virtual]
```

Sets all channels to voltage mode (STG3008-FA and STG400x only).

11.109.3.88 SetVoltageMode() [2/2]

```
virtual void SetVoltageMode (
    unsigned int channel )  [virtual]
```

Sets a channel to voltage mode (STG3008-FA and STG400x only).

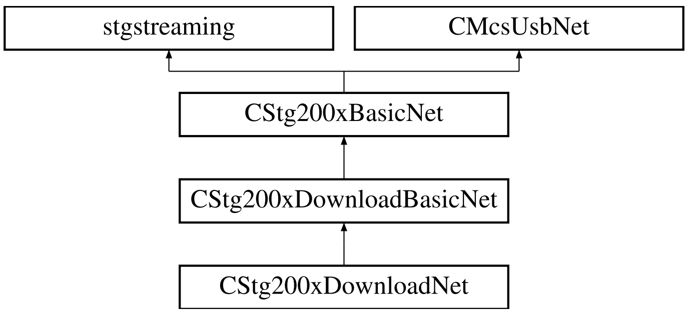
Parameters

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, RetriggerAction↩ EnumNet 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.

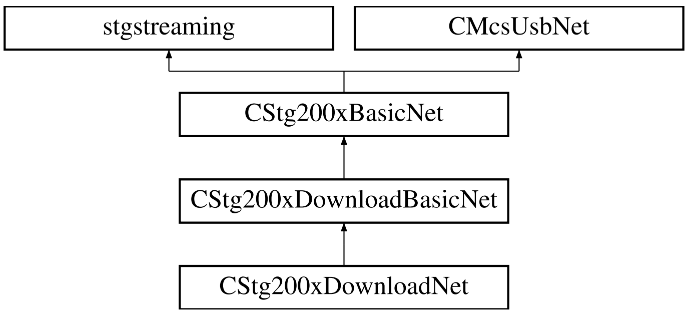
11.110.3 Property Documentation

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

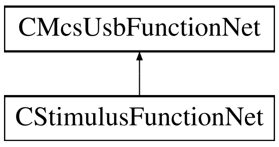
```
OnMwPollStatus^ MwPollStatusEvent [add], [remove], [raise]
```

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, S↩ TG_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

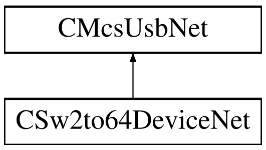
11.112.3 Event Documentation

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.

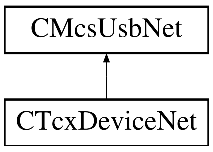
Parameters

`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°C.
- 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°C.
- void SetDevice (unsigned int channel, int device)
- void SetOnOff (unsigned int channel, bool on)

  Switches the specified channel on or off.
- void SetCalibration (unsigned int channel, int calib)
- void SetP (unsigned int device, int p_coeff)

  Sets the P-coefficient of the specified device.

- void SetI (unsigned int device, int i_coeff)

    Sets the I-coefficient of the specified device.
- void SetD (unsigned int device, int d_coeff)

    Sets the D-coefficient of the specified device.
- void SetMaxP (unsigned int device, int maxp)

    Sets the maximum heater power of the specified device.
- void SetHeaterLimit (unsigned int device, int heater_limit)
- void SetEnableHeaterLimit (unsigned int device, bool enable)
- void SetEnableThermocouple (unsigned int device, bool enable)
- void SetSensorType (unsigned int device, TcxSensorTypeEnumNet type)
- void SetDevname (unsigned int device, String^ Devicename)
- int GetSetpoint (unsigned int channel)

    Gets the target temperate of specified channel in units of 0.1 ℃.
- 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℃.

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℃.

11.114.3.78 SetThermocoupleNanovoltPerKelvin() `void SetThermocoupleNanovoltPerKelvin (`

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

Sets the proportional constant for the thermocouple.

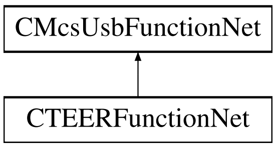
Parameters

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

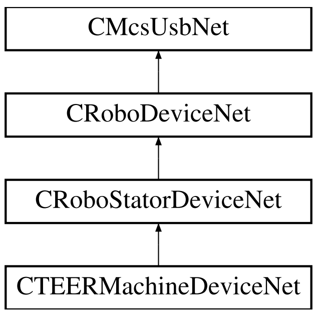
NumberOfCycles	number of cycles (sine or rect) to output (0 - loop forever)

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 ( )`

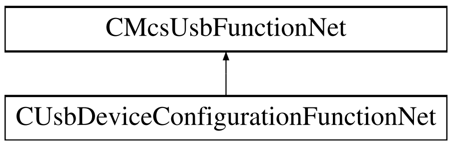
11.116.2 Property Documentation

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

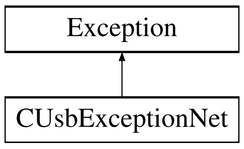
Parameters

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 )
```

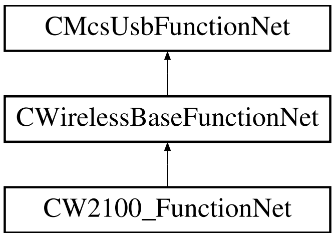
11.119.2 Member Data Documentation

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

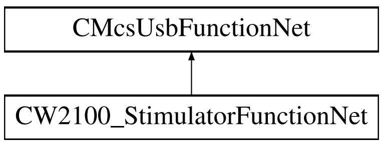
```
CPulseGeneratorFunctionNet^ PulseGenerator  [get]
```

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, DigitalStimulator↵ TriggerEventEnumNet 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 GetDigital↩
StimulatorTriggerSlope (
            int TimeSlotNr,
            DigitalStimulatorTriggerEventEnumNet trigger_event,
            int trigger_number )
```

11.121.2.9 GetNumberOfAnalogChannels() `uint32_t GetNumberOfAnalogChannels ( )`

11.121.2.10 GetNumberOfSyncoutChannels() `uint32_t GetNumberOfSyncoutChannels ( )`

11.121.2.11 GetNumberOfTriggerInputs() `uint32_t GetNumberOfTriggerInputs ( )`

11.121.2.12 GetStimulationPatternMemory() `uint32_t GetStimulationPatternMemory ( )`

11.121.2.13 GetTimeResolutionInNanoSeconds() `int GetTimeResolutionInNanoSeconds ( )`

Gets number of bits of the DAC resolution.

Returns

The time resolution in ns.

11.121.2.14 GetTimeSlot() `int GetTimeSlot ( )`

11.121.2.15 GetVoltageRangeInMicroVolt()

```
int GetVoltageRangeInMicroVolt (
    uint32_t channel )
```

Gets the Voltage Range of the specified channel in Microvolts.

Parameters

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 ^ Prepare←
Data (
            int channel,
            array< int32_t >^ amplitude,
            array< uint64_t >^ duration,
            STG_DestinationEnumNet destType,
            uint32_t repeat )
```

11.121.2.18 PrepareDataSync()

```
CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData ^ Prepare←
DataSync (
            int channel,
            array< int32_t >^ amplitude,
            array< uint32_t >^ Sync,
            array< uint64_t >^ duration,
            STG_DestinationEnumNet destType,
            uint32_t repeat )
```

11.121.2.19 SelectTimeSlot()

```
void SelectTimeSlot (
            int TimeSlotNr )
```

11.121.2.20 SendPreparedData()

```
void SendPreparedData (
            int channel,
            CStimulusFunctionNet::StimulusDeviceDataAndUnrolledData^ device_data_and_unrolled,
            STG_DestinationEnumNet destType )
```

11.121.2.21 SendStart()

```
void SendStart (
            uint32_t triggermap )
```

Start (Trigger) the STG. The startup delay is in the range of a few ms.

Parameters

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]`

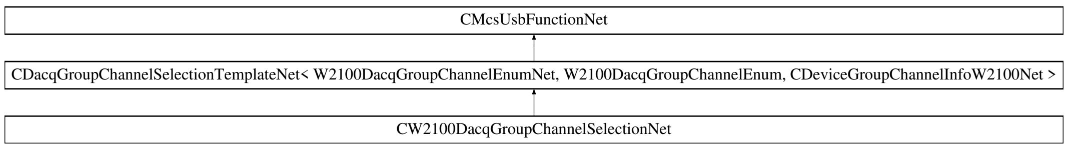
11.121.3.4 SYNC_BIT1 `const uint32_t SYNC_BIT1 = (1 << 3) [static]`

11.121.4 Event Documentation

11.121.4.1 PollStatusEvent `OnStgPollStatus^ PollStatusEvent`

11.122 CW2100DacqGroupChannelSelectionNet Class Reference

Inheritance diagram for CW2100DacqGroupChannelSelectionNet:


CMcsUsbFunctionNet
CDacqGroupChannelSelectionTemplateNet< W2100DacqGroupChannelEnumNet, W2100DacqGroupChannelEnum, CDeviceGroupChannelInfoW2100Net >
CW2100DacqGroupChannelSelectionNet

Public Member Functions

- CW2100DacqGroupChannelSelectionNet (CMcsUsbNet^ mcsusb)

Additional Inherited Members

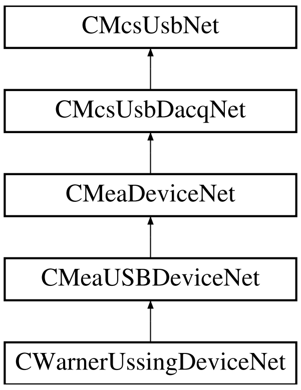
11.122.1 Constructor & Destructor Documentation

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 ( )`

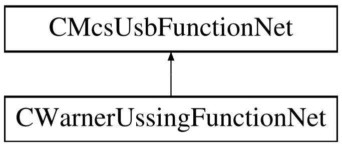
11.123.3 Property Documentation

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

Chamber↩ Id	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

Chamber↩ Id	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

Chamber↩ Id	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

Chamber↩ Id	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

Chamber↩ Id	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

Chamber↩ Id	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

Chamber↩ Id	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

Chamber↩ Id	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

Chamber↩ Id	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

Chamber↩ Id	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 GetNumberOfHardwareSlotsFor↩ Chambers ( )`

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

Chamber↩ Id	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

Chamber↩ Id	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

Chamber↩ Id	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

Chamber←↩ Id	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

Chamber←↩ Id	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

Chamber←↩ Id	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

Chamber↩ Id	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

Chamber↩ Id	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

Chamber↩ Id	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

Chamber↩ Id	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

Chamber↩ Id	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

Chamber↩ Id	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

Chamber↩ Id	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

Chamber↩ Id	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

Chamber↩ Id	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

Chamber↩ Id	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

Chamber↩ Id	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

Chamber↵ Id	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

Chamber↵ Id	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

Chamber↩ Id	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

Chamber↩ Id	index of hardware chamber slot (zero-based)

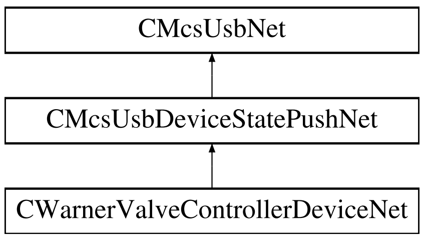
Returns

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

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 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

11.125.3.22 GetValveLedOn() `bool GetValveLedOn ( )`

Gets the LED state of the valve board

Returns

The LED state

11.125.3.23 GetValveManualGroup() `int32_t GetValveManualGroup ( uint16_t valve )`

Gets the valve manual group

Parameters

valve	The valve number

Returns

The manual valve group

11.125.3.24 GetValveManualState() `int32_t GetValveManualState ( uint16_t valve )`

Gets the valve manual on/off state

Parameters

valve	The valve number

Returns

The manual valve state

11.125.3.25 GetValveMode() `WvcValveModeEnumNet GetValveMode ( uint16_t valve )`

Reads the valve mode

Parameters

valve	The valve number

Returns

The valve mode

11.125.3.26 GetValvesActiveMap() `uint32_t GetValvesActiveMap ( )`

Gets the valves active/inactive states

Returns

The valves states

11.125.3.27 GetValvesManualStateMap() `uint32_t GetValvesManualStateMap ( )`

Gets the valves manual on/off states

Returns

The manual valves states

11.125.3.28 GetValveTableEntry() `void GetValveTableEntry (`

```
uint16_t valve,
uint16_t index,
[System::Runtime::InteropServices::Out] uint32_t% duration,
[System::Runtime::InteropServices::Out] bool% state )
```

Read an entry from the valve protocol table

Parameters

valve	The valve number
index	The index in the table
duration	the duration in ms
state	the state

11.125.3.29 IsDigitalOutPortInverted() `bool IsDigitalOutPortInverted (`

```
uint16_t digitalOutPort )
```

Gets the number of the valve which is mapped to a digital out port

Parameters

digitalOutPort	The digital out port

Returns

is inverted

11.125.3.30 IsValveDigitalInInverted() `bool IsValveDigitalInInverted ( uint16_t valve )`

Is digital in inverted

Parameters

valve	The valve number

Returns

is inverted

11.125.3.31 IsValveOpen() `bool IsValveOpen ( uint16_t valve )`

Is valve open

Parameters

valve	The valve number

Returns

is open

11.125.3.32 IsValveOpenInAnalogMode() `bool IsValveOpenInAnalogMode ( uint16_t valve )`

True, if the valve would be open when the device is in analog mode

Parameters

valve	The valve number

Returns

is open

11.125.3.33 IsValveOpenInDigitalMode() `bool IsValveOpenInDigitalMode ( uint16_t valve )`

True, if the valve would be open when the device is in digital mode

Parameters

valve	The valve number

Returns

is open

11.125.3.34 LoadValveTable() `void LoadValveTable ( )`

Load the current table from permanent memory

11.125.3.35 OnGetActiveRunningTableNumber()

```
delegate void OnGetActiveRunningTableNumber (
    uint32_t tableNumber )
```

11.125.3.36 OnGetAnalogThresholdHigh()

```
delegate void OnGetAnalogThresholdHigh (
    uint16_t valve,
    int32_t threshold )
```

11.125.3.37 OnGetAnalogThresholdLow()

```
delegate void OnGetAnalogThresholdLow (
    uint16_t valve,
    int32_t threshold )
```

11.125.3.38 OnGetAnalogVoltage()

```
delegate void OnGetAnalogVoltage (
    int32_t voltage )
```

11.125.3.39 OnGetCurrentNumberOfValves()

```
delegate void OnGetCurrentNumberOfValves (
    int32_t numberOfValves )
```

11.125.3.40 OnGetDigitalOutPortValve()

```
delegate void OnGetDigitalOutPortValve (
    uint16_t digitalOutPort,
    uint32_t valve )
```

11.125.3.41 OnGetDigitalPortDirection() `delegate void OnGetDigitalPortDirection (`

`uint16_t` *`port,`*
`PortDirectionEnumNet` *`direction`* `)`

11.125.3.42 OnGetDisplayMode() `delegate void OnGetDisplayMode (`

`WvcDisplayModeEnumNet` *`DisplayMode`* `)`

11.125.3.43 OnGetTableNamebyIndex() `delegate void OnGetTableNamebyIndex (`

`uint16_t` *`tableNumber,`*
`String^` *`tableName`* `)`

11.125.3.44 OnGetValveActive() `delegate void OnGetValveActive (`

`uint16_t` *`valve,`*
`int` *`valveActive`* `)`

11.125.3.45 OnGetValveBoardRevision() `delegate void OnGetValveBoardRevision (`

`uint32_t` *`revision`* `)`

11.125.3.46 OnGetValveDigitalInPort() `delegate void OnGetValveDigitalInPort (`

`uint16_t` *`valve,`*
`uint32_t` *`digitalInPort`* `)`

11.125.3.47 OnGetValveLedOn() `delegate void OnGetValveLedOn (`

`bool` *`ledon`* `)`

11.125.3.48 OnGetValveManualGroup() `delegate void OnGetValveManualGroup (`

`uint16_t` *`valve,`*
`int32_t` *`valveManualGroup`* `)`

11.125.3.49 OnGetValveManualState() `delegate void OnGetValveManualState (`

`uint16_t` *`valve,`*
`int32_t` *`valveManualState`* `)`

11.125.3.50 OnGetValveMode() `delegate void OnGetValveMode (`

```
uint16_t valve,
WvcValveModeEnumNet ValveMode )
```

11.125.3.51 OnIsDigitalOutPortInverted() `delegate void OnIsDigitalOutPortInverted (`

```
uint16_t digitalOutPort,
bool isInverted )
```

11.125.3.52 OnIsValveDigitalInInverted() `delegate void OnIsValveDigitalInInverted (`

```
uint16_t valve,
bool isInverted )
```

11.125.3.53 OnIsValveOpen() `delegate void OnIsValveOpen (`

```
uint16_t valve,
bool valveOpen )
```

11.125.3.54 OnIsValveOpenInAnalogMode() `delegate void OnIsValveOpenInAnalogMode (`

```
uint16_t valve,
bool valveOpen )
```

11.125.3.55 OnIsValveOpenInDigitalMode() `delegate void OnIsValveOpenInDigitalMode (`

```
uint16_t valve,
bool valveOpen )
```

11.125.3.56 OnTableEntryChanged() `delegate void OnTableEntryChanged (`

```
uint16_t tableNumber )
```

11.125.3.57 SetActiveRunningTableNumber() `void SetActiveRunningTableNumber (`

```
uint32_t tableNumber )
```

Sets the number of the tanle that is active for running

Parameters

tableNumber	The table number

11.125.3.58 SetAnalogThresholdHigh()

```
void SetAnalogThresholdHigh (
            uint16_t valve,
            int32_t threshold )
```

Sets the upper threshold for the analog in port per valve

Parameters

valve	The valve number
threshold	The threshold in mV

11.125.3.59 SetAnalogThresholdLow()

```
void SetAnalogThresholdLow (
            uint16_t valve,
            int32_t threshold )
```

Sets the lower threshold for the analog in port per valve

Parameters

valve	The valve number
threshold	The threshold in mV

11.125.3.60 SetCurrentEditTableNumber()

```
void SetCurrentEditTableNumber (
            uint32_t tableNumber )
```

Sets the number of the table that is current for editing

Parameters

tableNumber	The table number

11.125.3.61 SetDefault()

```
void SetDefault ( )
```

Sets the settings of the valve controller to default

11.125.3.62 SetDigitalOutPortInvert()

```
void SetDigitalOutPortInvert (
            uint16_t digitalOutPort,
            bool isInverted )
```

Map a valve to a digital out port

Parameters

digitalOutPort	The digital out port
isInverted	True if digital out is to be inverted

11.125.3.63 SetDigitalOutPortValve()

```
void SetDigitalOutPortValve (
    uint16_t digitalOutPort,
    uint32_t valve )
```

Map a valve to a digital out port

Parameters

digitalOutPort	The digital out port
valve	The valve number

11.125.3.64 SetDigitalPortDirection()

```
void SetDigitalPortDirection (
    uint16_t port,
    PortDirectionEnumNet direction )
```

Sets the direction of a digital port

Parameters

port	The port number
direction	the direction

11.125.3.65 SetDisplayMode()

```
void SetDisplayMode (
    WvcDisplayModeEnumNet DisplayMode,
    int32_t lockTimeMs )
```

Writes the display mode

Parameters

DisplayMode	The display mode
lockTimeMs	Locks the display for ms

11.125.3.66 SetTableName()

```
void SetTableName (
    String^ tableName )
```

Set the name of the current protocol table

Parameters

tableName	The name of the table

11.125.3.67 SetTableStep()

```
void SetTableStep (
    uint16_t valve,
    int32_t steps )
```

Skips the table protocol for a valve by steps

Parameters

valve	The valve number
steps	Number of steps

11.125.3.68 SetTableStepAll()

```
void SetTableStepAll (
    int32_t steps )
```

Skips the table protocol for all valves by steps

Parameters

steps	Number of steps

11.125.3.69 SetValveActive()

```
void SetValveActive (
    uint16_t valve,
    int valveActive )
```

Sets the valve active/inactive state

Parameters

valve	The valve number
valveActive	The valve state

11.125.3.70 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

11.125.3.71 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

11.125.3.72 SetValveLedOn()

```
void SetValveLedOn (
    bool ledon )
```

Gets the LED state of the valve board

Parameters

ledon	The LED state

11.125.3.73 SetValveManualGroup()

```
void SetValveManualGroup (
    uint16_t valve,
    int32_t valveManualGroup )
```

Sets the valve manual group

Parameters

valve	The valve number
valveManualGroup	The manual valve group

11.125.3.74 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

11.125.3.75 SetValveMode()

```
void SetValveMode (
    uint16_t valve,
    WvcValveModeEnumNet ValveMode )
```

Writes the valve mode

Parameters

valve	The valve number
ValveMode	The valve mode

11.125.3.76 SetValvesActiveMap()

```
void SetValvesActiveMap (
    uint32_t valvesActive )
```

Sets the valve active/inactive state

Parameters

valvesActive	The valves states

11.125.3.77 SetValvesManualStateMap()

```
void SetValvesManualStateMap (
    uint32_t valveaManualState )
```

Sets the valve manual on/off state

Parameters

valveaManualState	The manual valves states

11.125.3.78 SetValveTableEntry()

```
void SetValveTableEntry (
    uint16_t valve,
    uint16_t index,
    uint32_t duration,
    bool state )
```

Write an entry to the valve protocol table

Parameters

valve	The valve number
index	The index in the table
duration	the duration in ms
state	the state

11.125.3.79 StoreValveTable() `void StoreValveTable ( )`

Store the current table in permanent memory

11.125.4 Event Documentation

11.125.4.1 GetActiveRunningTableNumberEvent `OnGetActiveRunningTableNumber^ GetActiveRunning↩ TableNumberEvent [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^ IsValveDigitalInInverted←Event [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

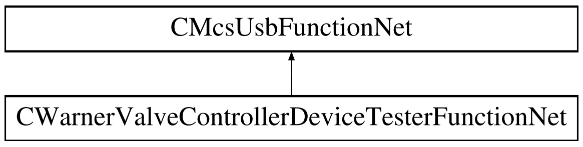
11.125.4.22 TableEntryChangedEvent `OnTableEntryChanged^ TableEntryChangedEvent [add], [remove], [raise]`

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 )`

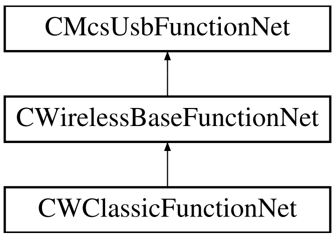
Sets the direction of the trigger/sync test port

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()

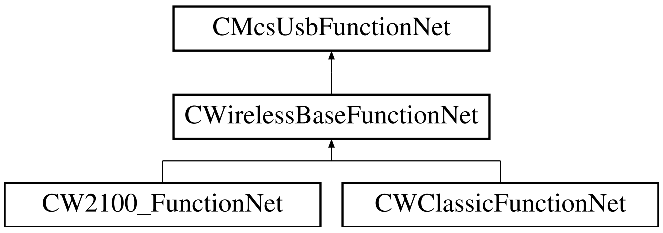
```
void SetWPADebugMode (
    unsigned int mode,
    unsigned int Device )
```

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]`

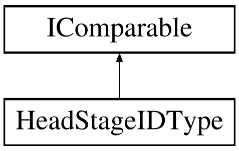
11.133.1.27 PIC3 `String ^ PIC3 = gcnew String( "PIC3" ) [static]`

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 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 )`

```
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