Namespace List
Here is a list of all namespaces with brief descriptions:

Mcs

Mcs::Usb

Generated on Fri Jan 9 2015 13:58:50 for McsUsbNet.dll for STG by dox *1.7.6.1*

- Main Page
- Namespaces
- Classes

- Namespace List
- Namespace Members

Namespaces
Mcs Namespace Reference

Namespaces

namespace Usb

Generated on Fri Jan 9 2015 13:58:50 for McsUsbNet.dll for STG by dox *1.7.6.1*

- Main Page
- Namespaces
- Classes

- Namespace List
- Namespace Members

- Mcs
- Usb

Classes | Enumerations | Functions
Mcs::Usb Namespace Reference

Classes

class	CUsbExceptionNet
	Exception class that is thrown in case of an USB error. More...
class	FirmwareDestinationNames
struct	DeviceIdNet

Device Id. More...

class DriverVersionNet

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

class CMcsUsbListEntryNet

McsUsbListEntryNet identifies a connected device. More...

class CMcsUsbListNet

Class to handle a list of connected MCS USB devices. More...

class CMcsUsbPointerContainer

class CMcsUsbNet

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

class CStg200xBasicNet

Base class for the Stg200x. More...

class CStg200xDownloadBasicNet

Base class for the STG200x series download mode. More...

class CStg200xDownloadNet

Main class for the STG download mode. More...

class CStg200xStreamingNet

Main class for the STG streaming mode. More...

Enumerations

enum EnSTG200x_STATUS {
OK,
NOT_CONNECTED,
DEVICE_NOT_FOUND
}

enum EnSTG200x_TRIGGER_STATUS {
STG200x_TRIGGER_IDLE,
STG200x_TRIGGER_RUNNING,
STG200x_TRIGGER_FINISHED
}

Functions

public delegate void OnDeviceArrivalRemoval (CMcsUsbListEntryNet^ entry)

Delegate to show a device arrival or removal.

public delegate void OnStg200xPollStatus (unsigned int status, array< int >^index_list)

public delegate void OnMwPollStatus (unsigned int CurrentTemp, unsigned int PlateState, unsigned int SwitchState)

public delegate void OnStg200xDataHandler (uint32_t trigger)

public
delegate
void OnStg200xErrorHandler ()

Enumeration Type Documentation

enum EnSTG200x_STATUS

Enumerator:

OK
NOT_CONNECTED
DEVICE_NOT_FOUND

enum EnSTG200x_TRIGGER_STATUS

Enumerator:

STG200x_TRIGGER_IDLE
STG200x_TRIGGER_RUNNING
STG200x_TRIGGER_FINISHED

Function Documentation

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

Delegate to show a device arrival or removal.

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

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

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

public delegate void Mcs::Usb::OnStg200xPollStatus (unsigned int *status*,
array< int >^ *index_list*
)

Generated on Fri Jan 9 2015 13:58:50 for McsUsbNet.dll for STG by dox *1.7.6.1*

- Main Page
- Namespaces
- Classes

- Class List
- Class Hierarchy
- Class Members

- Mcs
- Usb
- CUsbExceptionNet

Public Member Functions | Properties

CUsbExceptionNet Class Reference

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

List of all members.

Public Member Functions

CUsbExceptionNet (uint32_t status)
Constructor of a CUsbException.
CUsbExceptionNet (uint32_t status, String^ message)

Properties

uint32_t Status [get]

Detailed Description

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

Constructor & Destructor Documentation

CUsbExceptionNet (uint32_t *status*)

Constructor of a CUsbException.

Parameters:

status the status number

CUsbExceptionNet (uint32_t *status*,
String^ *message*
)

Property Documentation

uint32_t Status [get]

Generated on Fri Jan 9 2015 13:58:50 for McsUsbNet.dll for STG by dox *1.7.6.1*

- Main Page
- Namespaces
- Classes

- Class List
- Class Hierarchy
- Class Members

- Mcs
- Usb
- FirmwareDestinationNames

Static Public Attributes
FirmwareDestinationNames Class Reference

List of all members.

Static Public Attributes

static String^ DSP = gcnew String("DSP")
static String^ USB = gcnew String("USB")
static String^ MCU1 = gcnew String("MCU1")
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")

Member Data Documentation

String ^ Altera = gcnew String("Altera") `[static]`
String ^ BUS1_MCSBUS1 = gcnew String("Bus1McsBus1") `[static]`
String ^ BUS1_MCSBUS2 = gcnew String("Bus1McsBus2") `[static]`
String ^ DSP = gcnew String("DSP") `[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") `[static]`
String ^ MCSBUS1 = gcnew String("McsBus1") `[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 ^ 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 ^ MCU1 = gcnew String("MCU1") `[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 ^ USB = gcnew String("USB") `[static]`

Generated on Fri Jan 9 2015 13:58:50 for McsUsbNet.dll for STG by dox *1.7.6.1*

- Main Page
- Namespaces
- Classes

- Class List
- Class Hierarchy
- Class Members

- Mcs
- Usb
- DeviceIdNet

Public Member Functions | Public Attributes

DeviceIdNet Struct Reference

Device Id. More...

List of all members.

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

Detailed Description

Device Id.

Constructor & Destructor Documentation

DeviceIdNet ( )

DeviceIdNet (VendorIdEnumNet *vendor*,

ProductIdEnumNet	*product*,
int	*bcd*,
McsBusTypeEnumNet	*bustype*

)

DeviceIdNet (DeviceIdNet% *deviceId*)

Member Function Documentation

DeviceIdNet operator= (DeviceIdNet% *deviceId*)

Member Data Documentation

int BcdDevice

McsBusTypeEnumNet BusType

ProductIdEnumNet IdProduct

VendorIdEnumNet IdVendor

Generated on Fri Jan 9 2015 13:58:50 for McsUsbNet.dll for STG by dox *1.7.6.1*

- Main Page
- Namespaces
- Classes

- Class List
- Class Hierarchy
- Class Members

- Mcs
- Usb
- DriverVersionNet

Public Member Functions | Static Public Member Functions

DriverVersionNet Class Reference

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

List of all members.

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.

Static Public Member Functions

static String^ DriverVersionNet::FormatVersion (unsigned int v)

Detailed Description

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

Constructor & Destructor Documentation

DriverVersionNet ( )

Contructor.

~DriverVersionNet ( )

Destructor.

Member Function Documentation

static String ^ DriverVersionNet::FormatVersion (unsigned int *v*) `[static]`
CFirmwareDestinationNet GetDestinationCode (unsigned int *index*)

Get CFirmwareDestinationNet.

Parameters:
index by index of firmware destination

String ^ GetDestinationName (CFirmwareDestinationNet *dest*)

Get firmware destination name.

Parameters:
dest by CFirmwareDestionationNet

String ^ GetDestinationName (unsigned int *index*)

Get firmware destination name.

Parameters:
index by index of firmware destination

unsigned int GetMajor (CFirmwareDestinationNet *dest*)

Get the major version number of firmware destination.

Parameters:
dest by CFirmwareDestionationNet

unsigned int GetMajor (unsigned int *index*)

Get the major version number of firmware destination.

Parameters:
index by index of firmware destination

unsigned int GetMinor (CFirmwareDestinationNet *dest*)

Get the minor version number of firmware destination.

Parameters:
dest by CFirmwareDestionationNet

unsigned int GetMinor (unsigned int *index*)

Get the minor version number of firmware destination.

Parameters:
index by index of firmware destination

unsigned int GetNumEntries ( )

Get the number of available firmware destinations.

unsigned int GetStatus (CFirmwareDestinationNet *dest*)

Get status of firmware destination.

Parameters:
dest by CFirmwareDestionationNet

unsigned int GetStatus (unsigned int *index*)

Get status of firmware destination.

Parameters:
index by index of firmware destination

unsigned int GetVersionInt (CFirmwareDestinationNet *dest*)

Get the version number of firmware destination (major in high word, minor in low word)

Parameters:
dest by CFirmwareDestionationNet

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

String ^ GetVersionString (CFirmwareDestinationNet *dest*)

Get the version as a string in the format Major.Minor.

Parameters:
dest by CFirmwareDestionationNet

String ^ GetVersionString (unsigned int *index*)

Get the version as a string in the format Major.Minor.

Parameters:
index by index of firmware

Generated on Fri Jan 9 2015 13:58:50 for McsUsbNet.dll for STG by dox *1.7.6.1*

- Main Page
- Namespaces
- Classes

- Class List
- Class Hierarchy
- Class Members

- Mcs
- Usb
- CMcsUsbListEntryNet

Public Member Functions | Properties
CMcsUsbListEntryNet Class Reference

McsUsbListEntryNet identifies a connected device. More...

List of all members.

Public Member Functions

	CMcsUsbListEntryNet ()
	Initializes a new instance of the CMcsUsbListEntryNet class.
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

Properties

String^ Manufacturer
The Manufacturer ID of the device represented by this CMcsUsbListEntryNet.

String^ Product
The Product ID of the device represented by this CMcsUsbListEntryNet.

String^ DeviceName
The device name of the device represented by this CMcsUsbListEntryNet.

String^ SerialNumber
The serial number of the device represented by this CMcsUsbListEntryNet.

String^ HwVersion
The hardware revision of the device represented by this CMcsUsbListEntryNet.

String^ DevicePath
The DevicePath of the device represented by this CMcsUsbListEntryNet.

DeviceIdNet^ DeviceId

Detailed Description

McsUsbListEntryNet identifies a connected device.

Constructor & Destructor Documentation

CMcsUsbListEntryNet ( )

Initializes a new instance of the CMcsUsbListEntryNet class.

Member Function Documentation

virtual bool Equals (Object^ *obj*) `[override, virtual]`

Checks weather two CMcsUsbListEntryNet represent the same USB device.

Parameters:

obj The CMcsUsbListEntryNet to compare with.

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)".

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

Property Documentation

DeviceIdNet^ DeviceId

String^ DeviceName

The device name of the device represented by this CMcsUsbListEntryNet.

String^ DevicePath

The DevicePath of the device represented by this CMcsUsbListEntryNet.

String^ HwVersion

The hardware revision of the device represented by this CMcsUsbListEntryNet.

String^ Manufacturer

The Manufacturer ID of the device represented by this CMcsUsbListEntryNet.

String^ Product

The Product ID of the device represented by this CMcsUsbListEntryNet.

String^ SerialNumber

The serial number of the device represented by this CMcsUsbListEntryNet.

Generated on Fri Jan 9 2015 13:58:50 for McsUsbNet.dll for STG by dox *1.7.6.1*

- Main Page
- Namespaces
- Classes

- Class List
- Class Hierarchy
- Class Members

- Mcs
- Usb
- CMcsUsbListNet

Public Member Functions | Properties | Events

CMcsUsbListNet Class Reference

Class to handle a list of connected MCS USB devices. More...

List of all members.

Public Member Functions

	CMcsUsbListNet () Initializes a new instance of CMcsUsbListNet class.
	CMcsUsbListNet (OnDeviceArrivalRemoval^ devArrival, OnDeviceArrivalRemoval^ devRemoval) Initializes a new instance of CMcsUsbListNet class.
	~CMcsUsbListNet () Destructor: called by Dispose()
	!CMcsUsbListNet () Finalizer: called by GC before collecting.
void	Initialize (DeviceEnumNet McsUsbDevice) Initialize/Update the list of devices which are currently connected to the computer.
void	Initialize (array< DeviceIdNet^ >^DeviceIdList) Initialize/Update the list of devices which are currently connected to the computer.
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

Detailed Description

Class to handle a list of connected MCS USB devices.

Constructor & Destructor Documentation

CMcsUsbListNet ( )

Initializes a new instance of CMcsUsbListNet class.

CMcsUsbListNet (OnDeviceArrivalRemoval^ *devArrival*,
OnDeviceArrivalRemoval^ *devRemoval*
)

Initializes a new instance of CMcsUsbListNet class.

Parameters:

devArrival Callback to call when a new device is attached to the bus.

devRemoval Callback to call when a device is removed from the bus.

~CMcsUsbListNet ( )

Destructor: called by Dispose()

!CMcsUsbListNet ( )

Finalizer: called by GC before collecting.

Member Function Documentation

uint32_t GetNumberOfDevices ( )

Gets the number of devices currently in the list.

array<CMcsUsbListEntryNet^> ^ GetUsbListEntries ( )

Returns all entries from the list of USB Devices connected to the computer.

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.

void Initialize (DeviceEnumNet *McsUsbDevice*)

Initialize/Update the list of devices which are currently connected to the computer.

Parameters:

McsUsbDevice Specifies the type of devices to look for.

void Initialize (array< DeviceIdNet^ >^ *DeviceIdList*)

Initialize/Update the list of devices which are currently connected to the computer.

Parameters:

DeviceIdList Specifies a list of devices to look for.

bool IsDeviceTypeOf (CMcsUsbListEntryNet^ *entry*,
DeviceEnumNet *McsUsbDevice*
)

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)".

Property Documentation

uint32_t Count `[get]`

Gets the number of devices currently in the list.

Event Documentation

OnDeviceArrivalRemoval^ DeviceArrival
OnDeviceArrivalRemoval^ DeviceRemoval

Generated on Fri Jan 9 2015 13:58:50 for McsUsbNet.dll for STG by dox *1.7.6.1*

- Main Page
- Namespaces
- Classes

- Class List
- Class Hierarchy
- Class Members

- Mcs
- Usb
- CMcsUsbPointerContainer

Package Functions | Package Attributes
CMcsUsbPointerContainer Class Reference

List of all members.

Package Functions

CMcsUsbPointerContainer (CMcsUsb *pMcsUsb)

Package Attributes

CMcsUsb * Pointer

Constructor & Destructor Documentation

CMcsUsbPointerContainer (CMcsUsb * *pMcsUsb*) `[package]`

Member Data Documentation

CMcsUsb* Pointer `[package]`

Generated on Fri Jan 9 2015 13:58:50 for McsUsbNet.dll for STG by dox *1.7.6.1*

- Main Page
- Namespaces
- Classes

- Class List
- Class Hierarchy
- Class Members

- Mcs
- Usb
- CMcsUsbNet

Public Member Functions | Static Public Member Functions | Static Public Attributes | Package Attributes | Properties

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. More...

Inheritance diagram for CMcsUsbNet:

List of all members.

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 (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)
array< BYTE >^	GetSoftwareKey (unsigned int index)
void	SetSoftwareKey (unsigned int index, array< BYTE >^buffer)
void	RemoveSoftwareKey (unsigned int index)
void	AddSoftwareKey (String^ key)
bool	ValidKey (String^ key)
bool	ValidKey (String^ key, const BYTE ProgrammID, const BYTE majorversion)
bool	HasSoftwareKey (const BYTE ProgrammID, const BYTE majorversion)
bool	HasSoftwareKey (SoftwareKeyProgrammIdsNet::ProgrammIdsNet ProgrammID, const BYTE majorversion)
String^	GetSoftwareKeyString (const BYTE ProgrammID, const BYTE majorversion)
String^	GetSoftwareKeyString (SoftwareKeyProgrammIdsNet::ProgrammIdsNet ProgrammID, const BYTE majorversion)
bool	IsDeviceHighSpeedCapable ()
bool	IsDeviceHighSpeed ()
BYTE	GetDeviceCapableSpeed ()
BYTE	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)
unsigned int	ReadRegisterTimeSlot (unsigned int reg, int TimeSlot)
void	WriteRegister (unsigned int reg, unsigned int value)
void	WriteRegister (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 (unsigned int TargetOffset, unsigned int DeviceOffset, unsigned int DMA_reg, [System::Runtime::InteropServices::Out]unsigned int% DMA_value)
void	WriteEepromRegisterPreconfig (unsigned int TargetOffset, unsigned int DeviceOffset, unsigned int DMA_reg, unsigned int DMA_value)
void	EraseEepromRegisterPreconfig (unsigned int TargetOffset, unsigned int DeviceOffset, unsigned int DMA_reg)
unsigned int	GetLastUSBError ()
uint32_t	IfStatusGetLastUSBError (uint32_t status)
void	ThrowCUsbExceptionNet (uint32_t status)

unsigned int GetRFConnectionStatus ()
unsigned int GetImplantatVoltage ()
String^ GetHardwareRevision ()
unsigned int GetFirmwareVersion (CFirmwareDestinationNet destination)
UCHAR GetConfiguration ()
void SetConfiguration (UCHAR config)

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_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)

	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	WPAError_ScanningIsPending = ((0xA0220000L) | 0x0036)

Package Attributes

CMcsUsb *	m_pMcsUsb

Properties

virtual String^	SerialNumber `[get]`

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.

Constructor & Destructor Documentation

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.

Parameters:

bustype Type of device to use, either USB or PCI.

virtual ~CMcsUsbNet ( ) `[virtual]`

!CMcsUsbNet ( )

Member Function Documentation

void AddSoftwareKey (String^ *key*)

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.

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.

virtual void Disconnect ( ) [virtual]

Disconnect from a device.

void EraseEepromRegisterPreconfig (unsigned int *TargetOffset*,
unsigned int *DeviceOffset*,
unsigned int *DMA_reg*
)

UCHAR GetConfiguration ( )

BYTE GetDeviceCapableSpeed ( )

DeviceEnumNet GetDeviceEnum ( )

DeviceIdNet ^ GetDeviceId ( )

BYTE 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.

static String ^ GetErrorText (unsigned int *Status*) [static]

Gets the error text string that belongs to a status number.

Parameters:

[in] Status the status number you want the text for

Returns:

Error text string that belongs to the status number

unsigned int GetFirmwareVersion (CFirmwareDestinationNet *destination*)
String ^ GetHardwareRevision ( )
uint32_t GetIdent ([System::Runtime::InteropServices::Out] String^ % *Answer*)
unsigned int GetImplantatVoltage ( )
unsigned int GetLastUSBError ( )
unsigned int GetRFConnectionStatus ( )
virtual String ^ GetSerialNumber ( ) `[virtual]`

Query the Serial Number of the device.

Returns:

The Serial Number.

array<BYTE> ^ GetSoftwareKey (unsigned int *index*)
String ^ GetSoftwareKeyString (const BYTE *ProgrammID*,
const BYTE *majorversion*
)
String ^ GetSoftwareKeyString (SoftwareKeyProgrammIdsNet::ProgrammIdsNet *ProgrammID*,
const BYTE *majorversion*
)
virtual uint32_t GetStatus ([System::Runtime::InteropServices::Out] uint32_t% *iStatus*) `[virtual]`
CMcsUsbListEntryNet ^ GetUsbListEntry ( )
DriverVersionNet ^ GetVersion ( )
DriverVersionNet ^ GetVersion (CFirmwareDestinationNet *dest*)
bool HasSoftwareKey (const BYTE *ProgrammID*,
const BYTE *majorversion*
)
bool HasSoftwareKey (SoftwareKeyProgrammIdsNet::ProgrammIdsNet *ProgrammID*,
const BYTE *majorversion*
)
uint32_t IfStatusGetLastUSBError (uint32_t *status*)
virtual bool IsConnected ( ) `[virtual]`

Check if a device is Connected.

Returns:
true if the device is connected.

```
bool IsDeviceHighSpeed ( )
bool IsDeviceHighSpeedCapable ( )
bool ReadEepromRegisterPreconfig ( unsigned int                                   TargetOffset,
                                   unsigned int                                   DeviceOffset,
                                   unsigned int                                   DMA_reg,
                                   [System::Runtime::InteropServices::Out] unsigned int%  DMA_value
                                 )
unsigned int ReadRegister ( unsigned int  reg )
unsigned int ReadRegisterTimeSlot ( unsigned int  reg,
                                    int           TimeSlot
                                  )
void RemoveSoftwareKey ( unsigned int  index )
void SetConfiguration ( UCHAR  config )
void SetSoftwareKey ( unsigned int        index,
                      array< BYTE >^  buffer
                    )
void ThrowCUsbExceptionNet ( uint32_t  status )
unsigned int TxnGetSerialNumber ( )
void TxnSetSerialNumber ( unsigned int  number )
unsigned int TxnTestMemoryReadAndCheck ( unsigned short  index )
unsigned int TxnTestMemoryWrite ( unsigned short  index )
bool ValidKey ( String^  key )
bool ValidKey ( String^        key,
                const BYTE  ProgrammID,
                const BYTE  majorversion
              )
void WriteEepromRegisterPreconfig ( unsigned int  TargetOffset,
                                    unsigned int  DeviceOffset,
                                    unsigned int  DMA_reg,
                                    unsigned int  DMA_value
                                  )
void WriteRegister ( unsigned int  reg,
                     unsigned int  value
                   )
void WriteRegister ( 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
                           )
```

Member Data Documentation

CMcsUsb* m_pMcsUsb [package]

const uint32_t Status_AlreadyConfigured = (0xE0110001L) [static]

const uint32_t Status_BadStartFrame = (0xE0100A00L) [static]

const uint32_t Status_Btstuff = (0xE0100002L) [static]

const uint32_t Status_BufferOverrun = (0xE010000CL) [static]

const uint32_t Status_BufferUnderrun = (0xE010000DL) [static]

const uint32_t Status_Canceled = (0xE0110000L) [static]

const uint32_t Status_Canceling = (0xE0120000L) [static]

const uint32_t Status_ConnectedPipes = (0xE01F000AL) [static]

const uint32_t Status_ControlNotOwned = (0xE0100D00L) [static]

const uint32_t Status_Crc = (0xE0100001L) [static]

const uint32_t Status_DataOverrun = (0xE0100008L) [static]

const uint32_t Status_DataToggleMismatch = (0xE0100003L) [static]

const uint32_t Status_DataUnderrun = (0xE0100009L) [static]

const uint32_t Status_DeviceLocked = (0xE01F0010L) [static]

const uint32_t Status_DeviceNotFound = (0xE01F0003L) [static]

const uint32_t Status_DeviceRemoved = (0xE01F0008L) [static]

const uint32_t Status_DevNotResponding = (0xE0100005L) [static]

const uint32_t Status_EndpointHalted = (0xE0100030L) [static]

const uint32_t Status_ErrorBusy = (0xE0100400L) [static]

const uint32_t Status_ErrorShortTransfer = (0xE0100900L) [static]

const uint32_t Status_Fifo = (0xE0100010L) [static]

const uint32_t Status_FrameControlOwned = (0xE0100C00L) [static]

const uint32_t Status_InternalHcError = (0xE0100800L) [static]

const uint32_t Status_InvalidParameter = (0xE0100300L) [static]

const uint32_t Status_InvalidPipeHandle = (0xE0100600L) [static]
const uint32_t Status_InvalidUrbFunction = (0xE0100200L) [static]
const uint32_t Status_IoPending = (0xE01F0006L) [static]
const uint32_t Status_IoTimeout = (0xE01F0007L) [static]
const uint32_t Status_IsochRequestFailed = (0xE0100B00L) [static]
const uint32_t Status_NoBandwidth = (0xE0100700L) [static]
const uint32_t Status_NoMemory = (0xE0100100L) [static]
const uint32_t Status_NoSuchDevice = (0xE01F0002L) [static]
const uint32_t Status_NotAccessed = (0xE010000FL) [static]
const uint32_t Status_NotSupported = (0xE01F0005L) [static]
const uint32_t Status_PidCheckFailure = (0xE0100006L) [static]
const uint32_t Status_PipeNotLinked = (0xE01F0009L) [static]
const uint32_t Status_RequestFailed = (0xE0100500L) [static]
const uint32_t Status_Stall = (0xE0100004L) [static]
const uint32_t Status_Unconfigured = (0xE0110002L) [static]
const uint32_t Status_UnexpectedPid = (0xE0100007L) [static]
const uint32_t WPAError_ScanningIsPending = ((0xA0220000L) | 0x0036) [static]

Property Documentation

virtual String^ SerialNumber [get]

Generated on Fri Jan 9 2015 13:58:50 for McsUsbNet.dll for STG by dox *1.7.6.1*

- Main Page
- Namespaces
- Classes

- Class List
- Class Hierarchy
- Class Members

- Mcs
- Usb
- CStg200xBasicNet

Public Member Functions
CStg200xBasicNet Class Reference

Base class for the Stg200x. More...

Inheritance diagram for CStg200xBasicNet:

List of all members.

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 (bool% IsProgrammed, System::Runtime::InteropServices::ComTypes::FILETIME% timestamp, String^ %filename, Guid% guid)
Queries Download information from the STG.

void GetStgProgramInfo (bool% IsProgrammed, DateTime% timestamp, String^ %filename, Guid% guid)
Queries Download information from the STG.

void SetStgProgramInfo (DateTime timestamp, String^ filename, Guid guid)
Store Download information in the STG.

uint32_t GetMemory ()
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 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 enable)

virtual bool GetAutocalibrationDisabled (unsigned int channel)

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)

virtual uint32_t GetElectrodeMode (uint32_t electrode)
Gets the mode an electrode is in.

virtual void SetElectrodeDacMux (uint32_t electrode, uint32_t index, array< uint32_t >^dac)
Defines the DAC to use for an electrode.

virtual void SetElectrodeDacMux (uint32_t electrode, uint32_t index, uint32_t dac)

virtual uint32_t GetElectrodeDacMux (uint32_t electrode, uint32_t index)
Gets the DAC which is used for an electrode.

virtual void SetElectrodeEnable (uint32_t electrode, uint32_t index, array< bool

>^enable)

Enables or disables the stimulation switch for an electrode.

virtual void SetElectrodeEnable (uint32_t electrode, uint32_t index, bool enable)

virtual bool GetElectrodeEnable (uint32_t electrode, uint32_t index)

Gets weather an electrode is enabled or disabled for stimulation.

virtual void SetBlankingEnable (unsigned int electrode, bool enable)

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

virtual void SetBlankingEnable (unsigned int electrode, array< bool >^enable)

virtual bool GetBlankingEnable (unsigned int electrode)

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

virtual void SetEnableAmplifierProtectionSwitch (unsigned int electrode, bool enable)

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

virtual void SetEnableAmplifierProtectionSwitch (unsigned int electrode, array< bool >^enable)

virtual bool GetEnableAmplifierProtectionSwitch (unsigned int electrode)

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

virtual uint32_t GetNumberOfStimulationElectrodes ()

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 &EndIndex, unsigned int &Mode)

virtual void SetListmodeTriggerSource (unsigned int Sideband, TriggerSourceEnumNet Triggersource, int bitnum_offset)

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.

Detailed Description

Base class for the Stg200x.

From this class all STG related classes are derived: UsbNetDll::CStg200xDownloadBasicNet UsbNetDll::CStg200xDownloadNet for Download Mode and UsbNetDll::CStg200xStreamingNet for Streaming Mode. CStg200xBasicNet is the base class to control MCS STG device.

Constructor & Destructor Documentation

virtual ~CStg200xBasicNet ( ) `[virtual]`

The destructor.

Member Function Documentation

```
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.

```
void GetAnalogResolution ( int          channel,
                           [Out] int%   URes,
                           [Out] int%   IRes
                         )
```

Gets the resolution of the analog outputs.

Parameters:

channel The channel which is queried.

<param name="URes> The Voltage resolution in mV.</param> <param name="IRes> The Current resolution in uA.

virtual bool GetAutocalibrationDisabled (unsigned int *channel*) [virtual]

virtual bool GetBlankingEnable (unsigned int *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.

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.

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.

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.

virtual int32_t GetDACResolution ( ) [virtual]

Gets number of bits of the DAC resolution.

Returns:
The DAC resolution in bits.

virtual uint32_t GetElectrodeDacMux (uint32_t *electrode*,
uint32_t *index*
) [virtual]

Gets the DAC which is used for an electrode.

Parameters:
electrode The electrode number.
index The index for listmode.

Returns:
The DAC in use, can be 1, 2 or 3. If the electrode is grounded 0 is returned.

virtual bool GetElectrodeEnable (uint32_t *electrode*,
uint32_t *index*
) [virtual]

Gets weather an electrode is enabled or disabled for stimulation.

Parameters:
electrode The electrode number.
index The index for listmode.

Returns:
true if the electrode is enabled, false if it is disabled.

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.

virtual bool GetEnableAmplifierProtectionSwitch (unsigned int *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.

virtual uint32_t GetFAAmplification ( ) [virtual]

virtual uint32_t GetHeadstage ( ) [virtual]

virtual void GetListmodeIndexRange (unsigned int *Sideband*,
unsigned int & *StartIndex*,
unsigned int & *EndIndex*,
unsigned int & *Mode*
) [virtual]

virtual TriggerSourceEnumNet GetListmodeTriggerSource (unsigned int *Sideband*) [virtual]

uint32_t GetMemory ( )

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.

virtual uint32_t GetNumberOfAnalogChannels ( ) [virtual]

Gets the Number of available analog channels of the device.

Returns:
The number of analog channels.

virtual uint32_t GetNumberOfHWDACPaths ( ) [virtual]

Gets the Number of HW Stimulation DACs of the device.

Returns:
The number of independent HW Stimulation outputs.

virtual uint32_t GetNumberOfStimulationElectrodes ( ) [virtual]

virtual uint32_t GetNumberOfSyncoutChannels ( ) [virtual]

Gets the Number of available syncout channels of the device.

Returns:
The number of analog channels.

virtual uint32_t GetNumberOfTriggerInputs ( ) [virtual]

Gets the Number of trigger inputs of the device.

Returns:
The number of trigger inputs.

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.

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

Queries Download information from the STG.

If download information was stored by the use of CStg200xBasic::SetStgProgramInfo, this function can be used to query the timestamp and filename of the last download.

Parameters:

[out] IsProgrammed flag wether download information is valid
[out] timestamp timestamp of last download
[out] filename filename of the downlaoded waveform

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

Queries Download information from the STG.

If download information was stored by the use of CStg200xBasic::SetStgProgramInfo, this function can be used to query the timestamp and filename of the last download.

Parameters:

[out] IsProgrammed flag wether download information is valid
[out] timestamp timestamp of last download
[out] filename filename of the downlaoded waveform

```
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.

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.

virtual TriggerSourceEnumNet GetTriggerSource (unsigned int *triggernum*) [virtual]
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.

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.

virtual void ListModeSendStart (unsigned int *SidebandMask*) [virtual]
virtual void ListModeSendStop (unsigned int *SidebandMask*) [virtual]
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.

void SendStop (uint32_t *triggermap*)

Stop some or all triggers of the STG.

Parameters:

triggermap A bitmap of triggers which will be stopped.

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..

virtual void SetAutocalibrationDisabled (unsigned int *channel*,
bool *enable*
) [virtual]

virtual void SetBlankingEnable (unsigned int *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.

virtual void SetBlankingEnable (unsigned int *electrode*,
array< bool >^ *enable*
) [virtual]

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

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

Parameters:

channel The channel to change.

virtual void SetCurrentMode ( ) [virtual]

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

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.

```
virtual void SetElectrodeDacMux ( uint32_t              electrode,
                                  uint32_t              index,
                                  array< uint32_t >^    dac
                                )                       [virtual]
```

Defines the DAC to use for an electrode.

Parameters:

electrode The electrode number.

dac The DAC to use, can be 1, 2 or 3. To ground an electrode, use 0.

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

Enables or disables the stimulation switch for an electrode.

Parameters:

electrode The electrode number.

index The index for listmode.

enable 1 to enable the electrode, 0 to disable.

```
virtual void SetElectrodeEnable ( uint32_t  electrode,
                                  uint32_t  index,
                                  bool      enable
                                )           [virtual]
virtual void SetElectrodeMode ( uint32_t                          electrode,
                                array< ElectrodeModeEnumNet >^    mode
                              )                                   [virtual]
```

Puts an electrode in either automatic or manual mode.

Parameters:

electrode The electrode number.

mode 0 for automatic and 3 for manual mode.

```
virtual void SetElectrodeMode ( uint32_t              electrode,
                                ElectrodeModeEnumNet  mode
                              )                       [virtual]
virtual void SetEnableAmplifierProtectionSwitch ( unsigned int  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.

```
virtual void SetEnableAmplifierProtectionSwitch ( unsigned int      electrode,
                                                  array< bool >^    enable
                                                )                   [virtual]
virtual void SetFAAmplification ( unsigned int  amplification ) [virtual]
virtual void SetHeadstage ( unsigned int  headstage ) [virtual]
virtual void SetListmodeIndexRange ( unsigned int  Sideband,
                                     unsigned int  StartIndex,
                                     unsigned int  EndIndex,
                                     unsigned int  Mode
                                   )               [virtual]
virtual void SetListmodeTriggerSource ( unsigned int              Sideband,
                                        TriggerSourceEnumNet      Triggersource,
                                        int                       bitnum_offset
                                      )                           [virtual]
virtual void SetListmodeTriggerSource ( unsigned int              Sideband,
                                        TriggerSourceEnumNet      Triggersource
                                      )                           [virtual]
virtual void SetMeasurementMode ( unsigned int  channel ) [virtual]
```

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

Parameters:

channel The channel to change.

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.

```
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. It has no effect on the output of the waveform.

Parameters:

[in] timestamp timestamp of download

[in] filename filename of the downloaded waveform.

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

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

Parameters:

channel The channel to change.

virtual void SetVoltageMode ( ) [virtual]

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

Generated on Fri Jan 9 2015 13:58:50 for McsUsbNet.dll for STG by dox *1.7.6.1*

- Main Page
- Namespaces

- Classes

- Class List
- Class Hierarchy
- Class Members

- Mcs
- Usb
- CStg200xDownloadBasicNet

Public Member Functions | Properties

CStg200xDownloadBasicNet Class Reference

Base class for the STG200x series download mode. More...

Inheritance diagram for CStg200xDownloadBasicNet:

List of all members.

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.

void GetTrigger (array< uint32_t >^%channelmap, array< uint32_t >^%syncoutmap, array< uint32_t >^%repeat)

Queries the trigger settings for the STG. Note that all memory segments have their own trigger setting.

void GetSweepCount (array< uint32_t >^%sweeps, array< uint32_t >^%triggers)

Get the sweep and trigger count of the STG.

void ForceStatusEvent ()

Force a status event.

void ResetStatus (uint32_t triggermap)

Reset the status flag.

void SetCapacity (array< uint32_t >^channelCapacity, array< uint32_t >^syncCapacity)

Configures the memory layout of the current segment in download mode.

void GetCapacity ([Out] array< uint32_t >^%channelCapacity,[Out] array< uint32_t >^%syncCapacity)

Queries the memory layout of the current segment in download mode.

virtual void ClearSyncData (uint32_t channel)

Delete a SyncOut pattern from STG memory.

virtual void SendSyncData (uint32_t channel, array< WORD >^pData, array< uint64_t >^tData)

Uploads sync output data to the STG.

virtual void ClearChannelData (uint32_t channel)

Delete a Stimulus Pattern from STG memory.

virtual void SendChannelData (uint32_t channel, array< WORD >^pData, array< uint64_t >^tData)

Uploads analog data (stimulus patterns) to the STG.

virtual void EnableAutoReset ()

Enable AutoReset of the STG Status.

virtual void DisableAutoReset ()

Disable AutoReset of the STG Status.

virtual void SetupRetriggerMode (int8_t trigger, RetriggerActionEnumNet same_trigger, RetriggerActionEnumNet other_trigger)

Define the action on triggers while the STG is running.

virtual void SetupRetriggerMode (RetriggerActionEnumNet same_trigger, RetriggerActionEnumNet other_trigger)

Define the action on triggers while the STG is running.

Properties

CStimulusFunctionNet^

Stimulus [get]

Detailed Description

Base class for the STG200x series download mode.

Member Function Documentation

virtual void ClearChannelData (uint32_t *channel*) [virtual]

Delete a Stimulus Pattern from STG memory.

Parameters:

[in] channel specifies the channel to clear.

virtual void ClearSyncData (uint32_t *channel*) [virtual]

Delete a SyncOut pattern from STG memory.

Parameters:

[in] channel specifies the syncout channel to clear.

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()

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.

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 GetCapacity ([Out] array< uint32_t >^% *channelCapacity*,
[Out] array< uint32_t >^% *syncCapacity*
)

Queries the memory layout of the current segment in download mode.

For each segment, the memory layout has to be defined. Each channel and sync output can be given an individual amount of memory space as needed by the application.

Parameters:

[in] channelCapacity is a list of memeory sizes, with one entry per channel
[in] syncCapacity is a list of memeory sizes, with one entry per syncout

void GetSweepCount (array< uint32_t >^% *sweeps*,
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:

[out] sweeps on return contains the number of sweeps for each trigger

[out] triggers on return contains the number of trigger events seen for each trigger

void GetTrigger (array< uint32_t >^% *channelmap*,
array< uint32_t >^% *syncoutmap*,
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.
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.

void ResetStatus (uint32_t *triggermap*)

Reset the status flag.

Parameters:

[in] triggermap bitmap of trigger for which to reset the status

virtual void SendChannelData (uint32_t *channel*,
array< WORD >^ *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. If your application cannot handle 64 bit integers, use the STG200x_SendChannelData32() call instead.

Parameters:

[in] channel specifies the channel to append the data to.
[in] pData a list of datapoints
[in] tData a list of durations as int64_t. The time is given in units of µs.

virtual void SendSyncData (uint32_t *channel*,
array< WORD >^ *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:

[in] channel specifies the sync output channel to append the data to
[in] pData a list of datapoints
[in] tData a list of durations as int64_t. The time is given in units of µs.

```
void SetCapacity ( array< uint32_t >^  channelCapacity,
                   array< uint32_t >^  syncCapacity
                 )
```

Configures the memory layout of the current segment in download mode.

For each segment, the memory layout has to be defined. Each channel and sync output can be given an individual amount of memory space as needed by the application. Make sure the sum does not exceed the memory which is assigned to the currently selected segement.

Parameters:

[in] channelCapacity is a list of memeory sizes, with one entry per channel
[in] syncCapacity is a list of memeory sizes, with one entry per syncout

```
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:

[in] trigger	The trigger to change.
[in] same_trigger	Action for successive triggers in Normal Mode, and for triggers to the currently selected segment in Multi-File Mode.
[in] other_trigger	Action for successive triggers in Multi-File Mode for a trigger on a segment not currently selected. Not used in Normal Mode.

```
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:

[in] same_trigger	Action for successive triggers in Normal Mode, and for triggers to the currently selected segment in Multi-File Mode.
[in] other_trigger	Action for successive triggers in Multi-File Mode for a trigger on a segment not currently selected. Not used in Normal Mode.

```
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.
channelmap	For each trigger, a bitmap of channels that belong to this trigger.
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.

Property Documentation

CStimulusFunctionNet^ Stimulus `[get]`

Generated on Fri Jan 9 2015 13:58:50 for McsUsbNet.dll for STG by dox *1.7.6.1*

- Main Page
- Namespaces
- Classes

- Class List
- Class Hierarchy
- Class Members

- Mcs
- Usb
- CStg200xDownloadNet

Public Member Functions | Events

CStg200xDownloadNet Class Reference

Main class for the STG download mode. More...

Inheritance diagram for CStg200xDownloadNet:

List of all members.

Public Member Functions

	CStg200xDownloadNet ()
	CStg200xDownloadNet (OnStg200xPollStatus^ 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 dest_type) Prepare and send 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, uint32_t segmentflags) Switchs segment and starts trigger.
void	SendSegmentSelect (uint32_t segment, uint32_t 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.
uint32_t	QueryTriggerstatus ()

void	SetOutputMap (uint32_t ChannelLayout[])
int32_t	GetModuleTemp (unsigned int channel)
uint32_t	GetModuleCurrent (unsigned int channel)

Events

OnStg200xPollStatus^	Stg200xPollStatusEvent
OnMwPollStatus^	MwPollStatusEvent

Detailed Description

Main class for the STG download mode.

This class implements the STG download mode interface

Constructor & Destructor Documentation

CStg200xDownloadNet ( )
CStg200xDownloadNet (OnStg200xPollStatus^ *pollStatus*)

Use this constructor if you want to use the status callback.

~CStg200xDownloadNet ( )

Member Function Documentation

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.

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 four segments (number 0 to 3) 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.

```
uint32_t GetModuleCurrent ( unsigned int  channel  )
int32_t GetModuleTemp ( unsigned int  channel  )
void PrepareAndSendData ( uint32_t                    channel,
                          array< int32_t >^           Amplitude,
                          array< uint64_t >^          Duration,
                          STG_DestinationEnumNet      dest_type
                        )
```

Prepare and send 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.

Parameters:

channel	The channel number to send data to.
Amplitude	A list of datapoints as int32.
Duration	A list of durations as uint64. The time is given in units of µs.
dest_type	specifies wheather the data is for syncout, current or voltage stimulation.

```
uint32_t QueryTriggerstatus (  )
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).

```
void SendSegmentSelect ( uint32_t  segment,
                         uint32_t  segmentflags
                       )
```

Switchs segment.

Parameters:

segment	The number of the segment to select.
segmentflags	A bitmap of flags, bit 1: assign all channels to the trigger number equal to the segment.

void SendSegmentStart (uint32_t *triggermap*,
uint32_t *segment*,
uint32_t *segmentflags*
)

Switchs segment and starts trigger.

Parameters:

triggermap	A bitmap of triggers that will be started.
segment	The number of the segment to select.
segmentflags	A bitmap of flags, bit 1: assign all channels to the trigger number equal to the segment.

void SetOutputMap (uint32_t *ChannelLayout*[])

Event Documentation

OnMwPollStatus^ MwPollStatusEvent

OnStg200xPollStatus^ Stg200xPollStatusEvent

Generated on Fri Jan 9 2015 13:58:50 for McsUsbNet.dll for STG by dox *1.7.6.1*

- Main Page
- Namespaces
- Classes

- Class List
- Class Hierarchy
- Class Members

- Mcs
- Usb
- CStg200xStreamingNet

Public Member Functions | Events
CStg200xStreamingNet Class Reference

Main class for the STG streaming mode. More...

Inheritance diagram for CStg200xStreamingNet:

List of all members.

Public Member Functions

	CStg200xStreamingNet (uint32_t ringbuffer_size) Constructor.
	CStg200xStreamingNet (uint32_t ringbuffer_size, OnStg200xDataHandler^ dataHandler, OnStg200xErrorHandler^ errorHandler) Constructor.
	~CStg200xStreamingNet () Destructor.
void	EnableContinousMode () Enable the continous mode of the STG.
void	DisableContinousMode () Disable the continous mode of the STG.
uint32_t	GetFramesDone () Queries the number of frames sent to the STG.
uint32_t	GetFramesBuffered (uint32_t trigger) Queries the number of frames currently buffered in STG Memory.
uint32_t	GetCurrentRate (uint32_t trigger) Queries the rate at which frames are currently sent to the STG.
void	SetupTrigger (array< uint32_t >^channelmap, array< uint32_t >^syncoutmap, array< uint32_t >^digoutmap, array< uint32_t >^autostart, array< uint32_t >^callback_threshold) Configures the trigger settings for the STG.
void	GetTrigger ([Out]array< uint32_t >^%channelmap,[Out]array< uint32_t >^%syncoutmap,[Out]array< uint32_t >^%digoutmap,[Out]array< uint32_t >^%autostart) Queries the trigger settings for the STG.
void	SetCapacity (array< uint32_t >^dwTriggerCapacity) Configures the memory layout for the streaming mode of the STG.
void	GetCapacity ([Out]array< uint32_t >^%dwTriggerCapacity) Queries the memory layout for the streaming mode of the STG.
uint32_t	EnqueueData (uint32_t channel, array< short >^data) Sends data to the STG for a given channel.
uint32_t	GetDataQueueSpace (uint32_t channel) Queries the space availabe in the PC memory for a given data channel.
uint32_t	EnqueueSyncout (uint32_t channel, array< WORD >^data) Sends syncout data to the STG for a given channel.
uint32_t	GetSyncoutQueueSpace (uint32_t channel)

Queries the space availabe in the PC memory for a given syncout channel.

void StartLoop ()
Starts the streaming mode.

void StopLoop ()
Stops the streaming mode.

Events

OnStg200xDataHandler^ OnStg200xDataHandlerEvent

OnStg200xErrorHandler^ OnStg200xErrorHandlerEvent

Detailed Description

Main class for the STG streaming mode.

This class implements the STG streaming mode interface

Constructor & Destructor Documentation

CStg200xStreamingNet (uint32_t *ringbuffer_size*)

Constructor.

Parameters:

ringbuffer_size The ringbuffer_size size of the ringbuffer in PC memory.

CStg200xStreamingNet (uint32_t *ringbuffer_size*,
OnStg200xDataHandler^ *dataHandler*,
OnStg200xErrorHandler^ *errorHandler*
)

Constructor.

Parameters:

ringbuffer_size The ringbuffer_size size of the ringbuffer in PC memory.
dataHandler The data Handler callback.
errorHandler The error Handler callback.

~CStg200xStreamingNet ( )

Destructor.

Member Function Documentation

void DisableContinousMode ( )

Disable the continous mode of the STG.

Defines how the STG handles buffer underruns. If continous mode is switched off, the triggers are stopped automatically when a buffer runs empty.

void EnableContinousMode ( )

Enable the continous mode of the STG.

Defines how the STG handles buffer underruns. buffer runs empty. If continous mode is switched off, the triggers are stopped automatically when a buffer runs empty.

```
uint32_t EnqueueData ( uint32_t          channel,
                       array< short >^   data
                     )
```

Sends data to the STG for a given channel.

Parameters:

channel The channel number to send data to.

data A pointer to the data.

Returns:

Returns the number of enqueued bytes.

```
uint32_t EnqueueSyncout ( uint32_t          channel,
                          array< WORD >^    data
                        )
```

Sends syncout data to the STG for a given channel.

Parameters:

channel The channel number to send data to.

data A pointer to the data.

Returns:

Returns 0 on success.

void GetCapacity ([Out] array< uint32_t >^% *dwTriggerCapacity*)

Queries the memory layout for the streaming mode of the STG.

In streaming mode, each trigger can be given an individual amount of memory space as needed by the application.

Parameters:

dwTriggerCapacity A list of memory sizes in bytes, one with entry for each trigger.

uint32_t GetCurrentRate (uint32_t *trigger*)

Queries the rate at which frames are currently sent to the STG.

Parameters:

trigger The trigger number to query.

Returns:

Returns the rate at which frames are sent in frames 1/8 ms times 2^14.

uint32_t GetDataQueueSpace (uint32_t *channel*)

Queries the space availabe in the PC memory for a given data channel.

Parameters:

channel The channel number to query.

Returns:

Returns number of data points for which there is space.

uint32_t GetFramesBuffered (uint32_t *trigger*)

Queries the number of frames currently buffered in STG Memory.

Parameters:

trigger The trigger number to query.

Returns:

Returns the number of frames currently buffered in STG memory.

uint32_t GetFramesDone ( )

Queries the number of frames sent to the STG.

Returns:

Returns the number of frames already sent to the STG.

uint32_t GetSyncoutQueueSpace (uint32_t *channel*)

Queries the space availabe in the PC memory for a given syncout channel.

Parameters:

channel The channel number to query.

Returns:

Returns number of data points for which there is space.

```
void GetTrigger ( [Out] array< uint32_t >^% channelmap,
                  [Out] array< uint32_t >^% syncoutmap,
                  [Out] array< uint32_t >^% digoutmap,
                  [Out] array< uint32_t >^% autostart
                )
```

Queries the trigger settings for the STG.

Parameters:

channelmap For each trigger, a bitmap of channels which belong to this trigger.
syncoutmap For each trigger, a bitmap of syncout which belong to this trigger.
digoutmap For each trigger, a bitmap of digout which belong to this trigger.
autostart For each trigger, define whether this trigger should autostart.

```
void SetCapacity ( array< uint32_t >^ dwTriggerCapacity )
```

Configures the memory layout for the streaming mode of the STG.

In streaming mode, each trigger can be given an individual amount of memory space as needed by the application. Make sure the sum does not exceed the total memory available.

Parameters:

dwTriggerCapacity A list of memory sizes in bytes, one with entry for each trigger.

```
void SetupTrigger ( array< uint32_t >^ channelmap,
                    array< uint32_t >^ syncoutmap,
                    array< uint32_t >^ digoutmap,
                    array< uint32_t >^ autostart,
                    array< uint32_t >^ callback_threshold
                  )
```

Configures the trigger settings for the STG.

Parameters:

channelmap For each trigger, a bitmap of channels which belong to this trigger.
syncoutmap For each trigger, a bitmap of syncout which belong to this trigger
digoutmap For each trigger, a bitmap of digout which belong to this trigger.

autostart	For each trigger, define whether this trigger should autostart.
callback_threshold	The callback_threshold for each trigger, when the data handler should be called in precent of the buffer level.

void StartLoop ( )

Starts the streaming mode.

void StopLoop ( )

Stops the streaming mode.

Event Documentation

OnStg200xDataHandler^ OnStg200xDataHandlerEvent
OnStg200xErrorHandler^ OnStg200xErrorHandlerEvent

Generated on Fri Jan 9 2015 13:58:50 for McsUsbNet.dll for STG by dox *1.7.6.1*

- Main Page
- Namespaces
- Classes

- Namespace List
- Namespace Members

- All
- Functions
- Enumerations
- Enumerator

Here is a list of all namespace members with links to the namespace documentation for each member:

- DEVICE_NOT_FOUND : Mcs::Usb
- EnSTG200x_STATUS : Mcs::Usb
- EnSTG200x_TRIGGER_STATUS : Mcs::Usb
- NOT_CONNECTED : Mcs::Usb
- OK : Mcs::Usb
- OnDeviceArrivalRemoval() : Mcs::Usb
- OnMwPollStatus() : Mcs::Usb
- OnStg200xDataHandler() : Mcs::Usb
- OnStg200xErrorHandler() : Mcs::Usb
- OnStg200xPollStatus() : Mcs::Usb
- STG200x_TRIGGER_FINISHED : Mcs::Usb
- STG200x_TRIGGER_IDLE : Mcs::Usb

- STG200x_TRIGGER_RUNNING : Mcs::Usb

Generated on Fri Jan 9 2015 13:58:50 for McsUsbNet.dll for STG by dox *1.7.6.1*

- Main Page
- Namespaces
- Classes

- Namespace List
- Namespace Members

- All
- Functions
- Enumerations
- Enumerator

- OnDeviceArrivalRemoval() : Mcs::Usb
- OnMwPollStatus() : Mcs::Usb
- OnStg200xDataHandler() : Mcs::Usb
- OnStg200xErrorHandler() : Mcs::Usb
- OnStg200xPollStatus() : Mcs::Usb

Generated on Fri Jan 9 2015 13:58:50 for McsUsbNet.dll for STG by dox *1.7.6.1*

- Main Page
- Namespaces
- Classes

- Namespace List
- Namespace Members

- All
- Functions
- Enumerations
- Enumerator

- EnSTG200x_STATUS : Mcs::Usb
- EnSTG200x_TRIGGER_STATUS : Mcs::Usb

Generated on Fri Jan 9 2015 13:58:50 for McsUsbNet.dll for STG by dox *1.7.6.1*

- DEVICE_NOT_FOUND : Mcs::Usb
- NOT_CONNECTED : Mcs::Usb
- OK : Mcs::Usb
- STG200x_TRIGGER_FINISHED : Mcs::Usb
- STG200x_TRIGGER_IDLE : Mcs::Usb
- STG200x_TRIGGER_RUNNING : Mcs::Usb
