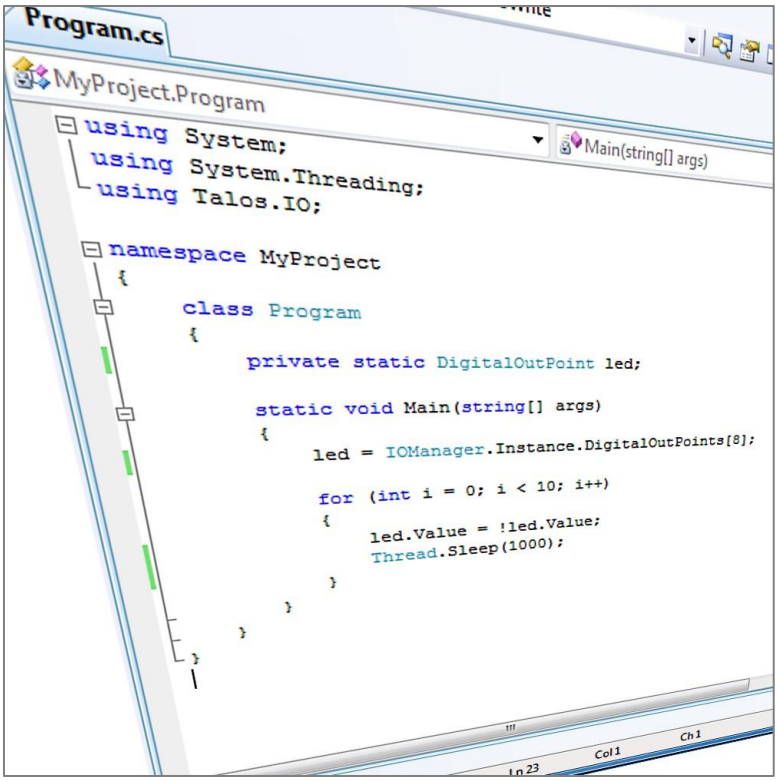
Talos™ I/O Framework Documentation

USER MANUAL



How to Get Assistance

When calling for technical assistance, please have the device installed and ready to run diagnostics. If possible, have your hardware manual and current settings ready.

The Sealevel website is an excellent resource located at www.sealevel.com. The most current software updates and user manuals are available via our homepage by clicking on the 'Product Manuals' or 'Software Drivers' links located under ‘Support’. Manuals and software can also be downloaded from the product page for your device.

The support section of our website provides answers for many common questions. Refer to this helpful resource by visiting http://www.sealevel.com/support/.

TECHNICAL SUPPORT

Monday – Friday
8:00 am to 5:00 pm EST
Phone: +1 (864) 843-4343
Email: support@sealevel.com

ISO 9001:2000

SL9210 Revision 10/2009
As further evidence to Sealevel’s commitment to Total Customer Satisfaction the Company’s Management System has achieved registration to ISO 9001:2000 in 2002. This provides one of the strongest assurances of product/service quality available. ISO 9001:2000 registration ensures Sealevel’s customers that the proper processes and business practices are in place to guarantee their satisfaction. Sealevel is audited by Quality Management Institute (QMI), North America’s largest management system registrar.

TRADEMARKS

Sealevel Systems, Incorporated acknowledges that all trademarks referenced in this manual are the service mark, trademark, or registered trademark of the respective company.

Contents

Chapter 1

Talos^{TM} Framework Documentation

1.1 License Agreement

This software is licensed solely for use with a Sealevel Systems, Incorporated product. The user may operate the software only when utilizing the affiliated Sealevel Systems, Incorporated product.

This license is in effect until terminated by Sealevel Systems, Incorporated or the user. The user may terminate the license only if the provided software, any copies, modifications and enhancements are returned or destroyed.

In addition to all remedies, which Sealevel Systems, Incorporated may have legal and equitable, if user should breach or threaten to breach, the provisions of this license, Sealevel Systems, Incorporated may individually and without notice, terminate the license granted here under. At such time all copies, modifications and enhancements must be returned or destroyed.

By installing the software provided, the user acknowledges and agrees to all the terms and conditions of this License Agreement.

1.2 Introduction

Sealevels Talos Framework provides numerous extensions to the Microsoft .NET Compact Framework. These extensions help to ease application development on supported Sealevel products. In addition to adding an easy-to-use, object-orient interface for access to Digital and Analog IO, the Talos Framework adds some features of the full .NET Framework that are not available in the compact edition.

- **Talos**

  Extensions to the Microsoft .NET Compact Framework in numerous areas

- **Talos::IO**

  Objects useful for reading and writing files and physical hardware

- **Talos::Protocols**

  Objects useful for communicating with remote devices

1.3 Getting Started

The Sealevel Talos Framework has been designed to be simple to use. The following code snippet demonstrates the use of the IO interface for manipulation of Digital Outputs on supported Sealevel hardware.

```
using System;
using Talos.IO;

namespace DigitalOutputExample
{
  class Program
  {
    static void Main(string[] args)
    {
      // Get the current instance of the IOManager
      IOManager manager = IOManager.Instance;

      // Display the count of the DigitalOutPoints
      Console.WriteLine("DigitalOutPoint Count: {0}", manager.DigitalOutPoints.Co
      unt);

      // Cycle through each point individually
      foreach (DigitalOutPoint point in manager.DigitalOutPoints)
      {
        // Display the initial state of the output
        Console.WriteLine("DigitalOutPoint {0}: {1} : {2}", point.Index, point.Va
      lue, point.Description);

        // Toggle the output
        point.Value = !point.Value;

        // Display the final state of the output
        Console.WriteLine("DigitalOutPoint {0}: {1} : {2}", point.Index, point.Va
      lue, point.Description);
      }
    }
  }
}
```

1.4 Warranty

Sealevel Systems, Incorporated makes no implied or express warranty of any kind with regard to this software or the documentation that is included. Sealevel Systems, Incorporated will not be liable for damages of any type resulting from the performance, use or furnishing of any of the supplied software. Any liability of Sealevel Systems, Incorporated will be limited to a refund of the purchase price or product replacement.

1.5 Copyright

This software is protected by United States copyright law, and international treaty provisions. User acknowledges that no title to the intellectual property in the software is transferred to user. User further acknowledges that title and full ownership rights to the software will remain the exclusive property of Sealevel Systems, Incorporated, and user will not acquire any ownership rights to the software. User agrees that any copies of the software will contain the same proprietary notices, which appear on and in the software.

1.6 Questions & Comments

Send your questions and comments to Sealevel Systems. For technical assistance, please include your model or part number and any device settings that may apply. Technical support is available Monday to Friday from 8:00AM to 5:00PM (US Eastern Time Zone, UTC-5 hours) by email (support@sealevel.com) or by phone at +1 (864) 843.4343.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 3

Class Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

Namespace Documentation

5.1 Talos Namespace Reference

Extensions to the Microsoft .NET Compact Framework in numerous areas.

Namespaces

- namespace Collections

  Objects representing collections and useful for manipulating collections.

- namespace IO

  Objects useful for reading and writing files and physcial hardware.

- namespace Protocols

  Objects useful for communicating with remote devices.

- namespace Reflection

  Objects useful for acquiring information about .NET assemblies.

- namespace Threading

  This is the Threading namespace of the Talos Framework.

Classes

- struct OwnerInformation

  This structure is used to store Windows CE device ownership information.

- class Environment

  This class provides version information about the current runtime environment. This class mimics some of the behavior of System.Environment, but extends some of the functionality to allow access to Sealevel Systems specific information.

- class Component

  A container of version information of a component of the Talos framework.

5.1.1 Detailed Description

Extensions to the Microsoft .NET Compact Framework in numerous areas. Sealevels Talos Framework provides numerous extensions to the Microsoft .NET Compact Framework. These extensions help to ease application development on supported Sealevel products. In addition to adding an easy-to-use, object-orient interface for access to Digital and Analog IO, the Talos Framework adds some features of the full .NET Framework that are not available in the compact edition.

5.2 Talos::Collections Namespace Reference

Objects representing collections and useful for manipulating collections.

Classes

- class NumericComparer

  Simple class for easy numeric string comparison.

5.2.1 Detailed Description

Objects representing collections and useful for manipulating collections. This portion of the Talos Framework contains collections and tools helpful for working with collections. Anything that isn't already available in the .NET Compact Framework that makes data storing and manipulation easier, will be found here.

5.3 Talos::IO Namespace Reference

Objects useful for reading and writing files and physcial hardware.

Classes

- class DeviceManager

  A manager of all devices that Talos is going to be dealing with. This is where points are actually instantiated.

- class IOManager

  The IOManager class handles all the IO available on the system.

- class SerialPort

  SerialPort allows for the interaction with COM devices.

- class FileStream

  Exposes a Stream around a file.

- class AnalogInPoint

  Base class for a AnalogInPoint.

- class AnalogOutPoint

  Base class for a AnalogOutPoint.

- class AnalogPoint

  Base class for a AnalogPoint.

- class CanPoint

  Base class for CanPoint.

- class CanMailbox

  Data structure for configuring a mailbox for receiving or sending messages.

- class Counter

  Base class for Counter.

- class DigitalInPoint

  Base class for DigitalInPoint.

- class DigitalOutPoint

  Base class for a DigitalOutPoint.

- class DigitalPoint

  Base class for a DigitalPoint.

- class Point

  Base class for a Point.

Enumerations

- enum Parity {
  None, Odd, Even, Mark,
  Space }
  The Parity enumeration for the SerialPort class.
- enum StopBits { One, OnePointFive, Two }
  The StopBits enumeration for the SerialPort class.
- enum DtrFlowControl { Disabled, Enabled, Handshake }
  The DtrFlowControl enumeration for the SerialPort class.
- enum RtsFlowControl { Disabled, Enabled, Handshake, Toggle }
  The RtrFlowControl enumeration for the SerialPort class.
- enum SerialError {
  Frame = 8, Overrun = 2, RxOver = 1, RxParity = 4,
  TxFull = 0x100 }
  The SerialError enumeration for the SerialPort class.
- enum CanMailboxMode { Disabled = 0, Receive = 1, Send = 3 }
  Mailbox modes for configuring mailboxes.
- enum Direction { Invalid, Input, Output }
  Direction of an IO Point.
- enum IOType { Unknown, Digital, Analog, Counter }
  Type of an IO Point.
- enum Polarity { Invalid, ActiveHigh, ActiveLow }
  Polarity of an IO point.
- enum Isolation { Invalid, Standard, Inverted }
  Isolation of an IO Point.
- enum Persistency { Invalid, Persistent, NonPersistent }
  Not implemented.
- enum DigitalDefault { Invalid, Set, Cleared }
  Not implemented.
- enum AnalogUnit {
  Invalid, MilliVolts, Volts, MilliAmps,
  Amps, Raw }
  AnalogUnit used in conjunction with AnalogPoint floating-point values.
- enum CounterMode { Free, Single, Range, Modulo }
  Counter can be configured to operate in a specified mode. The mode affects the counter value returned.

- enum CounterIndexMode { Data, Reset, Instant, None }

  This mode can be configured to perform different operations on the index.

- enum CounterIndexType

  This mode can be configured to assign the proper polarity of the index input signal.

- enum CounterWidth

  The counter can be configured to operate on n-byte register boundaries. This width also applies to instruction/data registers.

- enum CounterFlag { Index, Compare, Under, Over }

  Setting the flag essentially tells the quadrature counter which interrupt to properly report on.

- enum CounterUnits {

  RevolutionsPerHour, RevolutionsPerMinute, RevolutionsPerSecond, CountsPerMinute,

  CountsPerSecond, Counts }

  This value is used in ModCounter conversion formulas to determine the correct units for velocity or counts.

5.3.1 Detailed Description

Objects useful for reading and writing files and physcial hardware. This portion of the Talos Framework contains streams, classes, and methods helpful for reading and writing files and physcial hardware. Some classes in this namespace replicate pre-existing classes. In these cases, additional functionality has been added that is not available in the .NET Compact Framework.

5.3.2 Enumeration Type Documentation

5.3.2.1 enum Talos::IO::Parity

The Parity enumeration for the SerialPort class. Use this enumeration when setting the Parity property for a serial port connection. Parity is an error-checking procedure in which the number of 1s must always be the same, either even or odd, for each group of bits that is transmitted without error. In modem-to-modem communications, parity is often one of the parameters that must be agreed upon by sending parties and receiving parties before transmission can take place.

Enumerator:

None Option for no parity.

Odd Option for odd parity.

Even Option for even parity.

Mark Option for mark parity.

Space Option for space parity.

5.3.2.2 enum Talos::IO::StopBits

The StopBits enumeration for the SerialPort class. This enumeration specifies the number of stop bits to use. Stop bits separate each unit of data on an asynchronous serial connection. They are also sent continuously when no data is available for transmission.

Enumerator:

One Option for one stop bit.

OnePointFive Option for one and a half stop bits.

Two Option for two stop bits.

5.3.2.3 enum Talos::IO::DtrFlowControl

The DtrFlowControl enumeration for the SerialPort class.

Enumerator:

Disabled Option for no DTR flow control.

Enabled Option for enabling DTR flow control.

Handshake Option for Handshake control using DTR.

5.3.2.4 enum Talos::IO::RtsFlowControl

The RtrFlowControl enumeration for the SerialPort class.

Enumerator:

Disabled Option for no RTS flow control.

Enabled Option for enabling RTS flow control.

Handshake Option for Handshake control using RTS.

Toggle Option for Toggle control using RTS.

5.3.2.5 enum Talos::IO::SerialError

The SerialError enumeration for the SerialPort class. We should consider creating a serial events enum. Some items in the list don't belong.

Enumerator:

Frame Frame Error.

Overrun Overrun Error.

RxOver Receive Completed.

RxParity Received Parity Error.

TxFull Transmitter is full.

5.3.2.6 enum Talos::IO::CanMailboxMode

Mailbox modes for configuring mailboxes.

Enumerator:

Disabled Disables the mailbox.

Receive The first message received is stored in mailbox data registers. Data remain available until the next transfer request.

Send The last message received is stored in mailbox data register. The next message always overwrites the previous one. The application has to check whether a new message has not overwritten the current one while reading the data registers. The message stored in the mailbox data registers will try to win the bus arbitration immediately or later according to or not the Time Management Unit configuration.

5.3.2.7 enum Talos::IO::Direction

Direction of an IO Point. This tri-state value indicates the directionality of a particular IO Point.

Enumerator:

Invalid Invalid configuration.

Input Input.

Output Output.

5.3.2.8 enum Talos::IO::IOType

Type of an IO Point. This tri-state value indicates the type of a particular IO Point.

Enumerator:

Unknown Unknown IO type.

Digital Digital IO type.

Analog Analog IO type.

Counter Counter IO type.

5.3.2.9 enum Talos::IO::Polarity

Polarity of an IO point. Polarity is a user-configurable access modifier that is used to map hardware values to values to end-user digital states.

Enumerator:

Invalid Invalid state.

ActiveHigh High indicates the active state.

ActiveLow Low indicates the active state.

5.3.2.10 enum Talos::IO::Isolation

Isolation of an IO Point. Isolation describes a further configuration option for the mapping of digital output states to hardware values.

Enumerator:

Invalid Invalid value.

Standard Returns the current value.

Inverted Returns the current value inverted.

5.3.2.11 enum Talos::IO::Persistency

Not implemented. Not implemented. Persistency is an optional output point modifier used to enable a default power-up state for each point.

Enumerator:

Invalid Invalid value.

Persistent This value instructs the point to persist.

NonPersistent This value instructs the point to not persist.

5.3.2.12 enum Talos::IO::DigitalDefault

Not implemented. Not implemented. Default state describes the default power-up state of a digital output point when used in conjunction with Persistency.

Enumerator:

Invalid Invalid default.

Set Default state is true.

Cleared Default state is false.

5.3.2.13 enum Talos::IO::AnalogUnit

AnalogUnit used in conjunction with AnalogPoint floating-point values. This value is used in Analog to Digital conversion formulas to determine the floating point number to return to users.

Enumerator:

Invalid Invalid analog unit.

MilliVolts Analog unit of mV.

Volts Analog unit of V.

MilliAmps Analog unit of mA.

Amps Analog unit of A.

Raw Analog unit of Raw.

5.3.2.14 enum Talos::IO::CounterMode

Counter can be configured to operate in a specified mode. The mode affects the counter value returned.

Enumerator:

Free Ignore LFLAG.

Single BW and CY only.

Range Limit between 0 and DTR. Dir must reverse to resume counting.

Modulo where modulo value 'n' is stored in DTR

5.3.2.15 enum Talos::IO::CounterIndexMode

This mode can be configured to perform different operations on the index.

Enumerator:

Data DTR - CNTR.

Reset Clear CNTR.

Instant CNTR - OTR.

None ignore

5.3.2.16 enum Talos::IO::CounterIndexType

This mode can be configured to assign the proper polarity of the index input signal.

5.3.2.17 enum Talos::IO::CounterWidth

The counter can be configured to operate on n-byte register boundaries. This width also applies to instruction/data registers.

5.3.2.18 enum Talos::IO::CounterFlag

Setting the flag essentially tells the quadrature counter which interrupt to properly report on.

Enumerator:

Index At each index (I).

Compare When CNTR == DTR.

Under Transition 0 -> -1.

Over Transition 2^n -> 0.

5.3.2.19 enum Talos::IO::CounterUnits

This value is used in ModCounter conversion formulas to determine the correct units for velocity or counts.

Enumerator:

RevolutionsPerHour Revolutions per hour.

RevolutionsPerMinute Revolutions per minute.

RevolutionsPerSecond Revolutions per sec.

CountsPerMinute Counts per minute.

CountsPerSecond Counts per second.

Counts Counts.

5.4 Talos::Protocols Namespace Reference

Objects useful for communicating with remote devices.

Classes

- class ModbusClient
  A simple implementation of a basic Modbus client.
- class ModbusException
  Represents errors that occur during Modbus communications.

Enumerations

- enum InterfaceType { Invalid, Serial, Tcp }
  Type indicative of the Communications Interface used for a ModbusClient class.
- enum IdentificationType { Vendor, Product, Version }
  Type indicating which identification string is desired from a ReadIdentification command.

5.4.1 Detailed Description

Objects useful for communicating with remote devices. This portion of the Talos Framework contains classes and methods helpful for communicating with remote devices, such as Sealevel Modbus enabled distributed IO modules.

5.4.2 Enumeration Type Documentation

5.4.2.1 enum Talos::Protocols::InterfaceType

Type indicative of the Communications Interface used for a ModbusClient class.

Enumerator:

Invalid Not a valid interface value.
Serial Serial communication via a COM port.
Tcp Ethernet TCP Socket communications.

5.4.2.2 enum Talos::Protocols::IdentificationType

Type indicating which identification string is desired from a ReadIdentification command.

Enumerator:

Vendor Vendor name.
Product Product name.
Version Version Number.

5.5 Talos::Reflection Namespace Reference

Objects useful for acquiring information about .NET assemblies.

Classes

- class AssemblyInformation

  Allows for easier access to assembly information.

5.5.1 Detailed Description

Objects useful for acquiring information about .NET assemblies. This portion of the Talos Framework contains classes and methdos helpful for acquiring information about .NET assemblies. This namespace works in tandem with the built-in System.Reflection namespace to provide useful information about assemblies.

5.6 Talos::Threading Namespace Reference

This is the Threading namespace of the Talos Framework.

Classes

- class BackgroundWorker

  The BackgroundWorker component gives you the ability to execute time-consuming operations asynchronously ("in the background"), on a thread different from your application's main UI thread.

- class DoWorkEventArgs

  Provides data for the DoWork event handler.

- class ProgressChangedEventArgs

  Provides data for the ProgressChanged event.

- class RunWorkerCompletedEventArgs

  Provides data for the MethodNameCompleted event. MethodName is a placeholder for the first part of the method's name.

- class HighperformanceCounter

  This class provides an OOP wrapper around the existing Windows CE High Performance Counter API. The properties in this static class can be used to time actions with the highest possible accuracy.

- class Watchdog

  This is a simple C# OOP wrapper around the existing Windows CE watchdog API. This class can be used to create an object that will terminate a hung application or completely restart a malfunction device.

Enumerations

- enum WatchdogAction { None = (int) Watchdog.WDOG_NO_DFLT_ACTION, KillProcess = (int) Watchdog.WDOG_KILL_PROCESS, ResetDevice = (int) Watchdog.WDOG_RESET_DEVICE }

  These are the possible actions to occur when the Watchdog is triggered.

Functions

- delegate void DoWorkEventHandler (object sender, DoWorkEventArgs e)

  Represents the method that will handle the DoWork event. This class cannot be inherited.

- delegate void ProgressChangedEventHandler (object sender, ProgressChangedEventArgs e)

  Represents the method that will handle the ProgressChanged event of the BackgroundWorker class. This class cannot be inherited.

- delegate void RunWorkerCompletedEventHandler (object sender, RunWorkerCompletedEventArgs e)

  Represents the method that will handle the RunWorkerCompleted event of a BackgroundWorker class.

5.6.1 Detailed Description

This is the Threading namespace of the Talos Framework.

5.6.2 Enumeration Type Documentation

5.6.2.1 enum Talos::Threading::WatchdogAction

These are the possible actions to occur when the Watchdog is triggered.

Enumerator:

None Perform no action.

KillProcess Kill the process which created the Watchdog.

ResetDevice Reset the device.

5.6.3 Function Documentation

5.6.3.1 delegate void Talos::Threading::DoWorkEventHandler (object *sender*, DoWorkEventArgs *e*)

Represents the method that will handle the DoWork event. This class cannot be inherited.

Parameters:

sender The source of the event.

e A DoWorkEventArgs that contains the event data.

5.6.3.2 delegate void Talos::Threading::ProgressChangedEventHandler (object *sender*, ProgressChangedEventArgs *e*)

Represents the method that will handle the ProgressChanged event of the BackgroundWorker class. This class cannot be inherited.

Parameters:

sender The source of the event.

e A ProgressChangedEventArgs that contains the event data.

5.6.3.3 delegate void Talos::Threading::RunWorkerCompletedEventHandler (object *sender*, RunWorkerCompletedEventArgs *e*)

Represents the method that will handle the RunWorkerCompleted event of a BackgroundWorker class.

Parameters:

sender The source of the event.

e A RunWorkerCompletedEventArgs that contains the event data.

Chapter 6

Class Documentation

6.1 Talos::Collections::NumericComparer Class Reference

Simple class for easy numeric string comparison.

Inherits IComparer.

Public Member Functions

- int Compare (object x, object y)

  Compares one string to another. This method provides an easy way to order strings with numeric values embedded in them logically.

6.1.1 Detailed Description

Simple class for easy numeric string comparison.

6.1.2 Member Function Documentation

6.1.2.1 int Talos::Collections::NumericComparer::Compare (object *x*, object *y*)

Compares one string to another. This method provides an easy way to order strings with numeric values embedded in them logically.

Parameters:

x The object to compare to

y The object compared to the first parameter object

Returns:

Less than zero = x is less than y, Zero = x equals y, Greater than zero = x is greater than y

6.2 Talos::Component Class Reference

A container of version information of a component of the Talos framework.

Inherits ICloneable.

Public Member Functions

- Component (string name, Version version)
  Constructor for creating a new Component class. Basically just a holder for basic runtime information.
- object Clone ()
  Implementation of ICloneable interface method.
- override string ToString ()
  String representation of this class.

Properties

- string Name `[get]`
  The name identifier associated with this object.
- Version Version `[get]`
  The Version infomration associated with this object.

6.2.1 Detailed Description

A container of version information of a component of the Talos framework.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 Talos::Component::Component (string *name*, Version *version*)

Constructor for creating a new Component class. Basically just a holder for basic runtime information.

Parameters:

name A string representation of the Component name.
version Version information of the Component image.

6.2.3 Member Function Documentation

6.2.3.1 object Talos::Component::Clone ()

Implementation of ICloneable interface method.

Returns:

A reference to a new Component class exactly like this one.

6.2.3.2 override string Talos::Component::ToString ()

String representation of this class.

Returns:

A string representation of this class.

6.2.4 Property Documentation

6.2.4.1 string Talos::Component::Name [get]

The name identifier associated with this object.

6.2.4.2 Version Talos::Component::Version [get]

The Version infomration associated with this object.

6.3 Talos::Environment Class Reference

This class provides version information about the current runtime environment. This class mimics some of the behavior of System.Environment, but extends some of the functionality to allow access to Sealevel Systems specific information.

Properties

- static OperatingSystem OSVersion `[get]`

  This property allows access to Operating System version information. Note: This is the exact same information provided by System.Environment.

- static Component OSRuntime `[get]`

  This property allows access to information about the currently running Operating System. This property is primarily useful for acquiring further information about Sealevel Systems embedded Windows runtime images.

- static Version Version `[get]`

  This property allows access to the version information of the Talos framework.

- static string Name `[get, set]`

  This property allows access to the device name of the environment's host machine.

- static string Description `[get, set]`

  This property allows access to a device description of the environment's host machine.

- static string Processor `[get]`

  This property returns a string representation of the system's processor.

- static OwnerInformation Owner `[get, set]`

  This property is used to get/set the device owner information. This is a good place to store some unique information to help identify this device. The information kept here can also be used to provide global access for contact information should a problem with the device ever be found. See OwnerInformation for further details.

6.3.1 Detailed Description

This class provides version information about the current runtime environment. This class mimics some of the behavior of System.Environment, but extends some of the functionality to allow access to Sealevel Systems specific information.

6.3.2 Property Documentation

6.3.2.1 OperatingSystem Talos::Environment::OSVersion `[static, get]`

This property allows access to Operating System version information. Note: This is the exact same information provided by System.Environment.

6.3.2.2 Component Talos::Environment::OSRuntime [static, get]

This property allows access to information about the currently running Operating System. This property is primarily useful for acquiring further information about Sealevel Systems embedded Windows runtime images.

6.3.2.3 Version Talos::Environment::Version [static, get]

This property allows access to the version information of the Talos framework.

6.3.2.4 string Talos::Environment::Name [static, get, set]

This property allows access to the device name of the environment's host machine.

Exceptions:

NotImplementedException This property is only supported on Windows CE.

Exception Any other exceptions are critical errors caused by failed Registry access.

6.3.2.5 string Talos::Environment::Description [static, get, set]

This property allows access to a device description of the environment's host machine.

Exceptions:

NotImplementedException This property is only supported on Windows CE.

Exception Any other exceptions are critical errors caused by failed Registry access.

6.3.2.6 string Talos::Environment::Processor [static, get]

This property returns a string representation of the system's processor.

6.3.2.7 OwnerInformation Talos::Environment::Owner [static, get, set]

This property is used to get/set the device owner information. This is a good place to store some unique information to help identify this device. The information kept here can also be used to provide global access for contact information should a problem with the device ever be found. See OwnerInformation for further details.

6.4 Talos::IO::AnalogInPoint Class Reference

Base class for a AnalogInPoint.

Inherits Talos::IO::AnalogPoint.

Public Member Functions

- int GetSampledValue ()

  Return an analog to digital conversion. The conversion uses the specified SampleCount, SampleDelay, and SampleMethod to combine a set of samples and return a single value.

- float Convert (int rawValue)
- float Convert (int rawValue, AnalogUnit unit)
- int Convert (float value)

  Convert a floating point AnalogPoint value into a corresponding raw integer value.

- int Convert (float value, AnalogUnit unit)

  Convert a floating point AnalogPoint value into a corresponding raw integer value.

Static Public Member Functions

- static int Average (int[ ] values)

  Calculate the average of the values contained in the array passed in.

- static float Convert (int rawValue, float slope, float offset)

  Convert a raw AnalogPoint value into a corresponding floating point value.

- static int Convert (float value, float slope, float offset)

  Convert a floating point AnalogPoint value into a corresponding raw integer value.

Protected Member Functions

- AnalogInPoint ()

  Default constructor.

- abstract int[ ] GetValues (int count)

  Retreieve the number of samples required with the necessary wait between polls.

- abstract void SetDefaults ()

  Must be implemented by children to setup default Min/Max, Slope, and Offset. Used during the setting of the Unit property.

Properties

- int SampleCount [get, set]

  Gets and sets a number of readings will be taken and averaged together to collect a valid input value.

- int SampleDelay [get, set]

  Gets and sets a value that represents the minimum amount of time delay between each input reading in milliseconds.

- SampleDelegate SampleMethod [get, set]
- override string Mode [get, set]

  Gets and sets the conversion range for this point.

- override AnalogUnit Unit [get, set]

  Gets and sets the modifier that determines the default Slope and Offset for RawValue to Value conversions.

- override int RawValue [get, set]

  Gets and sets the Raw digital value for this point.

- override float Value [get, set]

  Gets and sets the floating point representation of this point in terms of the AnalogUnit specified through Unit.

- int MinRawValue [get, set]

  Gets the minimum possible RawValue for this point.

- int MaxRawValue [get, set]

  Gets the maximum possible RawValue for this point.

- ReadOnlyCollection< string > SupportedModes [get]

  Gets a list of supported conversion ranges for this point.

- float Slope [get, set]

  Gets the slope field allows for calibration of the conversion from raw digital values to floating point.

- float Offset [get, set]

  Gets the offset allows for further calibration of the digital to floating point number conversion.

- float MinValue [get]

  Gets the floating point representation of minimum possible Value this point is capable of in terms of the AnalogUnit specified through Unit.

- float MaxValue [get]

  Gets the floating point representation of maximum possible Value this point is capable of in terms of the AnalogUnit specified through Unit.

- ReadOnlyCollection< AnalogUnit > SupportedUnits [get]

  Gets a list of supported AnalogUnits for this point.

- int Index [get, set]

  The Talos assigned point ID number. This is relative to other Talos points of the same type.

- string Connection [get, set]

  Describes the connection type of the device.

- string ConnectionData [get, set]

  Describes the connection data for the connection type of the device.

- IOType Type [get, set]

  Describes the nature of the point. (Analog or Digital).

- Direction Direction [get, set]

  Describes the nature of the point. (Input or Output).

- string Description [get, set]

  A descriptive notation describing the hardware attached to the point. (Examples: "Overhead Light", "Door 3", "Motor 1", "Pump 3").

- bool Online [get, set]

  Denotes the availability status of the point. True corresponds to ONLINE, false to OFFLINE.

- string Function [get]

  Indicates the unique descriptive notation of the point. (Example: "Digital Output 24").

- string Identifier [get]

  Indicates the unique point identification code as assigned by TIO. (Example: "DO_24").

6.4.1 Detailed Description

Base class for a AnalogInPoint.

See also:

AnalogOutPoint

6.4.2 Constructor & Destructor Documentation

6.4.2.1 Talos::IO::AnalogInPoint::AnalogInPoint () [protected]

Default constructor. Sets the properties to the following values: Direction to Direction.Input, SampleCount to 5, and SampleDelay to 10.

6.4.3 Member Function Documentation

6.4.3.1 static int Talos::IO::AnalogInPoint::Average (int[ ] *values*) [static]

Calculate the average of the values contained in the array passed in.

Parameters:

values The array of values to be averaged.

Returns:

An integer value indicating the average of the values array.

6.4.3.2 int Talos::IO::AnalogInPoint::GetSampledValue ()

Return an analog to digital conversion. The conversion uses the specified SampleCount, SampleDelay, and SampleMethod to combine a set of samples and return a single value.

Returns:

A single value representing a analog to digital conversion.

Exceptions:

IOException Invalid point configuration. -OR- Cannot access specified AnalogInPoint.

6.4.3.3 abstract int [ ] Talos::IO::AnalogInPoint::GetValues (int *count*) [protected, pure virtual]

Retreieve the number of samples required with the necessary wait between polls.

Returns:

An array consisting of the required samples.

6.4.3.4 abstract void Talos::IO::AnalogInPoint::SetDefaults () [protected, pure virtual]

Must be implemented by children to setup default Min/Max, Slope, and Offset. Used during the setting of the Unit property.

6.4.3.5 static float Talos::IO::AnalogPoint::Convert (int *rawValue*, float *slope*, float *offset*) [static, inherited]

Convert a raw AnalogPoint value into a corresponding floating point value.

Parameters:

rawValue The value to convert using the given slope and offset.

slope The slope (m) to use in the conversion formula y = mx + b.

offset The offset (b) to use in the converion formula y = mx + b.

Returns:

A floating point value indicating the converted value.

6.4.3.6 float Talos::IO::AnalogPoint::Convert (int *rawValue*) [inherited]

Convert a raw AnalogPoint value into a corresponding floating point value.

Parameters:

rawValue The value to convert using the current Slope and Offset.

Returns:

A floating point value indicating the converted value.

6.4.3.7 float Talos::IO::AnalogPoint::Convert (int *rawValue*, AnalogUnit *unit*) [inherited]

Convert a raw AnalogPoint value into a corresponding floating point value.

Parameters:

rawValue The raw value to convert to floating point.
unit The units to use for this conversion.

Returns:

A floating point representation of the AnalogInPoint value.

Exceptions:

ArgumentException Specified unit is not supported.

6.4.3.8 static int Talos::IO::AnalogPoint::Convert (float *value*, float *slope*, float *offset*) [static, inherited]

Convert a floating point AnalogPoint value into a corresponding raw integer value.

Parameters:

value The value to convert using the given slope and offset.
slope The slope (m) to use in the conversion formula y = mx + b.
offset The offset (b) to use in the converion formula y = mx + b.

Returns:

An integer value indicating the converted value.

6.4.3.9 int Talos::IO::AnalogPoint::Convert (float *value*) [inherited]

Convert a floating point AnalogPoint value into a corresponding raw integer value.

Parameters:

value The value to convert using the current Slope and Offset.

Returns:

An integer value indicating the converted value.

6.4.3.10 int Talos::IO::AnalogPoint::Convert (float *value*, AnalogUnit *unit*) [inherited]

Convert a floating point AnalogPoint value into a corresponding raw integer value.

Parameters:

value The raw value to convert.

unit The units to use for this conversion.

Returns:

An integer value indicating the converted value.

Exceptions:

ArgumentException Specified unit is not supported.

6.4.4 Property Documentation

6.4.4.1 int Talos::IO::AnalogInPoint::SampleCount [get, set]

Gets and sets a number of readings will be taken and averaged together to collect a valid input value.

6.4.4.2 int Talos::IO::AnalogInPoint::SampleDelay [get, set]

Gets and sets a value that represents the minimum amount of time delay between each input reading in milliseconds.

6.4.4.3 SampleDelegate Talos::IO::AnalogInPoint::SampleMethod [get, set]

Gets and sets a value that defines the method of combining multiple samples over a short period of time into a single value. The default behavior is a simply to average. This behavior may be modified by supplying another delegate of type SampleDelegate.

Exceptions:

ArgumentNullException SampleDelegate may not be null.

6.4.4.4 override string Talos::IO::AnalogInPoint::Mode [get, set]

Gets and sets the conversion range for this point.

Exceptions:

ArgumentException Unsupported AnalogInPoint range.

Reimplemented from Talos::IO::AnalogPoint.

6.4.4.5 override AnalogUnit Talos::IO::AnalogInPoint::Unit [get, set]

Gets and sets the modifier that determines the default Slope and Offset for RawValue to Value conversions.

Exceptions:

ArgumentException Unsupported AnalogInPoint unit.

See also:

AnalogPoint.SupportedUnits

Reimplemented from Talos::IO::AnalogPoint.

6.4.4.6 override int Talos::IO::AnalogInPoint::RawValue [get, set]

Gets and sets the Raw digital value for this point.

Warning:

AnalogInPoints cannot have a RawValue assigned. Setting this property will result in the InvalidOperationException.

Exceptions:

InvalidOperationException AnalogInPoints cannot have a value assigned to them.

Reimplemented from Talos::IO::AnalogPoint.

6.4.4.7 override float Talos::IO::AnalogInPoint::Value [get, set]

Gets and sets the floating point representation of this point in terms of the AnalogUnit specified through Unit.

Warning:

AnalogInPoints cannot have a Value assigned. Setting this property will result in the InvalidOperationException.

Exceptions:

InvalidOperationException AnalogInPoints cannot have a value assigned to them.

Reimplemented from Talos::IO::AnalogPoint.

6.4.4.8 int Talos::IO::AnalogPoint::MinRawValue [get, set, inherited]

Gets the minimum possible RawValue for this point.

6.4.4.9 int Talos::IO::AnalogPoint::MaxRawValue [get, set, inherited]

Gets the maximum possible RawValue for this point.

6.4.4.10 ReadOnlyCollection<string> Talos::IO::AnalogPoint::SupportedModes [get, inherited]

Gets a list of supported conversion ranges for this point.

6.4.4.11 float Talos::IO::AnalogPoint::Slope [get, set, inherited]

Gets the slope field allows for calibration of the conversion from raw digital values to floating point.

6.4.4.12 float Talos::IO::AnalogPoint::Offset [get, set, inherited]

Gets the offset allows for further calibration of the digital to floating point number conversion.

6.4.4.13 float Talos::IO::AnalogPoint::MinValue [get, inherited]

Gets the floating point representation of minimum possible Value this point is capable of in terms of the AnalogUnit specified through Unit.

6.4.4.14 float Talos::IO::AnalogPoint::MaxValue [get, inherited]

Gets the floating point representation of maximum possible Value this point is capable of in terms of the AnalogUnit specified through Unit.

6.4.4.15 ReadOnlyCollection<AnalogUnit> Talos::IO::AnalogPoint::SupportedUnits [get, inherited]

Gets a list of supported AnalogUnits for this point.

6.4.4.16 int Talos::IO::Point::Index [get, set, inherited]

The Talos assigned point ID number. This is relative to other Talos points of the same type.

6.4.4.17 string Talos::IO::Point::Connection [get, set, inherited]

Describes the connection type of the device.

6.4.4.18 string Talos::IO::Point::ConnectionData [get, set, inherited]

Describes the connection data for the connection type of the device.

6.4.4.19 IOType Talos::IO::Point::Type [get, set, inherited]

Describes the nature of the point. (Analog or Digital).

6.4.4.20 Direction Talos::IO::Point::Direction [get, set, inherited]

Describes the nature of the point. (Input or Output).

6.4.4.21 string Talos::IO::Point::Description [get, set, inherited]

A descriptive notation describing the hardware attached to the point. (Examples: "Overhead Light", "Door 3", "Motor 1", "Pump 3").

6.4.4.22 bool Talos::IO::Point::Online [get, set, inherited]

Denotes the availability status of the point. True corresponds to ONLINE, false to OFFLINE.

6.4.4.23 string Talos::IO::Point::Function [get, inherited]

Indicates the unique descriptive notation of the point. (Example: "Digital Output 24").

6.4.4.24 string Talos::IO::Point::Identifier [get, inherited]

Indicates the unique point identification code as assigned by TIO. (Example: "DO_24").

6.5 Talos::IO::AnalogOutPoint Class Reference

Base class for a AnalogOutPoint.

Inherits Talos::IO::AnalogPoint.

Public Member Functions

- float Convert (int rawValue)
- float Convert (int rawValue, AnalogUnit unit)
- int Convert (float value)

  Convert a floating point AnalogPoint value into a corresponding raw integer value.

- int Convert (float value, AnalogUnit unit)

  Convert a floating point AnalogPoint value into a corresponding raw integer value.

Static Public Member Functions

- static float Convert (int rawValue, float slope, float offset)

  Convert a raw AnalogPoint value into a corresponding floating point value.

- static int Convert (float value, float slope, float offset)

  Convert a floating point AnalogPoint value into a corresponding raw integer value.

Protected Member Functions

- AnalogOutPoint ()

  Default constructor.

Properties

- override float Value `[get, set]`

  This is a floating point representation of the Analog input in terms of the AnalogUnit specified through Unit.

- override int RawValue `[get, set]`

  This is an interface to acquire the raw digital value recorded by the A/D converter.

- int MinRawValue `[get, set]`

  Gets the minimum possible RawValue for this point.

- int MaxRawValue `[get, set]`

  Gets the maximum possible RawValue for this point.

- ReadOnlyCollection< string > SupportedModes `[get]`

  Gets a list of supported conversion ranges for this point.

- virtual string Mode [get, set]

  Gets the currently selected conversion range for this point.

- float Slope [get, set]

  Gets the slope field allows for calibration of the conversion from raw digital values to floating point.

- float Offset [get, set]

  Gets the offset allows for further calibration of the digital to floating point number conversion.

- float MinValue [get]

  Gets the floating point representation of minimum possible Value this point is capable of in terms of the AnalogUnit specified through Unit.

- float MaxValue [get]

  Gets the floating point representation of maximum possible Value this point is capable of in terms of the AnalogUnit specified through Unit.

- ReadOnlyCollection< AnalogUnit > SupportedUnits [get]

  Gets a list of supported AnalogUnits for this point.

- virtual AnalogUnit Unit [get, set]

  Gets the value modifier that determines the default Slope and Offset for RawValue to Value conversions.

- int Index [get, set]

  The Talos assigned point ID number. This is relative to other Talos points of the same type.

- string Connection [get, set]

  Describes the connection type of the device.

- string ConnectionData [get, set]

  Describes the connection data for the connection type of the device.

- IOType Type [get, set]

  Describes the nature of the point. (Analog or Digital).

- Direction Direction [get, set]

  Describes the nature of the point. (Input or Output).

- string Description [get, set]

  A descriptive notation describing the hardware attached to the point. (Examples: "Overhead Light", "Door 3", "Motor 1", "Pump 3").

- bool Online [get, set]

  Denotes the availability status of the point. True corresponds to ONLINE, false to OFFLINE.

- string Function [get]

  Indicates the unique descriptive notation of the point. (Example: "Digital Output 24").

- string Identifier [get]

  Indicates the unique point identification code as assigned by TIO. (Example: "DO_24").

6.5.1 Detailed Description

Base class for a AnalogOutPoint.

See also:

AnalogInPoint

6.5.2 Constructor & Destructor Documentation

6.5.2.1 Talos::IO::AnalogOutPoint::AnalogOutPoint () [protected]

Default constructor. Sets the properties to the following values: Direction to Direction.Output.

6.5.3 Member Function Documentation

6.5.3.1 static float Talos::IO::AnalogPoint::Convert (int *rawValue*, float *slope*, float *offset*) [static, inherited]

Convert a raw AnalogPoint value into a corresponding floating point value.

Parameters:

rawValue The value to convert using the given slope and offset.

slope The slope (m) to use in the conversion formula y = mx + b.

offset The offset (b) to use in the converion formula y = mx + b.

Returns:

A floating point value indicating the converted value.

6.5.3.2 float Talos::IO::AnalogPoint::Convert (int *rawValue*) [inherited]

Convert a raw AnalogPoint value into a corresponding floating point value.

Parameters:

rawValue The value to convert using the current Slope and Offset.

Returns:

A floating point value indicating the converted value.

6.5.3.3 float Talos::IO::AnalogPoint::Convert (int *rawValue*, AnalogUnit *unit*) [inherited]

Convert a raw AnalogPoint value into a corresponding floating point value.

Parameters:

rawValue The raw value to convert to floating point.

unit The units to use for this conversion.

Returns:

A floating point representation of the AnalogInPoint value.

Exceptions:

ArgumentException Specified unit is not supported.

6.5.3.4 static int Talos::IO::AnalogPoint::Convert (float *value*, float *slope*, float *offset*) [static, inherited]

Convert a floating point AnalogPoint value into a corresponding raw integer value.

Parameters:

value The value to convert using the given slope and offset.

slope The slope (m) to use in the conversion formula y = mx + b.

offset The offset (b) to use in the converion formula y = mx + b.

Returns:

An integer value indicating the converted value.

6.5.3.5 int Talos::IO::AnalogPoint::Convert (float *value*) [inherited]

Convert a floating point AnalogPoint value into a corresponding raw integer value.

Parameters:

value The value to convert using the current Slope and Offset.

Returns:

An integer value indicating the converted value.

6.5.3.6 int Talos::IO::AnalogPoint::Convert (float *value*, AnalogUnit *unit*) [inherited]

Convert a floating point AnalogPoint value into a corresponding raw integer value.

Parameters:

value The raw value to convert.

unit The units to use for this conversion.

Returns:

An integer value indicating the converted value.

Exceptions:

ArgumentException Specified unit is not supported.

6.5.4 Property Documentation

6.5.4.1 override float Talos::IO::AnalogOutPoint::Value [get, set]

This is a floating point representation of the Analog input in terms of the AnalogUnit specified through Unit.

Exceptions:

NotImplementedException This property has not been implemented.

Reimplemented from Talos::IO::AnalogPoint.

6.5.4.2 override int Talos::IO::AnalogOutPoint::RawValue [get, set]

This is an interface to acquire the raw digital value recorded by the A/D converter.

Exceptions:

NotImplementedException This property has not been implemented.

Reimplemented from Talos::IO::AnalogPoint.

6.5.4.3 int Talos::IO::AnalogPoint::MinRawValue [get, set, inherited]

Gets the minimum possible RawValue for this point.

6.5.4.4 int Talos::IO::AnalogPoint::MaxRawValue [get, set, inherited]

Gets the maximum possible RawValue for this point.

6.5.4.5 ReadOnlyCollection<string> Talos::IO::AnalogPoint::SupportedModes [get, inherited]

Gets a list of supported conversion ranges for this point.

6.5.4.6 virtual string Talos::IO::AnalogPoint::Mode [get, set, inherited]

Gets the currently selected conversion range for this point.

Reimplemented in Talos::IO::AnalogInPoint.

6.5.4.7 float Talos::IO::AnalogPoint::Slope [get, set, inherited]

Gets the slope field allows for calibration of the conversion from raw digital values to floating point.

6.5.4.8 float Talos::IO::AnalogPoint::Offset [get, set, inherited]

Gets the offset allows for further calibration of the digital to floating point number conversion.

6.5.4.9 float Talos::IO::AnalogPoint::MinValue [get, inherited]

Gets the floating point representation of minimum possible Value this point is capable of in terms of the AnalogUnit specified through Unit.

6.5.4.10 float Talos::IO::AnalogPoint::MaxValue [get, inherited]

Gets the floating point representation of maximum possible Value this point is capable of in terms of the AnalogUnit specified through Unit.

6.5.4.11 ReadOnlyCollection<AnalogUnit> Talos::IO::AnalogPoint::SupportedUnits [get, inherited]

Gets a list of supported AnalogUnits for this point.

6.5.4.12 virtual AnalogUnit Talos::IO::AnalogPoint::Unit [get, set, inherited]

Gets the value modifier that determines the default Slope and Offset for RawValue to Value conversions.

Reimplemented in Talos::IO::AnalogInPoint.

6.5.4.13 int Talos::IO::Point::Index [get, set, inherited]

The Talos assigned point ID number. This is relative to other Talos points of the same type.

6.5.4.14 string Talos::IO::Point::Connection [get, set, inherited]

Describes the connection type of the device.

6.5.4.15 string Talos::IO::Point::ConnectionData [get, set, inherited]

Describes the connection data for the connection type of the device.

6.5.4.16 IOType Talos::IO::Point::Type [get, set, inherited]

Describes the nature of the point. (Analog or Digital).

6.5.4.17 Direction Talos::IO::Point::Direction [get, set, inherited]

Describes the nature of the point. (Input or Output).

6.5.4.18 string Talos::IO::Point::Description [get, set, inherited]

A descriptive notation describing the hardware attached to the point. (Examples: "Overhead Light", "Door 3", "Motor 1", "Pump 3").

6.5.4.19 bool Talos::IO::Point::Online [get, set, inherited]

Denotes the availability status of the point. True corresponds to ONLINE, false to OFFLINE.

6.5.4.20 string Talos::IO::Point::Function [get, inherited]

Indicates the unique descriptive notation of the point. (Example: "Digital Output 24").

6.5.4.21 string Talos::IO::Point::Identifier [get, inherited]

Indicates the unique point identification code as assigned by TIO. (Example: "DO_24").

6.6 Talos::IO::AnalogPoint Class Reference

Base class for a AnalogPoint.

Inherits Talos::IO::Point.

Inherited by Talos::IO::AnalogInPoint, and Talos::IO::AnalogOutPoint.

Public Member Functions

- float Convert (int rawValue)
- float Convert (int rawValue, AnalogUnit unit)
- int Convert (float value)

  Convert a floating point AnalogPoint value into a corresponding raw integer value.

- int Convert (float value, AnalogUnit unit)

  Convert a floating point AnalogPoint value into a corresponding raw integer value.

Static Public Member Functions

- static float Convert (int rawValue, float slope, float offset)

  Convert a raw AnalogPoint value into a corresponding floating point value.

- static int Convert (float value, float slope, float offset)

  Convert a floating point AnalogPoint value into a corresponding raw integer value.

Protected Member Functions

- AnalogPoint ()

  Default constructor.

Properties

- virtual int RawValue `[get, set]`

  Gets or sets the Raw digital value for this point.

- int MinRawValue `[get, set]`

  Gets the minimum possible RawValue for this point.

- int MaxRawValue `[get, set]`

  Gets the maximum possible RawValue for this point.

- ReadOnlyCollection< string > SupportedModes `[get]`

  Gets a list of supported conversion ranges for this point.

- virtual string Mode `[get, set]`

Gets the currently selected conversion range for this point.

- float Slope `[get, set]`
  Gets the slope field allows for calibration of the conversion from raw digital values to floating point.
- float Offset `[get, set]`
  Gets the offset allows for further calibration of the digital to floating point number conversion.
- virtual float Value `[get, set]`
  This is a floating point representation of this point in terms of the AnalogUnit specified through Unit.
- float MinValue `[get]`
  Gets the floating point representation of minimum possible Value this point is capable of in terms of the AnalogUnit specified through Unit.
- float MaxValue `[get]`
  Gets the floating point representation of maximum possible Value this point is capable of in terms of the AnalogUnit specified through Unit.
- ReadOnlyCollection< AnalogUnit > SupportedUnits `[get]`
  Gets a list of supported AnalogUnits for this point.
- virtual AnalogUnit Unit `[get, set]`
  Gets the value modifier that determines the default Slope and Offset for RawValue to Value conversions.
- int Index `[get, set]`
  The Talos assigned point ID number. This is relative to other Talos points of the same type.
- string Connection `[get, set]`
  Describes the connection type of the device.
- string ConnectionData `[get, set]`
  Describes the connection data for the connection type of the device.
- IOType Type `[get, set]`
  Describes the nature of the point. (Analog or Digital).
- Direction Direction `[get, set]`
  Describes the nature of the point. (Input or Output).
- string Description `[get, set]`
  A descriptive notation describing the hardware attached to the point. (Examples: "Overhead Light", "Door 3", "Motor 1", "Pump 3").
- bool Online `[get, set]`
  Denotes the availability status of the point. True corresponds to ONLINE, false to OFFLINE.
- string Function `[get]`
  Indicates the unique descriptive notation of the point. (Example: "Digital Output 24").
- string Identifier `[get]`
  Indicates the unique point identification code as assigned by TIO. (Example: "DO_24").

6.6.1 Detailed Description

Base class for a AnalogPoint.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 Talos::IO::AnalogPoint::AnalogPoint () [protected]

Default constructor. Sets the properties to the following values: Type to IOType.Analog, Slope to 0.00122F, and Offset to 0.

6.6.3 Member Function Documentation

6.6.3.1 static float Talos::IO::AnalogPoint::Convert (int *rawValue*, float *slope*, float *offset*) [static]

Convert a raw AnalogPoint value into a corresponding floating point value.

Parameters:

rawValue The value to convert using the given slope and offset.

slope The slope (m) to use in the conversion formula y = mx + b.

offset The offset (b) to use in the converion formula y = mx + b.

Returns:

A floating point value indicating the converted value.

6.6.3.2 float Talos::IO::AnalogPoint::Convert (int *rawValue*)

Convert a raw AnalogPoint value into a corresponding floating point value.

Parameters:

rawValue The value to convert using the current Slope and Offset.

Returns:

A floating point value indicating the converted value.

6.6.3.3 float Talos::IO::AnalogPoint::Convert (int *rawValue*, AnalogUnit *unit*)

Convert a raw AnalogPoint value into a corresponding floating point value.

Parameters:

rawValue The raw value to convert to floating point.

unit The units to use for this conversion.

Returns:

A floating point representation of the AnalogInPoint value.

Exceptions:

ArgumentException Specified unit is not supported.

6.6.3.4 static int Talos::IO::AnalogPoint::Convert (float *value*, float *slope*, float *offset*) [static]

Convert a floating point AnalogPoint value into a corresponding raw integer value.

Parameters:

value The value to convert using the given slope and offset.

slope The slope (m) to use in the conversion formula y = mx + b.

offset The offset (b) to use in the converion formula y = mx + b.

Returns:

An integer value indicating the converted value.

6.6.3.5 int Talos::IO::AnalogPoint::Convert (float *value*)

Convert a floating point AnalogPoint value into a corresponding raw integer value.

Parameters:

value The value to convert using the current Slope and Offset.

Returns:

An integer value indicating the converted value.

6.6.3.6 int Talos::IO::AnalogPoint::Convert (float *value*, AnalogUnit *unit*)

Convert a floating point AnalogPoint value into a corresponding raw integer value.

Parameters:

value The raw value to convert.

unit The units to use for this conversion.

Returns:

An integer value indicating the converted value.

Exceptions:

ArgumentException Specified unit is not supported.

6.6.4 Property Documentation

6.6.4.1 virtual int Talos::IO::AnalogPoint::RawValue [get, set]

Gets or sets the Raw digital value for this point.

Reimplemented in Talos::IO::AnalogInPoint, and Talos::IO::AnalogOutPoint.

6.6.4.2 int Talos::IO::AnalogPoint::MinRawValue [get, set]

Gets the minimum possible RawValue for this point.

6.6.4.3 int Talos::IO::AnalogPoint::MaxRawValue [get, set]

Gets the maximum possible RawValue for this point.

6.6.4.4 ReadOnlyCollection<string> Talos::IO::AnalogPoint::SupportedModes [get]

Gets a list of supported conversion ranges for this point.

6.6.4.5 virtual string Talos::IO::AnalogPoint::Mode [get, set]

Gets the currently selected conversion range for this point.

Reimplemented in Talos::IO::AnalogInPoint.

6.6.4.6 float Talos::IO::AnalogPoint::Slope [get, set]

Gets the slope field allows for calibration of the conversion from raw digital values to floating point.

6.6.4.7 float Talos::IO::AnalogPoint::Offset [get, set]

Gets the offset allows for further calibration of the digital to floating point number conversion.

6.6.4.8 virtual float Talos::IO::AnalogPoint::Value [get, set]

This is a floating point representation of this point in terms of the AnalogUnit specified through Unit.

Reimplemented in Talos::IO::AnalogInPoint, and Talos::IO::AnalogOutPoint.

6.6.4.9 float Talos::IO::AnalogPoint::MinValue [get]

Gets the floating point representation of minimum possible Value this point is capable of in terms of the AnalogUnit specified through Unit.

6.6.4.10 float Talos::IO::AnalogPoint::MaxValue [get]

Gets the floating point representation of maximum possible Value this point is capable of in terms of the AnalogUnit specified through Unit.

6.6.4.11 ReadOnlyCollection<AnalogUnit> Talos::IO::AnalogPoint::SupportedUnits [get]

Gets a list of supported AnalogUnits for this point.

6.6.4.12 virtual AnalogUnit Talos::IO::AnalogPoint::Unit [get, set]

Gets the value modifier that determines the default Slope and Offset for RawValue to Value conversions.

Reimplemented in Talos::IO::AnalogInPoint.

6.6.4.13 int Talos::IO::Point::Index [get, set, inherited]

The Talos assigned point ID number. This is relative to other Talos points of the same type.

6.6.4.14 string Talos::IO::Point::Connection [get, set, inherited]

Describes the connection type of the device.

6.6.4.15 string Talos::IO::Point::ConnectionData [get, set, inherited]

Describes the connection data for the connection type of the device.

6.6.4.16 IOType Talos::IO::Point::Type [get, set, inherited]

Describes the nature of the point. (Analog or Digital).

6.6.4.17 Direction Talos::IO::Point::Direction [get, set, inherited]

Describes the nature of the point. (Input or Output).

6.6.4.18 string Talos::IO::Point::Description [get, set, inherited]

A descriptive notation describing the hardware attached to the point. (Examples: "Overhead Light", "Door 3", "Motor 1", "Pump 3").

6.6.4.19 bool Talos::IO::Point::Online [get, set, inherited]

Denotes the availability status of the point. True corresponds to ONLINE, false to OFFLINE.

6.6.4.20 string Talos::IO::Point::Function [get, inherited]

Indicates the unique descriptive notation of the point. (Example: "Digital Output 24").

6.6.4.21 string Talos::IO::Point::Identifier [get, inherited]

Indicates the unique point identification code as assigned by TIO. (Example: "DO_24").

6.7 Talos::IO::CanMailbox Class Reference

Data structure for configuring a mailbox for receiving or sending messages.

Properties

- int ID `[get, set]`

  Mailbox ID.

- bool Extended `[get, set]`

  Enableds the Extended identifier for the CAN frames that will use 29 bits for identification purposes instead of 11 for thestandard CAN frame.

- int Mask `[get, set]`

  Mask for filtering the incoming messages.

- CanMailboxMode MailBoxMode `[get, set]`

  Configure the mailbox mode (send, receive).

- int NumberOfMailboxes `[get, set]`

  Number of mailboxes to use.

6.7.1 Detailed Description

Data structure for configuring a mailbox for receiving or sending messages.

6.7.2 Property Documentation

6.7.2.1 int Talos::IO::CanMailbox::ID `[get, set]`

Mailbox ID.

6.7.2.2 bool Talos::IO::CanMailbox::Extended `[get, set]`

Enableds the Extended identifier for the CAN frames that will use 29 bits for identification purposes instead of 11 for thestandard CAN frame.

6.7.2.3 int Talos::IO::CanMailbox::Mask `[get, set]`

Mask for filtering the incoming messages.

6.7.2.4 CanMailboxMode Talos::IO::CanMailbox::MailBoxMode `[get, set]`

Configure the mailbox mode (send, receive).

6.7.2.5 int Talos::IO::CanMailbox::NumberOfMailboxes [get, set]

Number of mailboxes to use.

6.8 Talos::IO::CanPoint Class Reference

Base class for CanPoint.

Inherits Talos::IO::Point.

Public Member Functions

- abstract void Configure (CanMailbox[ ] mailboxes)

  Configures the CAN mailboxes for the point. Each mailbox can be configured in receive or in transmit mode independently.

- abstract void Reset ()

  Clears the mailbox configurations and reset the point.

- abstract CanMessage[ ] Read (int id, int count)

  Reads a CanMessage from the point.

- abstract void Write (int id, CanMessage message)

  Write a CanMessage to the point.

- abstract void Write (int id, CanMessage[ ] messages)

  Write the CanMessages to the point.

Properties

- abstract int BaudRate `[get, set]`

  Gets and sets the baud rate for the CAN device. These changes will not take effect until the unit has been rebooted.

- int Index `[get, set]`

  The Talos assigned point ID number. This is relative to other Talos points of the same type.

- string Connection `[get, set]`

  Describes the connection type of the device.

- string ConnectionData `[get, set]`

  Describes the connection data for the connection type of the device.

- IOType Type `[get, set]`

  Describes the nature of the point. (Analog or Digital).

- Direction Direction `[get, set]`

  Describes the nature of the point. (Input or Output).

- string Description `[get, set]`

  A descriptive notation describing the hardware attached to the point. (Examples: "Overhead Light", "Door 3", "Motor 1", "Pump 3").

- bool Online [get, set]

  Denotes the availability status of the point. True corresponds to ONLINE, false to OFFLINE.

- string Function [get]

  Indicates the unique descriptive notation of the point. (Example: "Digital Output 24").

- string Identifier [get]

  Indicates the unique point identification code as assigned by TIO. (Example: "DO_24").

6.8.1 Detailed Description

Base class for CanPoint.

6.8.2 Member Function Documentation

6.8.2.1 abstract void Talos::IO::CanPoint::Configure (CanMailbox[ ] *mailboxes*) [pure virtual]

Configures the CAN mailboxes for the point. Each mailbox can be configured in receive or in transmit mode independently.

Parameters:

mailboxes An array of mailbox configurations

6.8.2.2 abstract void Talos::IO::CanPoint::Reset () [pure virtual]

Clears the mailbox configurations and reset the point.

6.8.2.3 abstract CanMessage [ ] Talos::IO::CanPoint::Read (int *id*, int *count*) [pure virtual]

Reads a CanMessage from the point.

Parameters:

id Mailbox settings ID

count Number of messages to read

Returns:

Array of messages read

6.8.2.4 abstract void Talos::IO::CanPoint::Write (int *id*, CanMessage *message*) [pure virtual]

Write a CanMessage to the point.

Parameters:

id Mailbox settings ID

message CanMessage to write

6.8.2.5 abstract void Talos::IO::CanPoint::Write (int *id*, CanMessage[ ] *messages*) [pure virtual]

Write the CanMessages to the point.

Parameters:

id Mailbox settings ID

messages CanMessages to write

6.8.3 Property Documentation

6.8.3.1 abstract int Talos::IO::CanPoint::BaudRate [get, set]

Gets and sets the baud rate for the CAN device. These changes will not take effect until the unit has been rebooted.

6.8.3.2 int Talos::IO::Point::Index [get, set, inherited]

The Talos assigned point ID number. This is relative to other Talos points of the same type.

6.8.3.3 string Talos::IO::Point::Connection [get, set, inherited]

Describes the connection type of the device.

6.8.3.4 string Talos::IO::Point::ConnectionData [get, set, inherited]

Describes the connection data for the connection type of the device.

6.8.3.5 IOType Talos::IO::Point::Type [get, set, inherited]

Describes the nature of the point. (Analog or Digital).

6.8.3.6 Direction Talos::IO::Point::Direction [get, set, inherited]

Describes the nature of the point. (Input or Output).

6.8.3.7 string Talos::IO::Point::Description [get, set, inherited]

A descriptive notation describing the hardware attached to the point. (Examples: "Overhead Light", "Door 3", "Motor 1", "Pump 3").

6.8.3.8 bool Talos::IO::Point::Online [get, set, inherited]

Denotes the availability status of the point. True corresponds to ONLINE, false to OFFLINE.

6.8.3.9 string Talos::IO::Point::Function [get, inherited]

Indicates the unique descriptive notation of the point. (Example: "Digital Output 24").

6.8.3.10 string Talos::IO::Point::Identifier [get, inherited]

Indicates the unique point identification code as assigned by TIO. (Example: "DO_24").

6.9 Talos::IO::Counter Class Reference

Base class for Counter.

Inherits Talos::IO::Point.

Public Member Functions

- abstract void SetDataValue (int data)

  Sets the data value for the counter.

- abstract void Configure (CounterMode mode, CounterIndexMode index, CounterIndexType indexType, CounterWidth width, CounterFlag flag)

  Configure the counter point.

- abstract void ResetCounter ()

  Reset the abstract counter.

- abstract void ResetPulse ()

  Reset the pulse trigger.

Protected Member Functions

- Counter ()

  Create protected abstract Counter.

Properties

- CounterMode Mode `[get, set]`

  Describes the counter configuration. (Free, Range, ...).

- CounterIndexMode IndexMode `[get, set]`

  Describes the index configuration. (Reset, Instant, ...).

- CounterIndexType IndexType `[get, set]`

  Describes the index configuration. (Reset, Instant, ...).

- CounterWidth Width `[get, set]`

  Describes the byte length used. (One, Two, Three, Four).

- CounterFlag Flag `[get, set]`

  Describes which event to report on. (Index, Compare, ...).

- abstract int Value `[get]`

  Returns the current value of the counter.

- int Index `[get, set]`

The Talos assigned point ID number. This is relative to other Talos points of the same type.

- string Connection `[get, set]`

  Describes the connection type of the device.

- string ConnectionData `[get, set]`

  Describes the connection data for the connection type of the device.

- IOType Type `[get, set]`

  Describes the nature of the point. (Analog or Digital).

- Direction Direction `[get, set]`

  Describes the nature of the point. (Input or Output).

- string Description `[get, set]`

  A descriptive notation describing the hardware attached to the point. (Examples: "Overhead Light", "Door 3", "Motor 1", "Pump 3").

- bool Online `[get, set]`

  Denotes the availability status of the point. True corresponds to ONLINE, false to OFFLINE.

- string Function `[get]`

  Indicates the unique descriptive notation of the point. (Example: "Digital Output 24").

- string Identifier `[get]`

  Indicates the unique point identification code as assigned by TIO. (Example: "DO_24").

6.9.1 Detailed Description

Base class for Counter.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 Talos::IO::Counter::Counter () `[protected]`

Create protected abstract Counter.

6.9.3 Member Function Documentation

6.9.3.1 abstract void Talos::IO::Counter::SetDataValue (int *data*) `[pure virtual]`

Sets the data value for the counter.

Parameters:

data The value to set for the appropriate limit.

6.9.3.2 abstract void Talos::IO::Counter::Configure (CounterMode *mode*, CounterIndexMode *index*, CounterIndexType *indexType*, CounterWidth *width*, CounterFlag *flag*) [pure virtual]

Configure the counter point.

Parameters:

mode CounterMode

index CounterIndexMode

indexType ConterIndexType

width CounterWidth

flag CounterFlag

6.9.3.3 abstract void Talos::IO::Counter::ResetCounter () [pure virtual]

Reset the abstract counter.

6.9.3.4 abstract void Talos::IO::Counter::ResetPulse () [pure virtual]

Reset the pulse trigger.

6.9.4 Property Documentation

6.9.4.1 CounterMode Talos::IO::Counter::Mode [get, set]

Describes the counter configuration. (Free, Range, ...).

6.9.4.2 CounterIndexMode Talos::IO::Counter::IndexMode [get, set]

Describes the index configuration. (Reset, Instant, ...).

6.9.4.3 CounterIndexType Talos::IO::Counter::IndexType [get, set]

Describes the index configuration. (Reset, Instant, ...).

6.9.4.4 CounterWidth Talos::IO::Counter::Width [get, set]

Describes the byte length used. (One, Two, Three, Four).

6.9.4.5 CounterFlag Talos::IO::Counter::Flag [get, set]

Describes which event to report on. (Index, Compare, ...).

6.9.4.6 abstract int Talos::IO::Counter::Value [get]

Returns the current value of the counter.

6.9.4.7 int Talos::IO::Point::Index [get, set, inherited]

The Talos assigned point ID number. This is relative to other Talos points of the same type.

6.9.4.8 string Talos::IO::Point::Connection [get, set, inherited]

Describes the connection type of the device.

6.9.4.9 string Talos::IO::Point::ConnectionData [get, set, inherited]

Describes the connection data for the connection type of the device.

6.9.4.10 IOType Talos::IO::Point::Type [get, set, inherited]

Describes the nature of the point. (Analog or Digital).

6.9.4.11 Direction Talos::IO::Point::Direction [get, set, inherited]

Describes the nature of the point. (Input or Output).

6.9.4.12 string Talos::IO::Point::Description [get, set, inherited]

A descriptive notation describing the hardware attached to the point. (Examples: "Overhead Light", "Door 3", "Motor 1", "Pump 3").

6.9.4.13 bool Talos::IO::Point::Online [get, set, inherited]

Denotes the availability status of the point. True corresponds to ONLINE, false to OFFLINE.

6.9.4.14 string Talos::IO::Point::Function [get, inherited]

Indicates the unique descriptive notation of the point. (Example: "Digital Output 24").

6.9.4.15 string Talos::IO::Point::Identifier [get, inherited]

Indicates the unique point identification code as assigned by TIO. (Example: "DO_24").

6.10 Talos::IO::DeviceManager Class Reference

A manager of all devices that Talos is going to be dealing with. This is where points are actually instantiated.

Public Member Functions

- void AddDevice (string type)

  Adds a device to the device manager.

- void RemoveDevice (Device device)

  Remove a device to the device manager.

- void LoadConfiguration ()

  Build the list of devices using the configuration file.

- void SaveConfiguration ()

  Save the current device list configuration. This will only persist if located on the in a persistance storage area. An example would be that a configuration file located in "\\nandflash\\TalosDeviceMap.xml" would persist but "\\Windows\\TalosDeviceMap.xml" would not.

Properties

- static DeviceManager Instance `[get]`

  This property represents the single available instance of the DeviceManager object (singleton).

- static ReadOnlyCollection< string > SupportedDevices `[get]`

  Returns a list of support device types.

- static string ConfigurationFile `[get]`

  Fall through the different storage locations to find the current configuration file. The storage location priority is:
  - *\Storage Card\TalosDeviceMap.xml*
  - *\Nandflash\TalosDeviceMap.xml*
  - *\Windows\TalosDeviceMap.xml.*

- ReadOnlyCollection< Device > Devices `[get]`

  This is a collection of all products currently being managed by the Talos framework. Reguardless of the actual hardware interface (Onboard, USB, Ethernet, etc.) it will be accessable here.

6.10.1 Detailed Description

A manager of all devices that Talos is going to be dealing with. This is where points are actually instantiated.

6.10.2 Member Function Documentation

6.10.2.1 void Talos::IO::DeviceManager::AddDevice (string *type*)

Adds a device to the device manager.

Parameters:

type The type of device. Check the DeviceManager.SupportedDevices list.

Exceptions:

ArgumentNullException The type parameter must be set

NotSupportedException The supplied type is not supported

6.10.2.2 void Talos::IO::DeviceManager::RemoveDevice (Device *device*)

Remove a device to the device manager.

Parameters:

device The device object to remove from the list

Exceptions:

ArgumentNullException Device parameter is null

ArgumentException Device not found in the Devices list

6.10.2.3 void Talos::IO::DeviceManager::LoadConfiguration ()

Build the list of devices using the configuration file.

Exceptions:

SystemException Failed to load the device configuration

See also:

ConfigurationFile

6.10.2.4 void Talos::IO::DeviceManager::SaveConfiguration ()

Save the current device list configuration. This will only persist if located on the in a persistance storage area. An example would be that a configuration file located in "\\nandflash\\TalosDeviceMap.xml" would persist but "\\Windows\\TalosDeviceMap.xml" would not.

See also:

ConfigurationFile

6.10.3 Property Documentation

6.10.3.1 DeviceManager Talos::IO::DeviceManager::Instance [static, get]

This property represents the single available instance of the DeviceManager object (singleton).

6.10.3.2 ReadOnlyCollection<string> Talos::IO::DeviceManager::SupportedDevices [static, get]

Returns a list of support device types.

6.10.3.3 string Talos::IO::DeviceManager::ConfigurationFile [static, get]

Fall through the different storage locations to find the current configuration file. The storage location priority is:

- \Storage Card\TalosDeviceMap.xml
- \Nandflash\TalosDeviceMap.xml
- \Windows\TalosDeviceMap.xml.

See also:

LoadConfiguration, SaveConfiguration

6.10.3.4 ReadOnlyCollection<Device> Talos::IO::DeviceManager::Devices [get]

This is a collection of all products currently being managed by the Talos framework. Reguardless of the actual hardware interface (Onboard, USB, Ethernet, etc.) it will be accessable here.

6.11 Talos::IO::DigitalInPoint Class Reference

Base class for DigitalInPoint.

Inherits Talos::IO::DigitalPoint.

Protected Member Functions

- DigitalInPoint ()

  Default constructor.

Properties

- int DebounceCount `[get, set]`

  Gets and sets an odd number of readings will be taken and averaged together to collect a valid input value. An even value will make the point go "Offline" (Online == false).

- int DebounceDelay `[get, set]`

  Gets and sets the value that represents the minimum amount of time delay between each input reading.

- override bool Value `[get, set]`

  Gets the values that indicates the binary value of the digital point. Possible values are Set (true) or Cleared (false).

- Polarity Polarity `[get, set]`

  Indicates the functionality of external hardware. This attribute modifies what the hardware will output when given a Value of Set or Cleared.

- int Index `[get, set]`

  The Talos assigned point ID number. This is relative to other Talos points of the same type.

- string Connection `[get, set]`

  Describes the connection type of the device.

- string ConnectionData `[get, set]`

  Describes the connection data for the connection type of the device.

- IOType Type `[get, set]`

  Describes the nature of the point. (Analog or Digital).

- Direction Direction `[get, set]`

  Describes the nature of the point. (Input or Output).

- string Description `[get, set]`

  A descriptive notation describing the hardware attached to the point. (Examples: "Overhead Light", "Door 3", "Motor 1", "Pump 3").

- bool Online `[get, set]`

  Denotes the availability status of the point. True corresponds to ONLINE, false to OFFLINE.

- string Function [get]

  Indicates the unique descriptive notation of the point. (Example: "Digital Output 24").

- string Identifier [get]

  Indicates the unique point identification code as assigned by TIO. (Example: "DO_24").

6.11.1 Detailed Description

Base class for DigitalInPoint.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 Talos::IO::DigitalInPoint::DigitalInPoint () [protected]

Default constructor. Sets the properties to the following values: Direction to Direction.Input, DebounceCount to 5, and DebounceDelay to 10.

6.11.3 Property Documentation

6.11.3.1 int Talos::IO::DigitalInPoint::DebounceCount [get, set]

Gets and sets an odd number of readings will be taken and averaged together to collect a valid input value. An even value will make the point go "Offline" (Online == false).

6.11.3.2 int Talos::IO::DigitalInPoint::DebounceDelay [get, set]

Gets and sets the value that represents the minimum amount of time delay between each input reading.

6.11.3.3 override bool Talos::IO::DigitalInPoint::Value [get, set]

Gets the values that indicates the binary value of the digital point. Possible values are Set (true) or Cleared (false).

Exceptions:

IOException Cannot access specified DigitalInPoint.

InvalidOperationException DigitalInPoints may not have a value assigned to them.

Reimplemented from Talos::IO::DigitalPoint.

6.11.3.4 Polarity Talos::IO::DigitalPoint::Polarity [get, set, inherited]

Indicates the functionality of external hardware. This attribute modifies what the hardware will output when given a Value of Set or Cleared.

6.11.3.5 int Talos::IO::Point::Index [get, set, inherited]

The Talos assigned point ID number. This is relative to other Talos points of the same type.

6.11.3.6 string Talos::IO::Point::Connection [get, set, inherited]

Describes the connection type of the device.

6.11.3.7 string Talos::IO::Point::ConnectionData [get, set, inherited]

Describes the connection data for the connection type of the device.

6.11.3.8 IOType Talos::IO::Point::Type [get, set, inherited]

Describes the nature of the point. (Analog or Digital).

6.11.3.9 Direction Talos::IO::Point::Direction [get, set, inherited]

Describes the nature of the point. (Input or Output).

6.11.3.10 string Talos::IO::Point::Description [get, set, inherited]

A descriptive notation describing the hardware attached to the point. (Examples: "Overhead Light", "Door 3", "Motor 1", "Pump 3").

6.11.3.11 bool Talos::IO::Point::Online [get, set, inherited]

Denotes the availability status of the point. True corresponds to ONLINE, false to OFFLINE.

6.11.3.12 string Talos::IO::Point::Function [get, inherited]

Indicates the unique descriptive notation of the point. (Example: "Digital Output 24").

6.11.3.13 string Talos::IO::Point::Identifier [get, inherited]

Indicates the unique point identification code as assigned by TIO. (Example: "DO_24").

6.12 Talos::IO::DigitalOutPoint Class Reference

Base class for a DigitalOutPoint.

Inherits Talos::IO::DigitalPoint.

Protected Member Functions

- DigitalOutPoint ()

  Default constructor.

- void SetInitial ()

  Sets the inital configuration of the point.

Properties

- Isolation Isolation `[get, set]`

  Isolation configures the operation of the digital output. The combination of this attribute and the polarity attributes are used to determine the effect of writting a Set or Cleared state to the digital output.

- Persistency Persistency `[get, set]`

  This attribute is used to determine if a digital output's value should be restored upon power-loss.

- DigitalDefault Default `[get, set]`

  The default attribute is the default state of a non-persisted digital output.

- override bool Value `[get, set]`

  Indicates the binary value of the digital point. Possible values are Set (true) or Cleared (false).

- Polarity Polarity `[get, set]`

  Indicates the functionality of external hardware. This attribute modifies what the hardware will output when given a Value of Set or Cleared.

- int Index `[get, set]`

  The Talos assigned point ID number. This is relative to other Talos points of the same type.

- string Connection `[get, set]`

  Describes the connection type of the device.

- string ConnectionData `[get, set]`

  Describes the connection data for the connection type of the device.

- IOType Type `[get, set]`

  Describes the nature of the point. (Analog or Digital).

- Direction Direction `[get, set]`

  Describes the nature of the point. (Input or Output).

- string Description `[get, set]`

A descriptive notation describing the hardware attached to the point. (Examples: "Overhead Light", "Door 3", "Motor 1", "Pump 3").

- bool Online `[get, set]`

  Denotes the availability status of the point. True corresponds to ONLINE, false to OFFLINE.

- string Function `[get]`

  Indicates the unique descriptive notation of the point. (Example: "Digital Output 24").

- string Identifier `[get]`

  Indicates the unique point identification code as assigned by TIO. (Example: "DO_24").

6.12.1 Detailed Description

Base class for a DigitalOutPoint.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 Talos::IO::DigitalOutPoint::DigitalOutPoint () `[protected]`

Default constructor. Sets the properties to the following values: Direction to Direction.Output, Isolation to Isolation.Invalid, Persistency to Persistency.Invalid, and Default to DigitalDefault.Invalid.

6.12.3 Member Function Documentation

6.12.3.1 void Talos::IO::DigitalOutPoint::SetInitial () `[protected]`

Sets the inital configuration of the point.

Exceptions:

IOException Invalid configuration information detected!

6.12.4 Property Documentation

6.12.4.1 Isolation Talos::IO::DigitalOutPoint::Isolation `[get, set]`

Isolation configures the operation of the digital output. The combination of this attribute and the polarity attributes are used to determine the effect of writting a Set or Cleared state to the digital output.

6.12.4.2 Persistency Talos::IO::DigitalOutPoint::Persistency `[get, set]`

This attribute is used to determine if a digital output's value should be restored upon power-loss.

6.12.4.3 DigitalDefault Talos::IO::DigitalOutPoint::Default `[get, set]`

The default attribute is the default state of a non-persisted digital output.

6.12.4.4 override bool Talos::IO::DigitalOutPoint::Value [get, set]

Indicates the binary value of the digital point. Possible values are Set (true) or Cleared (false).

Exceptions:

IOException Cannot access specified DigitalOutPoint.

Reimplemented from Talos::IO::DigitalPoint.

6.12.4.5 Polarity Talos::IO::DigitalPoint::Polarity [get, set, inherited]

Indicates the functionality of external hardware. This attribute modifies what the hardware will output when given a Value of Set or Cleared.

6.12.4.6 int Talos::IO::Point::Index [get, set, inherited]

The Talos assigned point ID number. This is relative to other Talos points of the same type.

6.12.4.7 string Talos::IO::Point::Connection [get, set, inherited]

Describes the connection type of the device.

6.12.4.8 string Talos::IO::Point::ConnectionData [get, set, inherited]

Describes the connection data for the connection type of the device.

6.12.4.9 IOType Talos::IO::Point::Type [get, set, inherited]

Describes the nature of the point. (Analog or Digital).

6.12.4.10 Direction Talos::IO::Point::Direction [get, set, inherited]

Describes the nature of the point. (Input or Output).

6.12.4.11 string Talos::IO::Point::Description [get, set, inherited]

A descriptive notation describing the hardware attached to the point. (Examples: "Overhead Light", "Door 3", "Motor 1", "Pump 3").

6.12.4.12 bool Talos::IO::Point::Online [get, set, inherited]

Denotes the availability status of the point. True corresponds to ONLINE, false to OFFLINE.

6.12.4.13 string Talos::IO::Point::Function [get, inherited]

Indicates the unique descriptive notation of the point. (Example: "Digital Output 24").

6.12.4.14 string Talos::IO::Point::Identifier [get, inherited]

Indicates the unique point identification code as assigned by TIO. (Example: "DO_24").

6.13 Talos::IO::DigitalPoint Class Reference

Base class for a DigitalPoint.

Inherits Talos::IO::Point.

Inherited by Talos::IO::DigitalInPoint, and Talos::IO::DigitalOutPoint.

Protected Member Functions

- DigitalPoint ()

  Default constructor.

Properties

- virtual bool Value `[get, set]`

  Indicates the binary value of the digital point. Possible values are Set (true) or Cleared (false).

- Polarity Polarity `[get, set]`

  Indicates the functionality of external hardware. This attribute modifies what the hardware will output when given a Value of Set or Cleared.

- int Index `[get, set]`

  The Talos assigned point ID number. This is relative to other Talos points of the same type.

- string Connection `[get, set]`

  Describes the connection type of the device.

- string ConnectionData `[get, set]`

  Describes the connection data for the connection type of the device.

- IOType Type `[get, set]`

  Describes the nature of the point. (Analog or Digital).

- Direction Direction `[get, set]`

  Describes the nature of the point. (Input or Output).

- string Description `[get, set]`

  A descriptive notation describing the hardware attached to the point. (Examples: "Overhead Light", "Door 3", "Motor 1", "Pump 3").

- bool Online `[get, set]`

  Denotes the availability status of the point. True corresponds to ONLINE, false to OFFLINE.

- string Function `[get]`

  Indicates the unique descriptive notation of the point. (Example: "Digital Output 24").

- string Identifier `[get]`

  Indicates the unique point identification code as assigned by TIO. (Example: "DO_24").

6.13.1 Detailed Description

Base class for a DigitalPoint.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 Talos::IO::DigitalPoint::DigitalPoint () [protected]

Default constructor. Sets the properties to the following values: Type to IOType.Digital and Polarity to Polarity.Invalid.

6.13.3 Property Documentation

6.13.3.1 virtual bool Talos::IO::DigitalPoint::Value [get, set]

Indicates the binary value of the digital point. Possible values are Set (true) or Cleared (false).

Reimplemented in Talos::IO::DigitalInPoint, and Talos::IO::DigitalOutPoint.

6.13.3.2 Polarity Talos::IO::DigitalPoint::Polarity [get, set]

Indicates the functionality of external hardware. This attribute modifies what the hardware will output when given a Value of Set or Cleared.

6.13.3.3 int Talos::IO::Point::Index [get, set, inherited]

The Talos assigned point ID number. This is relative to other Talos points of the same type.

6.13.3.4 string Talos::IO::Point::Connection [get, set, inherited]

Describes the connection type of the device.

6.13.3.5 string Talos::IO::Point::ConnectionData [get, set, inherited]

Describes the connection data for the connection type of the device.

6.13.3.6 IOType Talos::IO::Point::Type [get, set, inherited]

Describes the nature of the point. (Analog or Digital).

6.13.3.7 Direction Talos::IO::Point::Direction [get, set, inherited]

Describes the nature of the point. (Input or Output).

6.13.3.8 string Talos::IO::Point::Description [get, set, inherited]

A descriptive notation describing the hardware attached to the point. (Examples: "Overhead Light", "Door 3", "Motor 1", "Pump 3").

6.13.3.9 bool Talos::IO::Point::Online [get, set, inherited]

Denotes the availability status of the point. True corresponds to ONLINE, false to OFFLINE.

6.13.3.10 string Talos::IO::Point::Function [get, inherited]

Indicates the unique descriptive notation of the point. (Example: "Digital Output 24").

6.13.3.11 string Talos::IO::Point::Identifier [get, inherited]

Indicates the unique point identification code as assigned by TIO. (Example: "DO_24").

6.14 Talos::IO::FileStream Class Reference

Exposes a Stream around a file.

Inherits Stream.

Inherited by Talos::IO::SerialPort.

Public Member Functions

- FileStream (string location)
  Default constructor.
- override void Close ()
  Close the current stream.
- override void Flush ()
  This method is not supported. Will throw a NotSupportedException.
- virtual void Open ()
  Attempts to open the file at the current Location.
- override void SetLength (long value)
  This method is not supported. Will throw a NotSupportedException.
- override unsafe void Write (byte[ ] buffer, int offset, int length)
  Writes a specified number of bytes to the serial port using data from a buffer.
- override unsafe int Read (byte[ ] buffer, int offset, int length)
  Reads a block of bytes from the stream and writes the data in a given buffer.
- override long Seek (long offset, SeekOrigin origin)
  This method is not supported. Will throw a NotSupportedException.

Properties

- override bool CanRead `[get]`
  This parameter is not support. Any attempt to access will result in a NotSupportedException.
- override bool CanSeek `[get]`
  This parameter is not support. Any attempt to access will result in a NotSupportedException.
- override bool CanWrite `[get]`
  This parameter is not support. Any attempt to access will result in a NotSupportedException.
- int Handle `[get, set]`
  Returns the current handle of the FileStream.
- bool IsOpen `[get]`

Gets the current value indicating if a file or resource is opened.

- override long Length `[get]`

  This parameter is not support. Any attempt to access will result in a NotSupportedException.

- string Location `[get, set]`

  Location of the file or resource to be opened.

- override long Position `[get, set]`

  This parameter is not support. Any attempt to access will result in a NotSupportedException.

6.14.1 Detailed Description

Exposes a Stream around a file.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 Talos::IO::FileStream::FileStream (string *location*)

Default constructor.

Parameters:

location Location of the file or resource.

Sets the properties to the following values: Handle to -1, Location to the provide parameter.

6.14.3 Member Function Documentation

6.14.3.1 override void Talos::IO::FileStream::Close ()

Close the current stream.

Exceptions:

InvalidOperationException The specified port is not open.

See also:

Open

6.14.3.2 override void Talos::IO::FileStream::Flush ()

This method is not supported. Will throw a NotSupportedException.

Exceptions:

NotSupportedException This method is not supported by the FileStream.

Reimplemented in Talos::IO::SerialPort.

6.14.3.3 virtual void Talos::IO::FileStream::Open () [virtual]

Attempts to open the file at the current Location.

See also:

Close

Reimplemented in Talos::IO::SerialPort.

6.14.3.4 override void Talos::IO::FileStream::SetLength (long *value*)

This method is not supported. Will throw a NotSupportedException.

Exceptions:

NotSupportedException This method is not supported by the FileStream.

Reimplemented in Talos::IO::SerialPort.

6.14.3.5 override unsafe void Talos::IO::FileStream::Write (byte[ ] *buffer*, int *offset*, int *length*)

Writes a specified number of bytes to the serial port using data from a buffer.

Parameters:

buffer The byte array that contains the data to write to the port.

offset The zero-based byte offset in the buffer parameter at which to begin copying bytes to the port.

length The number of bytes to write.

Returns:

The number of bytes written.

Exceptions:

InvalidOperationException The specified port is not open.

ArgumentNullException The buffer passed is a null reference.

ArgumentOutOfRangeException The offset is outside a valid region of the buffer. -OR- The length is greater than the size of the buffer minus the offset.

Reimplemented in Talos::IO::SerialPort.

6.14.3.6 override unsafe int Talos::IO::FileStream::Read (byte[ ] *buffer*, int *offset*, int *length*)

Reads a block of bytes from the stream and writes the data in a given buffer.

Parameters:

buffer When this method returns, contains the specified byte array with the values between offset and (offset + length - 1) replaced by the bytes read from the current source.

offset The byte offset in array at which the read bytes will be placed.

length The maximum number of bytes to read.

Returns:

The total number of bytes read into the buffer. This might be less than the number of bytes requested if that number of bytes are not currently available, or zero if the end of the stream is reached.

Exceptions:

InvalidOperationException The specified port is not open.

ArgumentNullException The buffer passed is a null reference.

ArgumentOutOfRangeException The offset is outside a valid region of the buffer. -OR- The length is greater than the size of the buffer minus the offset.

6.14.3.7 override long Talos::IO::FileStream::Seek (long *offset*, SeekOrigin *origin*)

This method is not supported. Will throw a NotSupportedException.

Parameters:

offset This param is not used

origin This param is not used

Exceptions:

NotSupportedException This method is not supported by the FileStream.

6.14.4 Property Documentation

6.14.4.1 override bool Talos::IO::FileStream::CanRead [get]

This parameter is not support. Any attempt to access will result in a NotSupportedException.

Exceptions:

NotSupportedException This property is not supported.

Reimplemented in Talos::IO::SerialPort.

6.14.4.2 override bool Talos::IO::FileStream::CanSeek [get]

This parameter is not support. Any attempt to access will result in a NotSupportedException.

Exceptions:

NotSupportedException This property is not supported.

Reimplemented in Talos::IO::SerialPort.

6.14.4.3 override bool Talos::IO::FileStream::CanWrite [get]

This parameter is not support. Any attempt to access will result in a NotSupportedException.

Exceptions:

NotSupportedException This property is not supported.

Reimplemented in Talos::IO::SerialPort.

6.14.4.4 int Talos::IO::FileStream::Handle [get, set]

Returns the current handle of the FileStream.

6.14.4.5 bool Talos::IO::FileStream::IsOpen [get]

Gets the current value indicating if a file or resource is opened.

6.14.4.6 override long Talos::IO::FileStream::Length [get]

This parameter is not support. Any attempt to access will result in a NotSupportedException.

Exceptions:

NotSupportedException This property is not supported.

Reimplemented in Talos::IO::SerialPort.

6.14.4.7 string Talos::IO::FileStream::Location [get, set]

Location of the file or resource to be opened.

See also:

Open

6.14.4.8 override long Talos::IO::FileStream::Position [get, set]

This parameter is not support. Any attempt to access will result in a NotSupportedException.

Exceptions:

NotSupportedException This property is not supported.

6.15 Talos::IO::IOManager Class Reference

The IOManager class handles all the IO available on the system.

Public Member Functions

- void Reconfigure ()

  Save the current point configurations for all points.

Properties

- static IOManager Instance `[get]`

  This property represents the single available instance of the IOManager object.

- ReadOnlyCollection< DigitalInPoint > DigitalInPoints `[get]`

  This is a collection of all available Digital Input Points. Regardless of the actual Point's hardware interface (Onboard GPIO, USB IO, Ethernet IO, etc.) it can be modified using the same standard DigitalInPoint type.

- ReadOnlyCollection< DigitalOutPoint > DigitalOutPoints `[get]`

  This is a collection of all available Digital Output Points. Regardless of the Point's actual hardware interface (Onboard GPIO, USB IO, Ethernet IO, etc.) it can be modified using the same standard DigitalOutPoint type.

- ReadOnlyCollection< AnalogInPoint > AnalogInPoints `[get]`

  This is a collection of all available Analog Input Points. Regardless of the Point's actual hardware interface (Onboard GPIO, USB IO, Ethernet IO, etc.) it can be modified using the same standard AnalogInPoint type.

- ReadOnlyCollection< AnalogOutPoint > AnalogOutPoints `[get]`

  This is a collection of all available Analog Output Points. Regardless of the Point's actual hardware interface (Onboard GPIO, USB IO, Ethernet IO, etc.) it can be modified using the same standard AnalogOutPoint type.

- ReadOnlyCollection< SerialPoint > SerialPoints `[get]`

  This is a collection of all available Serial Points.

- ReadOnlyCollection< Counter > Counters `[get]`

  This is a collection of all available counters.

- ReadOnlyCollection< CanPoint > CanPoints `[get]`

  This is a collection of all available counters.

6.15.1 Detailed Description

The IOManager class handles all the IO available on the system.

6.15.2 Member Function Documentation

6.15.2.1 void Talos::IO::IOManager::Reconfigure ()

Save the current point configurations for all points. This method will reconfigure the IOManager with the specific device configuration located in the DeviceManager.

6.15.3 Property Documentation

6.15.3.1 IOManager Talos::IO::IOManager::Instance [static, get]

This property represents the single available instance of the IOManager object.

6.15.3.2 ReadOnlyCollection<DigitalInPoint> Talos::IO::IOManager::DigitalInPoints [get]

This is a collection of all available Digital Input Points. Regardless of the actual Point's hardware interface (Onboard GPIO, USB IO, Ethernet IO, etc.) it can be modified using the same standard DigitalInPoint type.

6.15.3.3 ReadOnlyCollection<DigitalOutPoint> Talos::IO::IOManager::DigitalOutPoints [get]

This is a collection of all available Digital Output Points. Regardless of the Point's actual hardware interface (Onboard GPIO, USB IO, Ethernet IO, etc.) it can be modified using the same standard DigitalOutPoint type.

6.15.3.4 ReadOnlyCollection<AnalogInPoint> Talos::IO::IOManager::AnalogInPoints [get]

This is a collection of all available Analog Input Points. Regardless of the Point's actual hardware interface (Onboard GPIO, USB IO, Ethernet IO, etc.) it can be modified using the same standard AnalogInPoint type.

6.15.3.5 ReadOnlyCollection<AnalogOutPoint> Talos::IO::IOManager::AnalogOutPoints [get]

This is a collection of all available Analog Output Points. Regardless of the Point's actual hardware interface (Onboard GPIO, USB IO, Ethernet IO, etc.) it can be modified using the same standard AnalogOutPoint type.

6.15.3.6 ReadOnlyCollection<SerialPoint> Talos::IO::IOManager::SerialPoints [get]

This is a collection of all available Serial Points.

6.15.3.7 ReadOnlyCollection<Counter> Talos::IO::IOManager::Counters [get]

This is a collection of all available counters.

6.15.3.8 ReadOnlyCollection<CanPoint> Talos::IO::IOManager::CanPoints [get]

This is a collection of all available counters.

6.16 Talos::IO::Point Class Reference

Base class for a Point.

Inherited by Talos::IO::AnalogPoint, Talos::IO::CanPoint, Talos::IO::Counter, Talos::IO::DigitalPoint, and Talos::IO::SerialPoint.

Protected Member Functions

- Point ()

  Default constructor.

Properties

- int Index `[get, set]`

  The Talos assigned point ID number. This is relative to other Talos points of the same type.

- string Connection `[get, set]`

  Describes the connection type of the device.

- string ConnectionData `[get, set]`

  Describes the connection data for the connection type of the device.

- IOType Type `[get, set]`

  Describes the nature of the point. (Analog or Digital).

- Direction Direction `[get, set]`

  Describes the nature of the point. (Input or Output).

- string Description `[get, set]`

  A descriptive notation describing the hardware attached to the point. (Examples: "Overhead Light", "Door 3", "Motor 1", "Pump 3").

- bool Online `[get, set]`

  Denotes the availability status of the point. True corresponds to ONLINE, false to OFFLINE.

- string Function `[get]`

  Indicates the unique descriptive notation of the point. (Example: "Digital Output 24").

- string Identifier `[get]`

  Indicates the unique point identification code as assigned by TIO. (Example: "DO_24").

6.16.1 Detailed Description

Base class for a Point.

6.16.2 Constructor & Destructor Documentation

6.16.2.1 Talos::IO::Point::Point () [protected]

Default constructor. Sets the properties to the following values: Index to 0, Connection to string.Empty, ConnectionData to string.Empty, Type to IOType to IOType.Unknown, Direction to Direction.Invalid, Description to string.Empty, and Online to false.

6.16.3 Property Documentation

6.16.3.1 int Talos::IO::Point::Index [get, set]

The Talos assigned point ID number. This is relative to other Talos points of the same type.

6.16.3.2 string Talos::IO::Point::Connection [get, set]

Describes the connection type of the device.

6.16.3.3 string Talos::IO::Point::ConnectionData [get, set]

Describes the connection data for the connection type of the device.

6.16.3.4 IOType Talos::IO::Point::Type [get, set]

Describes the nature of the point. (Analog or Digital).

6.16.3.5 Direction Talos::IO::Point::Direction [get, set]

Describes the nature of the point. (Input or Output).

6.16.3.6 string Talos::IO::Point::Description [get, set]

A descriptive notation describing the hardware attached to the point. (Examples: "Overhead Light", "Door 3", "Motor 1", "Pump 3").

6.16.3.7 bool Talos::IO::Point::Online [get, set]

Denotes the availability status of the point. True corresponds to ONLINE, false to OFFLINE.

6.16.3.8 string Talos::IO::Point::Function [get]

Indicates the unique descriptive notation of the point. (Example: "Digital Output 24").

6.16.3.9 string Talos::IO::Point::Identifier [get]

Indicates the unique point identification code as assigned by TIO. (Example: "DO_24").

6.17 Talos::IO::SerialPort Class Reference

SerialPort allows for the interaction with COM devices.

Inherits Talos::IO::FileStream.

Public Member Functions

- SerialPort ()

  Initializes a new instance of the SerialPort class.

- SerialPort (string portName)

  Initializes a new instance of the SerialPort class using the specified port name.

- SerialPort (string portName, int baudRate, Parity parity, int dataBits, StopBits stopBits)

  Initializes a new instance of the SerialPort class using the specified port name, baud rate, parity bit, data bits, and stop bit.

- override void Open ()

  Opens a new serial port connection.

- string ReadLine ()

  Read a data from the serial port until we read a NewLine "\n" or Carriage Return characer "\r".

- string ReadExisting ()

  Reads all immediately available bytes, based on the encoding, in both the stream and the input buffer of the SerialPort object.

- void Write (byte[ ] buffer)

  Writes data to the serial port output buffer.

- override void Write (byte[ ] buffer, int offset, int length)

  Write data to the serial port output buffer.

- override void WriteByte (byte value)

  Write a byte of data to the serial port output buffer.

- override void Flush ()

  This method calls DiscardInBuffer then DiscardOutBuffer.

- override void SetLength (long value)

  This method is not supported. Will throw a NotSupportedException.

- void DiscardInBuffer ()

  Discards data from the serial driver's receive buffer.

- void DiscardOutBuffer ()

  Discards data from the serial driver's transmit buffer.

- override void Close ()

Close the current stream.

- override unsafe int Read (byte[ ] buffer, int offset, int length)

  Reads a block of bytes from the stream and writes the data in a given buffer.

- override long Seek (long offset, SeekOrigin origin)

  This method is not supported. Will throw a NotSupportedException.

Static Public Member Functions

- static string[ ] GetPortNames ()

  Returns an array of serial port names for the current computer.

Properties

- bool AutoRts `[get, set]`

  Gets and sets the automatic toggling of the RTS pin on transmit.

- bool AutoDtr `[get, set]`

  Gets and sets the automatic toggling of the DTR pin on transmit.

- string PortName `[get, set]`

  Gets and sets the communications port name. Ex. COM1, COM3.

- int BaudRate `[get, set]`

  Gets and sets the baud rate.

- int DataBits `[get, set]`

  Get and set the standard length of data bits per byte.

- Parity Parity `[get, set]`

  Gets or sets the parity-checking protocol.

- StopBits StopBits `[get, set]`

  Gets or sets the standard number of stopbits per byte.

- int ReadTimeoutInterval `[get, set]`

  Gets and set the maximum time allowed to elapse between the arrival of two bytes on the communications line, in milliseconds.

- int ReadTimeoutConstant `[get, set]`

  Gets and sets a constant used to calculate the total time-out period for read operations, in milliseconds.

- int ReadTimeoutMultiplier `[get, set]`

  Gets and sets the multiplier used to calculate the total time-out period for read operations, in milliseconds.

- override int ReadTimeout `[get, set]`

The number of milliseconds before a time-out occurs when a read operation does not finish.

- override int WriteTimeout `[get, set]`

  Gets or sets the number of milliseconds before a time-out occurs when a write operation does not finish.

- int BytesToRead `[get]`

  Gets the number of bytes of data in the receive buffer.

- int BytesToWrite `[get]`

  Gets the number of bytes of data in the send buffer.

- bool Rts `[get, set]`

  Gets or sets the current values of the Request to Send (RTS) signal control line.

- bool Cts `[get]`

  Gets the current value of the Clear to Send (CTS) signal line.

- bool Ring `[get]`

  Gets the current value of the Ring Indicator (RI) signal line.

- bool Dtr `[get, set]`

  Gets or sets the current values of the Data Terminal Ready (DTR) signal control line.

- bool Dsr `[get]`

  Gets the current value of the Data Set Ready (DSR) signal control line.

- bool Dcd `[get]`

  Gets the current value of the Data Carrier Detect (DCD) signal control line.

- override bool CanRead `[get]`

  This parameter is not support. Any attempt to access will result in a NotSupportedException.

- override bool CanSeek `[get]`

  This parameter is not support. Any attempt to access will result in a NotSupportedException.

- override bool CanWrite `[get]`

  This parameter is not support. Any attempt to access will result in a NotSupportedException.

- override long Length `[get]`

  This parameter is not support. Any attempt to access will result in a NotSupportedException.

- int Handle `[get, set]`

  Returns the current handle of the FileStream.

- bool IsOpen `[get]`

  Gets the current value indicating if a file or resource is opened.

- string Location `[get, set]`

  Location of the file or resource to be opened.

- override long Position `[get, set]`

  This parameter is not support. Any attempt to access will result in a NotSupportedException.

6.17.1 Detailed Description

SerialPort allows for the interaction with COM devices. Use this class to control a serial port file resource. This class provides synchronous and event-driven I/O, access to pin and break states, and access to serial driver properties. Additionally, the functionality of this class can be wrapped in an internal Stream object, accessible through the BaseStream property, and passed to classes that wrap or use streams.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 Talos::IO::SerialPort::SerialPort ()

Initializes a new instance of the SerialPort class. This constructor uses default property values when none are specified. For example, the DataBits property defaults to 8, the Parity property defaults to the None enumeration value, the StopBits property defaults to 1, and a default port name of COM1.

6.17.2.2 Talos::IO::SerialPort::SerialPort (string *portName*)

Initializes a new instance of the SerialPort class using the specified port name.

Parameters:

portName The port to use (for example, COM1)

Use this constructor to create a new instance of the SerialPort class when you want to specify the port name.

6.17.2.3 Talos::IO::SerialPort::SerialPort (string *portName*, int *baudRate*, Parity *parity*, int *dataBits*, StopBits *stopBits*)

Initializes a new instance of the SerialPort class using the specified port name, baud rate, parity bit, data bits, and stop bit.

Parameters:

portName The port to use (for example, COM1)

baudRate The baud rate value

parity One of the Parity values

dataBits The data bits value

stopBits One of the StopBits values

Use this constructor to create a new instance of the SerialPort class when you want to specify the port name, the baud rate, the parity bit, data bits, and stop bit.

6.17.3 Member Function Documentation

6.17.3.1 override void Talos::IO::SerialPort::Open () [virtual]

Opens a new serial port connection.

Exceptions:

InvalidOperationException The specified port is not open.

See also:

FileStream.Close

Reimplemented from Talos::IO::FileStream.

6.17.3.2 string Talos::IO::SerialPort::ReadLine ()

Read a data from the serial port until we read a NewLine "\n" or Carriage Return characer "\r".

Returns:

The contents of the input buffer up to the first occurrence of a NewLine value.

Exceptions:

InvalidOperationException The specified port is not open.

See also:

FileStream.Read

6.17.3.3 string Talos::IO::SerialPort::ReadExisting ()

Reads all immediately available bytes, based on the encoding, in both the stream and the input buffer of the SerialPort object.

Returns:

The contents of the stream and the input buffer of the SerialPort object.

See also:

FileStream.Read

6.17.3.4 void Talos::IO::SerialPort::Write (byte[ ] *buffer*)

Writes data to the serial port output buffer.

Parameters:

buffer The byte array that contains the data to write to the port.

Exceptions:

InvalidOperationException The specified port is not open.

ArgumentNullException The buffer passed is a null reference.

6.17.3.5 override void Talos::IO::SerialPort::Write (byte[ ] *buffer*, int *offset*, int *length*)

Write data to the serial port output buffer.

Parameters:

buffer The byte array that contains the data to write to the port.

offset The zero-based byte offset in the buffer parameter at which to begin copying bytes to the port.

length The number of bytes to write.

Exceptions:

InvalidOperationException The specified port is not open.

ArgumentNullException The buffer passed is a null reference.

ArgumentOutOfRangeException The offset is outside a valid region of the buffer. -OR- The length is greater than the size of the buffer minus the offset.

Reimplemented from Talos::IO::FileStream.

6.17.3.6 override void Talos::IO::SerialPort::WriteByte (byte *value*)

Write a byte of data to the serial port output buffer.

Parameters:

value The data to write to the port.

6.17.3.7 override void Talos::IO::SerialPort::Flush ()

This method calls DiscardInBuffer then DiscardOutBuffer.

See also:

DiscardInBuffer, DiscardOutBuffer

Reimplemented from Talos::IO::FileStream.

6.17.3.8 override void Talos::IO::SerialPort::SetLength (long *value*)

This method is not supported. Will throw a NotSupportedException.

Exceptions:

NotSupportedException This method is not supported by the FileStream.

Reimplemented from Talos::IO::FileStream.

6.17.3.9 void Talos::IO::SerialPort::DiscardInBuffer ()

Discards data from the serial driver's receive buffer.

See also:

DiscardOutBuffer

6.17.3.10 void Talos::IO::SerialPort::DiscardOutBuffer ()

Discards data from the serial driver's transmit buffer.

See also:

DiscardInBuffer

6.17.3.11 static string [ ] Talos::IO::SerialPort::GetPortNames () [static]

Returns an array of serial port names for the current computer. The order of port names returned from GetPortNames is not specified. Use the GetPortNames method to query the current computer for a list of valid serial port names. For example, you can use this method to determine whether COM1 and COM2 are valid serial ports for the current computer. The port names are obtained from the system registry (for example, in Windows 98 environments this information resides in \HKEY_LOCAL_MACHINE\HARDWARE\DEVICEMAP\SERIALCOMM). If the registry contains stale or otherwise incorrect data then the GetPortNames method will return incorrect data.

Returns:

An array of serial port names for the current computer.

6.17.3.12 override void Talos::IO::FileStream::Close () [inherited]

Close the current stream.

Exceptions:

InvalidOperationException The specified port is not open.

See also:

Open

6.17.3.13 override unsafe int Talos::IO::FileStream::Read (byte[ ] *buffer*, int *offset*, int *length*) [inherited]

Reads a block of bytes from the stream and writes the data in a given buffer.

Parameters:

buffer When this method returns, contains the specified byte array with the values between offset and (offset + length - 1) replaced by the bytes read from the current source.

offset The byte offset in array at which the read bytes will be placed.

length The maximum number of bytes to read.

Returns:

The total number of bytes read into the buffer. This might be less than the number of bytes requested if that number of bytes are not currently available, or zero if the end of the stream is reached.

Exceptions:

InvalidOperationException The specified port is not open.

ArgumentNullException The buffer passed is a null reference.

ArgumentOutOfRangeException The offset is outside a valid region of the buffer. -OR- The length is greater than the size of the buffer minus the offset.

6.17.3.14 override long Talos::IO::FileStream::Seek (long *offset*, SeekOrigin *origin*) [inherited]

This method is not supported. Will throw a NotSupportedException.

Parameters:

offset This param is not used

origin This param is not used

Exceptions:

NotSupportedException This method is not supported by the FileStream.

6.17.4 Property Documentation

6.17.4.1 bool Talos::IO::SerialPort::AutoRts [get, set]

Gets and sets the automatic toggling of the RTS pin on transmit.

See also:

AutoDtr

6.17.4.2 bool Talos::IO::SerialPort::AutoDtr [get, set]

Gets and sets the automatic toggling of the DTR pin on transmit.

See also:

AutoRts

6.17.4.3 string Talos::IO::SerialPort::PortName [get, set]

Gets and sets the communications port name. Ex. COM1, COM3. Get a list of valid port names can be obtained using the GetPortNames method.

See also:

GetPortNames

6.17.4.4 int Talos::IO::SerialPort::BaudRate [get, set]

Gets and sets the baud rate. The baud rate must be supported by the user's serial driver. The default value is 9600 bits per second (bps). Value must be greater than zero.

Exceptions:

Exception Invalid BaudRate

6.17.4.5 int Talos::IO::SerialPort::DataBits [get, set]

Get and set the standard length of data bits per byte. The range of values for this property is from 5 through 9. The default value is 8.

Exceptions:

ArgumentOutOfRangeException Value must be between 5 and 9

6.17.4.6 Parity Talos::IO::SerialPort::Parity [get, set]

Gets or sets the parity-checking protocol. One of the IO.Parity values that represents the parity-checking protocol.

Exceptions:

ArgumentOutOfRangeException Parity value is not valid.

See also:

IO.Parity

6.17.4.7 StopBits Talos::IO::SerialPort::StopBits [get, set]

Gets or sets the standard number of stopbits per byte.

See also:

IO.StopBits

6.17.4.8 int Talos::IO::SerialPort::ReadTimeoutInterval [get, set]

Gets and set the maximum time allowed to elapse between the arrival of two bytes on the communications line, in milliseconds. During a ReadFile operation on the full framework, the time period begins when the first byte is received. During a ReadFile operation on the compact framework, the time period begins immediately. If the interval between the arrival of any two bytes exceeds this amount, the ReadFile operation is completed and any buffered data is returned. A value of zero indicates that interval time-outs are not used.

Exceptions:

ArgumentOutOfRangeException The ReadTimeoutInterval value is less than zero and not equal to InfiniteTimeout.

See also:

ReadTimeoutConstant, ReadTimeoutMultiplier

6.17.4.9 int Talos::IO::SerialPort::ReadTimeoutConstant [get, set]

Gets and sets a constant used to calculate the total time-out period for read operations, in milliseconds. For each read operation, this value is added to the product of the ReadTimeoutMultiplier member and the requested number of bytes. A value of zero for both the ReadTimeoutMultiplier and ReadTimeoutConstant members indicates that total time-outs are not used for read operations.

Exceptions:

ArgumentOutOfRangeException The ReadTimeoutConstant value is less than zero and not equal to InfiniteTimeout

See also:

ReadTimeoutInterval, ReadTimeoutMultiplier

6.17.4.10 int Talos::IO::SerialPort::ReadTimeoutMultiplier [get, set]

Gets and sets the multiplier used to calculate the total time-out period for read operations, in milliseconds. For each read operation, this value is multiplied by the requested number of bytes to be read.

Exceptions:

ArgumentOutOfRangeException The ReadTimeoutMultiplier value is less than zero and not equal to InfiniteTimeout

See also:

ReadTimeoutConstant, ReadTimeoutInterval

6.17.4.11 override int Talos::IO::SerialPort::ReadTimeout [get, set]

The number of milliseconds before a time-out occurs when a read operation does not finish. The read time-out value was originally set at 500 milliseconds in the Win32 Communications API. This property allows you to set this value. The time-out can be set to any value greater than zero, or set to InfiniteTimeout, in which case no time-out occurs. InfiniteTimeout is the default.

Note:

Users of the unmanaged COMMTIMEOUTS structure might expect to set the time-out value to zero to suppress time-outs. To suppress time-outs with the ReadTimeout property, however, you must specify InfiniteTimeout.

Exceptions:

ArgumentOutOfRangeException The read time-out value is less than zero and not equal to InfiniteTimeout.

See also:

WriteTimeout

6.17.4.12 override int Talos::IO::SerialPort::WriteTimeout [get, set]

Gets or sets the number of milliseconds before a time-out occurs when a write operation does not finish. The write time-out value was originally set at 500 milliseconds in the Win32 Communications API. This property allows you to set this value. The time-out can be set to any value greater than zero, or set to InfiniteTimeout, in which case no time-out occurs. InfiniteTimeout is the default.

Note:

Users of the unmanaged COMMTIMEOUTS structure might expect to set the time-out value to zero to suppress time-outs. To suppress time-outs with the WriteTimeout property, however, you must specify InfiniteTimeout.

Exceptions:

ArgumentOutOfRangeException The WriteTimeout value is less than zero and not equal to InfiniteTimeout.

See also:

ReadTimeout

6.17.4.13 int Talos::IO::SerialPort::BytesToRead [get]

Gets the number of bytes of data in the receive buffer. The receive buffer includes the serial driver's receive buffer as well as internal buffering in the SerialPort object itself.

Note:

The BytesToRead property can return a value larger than the ReadBufferSize property because the ReadBufferSize property represents only the Windows-created buffer while the BytesToRead property represents the SerialPort buffer in addition to the Windows-created buffer.

6.17.4.14 int Talos::IO::SerialPort::BytesToWrite [get]

Gets the number of bytes of data in the send buffer. The send buffer includes the serial driver's send buffer as well as internal buffering in the SerialPort object itself.

6.17.4.15 bool Talos::IO::SerialPort::Rts [get, set]

Gets or sets the current values of the Request to Send (RTS) signal control line.

See also:

Cts, Ring, Dtr, Dsr, Dcd

6.17.4.16 bool Talos::IO::SerialPort::Cts [get]

Gets the current value of the Clear to Send (CTS) signal line.

See also:

Rts, Ring, Dtr, Dsr, Dcd

6.17.4.17 bool Talos::IO::SerialPort::Ring [get]

Gets the current value of the Ring Indicator (RI) signal line.

See also:

Rts, Cts, Dtr, Dsr, Dcd

6.17.4.18 bool Talos::IO::SerialPort::Dtr [get, set]

Gets or sets the current values of the Data Terminal Ready (DTR) signal control line.

See also:

Rts, Cts, Ring, Dsr, Dcd

6.17.4.19 bool Talos::IO::SerialPort::Dsr [get]

Gets the current value of the Data Set Ready (DSR) signal control line.

See also:

Rts, Cts, Ring, Dtr, Dcd

6.17.4.20 bool Talos::IO::SerialPort::Dcd [get]

Gets the current value of the Data Carrier Detect (DCD) signal control line.

See also:

Rts, Cts, Ring, Dtr, Dsr

6.17.4.21 override bool Talos::IO::SerialPort::CanRead [get]

This parameter is not support. Any attempt to access will result in a NotSupportedException.

Exceptions:

NotSupportedException This property is not supported.

Reimplemented from Talos::IO::FileStream.

6.17.4.22 override bool Talos::IO::SerialPort::CanSeek [get]

This parameter is not support. Any attempt to access will result in a NotSupportedException.

Exceptions:

NotSupportedException This property is not supported.

Reimplemented from Talos::IO::FileStream.

6.17.4.23 override bool Talos::IO::SerialPort::CanWrite [get]

This parameter is not support. Any attempt to access will result in a NotSupportedException.

Exceptions:

NotSupportedException This property is not supported.

Reimplemented from Talos::IO::FileStream.

6.17.4.24 override long Talos::IO::SerialPort::Length [get]

This parameter is not support. Any attempt to access will result in a NotSupportedException.

Exceptions:

NotSupportedException This property is not supported.

Reimplemented from Talos::IO::FileStream.

6.17.4.25 int Talos::IO::FileStream::Handle [get, set, inherited]

Returns the current handle of the FileStream.

6.17.4.26 bool Talos::IO::FileStream::IsOpen [get, inherited]

Gets the current value indicating if a file or resource is opened.

6.17.4.27 string Talos::IO::FileStream::Location [get, set, inherited]

Location of the file or resource to be opened.

See also:

Open

6.17.4.28 override long Talos::IO::FileStream::Position [get, set, inherited]

This parameter is not support. Any attempt to access will result in a NotSupportedException.

Exceptions:

NotSupportedException This property is not supported.

6.18 Talos::OwnerInformation Struct Reference

This structure is used to store Windows CE device ownership information.

Public Attributes

- string Name

  The owner of this device.

- string Company

  The company associated with the owner.

- string Address

  The address of the owner/company.

- string WorkCC

  The work phone extension.

- string WorkAC

  The work phone area code.

- string WorkPhone

  The work phone number.

- string HomeCC

  The home phone extension.

- string HomeAC

  The home phone area code.

- string HomePhone

  The home phone number.

- bool ShowOwnerOnPowerUp

  Set this flag to display owner information at startup (no effect on headless devices).

- string Notes

  Owner specified notes and other general information.

- bool ShowNotesOnPowerUp

  Set this flag to display the owner notes at startup (no effect on headless devices).

6.18.1 Detailed Description

This structure is used to store Windows CE device ownership information.

6.19 Talos::Protocols::ModbusClient Class Reference

A simple implementation of a basic Modbus client.

Public Member Functions

- ModbusClient (int slaveId, Stream stream)

  This Modbus client is associated with the Modbus slave at the specified ID number on the specified Stream. The stream may be of either type SerialPort or Socket to work with this class.

- byte[ ] ModbusPoll (byte[ ] message, int bytesToExpect)

  Format the specified message and send it across the proper communication channel. This method allows for polling of custom Modbus commands.

- bool[ ] ReadDiscreteInputs (int start, int quantity)

  Read digital input point values.

- int[ ] ReadInputRegisters (int start, int quantity)

  Read the specified input registers.

- bool[ ] ReadCoils (int start, int quantity)

  Read the states of specified digital output points.

- int[ ] ReadHoldingRegisters (int start, int quantity)

  Read the specified configuration holding registers.

- void WriteSingleCoil (int index, bool state)

  Write a single digital output point.

- void WriteMultipleCoils (int start, bool[ ] states)

  Write multiple digital output points.

- void WriteSingleRegister (int index, int value)

  Write a 16-bit integer value to the specified configuration holding register.

- void WriteMultipleRegisters (int start, int[ ] values)

  Write multiple 16-bit integer values to a contiguous block of configuration holding registers.

Properties

- int SlaveId `[get, set]`

  Gets or sets the slave module ID number for which this client is in communication.

- Stream Stream `[get, set]`

  Gets the stream object through which this ModbusClient will communicate.

- InterfaceType InterfaceType `[get]`

  Gets the currently active interface used for communications by this client.

6.19.1 Detailed Description

A simple implementation of a basic Modbus client. Modbus is a serial communications protocol published by Modicon in 1979 for use with its programmable logic controllers (PLCs). It has become a de facto standard communications protocol in industry, and is now the most commonly available means of connecting industrial electronic devices.

For serial connections, two variants exist, with different representations of numerical data and slightly different protocol details. Modbus RTU is a compact, binary representation of the data. Modbus ASCII is human readable, and more verbose. Both of these variants use serial communication. The RTU format follows the commands/data with a cyclic redundancy check checksum, while the ASCII format uses a longitudinal redundancy check checksum. Nodes configured for the RTU variant will not communicate with nodes set for ASCII, and the reverse.

This class currently supports the Modbus RTU variant of communications. In addition to the standard serial port communication medium, this class will also accept a TCP/IP Socket as the communication medium. When this type of medium is specified, encapsulated Modbus RTU messages will be created.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 Talos::Protocols::ModbusClient::ModbusClient (int *slaveId*, Stream *stream*)

This Modbus client is associated with the Modbus slave at the specified ID number on the specified Stream. The stream may be of either type SerialPort or Socket to work with this class.

Parameters:

slaveId The desired Modbus slave ID [1,247].

stream The fully prepared, and connected SerialPort or Socket.

Exceptions:

ArgumentOutOfRangeException The slaveId value must be in the range [1,247].

ArgumentNullException The stream value cannot be null.

6.19.3 Member Function Documentation

6.19.3.1 byte [ ] Talos::Protocols::ModbusClient::ModbusPoll (byte[ ] *message*, int *bytesToExpect*)

Format the specified message and send it across the proper communication channel. This method allows for polling of custom Modbus commands.

Parameters:

message The modbus command to send.

bytesToExpect The number of expected bytes in the response message.

Returns:

The array corresponding to the modbus response.

Exceptions:

Exception Interface is invalid or stream is not associated with a valid Serial Port or TCP Client. -OR- No response received. -OR- Response too short. Received: {0} Expected: {1}

ArgumentNullException The message array must contain at least one byte.
ModbusException The modbus request failed with an exception.

6.19.3.2 bool [ ] Talos::Protocols::ModbusClient::ReadDiscreteInputs (int *start*, int *quantity*)

Read digital input point values.

Parameters:

start A value indicating the start index.
quantity A value indicating the number of inputs to read simulataneously.

Returns:

An array of booleans indicating the current state of the inputs.

6.19.3.3 int [ ] Talos::Protocols::ModbusClient::ReadInputRegisters (int *start*, int *quantity*)

Read the specified input registers.

Parameters:

start A value indicating the start position.
quantity A value indicating the number of operations to perform.

Returns:

An array of integers representing the unsigned 32-bit values of each register.

6.19.3.4 bool [ ] Talos::Protocols::ModbusClient::ReadCoils (int *start*, int *quantity*)

Read the states of specified digital output points.

Parameters:

start A value indicating the start position.
quantity A value indicating the number of operations to perform.

Returns:

An array of booleans indicating the current state of the outputs.

6.19.3.5 int [ ] Talos::Protocols::ModbusClient::ReadHoldingRegisters (int *start*, int *quantity*)

Read the specified configuration holding registers.

Parameters:

start A value indicating the start position.
quantity A value indicating the number of operations to perform.

Returns:

An array of integer values representing the 16-bit values of each register.

6.19.3.6 void Talos::Protocols::ModbusClient::WriteSingleCoil (int *index*, bool *state*)

Write a single digital output point.

Parameters:

index A value indicating the output number.

state A boolean value indicating the desired state to set.

6.19.3.7 void Talos::Protocols::ModbusClient::WriteMultipleCoils (int *start*, bool[ ] *states*)

Write multiple digital output points.

Parameters:

start A value indicating the start position.

states An array of boolean values indicating the desired states to set.

6.19.3.8 void Talos::Protocols::ModbusClient::WriteSingleRegister (int *index*, int *value*)

Write a 16-bit integer value to the specified configuration holding register.

Parameters:

index A value indicating the register index to modify.

value The 16-bit value to write to the specified register.

6.19.3.9 void Talos::Protocols::ModbusClient::WriteMultipleRegisters (int *start*, int[ ] *values*)

Write multiple 16-bit integer values to a contiguous block of configuration holding registers.

Parameters:

start A value indicating the starting register.

values An array of 16-bit integer values indicating the desired values to set.

6.19.4 Property Documentation

6.19.4.1 int Talos::Protocols::ModbusClient::SlaveId [get, set]

Gets or sets the slave module ID number for which this client is in communication.

6.19.4.2 Stream Talos::Protocols::ModbusClient::Stream [get, set]

Gets the stream object through which this ModbusClient will communicate.

6.19.4.3 InterfaceType Talos::Protocols::ModbusClient::InterfaceType [get]

Gets the currently active interface used for communications by this client.

6.20 Talos::Protocols::ModbusException Class Reference

Represents errors that occur during Modbus communications.

Inherits Exception.

Public Member Functions

- ModbusException (byte exception)

  Represents a critical Modbus communication error.

- ModbusException (byte exception, string information)

  Represents a critical Modbus communication error.

- ModbusException (byte exception, string information, Exception chain)

  Represents a critical Modbus communication error.

Properties

- byte ExceptionCode `[get, set]`

  The byte value of the modbus exception that has occured.

6.20.1 Detailed Description

Represents errors that occur during Modbus communications.

6.20.2 Constructor & Destructor Documentation

6.20.2.1 Talos::Protocols::ModbusException::ModbusException (byte *exception*)

Represents a critical Modbus communication error.

Parameters:

exception The exception code.

6.20.2.2 Talos::Protocols::ModbusException::ModbusException (byte *exception*, string *information*)

Represents a critical Modbus communication error.

Parameters:

exception The exception code.

information An additional string containing information about the exception.

6.20.2.3 Talos::Protocols::ModbusException::ModbusException (byte *exception*, string *information*, Exception *chain*)

Represents a critical Modbus communication error.

Parameters:

exception The exception code.

information An additional string containing information about the exception.

chain The exception that orignated this exception.

6.20.3 Property Documentation

6.20.3.1 byte Talos::Protocols::ModbusException::ExceptionCode `[get, set]`

The byte value of the modbus exception that has occured.

6.21 Talos::Reflection::AssemblyInformation Class Reference

Allows for easier access to assembly information.

Public Member Functions

- AssemblyInformation (Assembly assembly)

  Constructor that allows the reference of a specific assembly.

- AssemblyInformation ()

  Default constructor that gets the current calling essembly.

Properties

- string Name `[get]`

  The Name of the Assembly associated with this AssembyInformation object.

- string FullName `[get]`

  The FullName of the Assembly associated with this AssemblyInformation object.

- string CodeBase `[get]`

  The CodeBase of the Assembly associated with this AssemblyInformation object.

- string Copyright `[get]`

  The Copyright of the Assembly associated with this AssemblyInformation object.

- string Company `[get]`

  The Company of the Assembly associated with this AssemblyInformation object.

- string Description `[get]`

  The Description of the Assembly associated with this AssemblyInformation object.

- string Product `[get]`

  The Product of the Assembly associated with this AssemblyInformation object.

- string Title `[get]`

  The Title of the Assembly associated with this AssemblyInformation object.

- Version Version `[get]`

  The Version of the Assembly associated with this AssemblyInformation object.

6.21.1 Detailed Description

Allows for easier access to assembly information.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 Talos::Reflection::AssemblyInformation::AssemblyInformation (Assembly *assembly*)

Constructor that allows the reference of a specific assembly.

Parameters:

assembly Assembly to gather information from.

6.21.2.2 Talos::Reflection::AssemblyInformation::AssemblyInformation ()

Default constructor that gets the current calling essembly.

6.21.3 Property Documentation

6.21.3.1 string Talos::Reflection::AssemblyInformation::Name [get]

The Name of the Assembly associated with this AssembyInformation object.

6.21.3.2 string Talos::Reflection::AssemblyInformation::FullName [get]

The FullName of the Assembly associated with this AssemblyInformation object.

6.21.3.3 string Talos::Reflection::AssemblyInformation::CodeBase [get]

The CodeBase of the Assembly associated with this AssemblyInformation object.

6.21.3.4 string Talos::Reflection::AssemblyInformation::Copyright [get]

The Copyright of the Assembly associated with this AssemblyInformation object.

6.21.3.5 string Talos::Reflection::AssemblyInformation::Company [get]

The Company of the Assembly associated with this AssemblyInformation object.

6.21.3.6 string Talos::Reflection::AssemblyInformation::Description [get]

The Description of the Assembly associated with this AssemblyInformation object.

6.21.3.7 string Talos::Reflection::AssemblyInformation::Product [get]

The Product of the Assembly associated with this AssemblyInformation object.

6.21.3.8 string Talos::Reflection::AssemblyInformation::Title [get]

The Title of the Assembly associated with this AssemblyInformation object.

6.21.3.9 Version Talos::Reflection::AssemblyInformation::Version [get]

The Version of the Assembly associated with this AssemblyInformation object.

6.22 Talos::Threading::BackgroundWorker Class Reference

The BackgroundWorker component gives you the ability to execute time-consuming operations asynchronously ("in the background"), on a thread different from your application's main UI thread.

Inherits System::ComponentModel::Component.

Public Member Functions

- void CancelAsync ()
  Requests cancellation of a pending background operation.
- void ReportProgress (int percentProgress)
  Raises the ProgressChanged event.
- void ReportProgress (int percentProgress, object userState)
  Raises the ProgressChanged event.
- void RunWorkerAsync ()
  Starts execution of a background operation.
- void RunWorkerAsync (object argument)
  Starts execution of a background operation.

Properties

- bool CancellationPending `[get, set]`
  Gets a value indicating whether the application has requested cancellation of a background operation.
- bool WorkerReportsProgress `[get, set]`
  Gets or sets a value indicating whether the BackgroundWorker can report progress updates.
- bool WorkerSupportsCancellation `[get, set]`
  Gets or sets a value indicating whether the BackgroundWorker supports asynchronous cancellation.
- bool IsBusy `[get]`
  Gets a value indicating whether the BackgroundWorker is running an asynchronous operation.

Events

- DoWorkEventHandler DoWork
  Occurs when RunWorkerAsync is called.
- ProgressChangedEventHandler ProgressChanged
  Occurs when ReportProgress is called.
- RunWorkerCompletedEventHandler RunWorkerCompleted
  Occurs when the background operation has completed, has been canceled, or has raised an exception.

6.22.1 Detailed Description

The BackgroundWorker component gives you the ability to execute time-consuming operations asynchronously ("in the background"), on a thread different from your application's main UI thread. To use a BackgroundWorker, you simply tell it what time-consuming worker method to execute in the background, and then you call the RunWorkerAsync method. Your calling thread continues to run normally while the worker method runs asynchronously.

Note:

You must be careful not to manipulate any user-interface objects in your DoWork event handler. Instead, communicate to the user interface through the ProgressChanged and RunWorkerCompleted events. BackgroundWorker events are not marshaled across AppDomain boundaries. Do not use a BackgroundWorker component to perform multithreaded operations in more than one AppDomain.

If your background operation requires a parameter, call RunWorkerAsync with your parameter. Inside the DoWork event handler, you can extract the parameter from the DoWorkEventArgs.Argument property.

6.22.2 Member Function Documentation

6.22.2.1 void Talos::Threading::BackgroundWorker::CancelAsync ()

Requests cancellation of a pending background operation.

Exceptions:

InvalidOperationException Does not support cancel. You must WorkerSupportsCancellation be set to true.

See also:

RunWorkerAsync(), RunWorkerAsync(object)

6.22.2.2 void Talos::Threading::BackgroundWorker::ReportProgress (int *percentProgress*)

Raises the ProgressChanged event.

Parameters:

percentProgress The percentage, from 0 to 100, of the background operation that is complete.

If you need the background operation to report on its progress, you can call the ReportProgress method to raise the ProgressChanged event. The WorkerReportsProgress property value must true, or ReportProgress will throw an InvalidOperationException.

It is up to you to implement a meaningful way of measuring your background operation's progress as a percentage of the total task completed.

6.22.2.3 void Talos::Threading::BackgroundWorker::ReportProgress (int *percentProgress*, object *userState*)

Raises the ProgressChanged event.

Parameters:

percentProgress The percentage, from 0 to 100, of the background operation that is complete.

userState The state object passed to RunWorkerAsync.

If you need the background operation to report on its progress, you can call the ReportProgress method to raise the ProgressChanged event. The WorkerReportsProgress property value must true, or ReportProgress will throw an InvalidOperationException.

It is up to you to implement a meaningful way of measuring your background operation's progress as a percentage of the total task completed.

6.22.2.4 void Talos::Threading::BackgroundWorker::RunWorkerAsync ()

Starts execution of a background operation.

See also:

IsBusy, DoWork, CancelAsync

6.22.2.5 void Talos::Threading::BackgroundWorker::RunWorkerAsync (object *argument*)

Starts execution of a background operation.

Parameters:

argument A parameter for use by the background operation to be executed in the DoWork event handler.

Exceptions:

InvalidOperationException The BackgroundWorker is already running. -OR- You must subscribe to the DoWork event.

See also:

IsBusy, DoWork, CancelAsync

6.22.3 Property Documentation

6.22.3.1 bool Talos::Threading::BackgroundWorker::CancellationPending [get, set]

Gets a value indicating whether the application has requested cancellation of a background operation.

6.22.3.2 bool Talos::Threading::BackgroundWorker::WorkerReportsProgress [get, set]

Gets or sets a value indicating whether the BackgroundWorker can report progress updates.

6.22.3.3 bool Talos::Threading::BackgroundWorker::WorkerSupportsCancellation [get, set]

Gets or sets a value indicating whether the BackgroundWorker supports asynchronous cancellation.

6.22.3.4 bool Talos::Threading::BackgroundWorker::IsBusy [get]

Gets a value indicating whether the BackgroundWorker is running an asynchronous operation.

6.22.4 Event Documentation

6.22.4.1 DoWorkEventHandler Talos::Threading::BackgroundWorker::DoWork

Occurs when RunWorkerAsync is called.

6.22.4.2 ProgressChangedEventHandler Talos::Threading::BackgroundWorker::ProgressChanged

Occurs when ReportProgress is called.

6.22.4.3 RunWorkerCompletedEventHandler Talos::Threading::BackgroundWorker::RunWorkerCompleted

Occurs when the background operation has completed, has been canceled, or has raised an exception.

6.23 Talos::Threading::DoWorkEventArgs Class Reference

Provides data for the DoWork event handler.

Inherits CancelEventArgs.

Public Member Functions

- DoWorkEventArgs (object argument)

  Initializes a new instance of the DoWorkEventArgs class.

Properties

- object Argument `[get, set]`

  Gets a value that represents the argument of an asynchronous operation.

- object Result `[get, set]`

  Gets or sets a value that represents the result of an asynchronous operation.

6.23.1 Detailed Description

Provides data for the DoWork event handler.

6.23.2 Constructor & Destructor Documentation

6.23.2.1 Talos::Threading::DoWorkEventArgs::DoWorkEventArgs (object *argument*)

Initializes a new instance of the DoWorkEventArgs class.

Parameters:

argument Specifies an argument for an asynchronous operation.

6.23.3 Property Documentation

6.23.3.1 object Talos::Threading::DoWorkEventArgs::Argument `[get, set]`

Gets a value that represents the argument of an asynchronous operation.

6.23.3.2 object Talos::Threading::DoWorkEventArgs::Result `[get, set]`

Gets or sets a value that represents the result of an asynchronous operation.

6.24 Talos::Threading::HighperformanceCounter Class Reference

This class provides an OOP wrapper around the existing Windows CE High Performance Counter API. The properties in this static class can be used to time actions with the highest possible accuracy.

Properties

- static long Frequency `[get]`

  Returns the frequency of the built-in High Performance counter in Hz (1/s).

- static long Count `[get]`

  Returns a 64-bit representation of the current count.

6.24.1 Detailed Description

This class provides an OOP wrapper around the existing Windows CE High Performance Counter API. The properties in this static class can be used to time actions with the highest possible accuracy.

6.24.2 Property Documentation

6.24.2.1 long Talos::Threading::HighperformanceCounter::Frequency `[static, get]`

Returns the frequency of the built-in High Performance counter in Hz (1/s).

6.24.2.2 long Talos::Threading::HighperformanceCounter::Count `[static, get]`

Returns a 64-bit representation of the current count.

6.25 Talos::Threading::ProgressChangedEventArgs Class Reference

Provides data for the ProgressChanged event.

Inherits EventArgs.

Public Member Functions

- ProgressChangedEventArgs (int progressPercent, object userState)
  Initializes a new instance of the ProgressChangedEventArgs class.

Properties

- int ProgressPercentage `[get, set]`
  Gets the asynchronous task progress percentage.
- object UserState `[get, set]`
  Gets a unique user state.

6.25.1 Detailed Description

Provides data for the ProgressChanged event.

6.25.2 Constructor & Destructor Documentation

6.25.2.1 Talos::Threading::ProgressChangedEventArgs::ProgressChangedEventArgs (int *progressPercent*, object *userState*)

Initializes a new instance of the ProgressChangedEventArgs class.

Parameters:

progressPercent The percentage of an asynchronous task that has been completed.

userState A unique user state.

6.25.3 Property Documentation

6.25.3.1 int Talos::Threading::ProgressChangedEventArgs::ProgressPercentage `[get, set]`

Gets the asynchronous task progress percentage.

6.25.3.2 object Talos::Threading::ProgressChangedEventArgs::UserState `[get, set]`

Gets a unique user state.

6.26 Talos::Threading::RunWorkerCompletedEventArgs Class Reference

Provides data for the MethodNameCompleted event. MethodName is a placeholder for the first part of the method's name.

Inherits EventArgs.

Public Member Functions

- RunWorkerCompletedEventArgs (object result, Exception error, bool cancelled)
  Initializes a new instance of the RunWorkerCompletedEventArgs class.

Properties

- bool Cancelled `[get, set]`
  Gets a value indicating whether an asynchronous operation has been canceled.
- Exception Error `[get, set]`
  Gets a value indicating which error occurred during an asynchronous operation.
- object Result `[get, set]`
  Gets a value that represents the result of an asynchronous operation.

6.26.1 Detailed Description

Provides data for the MethodNameCompleted event. MethodName is a placeholder for the first part of the method's name.

6.26.2 Constructor & Destructor Documentation

6.26.2.1 Talos::Threading::RunWorkerCompletedEventArgs::RunWorkerCompletedEventArgs (object *result*, Exception *error*, bool *cancelled*)

Initializes a new instance of the RunWorkerCompletedEventArgs class.

Parameters:

result The result of an asynchronous operation.

error Any error that occurred during the asynchronous operation.

cancelled A value indicating whether the asynchronous operation was canceled.

6.26.3 Property Documentation

6.26.3.1 bool Talos::Threading::RunWorkerCompletedEventArgs::Cancelled `[get, set]`

Gets a value indicating whether an asynchronous operation has been canceled.

6.26.3.2 Exception Talos::Threading::RunWorkerCompletedEventArgs::Error [get, set]

Gets a value indicating which error occurred during an asynchronous operation.

6.26.3.3 object Talos::Threading::RunWorkerCompletedEventArgs::Result [get, set]

Gets a value that represents the result of an asynchronous operation.

6.27 Talos::Threading::Watchdog Class Reference

This is a simple C# OOP wrapper around the existing Windows CE watchdog API. This class can be used to create an object that will terminate a hung application or completely restart a malfunction device.

Public Member Functions

- Watchdog (string name, int period, int wait, WatchdogAction action)

  Create a new watchdog timer in the trusted CE WatchdogTimer API.

- void Start ()

  This method attempts to start this watchdog timer.

- void Stop ()

  This method attempts to stop this watchdog timer.

- void Reset ()

  This method attempts to refresh this watchdog timer. This method must be called before Period ms or the default Action will occur.

Properties

- string Name `[get, set]`

  This property can be used to retrieve the string used to identify this Watchdog at creation.

- int Period `[get, set]`

  This property can be used to retrieve the timeout period (in ms) the watchdog is configured to trigger after exceeding.

- int Wait `[get, set]`

  This property can be used to retrieve the additional wait time (in ms) the watchdog will continue to wait after exceeding the Period.

- WatchdogAction Action `[get, set]`

  This property can be used to retrieve the watchdog action to be preformed when the period + wait timeout expires.

6.27.1 Detailed Description

This is a simple C# OOP wrapper around the existing Windows CE watchdog API. This class can be used to create an object that will terminate a hung application or completely restart a malfunction device.

6.27.2 Constructor & Destructor Documentation

6.27.2.1 Talos::Threading::Watchdog::Watchdog (string *name*, int *period*, int *wait*, WatchdogAction *action*)

Create a new watchdog timer in the trusted CE WatchdogTimer API.

Parameters:

name A string representation of this watchdog name.

period The timeout period for this watchdog in ms.

wait The amount of time to wait after the watchdog period before taking action in ms.

action The action to take upon hitting the period + wait time.

6.27.3 Member Function Documentation

6.27.3.1 void Talos::Threading::Watchdog::Start ()

This method attempts to start this watchdog timer.

Exceptions:

Exception General failure exception. Remote call failed.

6.27.3.2 void Talos::Threading::Watchdog::Stop ()

This method attempts to stop this watchdog timer.

Exceptions:

Exception General failure exception. Remote call failed.

6.27.3.3 void Talos::Threading::Watchdog::Reset ()

This method attempts to refresh this watchdog timer. This method must be called before Period ms or the default Action will occur.

Exceptions:

Exception General failure exception. Remote call failed.

6.27.4 Property Documentation

6.27.4.1 string Talos::Threading::Watchdog::Name `[get, set]`

This property can be used to retrieve the string used to identify this Watchdog at creation.

6.27.4.2 int Talos::Threading::Watchdog::Period `[get, set]`

This property can be used to retrieve the timeout period (in ms) the watchdog is configured to trigger after exceeding.

6.27.4.3 int Talos::Threading::Watchdog::Wait `[get, set]`

This property can be used to retrieve the additional wait time (in ms) the watchdog will continue to wait after exceeding the Period.

6.27.4.4 WatchdogAction Talos::Threading::Watchdog::Action [get, set]

This property can be used to retrieve the watchdog action to be preformed when the period + wait timeout expires.