SerialPort Class

Reference

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v7.0.0

Represents a serial port resource.

In this article

C#

```
public class SerialPort : System.ComponentModel.Component
```

Inheritance Object → MarshalByRefObject → Component → SerialPort

Examples

The following code example demonstrates the use of the SerialPort class to allow two users to chat from two separate computers connected by a null modem cable. In this example, the users are prompted for the port settings and a username before chatting. Both computers must be executing the program to achieve full functionality of this example.

C#

```
// Use this code inside a project created with the Visual C# > Windows Desktop
> Console Application template.
// Replace the code in Program.cs with this code.

using System;
using System.IO.Ports;
using System.Threading;

public class PortChat
{
    static bool _continue;
    static SerialPort _serialPort;

    public static void Main()
    {
        string name;
        string message;
        StringComparer stringComparer = StringComparer.OrdinalIgnoreCase;
        Thread readThread = new Thread(Read);

        // Create a new SerialPort object with default settings.
        _serialPort = new SerialPort();

        // Allow the user to set the appropriate properties.
        _serialPort.PortName = SetPortName(_serialPort.PortName);
        _serialPort.BaudRate = SetPortBaudRate(_serialPort.BaudRate);
        _serialPort.Parity = SetPortParity(_serialPort.Parity);
        _serialPort.DataBits = SetPortDataBits(_serialPort.DataBits);
        _serialPort.StopBits = SetPortStopBits(_serialPort.StopBits);
        _serialPort.Handshake = SetPortHandshake(_serialPort.Handshake);

        // Set the read/write timeouts
        _serialPort.ReadTimeout = 500;
        _serialPort.WriteTimeout = 500;

        _serialPort.Open();
        _continue = true;
        readThread.Start();

        Console.Write("Name: ");
        name = Console.ReadLine();

        Console.WriteLine("Type QUIT to exit");

        while (_continue)
        {
            message = Console.ReadLine();

            if (stringComparer.Equals("quit", message))
            {
```

```
                _continue = false;
            }
            else
            {
                _serialPort.WriteLine(
                    String.Format("<{0}>: {1}", name, message));
            }
        }

        readThread.Join();
        _serialPort.Close();
    }

    public static void Read()
    {
        while (_continue)
        {
            try
            {
                string message = _serialPort.ReadLine();
                Console.WriteLine(message);
            }
            catch (TimeoutException) { }
        }
    }

    // Display Port values and prompt user to enter a port.
    public static string SetPortName(string defaultPortName)
    {
        string portName;

        Console.WriteLine("Available Ports:");
        foreach (string s in SerialPort.GetPortNames())
        {
            Console.WriteLine("   {0}", s);
        }

        Console.Write("Enter COM port value (Default: {0}): ", defaultPort-
Name);
        portName = Console.ReadLine();

        if (portName == "" || !(portName.ToLower()).StartsWith("com"))
        {
            portName = defaultPortName;
        }
        return portName;
    }
    // Display BaudRate values and prompt user to enter a value.
    public static int SetPortBaudRate(int defaultPortBaudRate)
    {
        string baudRate;
```

```
        Console.Write("Baud Rate(default:{0}): ", defaultPortBaudRate);
        baudRate = Console.ReadLine();

        if (baudRate == "")
        {
            baudRate = defaultPortBaudRate.ToString();
        }

        return int.Parse(baudRate);
    }

    // Display PortParity values and prompt user to enter a value.
    public static Parity SetPortParity(Parity defaultPortParity)
    {
        string parity;

        Console.WriteLine("Available Parity options:");
        foreach (string s in Enum.GetNames(typeof(Parity)))
        {
            Console.WriteLine("   {0}", s);
        }

        Console.Write("Enter Parity value (Default: {0}):",
defaultPortParity.ToString(), true);
        parity = Console.ReadLine();

        if (parity == "")
        {
            parity = defaultPortParity.ToString();
        }

        return (Parity)Enum.Parse(typeof(Parity), parity, true);
    }
    // Display DataBits values and prompt user to enter a value.
    public static int SetPortDataBits(int defaultPortDataBits)
    {
        string dataBits;

        Console.Write("Enter DataBits value (Default: {0}): ", defaultPort-
DataBits);
        dataBits = Console.ReadLine();

        if (dataBits == "")
        {
            dataBits = defaultPortDataBits.ToString();
        }

        return int.Parse(dataBits.ToUpperInvariant());
    }
```

```
    // Display StopBits values and prompt user to enter a value.
    public static StopBits SetPortStopBits(StopBits defaultPortStopBits)
    {
        string stopBits;

        Console.WriteLine("Available StopBits options:");
        foreach (string s in Enum.GetNames(typeof(StopBits)))
        {
            Console.WriteLine("   {0}", s);
        }

        Console.Write("Enter StopBits value (None is not supported and \n" +
         "raises an ArgumentOutOfRangeException. \n (Default: {0}):",
defaultPortStopBits.ToString());
        stopBits = Console.ReadLine();

        if (stopBits == "" )
        {
            stopBits = defaultPortStopBits.ToString();
        }

        return (StopBits)Enum.Parse(typeof(StopBits), stopBits, true);
    }
    public static Handshake SetPortHandshake(Handshake defaultPortHandshake)
    {
        string handshake;

        Console.WriteLine("Available Handshake options:");
        foreach (string s in Enum.GetNames(typeof(Handshake)))
        {
            Console.WriteLine("   {0}", s);
        }

        Console.Write("Enter Handshake value (Default: {0}):",
defaultPortHandshake.ToString());
        handshake = Console.ReadLine();

        if (handshake == "")
        {
            handshake = defaultPortHandshake.ToString();
        }

        return (Handshake)Enum.Parse(typeof(Handshake), handshake, true);
    }
}
```

Remarks

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.

The SerialPort class supports the following encodings: ASCIIEncoding, UTF8Encoding, UnicodeEncoding, UTF32Encoding, and any encoding defined in mscorlib.dll where the code page is less than 50000 or the code page is 54936. You can use alternate encodings, but you must use the ReadByte or Write method and perform the encoding yourself.

You use the GetPortNames method to retrieve the valid ports for the current computer.

If a SerialPort object becomes blocked during a read operation, do not abort the thread. Instead, either close the base stream or dispose of the SerialPort object.

Constructors

SerialPort()	Initializes a new instance of the SerialPort class.
SerialPort(IContainer)	Initializes a new instance of the SerialPort class using the specified IContainer object.
SerialPort(String)	Initializes a new instance of the SerialPort class using the specified port name.
SerialPort(String, Int32)	Initializes a new instance of the SerialPort class using the specified port name and baud rate.
SerialPort(String, Int32, Parity)	Initializes a new instance of the SerialPort class using the specified port name, baud rate, and parity bit.
SerialPort(String, Int32, Parity, Int32)	Initializes a new instance of the SerialPort class using the specified port name, baud rate, parity bit, and data bits.
SerialPort(String, Int32, Parity, Int32, StopBits)	Initializes a new instance of the SerialPort class using the specified port name, baud rate, parity bit, data bits, and stop bit.

Fields

InfiniteTimeout	Indicates that no time-out should occur.

Properties

BaseStream	Gets the underlying Stream object for a SerialPort object.
BaudRate	Gets or sets the serial baud rate.
BreakState	Gets or sets the break signal state.
BytesToRead	Gets the number of bytes of data in the receive buffer.
BytesToWrite	Gets the number of bytes of data in the send buffer.
CanRaiseEvents	Gets a value indicating whether the component can raise an event. (Inherited from Component)
CDHolding	Gets the state of the Carrier Detect line for the port.
Container	Gets the IContainer that contains the Component. (Inherited from Component)
CtsHolding	Gets the state of the Clear-to-Send line.
DataBits	Gets or sets the standard length of data bits per byte.
DesignMode	Gets a value that indicates whether the Component is currently in design mode. (Inherited from Component)
DiscardNull	Gets or sets a value indicating whether null bytes are ignored when transmitted between the port and the receive buffer.
DsrHolding	Gets the state of the Data Set Ready (DSR) signal.
DtrEnable	Gets or sets a value that enables the Data Terminal Ready (DTR) signal during serial communication.
Encoding	Gets or sets the byte encoding for pre- and post-transmission conversion of text.
Events	Gets the list of event handlers that are attached to this Component. (Inherited from Component)
Handshake	Gets or sets the handshaking protocol for serial port transmission of data using a value from Handshake.
IsOpen	Gets a value indicating the open or closed status of the SerialPort object.

NewLine	Gets or sets the value used to interpret the end of a call to the ReadLine() and WriteLine(String) methods.
Parity	Gets or sets the parity-checking protocol.
ParityReplace	Gets or sets the byte that replaces invalid bytes in a data stream when a parity error occurs.
PortName	Gets or sets the port for communications, including but not limited to all available COM ports.
ReadBufferSize	Gets or sets the size of the SerialPort input buffer.
ReadTimeout	Gets or sets the number of milliseconds before a time-out occurs when a read operation does not finish.
ReceivedBytesThreshold	Gets or sets the number of bytes in the internal input buffer before a DataReceived event occurs.
RtsEnable	Gets or sets a value indicating whether the Request to Send (RTS) signal is enabled during serial communication.
Site	Gets or sets the ISite of the Component. (Inherited from Component)
StopBits	Gets or sets the standard number of stopbits per byte.
WriteBufferSize	Gets or sets the size of the serial port output buffer.
WriteTimeout	Gets or sets the number of milliseconds before a time-out occurs when a write operation does not finish.

Methods

Close()	Closes the port connection, sets the IsOpen property to `false`, and disposes of the internal Stream object.
CreateObjRef(Type)	Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. (Inherited from MarshalByRefObject)
DiscardInBuffer()	Discards data from the serial driver's receive buffer.
DiscardOutBuffer()	Discards data from the serial driver's transmit buffer.
Dispose()	Releases all resources used by the Component.

	(Inherited from Component)
Dispose(Boolean)	Releases the unmanaged resources used by the SerialPort and optionally releases the managed resources.
Equals(Object)	Determines whether the specified object is equal to the current object. (Inherited from Object)
GetHashCode()	Serves as the default hash function. (Inherited from Object)
GetLifetimeService()	Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject)
GetPortNames()	Gets an array of serial port names for the current computer.
GetService(Type)	Returns an object that represents a service provided by the Component or by its Container. (Inherited from Component)
GetType()	Gets the Type of the current instance. (Inherited from Object)
InitializeLifetimeService()	Obtains a lifetime service object to control the lifetime policy for this instance. (Inherited from MarshalByRefObject)
MemberwiseClone()	Creates a shallow copy of the current Object. (Inherited from Object)
MemberwiseClone(Boolean)	Creates a shallow copy of the current MarshalByRefObject object. (Inherited from MarshalByRefObject)
Open()	Opens a new serial port connection.
Read(Byte[], Int32, Int32)	Reads a number of bytes from the SerialPort input buffer and writes those bytes into a byte array at the specified offset.
Read(Char[], Int32, Int32)	Reads a number of characters from the SerialPort input buffer and writes them into an array of characters at a given offset.
ReadByte()	Synchronously reads one byte from the SerialPort input buffer.
ReadChar()	Synchronously reads one character from the SerialPort input buffer.

ReadExisting()	Reads all immediately available bytes, based on the encoding, in both the stream and the input buffer of the SerialPort object.
ReadLine()	Reads up to the NewLine value in the input buffer.
ReadTo(String)	Reads a string up to the specified `value` in the input buffer.
ToString()	Returns a String containing the name of the Component, if any. This method should not be overridden. (Inherited from Component)
Write(Byte[], Int32, Int32)	Writes a specified number of bytes to the serial port using data from a buffer.
Write(Char[], Int32, Int32)	Writes a specified number of characters to the serial port using data from a buffer.
Write(String)	Writes the specified string to the serial port.
WriteLine(String)	Writes the specified string and the NewLine value to the output buffer.

Events

DataReceived	Indicates that data has been received through a port represented by the SerialPort object.
Disposed	Occurs when the component is disposed by a call to the Dispose() method. (Inherited from Component)
ErrorReceived	Indicates that an error has occurred with a port represented by a SerialPort object.
PinChanged	Indicates that a non-data signal event has occurred on the port represented by the SerialPort object.

Applies to

Product	Versions
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Platform Extensions	2.1, 2.2, 3.0, 3.1, 5, 6, 7

Product	Versions
Xamarin.Mac	3.0