System.IO.Ports Namespace

Reference

Contains classes for controlling serial ports. The most important class, SerialPort, provides a framework for synchronous and event-driven I/O, access to pin and break states, and access to serial driver properties. It can be used to wrap Stream objects, allowing the serial port to be accessed by classes that use streams.

Classes

Expand table

SerialDataReceivedEventArgs	Provides data for the DataReceived event.
SerialErrorReceivedEventArgs	Prepares data for the ErrorReceived event.
SerialPinChangedEventArgs	Provides data for the PinChanged event.
SerialPort	Represents a serial port resource.

Enums

Expand table

Handshake	Specifies the control protocol used in establishing a serial port communication for a SerialPort object.
Parity	Specifies the parity bit for a SerialPort object.
SerialData	Specifies the type of character that was received on the serial port of the SerialPort object.
SerialError	Specifies errors that occur on the SerialPort object.
SerialPin Change	Specifies the type of change that occurred on the SerialPort object.
StopBits	Specifies the number of stop bits used on the SerialPort object.

Delegates

Expand table

SerialDataReceivedEventHandler	Represents the method that will handle the DataReceived event of a SerialPort object.
SerialErrorReceivedEventHandler	Represents the method that will handle the ErrorReceived event of a SerialPort object.
SerialPinChangedEventHandler	Represents the method that will handle the PinChanged event of a SerialPort object.

Remarks

The namespace includes enumerations that simplify the control of serial ports, such as Handshake, Parity, SerialPinChange, and StopBits.

Handshake Enum

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: Handshake.cs

Specifies the control protocol used in establishing a serial port communication for a SerialPort object.

C#

```
public enum Handshake
```

Inheritance Object → ValueType → Enum → Handshake

Fields

Expand table

Name	Value	Description
None	0	No control is used for the handshake.
XOnXOff	1	The XON/XOFF software control protocol is used. The XOFF control is sent to stop the transmission of data. The XON control is sent to resume the transmission. These software controls are used instead of Request to Send (RTS) and Clear to Send (CTS) hardware controls.
RequestToSend	2	Request-to-Send (RTS) hardware flow control is used. RTS signals that data is available for transmission. If the input buffer becomes full, the RTS line will be set to `false`. The RTS line will be set to `true` when more room becomes available in the input buffer.
RequestToSendXOnXOff	3	Both the Request-to-Send (RTS) hardware control and the XON/XOFF software controls are used.

Examples

The following code example displays the possible values of the Handshake enumeration to the console, then prompts the user to choose one. This code example is part of a larger code example provided for the SerialPort class.

C#

```csharp
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

This enumeration is used with the Handshake property.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

Parity Enum

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: Parity.cs

Specifies the parity bit for a SerialPort object.

```
C#

public enum Parity
```

Inheritance Object → ValueType → Enum → Parity

Fields

Expand table

Name	Value	Description
None	0	No parity check occurs.
Odd	1	Sets the parity bit so that the count of bits set is an odd number.
Even	2	Sets the parity bit so that the count of bits set is an even number.
Mark	3	Leaves the parity bit set to 1.
Space	4	Leaves the parity bit set to 0.

Examples

The following code example displays the possible values of the Parity enumeration to the console, then prompts the user to choose one. This code example is part of a larger code example provided for the SerialPort class.

```
C#

// 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);
}
```

Remarks

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.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialData Enum

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialData.cs

Specifies the type of character that was received on the serial port of the SerialPort object.

```C#
public enum SerialData
```

Inheritance Object → ValueType → Enum → SerialData

Fields

Expand table

Name	Value	Description
Chars	1	A character was received and placed in the input buffer.
Eof	2	The end of file character was received and placed in the input buffer.

Remarks

This enumeration is used with the SerialPort.DataReceived event. You examine the type of character that was received by retrieving the value of the SerialDataReceivedEventArgs.EventType property. The EventType property contains one of the values from the `SerialData` enumeration.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)

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 Standard	2.0 (package-provided)

SerialDataReceivedEventArgs Class

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialDataReceivedEventArgs.cs

Provides data for the DataReceived event.

```C#
public class SerialDataReceivedEventArgs : EventArgs
```

Inheritance Object → EventArgs → SerialDataReceivedEventArgs

Remarks

This class is used with the DataReceived event.

Properties

Expand table

EventType	Gets or sets the event type.

Expand table

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialDataReceivedEventArgs.EventType Property

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialDataReceivedEventArgs.cs

Gets or sets the event type.

```
C#

public System.IO.Ports.SerialData EventType { get; }
```

Property Value

SerialData

One of the SerialData values.

Remarks

This property provides information about the event type that caused the DataReceived event.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialDataReceivedEventHandler Delegate

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialDataReceivedEventHandler.cs

Represents the method that will handle the DataReceived event of a SerialPort object.

```
C#

public delegate void SerialDataReceivedEventHandler(object sender,
SerialDataReceivedEventArgs e);
```

Parameters

`sender` Object

The sender of the event, which is the SerialPort object.

`e` SerialDataReceivedEventArgs

A SerialDataReceivedEventArgs object that contains the event data.

Remarks

When you create a SerialDataReceivedEventHandler delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about event-handler delegates, see Handling and Raising Events.

Extension Methods

Expand table

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialError Enum

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialError.cs

Specifies errors that occur on the SerialPort object.

```
C#

public enum SerialError
```

Inheritance Object → ValueType → Enum → SerialError

Fields

Expand table

Name	Value	Description
RXOver	1	An input buffer overflow has occurred. There is either no room in the input buffer, or a character was received after the end-of-file (EOF) character.
Overrun	2	A character-buffer overrun has occurred. The next character is lost.
RXParity	4	The hardware detected a parity error.
Frame	8	The hardware detected a framing error.
TXFull	256	The application tried to transmit a character, but the output buffer was full.

Remarks

This enumeration can be useful when handling the SerialPort.ErrorReceived event to detect and respond to errors when communicating data through a SerialPort. You examine the type of error by retrieving the SerialErrorReceivedEventArgs.EventType property. The EventType property contains one of the values from the `SerialError` enumeration.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialErrorReceivedEventArgs Class

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialErrorReceivedEventArgs.cs

Prepares data for the ErrorReceived event.

```
C#

public class SerialErrorReceivedEventArgs : EventArgs
```

Inheritance Object → EventArgs → SerialErrorReceivedEventArgs

Remarks

This class is used with the ErrorReceived event.

Properties

Expand table

EventType	Gets or sets the event type.

Expand table

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialErrorReceivedEventArgs.EventType Property

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialErrorReceivedEventArgs.cs

Gets or sets the event type.

```
C#

public System.IO.Ports.SerialError EventType { get; }
```

Property Value

SerialError

One of the SerialError values.

Remarks

This property provides information on the event type that caused the ErrorReceived event.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialErrorReceivedEventHandler Delegate

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialErrorReceivedEventHandler.cs

Represents the method that will handle the ErrorReceived event of a SerialPort object.

```
C#

public delegate void SerialErrorReceivedEventHandler(object sender,
SerialErrorReceivedEventArgs e);
```

Parameters

`sender` Object

The sender of the event, which is the SerialPort object.

`e` SerialErrorReceivedEventArgs

A SerialErrorReceivedEventArgs object that contains the event data.

Remarks

When you create a SerialErrorReceivedEventHandler delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about event-handler delegates, see Handling and Raising Events.

Extension Methods

Expand table

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPinChange Enum

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPinChange.cs

Specifies the type of change that occurred on the SerialPort object.

```
C#
```

```csharp
public enum SerialPinChange
```

Inheritance Object → ValueType → Enum → SerialPinChange

Fields

Expand table

Name	Value	Description
CtsChanged	8	The Clear to Send (CTS) signal changed state. This signal is used to indicate whether data can be sent over the serial port.
DsrChanged	16	The Data Set Ready (DSR) signal changed state. This signal is used to indicate whether the device on the serial port is ready to operate.
CDChanged	32	The Carrier Detect (CD) signal changed state. This signal is used to indicate whether a modem is connected to a working phone line and a data carrier signal is detected.
Break	64	A break was detected on input.
Ring	256	A ring indicator was detected.

Remarks

This enumeration is used with the PinChanged event.

A serial port pin changes state when it is asserted or unasserted.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPinChangedEventArgs Class

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPinChangedEventArgs.cs

Provides data for the PinChanged event.

```
C#

public class SerialPinChangedEventArgs : EventArgs
```

Inheritance Object → EventArgs → SerialPinChangedEventArgs

Remarks

This class is used with the PinChanged event.

Properties

Expand table

EventType	Gets or sets the event type.

Expand table

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPinChangedEventArgs.EventType Property

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPinChangedEventArgs.cs

Gets or sets the event type.

```
C#

public System.IO.Ports.SerialPinChange EventType { get; }
```

Property Value

SerialPinChange

One of the SerialPinChange values.

Remarks

This property provides information about the event type that caused the PinChanged event.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPinChangedEventHandler Delegate

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPinChangedEventHandler.cs

Represents the method that will handle the PinChanged event of a SerialPort object.

C#

```
public delegate void SerialPinChangedEventHandler(object sender,
SerialPinChangedEventArgs e);
```

Parameters

`sender` Object

The source of the event, which is the SerialPort object.

`e` SerialPinChangedEventArgs

A SerialPinChangedEventArgs object that contains the event data.

Remarks

When you create a SerialPinChangedEventHandler delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about event-handler delegates, see Handling and Raising Events.

Extension Methods

Expand table

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort Class

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Represents a serial port resource.

C#

```csharp
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#

```csharp
// 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}): ", defaultPortName);
        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}): ",
defaultPortDataBits);
        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

Expand table

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, 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.
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)	Initializes a new instance of the SerialPort class using the specified port name, baud rate, and parity bit.
SerialPort(String, Int32)	Initializes a new instance of the SerialPort class using the specified port name and baud rate.
SerialPort(String)	Initializes a new instance of the SerialPort class using the specified port name.

Fields

Expand table

InfiniteTimeout	Indicates that no time-out should occur.

Properties

Expand table

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.
CDHolding	Gets the state of the Carrier Detect line for the port.
CtsHolding	Gets the state of the Clear-to-Send line.
DataBits	Gets or sets the standard length of data bits per byte.
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.
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.
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

Expand table

Close()	Closes the port connection, sets the IsOpen property to `false`, and disposes of the internal Stream object.
DiscardInBuffer()	Discards data from the serial driver's receive buffer.
DiscardOutBuffer()	Discards data from the serial driver's transmit buffer.
Dispose(Boolean)	Releases the unmanaged resources used by the SerialPort and optionally releases the managed resources.
GetPortNames()	Gets an array of serial port names for the current computer.
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.
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

Expand table

Data Received	Indicates that data has been received through a port represented by the SerialPort object.
Error Received	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	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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

Product	Versions
.NET Standard	2.0 (package-provided)

SerialPort Constructors

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Initializes a new instance of the SerialPort class.

Overloads

Expand table

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.

SerialPort()

Source: SerialPort.cs

Initializes a new instance of the SerialPort class.

C#

```
public 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. This code example is part of a larger code example provided for the SerialPort class.

C#

```
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) { }
    }
}
```

Remarks

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.

Applies to

▼ .NET 10 (package-provided) and other versions

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort(IContainer)

Source: SerialPort.cs

Initializes a new instance of the SerialPort class using the specified IContainer object.

C#

```csharp
public SerialPort(System.ComponentModel.IContainer container);
```

Parameters

`container` IContainer

An interface to a container.

Exceptions

IOException

The specified port could not be found or opened.

Remarks

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.

Applies to

▼ .NET 10 (package-provided) and other versions

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort(String)

Source: SerialPort.cs

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

```csharp
C#

public SerialPort(string portName);
```

Parameters

`portName` String

The port to use (for example, COM1).

Exceptions

IOException

The specified port could not be found or opened.

Remarks

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

Applies to

▼ .NET 10 (package-provided) and other versions

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort(String, Int32)

Source: SerialPort.cs

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

```
C#

public SerialPort(string portName, int baudRate);
```

Parameters

`portName` String

The port to use (for example, COM1).

`baudRate` Int32

The baud rate.

Exceptions

IOException

The specified port could not be found or opened.

Remarks

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

Applies to

▼ .NET 10 (package-provided) and other versions

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort(String, Int32, Parity)

Source: SerialPort.cs

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

C#

```
public SerialPort(string portName, int baudRate, System.IO.Ports.Parity
parity);
```

Parameters

`portName` String

The port to use (for example, COM1).

`baudRate` Int32

The baud rate.

`parity` Parity

One of the Parity values.

Exceptions

IOException

The specified port could not be found or opened.

Remarks

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

Applies to

▼ .NET 10 (package-provided) and other versions

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort(String, Int32, Parity, Int32)

Source: SerialPort.cs

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

```
C#

public SerialPort(string portName, int baudRate, System.IO.Ports.Parity parity,
int dataBits);
```

Parameters

`portName` String

The port to use (for example, COM1).

`baudRate` Int32

The baud rate.

`parity` Parity

One of the Parity values.

`dataBits` Int32

The data bits value.

Exceptions

IOException

The specified port could not be found or opened.

Remarks

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, and data bits.

Applies to

▼ .NET 10 (package-provided) and other versions

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort(String, Int32, Parity, Int32, StopBits)

Source: SerialPort.cs

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

```
C#
```

```
public SerialPort(string portName, int baudRate, System.IO.Ports.Parity parity,
int dataBits, System.IO.Ports.StopBits stopBits);
```

Parameters

`portName` String

The port to use (for example, COM1).

`baudRate` Int32

The baud rate.

`parity` Parity

One of the Parity values.

`dataBits` Int32

The data bits value.

`stopBits` StopBits

One of the StopBits values.

Exceptions

IOException

The specified port could not be found or opened.

Remarks

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.

Applies to

▼ .NET 10 (package-provided) and other versions

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.InfiniteTimeout Field

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Indicates that no time-out should occur.

```
C#

public const int InfiniteTimeout = -1;
```

Field Value

Value = -1

Int32

Remarks

This value is used with the ReadTimeout and WriteTimeout properties.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.BaseStream Property

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Gets the underlying Stream object for a SerialPort object.

```C#
public System.IO.Stream BaseStream { get; }
```

Property Value

Stream

A Stream object.

Exceptions

InvalidOperationException

The stream is closed. This can occur because the Open() method has not been called or the Close() method has been called.

NotSupportedException

The stream is in a .NET Compact Framework application and one of the following methods was called:

BeginRead(Byte[], Int32, Int32, AsyncCallback, Object)BeginWrite(Byte[], Int32, Int32, AsyncCallback, Object)EndRead(IAsyncResult)EndWrite(IAsyncResult)

The .NET Compact Framework does not support the asynchronous model with base streams.

Remarks

Use this property for explicit asynchronous I/O operations or to pass the SerialPort object to a Stream wrapper class such as StreamWriter.

Any open serial port's BaseStream property returns an object that derives from the abstract Stream class, and implements read and write methods using the prototypes inherited from the Stream class: BeginRead, BeginWrite, Read, ReadByte, Write, and WriteByte. These methods can be useful when passing a wrapped serial resource to a Stream wrapper class.

Due to the inaccessibility of the wrapped file handle, the Length and Position properties are not supported, and the Seek and SetLength methods are not supported.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.BaudRate Property

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Gets or sets the serial baud rate.

C#

```
public int BaudRate { get; set; }
```

Property Value

Int32

The baud rate.

Exceptions

ArgumentOutOfRangeException

The baud rate specified is less than or equal to zero, or is greater than the maximum allowable baud rate for the device.

IOException

The port is in an invalid state.

-or-

An attempt to set the state of the underlying port failed. For example, the parameters passed from this SerialPort object were invalid.

Examples

The following example shows how to set the BaudRate property to `9600`.

C#

```
SerialPort mySerialPort = new SerialPort("COM1");

mySerialPort.BaudRate = 9600;
mySerialPort.Parity = Parity.None;
mySerialPort.StopBits = StopBits.One;
mySerialPort.DataBits = 8;
mySerialPort.Handshake = Handshake.None;
mySerialPort.RtsEnable = true;
```

The following 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. This code example is part of a larger code example provided for the SerialPort class.

C#

```
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) { }
        }
    }
```

Remarks

The baud rate must be supported by the user's serial driver. The default value is 9600 bits per second (bps).

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.BreakState Property

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Gets or sets the break signal state.

```
C#

public bool BreakState { get; set; }
```

Property Value

Boolean

`true` if the port is in a break state; otherwise, `false`.

Exceptions

IOException

The port is in an invalid state.

-or-

An attempt to set the state of the underlying port failed. For example, the parameters passed from this SerialPort object were invalid.

InvalidOperationException

The stream is closed. This can occur because the Open() method has not been called or the Close() method has been called.

Remarks

The break signal state occurs when a transmission is suspended and the line is placed in a break state (all low, no stop bit) until released. To enter a break state, set this property to `true`. If the port is already in a break state, setting this property again to `true` does not result in an exception. It is not possible to write to the SerialPort object while BreakState is `true`.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.BytesToRead Property

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

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

```C#
public int BytesToRead { get; }
```

Property Value

Int32

The number of bytes of data in the receive buffer.

Exceptions

InvalidOperationException

The port is not open.

Remarks

The receive buffer includes the serial driver's receive buffer as well as internal buffering in the SerialPort object itself.

Because the BytesToRead property represents both the SerialPort buffer and the Windows-created buffer, it can return a greater value than the ReadBufferSize property, which represents only the Windows-created buffer.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)

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 Standard	2.0 (package-provided)

SerialPort.BytesToWrite Property

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

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

C#

```
public int BytesToWrite { get; }
```

Property Value

Int32

The number of bytes of data in the send buffer.

Exceptions

IOException

The port is in an invalid state.

InvalidOperationException

The stream is closed. This can occur because the Open() method has not been called or the Close() method has been called.

Remarks

The send buffer includes the serial driver's send buffer as well as internal buffering in the SerialPort object itself.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)

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 Standard	2.0 (package-provided)

SerialPort.CDHolding Property

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Gets the state of the Carrier Detect line for the port.

```
C#

public bool CDHolding { get; }
```

Property Value

Boolean

`true` if the carrier is detected; otherwise, `false`.

Exceptions

IOException

The port is in an invalid state.

-or-

An attempt to set the state of the underlying port failed. For example, the parameters passed from this SerialPort object were invalid.

InvalidOperationException

The stream is closed. This can occur because the Open() method has not been called or the Close() method has been called.

Remarks

This property can be used to monitor the state of the carrier detection line for a port. No carrier usually indicates that the receiver has hung up and the carrier has been dropped.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.CtsHolding Property

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Gets the state of the Clear-to-Send line.

```
C#

public bool CtsHolding { get; }
```

Property Value

Boolean

`true` if the Clear-to-Send line is detected; otherwise, `false`.

Exceptions

IOException

The port is in an invalid state.

-or-

An attempt to set the state of the underlying port failed. For example, the parameters passed from this SerialPort object were invalid.

InvalidOperationException

The stream is closed. This can occur because the Open() method has not been called or the Close() method has been called.

Remarks

The Clear-to-Send (CTS) line is used in Request to Send/Clear to Send (RTS/CTS) hardware handshaking. The CTS line is queried by a port before data is sent.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.DataBits Property

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Gets or sets the standard length of data bits per byte.

C#

```
public int DataBits { get; set; }
```

Property Value

Int32

The data bits length.

Exceptions

IOException

The port is in an invalid state.

-or-

An attempt to set the state of the underlying port failed. For example, the parameters passed from this SerialPort object were invalid.

ArgumentOutOfRangeException

The data bits value is less than 5 or more than 8.

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. This code example is part of a larger code example provided for the SerialPort class.

C#

```
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) { }
    }
}
```

Remarks

The range of values for this property is from 5 through 8. The default value is 8.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.DiscardNull Property

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Gets or sets a value indicating whether null bytes are ignored when transmitted between the port and the receive buffer.

```
C#

public bool DiscardNull { get; set; }
```

Property Value

Boolean

`true` if null bytes are ignored; otherwise `false`. The default is `false`.

Exceptions

IOException

The port is in an invalid state.

-or-

An attempt to set the state of the underlying port failed. For example, the parameters passed from this SerialPort object were invalid.

InvalidOperationException

The stream is closed. This can occur because the Open() method has not been called or the Close() method has been called.

Remarks

This value should normally be set to `false`, especially for binary transmissions. Setting this property to `true` can cause unexpected results for UTF32- and UTF16-encoded bytes.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.DsrHolding Property

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Gets the state of the Data Set Ready (DSR) signal.

```csharp
public bool DsrHolding { get; }
```

Property Value

Boolean

`true` if a Data Set Ready signal has been sent to the port; otherwise, `false`.

Exceptions

IOException

The port is in an invalid state.

-or-

An attempt to set the state of the underlying port failed. For example, the parameters passed from this SerialPort object were invalid.

InvalidOperationException

The stream is closed. This can occur because the Open() method has not been called or the Close() method has been called.

Remarks

This property is used in Data Set Ready/Data Terminal Ready (DSR/DTR) handshaking. The Data Set Ready (DSR) signal is usually sent by a modem to a port to indicate that it is ready for data transmission or data reception.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.DtrEnable Property

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Gets or sets a value that enables the Data Terminal Ready (DTR) signal during serial communication.

```
C#

public bool DtrEnable { get; set; }
```

Property Value

Boolean

`true` to enable Data Terminal Ready (DTR); otherwise, `false`. The default is `false`.

Exceptions

IOException

The port is in an invalid state.

-or-

An attempt to set the state of the underlying port failed. For example, the parameters passed from this SerialPort object were invalid.

Remarks

Data Terminal Ready (DTR) is typically enabled during XON/XOFF software handshaking and Request to Send/Clear to Send (RTS/CTS) hardware handshaking, and modem communications.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.Encoding Property

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Gets or sets the byte encoding for pre- and post-transmission conversion of text.

```csharp
public System.Text.Encoding Encoding { get; set; }
```

Property Value

Encoding

An Encoding object. The default is ASCIIEncoding.

Exceptions

ArgumentNullException

The Encoding property was set to `null`.

ArgumentException

The Encoding property was set to an encoding that is not ASCIIEncoding, UTF8Encoding, UTF32Encoding, UnicodeEncoding, one of the Windows single byte encodings, or one of the Windows double byte encodings.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.Handshake Property

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Gets or sets the handshaking protocol for serial port transmission of data using a value from Handshake.

C#

```
public System.IO.Ports.Handshake Handshake { get; set; }
```

Property Value

Handshake

One of the Handshake values. The default is `None`.

Exceptions

IOException

The port is in an invalid state.

-or-

An attempt to set the state of the underlying port failed. For example, the parameters passed from this SerialPort object were invalid.

ArgumentOutOfRangeException

The value passed is not a valid value in the Handshake enumeration.

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. This code example is part of a larger code example provided for the SerialPort class.

C#

```
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) { }
    }
}
```

Remarks

When handshaking is used, the device connected to the SerialPort object is instructed to stop sending data when there is at least (ReadBufferSize-1024) bytes in the buffer. The device is instructed to start sending data again when there are 1024 or fewer bytes in the buffer. If the device is sending data in blocks that are larger than 1024 bytes, this may cause the buffer to overflow.

If the Handshake property is set to RequestToSendXOnXOff and CtsHolding is set to `false`, the XOff character will not be sent. If CtsHolding is then set to `true`, more data must be sent before the XOff character will be sent.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.IsOpen Property

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Gets a value indicating the open or closed status of the SerialPort object.

```C#
public bool IsOpen { get; }
```

Property Value

Boolean

`true` if the serial port is open; otherwise, `false`. The default is `false`.

Remarks

The IsOpen property tracks whether the port is open for use by the caller, not whether the port is open by any application on the machine.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.NewLine Property

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Gets or sets the value used to interpret the end of a call to the ReadLine() and WriteLine(String) methods.

```
C#

public string NewLine { get; set; }
```

Property Value

String

A value that represents the end of a line. The default is a line feed ("\n" in C# or vbLf in Visual Basic).

Exceptions

ArgumentException

The property value is empty.

ArgumentNullException

The property value is `null`.

Remarks

This property determines what value (byte) defines the end of a line for the ReadLine and WriteLine methods. By default the end-of-line value is a line feed character (`\n` in C#, Constants.vbLf in Visual Basic). You would change this to a different value if the particular serial device you're working with uses a different value for the same purpose.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.Parity Property

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Gets or sets the parity-checking protocol.

```
C#

public System.IO.Ports.Parity Parity { get; set; }
```

Property Value

Parity

One of the enumeration values that represents the parity-checking protocol. The default is None.

Exceptions

IOException

The port is in an invalid state.

-or-

An attempt to set the state of the underlying port failed. For example, the parameters passed from this SerialPort object were invalid.

ArgumentOutOfRangeException

The Parity value passed is not a valid value in the Parity enumeration.

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. This code example is part of a larger code example provided for the SerialPort class.

C#

```
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) { }
    }
}
```

Remarks

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.

If a parity error occurs on the trailing byte of a stream, an extra byte will be added to the input buffer with a value of 126.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.ParityReplace Property

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Gets or sets the byte that replaces invalid bytes in a data stream when a parity error occurs.

```
C#

public byte ParityReplace { get; set; }
```

Property Value

Byte

A byte that replaces invalid bytes.

Exceptions

IOException

The port is in an invalid state.

-or-

An attempt to set the state of the underlying port failed. For example, the parameters passed from this SerialPort object were invalid.

Remarks

If the value is set to the null character, parity replacement is disabled.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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

Product	Versions
.NET Standard	2.0 (package-provided)

SerialPort.PortName Property

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Gets or sets the port for communications, including but not limited to all available COM ports.

```
C#

public string PortName { get; set; }
```

Property Value

String

The communications port. The default is COM1.

Exceptions

ArgumentException

The PortName property was set to a value with a length of zero.

-or-

The PortName property was set to a value that starts with "\\".

-or-

The port name was not valid.

ArgumentNullException

The PortName property was set to `null`.

InvalidOperationException

The specified port is open.

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. This code example is part of a larger code example provided for the SerialPort class.

C#

```
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) { }
        }
    }
```

Remarks

A list of valid port names can be obtained using the GetPortNames method.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.ReadBufferSize Property

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Gets or sets the size of the SerialPort input buffer.

```
C#

public int ReadBufferSize { get; set; }
```

Property Value

Int32

The buffer size, in bytes. The default value is 4096; the maximum value is that of a positive int, or 2147483647.

Exceptions

ArgumentOutOfRangeException

The ReadBufferSize value set is less than or equal to zero.

InvalidOperationException

The ReadBufferSize property was set while the stream was open.

IOException

The ReadBufferSize property was set to an odd integer value.

Remarks

The ReadBufferSize property ignores any value smaller than 4096.

Because the ReadBufferSize property represents only the Windows-created buffer, it can return a smaller value than the BytesToRead property, which represents both the SerialPort buffer and the Windows-created buffer.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.ReadTimeout Property

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

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

```
C#

public int ReadTimeout { get; set; }
```

Property Value

Int32

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

Exceptions

IOException

The port is in an invalid state.

-or-

An attempt to set the state of the underlying port failed. For example, the parameters passed from this SerialPort object were invalid.

ArgumentOutOfRangeException

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

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. This code example is part of a larger code example provided for the SerialPort class.

C#

```
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) { }
    }
}
```

Remarks

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.

This property does not affect the BeginRead method of the stream returned by the BaseStream property.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.ReceivedBytesThreshold Property

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Gets or sets the number of bytes in the internal input buffer before a DataReceived event occurs.

```
C#

public int ReceivedBytesThreshold { get; set; }
```

Property Value

Int32

The number of bytes in the internal input buffer before a DataReceived event is fired. The default is 1.

Exceptions

ArgumentOutOfRangeException

The ReceivedBytesThreshold value is less than or equal to zero.

Remarks

The DataReceived event is also raised if an Eof character is received, regardless of the number of bytes in the internal input buffer and the value of the ReceivedBytesThreshold property.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)

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 Standard	2.0 (package-provided)

SerialPort.RtsEnable Property

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Gets or sets a value indicating whether the Request to Send (RTS) signal is enabled during serial communication.

```csharp
public bool RtsEnable { get; set; }
```

Property Value

Boolean

`true` to enable Request to Transmit (RTS); otherwise, `false`. The default is `false`.

Exceptions

InvalidOperationException

The value of the RtsEnable property was set or retrieved while the Handshake property is set to the RequestToSend value or the RequestToSendXOnXOff value.

IOException

The port is in an invalid state.

-or-

An attempt to set the state of the underlying port failed. For example, the parameters passed from this SerialPort object were invalid.

Remarks

The Request to Transmit (RTS) signal is typically used in Request to Send/Clear to Send (RTS/CTS) hardware handshaking.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.StopBits Property

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Gets or sets the standard number of stopbits per byte.

C#

```
public System.IO.Ports.StopBits StopBits { get; set; }
```

Property Value

StopBits

One of the StopBits values.

Exceptions

ArgumentOutOfRangeException

The StopBits value is None.

IOException

The port is in an invalid state.

-or-

An attempt to set the state of the underlying port failed. For example, the parameters passed from this SerialPort object were invalid.

Examples

The following example shows how to set the StopBits property to `One`.

C#

```
SerialPort mySerialPort = new SerialPort("COM1");

mySerialPort.BaudRate = 9600;
mySerialPort.Parity = Parity.None;
mySerialPort.StopBits = StopBits.One;
mySerialPort.DataBits = 8;
mySerialPort.Handshake = Handshake.None;
mySerialPort.RtsEnable = true;
```

The following 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. This code example is part of a larger example provided for the SerialPort class.

C#

```csharp
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) { }
        }
    }
```

Remarks

The default value for StopBits is One.

The StopBits.None value is not supported.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.WriteBufferSize Property

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Gets or sets the size of the serial port output buffer.

```csharp
public int WriteBufferSize { get; set; }
```

Property Value

Int32

The size of the output buffer. The default is 2048.

Exceptions

ArgumentOutOfRangeException

The WriteBufferSize value is less than or equal to zero.

InvalidOperationException

The WriteBufferSize property was set while the stream was open.

IOException

The WriteBufferSize property was set to an odd integer value.

Remarks

The WriteBufferSize property ignores any value smaller than 2048.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.WriteTimeout Property

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

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

C#

```
public int WriteTimeout { get; set; }
```

Property Value

Int32

The number of milliseconds before a time-out occurs. The default is InfiniteTimeout.

Exceptions

IOException

The port is in an invalid state.

-or-

An attempt to set the state of the underlying port failed. For example, the parameters passed from this SerialPort object were invalid.

ArgumentOutOfRangeException

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

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. This code example is part of a larger code example provided for the SerialPort class.

C#

```csharp
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) { }
    }
}
```

Remarks

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.

This property does not affect the BeginWrite method of the stream returned by the BaseStream property.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.Close Method

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Closes the port connection, sets the IsOpen property to `false`, and disposes of the internal Stream object.

C#

```csharp
public void Close();
```

Exceptions

IOException

The port is in an invalid state.

-or-

An attempt to set the state of the underlying port failed. For example, the parameters passed from this SerialPort object were invalid.

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. This code example is part of a larger code example provided for the SerialPort class.

C#

```csharp
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) { }
        }
    }
```

Remarks

Calling this method closes the SerialPort object and clears both the receive and transmit buffers. This method calls the Component.Dispose() method, which invokes the protected SerialPort.Dispose(Boolean) method with the `disposing` parameter set to `true`.

The best practice for any application is to wait for some amount of time after calling the Close method before attempting to call the Open method, as the port may not be closed instantly.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.DiscardInBuffer Method

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Discards data from the serial driver's receive buffer.

```
C#
public void DiscardInBuffer();
```

Exceptions

IOException

The port is in an invalid state.

-or-

An attempt to set the state of the underlying port failed. For example, the parameters passed from this SerialPort object were invalid.

InvalidOperationException

The stream is closed. This can occur because the Open() method has not been called or the Close() method has been called.

Remarks

This method is equivalent to the following Visual Basic code: `MSComm1.InBufferCount = 0`. It clears the receive buffer, but does not affect the transmit buffer.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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

Product	Versions
.NET Standard	2.0 (package-provided)

SerialPort.DiscardOutBuffer Method

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Discards data from the serial driver's transmit buffer.

```
C#

public void DiscardOutBuffer();
```

Exceptions

IOException

The port is in an invalid state.

-or-

An attempt to set the state of the underlying port failed. For example, the parameters passed from this SerialPort object were invalid.

InvalidOperationException

The stream is closed. This can occur because the Open() method has not been called or the Close() method has been called.

Remarks

This method is equivalent to the following Visual Basic code: `MSComm1.OutBufferCount = 0`. It clears the transmit buffer, but does not affect the receive buffer.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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

Product	Versions
.NET Standard	2.0 (package-provided)

SerialPort.Dispose(Boolean) Method

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Releases the unmanaged resources used by the SerialPort and optionally releases the managed resources.

```
C#

protected override void Dispose(bool disposing);
```

Parameters

`disposing` Boolean

`true` to release both managed and unmanaged resources; `false` to release only unmanaged resources.

Exceptions

IOException

The port is in an invalid state.

-or-

An attempt to set the state of the underlying port failed. For example, the parameters passed from this SerialPort object were invalid.

Remarks

This method is called by the public Dispose() method and the Finalize() method, if it has been overridden. Dispose() invokes the protected Dispose method with the `disposing` parameter set to `true`. `Finalize` invokes Dispose with `disposing` set to `false`.

When the `disposing` parameter is `true`, this method releases all resources held by any managed objects that this SerialPort references. This method invokes the Dispose() method of each referenced object.

This method flushes and closes the stream object in the BaseStream property.

Notes to Inheritors

Dispose() can be called multiple times by other objects. When overriding Dispose(Boolean), be careful not to reference objects that have been previously disposed of in an earlier call to Dispose(). For more information about how to implement Dispose(Boolean), see Implementing a Dispose Method.

For more information about Dispose() and Finalize(), see Cleaning Up Unmanaged Resources.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.GetPortNames Method

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.Win32.cs

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

```
C#

public static string[] GetPortNames();
```

Returns

String[]

An array of serial port names for the current computer.

Exceptions

Win32Exception

The serial port names could not be queried.

Examples

The following code example uses the GetPortNames method to display serial port names to the console.

```
C#

using System;
using System.IO.Ports;

namespace SerialPortExample
{
    class SerialPortExample
    {
        public static void Main()
        {
            // Get a list of serial port names.
            string[] ports = SerialPort.GetPortNames();
            Console.WriteLine("The following serial ports were found:");

            // Display each port name to the console.
            foreach(string port in ports)
            {
                Console.WriteLine(port);
            }

            Console.ReadLine();
        }
    }
}
```

Remarks

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, HKEY_LOCAL_MACHINE\HARDWARE\DEVICEMAP\SERIALCOMM). If the registry contains stale or otherwise incorrect data then the GetPortNames method will return incorrect data.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.Open Method

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Opens a new serial port connection.

```
C#

public void Open();
```

Exceptions

UnauthorizedAccessException

Access is denied to the port.

-or-

The current process, or another process on the system, already has the specified COM port open either by a SerialPort instance or in unmanaged code.

ArgumentOutOfRangeException

One or more of the properties for this instance are invalid. For example, the Parity, DataBits, or Handshake properties are not valid values; the BaudRate is less than or equal to zero; the ReadTimeout or WriteTimeout property is less than zero and is not InfiniteTimeout.

ArgumentException

The port name does not begin with "COM".

-or-

The file type of the port is not supported.

IOException

The port is in an invalid state.

-or-

An attempt to set the state of the underlying port failed. For example, the parameters passed from this SerialPort object were invalid.

InvalidOperationException

The specified port on the current instance of the SerialPort is already open.

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. This code example is part of a larger code example provided for the SerialPort class.

C#

```
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) { }
        }
    }
```

Remarks

Only one open connection can exist per SerialPort object.

The best practice for any application is to wait for some amount of time after calling the Close method before attempting to call the Open method, as the port may not be closed instantly.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.Read Method

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Reads from the SerialPort input buffer.

Overloads

Expand table

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.

Read(Byte[], Int32, Int32)

Source: SerialPort.cs

Reads a number of bytes from the SerialPort input buffer and writes those bytes into a byte array at the specified offset.

```
C#
public int Read(byte[] buffer, int offset, int count);
```

Parameters

buffer Byte[]

The byte array to write the input to.

offset Int32

The offset in `buffer` at which to write the bytes.

count Int32

The maximum number of bytes to read. Fewer bytes are read if `count` is greater than the number of bytes in the input buffer.

Returns

Int32

The number of bytes read.

Exceptions

ArgumentNullException

The buffer passed is `null`.

InvalidOperationException

The specified port is not open.

ArgumentOutOfRangeException

The `offset` or `count` parameters are outside a valid region of the `buffer` being passed. Either `offset` or `count` is less than zero.

ArgumentException

`offset` plus `count` is greater than the length of the `buffer`.

TimeoutException

No bytes were available to read.

Remarks

If it is necessary to switch between reading text and reading binary data from the stream, select a protocol that carefully defines the boundary between text and binary data, such as manually reading bytes and decoding the data.

Because the SerialPort class buffers data, and the stream contained in the BaseStream property does not, the two might conflict about how many bytes are available to read. The BytesToRead property can indicate that there are bytes to read, but these bytes might not be accessible to the stream contained in the BaseStream property because they have been buffered to the SerialPort class.

The Read method does not block other operations when the number of bytes read equals `count` but there are still unread bytes available on the serial port.

Applies to

▼ .NET 10 (package-provided) and other versions

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

Read(Char[], Int32, Int32)

Source: SerialPort.cs

Reads a number of characters from the SerialPort input buffer and writes them into an array of characters at a given offset.

C#

```
public int Read(char[] buffer, int offset, int count);
```

Parameters

`buffer` Char[]

The character array to write the input to.

`offset` Int32

The offset in `buffer` at which to write the characters.

`count` Int32

The maximum number of characters to read. Fewer characters are read if `count` is greater than the number of characters in the input buffer.

Returns

Int32

The number of characters read.

Exceptions

ArgumentException

`offset` plus `count` is greater than the length of the buffer.

-or-

`count` is 1 and there is a surrogate character in the buffer.

ArgumentNullException

The `buffer` passed is `null`.

ArgumentOutOfRangeException

The `offset` or `count` parameters are outside a valid region of the `buffer` being passed. Either `offset` or `count` is less than zero.

InvalidOperationException

The specified port is not open.

TimeoutException

No characters were available to read.

Remarks

Use this method for reading characters from the serial port.

If it is necessary to switch between reading text and reading binary data from the stream, select a protocol that carefully defines the boundary between text and binary data, such as manually reading bytes and decoding the data.

Because the SerialPort class buffers data, and the stream contained in the BaseStream property does not, the two might conflict about how many characters are available to read. The BytesToRead property can indicate that there are characters to read, but these characters might not be accessible to the stream contained in the BaseStream property because they have been buffered to the SerialPort class.

The Read method does not block other operations when the number of bytes read equals `count` but there are still unread bytes available on the serial port.

Applies to

▼ .NET 10 (package-provided) and other versions

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.ReadByte Method

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Synchronously reads one byte from the SerialPort input buffer.

```
C#

public int ReadByte();
```

Returns

Int32

The byte, cast to an Int32, or -1 if the end of the stream has been read.

Exceptions

InvalidOperationException

The specified port is not open.

TimeoutException

The operation did not complete before the time-out period ended.

-or-

No byte was read.

Remarks

This method reads one byte.

Use caution when using ReadByte and ReadChar together. Switching between reading bytes and reading characters can cause extra data to be read and/or other unintended behavior. If it is necessary to switch between reading text and reading binary data from the stream, select a protocol that carefully defines the boundary between text and binary data, such as manually reading bytes and decoding the data.

ⓘ Note

Because the SerialPort class buffers data, and the stream contained in the BaseStream property does not, the two might conflict about how many bytes are available to read. The BytesToRead property can indicate that there are bytes to read, but these bytes might not be accessible to the stream contained in the BaseStream property because they have been buffered to the SerialPort class.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.ReadChar Method

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Synchronously reads one character from the SerialPort input buffer.

```
C#
public int ReadChar();
```

Returns

Int32

The character that was read.

Exceptions

InvalidOperationException

The specified port is not open.

TimeoutException

The operation did not complete before the time-out period ended.

-or-

No character was available in the allotted time-out period.

Remarks

This method reads one complete character based on the encoding.

Use caution when using ReadByte and ReadChar together. Switching between reading bytes and reading characters can cause extra data to be read and/or other unintended behavior. If it is necessary to switch between reading text and reading binary data from the stream, select a protocol that carefully defines the boundary between text and binary data, such as manually reading bytes and decoding the data.

Note

Because the SerialPort class buffers data, and the stream contained in the BaseStream property does not, the two might conflict about how many bytes are available to read. The BytesToRead property can indicate that there are bytes to read, but these bytes might not be accessible to the stream contained in the BaseStream property because they have been buffered to the SerialPort class.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.ReadExisting Method

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

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

C#

```
public string ReadExisting();
```

Returns

String

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

Exceptions

InvalidOperationException

The specified port is not open.

Remarks

This method returns the contents of the stream and internal buffer of the SerialPort object as a string. This method does not use a time-out. Note that this method can leave trailing lead bytes in the internal buffer, which makes the BytesToRead value greater than zero.

If it is necessary to switch between reading text and reading binary data from the stream, select a protocol that carefully defines the boundary between text and binary data, such as manually reading bytes and decoding the data.

Note

The SerialPort class buffers data, but the stream object contained in the SerialPort.BaseStream property does not. Therefore, the SerialPort object and the stream object might differ on the number of bytes that are available to read. When bytes are buffered to the SerialPort object, the BytesToRead property includes these bytes in its value; however, these bytes might not be accessible to the stream contained in the BaseStream property.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.ReadLine Method

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Reads up to the NewLine value in the input buffer.

C#

```
public string ReadLine();
```

Returns

String

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

Exceptions

InvalidOperationException

The specified port is not open.

TimeoutException

The operation did not complete before the time-out period ended.

-or-

No bytes were read.

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. This code example is part of a larger code example provided for the SerialPort class.

C#

```
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) { }
    }
}
```

Remarks

Note that while this method does not return the NewLine value, the NewLine value is removed from the input buffer.

By default, the ReadLine method will block until a line is received. If this behavior is undesirable, set the ReadTimeout property to any non-zero value to force the ReadLine method to throw a TimeoutException if a line is not available on the port.

If it is necessary to switch between reading text and reading binary data from the stream, select a protocol that carefully defines the boundary between text and binary data, such as manually reading bytes and decoding the data.

> **Note**
>
> Because the SerialPort class buffers data, and the stream contained in the BaseStream property does not, the two might conflict about how many bytes are available to read. The BytesToRead property can indicate that there are bytes to read, but these bytes might not be accessible to the stream contained in the BaseStream property because they have been buffered to the SerialPort class.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.ReadTo(String) Method

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Reads a string up to the specified `value` in the input buffer.

```
C#
```

```csharp
public string ReadTo(string value);
```

Parameters

`value` String

A value that indicates where the read operation stops.

Returns

String

The contents of the input buffer up to the specified `value`.

Exceptions

ArgumentException

The length of the `value` parameter is 0.

ArgumentNullException

The `value` parameter is `null`.

InvalidOperationException

The specified port is not open.

TimeoutException

The operation did not complete before the time-out period ended.

Remarks

This method reads a string up to the specified `value`. While the returned string does not include the `value`, the `value` is removed from the input buffer.

If it is necessary to switch between reading text and reading binary data from the stream, select a protocol that carefully defines the boundary between text and binary data, such as manually reading bytes and decoding the data.

ⓘ **Note**

Because the SerialPort class buffers data, and the stream contained in the BaseStream property does not, the two might conflict about how many bytes are available to read. The BytesToRead property can indicate that there are bytes to read, but these bytes might not be accessible to the stream contained in the BaseStream property because they have been buffered to the SerialPort class.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.Write Method

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Writes data to the serial port output buffer.

Overloads

Expand table

Write(String)	Writes the specified string to the serial port.
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)

Source: SerialPort.cs

Writes the specified string to the serial port.

C#

```
public void Write(string text);
```

Parameters

`text` String

The string for output.

Exceptions

InvalidOperationException

The specified port is not open.

ArgumentNullException

`text` is `null`.

TimeoutException

The operation did not complete before the time-out period ended.

Remarks

Use this method when you want to write a string as output to a serial port.

If there are too many bytes in the output buffer and Handshake is set to XOnXOff then the SerialPort object may raise a TimeoutException while it waits for the device to be ready to accept more data.

By default, SerialPort uses ASCIIEncoding to encode the characters. ASCIIEncoding encodes all characters greater than 127 as (char)63 or '?'. To support additional characters in that range, set Encoding to UTF8Encoding, UTF32Encoding, or UnicodeEncoding.

Applies to

▼ .NET 10 (package-provided) and other versions

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

Write(Byte[], Int32, Int32)

Source: SerialPort.cs

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

```
C#

public void Write(byte[] buffer, int offset, int count);
```

Parameters

`buffer` Byte[]

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

`offset` Int32

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

`count` Int32

The number of bytes to write.

Exceptions

ArgumentNullException

The `buffer` passed is `null`.

InvalidOperationException

The specified port is not open.

ArgumentOutOfRangeException

The `offset` or `count` parameters are outside a valid region of the `buffer` being passed. Either `offset` or `count` is less than zero.

ArgumentException

`offset` plus `count` is greater than the length of the `buffer`.

TimeoutException

The operation did not complete before the time-out period ended.

Remarks

Use this method when you want to write to a byte buffer to create output to a serial port.

If there are too many bytes in the output buffer and Handshake is set to XOnXOff then the SerialPort object may raise a TimeoutException while it waits for the device to be ready to accept more data.

Applies to

▼ .NET 10 (package-provided) and other versions

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

Write(Char[], Int32, Int32)

Source: SerialPort.cs

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

```
C#

public void Write(char[] buffer, int offset, int count);
```

Parameters

`buffer` Char[]

The character array that contains the data to write to the port.

`offset` Int32

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

`count` Int32

The number of characters to write.

Exceptions

ArgumentNullException

The `buffer` passed is `null`.

InvalidOperationException

The specified port is not open.

ArgumentOutOfRangeException

The `offset` or `count` parameters are outside a valid region of the `buffer` being passed. Either `offset` or `count` is less than zero.

ArgumentException

`offset` plus `count` is greater than the length of the `buffer`.

TimeoutException

The operation did not complete before the time-out period ended.

Remarks

Use this method when you want to write to a character buffer to create output to a serial port.

If there are too many bytes in the output buffer and Handshake is set to XOnXOff then the SerialPort object may raise a TimeoutException while it waits for the device to be ready to accept more data.

By default, SerialPort uses ASCIIEncoding to encode the characters. ASCIIEncoding encodes all characters greater than 127 as (char)63 or '?'. To support additional characters in that range, set Encoding to UTF8Encoding, UTF32Encoding, or UnicodeEncoding.

Applies to

▼ .NET 10 (package-provided) and other versions

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.WriteLine(String) Method

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Writes the specified string and the NewLine value to the output buffer.

C#

```csharp
public void WriteLine(string text);
```

Parameters

`text` String

The string to write to the output buffer.

Exceptions

ArgumentNullException

The `text` parameter is `null`.

InvalidOperationException

The specified port is not open.

TimeoutException

The WriteLine(String) method could not write to the stream.

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. This code example is part of a larger code example provided for the SerialPort class.

C#

```csharp
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) { }
        }
    }
```

Remarks

If there are too many bytes in the input buffer and Handshake is set to XOnXOff then the SerialPort object may raise a TimeoutException while it waits for the device to be ready to accept more data.

The written output includes the NewLine string.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.DataReceived Event

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Indicates that data has been received through a port represented by the SerialPort object.

C#

```
public event System.IO.Ports.SerialDataReceivedEventHandler DataReceived;
```

Event Type

SerialDataReceivedEventHandler

Examples

This example adds a SerialDataReceivedEventHandler to DataReceived to read all the available data received on the COM1 port. Note that to test this code it is necessary to have hardware attached to COM1 that will send data.

C#

```
using System;
using System.IO.Ports;

class PortDataReceived
{
    public static void Main()
    {
        SerialPort mySerialPort = new SerialPort("COM1");

        mySerialPort.BaudRate = 9600;
        mySerialPort.Parity = Parity.None;
        mySerialPort.StopBits = StopBits.One;
        mySerialPort.DataBits = 8;
        mySerialPort.Handshake = Handshake.None;
        mySerialPort.RtsEnable = true;

        mySerialPort.DataReceived += new
SerialDataReceivedEventHandler(DataReceivedHandler);
```

```
        mySerialPort.Open();

        Console.WriteLine("Press any key to continue...");
        Console.WriteLine();
        Console.ReadKey();
        mySerialPort.Close();
    }

    private static void DataReceivedHandler(
                        object sender,
                        SerialDataReceivedEventArgs e)
    {
        SerialPort sp = (SerialPort)sender;
        string indata = sp.ReadExisting();
        Console.WriteLine("Data Received:");
        Console.Write(indata);
    }
}
```

Remarks

Data events can be caused by any of the items in the SerialData enumeration. Because the operating system determines whether to raise this event or not, not all parity errors may be reported.

The DataReceived event is also raised if an Eof character is received, regardless of the number of bytes in the internal input buffer and the value of the ReceivedBytesThreshold property.

PinChanged, DataReceived, and ErrorReceived events may be called out of order, and there may be a slight delay between when the underlying stream reports the error and when the event handler is executed. Only one event handler can execute at a time.

The DataReceived event is not guaranteed to be raised for every byte received. Use the BytesToRead property to determine how much data is left to be read in the buffer.

The DataReceived event is raised on a secondary thread when data is received from the SerialPort object. Because this event is raised on a secondary thread, and not the main thread, attempting to modify some elements in the main thread, such as UI elements, could raise a threading exception. If it is necessary to modify elements in the main Form or Control, post change requests back using Invoke, which will do the work on the proper thread.

For more information about handling events, see Handling and Raising Events.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.ErrorReceived Event

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Indicates that an error has occurred with a port represented by a SerialPort object.

```
C#

public event System.IO.Ports.SerialErrorReceivedEventHandler ErrorReceived;
```

Event Type

SerialErrorReceivedEventHandler

Remarks

Error events can be caused by any of the items in the SerialError enumeration. Because the operating system determines whether to raise this event or not, not all parity errors may be reported.

PinChanged, DataReceived, and ErrorReceived events may be called out of order, and there may be a slight delay between when the underlying stream reports the error and when code can when the event handler is executed. Only one event handler can execute at a time.

If a parity error occurs on the trailing byte of a stream, an extra byte will be added to the input buffer with a value of 126.

The ErrorReceived event is raised on a secondary thread when an error is received from the SerialPort object. Because this event is raised on a secondary thread, and not the main thread, attempting to modify some elements in the main thread, such as UI elements, could raise a threading exception. If it is necessary to modify elements in the main Form or Control, post change requests back using Invoke, which will do the work on the proper thread.

For more information about handling events, see Handling and Raising Events.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

SerialPort.PinChanged Event

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: SerialPort.cs

Indicates that a non-data signal event has occurred on the port represented by the SerialPort object.

C#

```csharp
public event System.IO.Ports.SerialPinChangedEventHandler PinChanged;
```

Event Type

SerialPinChangedEventHandler

Remarks

Serial pin changed events can be caused by any of the items in the SerialPinChange enumeration. Because the operating system determines whether to raise this event or not, not all parity errors may be reported. As part of the event, the new value of the pin is set.

The PinChanged event is raised when a SerialPort object enters the BreakState, but not when the port exits the BreakState. This behavior does not apply to other values in the SerialPinChange enumeration.

PinChanged, DataReceived, and ErrorReceived events may be called out of order, and there may be a slight delay between when the underlying stream reports the error and when the event handler is executed. Only one event handler can execute at a time.

The PinChanged event is raised on a secondary thread. Because this event is raised on a secondary thread, and not the main thread, attempting to modify some elements in the main thread, such as UI elements, could raise a threading exception. If it is necessary to modify elements in the main Form or Control, post change requests back using Invoke, which will do the work on the proper thread.

For more information about handling events, see Handling and Raising Events.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)
.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 Standard	2.0 (package-provided)

StopBits Enum

Definition

Namespace: System.IO.Ports

Assembly: System.IO.Ports.dll

Package: System.IO.Ports v10.0.0-preview.5.25277.114

Source: StopBits.cs

Specifies the number of stop bits used on the SerialPort object.

```
C#

public enum StopBits
```

Inheritance Object → ValueType → Enum → StopBits

Fields

Expand table

Name	Value	Description
None	0	No stop bits are used. This value is not supported by the StopBits property.
One	1	One stop bit is used.
Two	2	Two stop bits are used.
OnePointFive	3	1.5 stop bits are used.

Examples

The following example shows how to set the StopBits property to `One`.

```
C#

SerialPort mySerialPort = new SerialPort("COM1");

mySerialPort.BaudRate = 9600;
mySerialPort.Parity = Parity.None;
mySerialPort.StopBits = StopBits.One;
mySerialPort.DataBits = 8;
mySerialPort.Handshake = Handshake.None;
mySerialPort.RtsEnable = true;
```

The following code example displays the possible values of the StopBits enumeration to the console, then prompts the user to choose one. This code example is part of a larger code example provided for the SerialPort class.

C#

```
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);
}
```

Remarks

You use this enumeration when setting the value of the StopBits property on the SerialPort class. Stop bits separate each unit of data on an asynchronous serial connection. They are also sent continuously when no data is available for transmission.

The SerialPort class throws an ArgumentOutOfRangeException exception when you set the StopBits property to None.

Applies to

Product	Versions
.NET	8 (package-provided), 9 (package-provided), 10 (package-provided)

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 Standard	2.0 (package-provided)