
Reference Manual

Franson SerialTools .NET

Note! The web version will always be most up to date! Please visit <http://franson.com/serialtools>

The Reference Manual gives you detailed information about all classes used in SerialTools.

If your application need to read and write data to a serial port, start reading about the [Port](#) class.

If you want to create and use virtual ports, start by the [VPort](#) class.

A **License key** is necessary to use SerialTools. During development the key found [here](#) can be used. To distribute the component as part of your application you need to [purchase](#) a license. The `License.LicenseKey` property must be set to a valid license key by your application or else the component will refuse to work properly.

Technical support can be found in the user forum. We will constantly monitor and answer questions in the forum. The forum also includes frequently asked question (FAQ).

[FAQ for SerialTools](#)

[Browse the Technical support forum!](#)

[Search the Technical support forum!](#)

| Class | Class | Class | Enum | Enum | Enum |
|--------------------------------|-----------------------------------|----------------------------|---------------------------|-------------------------------|------------------------|
| VPort | Port | License | Handshake | StopBits | Parity |
| Properties | Properties | Properties | Values | Values | Values |
| Created | List | LicenseKey | None | One | No |
| ComPort | Enabled | | RTS | OneAndOneHalf | Odd |
| | BaudRate | | DTR | Two | Even |
| Methods | ComPort | | XonXoff | | Mark |
| DataFromPort | | | | | Space |
| DataToPort | Methods | | | | |
| PortStatus | Read | | | | |
| Dispose | Write | | | | |
| | Purge | | | | |
| Events | Dispose | | | | |
| OnDataFromPort | | | | | |
| OnOpened | ByteArrayToString | | | | |
| OnClosed | StringToByteArray | | | | |

More Properties Events

[StartTrigger](#) [OnRead](#)
[EndTrigger](#) [OnWritten](#)
[BufferSize](#) [OnDSR](#)
[Timeout](#) [OnCTS](#)
 [OnRI](#)
[MultiThreading](#) [OnDCD](#)
[Parent](#) [OnForceClose](#)
[NoEvents](#)

More Properties

[Parity](#)
[StopBits](#)
[ByteSize](#)
[StartTrigger](#)
[EndTrigger](#)
[BufferSize](#)
[Timeout](#)

[DSR](#)
[CTS](#)
[RI](#)
[DCD](#)

[Handshake](#)
[RTS](#)
[DTR](#)
[Break](#)
[InfraRed](#)

[MultiThreading](#)
[Parent](#)
[NoEvents](#)

Port

SerialTools **v1.00** and later

The `Port` class contains the core functionality to access the serial port.

Set a baudrate using `Port . BaudRate` and the port number using `Port . ComPort`.
`Port . Enabled` opens the specified serial port for reading and writing.

Data is **read** from the port with the `Port . OnRead` event or the `Port . Read` method and you **write** data using `Port . Write`.

By using `Port . StartTrigger` and `Port . EndTrigger` SerialTools can help you with simple **parsing of the data**.

`Port . Timeout` can be used for timeout handling.

If you want to read and write **binary data** take a closer look at `Port . ByteArrayToString` and `Port . StringToByteArray`.

For more advanced control of the serial port and handshaking options see methods and properties to the left.

Port.Enabled

SerialTools **v1.00** and later

Opens or closes the serial port specified by `Port . ComPort`.

VB.NET sample:

```
Dim objPort As New SerialNET.Port

objPort.BaudRate = 4800
objPort.ComPort = 2
objPort.Enabled = True           ' Starts serial port.
objPort.Write("Some data written to serial port")
```

NOTE! You must disable the port by setting `Enabled` to false before you unload the form you got the `Port . OnRead` event handler on. And you must call `Port . Dispose` after you have finished using the port object or else wont the application exit properly.

See also `Port . OnRead` and `Port . Write`.

Syntax

object.Enabled = Value

Value = object.Enabled

| Part | Type | Description |
|---------------|----------------|---|
| <i>object</i> | SerialNET.Port | An object that evaluates to an SerialNET.Port object. |
| <i>Value</i> | bool | true |

Port.BaudRate

SerialTools **v1.00** and later

Sets the baud rate of the serial port. The default value is 9600 baud.

See also `Port . ComPort` and `Port . Enabled`.

Syntax

object.BaudRate = Value

Value = object.BaudRate

| Part | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------------|---|
| <i>object</i> | SerialNET.Port | An object that evaluates to an SerialNET.Port object. |
| <i>Value</i> | int | Baud rate. |

Port.ComPort

SerialTools v1.00 and later

Specifies which serial port to use. The default value is 1.

On Windows CE / Pocket PC this is a value between 0 and 9. Note that COM0: is a possible COM port in Windows CE!

For Windows this is a value between 1 and 255.

See also Port . [BaudRate](#) and Port . [Enabled](#).

Syntax

object.ComPort = *Value*

Value = *object*.ComPort

| Part | Type | Description |
|---------------|----------------|---|
| <i>object</i> | SerialNET.Port | An object that evaluates to an SerialNET.Port object. |
| <i>Value</i> | int | Comm port. |

Port.Parity

SerialTools v1.00 and later

Sets parity. Default value is Parity.No.

See also the [Parity](#) Enumeration.

Syntax

object.Parity = *Value*

Value = *object*.Parity

| Part | Type | Description |
|---------------|-------------------------------|---|
| <i>object</i> | SerialNET.Port | An object that evaluates to an SerialNET.Port object. |
| <i>Value</i> | Parity (Enum) | Type of parity. |

Port.StopBits

SerialTools v1.00 and later

Sets number of stop bits. Default value is StopBits.One.

See also the [StopBits](#) Enumeration.

Syntax*object.StopBits = Value**Value = object.StopBits*

| Part | Type | Description |
|---------------|---------------------------------|---|
| <i>object</i> | SerialNET.Port | An object that evaluates to an SerialNET.Port object. |
| <i>Value</i> | StopBits (Enum) | Number of stop bits. |

Port.ByteSizeSerialTools **v1.00** and later

Sets number of bits in one byte, can be set ot 7 or 8. Default value is 8.

Syntax*object.ByteSize = Value**Value = object.ByteSize*

| Part | Type | Description |
|---------------|----------------|---|
| <i>object</i> | SerialNET.Port | An object that evaluates to an SerialNET.Port object. |
| <i>Value</i> | int | Number of bits in one byte. |

Port.StartTriggerSerialTools **v1.00** and later

Sets start trigger. Default value is null.

See Port . [OnRead](#) for how this functionallity is used.**Syntax***object.StartTrigger = Value**Value = object.StartTrigger*

| Part | Type | Description |
|---------------|----------------|---|
| <i>object</i> | SerialNET.Port | An object that evaluates to an SerialNET.Port object. |
| <i>Value</i> | string | string to trigger reception of data on. |

Port.EndTriggerSerialTools **v1.00** and later

Sets end trigger. Default value is null.

See Port . [OnRead](#) for how this functionallity is used.

Syntax

object.EndTrigger = Value

Value = object.EndTrigger

| Part | Type | Description |
|---------------|----------------|---|
| <i>object</i> | SerialNET.Port | An object that evaluates to an SerialNET.Port object. |
| <i>Value</i> | string | End trigger. |

Port.BufferSize

SerialTools **v1.00** and later

Sets the input buffer size. Default value is 0 (changed from default 100 to default 0 in v1.11).

BufferSize returns the number of bytes in the serial buffer that are still not parsed.

NOTE! If you set BufferSize to a value different from 0 then OnRead will not be called until BufferSize bytes of data has been received. By default BufferSize is set to 0, which means this feature is disabled.

See Port . [OnRead](#) for how this functionality is used.

Syntax

object.BufferSize = Value

Value = object.BufferSize

| Part | Type | Description |
|---------------|----------------|---|
| <i>object</i> | SerialNET.Port | An object that evaluates to an SerialNET.Port object. |
| <i>Value</i> | int | Input buffer size. |

Port.Timeout

SerialTools **v1.00** and later

Time in **milliseconds** before a timeout event is generated. A value of 0 means that the timeout functionality is disabled. Default value is 0.

See Port . [OnRead](#) and Port . [OnWritten](#) for more info.

Syntax

object.Timeout = Value

Value = object.Timeout

| Part | Type | Description |
|---------------|----------------|---|
| <i>object</i> | SerialNET.Port | An object that evaluates to an SerialNET.Port object. |
| <i>Value</i> | int | Timeout in milliseconds. 0 to disable. |

Port.DSR

SerialTools v1.00 and later

Reads the value of DSR (Data Set Ready).

See Port . [OnDSR](#) for more info.

Syntax

Value = object.DSR

| Part | Type | Description |
|---------------|----------------|---|
| <i>object</i> | SerialNET.Port | An object that evaluates to an SerialNET.Port object. |
| <i>Value</i> | bool | State of DSR. |

Port.CTS

SerialTools v1.00 and later

Reads the value of CTS (Clear To Send).

See Port . [OnCTS](#) for more info.

Syntax

Value = object.CTS

| Part | Type | Description |
|---------------|----------------|---|
| <i>object</i> | SerialNET.Port | An object that evaluates to an SerialNET.Port object. |
| <i>Value</i> | bool | State of CTS. |

Port.RI

SerialTools v1.00 and later

Reads the value of RI (Ring Indicator).

See Port . [OnRI](#) for more info.

Syntax

Value = object.RI

| Part | Type | Description |
|---------------|----------------|---|
| <i>object</i> | SerialNET.Port | An object that evaluates to an SerialNET.Port object. |
| <i>Value</i> | bool | State of RI. |

Port.DCD

SerialTools v1.00 and later

Reads the value of DCD (Data Carrier Detected).

See `Port.OnDCD` for more info.

Syntax

Value = *object*.DCD

| Part | Type | Description |
|---------------|----------------|---|
| <i>object</i> | SerialNET.Port | An object that evaluates to an SerialNET.Port object. |
| <i>Value</i> | bool | State of DCD. |

Port.Handshake

SerialTools v1.00 and later

Sets low level handshake method. Default value is `Handshake.None`.

RTS/CTS. Low level method of controlling sending of data. Set this property to `Handshake.RTS` to use RTS/CTS.

DSR/DTR. Low level method of controlling reception of data. Set this property to `Handshake.DSR` to use DSR/DTR.

XON/XOFF. (v1.20 and later) Set this property to `Handshake.XonXoff`

The values can be or'ed together.

See also the [Handshake](#) enumeration.

Syntax

object.Handshake = *Value*

Value = *object*.Handshake

| Part | Type | Description |
|---------------|----------------------------------|---|
| <i>object</i> | SerialNET.Port | An object that evaluates to an SerialNET.Port object. |
| <i>Value</i> | Handshake (Enum) | Type of used low level handshake. |

Port.RTS

SerialTools v1.00 and later

Sets the value of RTS (Request To Send). Cannot be set if `Port.Handshake` is set to `Handshake.RTS`

Syntax

object.RTS = Value

| Part | Type | Description |
|---------------|----------------|---|
| <i>object</i> | SerialNET.Port | An object that evaluates to an SerialNET.Port object. |
| <i>Value</i> | bool | State of RTS. |

Port.DTR

SerialTools **v1.00** and later

Sets the value of DTR (Data Terminal Ready). Cannot be set if Port . [Handshake](#) is set to Handshake . DTR

Note! Some devices and terminal programs demands this property to be set to true or else they will not transmit any data to be received by SerialTools. In SerialTools 1.20 (and later) DTR is set to true by default.

Syntax

object.DTR = Value

| Part | Type | Description |
|---------------|----------------|---|
| <i>object</i> | SerialNET.Port | An object that evaluates to an SerialNET.Port object. |
| <i>Value</i> | bool | State of DTR. |

Port.OnRead

SerialTools **v1.00** and later

Event that is raised when data is received on the serial port.

OnRead is called when data has been received on the serial port.

Timeout. If Port . [Timeout](#) is set to a value different from 0 (default) OnRead is called with null (C#) or Nothing (VB.NET) as argument if no data has been received during the time intervall specified by Timeout. Timeout is specified in miliseconds.

Buffer size. Use Port . [BufferSize](#) to specify how many bytes should be received before OnRead is called (default 0 - disabled).

Simple parsing. If Port . [EndTrigger](#) is set to a value different from null (default). OnRead will be called when the specified string is found in the input from the serial port. This is very usefull if you got data with a known format coming in to the serial port. If EndTrigger is used BufferSize is ignored.

If Port . [StartTrigger](#) is set to a value different from null (default). All data is ignored until the specified string is found in the serial port input. If both StartTrigger and EndTrigger is used only data between (including the triggers themself) the triggers will be passed on to

OnRead. If EndTrigger is null, OnRead will be called when BufferSize bytes has been received. If EndTrigger is null StartTrigger must be "retriggered" each time OnRead is called or else OnRead will be called with data each time the input buffer is full.

OnRead is called with a string as argument. To convert this string to binary data see Port . [StringToByteArray](#) and to convert from binary data to a string see Port . [ByteArrayToString](#)

See also Port . [OnForceClose](#) for more info on how to handle Bluetooth interruption.

See also Port . [Enabled](#) and Port . [Read](#).

Syntax

obj_OnRead (*Value*)

| Part | Type | Description |
|--------------|--------|---------------------------------|
| <i>obj</i> | | Event source. |
| <i>Value</i> | string | Data read from the serial port. |

Port.OnDSR

SerialTools **v1.00** and later

Called when DSR (Data Set Ready) changes value. DSR is traditionally used by modems when they are ready to transmitt data from the modem to the computer. The computer then answers by setting DTR to true. See Port . [Handshake](#) for more info on DTR/DSR handshake.

For most cases you don't need to bother about this event.

See also Port . [DSR](#).

Syntax

obj_OnDSR (*State*)

| Part | Type | Description |
|--------------|------|---------------|
| <i>obj</i> | | Event source. |
| <i>State</i> | bool | State of DSR. |

Port.OnCTS

SerialTools **v1.00** and later

Event that is called when CTS (Clear To Send) changes value. CTS is traditionally used by modems to answer a RTS (Request To Send) request from the computer.

See Port . [Handshake](#) for more info on RTS/CTS handshake.

For most cases you don't need to bother about this event.

See also Port . [CTS](#).

Syntax

obj_OnCTS (*State*)

| Part | Type | Description |
|--------------|------|---------------|
| <i>obj</i> | | Event source. |
| <i>State</i> | bool | State of CTS. |

Port.OnRI

SerialTools **v1.00** and later

Event that is called when RI (Ring Indicator) changes value. RI is traditionally used by modems to make the computer aware of that someone is calling the modem.

For most cases you don't need to bother about this event.

See also Port . [RI](#).

Syntax

obj_OnRI (*State*)

| Part | Type | Description |
|--------------|------|---------------|
| <i>obj</i> | | Event source. |
| <i>State</i> | bool | State of RI. |

Port.OnDCD

SerialTools **v1.00** and later

Event that is called when DCD (Data Carrier Detected) changes value. DCD is traditionally used by modems to make the computer aware of that a valid carrier is found on the phone line.

DCD is also known as RLSD (Receive Line Signal Detected). For most cases you don't need to bother about this event.

See also Port . [DCD](#).

Syntax

obj_OnDCD (*State*)

| Part | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|--------------|------|---------------|
| <i>obj</i> | | Event source. |
| <i>State</i> | bool | State of DCD. |

Port.OnForceClose

SerialTools v1.11 and later

This event is very useful to handle Bluetooth errors, removed USB serial adapters and similar none static serial ports.

Event that is called when a serial port is closed in an unexpected way. If you are using a Bluetooth serial port this will happen if the Bluetooth device is turned off or gets out of reach. Before this event is called the port has been closed. To reestablish the connection you need to reopen the port.

OnForceClose is called with the Win32 error code that caused serial port to fail. When a Bluetooth device is turned of or gets out of reach this value is typically 31.

Not all BT/USB serial port drivers generate a Win32 error when the device is removed from the system, therefor should you application also check for timeouts.

Syntax

`obj_OnForceClose (Value)`

| Part | Type | Description |
|--------------|------|---|
| <i>obj</i> | | Event source. |
| <i>Value</i> | int | Win32 error code that was the cause of the closed port. |

Port.Write

SerialTools v1.00 and later

Writes data to the serial port. Write returns after data is written to the port. Write takes a string as an argument, to convert binary data to a string see `Port.ByteArrayToString`.

News in v1.20

Write returns the number of bytes actually written. If write fails zero is returned. Write will time out and return after the time specified in `Port.Timeout`

If an event handler for `Port.OnWritten` is specified, then Write will return instantly with zero as return code, and OnWritten will be called with the number of bytes actually written to the serial port driver. If the attempt to write timed out OnWritten will have zero as argument.

See also `Port.Enabled` and `Port.OnWritten`.

Syntax

`BytesWritten = object.Write (Value)`

| Part | Type | Description |
|---------------------|----------------|---|
| <i>object</i> | SerialNET.Port | An object that evaluates to an SerialNET.Port object. |
| <i>Value</i> | string | Data to be written to the serial port. |
| <i>BytesWritten</i> | int | Bytes written. |

Port.Purge

SerialTools v1.00 and later

Resets all data in input and output buffers.

New in 1.20

Now the in and out buffer can be purged separately.

Syntax

object.Purge ()

object.Purge (InBuffer, OutBuffer)

| Part | Type | Description |
|------------------|----------------|---|
| <i>object</i> | SerialNET.Port | An object that evaluates to an SerialNET.Port object. |
| <i>InBuffer</i> | int | Purge in buffer if true. |
| <i>OutBuffer</i> | int | Purge out buffer if true. |

Port.Read

SerialTools v1.10 and later

Reads data from the serialport. Cannot be used if an event handler is assigned to Port . [OnRead](#) (.NET) or Port . [NoEvents](#) is set to true (ActiveX).

Read has two versions. The first takes a BufferSize and TimeOut as arguments. Setting BufferSize to 0 will return all data in serial buffer. But if the serial buffer is empty it will wait until something has arrived. TimeOut determines how long to wait (in miliseconds). Set TimeOut to 0 to disable time out.

The second version takes StartTrigger, EndTrigger and TimeOut as arguments. Read will parse the serial buffer and return data according to the rules specified for Port . [OnRead](#). If no matching data is found in the serial buffer TimeOut determines how long to wait before returning null / Nothing.

On time out Read() always returns null / Nothing.

If a Bluetooth connection is interrupted during Read, an exception is thrown and the port is closed. See the .NET Compact Framework samples for details. A Bluetooth connection is typically interrupted if the Bluetooth device is turned of or if it gets out of reach. See also Port . [OnForceClose](#) for more info on how to handle Bluetooth interruption.

Syntax 1

Value = object.Read (BufferSize, Timeout)

| Part | Type | Description |
|-------------------|----------------|---|
| <i>object</i> | SerialNET.Port | An object that evaluates to an SerialNET.Port object. |
| <i>Value</i> | string | Data received from serial port. Nothing/null if no data received. |
| <i>BufferSize</i> | int | See OnRead for details. |
| <i>Timeout</i> | int | See OnRead for details. |

Syntax 2

Value = object.Read (StartTrigger, EndTrigger, Timeout)

| Part | Type | Description |
|---------------------|----------------|---|
| <i>object</i> | SerialNET.Port | An object that evaluates to an SerialNET.Port object. |
| <i>Value</i> | string | Data received from serial port. Nothing/null if no data received. |
| <i>StartTrigger</i> | string | See OnRead for details. |
| <i>EndTrigger</i> | string | See OnRead for details. |
| <i>Timeout</i> | int | See OnRead for details. |

Port.ByteArrayToString

SerialTools v1.10 and later

If you want to write anything else than ascii strings (ascii value equals 1-127) you first need to create a byte array and then convert the array to a string using this **static** function. The string can then be used in Port.[Write](#), Port.[StartTrigger](#) and Port.[EndTrigger](#).

VB.NET sample:

```
Dim objPort As New SerialNET.Port

objPort.BaudRate = 4800
objPort.ComPort = 2
objPort.Enabled = True          ' Starts serial port.

Dim binary data As Byte() = {&H81, &H82}

' Write binary data
objPort.Write(SerialNET.Port.ByteArrayToString(binary data))
```

Note that it will not work to build your own "binary string" using chr(). You must use this function!

VB.NET sample (which **WILL NOT WORK**):

```
Dim objPort As New SerialNET.Port

objPort.BaudRate = 4800
```

```
objPort.ComPort = 2
objPort.Enabled = True          ' Starts serial port.

Dim string_data As String
string_data = Chr(&H81) & Chr(&H82)

' Write binary data, WILL NOT WORK!!!
objPort.Write(string_data)
```

See also `Port` . [StringToByteArray](#).

Syntax

`str` = `SerialNET.Port.ByteArrayToString (ByteArray)`

| Part | Type | Description |
|------------------------|--------|-----------------------------------|
| <code>str</code> | string | Byte array converted to a string. |
| <code>ByteArray</code> | byte[] | Byte array to be converted. |

Port.StringToByteArray

SerialTools v1.10 and later

If you want to read anything else than ascii strings (that is ascii value 0-127) you need to convert the received string to a byte array using this **static** function. The string has been received by `Port` . [Read](#) or `Port` . [OnRead](#).

VB.NET sample:

```
Dim objPort As New SerialNET.Port

objPort.BaudRate = 4800
objPort.ComPort = 2
objPort.Enabled = True          ' Starts serial port.

Dim binary_data() As Byte
Dim string_data As String

string_data = objPort.Read(0, 5000)
' Received binary data
binary_data = SerialNET.Port.StringToByteArray(string_data)
```

Note that it will not work to build your own byte array, e.g. using `String.ToCharArray()`. You must use this function.

VB.NET sample (which **WILL NOT WORK**):

```
Dim objPort As New SerialNET.Port

objPort.BaudRate = 4800
objPort.ComPort = 2
objPort.Enabled = True          ' Starts serial port.

Dim binary_data() As Char
Dim string_data As String

string_data = objPort.Read(0, 5000)
```

```
' Received binary data !! WILL NOT WORK !!  
binary data = string data.ToCharArray()
```

See also Port . [ByteArrayToString](#).

Syntax

ByteArray = SerialNET.Port.StringToByteArray (*Str*)

| Part | Type | Description |
|------------------|--------|-------------------------------------|
| <i>ByteArray</i> | byte[] | Byte array converted from a string. |
| <i>Str</i> | string | string to be converted. |

Port.MultiThreading

SerialTools v1.10 and later

Note! in SerialTools .NET 1.20 this property does not longer exists and is replaced by Port . [Parent](#).

If you are making a form based application you should not care about this property.

If set to false (default) all events will be marshaled to the main thread. This requires that you add the Port control to the form. See the samples for how to do that.

If set to true events will be called with a new thread (or any thread from the thread pool). If you are making an application without a form, you need to set this property to true and be aware of that you event handlers will be called in a new thread.

Syntax

object.MultiThreading = *Value*

Value = *object*.MultiThreading

| Part | Type | Description |
|---------------|----------------|---|
| <i>object</i> | SerialNET.Port | An object that evaluates to an SerialNET.Port object. |
| <i>Value</i> | bool | Multi threading on or off. |

Port.InfraRed

SerialTools v1.20 and later

Enables communication with a IR port on Windows CE.

Syntax

object.InfraRed = *Value*

| Part | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------------|---|
| <i>object</i> | SerialNET.Port | An object that evaluates to an SerialNET.Port object. |
| <i>Value</i> | bool | Enable/Disable IR on Windows CE. |

Port.Break

SerialTools v1.20 and later

Inserts and removes break for serial port.

Syntax

object.Break = *Value*

| Part | Type | Description |
|---------------|----------------|---|
| <i>object</i> | SerialNET.Port | An object that evaluates to an SerialNET.Port object. |
| <i>Value</i> | bool | Enable/Disable break. |

Port.Parent

SerialTools v1.20 and later

This property is only available in SerialTools .NET (not ActiveX)

If set to null/Nothing (default) then events will be called in a separate thread (multi-threading).

If you are making a form based application you should set this property to your form (this/Me) pointer. If you are not using events (no OnRead, OnWritten, etc..) you do not need to care about this property.

Syntax

object.Parent = *Value*

Value = *object*.Parent

| Part | Type | Description |
|---------------|------------------------------|---|
| <i>object</i> | SerialNET.Port | An object that evaluates to an SerialNET.Port object. |
| <i>Value</i> | System.Windows.Forms.Control | Control to marshal events to. |

Port.OnWritten

SerialTools v1.20 and later

Called when data has been successfully written to the serial driver using Port .[Write](#) or the write operation has timed out.

Note that this does not mean the data has been written to the serial port, only that it has been successfully passed on to the serial port driver. However in most cases this is the same thing.

Syntax

obj_OnWritten (*BytesWritten*)

| Part | Type | Description |
|---------------------|------|---|
| <i>obj</i> | | Event source. |
| <i>BytesWritten</i> | int | Number of bytes written. Zero on error. |

Port.Dispose

SerialTools v1.00 and later

.NET only!

Must be called when you have finished using the object. If not called you application will not unload.

Syntax

object.Dispose ()

| Part | Type | Description |
|---------------|----------------|---|
| <i>object</i> | SerialNET.Port | An object that evaluates to an SerialNET.Port object. |

Port.List

SerialTools v1.20 and later

Port enumeration. Returns a byte array of all available ports in the system. Very useful when making a dropdown or similar for the user to select a serial port from.

In .NET this is a **static** method.

Syntax

list = SerialNET.Port.List

| Part | Type | Description |
|-------------|--------|---|
| <i>list</i> | byte[] | Byte array containing the index of all available ports in the system. |

Port.NoEvents

SerialTools v1.20 and later

ActiveX only. Set this to true (default false) if you wont use events.

If NoEvents is set to true you can use Port . [Read](#).

If NoEvents is set to false (default) you can use the events Port . [OnRead](#), Port . [OnWritten](#), etc.

Syntax

object.NoEvents = Value
Value = object.NoEvents

| Part | Type | Description |
|---------------|----------------|---|
| <i>object</i> | SerialNET.Port | An object that evaluates to an SerialNET.Port object. |
| <i>Value</i> | bool | Events on or off. |

VPort

SerialTools **v2.00** and later

The VPort class contains the core functionality to create and use virtual serial ports. To access and use a normal serial port you **do not** use this class. Rather this class creates a new "virtual" serial port on your computer. This new "virtual" serial port can any 3rd party application connect to as it was a normal serial port connected to your computer. In this way you can send and receive data from a 3rd party application and emulate hardware, null modems, GPS receivers, scales, yes anything normally connected to a serial port.

A good way to view and understand a virtual serial port is to think of it as "the other side" of a serial port.

Set port number using VPort . [ComPort](#). VPort . [Created](#) creates the specified serial port.

Data is **read** from the port with the VPort . [OnDataFromPort](#) event or the VPort . [DataFromPort](#) method and you **write** data using VPort . [DataToPort](#).

By using VPort . [StartTrigger](#) and VPort . [EndTrigger](#) SerialTools can help you with simple **parsing of the data**.

VPort . [Timeout](#) can be used for timeout handling.

If you want to read and write **binary data** take a closer look at Port . [ByteArrayToString](#) and Port . [StringToByteArray](#).

It can be of interest to know if an application has opened a virtual port or not. If it is not opened there is no reason to send data to it for example. There are two events handling this VPort . [OnOpened](#) and VPort . [OnClosed](#)

VPort.Created

SerialTools **v2.00** and later

Creates or destroys the virtual serial port specified by VPort . [ComPort](#).

VB.NET sample:

```
Dim objVPort As New SerialNET.VPort
objVPort.ComPort = 4
```

```
objVPort.Created = True ' Creates serial port.
```

NOTE! You must destroy the port by setting `Created` to false before you unload the form you got the `VPort.OnDataFromPort` event handler on. And you must call `VPort.Dispose` after you have finished using the port object or else wont the application exit properly.

See also `VPort.OnDataFromPort` and `VPort.DataToPort`.

Syntax

object.Created = Value

Value = object.Created

| Part | Type | Description |
|---------------|-----------------|--|
| <i>object</i> | SerialNET.VPort | An object that evaluates to an SerialNET.VPort object. |
| <i>Value</i> | bool | true |

VPort.ComPort

SerialTools **v2.00** and later

Specifies which serial port to create. The default value is 0.

On Windows CE / Pocket PC this is a value between 0 and 9. Note that COM0: is a possible COM port in Windows CE!

For Windows this is a value between 1 and 255.

See also `VPort.Created`.

Syntax

object.ComPort = Value

Value = object.ComPort

| Part | Type | Description |
|---------------|-----------------|--|
| <i>object</i> | SerialNET.VPort | An object that evaluates to an SerialNET.VPort object. |
| <i>Value</i> | int | Comm port. |

VPort.StartTrigger

SerialTools **v2.00** and later

Sets start trigger. Default value is null.

See `VPort.OnDataFromPort` for how this functionality is used.

Syntax

object.StartTrigger = Value
Value = object.StartTrigger

| Part | Type | Description |
|---------------|-----------------|--|
| <i>object</i> | SerialNET.VPort | An object that evaluates to an SerialNET.VPort object. |
| <i>Value</i> | string | string to trigger reception of data on. |

VPort.EndTrigger

SerialTools **v2.00** and later

Sets end trigger. Default value is null.

See VPort . [OnDataFromPort](#) for how this functionality is used.

Syntax

object.EndTrigger = Value
Value = object.EndTrigger

| Part | Type | Description |
|---------------|-----------------|--|
| <i>object</i> | SerialNET.VPort | An object that evaluates to an SerialNET.VPort object. |
| <i>Value</i> | string | End trigger. |

VPort.BufferSize

SerialTools **v2.00** and later

Sets the input buffer size. Default value is 0.

BufferSize returns the number of bytes in the serial buffer that are still not parsed.

NOTE! If you set BufferSize to a value different from 0 then OnDataFromPort will not be called until BufferSize bytes of data has been received. By default BufferSize is set to 0, which means this feature is disabled.

See VPort . [OnDataFromPort](#) for how this functionality is used.

Syntax

object.BufferSize = Value
Value = object.BufferSize

| Part | Type | Description |
|---------------|-----------------|--|
| <i>object</i> | SerialNET.VPort | An object that evaluates to an SerialNET.VPort object. |
| <i>Value</i> | int | Input buffer size. |

VPort.Timeout

SerialTools v2.00 and later

Time in **milliseconds** before a timeout event is generated. A value of 0 means that the timeout functionality is disabled. Default value is 0.

See VPort . [OnDataFromPort](#) for more info.

Syntax

object.Timeout = *Value*

Value = *object*.Timeout

| Part | Type | Description |
|---------------|-----------------|--|
| <i>object</i> | SerialNET.VPort | An object that evaluates to an SerialNET.VPort object. |
| <i>Value</i> | int | Timeout in milliseconds. 0 to disable. |

VPort.OnDataFromPort

SerialTools v2.00 and later

OnDataFromPort is called when data has been received from the (virtual) serial port.

Timeout. If VPort . [Timeout](#) is set to a value different from 0 (default) OnDataFromPort is called with null (C#) or Nothing (eVB / VB6 / VB.NET) as argument if no data has been received during the time intervall specified by Timeout. Timeout is specified in milliseconds.

Buffer size. Use VPort . [BufferSize](#) to specify how many bytes should be received before OnDataFromPort is called (default 0 - disabled).

Simple parsing. If VPort . [EndTrigger](#) is set to a value different from null (default).

OnDataFromPort will be called when the specified string is found in the input from the serial port. This is very usefull if you got data with a known format coming in to the serial port. If EndTrigger is used BufferSize is ignored.

If VPort . [StartTrigger](#) is set to a value different from null (default). All data is ignored until the specified string is found in the serial port input. If both StartTrigger and EndTrigger is used only data between (including the triggers themself) the triggers will be passed on to OnDataFromPort. If EndTrigger is null, OnDataFromPort will be called when BufferSize bytes has been received. If EndTrigger is null StartTrigger must be "retriggered" each time OnDataFromPort is called or else OnDataFromPort will be called with data each time the input buffer is full.

OnDataFromPort is called with a string as argument. To convert this string to binary data see Port . [StringToByteArray](#) and to convert from binary data to a string see Port . [ByteArrayToString](#)

See also VPort . [Created](#) and VPort . [DataFromPort](#).

Syntax

obj_OnDataFromPort (*Value*)

| Part | Type | Description |
|--------------|--------|---------------------------------|
| <i>obj</i> | | Event source. |
| <i>Value</i> | string | Data read from the serial port. |

VPort.DataToPort

SerialTools v2.00 and later

Writes data to the serial port. Write returns instantly. DataToPort takes a string as an argument, to convert binary data to a string see Port . [ByteArrayToString](#).

Note! The virtual port must be opened by an application using the port or else this method will generated an exception. The events VPort . [OnOpened](#), VPort . [OnClosed](#) or the method VPort . [PortStatus](#) can be used to monitor if the port is opened or closed.

Case 1 (generating exception):

Your application creates a virtual port using VPort.Created.

Your application calls VPort.DataToPort()

An exception is thrown!!

Case 2 (no errors):

Your application creates a virtual port using VPort.Created.

3rd party application opens the port.

Your application calls VPort.DataToPort()

Data is transfered to 3rd party application.

See also VPort . [Created](#).

Syntax

BytesWritten = object.Write (*Value*)

| Part | Type | Description |
|---------------------|-----------------|--|
| <i>object</i> | SerialNET.VPort | An object that evaluates to an SerialNET.VPort object. |
| <i>Value</i> | string | Data to be written to the serial port. |
| <i>BytesWritten</i> | int | Bytes written. |

VPort.DataFromPort

SerialTools v2.00 and later

Reads data from the serialport. Cannot be used if an event handler is assigned to VPort . [OnDataFromPort](#) (.NET) or VPort . [NoEvents](#) is set to true (ActiveX).

Read has two versions. The first takes a `BufferSize` and `Timeout` as arguments. Setting `BufferSize` to 0 will return all data in serial buffer. But if the serial buffer is empty it will wait until something has arrived. `Timeout` determines how long to wait (in milliseconds). Set `Timeout` to 0 to disable time out.

The second version takes `StartTrigger`, `EndTrigger` and `Timeout` as arguments. Read will parse the serial buffer and return data according to the rules specified for `VPort.OnDataFromPort`. If no matching data is found in the serial buffer `Timeout` determines how long to wait before returning null / Nothing.

Note! `DataFromPort` will through an exception if the port is closed before the call, or is closed during the call to `DataFromPort`.

On time out `DataFromPort()` always returns null / Nothing.

Syntax 1

Value = `object.DataFromPort (BufferSize, Timeout)`

| Part | Type | Description |
|-------------------|-----------------|---|
| <i>object</i> | SerialNET.VPort | An object that evaluates to an SerialNET.VPort object. |
| <i>Value</i> | string | Data received from serial port. Nothing/null if no data received. |
| <i>BufferSize</i> | int | See <code>OnDataFromPort</code> for details. |
| <i>Timeout</i> | int | See <code>OnDataFromPort</code> for details. |

Syntax 2

Value = `object.DataFromPort (StartTrigger, EndTrigger, Timeout)`

| Part | Type | Description |
|---------------------|-----------------|---|
| <i>object</i> | SerialNET.VPort | An object that evaluates to an SerialNET.VPort object. |
| <i>Value</i> | string | Data received from serial port. Nothing/null if no data received. |
| <i>StartTrigger</i> | string | See <code>OnDataFromPort</code> for details. |
| <i>EndTrigger</i> | string | See <code>OnDataFromPort</code> for details. |
| <i>Timeout</i> | int | See <code>OnDataFromPort</code> for details. |

VPort.MultiThreading

SerialTools v2.00 and later

ActiveX only. For .NET see `VPort.Parent`.

If you are making a form based application you should not care about this property.

If set to false (default) all events will be marshaled to the main thread. This requires that you add the Port control to the form. See the samples for how to do that.

If set to true events will be called with a new thread (or any thread from the thread pool). If you are making an application without a form, you need to set this property to true and be aware of that you event handlers will be called in a new thread.

Syntax

object.MultiThreading = *Value*

Value = *object*.MultiThreading

| Part | Type | Description |
|---------------|-----------------|--|
| <i>object</i> | SerialNET.VPort | An object that evaluates to an SerialNET.VPort object. |
| <i>Value</i> | bool | Multi threading on or off. |

VPort.Parent

SerialTools **v2.00** and later

This property is only available in SerialTools .NET (not ActiveX)

If set to null/Nothing (default) then events will be called in a separate thread (multi-threading).

If you are making a form based application you should set this property to your form (this/Me) pointer. If you are not using events (no OnDataFromPort, OnWritten, etc..) you do not need to care about this property.

Syntax

object.Parent = *Value*

Value = *object*.Parent

| Part | Type | Description |
|---------------|------------------------------|--|
| <i>object</i> | SerialNET.VPort | An object that evaluates to an SerialNET.VPort object. |
| <i>Value</i> | System.Windows.Forms.Control | Control to marshal events to. |

VPort.Dispose

SerialTools **v2.00** and later

.NET only!

Must be called when you have finished using the object. If not called you application will not unload.

Syntax

object.Dispose ()

| Part | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|-----------------|--|
| <i>object</i> | SerialNET.VPort | An object that evaluates to an SerialNET.VPort object. |
|---------------|-----------------|--|

VPort.OnOpened

SerialTools **v2.00** and later

Event that is called when an application opens the virtual serial port.

Syntax

`obj_OnOpened ()`

| Part | Type | Description |
|------------|------|---------------|
| <i>obj</i> | | Event source. |

VPort.OnClosed

SerialTools **v2.00** and later

Event that is called when an application closes the virtual serial port.

Syntax

`obj_OnClosed ()`

| Part | Type | Description |
|------------|------|---------------|
| <i>obj</i> | | Event source. |

VPort.PortStatus

SerialTools **v2.00** and later

Gets status from port. Cannot be used if an event handler is assigned to VPort . [OnOpened](#), or VPort . [OnClosed](#) (.NET) or VPort . [NoEvents](#) is set to true (ActiveX).

PortStatus can be used to determine when a virtual port is opened or closed by an application (using the virtual port). PortStatus will lock until the requested state has happened or the function has timed out.

Syntax

`result = object.PortStatus (Status, Timeout)`

| Part | Type | Description |
|----------------|-----------------|---|
| <i>object</i> | SerialNET.VPort | An object that evaluates to an SerialNET.VPort object. |
| <i>Status</i> | bool | Status to wait for. True = port opened. False = port closed. |
| <i>Timeout</i> | int | Timeout in milliseconds. 0 = wait for ever. |
| <i>result</i> | bool | True if the requested state happend. False on timeout or if the virtual port was destroyed. |

VPort.NoEvents

SerialTools **v2.00** and later

ActiveX only. Set this to true (default false) if you wont use events.

If NoEvents is set to true you can use VPort . [DataFromPort](#) and VPort . [PortStatus](#).

If NoEvents is set to false (default) you can use the events VPort . [OnDataFromPort](#), VPort . [OnOpened](#) and VPort . [OnClosed](#).

Syntax

object.NoEvents = Value

Value = object.NoEvents

| Part | Type | Description |
|---------------|-----------------|--|
| <i>object</i> | SerialNET.VPort | An object that evaluates to an SerialNET.VPort object. |
| <i>Value</i> | bool | Events on or off. |

License

SerialTools **v1.00** and later

Activates the component. Your application must contain the following code before starting to use the objects in SerialTools:

```
Dim objLicense As New License  
objLicense.LicenseKey = key
```

License.LicenseKey

SerialTools **v1.00** and later

A license key is necessary to use the component. During development the key found [here](#) can be used. To distribute the component as part of your application you need to [purchase](#) a license.

This code should be included somewhere in the start of your application before accessing any other objects in the SerialTools component.

```
Dim objLicense As New License  
objLicense.LicenseKey = key
```

Syntax

object.LicenseKey = Key

| Part | Type | Description |
|---------------|-------------------|--|
| <i>object</i> | SerialNET.License | An object that evaluates to an SerialNET.License object. |

| | | |
|------------|--------|-------------|
| <i>Key</i> | string | License key |
|------------|--------|-------------|

Handshake SerialTools **v1.00** and later

Enumeration used by Port . [Handshake](#) to set the type of low level handshake.

Handshake.None SerialTools **v1.00** and later

Value to disable handshake. See Port . [Handshake](#) for more info.

Numeric value: 0

Handshake.RTS SerialTools **v1.00** and later

Value for RTS/CTS handshake. See Port . [Handshake](#) for more info.

Numeric value: 1

Handshake.DTR SerialTools **v1.00** and later

Value for DSR/DTR handshake. See Port . [Handshake](#) for more info.

Numeric value: 2

Handshake.XonXoff SerialTools **v1.20** and later

Value for Xon/Xoff handshake. See Port . [Handshake](#) for more info.

Numeric value: 4

StopBits SerialTools **v1.00** and later

Enumeration used by Port . [StopBits](#).

StopBits.One SerialTools **v1.00** and later

Value to that represents one stop bit. See Port . [StopBits](#) for more info.

Numeric value: 0

StopBits.OneAndOneHalf

SerialTools **v1.00** and later

Value to that represents one and a half stop bit. See Port . [StopBits](#) for more info.

Numeric value: 1

StopBits.Two

SerialTools **v1.00** and later

Value to that represents two stop bits. See Port . [StopBits](#) for more info.

Numeric value: 2

Parity

SerialTools **v1.00** and later

Enumeration used by Port . [Parity](#).

Parity.No

SerialTools **v1.00** and later

Value to that represents no parity. See Port . [Parity](#) for more info.

Numeric value: 0

Parity.Odd

SerialTools **v1.00** and later

Value to that represents odd parity. See Port . [Parity](#) for more info.

Numeric value: 1

Parity.Even

SerialTools **v1.00** and later

Value to that represents even parity. See Port . [Parity](#) for more info.

Numeric value: 2

Parity.Mark

SerialTools **v1.00** and later

Value to that always sets parity. See Port . [Parity](#) for more info.

Numeric value: 3

Parity.Space

SerialTools v1.00 and later

Value to that always clears parity. See Port . [Parity](#) for more info.

Numeric value: 4



100% managed code

© 2002-2005 Franson Technology AB, All rights reserved (franson.com)