

Cet article a fait l'objet d'une traduction automatique. Pour afficher l'article en anglais, activez la case d'option Anglais. Vous pouvez également afficher le texte anglais dans une fenêtre contextuelle en faisant glisser le pointeur de la souris sur le texte traduit.

SerialPort, classe

.NET Framework (current version)

Représente une ressource de port série.

Pour parcourir le code source .NET Framework pour ce type, consultez la [Reference Source](#).

Espace de noms: [System.IO.Ports](#)
Assembly: System (dans System.dll)

Hierarchie d'héritage





Syntaxe

C#

















```
public class SerialPort : Component
```















Constructeurs

	Nom	Description
	SerialPort()	Initialise une nouvelle instance de la classe SerialPort.
	SerialPort(IContainer)	Initialise une nouvelle instance de la classe SerialPort à l'aide de l'objet IContainer spécifié.
	SerialPort(String)	Initialise une nouvelle instance de la classe SerialPort avec le nom de port spécifié.
	SerialPort(String, Int32)	Initialise une nouvelle instance de la classe SerialPort avec le nom de port et la vitesse (en bauds) spécifiés.
	SerialPort(String,	



	Int32, Parity)	Initialise une nouvelle instance de la classe SerialPort avec le nom de port, la vitesse (en bauds) et le bit de parité spécifiés.
	SerialPort(String, Int32, Parity, Int32)	Initialise une nouvelle instance de la classe SerialPort avec le nom de port, la vitesse (en bauds), le bit de parité et les bits de données spécifiés.
	SerialPort(String, Int32, Parity, Int32, StopBits)	Initialise une nouvelle instance de la classe SerialPort avec le nom de port, la vitesse (en bauds), le bit de parité, les bits de données et le bit d'arrêt spécifiés.





















Propriétés








	Nom	Description
	BaseStream	Obtient l'objet Stream sous-jacent pour un objet SerialPort.
	BaudRate	Obtient ou définit la vitesse en bauds série.
	BreakState	Obtient ou définit l'état du signal d'arrêt.
	BytesToRead	Obtient le nombre d'octets de données dans la mémoire tampon de réception.
	BytesToWrite	Obtient le nombre d'octets de données dans la mémoire tampon d'envoi.
	CanRaiseEvents	Obtient une valeur indiquant si le composant peut déclencher un événement.(Hérité de Component .)
	CDHolding	Obtient l'état de la ligne de détection de porteuse pour le port.
	Container	Obtient les IContainer qui contient la Component .(Hérité de Component .)
	CtsHolding	Obtient l'état de la ligne CTS (Clear-To-Send).
	DataBits	Obtient ou définit la longueur standard des bits de données par octet.
	DesignMode	Obtient une valeur qui indique si la Component est actuellement en mode design. (Hérité de Component .)
	DiscardNull	Obtient ou définit une valeur indiquant si les octets null sont ignorés lorsqu'ils sont transmis entre le port et la mémoire tampon de réception.
	DsrHolding	Obtient l'état du signal DSR (Data Set Ready).
	DtrEnable	Obtient ou définit une valeur qui active le signal DTR lors d'une communication série.
	Encoding	Obtient ou définit l'octet d'encodage pour la conversion de texte avant et après la transmission.
	Events	

		Obtient la liste des gestionnaires d'événements associés à cette Component .(Hérité de Component .)
	Handshake	Obtient ou définit le protocole de négociation pour la transmission de données par le port série en utilisant une valeur de Handshake .
	IsOpen	Obtient une valeur indiquant l'état ouvert ou fermé de l'objet SerialPort.
	NewLine	Obtient ou définit la valeur utilisée pour interpréter la fin d'un appel aux méthodes ReadLine et WriteLine .
	Parity	Obtient ou définit le protocole de contrôle de parité.
	ParityReplace	Obtient ou définit l'octet qui remplace les octets non valides dans un flux de données quand une erreur de parité se produit.
	PortName	Obtient ou définit le port pour les communications, y compris mais non limité à tous les ports COM disponibles.
	ReadBufferSize	Obtient ou définit la taille de la mémoire tampon SerialPort.
	ReadTimeout	Obtient ou définit le nombre de millisecondes avant un dépassement du délai d'attente quand une opération de lecture ne se termine pas.
	ReceivedBytesThreshold	Obtient ou définit le nombre d'octets dans la mémoire tampon d'entrée interne avant qu'un événement DataReceived ne se produise.
	RtsEnable	Obtient ou définit une valeur indiquant si le signal RTS (Request to Send) est activé lors d'une communication série.
	Site	Obtient ou définit le ISite de la Component .(Hérité de Component .)
	StopBits	Obtient ou définit le nombre standard de bits d'arrêt par octet.
	WriteBufferSize	Obtient ou définit la taille de la mémoire tampon de sortie du port série.
	WriteTimeout	Obtient ou définit le nombre de millisecondes avant qu'un dépassement du délai d'attente se produise quand une opération d'écriture ne se termine pas.


Méthodes

	Nom	Description
	Close()	Ferme la connexion au port, affecte à la propriété IsOpen la valeur false et supprime l'objet Stream interne.
	CreateObjRef(Type)	Crée un objet qui contient toutes les informations requises pour générer un proxy permettant de communiquer avec un objet distant.(Hérité de MarshalByRefObject .)





	DiscardInBuffer()	Ignore les données de la mémoire tampon de réception du pilote série.
	DiscardOutBuffer()	Ignore les données de la mémoire tampon de transmission du pilote série.
	Dispose()	Libère toutes les ressources utilisées par Component .(Hérité de Component .)
	Dispose(Boolean)	Libère les ressources non managées utilisées par SerialPort et libère éventuellement les ressources managées.(Remplace Component.Dispose(Boolean) .)
	Equals(Object)	Détermine si l'objet spécifié est identique à l'objet actuel.(Hérité de Object .)
	Finalize()	Libère les ressources non managées et exécute d'autres opérations de nettoyage avant la récupération du Component par le garbage collection.(Hérité de Component .)
	GetHashCode()	Fait office de fonction de hachage par défaut.(Hérité de Object .)
	GetLifetimeService()	Récupère l'objet de service de durée de vie en cours qui contrôle la stratégie de durée de vie de cette instance.(Hérité de MarshalByRefObject .)
	GetPortNames()	Obtient un tableau de noms de ports série pour l'ordinateur actuel.
	GetService(Type)	Retourne un objet qui représente un service fourni par Component ou par son Container .(Hérité de Component .)
	GetType()	Obtient le Type de l'instance actuelle.(Hérité de Object .)
	InitializeLifetimeService()	Obtient un objet de service de durée de vie pour contrôler la stratégie de durée de vie de cette instance.(Hérité de MarshalByRefObject .)
	MemberwiseClone()	Crée une copie superficielle du Object actuel.(Hérité de Object .)
	MemberwiseClone(Boolean)	Crée une copie superficielle de l'utilisateur actuel MarshalByRefObject objet.(Hérité de MarshalByRefObject .)
	Open()	Ouvre une nouvelle connexion de port série.
	Read(Byte[], Int32, Int32)	Lit un certain nombre d'octets de la mémoire tampon d'entrée SerialPort et écrit ces octets dans un tableau d'octets au décalage spécifié.
	Read(Char[], Int32, Int32)	Lit un certain nombre de caractères de la mémoire tampon d'entrée SerialPort et écrit ces caractères dans un tableau de caractères à un décalage donné.
	ReadByte()	Lit de façon synchrone un octet de la mémoire tampon d'entrée SerialPort.
	ReadChar()	Lit de façon synchrone un caractère de la mémoire tampon d'entrée SerialPort.
	ReadExisting()	

)	Lit tous les octets immédiatement disponibles, en fonction de l'encodage, dans le flux et dans la mémoire tampon d'entrée de l'objet SerialPort.
	ReadLine()	Lit jusqu'à la valeur NewLine dans la mémoire tampon d'entrée.
	ReadTo(String)	Lit une chaîne jusqu'à la valeur <i>value</i> spécifiée dans la mémoire tampon d'entrée.
	ToString()	Retourne un String contenant le nom de la Component , le cas échéant. Cette méthode ne doit pas être remplacée.(Hérité de Component .)
	Write(Byte[], Int32, Int32)	Écrit un nombre spécifié d'octets sur le port série en utilisant les données d'une mémoire tampon.
	Write(Char[], Int32, Int32)	Écrit un nombre spécifié de caractères sur le port série en utilisant les données d'une mémoire tampon.
	Write(String)	Écrit la chaîne spécifiée sur le port série.
	WriteLine(String)	Écrit la chaîne spécifiée et la valeur NewLine dans la mémoire tampon de sortie.

Champs

	Nom	Description
	InfiniteTimeout	Indique qu'aucun dépassement du délai d'attente ne doit se produire.

Événements

	Nom	Description
	DataReceived	Indique que des données ont été reçues via un port représenté par l'objet SerialPort.
	Disposed	Se produit lorsque le composant est supprimé par un appel à la Dispose (méthode). (Hérité de Component .)
	ErrorReceived	Indique qu'une erreur s'est produite par rapport à un port représenté par l'objet SerialPort.
	PinChanged	Indique qu'un événement de signal non lié aux données s'est produit sur le port représenté par l'objet SerialPort.

Notes

Remarque

Pour afficher le code source .NET Framework pour ce type, consultez la [Reference Source](#). Vous pouvez parcourir le code source en ligne, télécharger la référence hors connexion et parcourir les sources (y compris les correctifs et mises à jour) pendant le débogage ; see [instructions](#).

Utilisez cette classe pour contrôler une ressource de fichier de port série. Cette classe fournit des e/s synchrones et pilotées par événements, l'accès aux États de broche et d'arrêt et l'accès aux propriétés du pilote série. En outre, les fonctionnalités de cette classe peuvent être encapsulées dans une liste interne [Stream](#) objet, accessible via la [BaseStream](#) propriété et passé aux classes qui encapsulent ou utilisent des flux de données.

La SerialPort classe prend en charge les encodages suivants : [ASCIIEncoding](#), [UTF8Encoding](#), [UnicodeEncoding](#), [UTF32Encoding](#), et tout encodage défini dans mscorlib.dll où la page de codes est inférieure à 50000 ou la page de codes est 54936. Vous pouvez utiliser d'autres encodages, mais vous devez utiliser le [ReadByte](#) ou [Write](#) (méthode) et exécuter l'encodage vous-même.

Vous utilisez la [GetPortNames](#) méthode pour récupérer les ports valides pour l'ordinateur actuel.

Si un SerialPort devient bloqué pendant une opération de lecture, ne pas abandonner le thread. Au lieu de cela, fermez la base de diffuser ou d'éliminer les SerialPort objet.

Exemples

L'exemple de code suivant illustre l'utilisation de la SerialPort classe pour permettre à deux utilisateurs de chat à partir de deux ordinateurs distincts reliés par un câble null modem. Dans cet exemple, les utilisateurs sont invités pour les paramètres de port et un nom d'utilisateur avant de conversation. Les deux ordinateurs doivent exécuter le programme pour obtenir toutes les fonctionnalités de cet exemple.

C#

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

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

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

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

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

```

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

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

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

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

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

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

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

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

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

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

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

```

```

    {
        Console.WriteLine("    {0}", s);
    }

    Console.Write("Enter COM port value (Default: {0}): ", 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);
}
}

```

Sécurité

SecurityPermission

for the ability to call unmanaged code. Associated enumeration:

F:System.Security.Permissions.SecurityPermissionFlag.UnmanagedCode

Informations de version

.NET Framework

Disponible depuis 2.0

Sécurité des threads

Tous les membres statiques (**Shared** en Visual Basic) publics de ce type sont thread-safe. Il n'est pas garanti que les membres d'instance soient thread-safe.

Voir aussi

[System.IO.Ports](#), espace de noms

[Retour au début](#)

© 2016 Microsoft