VBMan
Components for RS-232C
Programming Manual

version 8.00

TechKnowledge

VBMan Components for RS-232C

はじめに

製品の概要

VBMan components for RS-232C version 8.00をご利用いただき誠にありがとうございます。当製品はMicrosoft Visual Studio 2019～2022/Microsoft .NET framework 4.72以降または.NET 7.0でご利用いただけるRS-232Cシリアル通信をサポートする.NET カスタム・コンポーネントです。以下は当コンポーネントの概要・特徴です。

* 信頼性と実績
  当コンポーネントは1994年の16bit/VBXコンポーネントの時代から32bitOCX,ActiveXコントロール、.NET framework 2.0 コンポーネントの時代を経て、.NET framework 4, .NET 5～7をサポートすることになりました。RS-232C用ソフトウェア・コンポーネントとして多くのお客様のアプリケーションに組み込まれており、多数の実績がございます。
* .NETコンポーネント
  当コンポーネントは.NET frameworkのComponentModelクラス仕様に沿って作成されていますのでVisual Studio 各バージョンのフォームエディタによりプロパティ設定等がビジュアルに実行可能で初期化コードも自動生成されます。また、当コンポーネントを利用できる,NET言語としてはVisual Basic.NET/C#/J#/Visual C++(Managed Application)があります。アプリケーションとしてはWindows Form/Web Form/コンソールアプリケーションなどをサポートしています。

- 通信イベントのサポート
  当コントロールは通信イベント監視スレッドにより、回線状況、ファイル転送の完了、エラー・ステータス、送受信文字などを通信イベントとしてユーザープログラムに通知します。
- デバック機能
  DebugTraceプロパティを設定すると送受信データをトレースプログラムに16/10進表示することができます。アプリケーションのデバッグ効率を改善します。
- ファイル転送
  Xmodem Check Sum, Xmodem CRC, XModem 1Kによるファイル転送が可能です。
- スレッド優先順位の設定
  さまざまな動作環境に対応するために通信管理スレッドの優先順位をプロパティで設定することが可能です。
- 送受信タイムアウトの設定
  通常は感知することのできない接続された通信機器の電源断などでも送受信タイムアウトをプロパティで指定することにより、アプリケーションのハングアップ等を回避できます。

開発ライセンス

開発ライセンスとは、本製品１ライセンスを１台のパーソナル・コンピュータ・システムで１開発者が利用することが出来る権利です。当製品のディスケットを複製して、複数人でのご使用等は著作権法違反となり罰せられます。ご注意ください。

- 当ソフトウェア製品は、著作権法及び国際著作権条約をはじめ、その他の無体財産権に関する法律ならびに条約によって保護されています。
- 当ソフトウェアに対するリバースエンジニアリング及び、改変は一切禁止します。
- 本製品をラップする形で作成した同様の機能を持ったカスタム・コントロール製品の販売は禁止いたします。
- 当製品の使用権はいかなる方法によっても第三者に譲渡および貸与することは出来ません。
- 使用権は当製品を開梱したときに発効します。商品パッケージ開梱後の返品はできませんので予めご了承ください。
- 使用権は以下のいずれかの事由が起こった場合に消滅します。
  購入者が製品に同封されているユーザー登録書を返送しない場合。
  購入者が使用規定に違反した場合。
  プログラム・ディスク、印刷物などを使用権の範囲外の目的で複製した場合。

ランタイムライセンス

ランタイムライセンスとは弊社製品のコンポーネントをお客様のアプリケーションと一緒に配布して動作させる使用権です。当製品は１開発ライセンス＋１ランタイムライセンスのパッケージとさせていただいております。追加ランタイムライセンスにつきましてはシステムラボまでお問合わせください。

保証規定

当製品、および付随する著作物に対して商品性及び特定の目的への適合性などについての保証を含むいかなる保証もそれを明記する、しないに関わらず提供されることはありません。

当製品の著作者及び、製造、配布に関わるいかなる者も、当ソフトウェアの不具合によって発生する損害に対する責任は、それが直接的であるか間接的であるか、必然的であるか偶発的であるかに関わらず、負わないものとします。それは、その損害の可能性について、開発会社に事前に知らされていた場合でも同様です。

ユーザーサポート

- ユーザー登録
  ユーザー登録はウェブまたは製品に添付のユーザー登録用紙で可能です。登録用紙の場合は必要事項をご記入の上、販売会社システムラボまでーるご送付ください。ユーザー登録が行われていないと、ユー

ザーサポートが受けられない場合があります。必ずご登録をお願いいたします。

- お問い合わせの方法
  どうしても解決できない問題が発生した場合には、技術サポートをご利用ください。あらかじめマニュアルの最終ページの調査依頼書に相当の情報をご用意いただきメールでお送りいただければ、折り返しご連絡をさせていただきます。**当製品につきましては、製品の性格上複雑なやりとりになる場合が多いため、電話によるユーザーサポートはありません。**また、問い合わせの内容によっては、再現調査などのために回答に時間がかかる場合がありますのでご了承をお願いいたします。

- 登録内容の変更について
  転居などによるご住所や電話番号など、登録内容に変更が生じた場合には、郵送またはファックスにて、販売会社システムラボまでご連絡をいただきますようお願いいたします。なお、電話による口頭での連絡変更は受けかねますので、よろしくお願いいたします。

- 併用される他社製品について
  当社製品と併用される、他社製品の使用方等についてのご質問をお受けすることがあります。しかし、他社製品に関しましてはお答えできない場合があります。他社製品につきましては、該当開発・販売会社にご連絡ください。

- 無償サポート期間について
  無償サポート期間はユーザー登録後、2インシデントを上限とし最初のお問い合わせから90日間となっております。有償サポートにつきましては販社システムラボにて承っておりますのでご連絡ください。

- サポートのパフォーマンスについて
  簡単なお問い合わせであれば１労働日以内を目標にサポートをいたします。お問い合わせの内容により調査などのために回答に時間がかかる場合がありますのであらかじめご了承ください。また弊社が別途定めた定休日にはサポート休止する場合があります。サポートに優先順位はありません。当着順に処理いたしますのでよろしくお願いいたします。

販売元

株式会社システムラボ

東京都北区田端6丁目1番1号　田端アスカタワー12Ｆ

電話:	03-5809-0893
ファックス:	03-4578-9261
サーポートメール:	support@systemlab.co.jp
Web:	www.systemlab.co.jp

開発元・サポート

株式会社テクナレッジ

東京都世田谷区駒沢２丁目１６番１号　サンドービル９Ｆ

電話:	03-3421-7621
ファックス:	03-3421-6691
サポートメール:	support@techknowledge.co.jp
Web:	www.techknowledge.co.jp

商標登録

インストール

VBMan components for RS-232Cのインストールについて説明します。

システム条件

VBMan components for RS-232C 8.0が対応するフレームワークは以下です。

- Microsoft .NET framework 4.72 ～
- Microsoft .NET 7～

VBMan components for RS-232CはMicrosoft .NET framework 4.72以降をサポートする以下の64bit オペレーティング・システムが動作するパーソナル･コンピュータ環境で動作します。

- Windows 10～11
- Windows Server 2008～2022

VBMan components for RS-232Cは以下のMicrosoft .NET framework 対応言語で動作いたします。

- Microsoft Visual Basic.NET
- Microsoft Visual C#
- Microsoft Visual J#
- Microsoft Managed C++/CLI

IDE(統合開発環境)のサポートは以下です。

- Microsoft Visual Studio 2019～2022

セットアップの起動

VBMan components for RS-232Cのセットアップ・プログラムを起動します。setup.exeは64bit

システムのプログラムフォルダーにVBMan components for RS-232Cが作成され、ヘルプ、サンプル等がメニューに登録されます。64BIT　OSには当製品の32BIT/64BIT版どちらもインストールすることができます。

インストール・ファイル一覧

以下に当製品のインストールされるファイルの一覧を示します。インストールディレクトリは<instdir>と表記します。32BIT OSのシステムディレクトリを<sysdir_X86>, 64BIT OSのシステムディレクトリを<sysdir_X64>と表記します。

モジュール再配布の欄にご注意ください。再配布不可と記載されるモジュールはランタイムライセンスを取得された場合に配布可能となるファイルです。再配布不可と記載されるファイルにつきましては開発機のみでのご利用可能となります。ランタイムライセンスを取得せずに開発機以外に配布した場合はソフトウェア著作権侵害となる場合がございますのでご注意ください。

モジュール名	概　　要	再配布
<instdir>\bin\CommLib.dll	.NET framework シリアル通信コンポーネント	可
<instdir>\bin\CommLibCore.dll	.NET core シリアル通信コンポーネント	可
<instdir>\bin\vbmtrace.exe	デバッグトレース表示ユーティリティ	不可
<instdir>\bin64\CommLib.dll	64bit .NET framework シリアル通信コンポーネント	可
<instdir>\bin64\CommLibCore.dll	64bit .NET core シリアル通信コンポーネント	可
<instdir>\bin64\vbmtrace.exe	デバッグトレース表示ユーティリティ 64BIT 版	不可
<sysdir_X86>\CommShm.dll	32BIT 共有メモリーモジュール	可
<sysdir_X64>\CommShm64.dll	64BIT 共有メモリーモジュール	可
<instdir>\man\CommLib800.html	リリースノート	不可
<instdir>\man\CommLib800.pdf	PDFマニュアル	不可
<instdir>\samples\vb.net\ *.*	VB.NET .net frameworkサンプルプログラム	不可
<instdir>\samples\cs\ *.*	C# .net frameworkサンプルプログラム	不可
<instdir>\core_samples\vb.net*.*	VB.NET .NET 7.0 サンプルプログラム	不可
<instdir>\core_samples\vb.net*.*	C# .NET 7.0 サンプルプログラム	不可

プロジェクト開始

IDEへの登録方法

.NET framework アプリにVisual StudioのIDEのツールボックスに当コンポーネントを登録する方法について説明します。登録方法はどのCommunity版も含めてどのIDEでもほぼ共通です。

Visual Basic.NET

Microsoft Visual Basic.NETからVBMan components for RS-232Cを使う方法を説明します。

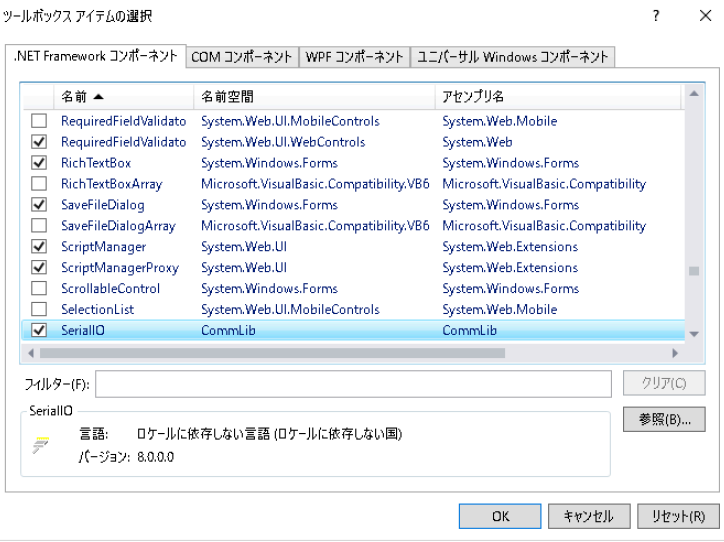
① Visual Studioを起動します。
② 新規プロジェクトをファイルメニューから選択します。
③ 言語をVisual Basic.NETを選択します。
④ プロジェクトタイプは「Windowsアプリケーション」を選択します。
⑤ ツールボックスからVBMan を追加したいタブを選択するか、新規タブを作成し、該当タブを開きます。
⑥ マウスの右クリックで表示されるメニューで「アイテムの選択」を選択します。
⑦ 「.NET frameworkコンポーネント」タブを選択します。
⑧ 「参照」ボタンを押しデフォルトインストールの場合は以下のフォルダにあるCommLib.DLLを選択します。

c:\Program Files\techknowledge\VBMan Components for RS-232C 8.00\bin64

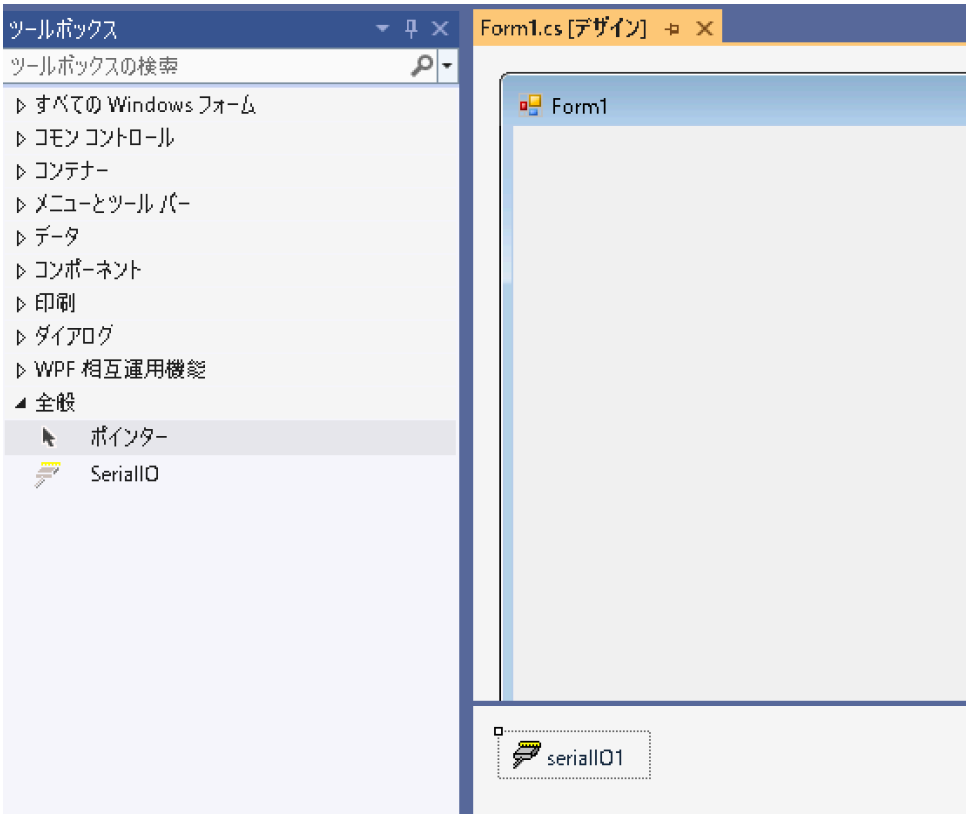
32BIT版をご利用の場合は以下です。
c:\Program Files\techknowledge\VBMan Components for RS-232C 8.00\bin

⑨ 追加された「SerialIO」コントロールをWindowsフォームにドラッグして利用します。

以下は手順⑦の実行画面例です。



以下は全般タブにVBMan components for RS-232Cを追加しWindowsFormでの編集画面例です。



Visual C#

Visual C#の場合は手順③で選択する言語をVisual C#とする以外はVisual Basic.NETの場合と同一です。

Visual C++/CLI

Visual C++の場合、アプリケーションの種類として.NET frameworkまたは.NETを手順③で選択してください。それ以外の手順はVisual Basic.NETの場合と同一です。(Visual C++バージョンによってはアプリケーションの種類に表示されないことがあります)

Namespace/Class名について

当コンポーネントのNamespaceはCommLibとなります。また、Class名はSerialIOとなります。以下はVisual C#で当コンポーネントのオブジェクトを生成するコード例です。

```
CommLib.SerialIO rs232c = new CommLib.SerialIO;
```

IDEのフォームデザイナを使う場合には上記のようなオブジェクト生成コードはIDEにより自動的に生成されます。

.NETアプリについて

Visual Studio 2022でアプリ作成時に .NET 7.0を選択してください。プロジェクト参照から参照ボタンをクリックして以下を選択します。

c:¥Program Files¥techknowledge¥VBMan Components for RS-232C 8.00¥bin64¥CommLibCore.dll

ビルドディレクトリにijwhost.dllをコピーします。アプリのルートディレクトリにコマンドプロンプトを開きDebugビルドを対象にした場合の例です。

```
cd bin¥Debug¥net7.0-windows
copy c:¥Program Files¥techknowledge¥VBMan Components for RS-232C 8.00¥bin64¥ijwhost.dll
```

コンパチビリティ

.NET framework 2.0～7.0 版とのコンパチビリティ

当製品バージョン 8.0は新しいバージョンであるMicrosoft .NET framework 4.72以降とVisual Studio 2019以降に対応したもので、基本的な動作には変更はありません。

旧バージョンのComLib.DLLの参照をComLib.DLLバージョン8.0.0への参照に変更することでお客様のアプリケーションを.NET framework 4.xアプリケーションとすることが出来ます。

COM コンポーネント版とのコンパチビリティ

ここではVBMan Control for RS-232C V4.xとの相違点を記述します。今回のバージョンではOLEではメソッドのオーバーロードや呼び出し側の言語でのスレッドサポートにより整理した機能に加えて16bit/VBX時代からのコンパチビリティのためのメソッドやActiveXコントロールを完全にサポートしていなかった言語をサポートするメソッド等は整理してとてもすっきりしたものになっています。プロパティ名やメソッド名は従来版との関連付けは容易なのでプログラムのアップグレードは比較的容易と考えておりまして、アップグレードの手間に比べて機能豊富な.NET Frameworkとの親和性による扱い安さによる利便性が得られることが大きなメリットになると考えております。

① NETコンポーネントに変更したため.NET言語から利用する場合にはラッパークラスが不要になりました。
② .NETコンポーネントではメソッドのオーバーロードが可能なりました。従来版ではデータタイプによって異なるメソッド名を与えていましたがオーバーロードすることで複数のメソッド名をSendやReceiveのようにシンプルにまとめることができました。
③ プロパティのプリフィックスVcを削除しました。
④ CLRでスレッドクラスが提供されたのでファイル転送内部ではスレッド起動しないように変更しました。SendFile/ReceiveFileメソッドはスレッドから利用してください。
⑤ 従来版ではユーティリティとして別DLLになっていたCRC計算のメソッドはSerialIOクラスに取り込まれました。

サンプルコード

概要

サンプルコードはVisual Studio 2022 でx64ビルド,.NET framework 4.8または.NET 7.0向けに設定されています。

管理者権限について

パッケージのデフォルトインストールフォルダーにあるサンプルコードをVisual Studioで開く場合にはビルド結果書込権限が必要なのでVisual Studioを管理者権限で実行する必要があります。サンプルコードをフォルダーごとユーザーフォルダー下へコピーして確認されることをお勧めします。

サンプルコードの参照設定について

サンプルコードでは当コンポーネントの参照を以下に設定してあります。

c:\Program Files\TechKnowledge\VBMan Components for RS-232C 8.00\bin64

32BITビルドする場合は参照先を以下に変更してください。

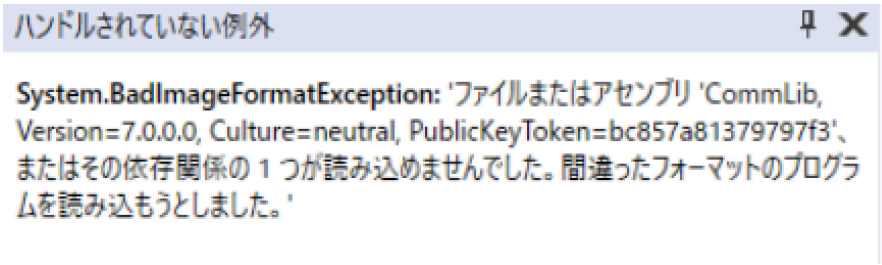
c:\Program Files\TechKnowledge\VBMan Components for RS-232C 8.00\bin

サンプルコードを別のフォルダーにコピーすると参照設定が外れることがあります。一旦CommLib.dll/CommLibCore.dllへの参照を削除して再設定が必要となります。

サンプルコードのCPUアーキテクチャについて

AnyCPUアーキテクチャでサンプルコードは作成されていますが、ご利用のCommLib.DLL CPUアーキテクチャに合わせて X86またはX64に変更してビルドしてください。

CPUアーキテクチャが合致しない場合は以下のような例外が発生します。

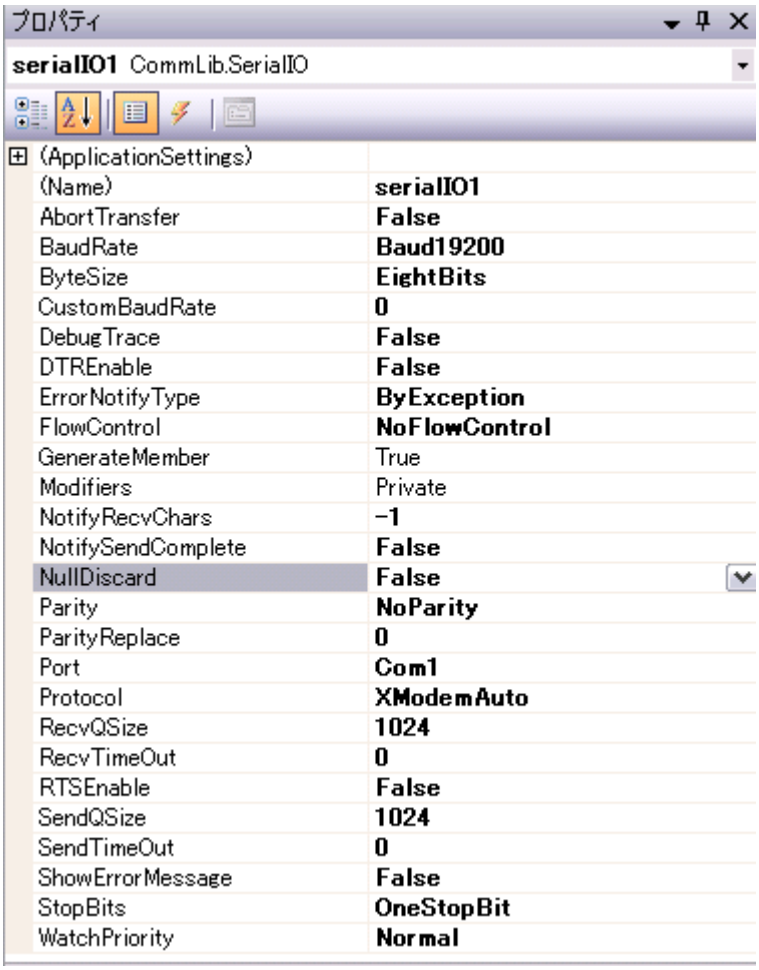


カスタム・プロパティ

VBMan components for RS-232Cでは以下のカスタム・プロパティを設定することで通信アプリケーション通信条件やカスタム・コントロールの動作などを設定します。カスタム・プロパティで通信条件を設定することでコード記述量を減らすことができます。

カスタム・プロパティ名	詳　　細
AbortTransfer	ファイル転送の中断
BaudRate	通信速度
ByteSize	通信データサイズ
CustomBaudRate	任意の通信速度設定
DebugTrace	デバッグ・トレースの指定
DTREnable	通信開始時にDTRラインをイネーブルにする。
FlowControl	フロー制御指定
InBufferCount	受信バッファにある文字バイト数。
LastMajorErrCode	最後のメジャーエラーコードを保持。
LastMinorErrCode	最後のマイナーエラーコードを保持。
NotifyRecvChar	CommRecvイベントを発生させる受信文字バイト数
NotifySendComplete	CommSendイベントを発生させます。
NullDiscard	ヌルを受信したら無視する
Parity	パリティ
ParityReplace	パリティ・エラー発生時に置換する文字を設定
Port	通信ポート指定
Progress	ファイル転送の進捗を得る
Protocol	ファイル転送プロトコルを指定
RecvQSize	受信キュー・サイズ
RecvTimeOut	受信タイムアウト
RTSEnable	通信開始時にRTSラインをイネーブルにする
SendQSize	送信キュー・サイズ
SendTimeOut	送信タイムアウト
StopBits	ストップ・ビット
WatchPriority	通信監視スレッドの実行プライオリティ
UseOldInitMethod	シリアル通信初期化方法の選択

以下はプロパティ・ウィンドウ表示例です。



AbortTransfer

ファイル転送中にこのプロパティにtrueを設定するとファイル転送を中断します。接続するソフト、タイミングによって切断の結果が異なることがありますので、ご注意ください。

Visual Basic.NETサンプル

```
SerialIO1.AbortTransfer = True
```

BaudRate

シリアル非同期通信の速度を設定します。ウィンドウズでサポートされる通信速度がプロパティ・ウインドウで選択できます。以下の通信速度をプロパティに設定します。プロパティのデータ型はBaudRateValues型で列挙されます。以下の値が設定可能値です。

BaudRateValues	ボーレート(bps)
Baud75	75
Baud110	110

Baud150	150
Baud300	300
Baud600	600
Baud1200	1200
Baud2400	2400
Baud4800	4800
Baud9600	9600
Baud14400	14400
Baud19200	19200
Baud22800	28800
Baud38400	38400
Baud57600	57600
Badu115200	115200

また、通信ポートをオープンしている状態でこのプロパティをセットした場合、変更は有効になりません。次回のポート・オープン時に変更が反映されます。

ByteSize

７ビットまたは８ビットを指定します。プロパティのデータ型はByteSizeValues型です。列挙型のプロパティで以下の値を設定します。

ByteSizeValues	バイト・サイズ
SevenBits	7bit
EngithBits	8bit

また、通信ポートをオープンしている状態でこのプロパティをセットした場合、変更は有効になりません。次回のポート・オープン時に変更が反映されます。

CustomBaudRate

通信速度は通常BaudRateで固定値を設定しますが特定の機器に接続するような場合にはこちらのプロパティで任意の通信速度を設定して通信することが可能です。通信速度の上限・下限はご利用になるパソコンのシリアル通信チップや通信ボードの仕様により異なります。当プロパティ設定が0の場合はBaudRateプロパティを参照して通信速度を決定します。

DebugTrace

プログラム開発時に有効なトレースの出力を指定します。このプロパティをTrueにするとデバッグト

レースを出力します。デバッグトレースはSend/Receiveメソッドについて送受信されたデータをvbmtrace.exeで16進または10進ダンプ表示します。デバッグトレースを使うとデータをvbmtrace.exeに転送するオーバー・ヘッドが発生します。タイミング・クリティカルなアプリケーションではこのメソッドをTrueに設定するとアプリケーションの動作が変わってしまう場合もありますのでご注意ください。

DTREnable

DTRラインの制御を指定します。プロパティをTrueに設定すると、通信開始時にDTRラインをイネーブル状態に設定します。また、通信ポートをオープンしている状態でこのプロパティをセットした場合、変更は有効になりません。次回のポート・オープン時に変更が反映されます。以下はサンプル･コードです。

Visual Basic.NETサンプル

```
SerialIO1.DTREnable = True
```

ErrorNotifyType

VBMan Components for RS-2323Cでは通信等のエラーの通知方法として通常の例外と従来版とのコンパチビリティを考慮したイベントでの通知を選択することが出来ます。データ型はErrorNotifyValuesになります。

ErrorNotifyValues	意味
ByException	例外を発生させてエラーを通知します。デフォルト値になります。例外はCommException型がスローされます。
ByEvent	ActvieX版とのコンパチビリティを考慮してOnCommErrorイベントにてエラーを通知します。

FlowControl

フロー制御を設定します。なし、ハードウェア、XOn/XOffが設定可能です。データ型はFlowControlValuesになります。通信ポートをオープンしている状態でこのプロパティをセットした場合、変更は有効になりません。次回のポート・オープン時に変更が反映されます。プロパティのデータ型は列挙型で以下の値を設定します。

FlowControlValues	フロー制御
NoFlowControl	なし
SoftFlowControl	XON/XOFF制御
HardFlowControl	ハードウェア制御

ハードウェアフロー制御を選択した場合、 Windows APIレベルでRTS(request-to-send),DTR(data-terminal-ready)フロー制御を設定し、CTS(clear-to-send)タイムアウト、DSR(data-set-ready)タイムアウトは30msに設定します。

Visual Basic.NETサンプル

```
With SerialIO1
    .Port = CommLib.PortValues.Com1
    .FlowControl = CommLib.FlowControlValues.SoftFlowControl
End With
```

InBufferCount

プロパティを参照した時点での受信バッファ内にある受信データのバイト数を返します。参照のみ可能です。

Visual Basicサンプル

```
Dim b(1) As Byte
While SerialIO1.InBufferCount > 0
   SerialIO1.Receive(b)
   If b(0) = 1 Then
      Exit While
    End If
End While
```

LastMajorErrCode

通信エラーが発生した場合にはイベントまたは例外にてアプリケーションプログラムにメジャーエラーコードとマイナーエラーコードが通知されます。このプロパティはこのプロパティを参照した時点で最後に発生したエラーのメジャーエラーコードを返します。参照のみ可能です。

LastMinorErrCode

通信エラーが発生した場合にはイベントまたは例外にてアプリケーションプログラムにメジャーエラーコードとマイナーエラーコードが通知されます。このプロパティはこのプロパティを参照した時点で最後に発生したエラーのマイナーエラーコードを返します。参照のみ可能です。

NotifyRecvChars

通信バッファに受信したバイト数がこのプロパティに指定する値以上になると、CommRecvイベントが発生します。このプロパティに-1を設定した場合、CommRecvイベントは発生しません。プロパティに

0を指定した場合は受信したデータのバイト数に関係なく受信データが存在すればCommRecvイベントが発生します。以下はサンプル･コードです。

Visual Basic.NETサンプル

```
SerialIO1.NoticyRecvChars = 1
...
...

‘ 1 byte受信したら以下のイベントが発生する。
Private Sub SerialIO1_OnCommReceive(ByVal sender As Object, _
      ByVal args As CommLib.CommReceiveEventArgs) _
      Handles SerialIO2.OnCommReceive
   Dim sje = System.Text.Encoding.GetEncoding("shift-jis")
   Dim s As String
   s = sje.GetString(args.Data())
   System.Diagnostics.Debug.WriteLine(s)
End Sub
```

また、通信ポートをオープンしている状態でこのプロパティをセットした場合、変更は有効になりません。次回のポート・オープン時に変更が反映されます。X-Modemによるファイル受信をする場合にはこのプロパティを-1に設定する必要があります。-1以外の値をセットしている場合はファイル転送プロトコルエラー等が発生しますのでご注意ください。

NotifySendComplete

送信完了イベントの発生の有無を指定します。このプロパティをTrueに設定した場合、データの送信が完了して送信バッファが空になるとCommSendイベントが発生します。また、通信ポートをオープンしている状態でこのプロパティをセットした場合、変更は有効になりません。次回のポート・オープン時に変更が反映されます。

NullDiscard

ヌル文字の処理方法を設定します。このプロパティをTrueに設定するとヌル文字を受信した場合は無視され、ユーザー・プログラムには返されなくなります。また、通信ポートをオープンしている状態でこのプロパティをセットした場合、変更は有効になりません。次回のポート・オープン時に変更が反映されます。

Parity

パリティ・ビットを設定します。データ型はParityValuesです。以下の値が設定可能です。

ParityValues	パリティ

NoParity	なし
OddParity	奇数
EvenParity	偶数

また、通信ポートをオープンしている状態でこのプロパティをセットした場合、変更は有効になりません。次回のポート・オープン時に変更が反映されます。

ParityReplace

パリティ・エラーが発生した場合に置き換える文字を指定します。デフォルトは"?"です。通信ポートをオープンしている状態でこのプロパティをセットした場合、変更は有効になりません。次回のポート・オープン時に変更が反映されます。

Port

通信ポートを選択します。プロパティのデータ型は列挙型でPortValuesです。COM20までの値を設定可能です。当プロパティはOpenCommメソッドの呼び出し前に設定してください。また、通信ポートをオープンしている状態でこのプロパティをセットした場合、変更は有効になりません。次回のポート・オープン時に変更が反映されます。

Visual Basic.NETサンプル

```
SerialIO1.Port = CommLib.PortValues.Com1
```

Progress

ファイル転送の進み具合を示します。プロパティのデータ型はshortです。Xmodemプロトコルによる転送をしている場合は128バイト単位のパケット数がこのプロパティに設定されます。参照のみ可能です。

Protocol

ファイル転送プロトコルを設定します。プロパティのデータ型はProtocolValuesです。以下の設定が可能です。Xmodemプロトコルを利用する場合、通信パラメータは8bit,パリティ1,フロー制御なしに設定する必要があることにご注意ください。

ProtocolValues	プロトコルと意味
XmodemAuto	Xmodem CRC, CheckSum,1Kを自動切り替え
XmodemCheckSum	Xmodem CheckSum
XmodemCrc	Xmodem CRC
Xmodem1K	Xmodem 1K

RecvQSize

受信キュー・サイズをバイト単位で整数で指定します。通信開始前に設定される必要があります。デフォルトは1024バイトです。コードで設定する場合は以下のようになります。受信・キューの最大サイズはオペレーティング・システムで用意される通信デバイス・ドライバーの仕様に依存します。また、通信ポートをオープンしている状態でこのプロパティをセットした場合、変更は有効になりません。次回のポート・オープン時に変更が反映されます。

Visual Basic.NETサンプル

```
With SerialIO1
    Try
      .RecvQSize = 128
      .SendQSize = 128
      .OpenComm()
    Catch ex As CommLib.CommException
      System.Diagnostics.Debug.WriteLine(ex.Message)
    End Try
End With
```

RecvTimeOut

Receiveメソッドで文字列を受信するとき、受信バイト数をパラメータで指定した場合に受信タイムアウトをこのプロパティで指定可能です。単位はミリ・セカンド(1/1000秒)です。プロパティのデータ型はLong型です。タイム・アウトはエラー・イベントにERR_RECV_TIMEOUTが発生し、RecvStringにはその時点まで受信した文字列が返されます。４９日以上連続稼動しているパソコンでは動作しない可能性もありますので、ご注意ください。[1]

Visual Basic.NETサンプル

```
Dim r$
     With SerialIO1
       .RecvTimeOut = 1000  '一秒
       .Receive(r$)
 End With
```

RTSEnable

このプロパティをTrueに設定すると通信開始時にRTSラインをイネーブルにします。通信ポートをオープンしている状態でこのプロパティをセットした場合、変更は有効になりません。次回のポート・オープン時に変更が反映されます。以下はサンプル･コードです。

---

[1] タイム・アウトの計測にWin32APIのGetThickCountを使っています。このAPIは49日でカウントがラップするので、連続稼動はできないAPIです。現実にはタイムアウトで49日以上を指定することは希と思いますが念のため記述しています。

Visual Basic.NETサンプル

```
Comm.RTSEnable = True
```

SendQSize

送信キューのサイズをバイト単位で整数で指定します。デフォルトは1024バイトです。このプロパティは通信開始前に設定される必要があります。コードで設定するサンプルは、RecvQSize を参照してください。また、通信ポートをオープンしている状態でこのプロパティをセットした場合、変更は有効になりません。次回のポート・オープン時に変更が反映されます。

SendTimeOut

送信タイムアウトをmsec単位で指定します。通信のために呼び出すAPIのパフォーマンス（COMMデバイス・ドライバーも関係します）によっては正確なmsec単位でのタイムアウトができない場合もありますので、ご了承ください。プロパティの値に０を指定した場合はタイムアウトしないで送信が完了するまで待ちになります。

Visual Basic.NET サンプル

```
Dim rc As Short
Const ERR_SEND_TIMEOUT = 158

SerialIO1.SendTimeOut = 1000 'タイムアウトを一秒
rc = SerialIO.Send("atz" & vbCrLf)
If rc = ERR_SEND_TIMEOUT Then
    MsgBox("送信タイムアウトです")
End If
```

ShowErrorMessage

当プロパティがTrue設定の場合には、X-Modemファイル転送中にエラーが発生した場合にエラー内容をメッセージボックスで表示します。

StopBits

ストップ・ビットを設定します。1,1.5,2ビットを設定可能です。プロパティのデータ型はStopBitsValuesです。設定は以下の対応になります。

StopBitValues	ストップビット
OneStopBit	1
OneAndHalrStopBit	1.5

TwoStopBit	2

また、通信ポートをオープンしている状態でこのプロパティをセットした場合、変更は有効になりません。次回のポート・オープン時に変更が反映されます。

Stream

System.IO.Streamクラスから導出したSerialStreamクラスを返します。.NET frameworkのStreamクラスのようにコードしたい場合はこのプロパティを参照してSerailStreamクラスのメソッドにてシリアル通信コードを記述することが出来ます。

WatchPriority

VBMan Components for RS-2323Cでは通信イベントの監視をバックグラウンドのスレッドで処理しています。このプロパティでは通信イベント監視スレッドの優先順位を指定します。たとえば、通信中に同時に実行しているデータ・ベース・アクセスが極端に遅くなるような場合はこのプロパティの値を調節してCPUパワーの配分を調節することができます。以下の５段階の値をこのプロパティに設定することができます。プロパティのデータ型はPriorityValuesとなります。このプロパティはポートのオープン時にスレッドが起動されるときに参照されます。ポート・オープン中にこのプロパティを設定した場合には設定値は有効にはなりません。ポートをオープンする前にこのプロパティの値を設定してください。

PriorityValues	意味
Lowest	最低
Below Normal	通常より低い
Normal	通常
Above Normal	通常より高い
Highest	最高

UseOldInitMethod

True設定の場合はシリアル通信の初期化について旧来のWin32 APIを使用します。デフォルト値はFalse設定となります。

カスタム・メソッド

ここではVBMan components for RS-232Cで利用可能なメソッドについて説明します。Visual Basic等からこれらのカスタム・メソッドの呼び出しコードを記述することにより、通信アプリケーションを作成します。

メソッド名	詳　　細
ClearBreak	ブレーク状態のクリア
ClearDTR	DTRラインのクリア
ClearRTS	RTSラインのクリア
CloseComm	通信の終了
Crc16	CRC-16を計算
Crc32	CRC-32を計算
FlushComm	通信キューにあるデータを破棄
GetCTS	CTSライン状態を得る
GetDSR	DSRライン状態を得る
GetRLSD	RLSDの状態を得る
GetRing	Ringの状態を得る
OpenComm	通信の開始
Receive	データを受信
ReceiveFile	ファイル受信の開始
Send	データを送信
SendBreak	プレーク信号送信
SendFile	ファイル送信の開始
SetDTR	DTRラインをオンにする
SetRTS	RTSラインをオンにする
Stream	SerailStreamクラスを返します
Transmit	バッファされている送信データより優先して１バイトを送信

ClearBreak

書式
ClearBreak() As Short

戻り値
メジャー通信エラー･コード。エラー･コード一覧を参照してください。

解説
通信ポートをブレーク状態から通常の通信状態にもどします。戻り値はエラー・イベントに渡される最初のパラメータ(MajorErrorCode)と同じ値です。

Visual Basic.NETサンプル
```
Dim rc As Integer
rc = SerialIO1.ClearBreak
```

ClearDTR

書式
ClearDTR() As Short

戻り値
メジャー通信エラー･コード。エラー･コード一覧を参照してください。

解説
DTRラインをオフにします。戻り値はエラー・イベントに渡される最初のパラメータ(MajorErrorCode)と同じ値です。

Visual Basic.NETサンプル
```
Dim rc As Integer
rc = SerialIO1.ClearDTR
```

ClearRTS

書式
ClearRTS() As Short

戻り値
メジャー通信エラー･コード。エラー･コード一覧を参照してください。

解説
RTSラインをオフにします。戻り値はエラー・イベントに渡される最初のパラメータ(MajorErrorCode)と同じ値です。

Visual Basic.NETサンプル

```
Dim rc As Short
rc = SerialIO1.ClearDTR
```

CloseComm

書式
CloseComm() As Short

戻り値
メジャー通信エラー･コード。エラー･コード一覧を参照してください。

解説
通信ポートをクローズします。以下はサンプルです。戻り値はエラー・イベントに渡される最初のパラメータ(MajorErrorCode)と同じ値です。

Visual Basic.NETサンプル

```
Dim rc As Short
rc = SerialIO1.CloseComm
```

Crc16

書式
Crc16(byteArra() As Byte) As Uint16

パラメータ
CRC16を計算するデータ領域をByte配列で指定します。

戻り値
CRC16計算結果

解説
CRC16を計算します。計算範囲は指定したByte配列の全要素となります。シリアル通信には直接関係ありませんが旧バージョンにて提供されていたユーティリティの機能を提供しています。

Crc32

書式
Crc32(byteArray() As Byte)　As Long

パラメータ
CRC16を計算するデータ領域をByte配列で指定します。

戻り値

CRC32計算結果

解説
CRC32を計算します。計算範囲は指定したByte配列の全要素となります。シリアル通信には直接関係ありませんが旧バージョンにて提供されていたユーティリティの機能を提供しています。

FlushComm

通信キュー・バッファをフラッシュします。システムの通信バッファにある未送信データは破棄されます。受信データについても同様に破棄されます。

書式
FlushComm(QueueType As Integer) As Short

パラメータ
メソッドの動作を指定します。以下の値が指定可能です。

値	キュー・タイプ
1	実行中の送信処理を中断します
2	実行中の受信処理を中断します
4	送信キューをクリアします
8	受信キューをクリアします

戻り値
MajorErrorCodeの値を返します。

解説
通信キュー・バッファのデータを破棄します。送受信両方のキューを1度のメソッド呼び出しで破棄する場合は値１２を指定します。

GetCTS

書式
GetCTS(ByRef cts As Boolean) As Short

パラメータ
CTSラインの状態を論理値で返します。True時にCTSはオンです。

戻り値
MajorErrorCodeの値を返します。

解説
CTSラインの状態を取得します。一般的にはラインステータスが変わった時点でOnCommNotifyイベントが発生しますから、このイベント中でどのラインのステータスがどのように変化したのかを取得するためにこのメソッドを使います。

Visual Basic.NETサンプル

```
Private Sub SerialIO1_OnCommLine(ByVal sender As Object, _
      ByVal args As CommLib.CommLineEventArgs) _
      Handles SerialIO1.OnCommLine
   Dim rc As Short
   Dim b As Boolean

   rc = SerialIO1.GetCTS(b)
   If b = True Then
     System.Diagnostics.Debug.WriteLine("CTS is on")
   Else
     System.Diagnostics.Debug.WriteLine("CTS is off")
   End If
End Sub
```

GetDSR

書式
GetDSR(ByRef dsr As Boolean) As Short

パラメータ
DSRラインの状態を論理値で返します。Trueの場合にDSRはオンです。

戻り値
MajorErrorCodeの値を返します。

解説
DSRラインの状態を取得します。一般的にはラインステータスが変わった時点でOnCommNotifyイベントが発生しますから、このイベント中でどのラインのステータスがどのように変化したのかを取得するためにこのメソッドを使います。

Visual Basic.NETサンプル

```
Private Sub SerialIO1_OnCommLine(ByVal sender As Object, _
      ByVal args As CommLib.CommLineEventArgs) _
      Handles SerialIO1.OnCommLine
   Dim rc As Short
   Dim b As Boolean

   rc = SerialIO1.GetDSR(b)
```

```
    If b = True Then
      System.Diagnostics.Debug.WriteLine("DSR is on")
    Else
      System.Diagnostics.Debug.WriteLine("DSR is off")
    End If
End Sub
```

GetRLSD

書式
GetRLSD(ByRef rlsd As Boolean) As Short

パラメータ
RLSD[2]の状態を論理値で返します。

戻り値
MajorErrorCodeの値を返します。

解説
RLSD信号の状態を取得します。

Visual Basic.NETサンプル

```
Private Sub SerialIO1_OnCommLine(ByVal sender As Object, _
      ByVal args As CommLib.CommLineEventArgs) _
      Handles SerialIO1.OnCommLine
    Dim rc As Short
    Dim b As Boolean

    rc = SerialIO1.GetRLSD(b)
    If b = True Then
      System.Diagnostics.Debug.WriteLine("RLSD is on")
    Else
      System.Diagnostics.Debug.WriteLine("RLSD is off")
    End If
End Sub
```

GetRing

書式
GetRing(ByRef ring As Boolean) As Short

パラメータ
Ringラインの状態を論理値で返します。Ring時にはTrueを返します。

[2] receive-line-single-detect

解説
Ringラインの状態を取得します。一般的にはラインステータスが変わった時点でOnCommNotifyイベントが発生しますから、このイベント中でどのラインのステータスがどのように変化したのかを取得するためにこのメソッドを使います。

Visual Basic.NETサンプル

```
Private Sub SerialIO1_OnCommLine(ByVal sender As Object, _
ByVal args As CommLib.CommLineEventArgs) _
Handles SerialIO1.OnCommLine
    Dim rc As Short
    Dim b As Boolean

    rc = SerialIO1.GetRing(b)
    If b = True Then
      System.Diagnostics.Debug.WriteLine("Ringing ! ")
    End If
End Sub
```

OpenComm

書式
OpenComm() As Short

戻り値
メジャー通信エラー・コード。エラー・コード一覧を参照してください。

解説
通信ポートをオープンします。以下はサンプル・コードです。戻り値はエラー・イベントに渡される最初のパラメータ(MajorErrorCode)と同じ値です。

Visual Basic.NETサンプル

```
Private Sub Form1_Load(ByVal sender As System.Object, _
   ByVal e As System.EventArgs) Handles MyBase.Load
 SerialIO1.OpenComm()
End Sub
```

Receive

書式
Receive(Buffer() As Byte) As Short
Receive(Buffer() As Byte, Size As Integer) As Short
Receive(Buffer() As Byte, Offset As Integer, Size As Integer) As Short
Receive(ByRef Buffer As String) As Short

Receive(ByRef Buffer As String, Size As Integer) As Short
Receive(ByRef Buffer As Int16) As Short
Receive(ByRef Buffer As Int32) As Short
Receive(ByRef Buffer As Int64) As Short
Receive(ByRef Buffer As Single) As Short
Receive(ByRef Buffer As Double) As Short
Receive(Pointer As IntPtr) As Short

パラメータ
各オーバーロードにて最初のパラメータは受信するデータになります。パラメータを２個指定するタイプのオーバーロードは第２パラメータは受信するデータのサイズになります。パラメータを３個指定するタイプのオーバーロードでは第２パラメータはオフセット、第３パラメータはサイズになります。

戻り値
メジャー通信エラー･コード。エラー･コード一覧を参照してください。

解説
シリアル通信により回線に送信されてきたデータを受信します。シリアル通信に送られてくるデータは基本的にはバイト型のデータですが、プログラミング上の利便性を考慮して.NET　frameworkの基本的な型に変換してデータを受信します。データを受け取る領域はこのメソッドの呼び出し以前に確保しておく必要があります。当メソッドで指定するデータのサイズとオフセット単位はバイトになります。受信サイズ指定または配列サイズバイト数またはデータサイズを受信するまでブロックされます。String型に受信する場合は回線からのデータはShift-JIS漢字コードと仮定して変換されます。

Visual Basic.NETサンプル

```
Dim buffer(10) As Byte

SerialIO1.Receive(Buffer)
```

Visual C#サンプル

```
Byte [] buffer = new Byte[10];
Short rc = SerialIO1.Receive(buffer);
```

ReceiveFile

書式
ReceiveFile(FileName As String) As Integer

パラメータ
受信するファイル名。フルパス指定しない場合はカレント･ディレクトリにあるファイルに受信します。

戻り値
メジャー通信エラー･コード。エラー･コード一覧を参照してください。

解説

X-Modemプロトコルによるファイル受信を開始します。パラメータは受信するファイル名です。このメソッドを使う前にProtocolプロパティにファイル転送プロトコルを設定します。ファイル転送中の進捗状況はProgressプロパティにあります。ファイル転送が終了すると、OnCommTransferEndイベントが発生します。ファイル転送は時間が掛かるのでウィンドウズアプリケーションの場合はスレッドから呼び出すようにしてください。

Visual Basic.NETサンプル

```
Private Sub ReceiveThreadProc()
    Dim rc As Short

    rc = SerialIO2.ReceiveFile("c:\tmp\receive.txt")
End Sub

Private Sub SendThreadProc()
    Dim rc As Short
    rc = SerialIO1.SendFile("c:\tmp\send.txt")
End Sub

Private Sub Start_transfer_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles Button2.Click

    Dim rth = New System.Threading.Thread(AddressOf ReceiveThreadProc)
    rth.Start()

    Dim sth = New System.Threading.Thread(AddressOf SendThreadProc)
    sth.Start()
End Sub
```

Send

書式

```
Send(data As Byte) As Short
Send(data()As Byte) As short
Send(data() As Byte, offset as Integer, length As Integer) As Short
Send(ptr As IntPtr) As Short
Send(data As String) As Short
Send(data As Int16) As Short
Send(data As Int32) As Short
Send(data As Int64) As Short
Send(data As Single) As Short
Send(data As Double) As Short
```

パラメータ

各オーバーロードにて最初のパラメータは送信するデータになります。配列指定でサイズの無いオーバー

ロードは配列の先頭から配列の要素全てを送信します。配列指定でサイズとオフセット指定があるものはオフセット、サイズで指定される配列要素のみ送信します。

戻り値

メジャー通信エラー・コード。エラー･コード一覧を参照してください。

解説

データを送信します。文字列データに漢字が含まれる場合はShift-JIS漢字コードに変換して送信されます。

Visual Basic.NETサンプル

```
Dim rc As Integer
Dim data(5) as Byte

Data(0) = 5
Data(1) = 32
Data(2) = 41
Data(3)= 42
Data(4) = 6

rc = SerialIO1.Send(data)
```

SendBreak

書式

SendBreak() As Short

戻り値

メジャー通信エラー・コード。エラー･コード一覧を参照してください。

解説

通信ポートをブレーク状態にします。以下はサンプルです。ブレーク状態の解除にはClearBreakメソッドを使用します。戻り値はエラー・イベントに渡される最初のパラメータ(MajorErrorCode)と同じ値です。

Visual Basic.NETサンプル

```
Dim rc As Integer
rc = SerialIO1.SendBreak
```

SendFile

書式

SendFile(FileName As String) As Short

パラメータ

送信するファイル名。フルパス指定しない場合はカレント･ディレクトリのファイルを指定したものとみなします。

戻り値
メジャー通信エラー・コード。エラー･コード一覧を参照してください。

解説
X-Modemプロトコルによるファイル送信を開始します。パラメータは送信するファイル名です。このメソッドを使う前にProtocolプロパティにファイル転送プロトコルを設定します。ファイル転送中の進捗状況はProgressプロパティにあります。ファイル転送が終了すると、OnCommTransferEndイベントが発生します。ウィンドウズアプリケーションでは時間の掛かる処理になるのでスレッドでこのメソッドを起動することをお勧めします。

参照
ReceiveFileメソッド、OnCommTransferEndイベント

SetDTR

書式
SetDTR() As Short

戻り値
メジャー通信エラー・コード。エラー･コード一覧を参照してください。

解説
DTRラインをオンにします。

SetRTS

書式
SetRTS() As Short

戻り値
メジャー通信エラー・コード。エラー･コード一覧を参照してください。

解説
RTSラインをオンにします。

Transmit

書式
Transmit(byteDatar As Byte) As Short

パラメータ

送信する 1 文字。

戻り値

メジャー通信エラー・コード。エラー･コード一覧を参照してください。

解説

1 文字を送信します。指定された文字は、送信バッファの先頭に置かれます。当メソッドの送信結果はトレースツールには表示されません。

Visual Basic.NETサンプル

```
Dim rc As Short
Dim b As Byte

b = 13
rc = SerialIO1.Transmit(b) ' 改行コードを送る
```

カスタム・イベント

この章ではVBMan components for RS-232Cのカスタム・イベントについて解説します。当通信コントロールでは以下のカスタム・イベントがサポートされます。

イベント名	概要
CommError	通信エラーイベント
CommNotify	ライン状況イベント
CommRecv	データ受信イベント
CommSend	データ送信イベント
CommTransferEnd	ファイル転送終了イベント

カスタムイベントはコンポーネント内部のスレッドでイベントを起動していますので、イベントハンドラ内でで、スレッドセーフでないプロパティやメソッドにアクセスすることが出来ません。このような場合はDelegateを使って処理を委譲するコードが必要になります。以下は受信イベントでタイマーを起動するコード例です。

```
delegate void StartTimerDelegate ();
private void StartTimer () { timer1.Start(); }
private void timer1_Tick (…) { timer1.Stop(); serialIO1.Send("ACK"); }
private void serialIO1_OnCommReceive (…)
{
  Invoke(new StartTimerDelegate(StartTimer), null);
}
```

OnCommError

書式

```
OnCommError(ByVal sender As Object, _
            ByVal args As CommLib.CommErrorEventArgs)
```

パラメータ

sender
エラーを通知したオブジェクトへの参照。

args
エラーの詳細を通知するCommErrorEventArgs型のパラメータです。CommErrorEventArgsにはMajorErrorCodeとMinorErrorCodeが含まれます。これらの値を参照することで通信エラーの詳細を得ることが出来ます。コードの詳細はAppendix-Aに記載がされています。

概要
VBMan components for RS-232Cは通信エラーを例外またはイベントで通知することが出来ます。イベントで通知する場合にはErrorNotifyTypeプロパティをByEventに設定にすると上記の形式でエラー

イベントが発生します。

OnCommLine

書式

```
OnCommLine(ByVal sender As Object, _
        ByVal args As CommLineEventArgs)
```

パラメータ

sender
エラーを通知したオブジェクトへの参照。

Args
CommLineEventArgs型のパラメータによりEventMask値を通知します。

概要

VBMan components for RS-232Cはライン状況が変化した場合にイベントでユーザー・プログラムに通知します。ライン状況イベントに渡されるパラメータはイベントマスク値です。通知されるライン状況はDTR,CTS,RLSD,Ringです。

イベント・マスク値は以下の値です。(16進表示)

CTS	0x0008
DSR	0x0010
RLSD	0x0020
RING	0x0100

OnCommRecv

書式

```
OnCommReceive(ByVal sender As Object, _
    ByVal args As CommReceiveEventArgs)
```

パラメータ

sender
エラーを通知したオブジェクトへの参照。

args
受信したデータを保持するCommReceiveEventArgs型のデータです。バイト配列データがメンバーの

Dataに保持されます。

概要

プロパティNotifyRecvCharに0以外の値を設定されており、通信中にこのプロパティの設定値以上に受信バッファにデータが存在する時CommRecvイベントが発生します。このイベントにはその時点で通信バッファにあるデータがすべて渡されます。このイベントはコンポーネント内部の通信監視スレッドから呼び出されるため、このイベント内ではCloseCommメソッドを呼び出すことは出来ません。

Visual Basic.NETサンプル

```
Private Sub SerialIO2_OnCommReceive(ByVal sender As Object, _
      ByVal args As CommLib.CommReceiveEventArgs) _
      Handles SerialIO2.OnCommReceive
Dim sje = System.Text.Encoding.GetEncoding("shift-jis")
Dim s As String
s = sje.GetString(args.Data())
System.Diagnostics.Debug.WriteLine(s)
End Sub
```

OnCommSend

書式

```
OnCommSend(ByVal sender As Object, _
            ByVal args As CommSendEventArgs)
```

パラメータ

sender
イベントを通知したオブジェクトへの参照。

args
特別な情報は保持しません。

概要

プロパティNotifySendCompleteにTrueの値が設定されており、送信キューが空になった時にCommSendイベントが発生します。

Visual Basic.NETサンプル

```
Private Sub SerialIO1_OnCommSend(ByVal sender As Object, _
      ByVal args As CommLib.CommSendEventArgs) _
      Handles SerialIO1.OnCommSend
   System.Diagnostincs.Debug.WriteLine("send queue is empty")
End Sub
```

OnCommTransferEnd

書式

```
OnCommTransferEnd(ByVal sender As Object, _
                  ByVal args As CommTransferEndEventArgs)
```

パラメータ

sender
イベントを通知したオブジェクトへの参照。

args
CommTransferEndEventArgs型でメンバーのResultsにファイル転送の結果を保持します。

概要

ファイル転送メソッドReceiveFile,SendFileメソッド完了時に発生するイベントです。ファイル転送が終了すると、このイベントに転送の終了が通知されます。イベントにはファイル転送の完了コードが返されます。詳細はエラー・コード一覧を参照してください。

Visual Basic.NETサンプル

```
Private Sub SerialIO_OnCommTransferEnd( _
        ByVal sender As Object, _
        ByVal args As CommLib.CommTransferEndEventArgs) _
            Handles SerialIO1.OnCommTransferEnd
System.Diagnostics.Debug.WriteLine("transfer end")
End Sub
```

デバッグトレース表示ツール

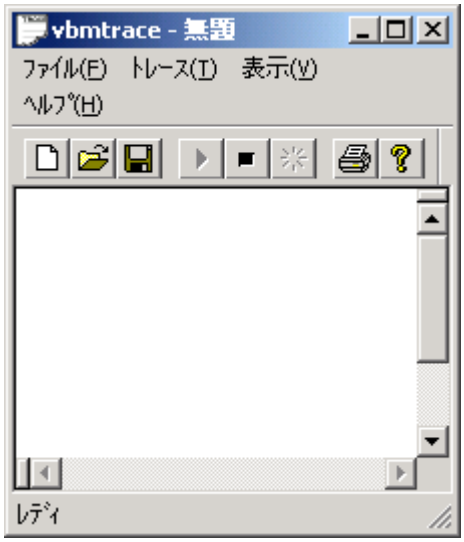
この章ではVBMan components for RS-232Cのデバッグトレース表示ツールについて説明します。

コンポーネントプロパティ設定

デバッグトレースを開始するにはコンポーネントのDebugTraceプロパティをTrueに設定してから通信を開始します。

起動方法

プログラムメニューのVBMan Components for RS-232Cから「デバッグトレース表示ツール」を選択します。以下のようなウィンドウが表示されます。

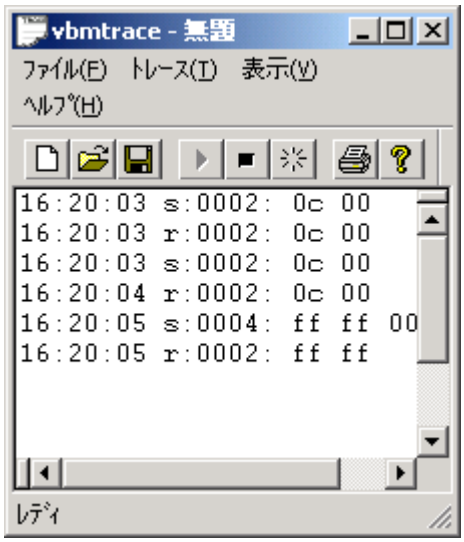


トレース スタート/ストップ

起動後にはトレースはスタート状態となります。トレース取得を止めたい場合にはメニューからトレーススストップを選択します。

トレース表示

通信トレースはデフォルトで１６進表示で以下のように表示されます。１０進表示したい場合には「表示」メニューの下の「１６進表示」のチェックをはずします。



ファイルへの保存

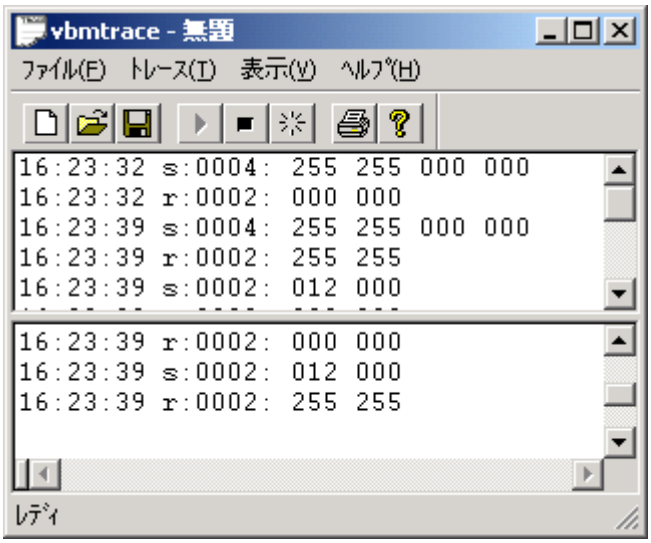
「ファイル」メニューから「名前を付けて保存」を選択すると保存するファイル名の問い合わせダイアログが表示されますので保存ファイル名を設定すると、表示されている画面の内容がテキストファイルに出力されます。

トレースデータのクリア

「トレース」メニューからクリアを選択すると現在表示されているデータがすべて消去されます。

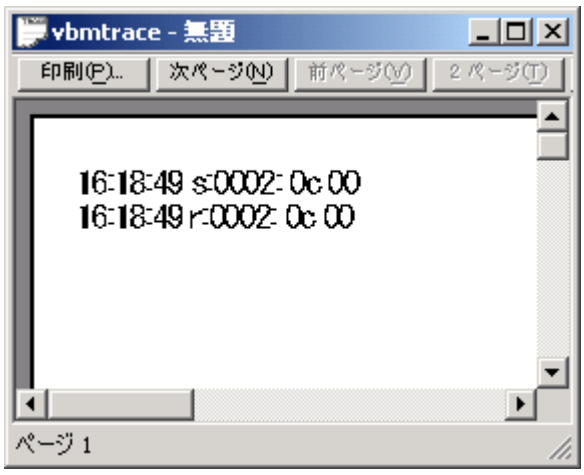
画面分割表示

「表示」メニューから分割を選択するとマウスにより表示エリアを上下・左右に最大４分割表示することが出来ます。以下は上下２画面に表示する例です。



印刷および印刷プレビュー

印刷および印刷プレビューが可能です。以下は印刷プレビュー画面の例です。


vbmtrace - 無題
印刷(P)...
次ページ(N)
前ページ(V)
2 ページ(T)
16:18:49 s:0002: 0c 00
16:18:49 r:0002: 0c 00
ページ 1

Appendix-A　エラー・コード

メジャー・エラー・コード

以下はエラー・イベント・プロシージャに通知される最初のパラメータ(MajroErrorCode部分)の説明です。

ERR_OPEN	100	通信ポートがオープンできません。
ERR_BUILD_DCB	101	Data Control Blockが作成できませんでした。
ERR_COMM_STATE	102	通信状態エラー。詳細はMinorErrorCodeを調べてください。
ERR_NO_MEM	103	メモリが不足しています。
ERR_BUFFER_SHORT	104	受信するキューのサイズが小さい。 キュー・サイズを大きくしてください。
ERR_READ_COMM	105	通信ポートから読み込めません。 詳細はMinorErrorCodeを調べてください。
ERR_WRITE_COMM	106	通信ポートに出力できません。 詳細はMinorErrorCodeを調べてください。
ERR_CLEAR_BREAK	107	ブレーク状態をクリアできません。
ERR_SET_BREAK	108	ブレーク状態に移行できません。
ERR_TRANSMIT_CHAR	109	SendCharに失敗しました。
ERR_INVALID_SIZE	110	SendCharで文字列が指定されました。
ERR_NOT_OPEN	111	通信ポートがオープンしていません。
ERR_ALREADY_OPEN	112	２度通信ポートのオープンを試みました。
ERR_INVALID_DEVICE_NAME	113	通信ポートの指定が不正です。
ERR_FLUSH_COMM	114	通信キューの廃棄に失敗しました。
ERR_RECV_TIMEOUT	115	受信タイムアウト
ERR_CREATE_EVENT	116	イベントの作成に失敗しました。
ERR_RECV_LENGTH_TOO_LONG	117	RecvStringメソッドへのパラメータが大きすぎます。
ERR_THREAD	118	スレッドの作成に失敗しました。
ERR_CLEAR_DTR	119	DTRのクリアに失敗しました。
ERR_SET_DTR	120	DTRラインをセットできません。
ERR_CLEAR_RTS	121	RTSラインをクリアできません。
ERR_SET_RTS	122	RTSラインをセットできません。
ERR_GET_MODEM_STATUS	123	モデムの状態を取得することに失敗しました。
ERR_COMM_LINE	124	ライン状態を取得することに失敗しました。
ERR_IN_TRANSFER	125	ファイル転送中です。
ERR_NOT_IN_TRANSFER	126	ファイル転送中ではありません。
ERR_SEND_TIMEOUT	127	送信タイムアウトです。

ERR_TYPE_INVALID	128	Byte型の配列を指定してください。また指定するByte型の配列は１次元のみ指定可能です。
ERR_NO_DATA	129	Receiveメソッドの呼び出し時点で通信バッファにはデータが存在しませんでした。
ERR_SHOULD_BE_8_BITS	130	X-MODEM通信のメソッドが呼び出されましたが、ByteSizeプロパティが7Bit設定になっています。X-MODEM通信では8Bit設定が必要です。
ERR_INVALID_ARRAY_SIZE	131	オフセットやサイズ指定のあるメソッドにおいて、配列サイズを超えてオフセット/サイズ指定されています。
ERR_NO_EVENT_HANDLER_FOR_TRANSFER_END_EVENT	132	ファイル転送終了イベントが発生していますが、イベントハンドラーの定義がありません。
ERR_INCORRECT_STREAM_CREATION	133	ストリームを参照しましたが通信インスタンスが生成されていないようです。
ERR_NO_EVENT_HANDLER_FOR_RECV_EVENT	134	受信イベントが発生していますがイベントハンドラーの定義がありません。
ERR_NO_EVENT_HANDLER_FOR_SEND_EVENT	135	送信完了イベントが発生していますがイベントハンドラーの定義がありません。

マイナー・エラー・コード

エラー・イベント・プロシージャに通知される2番目のパラメータはGetLastError APIからの戻り値です。エラーの詳細はWindows SDKのドキュメント等を参照してください。エラーの定義はWindows SDKにあるヘッダー・ファイルwinerror.hにあります。

https://msdn.microsoft.com/en-us/library/windows/desktop/ms681382(v=vs.85).aspx

ファイル転送ステータス

以下はファイル転送のステータスです。CommTransferEndカスタム・イベントのパラメータに設定される値です。

シンボル	値	意味
ERR_XMODEM_FILE_EXSIST	2101	バイナリ・ファイル受信で指定したファイルがすでに存在します。
ERR_XMODEM_FILE_OPEN	2102	ファイルをオープンできません。
ERR_XMODEM_FILE_READ	2103	ファイルの読み込みエラー。ディスクまたはファイルシステムが破損しています。chkdsk,scandiskでディスクを検査してください。

ERR_XMODEM_FILE_WRITE	2104	ファイルの書き込みエラー。ディスクまはたファイルシステムが破損しているか、ファイルシステムに空領域が無いと思われます。chkdsk,scandiskでディスクを検査してください。
ERR_XMODEM_SEND_CHAR	2105	文字を送信できません。
ERR_XMODEM_SEND_SHORT	2106	CRCまたはシーケンス送信エラー
ERR_XMODEM_SEND_PACKET	2107	パケットを送信できません。
ERR_XMODEM_RECV_CHAR	2108	文字受信エラー
ERR_XMODEM_RECV_SHORT	2109	CRCまたはシーケンス受信エラー
ERR_XMODEM_RECV_PACKET	2110	パケット受信エラー
ERR_XMODEM_SEQ	2111	シーケンス番号に誤り
ERR_XMODEM_PROTOCOL	2112	プロトコル指定が誤り
ERR_XMODEM_RETRY_OUT	2113	リトライ・エラー
ERR_XMODEM_ABORT_TRANSFER	2114	ファイル転送の中断

Appendix-B　トラブルシューティング

ここではVBMan components for RS-232Cを使ってアプリケーションを開発する場合に多く発生するトラブルについての解決方法を記述します。最新の情報、追加情報につきましてはテクナレッジのサポートwebをご参照ください。URLは以下になります。

http://www.techknowledge.co.jp/tn2/techinfo.shtml

USBシリアルアダプタでの利用

弊社でRS-232c機器をUSBポートから接続するための変換アダプタをいくつかテストしました。問題なくご利用いただけることを確認しております。最近のPCでは1ポートしかサポートしない機種も多いので、複数ポートをご利用になる場合に便利だと思います。

Xmodemで転送中に進捗状況を表示したい

Xmodemプロトコルでは転送するデータ・サイズをあらかじめ交換しないため、何パーセント転送済みなのかを知ることはできません。アプリケーションでパーセント表示が必要な場合はあらかじめ転送するファイルサイズを交換するプログラムを作成して、このプロパティからパケット数を得て計算してください。ファイル転送されたパケット数はProgressプロパティに保持されていますので、X-Modemプロトコルの場合はこれに128バイトのパケットサイズを乗算するとファイル転送したサイズを求めることができます。また、X-Modem 1Kの場合は1024を乗算します。

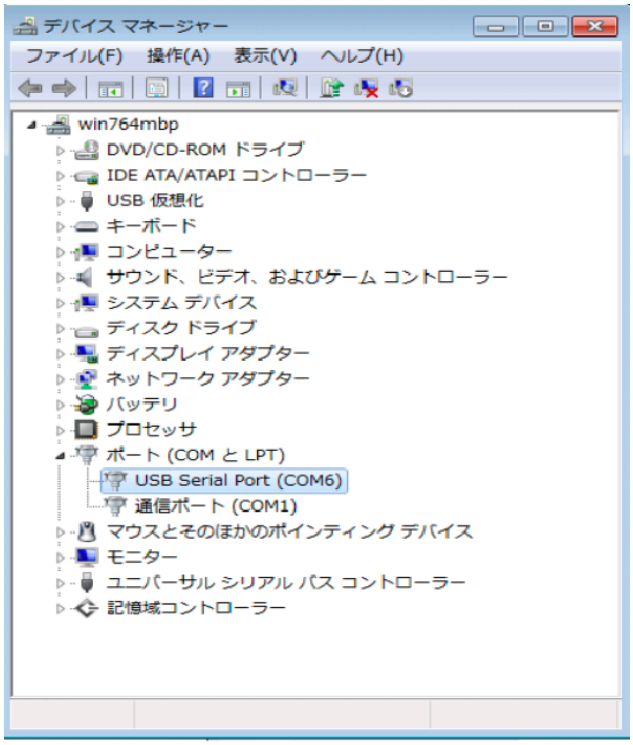
開発したアプリケーションのインストールに含めるファイルはどれか？

VBMan Components for RS-232Cを使って開発されたアプリケーションを配布するセットアップにはCommLib.DLLとCommShm.DLL(64BIT OSではCommShm64.DLL)を含めることでアプリケーションのモジュール依存を解決することが出来ます。配置するディレクトリですがCommLib.DLLに関してはアプリケーションの実行ファイルと同じディレクトリで問題ありませんが、CommShm.DLLはシステムディレクトリ等のパスで指定されるディレクトリに配置してください。またこれら DLL Microsoft VC++ 2022 でビルドしていますので再配布ライタイム (VC_REDIST) に依存します。

https://www.microsoft.com/ja-jp/download/details.aspx?id=26999

どのポートが有効か確認する方法

コントロールパネルのデバイスマネジャーで接続可能なポートを知ることができます。



Appendix-C 付録

シリアル通信

パーソナルコンピュータは、外部と通信するために、通常２種類のI/Oポートを備えています。一つは、モデムを使った通信に利用するシリアルポートで、もう一つは、プリンタとの接続に使うパラレルレポートです。

シリアルポートは、１本の線を使って１ビットずつ送受信するので、ビットデバイスと呼ばれます。ビットデバイスは、同じ情報を送るのにバイトデバイスの８倍の時間が必要ですが、２～３ほんの先からなる安価なケーブルを使える利点があります。実際、双方向通信に必要なのは、送信用、受信用、接地用の３本だけです。

双方向通信

双方向通信には、半二重方式と全二重方式があります。半二重方式は、データを双方向に送りますが、送信中には受信が、また受信中には送信ができません。半二重方式は、モデム間の通信方式としてよく使われます。全二重方式は、送信しながら同時に受信もできる方式です。コンピュータのシリアルポートは全二重方式を採用しており、送信と受信には別の線を使います。１つの回線で全二重通信をサポートしているモデムもあります。

双方向通信のほかに、データを一方方向にしか送信できない単方向通信があります。これは最も単純な通信方式で、端末は受信専用、ホストは送信専用として働きます。パラレルプリンタポートでは、コンピュータからプリンタに一方的にデータを送るだけなので、この方式を採用しています。

スタートビットとストップビット

非同期通信でデータを送る時は、データビットの前後にスタートビットとストップビットを送信します。データビット長は５、６、７または８ビットに設定します。送信側と受信側は、スタートビットとストップビットのタイミングと同様に、このデータビット長も合わせる必要があります。

データビット長を７ビットにすると、127以上のASC II コードは送ることができません。５ビットでは、最高でも３１までのASC II コードに制限されます。データビットに続いて送信するストップビットの値は１（マーク状態）で、直前のビットの値が１でも、この値は正しく検出されます。なお、ストップビット長は１、1.5、２ビットのいずれかに設定します。

パリティビット

パリティビットは、転送中に生じた誤りを検出するためのもので、データビットとストップビットの間に挿入します。

このパリティビットは、データビット中のマーク状態（値が１）の数が偶数か奇数かを１ビットで表します。パリティには、マーク状態が偶数個の時にパリティビットの値を０にする偶数パリティと、奇数個のときに値を０にする奇数パリティがあります。例えば、偶数パリティを選択すると、データ0110011のパリティビットは０になり、データ11010110のパリティビットは１になります。

パリティビットを使った誤り検出は、完全なものではありません。１ビットの誤りは検出できますが、ビット誤りが遇数個（例えば、値１のビット２個を値０として誤って受信した時）あれば、検出できません。また、パリティビットは、誤りを検出するだけで訂正することはできません。

フロー制御

シリアルデータの場合、データは連続して送信側から受信側へ送られます。受信したデータは、直ちに読み取らなければなりません。読み取る前に次のデータが到着すると、直前のデータが失われてしまうからです。そこで、受信したデータを読み取るまでの間、受信バッファにデータを保存します。これにより、データを受信してから読み取るまで時間に余裕ができるので、データ受信中に別の処理を実行できるようになります。

受信バッファをメモリに割り当てたり、受信バッファにデータを読み込んだりするのは、通信ソフトです。バッファにデータを書き込む速さよりも通信ソフトがデータを読む速さの方が遅いと、バッファはすぐに一杯になり、その後に受信するデータはすべて失われます。そこで、受信バッファが一杯になった時は、シグナルを送って送信を停止し、受信バッファが空いてから再びシグナルを送って送信を再開します。このシグナルのやり取りをハンドシェークと呼び、ハンドシェークを使ってデータの流れを調整することをフロー制御といいます。

ＲＳ－２３２Ｃインタフェース

RS－232Cの”RS”は標準仕様(Recommend Standard)を意味します。また、”232”は標準仕様の認識番号で、”C”はその標準仕様の最新版であることを表しています。大部分のコンピュータのシリアルポートはRS-232Cに準拠しています。RS-232Cは25ピンの”D”コネクタ（そのうち２２ピンを使用）を使うことになっています。しかし、ほとんどのピンはパーソナルコンピュータ間の通信に必要ないので、最近では9ピンのコネクタがよく使われます。

VBMan components for RS-232C 8.00調査依頼

以下の調査依頼フォームに必要事項を記入してユーザー・サポートまでファックスまたはインターネットでメールしてください。折り返し担当者が技術サポートの連絡を差し上げます。

日付	
会社名	
登録ユーザー名	
製品シリアル番号	
電話番号	
ファックス番号	
メール・アドレス	
接続デバイス	
OSとバージョン	
言語とバージョン	
問題点など具体的に再現可能な様にご記入ください。	
添付資料	

VBMan components for RS-232C
version 8.00

マニュアル第1版
2022年1月15日　第1刷発行

Printed In Japan