

VBMan Components for RS-232C Programming Manual

Version 6.00

TechKnowledge

VBMan Components for RS-232C

はじめに	6
製品の概要	6
開発ライセンス	7
ランタイムライセンス	8
保証規定	8
ユーザーサポート	8
販売元、ユーザーサポート	11
開発元	11
商標登録	11
インストール	13
システム条件	13
セットアップの起動	13
インストール・ファイル一覧	14
プロジェクト開始	15
IDEへの登録方法	15
Visual Basic.NET	15
Visual C#	16
Visual J#	17
Visual C++	17
NAMESPACE/CLASS名について	17
コンパチビリティ	18
.NET FRAMEWORK 1.1 版とのコンパチビリティ	18
COM コンポーネント版とのコンパチビリティ	18
カスタム・プロパティ	20

AbortTransfer	22
BaudRate.....	22
ByteSize	23
CustomBaudRate.....	23
DebugTrace	23
DTREnable.....	23
ErrorNotifyType	24
FlowControl	24
InBufferCount	25
LastMajorErrCode.....	25
LastMinorErrCode	25
NotifyRecvChars.....	25
NotifySendComplete	26
NullDiscard.....	26
Parity.....	27
ParityReplace.....	27
Port.....	27
Progress	27
Protocol.....	28
RecvQSize.....	28
RecvTimeOut.....	29
RTSEnable	29
SendQSize.....	29
SendTimeOut.....	30
ShowErrorMessage	30
StopBits	30
Stream	31
WatchPriority	31
カスタム・メソッド.....	32

ClearBreak.....	33
ClearDTR.....	33
ClearRTS.....	33
CloseComm.....	34
Crc16.....	34
Crc32.....	35
FlushComm.....	35
GetCTS.....	36
GetDSR.....	37
GetRLSD.....	37
GetRing.....	38
OpenComm.....	39
Receive.....	39
ReceiveFile.....	40
Send.....	42
SendBreak.....	42
SendFile.....	43
SetDTR.....	43
SetRTS.....	44
Transmit.....	44
カスタム・イベント.....	46
OnCommError.....	46
OnCommLine.....	47
OnCommRecv.....	48
OnCommSend.....	48
OnCommTransferEnd.....	49
デバッグトレース表示ツール.....	51
コンポーネントプロパティ設定.....	51

起動方法	51
トレース スタート/ストップ	51
トレース表示	52
ファイルへの保存	52
トレースデータのクリア	52
画面分割表示	52
印刷および印刷プレビュー	53
APPENDIX-A エラー・コード	54
APPENDIX-B トラブルシューティング	58
USBシリアルアダプタでの利用	58
Xmodemで転送中に進捗状況を表示したい	58
拡張ボードでの動作	58
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?	59
APPENDIX-C 付録	60
シリアル通信	60
双方向通信	60
スタートビットとストップビット	60
パリティビット	61
フロー制御	61
RS-232Cインタフェース	61
VBMAN COMPONENTS FOR RS-232C 6 00 調査依頼	63

はじめに

製品の概要

VBMan components for RS-232C version 6.00をお買いあげくださり誠にありがとうございます。当製品はMicrosoft Visual Studio.NET 2005/Microsoft .NET framework 2.0からご利用いただけるRS-232Cシリアル通信をサポートするカスタム・コンポーネント製品です。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コントロールの時代を経て、このたびVisual Studio.NETのコンポーネント形式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 2.0のComponentModelクラス仕様に沿って作成されていますのでVisual Studio.NET 2005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。当製品のディスクを複製して、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。
- 当製品の使用权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。
- 使用权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きます。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い合わせください。

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ファッ

クスでご送付ください。ウェブの場合は以下のURLからご登録ください。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。必ずご返送/ご登録をお願いいたします。

ユーザー登録URL:

<http://www.systemlab.co.jp/vbman/productvb7.htm>

- 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、システムラボの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お問い合わせ事項を記入していただき、それをファックス、インターネット・メールまたはe-mail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いたしておりませんので、ご了承をお願いいたします。また、問い合わせの内容によっては、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、かさねて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、よろしくをお願いいたします。

- 併用される他社製品について

当社製品と併用される、他社製品の使い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、最初のお問い合わせから90日間となつ

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も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。到着順に処理いたしますのでよろしくお願いいたします。

SAMPLE

販売元、ユーザーサポート

Systemlab®

株式会社システムラボ

東京都杉並区上荻1丁目5番8号 直長ビル7F

電話: 03-5397-7511

ファックス: 03-5397-7521

サポートメール: support@systemlab.co.jp

Web: www.systemlab.co.jp

開発元

TechKnowledge

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

当マニュアルに記載される商標または登録商標は該当各社様の商標または登録商標です。

SAMPLE

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 6.0 が対応するフレームワークは以下になります。

- Microsoft .NET framework 2.0

VBMan components for RS-232C は Microsoft .NET framework 2.0 がサポートする以下のオペレーティング・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 2000
- Windows XP
- Windows 2003 Server

VBMan components for RS-232C は以下の Microsoft .NET 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++

IDE(統合開発環境)としてのサポートは以下になります。

- Microsoft Visual Studio.NET 2005

セットアップの起動

VBMan components for RS-232C の CD-ROM をドライブに挿入しセットアップ・プログラム(setup.exe)を起動します。セットアップ・プログラムではインストール・ディレクトリを指定するだけで、コントロール・パックのインストール完了します。フォルダー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。

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。オペレーティングシステムのシステムディレクトリを<SYSDIR>と表記します。

モジュールの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を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	RS-232C 通信コンポーネント	可
<sysdir>%CommShm.DLL	共有メモリーモジュール	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>%man%CommLib600.html	リリースノート	不可
<instdir>%man%CommLib600.pdf	PDF マニュアル	不可
<instdir>%Samples%VB.NET% *.*	VB.NET 用サンプルプログラム	不可
<instdir>%Samples%CS% *.*	C#用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

最初に Visual Studio 2005 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 Express 版も含めてどの IDE でも共通です。

Visual Basic.NET

Microsoft Visual Basic.NET 2005 から VBMan components for RS-232C を使う方法を説明します。

- ① Visual Studio.NET 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。

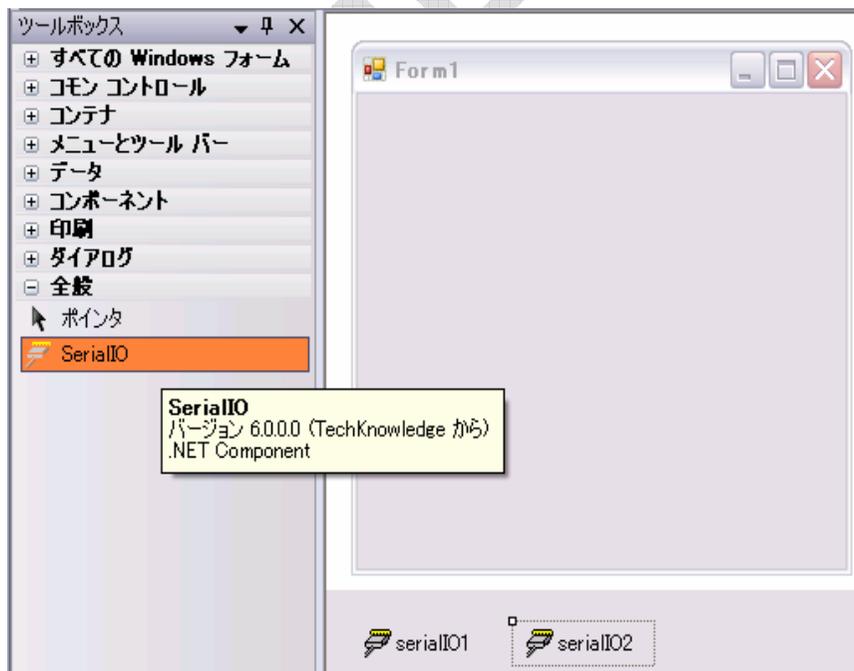
c:\Program Files\techknowledge\VBMan Components for RS-232C 6.00\bin

- 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 J#

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

Visual C++

Visual C++の場合、アプリケーションの種類として.NET を手順③で選択してください。(Managed C++アプリケーションを選択する)それ以外の手順は Visual Basic.NET の場合と同一です。

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ー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。

コンパチビリティ

.NET framework 1.1 版とのコンパチビリティ

当製品の一つ前バージョンである VBMan Components for RS-232C 5.0x は Microsoft .NET framework 1.1 と Visual Studio.NET 2003 を動作環境としておりました。当製品バージョン 6.0 ではそれぞれ新しいバージョンである Microsoft .NET framework 2.0 と Visual Studio.NET 2005 に対応したもので、基本的な動作には変更はありません。

Visual Studio .NET 2005 のバージョン変換ウィザードでお客様の既存のプロジェクトを変換した場合には参照される ComLib.DLL は旧バージョンの 5.0.1 または 5.0.0 を参照してプロジェクトが変換されます。従って参照設定を ComLib.DLL バージョン 6.0.0 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完全に .NET framework 2.0 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

ここでは旧バージョンの VBMan Control for RS-232C V4.0 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た、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やす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になりました。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。

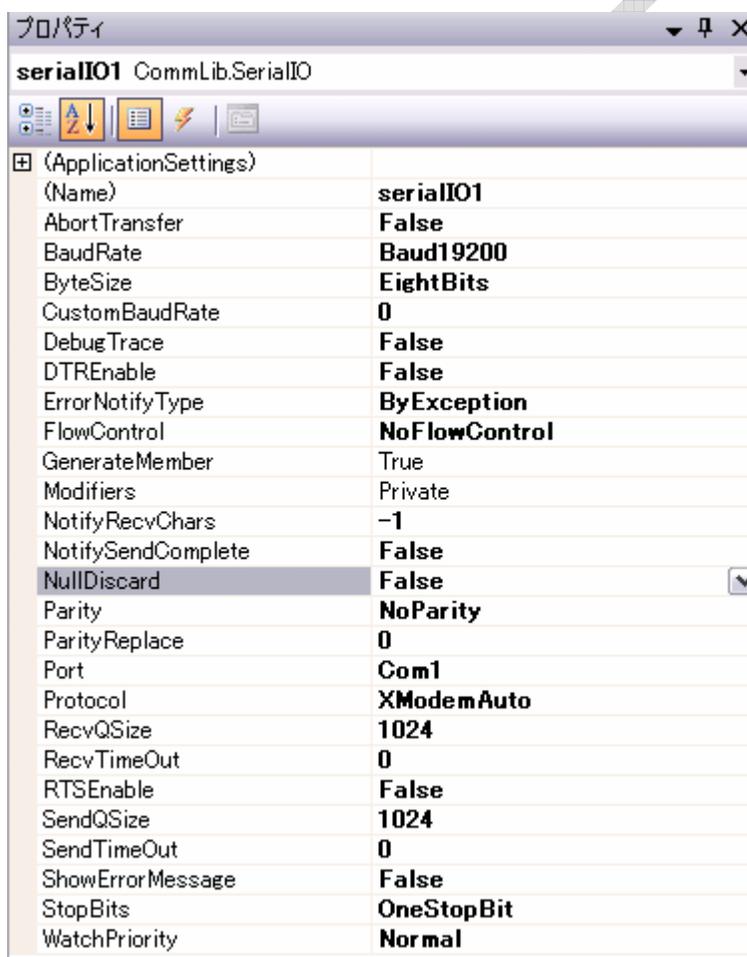
カスタム・プロパティ

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

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・ウィ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
Baud115200	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。

DIREnable

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-232C 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	ActiveX 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 Serial101.InBufferCount > 0
    Serial101.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

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せん。

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

```
Serial101.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ります。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 ¹

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 Serial101
Try
    .RecvQSize = 128
    .SendQSize = 128
    .OpenComm()
Catch ex As CommLib.CommException
```

¹ Xmodem-1Kを使用する場合、プロトコルではデータ・パケットは1,024バイトですが、送受信データがバッファリングされることを考慮して、SendQSize,RecvQSize プロパティを 1,200 バイト程度に設定してください。Xmodem-1Kでは送信側でSTXを送信することでプロトコルを切り替えるため、受信側のXmodem-1Kの設定は必須ではありません。

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・秒(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日以上連続稼働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。²

Visual Basic.NETサンプル

```
Dim r$  
With Serial101  
    .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

```
Comm.RTSEnable = True
```

SendQSize

² 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、連続稼働はできない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。

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

Serial101.SendTimeout = 1000 ' タイムアウトを一秒
rc = Serial101.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

OneAndHalfStopBit	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 SerialStream 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できます。

WatchPriority

VBMan Components for RS-232C 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

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	ブレーク信号送信
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	SerialStream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

通信ポートをブレイ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 Serial101_OnCommLine(ByVal sender As Object, _  
    ByVal args As CommLib.CommLineEventArgs) _  
    Handles Serial101.OnCommLine  
Dim rc As Short  
Dim b As Boolean  
  
rc = Serial101.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 Serial101_OnCommLine(ByVal sender As Object, _  
    ByVal args As CommLib.CommLineEventArgs) _  
    Handles Serial101.OnCommLine  
    Dim rc As Short  
    Dim b As Boolean  
  
    rc = Serial101.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

RLSD³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 Serial101_OnCommLine(ByVal sender As Object, _  
    ByVal args As CommLib.CommLineEventArgs) _  
    Handles Serial101.OnCommLine  
    Dim rc As Short  
    Dim b As Boolean  
  
    rc = Serial101.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。

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化した時点で 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。

³ receive-line-single-detect

Visual Basic.NETサンプル

```
Private Sub Serial101_OnCommLine(ByVal sender As Object, _  
    ByVal args As CommLib.CommLineEventArgs) _  
    Handles Serial101.OnCommLine  
    Dim rc As Short  
    Dim b As Boolean  
  
    rc = Serial101.GetRing(b)  
    If b = True Then  
        System.Diagnostics.Debug.WriteLine("Ringin!")  
    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  
    Serial101.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 = Serial102.ReceiveFile("c:¥tmp¥receive.txt")  
End Sub
```

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

```
rc = Serial101.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

通信ポートをブレイク状態にします。以下はサンプルです。ブレイク状態の解除には 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 = SerialO1.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 = Serial101.Transmit(b) ' 改行コードを送る
```

SAMPLE

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、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 CommEventArgs 型のパラメータです。CommEventArgs 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 Serial102_OnCommReceive (ByVal sender As Object, _  
                                     ByVal args As CommLib.CommReceiveEventArgs) _  
    Handles Serial102.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

エラー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 SerialI01_OnCommSend(ByVal sender As Object, _  
    ByVal args As CommLib.CommSendEventArgs) _  
    Handles SerialI01.OnCommSend  
    System.Diagnostics.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

エラー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

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.CommTransferEventArgs) _  
    Handles SerialIO1.OnCommTransferEnd  
    System.Diagnostics.Debug.WriteLine("transfer end")  
End Sub
```

SAMPLE

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。

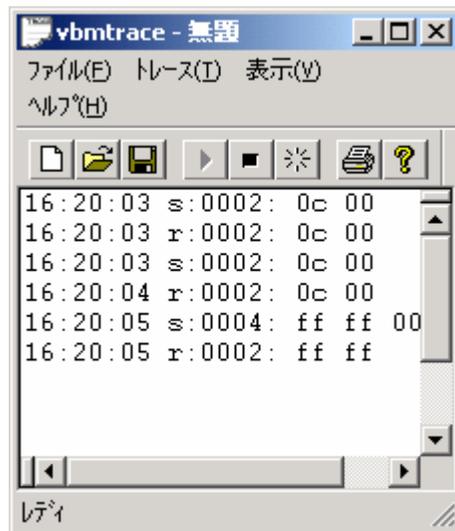


トレース スタート/ストップ

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

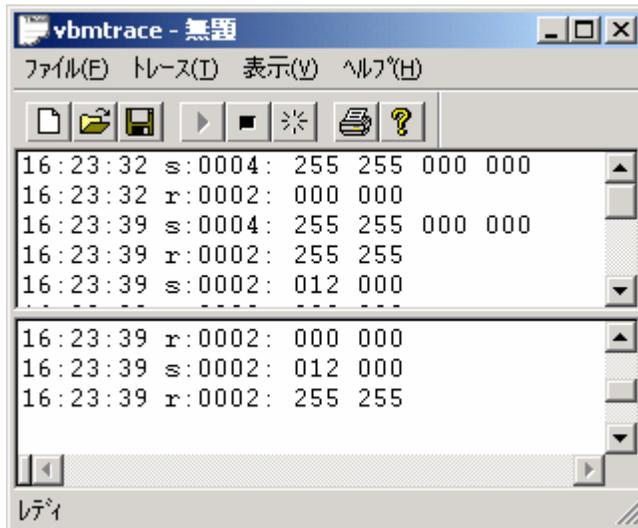
「ファイル」メニューから「名前を付けて保存」を選択すると保存するファイル名の問い合わせダイアログが表示されますので保存ファイル名を設定すると、表示されている画面の内容がテキストファイルに出力されます。

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。

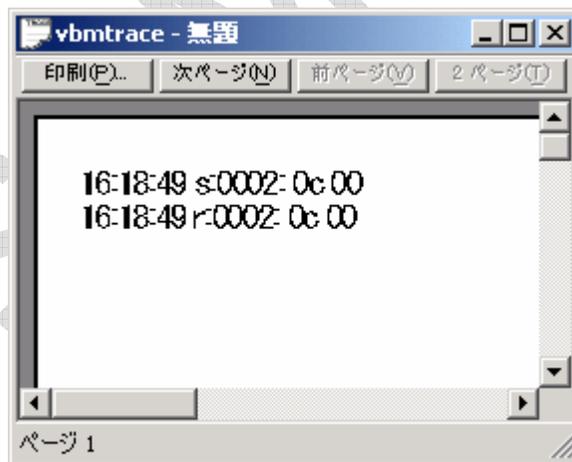
画面分割表示

「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。



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 (MajorErrorCode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레이크状態をクリアできません。
ERR_SET_BREAK	108	ブ레이크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。エラーの詳細はマイクロソフトWin32SDKのドキュメント等を参照してください。エラーの定義はWin32 SDKにあるヘッダー・ファイルwinerror.hにあります。Win32 SDK等の資料はMicrosoft Developer Network (MSDN)より入手できます。

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。ディスクまたは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る。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/techinfo.html>

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。弊社で動作確認したのは PCI コミュニケーション社の URS-03 とトライコーポレーション JUSTY UCR-01 です。

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。

拡張ボードでの動作

VBMan components for RS-232C は各種拡張ボードでの動作確認は基本的にはしていません。VBMan のソフトウェア構造としては Win32 API を呼び出して通信ドライバーとデータを交換します。ハードウェアを直接制御することはありません。通信ポートとして COM1 から COM20 まで指定できるようになっていますが、これらの指定は Windows/Win32 API へのポートのオープン時に一度指定するだけです。従って、各種拡張ボードでの動作はご利用になら

れるOSでメーカー様から提供されるデバイス・ドライバーがWin32 APIを經由して正常に動作するものでしたら VBMan components for RS-232C でも問題なくご利用いただけます。

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 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。

SAMPLE

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、ビット誤りが偶数個(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には、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イクと呼び、ハンドシェイ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。

SAMPLE

VBMan components for RS-232C 6 00 調査依頼

以下の調査依頼フォームをコピーし、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、同様の項目を記入して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。申し訳ありませんが**電話によるサポートは受け付けておりません。ご了承ください。**

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	
使用パソコン機種	
接続デバイス	
OSとバージョン	
言語とバージョン	
お問合わせ内容、問題記述など、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 6.00

マニュアル第2版

2006年1月11日 第1刷発行

版權・著作 株式会社テクナレッジ

Printed In Japan