

シリアル通信コンポーネント

# **VBMan Control for RS-232C**

**Version 4.50**

 **TechKnowledge**

# VBMan Control for RS-232C

はじめに.....	4
開発ライセンス.....	5
ランタイムライセンス.....	6
保証規定.....	7
ユーザー・サポート.....	7
販売元.....	8
開発元/サポート.....	8
インストール.....	9
システム条件.....	9
セットアップの起動.....	9
インストール・ファイル一覧.....	9
プロジェクト開始.....	11
VISUAL BASIC.NET.....	11
VISUAL C#.....	12
VISUAL BASIC 6.0.....	12
カスタム・プロパティ.....	13
VcAutoOpen.....	14
VcBaudRate.....	14
VcByteSize.....	15
VcCustomBaudRate.....	15
VcDeviceName.....	15
VcDebugTrace.....	16
VcDTREnable.....	16
VcFileTransferPriority.....	16
VcFlowControl.....	17
VcInBufferCount.....	17
VcMode.....	18
VcNotifyRecvChars.....	18
VcNotifySendComplete.....	18
VcNullDiscard.....	19
VcParity.....	19
VcParityReplace.....	19
VcProgress.....	19
VcProtocol.....	19
VcRecvEventType.....	20
VcRecvNotifyOnly.....	20
VcRecvQSize.....	20

VcRecvTimeOut.....	21
VcRTSEnable.....	21
VcSendQSize.....	21
VcSendTimeOut.....	21
VcShowErrorMessage.....	22
VcStopBits.....	22
VcWatchPriority.....	22
<b>カスタム・メソッド.....</b>	<b>24</b>
AbortTransfer.....	24
ClearBreak.....	25
ClearDTR.....	25
ClearRTS.....	25
CloseComm.....	26
FlushComm.....	26
GetCTS.....	27
GetDSR.....	27
GetRLSD.....	28
GetRing.....	28
OpenComm.....	29
RecvBinaryBytes.....	29
RecvFile.....	30
RecvString.....	31
RecvUnicodeString.....	32
SendBinaryBytes.....	32
SendBreak.....	33
SendByte.....	33
SendChar.....	34
SendFile.....	34
SendString.....	35
SendUnicodeString.....	35
SetDTR.....	36
SetRTS.....	36
<b>カスタム・イベント.....</b>	<b>36</b>
CommError.....	37
CommNotify.....	37
CommRecv.....	38
CommRecvVar.....	38
CommSend.....	39
CommTransferEnd.....	39
<b>通信ユーティリティ.....</b>	<b>40</b>

カスタム・メソッド.....	40
Crc16.....	40
Crc32.....	41
<b>エラー・コード .....</b>	<b>43</b>
<b>トラブルシューティング .....</b>	<b>45</b>
USBシリアルアダプタでの利用.....	45
Xmodemで転送中に進捗状況を表示したい.....	45
Accessでプロパティが保存されない.....	45
拡張ボードでの動作 .....	45
Visual Basicセットアップ・ウィザードでのモジュール配布.....	45
サポートWEBサイトから新バージョンをダウンロードしてインストールしたら動作が変わった .....	46
Accessでのバイナリ・データ通信 .....	46
<b>付録.....</b>	<b>48</b>
シリアル通信 .....	48
双方向通信 .....	48
スタートビットとストップビット .....	48
パリティビット .....	48
フロー制御.....	49
RS-232Cインタフェース.....	49
<b>VBMAN CONTROL FOR RS-232C調査依頼.....</b>	<b>50</b>

## はじめに

---

VBMan Control for RS-232C version 4.5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等COMコンポーネントをサポートする言語からご利用いただけるRS-232c通信をサポートするカスタム・コントロール製品です。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コントロールを継続して販売・サポートしています。当バージョンから64bit OCXをサポートします。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。
- モジュール・サイズの極小化  
当コンポーネントはVisual Studio.NETでサポートされるATLを使ってビルドされています。ATLを利用しコンポーネントの信頼性、配布の簡便性のためにモジュールサイズの極小化を実現しています。

- バイナリ・データ送信メソッドのサポート  
Windows環境では内部文字コードとしてUnicodeが採用されており、文字列は2バイトで管理されるようになりました。従来の16bit環境ではバイナリ・データも文字列型で管理できましたが、現座のWindowsでバイナリ・データは専用のデータ型を使うように仕様が変更になりました。この仕様変更に対応してVBMan Control for RS-232CではVisual Basic等のByte型を送受信できるメソッドをサポートしています。
- C#言語のサポート  
当バージョンからVisual Studio.NETの新しい言語であるC#をサポートします。サンプルコードを添付しました。
- Unicode文字列送受信のサポート  
当バージョンからUnicode形式のデータをRS-232Cライン上にそのまま展開して送受信するメソッドをサポートします。
- 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。また、当バージョンから受信イベントのパラメータとしてVariant型を選択可能となりました。バイナリ受信時が必要な場合にはコードが簡便になります。
- デバッグ機能  
VcDebugTrace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進表示することが出来ます。アプリケーションのデバッグ効率を改善します。
- マルチ・スレッドによるバックグラウンドファイル転送  
Xmodem Check Sum, Xmodem CRCによるファイル転送が可能です。これらの処理はWIN32マルチスレッドにより効率よく実行されます。
- 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、ファイル転送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。そう
- 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。
- 当製品の著作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  
製品には、ユーザー登録用紙を添付しています。お買い上げのあと、できるだけ早い機会に必要な事項をご記入の上、販売会社システム・ラボまでご送付ください。この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。必ずご返送をお願いいたします。
- お問い合わせの方法  
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。マニュアル最終ページにある調査依頼書にお問い合わせ事項を記入していただき、それをインターネット・メールで support@systemlab.co.jp宛お送りいただければ、折り返しご連絡をいたします。**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。**また、問い合わせの内容によっては、回答に時間がかかる場合があります。ご了承ください。
- 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、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ません。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日間となっております。有償サポートにつきましては販売会社システム・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。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。

## 販売元



株式会社システムラボ

東京都北区田端6-1-1 田端アスカタワー 1 2階

電話: 03-5809-0839  
ファックス: 03-4587-9261  
サポートメール: support@systemlab.co.jp  
Web: www.systemlab.co.jp

## 開発元/サポート



(株) 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。

# インストール

---

VBMan Control 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 Control for RS-232C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。今後の動作環境につきましては弊社webサイトにてご確認ください。

- Windows ~ 11
- Windows Server ~ 2022

VBMan Control for RS-232Cは以下のコントロール・コンテナ・アプリケーションでの動作をサポートしています。

- Microsoft Visual Basic.NET
- Microsoft Visual C#
- Microsoft Visual Basic 6.0
- Microsoft Access 2010-2022
- Microsoft Excel 2010-2022

Microsoft Excelからの利用方法は弊社ウェブに別ドキュメントとサンプルコードをご用意してありますので参照ください。

[http://www.techknowledge.co.jp/pdf/how\\_to\\_serial\\_on\\_excel.pdf](http://www.techknowledge.co.jp/pdf/how_to_serial_on_excel.pdf)

<http://www.techknowledge.co.jp/modules/serialSample.xlsm.zip>

## セットアップの起動

VBMan Control for RS-232CダウンロードしたZIP内のセットアップ・プログラム(setup.exe)を管理者権限で起動します。メニューへVBMan Control for RS-232C 4.5が作成され、PDFマニュアルやサンプル等が登録されます。トライアル版をご利用された場合は事前にトライアル版のアンインストールをお願いいたします。

## 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。デフォルトインストール・ディレクトリ以下の表記になります。デフォルトのインストール・ディレクトリは以下です。

c:\Program Files\TechKnowledge¥VBMan Controls for RS-232C 4.5

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環境のみでのご利用可能となります。

モジュール名	概 要	再配布
bin¥vbmcom32.ocx	32bit シリアル通信モジュール	可
bin¥vbmsm32.dll	32bit 共有メモリーモジュール	可
bin¥vbmtrace.exe	デバッグ・トレース表示ユーティリティ	不可
bin64¥vbmcom64.ocx	64bit シリアル通信モジュール	可
bin64¥vbmsm32.dll	64bit 共有メモリーモジュール	可
man¥vbmcom450.pdf	マニュアル	不可
samples¥access¥barcode.mdb	Access用サンプル	不可
samples¥delphi¥barcode.*	Delphi用バーコード・サンプル	不可
samples¥delphi¥bcsamp.*	Delphi用バーコード・サンプル	不可
samples¥delphi¥binary.*	Delphi用バイナリ送受信サンプル	不可
samples¥delphi¥binsamp.*	Delphi用バイナリ送受信サンプル	不可
samples¥ie¥barcode.html	インターネット・エクスプローラー用サンプル	不可
samples¥vb6¥barcode.vbp	VB6.0用ターミナル・サンプル・フォーム	不可
samples¥vb6¥barcode.frm	VB 6.0用バーコード読み取りサンプル・フォーム	不可
samples¥vb6¥term.vbp	VB6.0用ターミナル・サンプル・プロジェクトファイル	不可
samples¥vb6¥term.frm	VB6.0用ターミナル・サンプル・フォーム	不可
samples¥vb6¥barcode.vbp	VB6.0用バーコード読み取りサンプル・プロジェクトファイル	不可
samples¥vb6¥barcode.frm	VB 6.0用バーコード読み取りサンプル・フォーム	不可
samples¥vb.net¥term¥**	VB.NET用ターミナルサンプル	不可
samples¥cs¥term¥**	C#用ターミナルサンプル	不可
samples¥cs¥binary¥**	C#用バイナリ送受信サンプル	不可

# プロジェクト開始

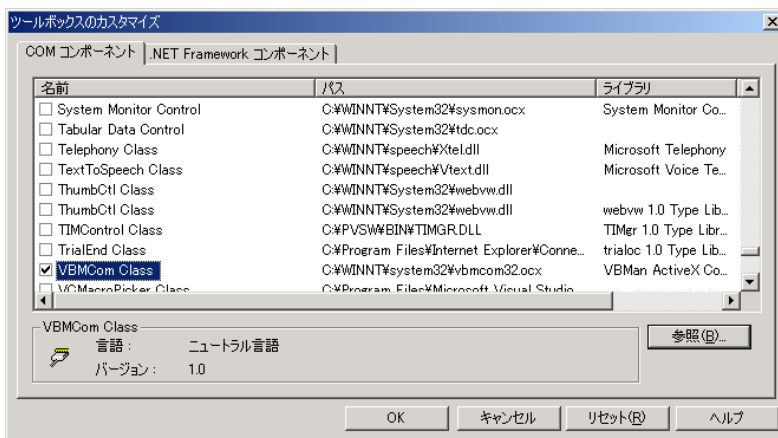
ActiveX コントロールは多くのコンテナ・アプリケーションから利用可能ですが代表的なコンテナとして Visual Basic.NET, Visual Basic 6.0,での利用方法を説明します。

## Visual Basic.NET

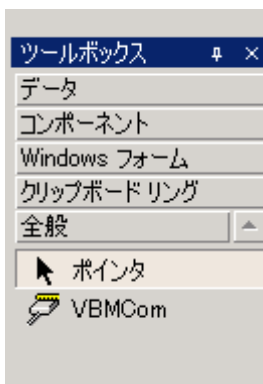
Microsoft Visual Basic.NETからVBMan Control for RS-232Cを使う方法を説明します。

- 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「ツールボックスのカスタマイズ」を選択します。
- ⑦ 「COMコンポーネント」タブを選択します。（デフォルト）
- ⑧ 「参照」ボタンを押し、システム・ディレクトリにあるvbmcom32.ocxを選択します
- ⑨ 追加された「VBMCom」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 Control for RS-232Cを追加した例です。



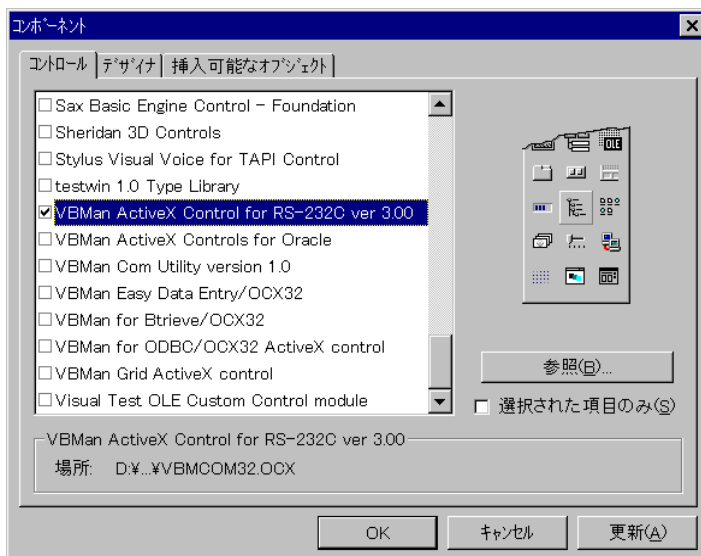
## Visual C#

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 Basic 6.0

Microsoft Visual Basic 6.0からVBMan Control for RS-232Cを使う方法を説明します。

- ① Visual Basic を起動します。
- ② VBMan Control for RS-232Cを組み込みたいプロジェクトをオープンします。
- ③ 「プロジェクト」メニューから「コンポーネント」を選択します。
- ④ リスト・ボックスから「VBMan Control for RS-232C ver 4.00」を選択し、OKボタンを押します。<sup>1</sup>



- ⑤ Visual Basicのツール・ボックスにVBMan Control for RS-232Cコントロールのアイコンが追加されます。
- ⑥ 通信をしたいフォームにVBMan Control for RS-232Cコントロールをマウス・ドラッグで配置します。実行時には不可視になりますのでフォームのデザイン時に邪魔にならない適当な場所に置いてください。
- ⑦ マウスの右ボタンをクリックして「プロパティ」を選択します。以下のようなダイアログが表示されますので、通信条件など基本的な設定をします。

<sup>1</sup> リスト・ボックスにこの項目がない場合は、インストールに失敗していると思われます。当マニュアルのトラブルシューティングを参照して環境を整えて再インストールをしてください。



- ⑧ 通信条件の設定が終わったらVisual Basicのアプリケーションが通信を開始したいイベントに通信のメソッドを記述することでアプリケーションを完成します。

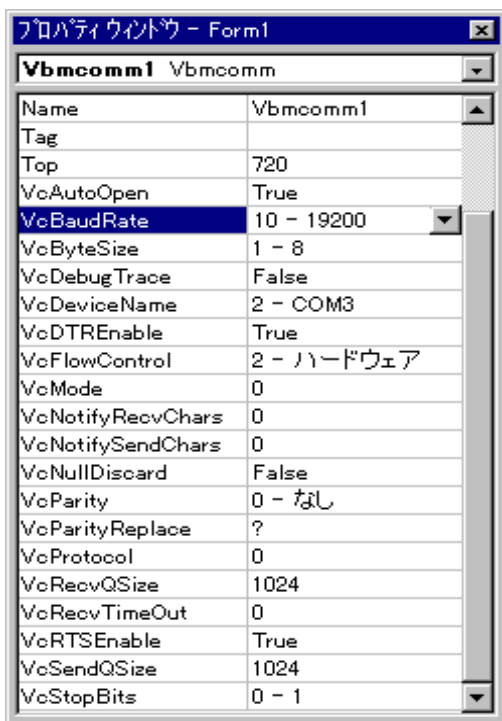
## カスタム・プロパティ

VBMan Control 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
VcAutoOpen	通信の自動開始
VcBaudRate	通信速度
VcByteSize	通信データサイズ
VcDebugTrace	デバッグ・トレースの指定
VcDeviceName	通信ポート指定、(6ポート)
VcDTREnable	通信開始時にDTRラインをイネーブルにする。
VcFileTransferPriority	ファイル転送スレッドのプライオリティを指定
VcFlowControl	フロー制御指定
VcInBufferCount	受信バッファにある文字バイト数。
VcMode	テキスト・バイナリ指定
VcNotifyRecvChar	CommRecvイベントを発生させる受信文字バイト数
VcNotifySendComplete	CommSendイベントを発生させる
VcNullDiscard	ヌルを受信したら無視する
VcParity	パリティ
VcParityReplace	パリティ・エラー発生時に置換する文字を設定
VcProgress	ファイル転送の進捗を得る
VcProtocol	ファイル転送プロトコルを指定

VcRecvQSize	受信キュー・サイズ
VcRecvTimeOut	受信タイムアウト
VcRTSEnable	通信開始時にRTSをラインをイネーブルにする
VcSendQSize	送信キュー・サイズ
VcSendTimeOut	送信タイムアウト
VcShowErrorMessage	ファイル転送中のエラー・メッセージの表示方法を指定
VcStopBits	ストップ・ビット

以下はVisual Basicプロパティ・ウィンドウでの設定例です。



## VcAutoOpen

通信の開始にはポートのオープン・クローズ処理が必要ですが、このプロパティをTrueにした場合はそれらの処理をコントロールの生成、破壊と同時に自動的に処理します。このプロパティをFalseとした場合はアプリケーションにOpenComm,CloseCommメソッドを呼び出すコードが必要です。プロパティ・ウィンドウではチェック・ボックスをチェックした場合にAutoOpenモードとなります。

## VcBaudRate

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short型で列挙されます。以下の値が設定可能値です。

プロパティ値	ボーレート(bps)
0	75

1	110
2	150
3	300
4	600
5	1200
6	2400
7	4800
8	9600
9	14400
10	19200
11	28800
12	38400
13	57600
14	115200

VBMan Control for RS-232CはウィンドウズのAPIを呼び出して通信します。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ても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。

## VcByteSize

---

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short型です。列挙型のプロパティで以下の値を設定します。

プロパティ値	バイト・サイズ
0	7bit
1	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。

## VcCustomBaudRate

---

通信速度は通常VcBaudRate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VcBaudRateプロパティを参照して通信速度を決定します。

## VcDeviceName

---

シリアルポートを選択します。プロパティのデータ型は列挙型で以下の値を設定可能です。<sup>2</sup>

プロパティ値	通信ポート
0	COM1
1	COM2
2	COM3
3	COM4
4	COM5
5	COM6
6	COM7
7	COM8
8	COM9
9	COM10

また、シリアル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。次のオープン時に変更が反映されます。

## VcDebugTrace

---

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/Recv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。

## VcDTREnable

---

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。

```
Comm.VcDTREnable = True
```

## VcFileTransferPriority

---

VBMan Control for RS-2323Cではファイル転送はバックグラウンドのスレッドで処理されます。このプロパティではファイル転送スレッドの優先順位を指定します。たとえば、ファイル転送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。このプロパティはファイル転送中に設定した場合は、実行中のファイル転送のプライオリティには有効ではありません。次のファイル転送から設定した値が有効になります。

---

<sup>2</sup> 10ポートまでプロパティ・ウィンドウで設定可能です。さらに大きな値を設定する場合はポートをオープンする前にコードで設定することで可能です。

プロパティ値	意味
0	Lowest
1	Below Normal
2	Normal
3	Above Normal
4	Highest

## VcFlowControl

---

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。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。

プロパティ値	フロー制御
0	なし
1	XON/XOFF
2	ハードウェア

### Visual Basicサンプル

```
Dim rc As Integer
```

```
Comm.VcDeviceName = 1 ' COM2
Comm.VcFlowControl = 1 ' Xon/Xoff
rc = Comm.OpenComm
```

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。

## VcInBufferCount

---

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。

### Visual Basicサンプル

```
Dim tmp$,txt$
```

```
While Comm.VcInBufferCount <> 0
    tmp$ = Comm.RecvString(1) ' 1文字ずつ受信してEOTを調べる。
    If tmp$ = EOT Then
        Text1.Text = txt$
        txt$ = ""
    Else
```

```

    txt$ = txt$ & Tmp$
End If
Wend

```

## VcMode

---

通信モードを設定します。整数型のプロパティで、以下の値を設定します。

プロパティ値	モード
0	バイナリ
1	テキスト

テキスト・モードの場合は行末にCR+LF( chr\$(13) + chr\$(10) )を追加してデータを送信し、受信の時はCR+LFを削除してユーザー・アプリケーションに返します。バイナリ・モードでは、送受信データがそのまま転送されます。プロパティ・ウィンドウではチェック・ボックスを選択した状態がテキスト・モードとなります。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。次のポート・オープン時に変更が反映されます。

## VcNotifyRecvChars

---

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サンプル

```

Comm.VcNotifyRecvChars = 1
...
' 一文字受信したら以下のイベントが発生する。
Sub Comm_CommRecv( RecvChar As String )
    If RecvChar = Chr$(13) Then
        ' 改行を処理
    End If
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。次のポート・オープン時に変更が反映されます。

## VcNotifySendComplete

---

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。次のポート・オープン時に変更が反映されます。

## VcNullDiscard

---

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。

## VcParity

---

パリティ・ビットを設定します。なし、偶数、奇数が選択可能です。プロパティのデータ型は列挙型です。以下の値が設定可能です。

プロパティ値	パリティ
0	なし
1	奇数
2	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。

## VcParityReplace

---

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。

## VcProgress

---

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。

## VcProtocol

---

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short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。

プロパティ値	プロトコル
0	Xmodem CRC, CheckSum,1Kを自動切り替え
1	Xmodem CheckSum
2	Xmodem CRC
3	Xmodem 1K

## VcRecvEventType

---

受信イベントのタイプを選択します。

プロパティ値	意味
AsString	RecvCommイベントを発生させます。
AsVariantString	RecvCommVarイベントを発生させます。パラメータはVariant型で文字列型データが含まれます。
AsVarinatBinary	RecvCommVarイベントを発生させます。パラメータはVariant型ですがSystem.Byte型データが含まれます。

Visual Basic 6.0等イベントのパラメータとしてVariant型がサポートされていない言語では当プロパティの値をAsStringとしてご利用ください。Visual Basic.NET/Viasul C#におきましてもBinaryの配列はパラメータ型としてサポートされておりませんでしたので（当社調査）、VcNotrifyRecvCharに1以上の値を設定した場合にも1文字ずつイベントが発生しますのでご注意ください。

## VcRecvNotifyOnly

---

このプロパティがFalseに設定された場合は、VcNotfyRecvCharsプロパティで指定したバイト数がOnCommイベントの文字型パラメータでアプリケーションに返されます。

このプロパティがTrueに設定された場合はOnCommイベントのパラメータは常にヌル文字列になります。

OnCommイベントでは必要なメソッドを呼び出して、受信キューにあるデータを読み出すことが可能になります。以下はOnCommイベントでバイナリ・データを受信するサンプルです。

### Visual Basicサンプル

```
Private Sub VBCom2_CommRecv(strRecv As String)
Dim b(0 To 2) As Byte, rc As Integer
rc = VBCom2.RecvBinaryBytes(b)
Debug.Print b(0), b(1), b(2)
End Sub
```

## VcRecvQSize

---

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。次回のポート・オープン時に変更が反映されます。

### Visual Basicサンプル

```
Dim rc As Integer
Comm.VcRecvQSize = 128
```

```
Comm.VcSendQSize = 128
rc = Comm.OpenComm
If rc <> 0 Then
    MsgBox "通信オープンエラー" & Cstr(rc)
End If
```

## VcRecvTimeOut

---

RecvString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。<sup>3</sup>

## Visual Basicサンプル

```
Dim r$

VBManCom1.VcRecvTimeOut = 1000'一秒
r$ = VBManCom1.RecvString(10)
```

## VcRTSEnable

---

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。

```
Comm.VcRTSEnable = True
```

## VcSendQSize

---

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、VcRecvQSize 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。

## VcSendTimeOut

---

送信タイムアウトをmsec単位で指定します。VBManが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、あらかじめご了承ください。プロパティの値に0を指定した場合はタイムアウトはしないで送信が完了するまで待ちになります。

---

<sup>3</sup> 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サンプル

```
Dim rc As Integer
Const ERR_SEND_TIMEOUT = 158

Comm.VcSendTimeOut = 1000      'タイムアウトを一秒
rc = Comm.SendString("atz" & Chr(13) & Chr(10) )
If rc = ERR_SEND_TIMEOUT Then
    MsgBox "送信タイムアウトです"
End If
```

## VcShowErrorMessage

---

ファイル転送中のエラーが発生した場合のメッセージの表示方法を指定します。このプロパティにTrue値を設定した場合にはエラー・メッセージをメッセージ・ボックスで表示します。False値を指定した場合にはメッセージ・ボックスは表示されません。この場合、CommTransferEndイベントに通知されるエラー・ステータスでファイル転送の完了を知ることができます。

## VcStopBits

---

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shortです。設定は以下の対応になります。

プロパティ値	ストップビット
0	1
1	1.5
2	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。次のポート・オープン時に変更が反映されます。

## VcWatchPriority

---

VBMan Control 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。

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。

プロパティ値	意味
--------	----

0	Lowest
1	Below Normal
2	Normal
3	Above Normal
4	Highest

## カスタム・メソッド

---

ここではVBMan Control for RS-232Cで利用可能なメソッドについて説明します。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
AbortTransfer	ファイル転送を中断
ClearBreak	ブ레이크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
FlushComm	通信キューのデータを破棄
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
RecvBinaryBytes	Byte配列を送信
RecvFile	ファイル受信の開始
RecvString	文字列を受信
RecvUnicodeString	Unicode文字列を受信します。
SendBinaryBytes	バイナリを含むByte配列を送信
SendBreak	ブ레이크信号送信
SendByte	現在の送信バッファにあるデータを送信
SendChar	1文字を送信
SendFile	ファイル送信の開始
SendString	文字列を送信
SendUnicodeString	Unicode文字列としてデータをラインに出力します。
SetDTR	DTRラインをオンにする
SetRTS	RTSラインをオンにする

### AbortTransfer

---

#### 書式

AbortTransfer () As Integer

#### 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**

バイナリ・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サンプル**

```
Dim rc As Integer  
rc = VBManComm1.AbortTransfer
```

## **ClearBreak**

---

### **書式**

ClearBreak() As Integer

### **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サンプル**

```
Dim rc As Integer  
rc = VBManComm1.ClearBreak
```

## **ClearDTR**

---

### **書式**

ClearDTR() As Integer

### **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サンプル**

```
Dim rc As Integer  
rc = VBManComm1.ClearDTR
```

## **ClearRTS**

---

### **書式**

ClearRTS() As Integer

### **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サンプル

```
Dim rc As Integer  
rc = VBManComm1.ClearDTR
```

### CloseComm

---

#### 書式

CloseComm() As Integer

#### 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。VcAutoOpenプロパティがTrueの場合は使用できません。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サンプル

```
Dim rc As Integer  
rc = VBManComm1.CloseComm
```

### FlushComm

---

通信キュー・バッファをフラッシュします。

#### 書式

FlushComm(QueueType As Integer) As Integer

#### 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1 2を指定します。

## GetCTS

---

### 書式

GetCTS() As Integer

### 戻り値

CTSラインの状態を論理値で返します。

### 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サンプル

```
Sub Comm1_OnCommNotify()  
Dim status as integer  
  
status = Comm1.GetCTS()  
If status = True Then  
    Debug.Print "CTSラインはオンになりました"  
Else  
    Debug.Print "CTSラインはオフになりました"  
End If  
End Sub
```

## GetDSR

---

### 書式

GetDSR() As Integer

### 戻り値

DSRラインの状態を論理値で返します。

### 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サンプル

```
Sub Comm1_OnCommNotify()  
Dim status as integer
```

```
status = Comm1.GetDSR()
If status = True Then
    Debug.Print "DSRラインはオンになりました"
Else
    Debug.Print "DSRラインはオフになりました"
End If
End Sub
```

## GetRLSD

---

### 書式

GetRLSD() As Integer

### 戻り値

RLSD<sup>4</sup>の状態を論理値で返します。

### 解説

RLSD信号の状態を取得します。

### Visual Basicサンプル

```
Sub Comm1_OnCommNotify()
Dim status as integer

status = Comm1.GetRLSD();
If status = True Then
    Debug.Print "RLSDはオンになりました"
Else
    Debug.Print "RLSDオフになりました"
End If
End Sub
```

## GetRing

---

### 書式

GetRing() As Integer

### 戻り値

Ringラインの状態を論理値で返します。

### 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。

---

<sup>4</sup> receive-line-single-detect

## Visual Basicサンプル

```
Sub Comm1_OnCommNotify()  
Dim status as integer  
  
statuc = Comm1.GetRing();  
If status = True Then  
    Debug.Print "Ringing!"  
End If  
End Sub
```

## OpenComm

---

### 書式

OpenComm() As Integer

### 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

VcAutoOpenプロパティがFalseの場合に、コードで明示的に通信ポートをオープンする場合に使います。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サンプル

```
Sub Form1_Load()  
Dim rc As Integer  
rc = VBManComm1.OpenComm  
If rc <> 0 Then  
    MsgBox "ポートのオープンに失敗 " & CStr(rc)  
End If  
End Sub
```

## RecvBinaryBytes

---

### 書式

RecvBinaryBytes( RecvData As Variant, \_  
Length As Variant ) As Integer

### パラメータ

RecvData

データを受け取るエリアをByte型配列で指定します。

Length

受信する文字列の長さ (バイト)。0を指定した場合は、メソッドを呼び出した時点で受信バッファにあるデータすべてを返します。このパラメータは省略可能で省略した場合は0を指定した場合と同じ動作をします。受信するデータ長を指定した場合は、指定されたバイト数だけ受信するまで、メソッド内部でブロッキングされま

す。パラメータを省略した場合、または0を指定した場合はメソッド内部ではブロックされません。受信データが無い場合は戻り値にERR\_NO\_DATAステータスが設定され制御が呼び出し側に戻されます。

### **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**

Visual Basicを使った場合にはByte型の配列に直接データを受信します。Visual Basic以外のコントロール・コンテナではVARIANT型の1バイトデータ配列としてデータの受信が可能になります。VARIANT型の扱いについては各コントロール・コンテナの仕様またはWind32APIのVARIANT型の説明やSafeArray関連APIの説明をご覧ください。

第2パラメータは省略することができます。第2パラメータを省略した場合または0を指定した場合は、メソッドを呼び出した時点で受信バッファにあるデータより配列サイズが小さい場合は、配列サイズまでデータを読み込み、それを超えるデータについては通信バッファに残されます。

### **Visual Basicサンプル**

```
Dim b(0 To 10) As Byte
Dim rc As Integer, Dim l As Integer
```

```
rc = Comm1.RecvBinaryBytes(b)
If rc <> 0 Then
    MsgBox "エラー " & CStr(rc)
    Stop
End If
```

```
For l = 0 To 9
    Debug.Print Chr(b(l))
Next l
```

## **RecvFile**

---

### **書式**

RecvFile( 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**

マルチスレッド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VcProtocolプロパティにファイル転送プロトコルを設定します。このメソッドはファイル転送のスレッドを起動してすぐに呼び出し側のユーザー・プログラムに制御を戻します。スレッドの起動に失敗した場合はこのメソッドの戻り値を設定してエラーを返します。ファイル転送中の進捗状況はVcStatus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。

## Visual Basicサンプル

```
Sub StartRecvFile()  
Dim rc As Integer, fn$  
fn$ = InputBox("受信ファイル名を入力してください")  
rc = Comm1.ReceiveFile(fn$)  
If rc <> 0 Then  
    MsgBox "ファイル受信を開始できません。"  
End If  
End Sub  
' ファイル転送が終了するとこのイベントが発生する。  
Sub Comm1_OnCommTransferEnd( sStatus As Integer )  
If sStatus = 0 Then  
    MsgBox "ファイル転送が正常に終了しました"  
Else  
    MsgBox "ファイル転送が異常終了しました" & CStr(sStatus)  
End If  
End Sub
```

## RecvString

---

### 書式

```
RecvString( SizeToRecv As Integer ) As String
```

### パラメータ

受信する文字列の長さ（バイト）。0を指定した場合は、受信バッファにあるデータすべてを返します。

### 戻り値

受信した文字列。

### 解説

文字列を受信します。パラメータは受信する文字数（バイト）です。パラメータに0を指定した場合は受信バッファにデータが無い場合はヌルがかえされます。受信する文字数を指定した場合、指定した文字数が得られるまで、VBManコントロール内部でループします。ループしている間はWindowsメッセージ処理ができなくなります。無限ループを避けるためには、パラメータ0を指定し、VBAレベルでDoEventsを使ってメッセージを処理しながら、文字列を受信した方が安全です。

## Visual Basicサンプル

```
Function RecvNChar( NumOfCharsToRecv As Integer ) As String  
Dim r$, Result$  
Static SaveResult$          ' 前回の呼び出しで通信バッファの残りを保存  
  
Result$ = SaveResult$  
SaveResult$ = ""  
  
While Len(Result$) < NumOfCharsToRecv  
    r$ = VBManComm1.RecvString(0)  
    If r$ <> "" Then
```

```

    Result$ = Result$ & r$
End If
DoEvents
Wend

' 結果を処理
If Result$ > NumOfCharsToRecv Then
' 通信バッファには要求されたバイト数より多くのデータが存在した。
    RecvNChar = Left$( Result$, NumOfCharsToRecv )
    SaveResult$ = Right$( Result$, Len(Result$) - NumOfCharsToRecv)
Else
    RecvNChar = Result$
End If
End Function

```

---

## RecvUnicodeString

### 書式

RecvUnicodeString ( Length As Integer ) As String

### パラメータ

受信するデータバイト長です。Unicodeの場合2バイトで1文字になります。0を指定した場合はこのメソッドが呼び出された時点での受信バッファのデータ全てからUnicode文字列を生成します。

### 戻り値

受信したUnicode文字列。

### 解説

ライン上にUnicode形式でデータが送られてきている場合はこのメソッドにて受信することが簡便です。7BITデータについても16BITデータ（2バイト）でデータが送信されていることが必要になります。また、奇数バイトSHIFT-JISデータがライン上に流れている場合は従来のRecvStringメソッド等をご利用ください。

---

## SendBinaryBytes

### 書式

SendBinaryBytes( lpData As Variant ) As Integer

### パラメータ

lpData                      送信するデータ

### 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

Visual Basicをお使いの場合にはパラメータで指定されたByte型のデータを送信します。それ以外のコントロール・コンテナではVARIANT型の1バイト配列データの扱いになりますので各コントロール・コンテナのマニュアル等をご参照ください。Win32 APIではVARIANT型とSafeArray系のAPIの説明をご参照ください。送信するデータのサイズは宣言したByte型の配列のサイズになります。

## Visual Basic サンプル

Dim b(0 To 5) As Byte

Dim rc As Integer

b(0) = 0                    ' Packet size first byte  
b(1) = 5                    ' Packet size second byte  
b(2) = 2                    ' STX  
b(3) = AscB("T")          ' Text body  
b(4) = 3                    ' ETX

rc = VBMMComm1.SendBinaryBytes(b)

If rc <> 0 Then

    MsgBox "SendBinaryBytesエラー " & CStr(rc)

    Stop

End If

## **SendBreak**

---

### **書式**

SendBreak() As Integer

### **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サンプル

Dim rc As Integer

rc = VBManComm1.SendBreak

## **SendByte**

---

### **書式**

SendByte( NumOfChars As Integer ) As Integer

### **パラメータ**

送信データのサイズ。送信バッファ長より大きい値は指定できません。

### **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**

このメソッドがよびだされた時の送信バッファの内容を指定されたバイト数だけ送信します。Visual Basic 4.0J 32Bit版以降はString型がUnicodeからShift JISに変換される為、0x81からのシフトJISコードの先頭パイ

ト・コードがSendBinaryメソッドで送信できないことに対応してバージョン2.00aから追加されたメソッドです。送信するバイナリ・データを送信バッファに設定するには、SendBufferプロパティを使ってください。

### **注意**

このメソッドは旧バージョンとのコンパチビリティのために残されています。新規にコードを開発する場合はSendBinaryBytesメソッドをお使いください。

### **Visual Basicサンプル**

```
Dim l as integer, rc as integer
```

```
‘ SendBufferは0オフセットでデータ型はIntegerです。
```

```
For l = 0 To 255
```

```
    VBManComm1.SendBuffer(l) = l
```

```
Next l
```

```
‘ パラメータは送信するバイト数です。
```

```
VBManComm1.SendByte(256)
```

```
If rc <> 0 Then
```

```
    MsgBox “send byte error “ & CStr(rc)
```

```
End If
```

## **SendChar**

---

### **書式**

```
SendChar( aChar As String ) As Integer
```

### **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サンプル**

```
Dim rc As Integer
```

```
rc = VBManComm1.SendChar(chr$(13))
```

```
‘ 改行コードを送る
```

## **SendFile**

---

### **書式**

```
SendFile( FileName As String ) As Integer
```

### **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

マルチスレッド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VcProtocolプロパティにファイル転送プロトコルを設定します。このメソッドはファイル転送のスレッドを起動してすぐに呼び出し側のユーザー・プログラムに制御を戻します。スレッドの起動に失敗した場合はこのメソッドの戻り値を設定してエラーを返します。ファイル転送中の進捗状況はVcStatus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。

## 参照

ReceiveFileメソッド、OnCommTransferEndイベント

## SendString

---

### 書式

SendString( aStr As String ) As Integer

### パラメータ

送信する文字列。

### 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

文字列を送信します。ライン上には7bitキャラクターはそのままASCIIコードで送信されます。漢字が含まれる場合はShift-JISコードにてラインに送信されます。Unicode文字列をそのままラインに送信したい場合はSendUnicodeStringメソッドをご利用ください。バイナリ・データにつきましてはSendBinaryBytesメソッドをご利用ください。

以下は、モデムを初期化する文字列を送信する例です。

### Visual Basicサンプル

```
Dim s$, rc As Integer
```

```
s$ = "ATZ" & Chr$(13) & Chr$(10)  
rc = VbManComm1.SendString(s$)
```

## SendUnicodeString

---

### 書式

SendUnicodeString( aStr As String ) As Integer

### パラメータ

送信する文字列。

### 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

文字列を送信します。ライン上にもUnicode形式のままデータを送信します。このメソッドで送ったデータはRecvUnicodeStringメソッドで受信することが可能です。7bit文字でもライン上には2バイト出力されることにご注意ください。Visual Basic等32bit言語では文字列内部コードはUnicodeを採用していますが、一般的に周辺機器に漢字データを送る場合はSHIFT-JISコードになりますのでSendString等のメソッドをお使いください。

以下は文字列を送信する例です。

### Visual Basicサンプル

```
Dim s$, rc As Integer
```

```
s$ = "ユニコード"
```

```
rc = VBManComm1.SendString(s$)
```

## SetDTR

---

### 書式

```
SetDTR() As Integer
```

### 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 Integer
```

### 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。

## カスタム・イベント

---

この章ではVBMan Control 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(パラメータは文字列)
CommRecvVar	データ受信イベント(パラメータはVariant)
CommSend	データ送信イベント
CommTransferEnd	ファイル転送終了イベント

## CommError

---

VBMan Control for RS-232C通信エラーの通知にイベントを発生させます。エラー・イベントは以下の形式です。

CommError( MajorErrorCode As Integer, MinorErrorCode As Long )

### MajorErrorCode

主エラーコード。エラーの主たる原因を示します。

### MinorErrorCode

詳細情報エラー・コード。送受信時のエラーについて詳細情報が通知されます。

## CommNotify

---

VBMan Control 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

以下はコード・サンプルです。

```
Sub Comm1_CommNotify( dwEventMask As Long )
Dim Stat As Integer
Const EV_CTS = &H8
Const EV_DSR = &H10
Const EV_RLSD = &H20
```

```
Const EV_RING = &H100
```

```
Select Case dwEventMask  
    Case dwEventMask And EV_CTS  
        Debug.Print "CTS is On"  
    Case dwEventMask And EV_DSR  
        Debug.Print "DSR in on"  
    Case dwEventMask And EV_RLSD  
        Debug.Print "RLSD is On"  
    Case dwEventMask And EV_RING  
        Debug.Print "Ringing"  
End Select  
  
End Sub
```

## **CommRecv**

---

プロパティVcNotifyRecvChar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。以下はサンプル・コードです。Visual C++の場合は通常のANSI文字列ではなくBSTR型の文字列でデータがイベントに渡されることに注意してください。

### **Visual Basicサンプル**

```
Sub Comm1_CommRecv( RecvStr As String )  
    Text1.Text = Text1.Text & RecvStr  
End Sub
```

### **Visual C++サンプル**

```
void CTestDlg::OnCommRecv(BSTR FAR* strRecv)  
{  
    CString cstr(*strRecv);  
    TRACE("受信文字列 = %s\n",cstr.GetBuffer(0));  
}
```

## **CommRecvVar**

---

プロパティVcNotifyRecvCharに0以上の値を設定されており、プロパティVcRecvEventTypeにAsString(=値0)が設定されている場合に発生するイベントです。通信中にVcNotifyRecvChars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CommRecvVar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。イベントのパラメータはVariant型になっています。この形式のイベントはVisual Basic.NET/Visual C#等ではサポートされますが、Visual Basic 6.0等サポートされない言語もありますのでご注意ください。

### **Visual Basic.NETサンプル**

```

Private Sub AxVBMCom2_CommRecvVar(ByVal sender As Object,
    ByVal e As AxVBMCOM32Lib._VBMEvents_CommRecvVarEvent) Handles
    AxVBMCom2.CommRecvVar
System.Diagnostics.Debug.WriteLine(e.vData)
End Sub

```

## Visual C# サンプル

```

private void axVBMCom2_CommRecvVar(object sender,
AxVBMCOM32Lib._VBMEvents_CommRecvVarEvent e)
{
    if(axVBMCom2.VcRecvEventType == VBMCOM32Lib.enumRecvEventType.AsVariantString)
    {
        System.Diagnostics.Debug.WriteLine("recv var " + Convert.ToString(e.vData.ToString()));
    } else {
        System.Diagnostics.Debug.WriteLine("recv varbinary:" + e.vData.GetType().ToString() + ";" +
            e.vData.ToString());
    }
}

```

## CommSend

---

プロパティVcNotifySendComplete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。

## CommTransferEnd

---

ファイル転送メソッドRecvFile,SendFileはスレッドをスタートさせるとすぐにユーザー・プログラムに制御を戻します。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。

```

Sub Comm1_CommTransferEnd( sStat As Integer )
If sStat = 0 Then
    MsgBox “ファイル転送が正常に終了しました。”
Else
    MsgBox “ファイル転送が失敗しました。” & Cstr(sStat)
End If
End Sub

```

# 通信ユーティリティ

---

VBMan Control for RS-232Cにはvbmcu32.ocxという通信サポート用のActiveX Controlが添付されます。これらのカスタム・コントロールもATLを使って作成されております。この章ではこのカスタム・コントロールによるCRC計算方法を説明します。

## カスタム・メソッド

### Crc16

---

#### 書式

Crc16( Buf() As Byte,  
Size As Integer,  
Result As Integer) As Integer

#### パラメーター

Buf	Byte型の配列で指定するCRC16を計算する領域
Size	上記配列での有効なサイズ。省略した場合は配列サイズになります。
Result	CRC16計算結果

#### 戻り値

0	正常終了
-1	パラメータ・エラー。配列のサイズ、ディメンション等の指定間違い

#### 概要

16bit CRCを計算します。

#### サンプル・コード

以下のサンプルではText1というテキスト・ボックスから入力されたファイルのcrc16を計算してResultというラベルに表示します。

```
Private Sub Command1_Click()  
On Error GoTo err_handler  
Dim buf(0 To 1000) As Byte  
Dim ch  
Dim idx As Integer, crc16 As Integer, rc As Integer, crc32 As Long
```

```

idx = 0
Open Text1.Text For Input As #1
Do While Not EOF(1)      ' ファイルの終端までループを繰り返します。
  ch = Input(1, #1)      ' 1 文字のデータを読み込みます。
  buf(idx) = AscB(ch)    '
  idx = idx + 1          '
Loop
Close #1

Debug.Print "file size = ", idx

If optCrc16.Value = True Then
  rc = util.crc16(buf, idx, crc16)
  Result.Caption = Hex$(crc16)
Else
  rc = util.crc32(buf, idx, crc32)
  Result.Caption = Hex$(crc32)
End If

Exit Sub

'-----
'
err_handler:
MsgBox "Err = " & CStr(Err)
On Error Resume Next
Close #1

End Sub

```

## Crc32

---

### 書式

```

Crc32( Buf() As Byte,
      Size As Integer,
      Result As Long) As Integer

```

### パラメーター

Buf	Byte型の配列で指定するCRC32を計算する領域
Size	上記配列での有効なサイズ。省略した場合は配列サイズになります。
Result	CRC32計算結果

### 戻り値

0	正常終了
-1	パラメータ・エラー。配列のサイズ、ディメンション等の指定間違い

### 概要

32bit CRCを計算します。

## 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・プロセス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	RecvBinaryBytesメソッド等の呼び出し時点で通信バッファにはデータが存在していませんでした。
-------------	-----	--

## 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。

## 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

## トラブルシューティング

---

ここではVBMan Control 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ではシリアルポートが無い機種も多いのでご検討ください。

### 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わか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VcProgress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。

### Accessでプロパティが保存されない

---

Access97のフォームで当コントロールをご利用になる場合はプロパティ値としてデフォルト値以外を設定する場合は、Form\_Loadでコードで各プロパティを設定してからOpenCommメソッドを呼び出してください。Samples¥AccessフォルダーにBcode.mdbにサンプルがありますのでご参照ください。

### 拡張ボードでの動作

---

VBMan Control for RS-232Cは各種拡張ボードでの動作確認は基本的にしておりません。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ようなことはありません。通信ポートとしてCOM1からCOM10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を経由して正常に動作するものでしたらVBMan Control for RS-232Cでも問題なくご利用いただけます。

### Visual Basicセットアップ・ウィザードでのモジュール配布

---

VBMan Control for RS-232Cを使ったアプリケーションをVisual Basicのセットアップ・ウィザードを使って、インストール・プログラムを作成する場合、以下の行をWindowsのディレクトリにあるswdepend.iniファイルに追加することで、配布プログラムにOCXを自動的に追加することができます。OCXのファイル名を[]の中に記述して以下の5行を追加することになります。セットアップ・ウィザード(7/7)で「モジュールの追加」でOCXを指定すれば、以下の設定は不要です。実行時のモジュールとしてvbmsm32.dllを同ウィザードにて指定することも必須です。

```
[VBMCOM32.OCX]
Register=$(DLLSelfRegister)
Dest=$(WinSysPath)
Uses1=OCX Runtime Support
Uses2=
```

## サポートWEBサイトから新バージョンをダウンロードしてインストールしたら動作が変わった

---

サポートWEBサイト ([www.techknowledge.co.jp](http://www.techknowledge.co.jp)) では、VBMan Control for RS-232Cの最新バージョンがダウンロード可能になっています。最新版をダウンロードして利用する場合には、\Windows\system等にあるvbmcom32.ocxファイルを削除して、regsvr32.exeでvbmcom32.ocxを再度登録してください。拡張子.ocxのファイルはOCXのプロパティ情報などをキャッシュしているファイルです。プロパティを追加した新バージョンをインストールするとタイプ情報の不一致から動作が不安定になることがあります。また、古いバージョンのOCXで作成されたEXEファイルは再度コンパイルが必要になる場合もあります。再度コンパイルするために開発モードで実行したVisual Basicではフォームを一旦ひらいてvbmcom32.ocxのプロパティをご確認ください。正常な値が設定されていない場合はコントロールをはり直して、正確なプロパティ値を設定してください。

## Accessでのバイナリ・データ通信

---

AccessではByte型の配列操作に不具合がありバイナリデータをSendBinaryBytes, RecvBinaryBytesメソッドでは送受信することができません。回避するための方法としてRecvBuffer/SendBufferプロパティ、SendByteメソッドをお使いください。

以下はコードサンプルです。

### 1.RecvBufferプロパティについて

RecvBufferを参照することによって受信データのバイナリ値をInteger型（2バイト整数）の変数に取得することが可能です。プロパティは一次元配列でインデックスは0からとなります。

```
Dim tmp$, v as Integer, i As Integer
```

```
tmp$ = Com.RecvString(0)
If tmp$ <> "" Then          ' 受信データ有?
  For i = 0 To Len(tmp$) - 1
    v = Com.RecvBuffer(i)
    Debug.Print Cstr(v)
  Next i
End If
```

### 2.SendBufferプロパティとSendByteメソッド

SendBufferプロパティにセットした送信データはSendByteメソッドで送信することができます。

```
Dim rc As Integer, i As Integer
For i = 0 to 255
  Com.SendBuffer(i) = i
Next i
```

```
rc = Com.SendByte(256)
If rc <> 0 Then
    MsgBox "SendByte エラー " & CStr(rc)
End If
```

Access等のVBAでは&H20以下のバイナリ・データはSendString/RecvStringメソッドで送受信可能です。&H80以上のデータ通信が無い場合はSendString/RecvStringメソッドを使ったほうが、コードが簡略化される場合があります。尚、RecvStringメソッドで受け取った文字列のバイナリ値を知る場合にはAscB関数を使ってください。

### 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 Control for RS-232C調査依頼

以下の調査依頼フォームの項目を記入してインターネットで[support@techknowledge.co.jp](mailto:support@techknowledge.co.jp)宛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さしあげます。もうしわけありませんが**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	
お問い合わせ内容、問題記述など、具体的に再現可能なようにご記入ください。	
添付資料	

VBMan Control for RS-232C version 4.5

マニュアル第6版

2025年2月10日 第1刷発行

版權・著作 株式会社テクナレッジ

Printed In Japan