VBMan
Components for RS-232C/CF
Programming Manual

Version 5.10

TechKnowledge

VBMan Components for RS-232C/CF

はじめに

製品の概要

VBMan Components for RS-232C/CF version 5.01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Pocket PC/Windows CE.NETなどMicrosoft .NET Compact Framework が動作する環境向けのシリアル通信をサポートする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
  当製品は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、2004年の.NET framework専用コンポーネントと、マイクロソフトのソフトウェア部品の仕様に沿って進化してきました。このたび.NET Compact Framework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シリアル用ソフトウェア・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 Compact Framework用コンポーネント
  スマートデバイス用アプリケーションの開発においてシリアル通信部分は比較的開発に時間がとられる部分です。当コンポーネントをご利用になることで通信部分の開発時間の短縮およびコストの削減が可能です。
- デザイン時専用コンポーネントのご提供
  メモリの少ないスマートデバイスにおいてはVS.NET 2003で開発する場合に用いるデザイン時専用コンポーネントと実行専用コンポーネントを分けることでスマートデバイスでの実行時メモリのフットプリントを最小化することが出来ます。
- オーバーロードにより.NETデータ型をサポートするメソッド
  複数の.NETのデータ型オーバーロードしたSend/Receiveメソッドが提供され

ます。基本的なデータ型であればバイト配列に変換するような手間は必要ありません。

- 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。
- エラーの通知方法が選択可能
  エラーをエラーイベントで通知するか、例外として通知するかをプロパティにて変更することが出来ます。お客様のアプリケーションのエラーハンドリングストラテジーに沿ったプログラミングが可能となっています。
- 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、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。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、ユーザーサポート

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

サーポートメール: support@systemlab.co.jp

Web: www.systemlab.co.jp

開発元

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。

インストール

VBMan Components for RS-232C/CF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/CF は Microsoft .NET Compact Framework 1.1 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 PDA 環境で動作します。

- Microsoft Windows CE.NET
- PocketPC 2002/2003

VBMan Components for RS-232C/CF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#

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 2003

セットアップの起動

VBMan Components for RS-232C/CF 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¥CommCFLib.dll	RS-232C 通信コンポーネント	可
<instdir>¥bin¥CommCfLib.Design.Dll	デザインコンポーネント	可
<instdir>¥man¥CommLib510cf.html	リリースノート	不可
<instdir>¥man¥CommLib510cf.pdf	PDF マニュアル	不可
<instdir>¥Samples¥ce¥VB.NET¥*.*	Windows CE.NET 用 VB.NET 用サンプルプログラム	不可
<instdir>¥Samples¥ce¥CS¥*.*	Windows CE.NET 用 C#用サンプルプログラム	不可
<instdir>¥Samples¥ppc¥VB.NET¥*.*	Windows CE.NET 用 VB.NET 用サンプルプログラム	不可
<instdir>¥Samples¥ppc¥CS¥*.*	Windows CE.NET 用 C#用サンプルプログラム	不可

プロジェクト開始

IDE へのデザイン時コンポーネントの登録方法

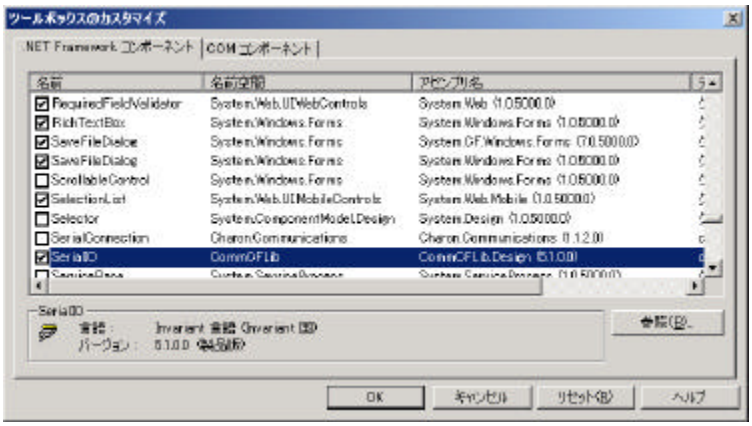
最初に Visual Studio 2003 の IDE のツールボックスにデザイン時コンポーネントを登録する方法について説明します。ツールボックスへのデザイン時コンポーネントの登録することにより IDE にコンポーネント生成コードを自動生成させることが出来ます。以下ツールボックスへデザイン時コンポーネントを登録する手順です。

① ツールボックスに登録したいタブを選択してマウスの右クリックメニューから「アイテムの追加と削除」を選択します。

② 「.NET Framework」のタブが表示されますので「参照」ボタンをクリックします。

③ ファイル指定ダイアログが表示されますのでデフォルトのインストール時であれば「c:¥Program Files¥TechKnowledge¥VBMan Components for RS-232C¥bin¥CommCfLib.Design.Dll」を選択して「OK」ボタンをクリックします。

④ 以下の CommCFLib が選択された状態になりますので「OK」ボタンを押します。



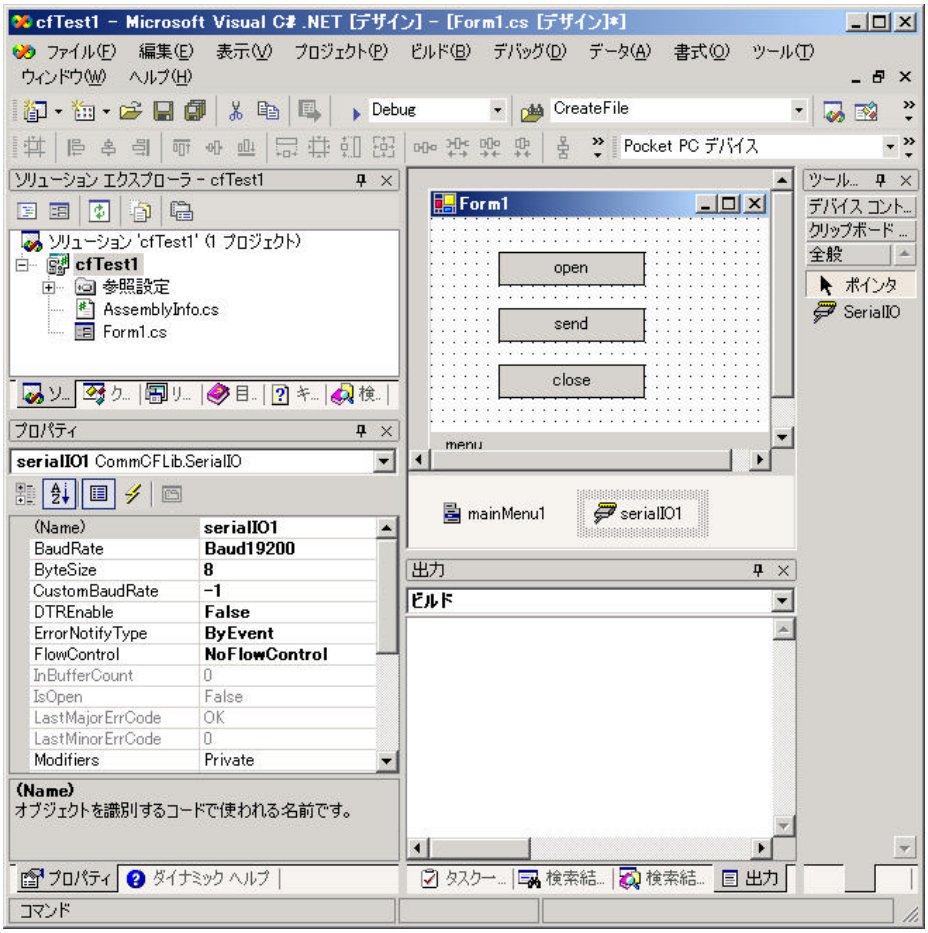
上記でデザイン時コンポーネントのツールボックスへの登録が出来ました。次は言語毎にプロジェクトからの参照方法をご説明します。

Visual Basic.NET での参照設定方法

デザインコンポーネントと実行時コンポーネントはデザイン属性により関連付けられていますが、実行ファイルを.CAB に纏める場合などに IDE は実行時コンポーネントの存在場所を知る必要があります。IDE には参照設定で実行時コンポーネントを指定することになります。以下は Microsoft Visual Basic.NET で VBMan Components for RS-232C/CF の参照設定方法です。

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 Visual Basic.NET を選択します。
④ プロジェクトタイプとして「スマートデバイスアプリケーション」を選択します。
⑤ 「スマートデバイスアプリケーションウィザード」の指定にしたがってターゲットデバイスとプロジェクトの種類を選択します。
⑥ 展開されたプロジェクトの「ソリューションエクスプローラ」から「参照設定」を右クリックします。
⑦ 「参照の追加」を選択します。
⑧ 「.NET」タブの状態で「参照」ボタンをクリックします。
⑨ デフォルトインストールであれば、「c:¥Program Files¥TechKnowledge¥VBMan Components for RS-232C/bin/CommCFLib.dll」を選択して「OK」ボタンをクリックします。
⑩ フォームを使うプロジェクトタイプであればツールボックスから「SerialIO」をドラッグすることで IDE によりシリアル通信コントロールの初期化コードが展開されます。

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



Visual C#での利用方法

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。

コンポーネント参照についての注意事項

Windows Form を使うプロジェクトにてツールボックスにデザイン時コンポーネントを登録していても、プロジェクトの参照に実行時コンポーネントを登録していない場

合は、次回プロジェクトを開いた場合に IDE がコンポーネントの特定が出来なくなり、フォームデザイナ下部からSeriallO クラスのアイコンが消えることがありますのでご注意ください。このような状況になった場合は再度参照設定をして、フォームにコントロールの貼り付けをして、IDE が生成するコードを手直しすることでプロジェクトを元に戻すことが出来ます。

Namespace/Class 名について

当コンポーネントの Namespace は CommCFLib となります。また、Class 名は SeriallO 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。

```
CommCFLib.SerialIO rs232c = new CommCFLib.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。

コンパチビリティ

ここではWin32 環境フル・フレームワークで動作するVBMan Components for RS-232C ver5.x との機能的な相違点について記載します。基本的には.NET Compact Framework 版は下位互換になりますがメモリの少ない Pocket PC/Windows CE 環境を考慮して主要な機能を移植するという考え方は.NET Framework と.NET Compact Framework のインプリメント方針を踏襲しています。以下.NET Compact Framework 版のことをCF版と略して記載します。

① CF版ではネームスペースがCommCFLib となります。
② CF 版では実行時コンポーネントのファイル名が CommCFLib.DLL となります。
③ CF 版ではメソッドからの戻り値は従来の short 値から ErrorCodes という short 型の列挙型になります。
④ CF版ではファイル転送系メソッドがサポートされません。
⑤ CF版ではデバッグトレース機能がサポートされません。
⑥ CF版では Stream プロパティはサポートされません。

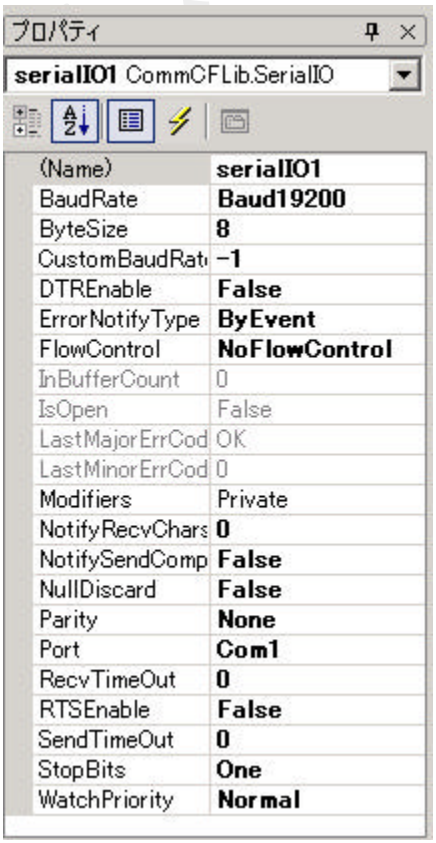
カスタム・プロパティ

VBMan Components for RS-232C/CF 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
BaudRate	通信速度
ByteSize	通信データサイズ
CustomBaudRate	任意の通信速度設定
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します。
Parity	通信パリティ
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

以下はデザインコンポーネントを使ってIDE にて開発する場合のプロパティ・ウィンドウ表示例です。



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。上記の通信速度の最高値についてはお使いになるハードウェアやマップされる通信方法により通信可能でない場合もあります。

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。

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 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 = CommCFLib.PortValues.Com1
  .FlowControl = CommCFLib.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
...
...

‘ 1byte 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 CommCFLib.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。

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	パリティ設定
NoParity	なし
OddParity	奇数パリティ
EvenParity	偶数パリティ

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 = CommCFLib.PortValues.Com1
```

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 CommCFLib.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。

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。

---

[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。

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 CommCFLib.ErrorCodes
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
```

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。

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

カスタム・メソッド

ここではVBMan Components for RS-232C/CF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
Send	データを送信
SendBreak	ブレーク信号送信
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 ErrorCodes

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 ErrorCodes
rc = SerialIO1.ClearBreak
```

ClearDTR

書式
ClearDTR() As ErrorCodes

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 ErrorCodes
rc = SerialIO1.ClearDTR
```

ClearRTS

書式
ClearRTS() As ErrorCodes

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 ErrorCodes
rc = SerialIO1.ClearDTR
```

CloseComm

書式
CloseComm() As ErrorCodes

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 ErrorCodes
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 UInt32

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 ErrorCodes

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 ErrorCodes

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 CommCFLib.CommLineEventArgs) _
  Handles SerialIO1.OnCommLine
Dim rc As ErrorCodes
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 ErrorCodes

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 CommCFLib.CommLineEventArgs) _
  Handles SerialIO1.OnCommLine
Dim rc As ErrorCodes
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 ErrorCodes

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 CommCFLib.CommLineEventArgs) _
  Handles SerialIO1.OnCommLine
Dim rc As ErrorCodes
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 ErrorCodes

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わった時点

[2] receive-line-single-detect

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 CommCFLib.CommLineEventArgs) _
  Handles SerialIO1.OnCommLine
Dim rc As ErrorCodes
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 ErrorCodes

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(ByRef *Buffer* As String) As ErrorCodes
Receive(ByRef *Buffer* As String, Size As Integer) As ErrorCodes
Receive(ByRef *Buffer* As Int16) As ErrorCodes
Receive(ByRef *Buffer* As Int32) As ErrorCodes
Receive(ByRef *Buffer* As Int64) As ErrorCodes
Receive(ByRef *Buffer* As UInt16) As ErrorCodes
Receive(ByRef *Buffer* As UInt32) As ErrorCodes
Receive(ByRef *Buffer* As UInt64) As ErrorCodes
Receive(ByRef *Buffer* As Single) As ErrorCodes
Receive(ByRef *Buffer* As Double) As ErrorCodes
Receive(*Buffer()* As Byte) As ErrorCodes
Receive(*Buffer()* As Byte, *Size* As Integer) As ErrorCodes
Receive(*Buffer()* As Byte, *Offset* As Integer, *Size* As Integer) As ErrorCodes

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);
```

Send

書式

Send(*data* As String) As ErrorCodes
Send(*data* As Int16) As ErrorCodes
Send(*data* As Int32) As ErrorCodes
Send(*data* As Int64) As ErrorCodes
Send(*data* As UInt16) As ErrorCodes
Send(*data* As UInt32) As ErrorCodes
Send(*data* As UInt64) As ErrorCodes
Send(*data* As Single) As ErrorCodes
Send(*data* As Double) As ErrorCodes
Send(*data* As Byte) As ErrorCodes
Send(*data*()As Byte) As ErrorCodes
Send(*data*() As Byte, *offset* as Integer, *length* As Integer) As ErrorCodes

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 ErrorCodes

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 ErrorCodes
rc = SerialIO1.SendBreak
```

SetDTR

書式
SetDTR() As ErrorCodes

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 ErrorCodes

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 ErrorCodes

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 ErrorCodes
Dim b As Byte

b = 13
rc = SerialI01.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/CF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

OnCommError

書式

```
OnCommError(ByVal sender As Object, _
            ByVal args As CommCFLib.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/CF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/CF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 CommCFLib.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
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 SerialIO1_OnCommSend(ByVal sender As Object, _
    ByVal args As CommCFLib.CommSendEventArgs) _
    Handles SerialIO1.OnCommSend
System.Diagnostincs.Debug.WriteLine(“send queue is empty”)
End Sub
```

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。CommCFLibネームスペースにはErrorCodes列挙で定義されています。

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_LO	117	RecvStringメソッドへのパラメータが大きすぎま

NG		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。

ERR_CLOSE	136	通信ポートクローズ時のエラーです。
ERR_PORT_IS_UNAVAILABLE	137	通信可能なポートが見つかりませんでした。
ERR_CLEARE_COMM_ERROR	138	ClearComm Error API実行の戻り値がエラーとなっています。詳細はMinorErrorCodeにあります。

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。VS.NET 2003ではツールから「エラールックアップ」で詳細が検索できます。Win32 SDK等の資料はMicrosoft Developer Network (MSDN)より入手できます。

Appendix-B　トラブルシューティング

ここでは VBMan Components for RS-232C/CF を使って PocketPC または Windows CE.NET 向け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

エミュレータの COM ポート設定方法について

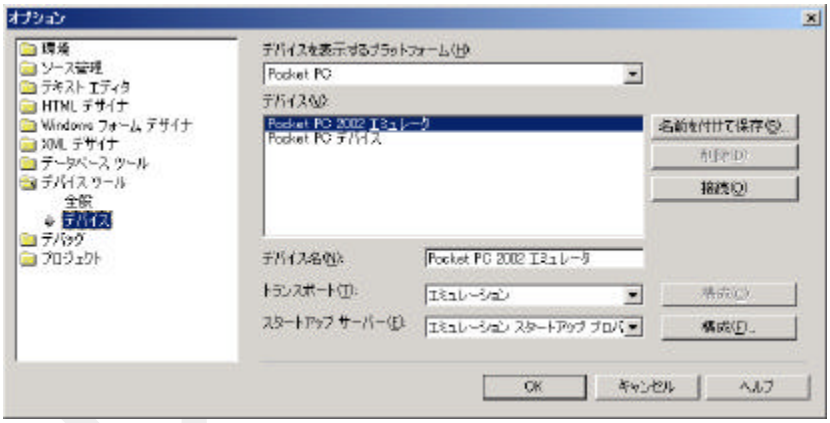
Windows CE 開発ツールに添付されているエミュレータでは COM ポートに関して不具合があり、以下の Microsoft Knowledge Base にて公開されています。

http://support.microsoft.com/kb/230756/ja

弊社で VS.NET 2003 に添付されるPocket/Windows CE 用のエミュレータで試したところ、COM ポートをひとつアサインするだけであれば問題なく動作することは確認できました。

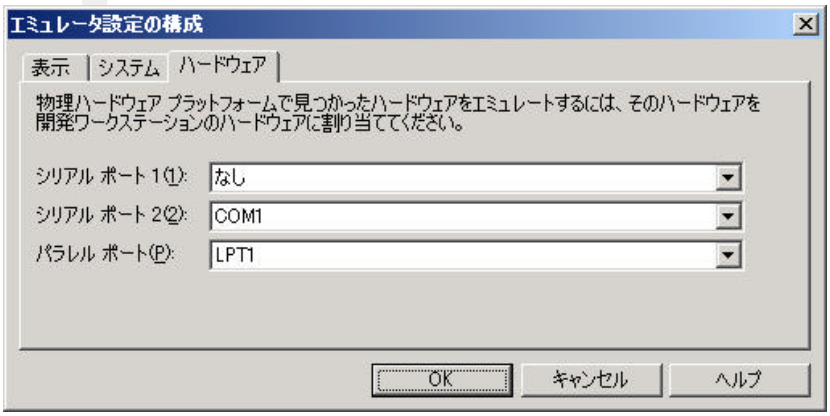
以下は VS.NET 2003 での COM ポート指定方法です。

① PC 側の COM ポートの空いていることを確認します。他の通信に使われていないことを確認してください。
② 実行中のエミュレータがあれば停止します。このとき、状態の保存は選択しないで、完全にシャットダウンします。
③ VS.NET 2003 の「ツール」メニューから「オプション」を選択します。
④ オプションダイアログの右のツリーから「デバイスツール」を選択します。
⑤ さらに「デバイス」のサブ・ツリーを選択します。



⑥　ご利用のプラットフォームとデバイスを選択して「スタートアップサーバー」の右の「構成」を選択します。

⑦　「エミュレータ設定の構成」ダイアログで「ハードウェア」タブを選択します。



⑧　シリアルポート1（1）:に指定した場合は動作しないこともあるようです。弊社ではシリアルポート2（2）:にCOMポートを指定したところ通信が可能でした。

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/CF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CommCFLib.DLLが自動的にCABファイルに設定されます。他にシリアル通信コンポーネントが動作するために必要なファイルはありません。

ポート数の上限について

VBMan Components for RS-232C/CF では Port は COM1～COM20 まで定義されていますがこの ポート全てを同時に動作させることは Pocket PC/Windows CE のハードウェアでは不可能と思われます。Win32 版とのコンパチビリティから定義可能なポート上限を設定しています。RS-232C ポート以外にも USB や BlueTooth デバイスが COM ポートにマップされることを考慮して多くのポートを定義可能としています。

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/CF 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/CF
version 5.10

マニュアル第1版
2005年5月31日　第1刷発行

Printed In Japan