取扱注意

# 製品仕様書

製品名　制御用Ｒ／Ｗ向けクラスライブラリ

形式　V31Control.dll

仕様書名　リファレンスマニュアル

仕様書番号　Ｍ１０２５－７３８０Ａ０６

総構成

| | |
|---|---|
| ハードウェア仕様書 | Ｍ１０５４－７６９５Ａ |
| ソフトウェア仕様書 | Ｍ１０５４－７６９６Ａ |
| リファレンスマニュアル | Ｍ１０２５－７３８０Ａ |

三和ニューテック株式会社
開 発 部

| ２０１３年６月２０日 | | |
|---|---|---|
| 承　認 | 審　査 | 作　成 |
| SNT 13.6.28 長倉 | SNT 13.6.28 川越 | SNT 13.6.20 近藤 |
| 長倉 | 川越 | 近藤 |

変更経歴書

| 版 | 記号 | 年月日 | 項番 | 変更内容 | 変更理由 | 担当 |
|---|---|---|---|---|---|---|
| 00 | － | 2006/10/20 | － | 新規発行 | | 川崎 |
| 01 | ① | 2009/05/06 | － | USB 向けクラスライブラリのメソッド追加 | RWControl の機能を拡張し、USB 対応化したため。 | 川崎 |
| | | 2009/07/16 | 1 | 対応する OS の記載見直し | 記載内容の見直し | 川崎 |
| | | | | 記載されているコード、サンプルソースの言語を C#と明記した。 | | |
| | | | － | 社内で規定されている技術用語に表記を合わせた。 | | |
| | | | 3 | USB ドライバー②のインストール方法について追記した | | |
| 02 | ② | 2009/07/21 | － | ドライバ表記をドライバーに変更した | 記載内容見直し | 川崎 |
| | | | | バージョン表記が古かったので変更した。 | | |
| | | | | 誤字・脱字の修正 | | |
| | | 2009/11/18 | － | 記載漏れの修正 | 記載内容見直し | 川崎 |
| 03 | ③ | 2009/11/30 | － | 使用しないコマンドの削除 | コマンド仕様の見直し | 川崎 |
| 04 | ④ | 2010/11/04 | － | メッセージループ表記を Windows メッセージループに変更 | 記載内容見直し | 猪俣 |
| | | | | V31Control.LockEvents プロパティが真の場合の注意事項に修正 | | |
| | | | | バージョン表記見直し | | |
| 05 | ⑤ | 2013/04/16 | １－２. | 動作確認済みのＯＳと環境の追加・変更 | 記載内容見直し | 川越 |
| | | | １－５. | 本クラスライブラリの名前空間とアセンブリ情報のバージョン表記見直し | 記載内容見直し | |
| | | | ４. | 保証するＯＳの表記見直しとインストール手順の追加 | 記載内容見直し | |
| | | | － | 章の構成変更 | 構成の見直し | |
| | | | ３－３. | VbBaudRate、VbParity、VbStopBits プロパティの追加 | VisualBasic 言語による開発対応のため。 | |
| | | | ５. | アンインストール方法の追加 | 記載内容見直し | |
| 06 | ⑥ | 2013/06/20 | 1-2 | ｸﾗｽﾗｲﾌﾞﾗﾘ OS 対応表を追記 | 記載内容見直し | 近藤 |
| | | | | .NET Framework 3.5 動作環境参考表を削除し、参照先説明追記 | 記載内容見直し | |
| | | | 1-5 | 本ｸﾗｽﾗｲﾌﾞﾗﾘの名前空間とｱｾﾝﾌﾞﾘ情報のﾊﾞｰｼﾞｮﾝ表記見直し | 記載内容見直し | |
| | | | 3-4 | Windows7 64bit 版での USB 通信使用要件追記 | 要件明確化のため | |
| | | | 3-15-13 | ﾘｰﾄﾞﾓｰﾄﾞ指定ﾊﾟﾗﾒｰﾀに、「34h 逆可変長(1～69 ﾊﾞｲﾄ)ﾘｰﾄﾞ」を追記 | ﾊﾟﾗﾒｰﾀﾃﾞｰﾀ追加のため | |
| | | | 3-15-15 | ﾗｲﾄﾓｰﾄﾞ指定ﾊﾟﾗﾒｰﾀに、「34h 逆可変長(1～69 ﾊﾞｲﾄ)ﾗｲﾄ」を追記 | ﾊﾟﾗﾒｰﾀﾃﾞｰﾀ追加のため | |
| | | | | 「ﾘｰﾄﾞ」を「ﾗｲﾄ」に訂正 | 誤記訂正 | |
| | | | | | | |

| 版 | 記号 | 年月日 | 項番 | 変更内容 | 変更理由 | 担当 |
|---|---|---|---|---|---|---|
| 06 | ⑥ | 2013/06/20 | 4-3 | 見出し項目に”(32bit 版)”を追記 | 64bit 版との区別明確化のため | 近藤 |
| | | | 4-4 | Windows7 64bit 版でのｲﾝｽﾄｰﾙ手順追記 | ｲﾝｽﾄｰﾙ手順追加のため | |
| | | | 5-1 | 見出し項目に”(32bit 版)”を追記 | 64bit 版との区別明確化のため | |
| | | | 5-2 | Windows7 64bit 版でのｱﾝｲﾝｽﾄｰﾙ手順追記 | ｱﾝｲﾝｽﾄｰﾙ手順追加のため | |
| | | | | | | |

# 目　次

～はじめに～

本クラスライブラリは、三和ニューテック株式会社製の磁気カードターミナル「ＡＢＳ－Ｖ３１」を使用したシステムのアプリケーション開発を容易にするためのクラスライブラリです。
本クラスライブラリを利用することで、「ＡＢＳ－Ｖ３１」とのシリアルやUSBといった通信プロトコルの詳細を意識することなく、簡単に「ＡＢＳ－Ｖ３１」が提供する通信機能の全てを操作することができます。

**注意）本クラスライブラリを用いてアプリケーションを作成する際には、本マニュアルだけではなく、必ず「ＡＢＳ－Ｖ３１　ソフトウェア仕様書」を参照するようにして下さい。将来において、レスポンス、引数など本マニュアルの記載とソフトウェア仕様書が異なる場合、ソフトウェア仕様書の記載に沿ってアプリケーションを作成して下さい。**

# 1. クラスライブラリの概要

本クラスライブラリを用いてアプリケーションを作成する際に考慮すべき事柄について列挙します。

## 1-1. .NET Framework 3.5 に対応

本クラスライブラリは、Microsoft社の.NET Framework 3.5に対応しており、その実体は.NET Frameworkのアセンブリになっています。

厳密名を持っていますのでGAC(Global Assembly Cache)に配置し、複数のアプリケーションから共有することが可能です。また、.NET Frameworkが提供するバージョン管理、セキュリティチェックも有効になります。

当然ですが、.NET Frameworkに対応したIL(Intermediate Language)コンパイラを使用することにより、あらゆる開発言語から本クラスライブラリが提供する全ての機能を利用できます。

## 1-2. 動作確認済みの OS と環境

本クラスライブラリは、以下のOSで動作することを確認しております。

⑥

| バージョン | 対応OS |
|---|---|
| 2.3.0.0 | Windows XP Professional SP3<br>Windows 7　32bit版 |
| 3.0.0.0 | Windows 7　64bit版 |

※本クラスライブラリは、バージョンにより使用するOSが異なりますので、注意してください。
上記のバージョンとOSの組み合わせで動作確認を行っております。⑥

.NET Framework が導入可能な、Windows Server 2003、Windows Server 2008、Windows Vista①など他のWindows製品でも動作可能と思われますが、上記以外の環境で使用する場合には、アプリケーション開発時に十分にテストを行うようお願いいたします。ただし、USBを用いて開発を行う場合、弊社製のUSBデバイスドライバーが動作する環境に依存します。

.NET Framework 3.5動作環境については、Microsoft社ダウンロードセンターにある、.NET Framework 3.5 サイトを参照してください。⑥

## 1-3. 開発環境

開発環境は、.NET Framework 3.5の開発要件に加えて、作成するアプリケーションの規模や開発言語によっても異なります。開発着手前に十分にご検討下さい。

また、本リファレンス、添付したサンプルソースは、C#言語で記述いたします。他のVB.NETなどの.NET言語であれば、本クラスライブラリと組み合わせてアプリケーションの作成を行うことができます。①

## 1-4. 使用可能な通信デバイス

本クラスライブラリでシリアル通信を用いてリーダライターの制御を行う場合、WindowsAPIを使ってシリアル通信デバイスの操作を行います。

従って、OSが認識するシリアルポートであれば、通常問題なく動作すると考えられます。

しかし、ごくまれに古いUSBタイプのシリアルポートを使用する場合に、シリアルポートメーカーから最新のドライバーを入手するなど、本クラスライブラリを使ってアプリケーションを開発する側での対応が必要になる可能性があります。

また、本クラスライブラリでUSB通信を用いてリーダライターの制御を行う場合、ターゲットホストのUSBホストコントローラを弊社製デバイスドライバーが正しく認識し、制御する必要があります。

本クラスライブラリは、Windows XP Professional SP3での動作検証を行っております。それ以外のWindowsファミリOSでも動作すると思われますが、その際には動作確認を十分に行って、導入してください。

## 1-5. 本クラスライブラリの名前空間とアセンブリ情報

本クラスライブラリの名前空間とアセンブリ情報を下記に示します。お使いのクラスライブラリのアセンブリ情報をご確認下さい。

| 項目 | 値 |
|---|---|
| 名前空間 | SanwaNewtec.DeviceControls |
| アセンブリ名 | SanwaNewtec.DeviceControls.V31Control.dll |
| バージョン | 3.0.0.0②④⑤⑥ |
| 言語 | ニュートラル言語 |
| GUID | 8b83a109-699d-4fe0-af6c-f2a836e182da |
| 公開キー | 00 24 00 00 04 80 00 00 94 00 00 00 06 02 00 00<br>00 24 00 00 52 53 41 31 00 04 00 00 01 00 01 00<br>C9 03 04 9B E8 CD AC 2A D2 DC 95 CC 20 97 08 D6<br>60 B0 61 C2 51 48 66 87 07 D5 FE 86 ED 86 7A 7B<br>65 5C 55 6F E1 94 80 2C 50 B0 C8 E0 8A C4 D2 46<br>AA FE DE 18 4E 54 E7 55 8D 8A DE 60 A9 B1 9D 61<br>BF A4 B7 EF 93 D2 CE B5 A7 98 28 D6 91 EB B6 67<br>42 AF B7 58 B7 AC 8F E5 B5 B8 22 DB 17 82 F9 8F<br>C9 A2 BF B4 F3 A6 18 CA 9B 41 85 F0 12 16 D3 A3<br>A0 18 60 7C A4 DB CD 4E 0B E8 2E 18 A7 29 C2 CA |
| トークン | 50 59 84 83 E4 5D C6 94 |

## 1-6. 本クラスライブラリ固有の注意点

本クラスライブラリを使ったアプリケーションの開発は、.NET Frameworkが提供する技術に大きく依存しています。本書では、.NETテクノロジそのものについての説明は行いませんので、アプリケーション作成の際には、同技術に関するMicrosoft社の技術文書や、Microsoft社が提供するMSDN(http://www.microsoft.com/japan/msdn/)などをご活用下さい。

本クラスライブラリでは、リーダライターとの通信を、アプリケーションスレッドとは別のスレッドで行っています。このため、本クラスライブラリから提供されるクラスをインスタンス化する場合には、インスタンスの破棄を確実に行うようにしてください。　これが行われない場合、アプリケーションの動作が不安定になる可能性があります。

以下、プログラミングに関するヒントを示します。

### 1-6-1. Windows Form で使用する場合

Windows Formで使用する場合、フォームのロード時にインスタンスを作成し、フォームのクローズ時にインスタンスのDispose()メソッドを呼び出すようにして下さい。

このようにすることで、フォームのクローズ時に確実にクラスライブラリ内のスレッドを安全に停止できます。

```
public class MyForm: Form
{
   V31SerialControl  _v31= null ; // インスタンス
   // フォームをロードする時
   private void MyForm_Load(object sender, EventArgs e )
   {
      _v31= new V31SerialControl() ; // インスタンス作成
   }
   // フォームを破棄する時
   private void MyForm_FormClosed(object sender, FormClosedEventArg e)
   {
      _v31.Dispose() ; // インスタンスを破棄
   }
}
```

### 1-6-2. Windows Form 以外での呼び出し

インスタンスを生成後、使用し終わったらDispose()メソッドを確実に呼び出してください。

また、例外を確実に捕捉するコードを記述することで、作成するアプリケーションをより堅牢にすることができます。

```
using(V31SerialControl v31= new V31SerialControl() ){
  try {
    v31.PortName= "COM5" ; // COM5ポートを使用する
    v31.Open() ;           // 通信デバイスを開く
    v31.Initialize() ;     // 初期化コマンド実行
  }
  catch(Exception ex ){
    // エラーが発生したら要因を表示する
    MessageBox.Show( ex.Message ) ;
  }
  finally {
    v31.Close() ; // エラーが発生しても確実に通信デバイスを閉じる
  }
}
```

## 2. サンプルプログラム

本製品には、C#言語で記述されたサンプルプログラムが添付されています。本クラスライブラリが提供する全てのコマンドの呼び出し方法が記述されていますので、参考にして下さい。

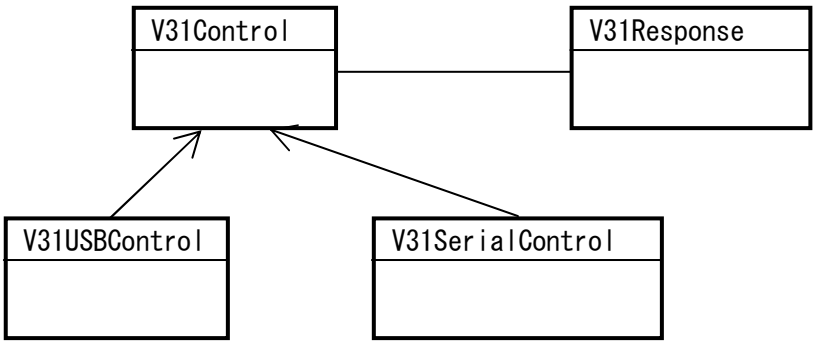
Microsoft Visual Studio 2008でソリューションを開き、サンプルプログラムをビルドしたり、デバッガで処理の追跡を行うことで、クラスライブラリへの理解をより深めることができると思います。

サンプルプログラムは、クラスライブラリの使用方法の一例を示しただけで、アプリケーション作成へのアプローチに対してなんら制限を加えるものではありません。実際のアプリケーション開発では、サンプルコードに囚われることなく、自由な発想でアプリケーションに最適なソリューションを構築してみて下さい。

# 3. 関数リファレンス

## 3-1. クラスライブラリの構成

クラスライブラリは、(1)リーダライターからのレスポンスの各項目に簡単にアクセスするための**リーダライターレスポンスクラス（V31Response）**と、コンピューターの通信デバイスを設定し、リーダライターに対して各コマンドを送信し、そのレスポンスを受信する仮想的な関数インタフェースを持つ**リーダライター制御クラス（V31Control）**と、これから派生した実デバイスとのアクセスを行う**シリアル通信リーダライター制御クラス（V31SerialControl）**と**USB通信リーダライター制御クラス（V31USBControl）**という２つのクラスと、**サポートのための列挙型定数**から構成されます。




- **リーダライター制御クラス（V31Control）**の各メソッド用いて、リーダライターにコマンドを送信した場合、そのレスポンスのバイトストリームはメソッドの戻り値になります。

作成されるアプリケーションプログラムで、レスポンスデータの全てにアクセスしたい場合には、戻り値を取得して、アプリケーション固有の処理を行うように記述して下さい。

- **リーダライター制御クラス（V31Control）**の各メソッドは、**リーダライターレスポンスクラス(V31Response)**を参照するタイプのものを含めてオーバーロードされて定義されています。

リーダライターからのレスポンスの一部だけ⚠にアクセスしたい場合には、**V31Response**インスタンスを渡して、メソッド呼び出しを行い、**V31Response**のプロパティにアクセスして、アプリケーション固有の処理を行うように記述して下さい。

- **リーダライター制御クラス（V31Control）**のプロパティ、メソッドの操作において問題が発生した場合、クラスライブラリから例外が送出されます。

アプリケーションは例外を捕捉して例外への対応を行ない、堅牢なシステムを作成されるようお願いいたします。

## 3-2. リーダライターレスポンスクラス（V31Response）

リーダライターへのコマンドに対するレスポンスを保持し、そのフィールドに簡単にアクセスするためのクラスです。このクラスのプロパティは全て読み取り専用です。

**名前空間　　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

public class V31Response⚠2

**パブリックプロパティの一覧**

| プロパティ | 説　明 |
|---|---|
| CMD | レスポンスのコマンド（CMD）を示します。 |
| RPS1 | レスポンスのカード状態（RPS1）を示します。 |
| RPS2 | レスポンスのエラー状態（RPS2）を示します。 |
| RPS3 | レスポンスの処理状態（RPS3）を示します。 |
| Data | レスポンスのデータ列（バイト列）を示します。データのないコマンドに対する応答では、nullとなります。 |
| DataLength | レスポンスのデータ列長を示します。V31プロトコルの制限から、この値は０～２４９の範囲になります。データのないコマンドに対する応答では、０を返します。 |

**パブリックメソッドの一覧**

パブリックメソッドはありません。

**《解説》**

以下のプログラムは、リーダライターに対してカード排出要求を発行し、カードが排出されるまで待機する時の記述例です。

V31Response⚠2を用いた場合と用いない場合のプログラムを比較してみてください。

```
using(V31SerialControl v31= new V31SerialControl() ){
  V31Response res= null ; // V31Responseのインスタンスをnullで作成
  try {
    v31.PortName = "COM5" ; // COM5ポートにV31が接続
    v31.Open() ;            // 通信ポートのオープン
    //
    // カード排出＆カードなし待ち
    v31.Eject( ref res ) ;  // カード排出
    if(res.RPS2!= 0x30 ) throw new Exception("エラー発生" ) ;
    while( res.RPS1!= 0x30 ){
      Thread.Sleep( 100 ) ;        // 再送要求には100ms以上待機
      v31.GetResponse( ref res ) ; // 再送要求
      if(res.RPS2!= 0x30 ) throw new Exception("エラー発生" ) ;
    }
    // カード排出完了
  }
```

```
  catch(Exception ex ){
    MessageBox.Show( ex.Message ) ;  // エラー表示
  }
  finally {
    v31.Close() ; // 通信ポートを閉じる
  }
}
```

同じ処理をV31Responseを用いずに記述した場合、バイトストリーム列に直接アクセスする必要があり、プログラムの可読性が劣ります。

```
using(V31SerialControl v31= new V31SerialControl() ){
  byte[] res= null ; // バイトストリーム列をnullで作成
  try {
    v31.PortName = "COM5" ; // COM5ポートにV31が接続
    v31.Open() ;            // 通信ポートのオープン
    //
    // カード排出＆カードなし待ち
    res= v31.Eject() ;  // カード排出
    if(res[4]!= 0x30 ) throw new Exception("エラー発生" ) ;
    while( res[3]!= 0x30 ){
      Thread.Sleep( 100 ) ;        // 再送要求には100ms以上待機
      res= v31.GetResponse() ;     // 再送要求
      if(res[4]!= 0x30 ) throw new Exception("エラー発生" ) ;
    }
    // カード排出完了
  }
  catch(Exception ex ){
    MessageBox.Show( ex.Message ) ;  // エラー表示
  }
  finally {
    v31.Close() ; // 通信ポートを閉じる
  }
}
```

## 3-3. シリアル通信リーダライター制御クラス（V31SerialControl）

コンピューターのシリアル通信デバイスを制御するクラスです。リーダライターへのコマンド送信、レスポンス受信は、V31Controlで行います。

**名前空間　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public class V31SerialControl: V31Control
```

**パブリックプロパティの一覧**

| プロパティ | 説　明 |
|---|---|
| PortNo | 通信ポート番号を取得または設定します。 |
| PortName | 通信ポート名を取得または設定します。 |
| BaudRate | 通信速度（ボーレイト）を取得または設定します。 |
| Parity | パリティを取得または設定します。 |
| StopBits | ストップビットを取得または設定します。 |
| CTS | CTSの状態を取得します。 |

**パブリックメソッドの一覧**

| メソッド名 | ｺﾏﾝﾄﾞｺｰﾄﾞ | 説　明 |
|---|---|---|
| GetPortNames() | ー | コンピューターで使用可能なシリアルポート名の配列を返します。 |
| V31SerialControl() | ー | V31SerialControlインスタンスを生成します。 |

**パブリックイベントの一覧**

| イベント | 説　明 |
|---|---|
| ChangeCTS | CTSの状態が変化したら発生する。 |

### 3-3-1. V31SerialControl.PortNo プロパティ

通信ポート番号を取得、または設定します。

**名前空間　　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public byte PortNo { get ; set ; }
```

**《解説》**

・ポート番号には、未オープンのシリアルポート番号が指定できます。
・ポート番号は、１～２５５の範囲の整数値です。
・使用できるポート番号は、コントロールパネルから確認できます。またV31SerialControl.GetPortNames()メソッドによりポート名を列挙できます。

**《例外》**

| 例　外 | 要　因 |
|---|---|
| System.IO.IOException | 既に開いているポート番号を指定した。<br>コンピューターで使用できないポート番号を指定した。 |

### 3-3-2. V31SerialControl.PortName プロパティ

通信ポート名を取得、または設定します。

**名前空間　　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public string PortName { get ; set ; }
```

**《解説》**

・ポート名は、"COM"で始まる整数値で表記されます
・ポート名には、未オープンのシリアルポート名が指定できます。
・使用できるポート名は、コントロールパネルから確認できます。またV31SerialControl.GetPortNames()メソッドによりポート名を列挙できます。

**《例外》**

| 例　外 | 要　因 |
|---|---|
| System.ArgumentNullException | ポート名にnullを指定した。 |
| System.FormatException | ポート名が"COM"で始まる整数値でない。 |
| System.IO.IOException | 既に開いているポート番号を指定した。<br>コンピューターで使用できないポート番号を指定した。 |

### 3-3-3. V31SerialControl.BaudRate プロパティ

通信速度（ボーレイト）を取得、または設定します。

**名前空間　　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public BAUDRATE BaudRate { get ; set ; }
```

**《解説》**

・通信速度は、BAUDRATE列挙型で指定します。
・指定できる通信速度は、1200bps、2400bps、4800bps、9600bps、19200bps、38400bpsの６種ですが、「ＡＢＳ－Ｖ３１」で使用できるのは、4800bps、9600bps、19200bps、38400bpsの４種のみです。
・低速の通信速度は、他の機器のために設定できるようになっています。
・通信速度と列挙名の対応を下記の表で示します。

public enum BAUDRATE: int

| 列挙名 | 通信速度 |
|---|---|
| BAUDRATE._1200 | 1200bps |
| BAUDRATE._2400 | 2400bps |
| BAUDRATE._4800 | 4800bps |
| BAUDRATE._9600 | 9600bps |
| BAUDRATE._19200 | 19200bps |
| BAUDRATE._38400 | 38400bps |

・整数値で指定する場合には、9600などのリテラル値を用いて、(BaudRate)でキャストして使用して下さい。

**《例外》**

| 例　外 | 要　因 |
|---|---|
| System.ArgumentOutOfRangeException | 通信速度に上記６種以外の値を設定しようとした。 |
| System.IO.IOException | 既に開いているV31SerialControlに対して、通信速度の設定を行おうとした。 |

### 3-3-4. V31SerialControl.Parity プロパティ

パリティを取得、または設定します。

**名前空間　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public PARITY Parity { get ; set ; }
```

**《解説》**

・パリティは、PARITY列挙型で指定します。
・指定できるパリティは、「パリティなし」，「偶数パリティ」，「奇数パリティ」の３種類です。
・パリティと列挙名の対応を下記の表で示します。

public enum PARITY: byte

| 列挙名 | パリティ |
|---|---|
| PARITY.NO | パリティなし |
| PARITY.EVEN | 偶数パリティ |
| PARITY.ODD | 奇数パリティ |

・整数値をPARITY型にキャストして、パリティの設定を行うことも可能ですが、パリティの値が離散値なので、注意して記述してください。

**《例外》**

| 例　外 | 要　因 |
|---|---|
| System.ArgumentOutOfRangeException | パリティに上記３種以外の値を設定しようとした。 |
| System.IO.IOException | 既に開いているV31SerialControlに対して、パリティの設定を行おうとした。 |

### 3-3-5. V31SerialControl.StopBits プロパティ

ストップビットを取得、または設定します。

**名前空間　　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public STOPBITS StopBits { get ; set ; }
```

**《解説》**

・ストップビットは、STOPBITS列挙型で指定します。
・指定できるストップビットは、「１ビット」，「２ビット」の２種類ですが、「ＡＢＳ－Ｖ３１」で使用できるのは、１ビットの一種類のみです。
・ストップビットと列挙名の対応を下記の表で示します。

public enum STOPBITS: byte

| 列挙名 | ストップビット |
|---|---|
| STOPBITS.ONE | １ビット |
| STOPBITS.TWO | ２ビット |

・整数値をSTOPBITS型にキャストして、ストップビットの設定を行うことも可能ですが、ストップビットの値が離散値なので、注意して記述してください。

**《例外》**

| 例　外 | 要　因 |
|---|---|
| System.ArgumentOutOfRangeException | ストップビットに上記２種以外の値を設定しようとした。 |
| System.IO.IOException | 既に開いているV31SerialControlに対して、ストップビットの設定を行おうとした。 |

### 3-3-6. V31SerialControl.CTS プロパティ

ＣＴＳの状態を取得します。

**名前空間　　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public bool CTS { get ; }
```

**《解説》**

・リーダライターは受信準備が整うと、RTS（ホスト側ではCTS）をONにして受信準備が整ったことをホストに通知します。本プロパティを参照することで、リーダライターの受信準備が完了したこと知ることができます。
・CTSのOFF時間を監視することで、リーダライターとホストが接続されていない、あるいは、リーダライターの電源がOFFしているなどの判定にも利用することができます。
・V31SerialControlが閉じている状態で本プロパティを参照すると例外が送出されます。

**《例外》**

| 例　外 | 要　因 |
|---|---|
| System.IO.IOException | V31SerialControlがクローズ状態である。<br>通信デバイスの操作に失敗した。 |

### 3-3-7. V31SerialControl.VbBaudRate プロパティ△5

通信速度（ボーレイト）を取得、または設定します。VisualBasicによる開発用プロパティです。

**名前空間　　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public VBEBAUDRATE VbBaudRate { get ; set ; }
```

**《解説》**

・通信速度は、VBEBAUDRATE列挙型で指定します。
・指定できる通信速度は、1200bps、2400bps、4800bps、9600bps、19200bps、38400bpsの６種ですが、「ＡＢＳ－Ｖ３１」で使用できるのは、4800bps、9600bps、19200bps、38400bpsの４種のみです。
・低速の通信速度は、他の機器のために設定できるようになっています。
・通信速度と列挙名の対応を下記の表で示します。

public enum VBEBAUDRATE: int

| 列挙名 | 通信速度 |
|---|---|
| VBEBAUDRATE._1200 | 1200bps |
| VBEBAUDRATE._2400 | 2400bps |
| VBEBAUDRATE._4800 | 4800bps |
| VBEBAUDRATE._9600 | 9600bps |
| VBEBAUDRATE._19200 | 19200bps |
| VBEBAUDRATE._38400 | 38400bps |

・整数値で指定する場合には、9600などのリテラル値を用いて、(VbBaudRate)でキャストして使用して下さい。

**《例外》**

| 例　外 | 要　因 |
|---|---|
| System.ArgumentOutOfRangeException | 通信速度に上記６種以外の値を設定しようとした。 |
| System.IO.IOException | 既に開いているV31SerialControlに対して、通信速度の設定を行おうとした。 |

### 3-3-8. V31SerialControl.VbParity プロパティ△5

パリティを取得、または設定します。VisualBasicによる開発用プロパティです。

**名前空間　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public VBEPARITY VbParity { get ; set ; }
```

**《解説》**

・パリティは、VBEPARITY列挙型で指定します。
・指定できるパリティは、「パリティなし」，「偶数パリティ」，「奇数パリティ」の３種類です。
・パリティと列挙名の対応を下記の表で示します。

public enum VBEPARITY: byte

| 列挙名 | パリティ |
|---|---|
| VBEPARITY.NO | パリティなし |
| VBEPARITY.EVEN | 偶数パリティ |
| VBEPARITY.ODD | 奇数パリティ |

・整数値をVBEPARITY型にキャストして、パリティの設定を行うことも可能ですが、パリティの値が離散値なので、注意して記述してください。

**《例外》**

| 例　外 | 要　因 |
|---|---|
| System.ArgumentOutOfRangeException | パリティに上記３種以外の値を設定しようとした。 |
| System.IO.IOException | 既に開いているV31SerialControlに対して、パリティの設定を行おうとした。 |

### 3-3-9. V31SerialControl.VbStopBits プロパティ△5

ストップビットを取得、または設定します。VisualBasicによる開発用プロパティです。

**名前空間　　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public VBESTOPBITS VbStopBits { get ; set ; }
```

**《解説》**

・ストップビットは、VBESTOPBITS列挙型で指定します。
・指定できるストップビットは、「１ビット」，「２ビット」の２種類ですが、「ＡＢＳ－Ｖ３１」で使用できるのは、１ビットの一種類のみです。
・ストップビットと列挙名の対応を下記の表で示します。

public enum VBESTOPBITS: byte

| 列挙名 | ストップビット |
|---|---|
| VBESTOPBITS.ONE | １ビット |
| VBESTOPBITS.TWO | ２ビット |

・整数値をVBESTOPBITS型にキャストして、ストップビットの設定を行うことも可能ですが、ストップビットの値が離散値なので、注意して記述してください。

**《例外》**

| 例　外 | 要　因 |
|---|---|
| System.ArgumentOutOfRangeException | ストップビットに上記２種以外の値を設定しようとした。 |
| System.IO.IOException | 既に開いているV31SerialControlに対して、ストップビットの設定を行おうとした。 |

### 3-3-10. V31SerialControl.GetPortNames() メソッド

コンピューターで使用可能なシリアルポート名の配列を返します。

**名前空間　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public static string[] GetPortNames()
```

**《解説》**

・コンピューターで使用可能なシリアルポート名の配列を返します。
・現在、オープン中のシリアルポート名は含まれません。
・シリアルポート名は、"COM"で始まる整数値で表記されます。

**《引数》**

なし。

**《戻り値》**

シリアルポート名を示す文字列の配列。

**《例外》**

なし。

**《コード例》**

・本メソッドを用いて、System.Windows.Forms.ComboBoxにシリアルポート名の一覧を設定するコード例を示します。

```
using System.Windows.Forms ;
using SanwaNewtec.DeviceControls ;

class MyForm: Form
{
  System.Windows.Forms.ComboBox  cbPortName ; // ポート名を保持するコンボボックス
  public MyForm()
  {
    cbPortName= new ComboBox() ;
  }
  private void MyForm_Load(object sender, EventArgs e )
  {
    // コンピューターで使用できるシリアルポート名を列挙し、コンボボックスに
    // 項目リストとして設定する。
    cbPortName.Items.AddRange( V31SerialControl.GetPortNames() ) ;
  }
}
```

### 3-3-11. V31SerialControl.ChangeCTS イベント

CTSの状態が変化したら発生する。

**名前空間　　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public delegete void ChangeCTSEventHandler(object sender, CTSEventArgs e)
public event ChangeCTSEventHandler  ChangeCTS
```

**《解説》**

・リーダライターのRTS（ホスト側CTS）オン、オフに合わせて発生するイベントです。
・V31SerialControlがオープンした状態で発生します。
・CTSのON/OFFは、CTSEventArgsのCTSプロパティで参照できます。
・イベントハンドラは、V31SerialControlの通信スレッドから呼び出されるため、イベントハンドラ内でWindows Formのコントロールの操作を行う場合、Invoke呼び出しを行い、スレッド境界を越えるよう配慮してください。
・イベントハンドラを登録し、Invoke呼び出しを行う際、V31Control.LockEventsプロパティを真にしないで下さい。アプリケーションのメッセージループ呼び出しが抑制されるため、本イベントが発生した場合、V31SerialControlの通信スレッドが停止し、その結果、アプリケーションが停止します。

**《プログラム例》**

下記のプログラムは、CTSオン、オフイベント発生時にSystem.Windows.Forms.Labelのテキスト文字列を変更し、ユーザーに視覚的に通知します。

```
class MyForm: Form
{
  V31SerialControl            _v31= null ;
  System.Windows.Forms.Label  lbCTS ; // CTS-ON,OFF 表示ラベル
  private void MyForm_Load(object sender, EventArgs e )
  {
    _v31= new V31SerialControl() ;
    // イベントハンドラの登録
    _v31.ChangeCTS+= new V31SerialControl.ChageCTSEventHandler(OnChageCTSHandler );
  }
  private void OnChangeCTSHandler(object sender, CTSEventArgs e )
  {
    // Invokeで匿名メソッド呼び出し
    Invoke( new MethodInvoker(
      delegete {
        lbCTS.Text= e.CTS? "CTS-ON": "CTS-OFF" ; // ラベルの表示の変化
      }
    ) ) ;
  }
}
```

### 3-4. USB 通信リーダライター制御クラス（V31USBControl）

コンピューターのUSBホストコントローラを制御するクラスです。本クラスを使用するためには、ターゲットホストにV31用USB通信デバイスドライバーを予めインストールしておく必要があります。リーダライターへのコマンド送信、レスポンス受信はV31Controlで行います。

**名前空間　　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public class V31USBControl: V31Control
```

⚠6 ※Windows7 64bit版で使用する場合
ターゲットホストに、Microsoft WinUSBドライバーを予めインストールしておく必要があります。インストール手順については、「4.USBデバイスドライバーのインストール方法」を参照してください。

USBには、通信速度などの設定パラメーターがないため固有のプロパティ、メソッドは存在しません。

### 3-5. リーダライター制御クラス（V31Control）

コンピューターの通信デバイスを制御し、リーダライターへコマンドを送信し、そのレスポンスを取得するためのクラスです。実際の通信デバイスへの詳細な設定は、通信デバイス毎に設けられたV31SerialControlとV31USBControlで行います。

**名前空間　　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public class V31Control: IDisposable
```

**パブリックプロパティの一覧**

| プロパティ | 説　明 |
|---|---|
| Timeout | コマンド実行時のタイムアウトを取得または設定します。 |
| RetryTime | リーダライターが無応答時にコマンドを再送するまでの待機時間を取得または設定します。 |
| IsOpen | 通信デバイスが開いていれば真を、閉じていれば偽を返します。 |
| LockEvents | コマンド実行時におけるメッセージループの無効状態を取得、または設定します。 |

**パブリックメソッドの一覧**

| メソッド名 | ｺﾏﾝﾄﾞｺｰﾄﾞ | 説　明 |
|---|---|---|
| GetNecessaryBytes() | － | イメージの幅と高さ（ドット単位）を与えると、そのイメージを保持するために必要なメモリサイズをバイト数で返します。 |
| V31Control() | － | V31Controlのインスタンスを生成します。 |
| Dispose() | － | V31Controlのインスタンスを破棄します。 |
| Close() | － | 通信デバイスを解放し、V31Controlをクローズします。 |
| Open() | － | 通信デバイスを取得し、V31Controlを開きます。 |
| Initialize() | 10h | イニシャルコマンドを実行します。 |
| StatusRead() | 20h | ステータスリードコマンドを実行します。 |
| GetSensorStatus() | 21h | Ｒ／Ｗ搬送路のセンサー情報を取得します。 |
| Read() | 33h | リードコマンドを実行します。 |
| Cancel() | 40h | キャンセルコマンドを実行します。 |
| Write() | 53h | ライトコマンドを実行します。 |
| RegisterGaiji() | 76h | 外字登録コマンドを実行します。 |
| RegisterImage() | 77h | イメージデータ登録コマンドを実行します。 |
| PrintSetting() | 78h | 印字設定コマンドを実行します。 |
| PrintImage() | 7Bh | イメージデータ印字コマンドを実行します。 |
| PrintString() | 7Ch | 文字印字コマンドを実行します。 |
| PrintBarcode() | 7Eh | バーコード印字コマンドを実行します。 |
| Eject() | 80h | カード排出コマンドを実行します。 |
| Cleaning() | A0h | クリーニングコマンドを実行します。 |
| SetClock() | E1h | 時計設定コマンドを実行します。 |
| GetVersion() | F0h | バージョン取得コマンドを実行します。 |
| GetClock() | F1h | 現在時刻取得コマンドを実行します。 |
| GetResponse() | － | レスポンス再送要求（ＥＮＱ送信）を実行します。 |
| SendCommand() | － | 任意のコマンド列を送信します。 |

### 3-5-1. V31Control.Timeout プロパティ

コマンド実行時のタイムアウトを取得、または設定します。

**名前空間　　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public int Timeout { get ; set ; }
```

**《解説》**

・V31Controlは、リーダライターへのコマンド送信の際、リーダライターからレスポンスが返るまで、また、返ったレスポンスが正しいフォーマットであるか確認を行い、問題があれば、コマンドやＥＮＱの再送信（USBであれば、インタラプトトランザクションの実行）を行います。
・本Timeoutプロパティは、このV31Controlのタイムアウト時間をミリ秒単位で設定し、コマンド送信、すなわち、V31Controlに対するメソッド呼び出しからこの時間が経過すると、リーダライターとの通信中であっても処理を中断して、アプリケーションに復帰します。
・タイムアウトが発生してアプリケーションに復帰する場合には、例外としてTimeoutExceptionが送出されます。
・Timeout値として、５０～６０，０００[msec]まで設定可能です。

**《例外》**

| 例　外 | 要　因 |
|---|---|
| System.ArgumentOutOfRangeException | タイムアウトに不正な値を設定しようとした。 |
| System.InvalidOperationException | リーダライターと通信中のV31Controlに対して、タイムアウトの設定を行おうとした。 |

### 3-5-2. V31Control.RetryTime プロパティ

リーダライターが無応答時にコマンドを再送するまでの待機時間を取得、または設定します。

**名前空間　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public int RetryTime { get ; set ; }
```

**《解説》**

・V31Controlは、リーダライターへのコマンド送信後、リーダライターが無応答の場合、再度コマンドを送信します。この時の無応答と判定してからコマンドを再送するまでの待機時間をミリ秒単位で設定します。
・具体的にはコマンド送信からACK受信までの待機時間と、ENQ送信からレスポンス受信までの時間のことです。（USBの場合、NAK受信後からの再送時間になります）
・RetryTime値として、１０～１，０００[msec]まで設定可能です。
・RetryTime値より短い時間で、Timeout値が設定された場合、Timeout値の方が優先されます。このような矛盾した設定を行っても例外は送出されませんのでご注意下さい。

**《例外》**

| 例　外 | 要　因 |
|---|---|
| System.ArgumentOutOfRangeException | RetryTimeに不正は値を設定しようとした。 |
| System.InvalidateOperationException | リーダライターと通信中のV31Controlに対して、タイムアウトの設定を行おうとした。 |

### 3-5-3. V31Control.IsOpen プロパティ

V31Controlが開いていれば真を、閉じていれば偽を返します。

**名前空間　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public bool IsOpen { get ; }
```

**《解説》**

・V31Controlインスタンスに対し、Open()メソッドを呼び出すと、V31Controlは通信デバイスをオープンし、リーダライターとの送受信準備を行います。本プロパティを参照することで、RWControlが開いているかどうかを知ることができます。
・本プロパティが真を返すとき、通信デバイスの設定を行うことはできません。

**《例外》**

なし

### 3-5-4. V31Control.LockEvents プロパティ

V31Controlのコマンド実行時にWindowsメッセージループ④の無効状態を取得、あるいは設定します。

**名前空間　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public bool LockEvents { get ; set ; }
```

**《解説》**

・V31Controlを使ってリーダライターにコマンド送信を行った場合、レスポンスが返るかタイムアウトが発生するまで呼出元の処理は停止されます。この時、呼出元のWindowsメッセージループ④を無効化するかどうかを本プロパティで設定できます。
・インスタンス生成時のデフォルト値は、偽、すなわち、Windowsメッセージループ④が有効になっています。
・このプロパティが真で、Windows Formを使用するとき、V31Controlがリーダライターと通信中の場合、Windows Formのコントロールの制御が行えないことがあります。④

**《例外》**

| 例　外 | 要　因 |
|---|---|
| System.InvalidOperationException | リーダライターと通信中のRWColtrolに対して、Windowsメッセージループ④の有効／無効化を行おうとした。 |

### 3-5-5. V31Control.GetNecessaryBytes() メソッド

イメージの幅と高さ（ドット単位）を与えると、そのイメージを保持するために必要なメモリサイズをバイト数で返します。

**名前空間**　SanwaNewtec.DeviceControls
**アセンブリ**　SanwaNewtec.DeviceControls.V31Control.dll

**《構文》**

```
public static uint GetNecessaryBytes(ushort width, ushort height)
```

**《解説》**

・イメージは、１ドット＝１ビットで表現されますが、イメージ幅は１バイト単位で管理されるため、幅１ドット×高さ１０ドットのイメージであっても、１バイト×１０ヶ＝１０バイトのメモリが必要になります。※2
・本メソッドは、この必要なメモリサイズをバイト単位でします。

**《引数》**

| | |
|---|---|
| width | ドット単位でのイメージ幅 |
| height | ドット単位でのイメージ高さ |

**《戻り値》**

イメージを保持するために必要なバイト数。

**《例外》**

なし

### 3-5-6. V31Control.V31Control() コンストラクタ

V31Controlのインスタンスを生成します。

**名前空間**　SanwaNewtec.DeviceControls
**アセンブリ**　SanwaNewtec.DeviceControls.V31Control.dll

**《構文》**

```
public V31Control()
```

**《解説》**

V31Controlクラスのデフォルトコンストラクタです。

**《引数》**

なし。

**《例外》**

なし。

### 3-5-7. V31Control.Dispose() メソッド

V31Controlのインスタンスを破棄します。

**名前空間　　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public void Dispose()
```

**《解説》**

- V31Controlインスタンスによって使用されているアンマネージリソースを解放します。V31Controlインスタンスを使い終わったら必ず呼び出す必要があります。
- 確実な方法は、前述しましたが、Windows Formであれば、フォームのロードでインスタンスを生成し、フォームのクローズ時にDispose()呼び出しを行うことです。
- Visual Basic .NET、Visual C#であれば、Using（C#ではusing）文を使うことで、Dispose()呼び出しを確実に行うことができます。具体的には、同言語のリファレンスを参照して下さい。

**《引数》**

なし。

**《戻り値》**

なし。

**《例外》**

なし。

### 3-5-8. V31Control.Close() メソッド

通信デバイスを解放し、V31Controlをクローズします。

**名前空間　　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public void Close()
```

**《解説》**

・リーダライターとの通信を行う内部スレッドを停止し、通信デバイスを閉じます。
・通信デバイスを閉じた後は、通信設定（ポート名、ボーレイトなど）を変更することができます。
・通信デバイスに問題がなければ、再び開くことも可能です。
・既に閉じているV31Controlインスタンスに対して再度Close()メソッドを呼び出しても何ら問題はありません。

**《引数》**

なし。

**《戻り値》**

なし。

**《例外》**

なし。

### 3-5-9. V31Control.Open() メソッド

通信デバイスを取得し、V31Controlを開きます。

**名前空間　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public void Open()
```

**《解説》**

・通信デバイスを開いて、リーダライターとの通信を行うスレッドを内部的に起動します。
・通信デバイスを開いた後は、通信設定（ポート名、ボーレイト）などの変更は行えません。
これらの変更を行った場合、例外が送出されます。
・既に開いているV31Controlに対して、再度Open()メソッドを呼び出すと例外が送出されます。

**《引数》**

なし。

**《戻り値》**

なし。

**《例外》**

| 例　外 | 要　因 |
|---|---|
| System.InvalidOperationException | 既にオープン済みのRWColtrolに対して、再度オープンメソッドを呼び出した。 |
| System.IO.IOException | 通信デバイスの操作でエラーが発生した。 |

### 3-5-10. V31Control.Initialize() メソッド

イニシャルコマンド（ＣＭＤ＝１０ｈ）を実行します。

**名前空間　　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public byte[] Initialize(byte iniMode, byte rspMode ) ;
public byte[] Initialize(ref V31Response res, byte iniMode, byte rspMode ) ;②
```

**《解説》**

・エラーのクリアー、モーターの動作確認、およびカード挿入待ちの解除などリーダライターの初期化処理を行います。
・イニシャル処理が正常終了していない場合には、以後のリーダライターの動作を保証できませんので、レスポンスを取得し、異常のないことを確認してから処理を行うよう、アプリケーションを作成して下さい。

**《引数》**

iniMode　動作モード指定
　30h　イニシャル動作のみ
　31h　イニシャル動作＋カード排出
　32h　リセット動作

rspMode　レスポンスモード指定
　30h　互換１モード（不正処理、警告レスポンスなし）
　31h　互換２モード（不正処理レスポンスあり、警告レスポンスなし）
　32h　V31モード（不正処理、警告レスポンスあり）

**《戻り値》**

受信パケットのバイトストリーム列

**《例外》**

| 例　外 | 要　因 |
|---|---|
| System.ArgumentOutOfRangeException | 引数の指定が誤っている。 |
| System.InvalidateOperationException | V31Controlが未オープン。 |
| | リーダライターと通信中のため、本メソッドが実行できない。 |
| System.IO.IOException | 通信デバイスの操作でエラーが発生した。 |
| System.TimeoutException | タイムアウトが発生した。 |

### 3-5-11. V31Control.StatusRead() メソッド

ステータスリードコマンド（ＣＭＤ＝２０ｈ）を実行します。

**名前空間　　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public byte[] StatusRead() ;
public byte[] StatusRead(ref V31Response res ) ;⚠2
```

**《解説》**

・リーダライター内のカードの有無､およびカードの位置情報を含むレスポンスを取得します。
・カード処理が終了して、カードを排出した場合には、GetResponse()メソッドを用いるか、本メソッドを用いて、カードの挿入口からカードが取り去られることを監視して下さい。
・カードが残留していると次のカードの取り込みが実行不可となります。

**《引数》**

なし。

**《戻り値》**

受信パケットのバイトストリーム列。

**《例外》**

| 例　外 | 要　因 |
|---|---|
| System.InvalidateOperationException | V31Controlが未オープン。 |
| | リーダライターと通信中のため、本メソッドが実行できない。 |
| System.IO.IOException | 通信デバイスの操作でエラーが発生した。 |
| System.TimeoutException | タイムアウトが発生した。 |

### 3-5-12. V31Control.GetSensorStatus() メソッド

センサー情報取得コマンド（ＣＭＤ＝２１ｈ）を実行します。

**名前空間　　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public byte[] StatusRead(ref bool sensor1, ref bool sensor2, ref bool sensor3,
                         ref bool sensor4, ref bool sensor5 );
public byte[] StatusRead(ref V31Response res, ref bool sensor1, ref bool sensor2,
                         ref bool sensor3, ref bool sensor4, ref bool sensor5 );※2
```

**《解説》**

・Ｒ／Ｗ搬送路のセンサー情報を取得します。
・各センサー情報は、遮光された状態がＯＮ、透過状態がＯＦＦで返されます。
・搬送路センサーは、カード挿入口から奥に向かって順に、センサー１，２，３，４，５となります。

**《引数》**

| | |
|---|---|
| sensor1 | センサー１(ON/OFF)状態 |
| sensor2 | センサー２(ON/OFF)状態 |
| sensor3 | センサー３(ON/OFF)状態 |
| sensor4 | センサー４(ON/OFF)状態 |
| sensor5 | センサー５(ON/OFF)状態 |

**《戻り値》**

受信パケットのバイトストリーム列。

**《例外》**

| 例　外 | 要　因 |
|---|---|
| System.InvalidateOperationException | V31Controlが未オープン。 |
| | リーダライターと通信中のため、本メソッドが実行できない。 |
| System.IO.IOException | 通信デバイスの操作でエラーが発生した。 |
| System.TimeoutException | タイムアウトが発生した。 |

### 3-5-13. V31Control.Read() メソッド

リードコマンド（ＣＭＤ＝３３ｈ）を実行します。

**名前空間　　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public byte[] Read(byte mode, byte check, byte track ) ;
public byte[] Read(ref V31Response res, byte mode, byte check, byte track ) ;②
```

**《解説》**

・カードの取り込み、カードの磁気情報の読み込みを行います。

**《引数》**

mode　リードモード指定

30h　標準（69バイト）リード
31h　ＣＲＡ－２２００互換リード
32h　カード取り込み
33h　可変長（１～69バイト）リード
34h　逆可変長（１～69バイト）リード⑥

check　リードデータチェック

30h　パリティあり（７ビット磁気リード）
31h　パリティなし（８ビット磁気リード）

track　トラック指定

30h　トラック１のリード指定
31h　トラック２のリード指定
32h　トラック３のリード指定
33h　トラック１、２のリード指定
34h　トラック１、３のリード指定
35h　トラック２、３のリード指定
36h　トラック１、２、３全てのリード指定

**《戻り値》**

・受信パケットのバイトストリーム列

**《例外》**

| 例　外 | 要　因 |
|---|---|
| System.ArgumentOutOfRangeException | 引数の指定が誤っている。 |
| System.InvalidateOperationException | V31Controlが未オープン。 |
| | リーダライターと通信中のため、本メソッドが実行できない。 |
| System.IO.IOException | 通信デバイスの操作でエラーが発生した。 |
| System.TimeoutException | タイムアウトが発生した。 |

### 3-5-14. V31Control.Cancel() メソッド

キャンセルコマンド（ＣＭＤ＝４０ｈ）を実行します。

**名前空間　　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public byte[] Cancel() ;
public byte[] Cancel(ref V31Response res ) ;⚠2
```

**《解説》**

・リードコマンド、クリーニングコマンド発行時の「カード挿入待ち」状態を解除します。

**《引数》**

なし。

**《戻り値》**

受信パケットのバイトストリーム列。

**《例外》**

| 例　外 | 要　因 |
|---|---|
| System.InvalidateOperationException | V31Controlが未オープン。 |
| | リーダライターと通信中のため、本メソッドが実行できない。 |
| System.IO.IOException | 通信デバイスの操作でエラーが発生した。 |
| System.TimeoutException | タイムアウトが発生した。 |

### 3-5-15. V31Control.Write() メソッド

ライトコマンド（ＣＭＤ＝５３ｈ）を実行します。

**名前空間　　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public byte[] Write(byte mode, byte check, byte track, byte[] writeData ) ;
public byte[] Write(ref V31Response res, byte mode, byte check, byte track,
                    byte[] writeData ) ;②
```

**《解説》**

・カードに磁気情報を書き込みます。

**《引数》**

mode　ライトモード指定

30h　標準（69バイト）ライト⑥
31h　ＣＲＡ－２２００互換ライト⑥
33h　可変長（１～69バイト）ライト⑥
34h　逆可変長（１～69バイト）ライト⑥

check　パリティ

30h　パリティ指定（７ビット磁気ライト⑥）
31h　パリティなし（８ビット磁気ライト⑥）

track　トラック指定

30h　トラック１のライト指定
31h　トラック２のライト指定
32h　トラック３のライト指定
33h　トラック１、２のライト指定
34h　トラック１、３のライト指定
35h　トラック２、３のライト指定
36h　トラック１、２、３全てのライト指定

**《戻り値》**

・受信パケットのバイトストリーム列

**《例外》**

| 例　外 | 要　因 |
|---|---|
| System.ArgumentOutOfRangeException | 引数の指定が誤っている。 |
| System.ArgumentNullException | ライトデータがnullである。 |
| System.InvalidateOperationException | V31Controlが未オープン。 |
| | リーダライターと通信中のため、本メソッドが実行できない。 |
| System.IO.IOException | 通信デバイスの操作でエラーが発生した。 |
| System.TimeoutException | タイムアウトが発生した。 |

### 3-5-16. V31Control.RegisterGaiji() メソッド

外字登録コマンド（ＣＭＤ＝７６ｈ）を実行します。

**名前空間　　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public byte[] RegisterGaiji(byte number, byte size, byte[] gaijiData ) ;
public byte[] RegisterGaiji(ref V31Response res, byte number, byte size,
                            byte[] gaijiData ) ;②
```

**《解説》**

・リーダライターに外字を登録します。
・登録できる外字は、24×24ドット，16×16ドット，12×24ドット，8×16ドットの４種類です。
　与える外字データの大きさは､V31Control.GetNecessaryBytes()メソッドで算出した大きさで定義する必要があります。

**《引数》**

number　外字番号

size　サイズ
　30h　全角24×24ドット指定
　31h　全角16×16ドット指定
　32h　半角12×24ドット指定
　33h　半角 8×16ドット指定

gaijiData　登録するフォントデータ

**《戻り値》**

受信パケットのバイトストリーム列。

**《例外》**

| 例　外 | 要　因 |
|---|---|
| System.ArgumentOutOfRangeException | 引数の指定が誤っている。 |
| System.ArgumentNullException | 外字がnullである。 |
| System.InvalidateOperationException | V31Controlが未オープン。 |
| | リーダライターと通信中のため、本メソッドが実行できない。 |
| System.IO.IOException | 通信デバイスの操作でエラーが発生した。 |
| System.TimeoutException | タイムアウトが発生した。 |

### 3-5-17. V31Control.RegisterImage() メソッド

イメージデータ登録コマンド（ＣＭＤ＝７７ｈ）を実行します。

**名前空間　　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public byte[] RegisterImage(ushort address, ushort width, ushort height,
                            byte[] imageData ) ;
public byte[] RegisterImage(ref V31Response res, ushort address, ushort width,
                            ushort height, byte[] imageData ) ;②
```

**《解説》**

・リーダライターの指定したアドレス上にイメージを登録します。
・アドレスは、0000h～FFFEhの範囲内で、かつ偶数アドレスで指定する必要があります。
・イメージ幅は、1～320ドットの範囲で指定して下さい。②
・イメージ高さは、1～576ドットの範囲で指定して下さい。
・イメージデータの大きさは、V31Control.GetNecessaryByte()で取得した値に揃える必要があります。
・イメージデータの転送は、V31Controlインスタンス内で、送信データを複数のパケットに分割して送信します。従って多量のデータを送る際には、Timeoutプロパティで設定するタイムアウト値を長めに再設定して下さい。

**《引数》**

| | |
|---|---|
| address | 登録アドレス② |
| width | イメージ幅 |
| height | イメージ高さ |
| imageData | イメージデータ |

**《戻り値》**

受信パケットのバイトストリーム列。

**《例外》**

| 例　外 | 要　因 |
|---|---|
| System.ArgumentOutOfRangeException | 引数の指定が誤っている。 |
| System.ArgumentNullException | イメージデータがnullである。 |
| System.InvalidateOperationException | V31Controlが未オープン。 |
| | リーダライターと通信中のため、本メソッドが実行できない。 |
| System.IO.IOException | 通信デバイスの操作でエラーが発生した。 |
| System.TimeoutException | タイムアウトが発生した。 |

### 3-5-18. V31Control.PrintSetting() メソッド

印字設定コマンド（ＣＭＤ＝７８ｈ）を実行します。

**名前空間　　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

［半角フォント指定あり］

```
public byte[] PrintSetting(byte card, byte mode, byte align, byte font ) ;
public byte[] PrintSetting(ref V31Response res, byte card, byte mode, byte align,
                           byte font ) ;⚠2
```

［半角フォント指定なし］

```
public byte[] PrintSetting(byte card, byte mode, byte align ) ;
public byte[] PrintSetting(ref V31Response res, byte card, byte mode,
                           byte align ) ;⚠2
```

**《解説》**

・各種印字コマンドの印字条件を設定します。
・半角フォント指定を行わないメソッド呼び出しが行われた場合、リーダライターの標準フォントが使われます。

**《引数》**

card　カード種類

| | | | | |
|---|---|---|---|---|
| 30h | 白濁リライト | リコー | ＦＢ－６６１Ｍ | 発色：白 |
| 32h | ロイコカード | 三菱製紙 | ＴＲＣＧＡＡＣＳ／ＣＨ | 発色：青 |
| 33h | ９０℃サーマル | 水三島紙工 | ＳＮＴ－２３１－Ｎ１－Ｔ２ | 発色：黒 |
| 34h | １１０℃サーマル | 水三島紙工 | ＳＮＴ－２２０－Ｔ３ | 発色：黒 |
| 37h | ロイコリライト | 三菱製紙 | ＴＲＣＧ９９ＳＳ／ＳＨ | 発色：青 |
| 38h | ロイコリライト | リコー | ＣＲ－６３１ＦＢ | 発色：黒 |
| 39h | 紙カード | サトー | | 発色：黒 |
| 3Ah | ＲＥＤカード | 小林記録紙 | ＫＳＮＢ－５３６０／５３７６ | 発色：白 |
| 3Bh | ロイコリライト | 三菱製紙 | ＴＲＣＧＢ３ＢＳＢ | 発色：黒 |

mode　印字方法

30h　消去＋印字
31h　重ね書き印字

align　印字座標

30h　標準座標
33h　左回転座標
34h　右回転座標

font　印字する半角フォント

30h　標準フォント（デフォルト値）
31h　軸太フォント

**《戻り値》**

受信パケットのバイトストリーム列。

**《例外》**

| 例　外 | 要　因 |
|---|---|
| System.ArgumentOutOfRangeException | 引数の指定が誤っている。 |
| System.InvalidateOperationException | V31Controlが未オープン。 |
| | リーダライターと通信中のため、本メソッドが実行できない。 |
| System.IO.IOException | 通信デバイスの操作でエラーが発生した。 |
| System.TimeoutException | タイムアウトが発生した。 |

### 3-5-19. V31Control.PrintImage() メソッド

イメージデータ印字コマンド（ＣＭＤ＝７Ｂｈ）を実行します。

**名前空間　　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

［登録アドレス指定］

```
public byte[] PrintImage(byte mode, byte clear, ushort width, ushort height,
                         ushort x, ushort y, ushort address ) ;
public byte[] PrintImage(ref V31Response res, byte mode, byte clear, ushort width,
                         ushort height, ushort x, ushort y, ushort address ) ;⚠2
```

［送信イメージデータ指定］

```
public byte[] PrintImage(byte mode, byte clear, ushort width, ushort height,
                         ushort x, ushort y, byte[] imageData ) ;
public byte[] PrintImage(ref V31Response res, byte mode, byte clear, ushort width,
                         ushort height, ushort x, ushort y, byte[] imageData ) ;⚠2
```

**《解説》**

・指定された座標にイメージデータを印字します。
・イメージデータの印字には、予めリーダライターに登録されているイメージデータを呼び出して印字する方法と、印字するイメージデータをコマンド実行の度に送信して印字する２つのパターンがあります。必要に応じてアプリケーションで最適と思われる方法と選択してください。

**《引数》**

mode　実行モード指定
- 30h　展開＋印字
- 31h　展開
- 32h　展開＋印字＋排出
- 33h　登録データの展開＋印字
- 34h　登録データの展開
- 35h　登録データの展開＋印字＋排出

clear　印字バッファークリアー指定
- 30h　印字バッファーをクリアー
- 31h　印字バッファーをクリアーしない

| | |
|---|---|
| width | Ｘ軸イメージ幅 |
| height | Ｙ軸イメージ幅 |
| x | Ｘ軸印字位置 |
| y | Ｙ軸印字位置 |
| address | 登録アドレス |
| imageData | イメージデータ |

**《戻り値》**

受信パケットのバイトストリーム列。

**《例外》**

| 例　外 | 要　因 |
|---|---|
| System.ArgumentOutOfRangeException | 引数の指定が誤っている。 |
| System.ArgumentNullException | 送信イメージデータ指定呼び出しで、イメージデータがnullである。 |
| System.InvalidateOperationException | V31Controlが未オープン。 |
| | リーダライターと通信中のため、本メソッドが実行できない。 |
| System.IO.IOException | 通信デバイスの操作でエラーが発生した。 |
| System.TimeoutException | タイムアウトが発生した。 |

### 3-5-20. V31Control.PrintString() メソッド

文字印字コマンド（ＣＭＤ＝７Ｃｈ）を実行します。

**名前空間　　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public byte[] PrintString(byte mode, byte clear, byte line, byte[] strData ) ;
public byte[] PrintString(ref V31Response res, byte mode, byte clear, byte line,
                          byte[] strData ) ;②
```

**《解説》**

・行指定モードと任意座標指定モードを選択して印字を行います。
・任意座標指定モードでは、指定された任意の印字開始位置から印字します。
・印字位置指定がない場合には、座標(000,000)から展開します。

**《引数》**

mode　実行モード指定
　30h　展開＋印字
　31h　展開
　32h　展開＋印字＋排出

clear　印字バッファークリアー指定
　30h　印字バッファーをクリアー
　31h　印字バッファーをクリアーしない

line　展開開始行
　00h　　任意座標モード
　01h～　行指定モード②
　　01h～14h　標準座標（１～20行）
　　01h～0Dh　左回転座標（１～13行）
　　01h～0Dh　右回転座標（１～13行）

strData　文字データ

**《戻り値》**

受信パケットのバイトストリーム列。

**《例外》**

| 例　外 | 要　因 |
|---|---|
| System.ArgumentOutOfRangeException | 引数の指定が誤っている。 |
| System.ArgumentNullException | 印字データがnullである。 |
| System.InvalidateOperationException | V31Controlが未オープン。 |
| | リーダライターと通信中のため、本メソッドが実行できない。 |
| System.IO.IOException | 通信デバイスの操作でエラーが発生した。 |
| System.TimeoutException | タイムアウトが発生した。 |

### 3-5-21. V31Control.PrintBarcode() メソッド

バーコード印字コマンド（ＣＭＤ＝７Ｅｈ）を実行します。

**名前空間　　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public byte[] PrintBarcode(byte mode, byte clear, byte kind, byte tip, byte width,
                           byte height, byte x, byte y, byte[] barcodeData ) ;
public byte[] PrintBarcode(ref V31Response res, byte mode, byte clear, byte kind,
                           byte tip, byte width, byte height, byte x, byte y,
                           byte[] barcodeData ) ;⚠2
```

**《解説》**

・バーコードの印字を行います。

**《引数》**

mode　実行モード指定
- 30h　展開＋印字
- 31h　展開
- 32h　展開＋印字＋排出

clear　印字バッファークリアー指定
- 30h　印字バッファーをクリアー
- 31h　印字バッファーをクリアーしない

kind　バーコード種類指定
- 30h　ＪＡＮ標準
- 31h　ＪＡＮ短縮
- 32h　ＮＷ－７
- 33h　ＣＯＤＥ３９

tip　バーコード下の解説文字の有無
- 30h　あり
- 31h　なし

| | |
|---|---|
| width | バーコード幅指定 |
| height | バーコード高さ指定 |
| x | 印字Ｘ座標 |
| y | 印字Ｙ座標 |
| barcodeData | バーコードデータ |

**《戻り値》**

受信パケットのバイトストリーム列。

**《例外》**

| 例　外 | 要　因 |
|---|---|
| System.ArgumentOutOfRangeException | 引数の指定が誤っている。 |
| System.ArgumentNullException | バーコードデータがnullである。 |
| System.InvalidateOperationException | V31Controlが未オープン。 |
| | リーダライターと通信中のため、本メソッドが実行できない。 |
| System.IO.IOException | 通信デバイスの操作でエラーが発生した。 |
| System.TimeoutException | タイムアウトが発生した。 |

### 3-5-22. V31Control.Eject() メソッド

カード排出コマンド（ＣＭＤ＝８０ｈ）を実行します。

**名前空間　　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public byte[] Eject() ;
public byte[] Eject(ref V31Response ) ;⚠2
```

**《解説》**

リーダライター内のカードの排出を行います。

**《引数》**

なし。

**《戻り値》**

受信パケットのバイトストリーム列。

**《例外》**

| 例　外 | 要　因 |
|---|---|
| System.InvalidateOperationException | V31Controlが未オープン。 |
| | リーダライターと通信中のため、本メソッドが実行できない。 |
| System.IO.IOException | 通信デバイスの操作でエラーが発生した。 |
| System.TimeoutException | タイムアウトが発生した。 |

### 3-5-23. V31Control.Cleaning() メソッド

クリーニングコマンド（ＣＭＤ＝Ａ０ｈ）を実行します。

**名前空間　　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public byte[] Cleaning() ;
public byte[] Cleaning(ref V31Response ) ;⚠2
```

**《解説》**

・リーダライターにクリーニング動作を要求します。

**《引数》**

なし。

**《戻り値》**

受信パケットのバイトストリーム列。

**《例外》**

| 例　外 | 要　因 |
|---|---|
| System.InvalidateOperationException | V31Controlが未オープン。 |
| | リーダライターと通信中のため、本メソッドが実行できない。 |
| System.IO.IOException | 通信デバイスの操作でエラーが発生した。 |
| System.TimeoutException | タイムアウトが発生した。 |

### 3-5-24. V31Control.SetClock() メソッド

時計設定コマンド（ＣＭＤ＝Ｅ１ｈ）を実行します。

**名前空間　　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public byte[] SetClock(System.DateTime dt ) ;
public byte[] SetClock(ref V31Response, System.DataTime dt ) ;△2
```

**《解説》**

リーダライターの内蔵時計の時刻を設定します。

**《引数》**

dt　設定したい日時

**《戻り値》**

受信パケットのバイトストリーム列。

**《例外》**

| 例　外 | 要　因 |
|---|---|
| System.InvalidateOperationException | V31Controlが未オープン。 |
| | リーダライターと通信中のため、本メソッドが実行できない。 |
| System.IO.IOException | 通信デバイスの操作でエラーが発生した。 |
| System.TimeoutException | タイムアウトが発生した。 |

### 3-5-25. V31Control.GetVersion() メソッド

バージョン取得コマンド（ＣＭＤ＝Ｆ０ｈ）を実行します。

**名前空間　　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public byte[] GetVersion() ;
public byte[] GetVersion(ref V31Response res ) ;⚠2
```

**《解説》**

・リーダライターのソフトウェアバージョンを取得します。

**《引数》**

なし。

**《戻り値》**

受信パケットのバイトストリーム列。

**《例外》**

| 例　外 | 要　因 |
|---|---|
| System.InvalidateOperationException | V31Controlが未オープン。 |
| | リーダライターと通信中のため、本メソッドが実行できない。 |
| System.IO.IOException | 通信デバイスの操作でエラーが発生した。 |
| System.TimeoutException | タイムアウトが発生した。 |

### 3-5-26. V31Control.GetClock() メソッド

現在時刻取得コマンド（ＣＭＤ＝Ｆ１ｈ）を実行します。

**名前空間　　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public byte[] GetClock(ref System.DateTime dt ) ;
public byte[] GetClock(ref V31Response res, ref System.DateTime dt ) ;※2
```

**《解説》**

・リーダライターの内蔵時計の設定値を読み出します。

**《引数》**

dt　リーダライターの内蔵時計の設定値

**《戻り値》**

受信パケットのバイトストリーム列

**《例外》**

| 例　外 | 要　因 |
|---|---|
| System.InvalidateOperationException | V31Controlが未オープン。 |
| | リーダライターと通信中のため、本メソッドが実行できない。 |
| System.IO.IOException | 通信デバイスの操作でエラーが発生した。 |
| System.TimeoutException | タイムアウトが発生した。 |
| System.FormatException | リーダライターからのレスポンスの書式が誤っている。 |

### 3-5-27. V31Control.GetResponse() メソッド

レスポンス再送要求（ＥＮＱ送信）を実行します。

**名前空間　　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

```
public byte[] GetResponse() ;
public byte[] GetResponse(ref V31Response res ) ;⚠2
```

**《解説》**

・リーダライターにＥＮＱを送信し、そのレスポンスを取得します。
・コマンド送信からＥＮＱを連続送信する場合、送信間隔として100ms以上確保してください。

**《引数》**

なし。

**《戻り値》**

受信パケットのバイトストリーム列。

**《例外》**

| 例　外 | 要　因 |
|---|---|
| System.InvalidateOperationException | V31Controlが未オープン。 |
| | リーダライターと通信中のため、本メソッドが実行できない。 |
| System.IO.IOException | 通信デバイスの操作でエラーが発生した。 |
| System.TimeoutException | タイムアウトが発生した。 |

### 3-5-28. V31Control.SendCommand() メソッド

任意のコマンド列を送信します。

**名前空間　SanwaNewtec.DeviceControls**
**アセンブリ　SanwaNewtec.DeviceControls.V31Control.dll**

**《構文》**

［コマンドのみ］
public byte[] SendCommand(byte cmd ) ;
public byte[] GetResponse(ref V31Response res, byte cmd ) ;②
［付加データあり］
public byte[] SendCommand(byte cmd, byte[] data )
public byte[] GetResponse(ref V31Response res, byte cmd, byte[] data ) ; ②

**《解説》**

・リーダライターに任意のコマンドを送信します。また、付加データを指定することも可能です。
・付加データを指定する場合、データ長は、１～２４９バイトの範囲で指定してください。この範囲外で指定すると、例外を送出します。

**《引数》**

cmd　コマンド
data　付加データ

**《戻り値》**

受信パケットのバイトストリーム列

**《例外》**

| 例　外 | 要　因 |
|---|---|
| System.ArgumentOutOfRangeException | 引数の指定が誤っている。 |
| System.ArgumentNullException | 付加データありの場合に、付加データがnullである。 |
| System.InvalidateOperationException | V31Controlが未オープン。 |
| | リーダライターと通信中のため、本メソッドが実行できない。 |
| System.IO.IOException | 通信デバイスの操作でエラーが発生した。 |
| System.TimeoutException | タイムアウトが発生した。 |

本メソッドは、V31Controlがサポートしていない任意のコマンドを送信することができ、リーダライター製品の機能強化にV31Controlのバージョンアップが遅れた場合の一時しのぎ的な使い方、あるいは、過去のリーダライター製品に対して、V31Controlがサポートしていないコマンドを送信するために実装されています。
本メソッドを利用してアプリケーションを開発する場合には、対象となるリーダライターの製品仕様書を熟読の上、慎重に取り扱われますようお願いたします。

## 4. USB デバイスドライバーのインストール方法①②

USB通信を用いてリーダライターの制御を行う場合、アプリケーションを動作させるコンピューターにデバイスドライバーをインストールする必要があります。

デバイスドライバーは、添付のサンプルソースを納めたパッケージ内に格納されています。以下の手順に従い、インストールを行って下さい。

また、添付のデバイスドライバーは、動作確認済みのOS上での動作のみを保証しますので、他のOSが動作するコンピューターに添付のデバイスドライバーをインストールすることは避けて下さい。⑤

### 4-1. 共通

#### 4-1-1. リーダライターの電源が切れていることを確認して、コンピューターの USB ポートに接続

お使いのコンピューター上で、他のアプリケーションが動作していないことを確認して下さい。動作中のアプリケーションがある場合、これを終了して下さい。

リーダライターの電源がOFFになっていることを確認して、コンピューターのUSBポートに接続して下さい。

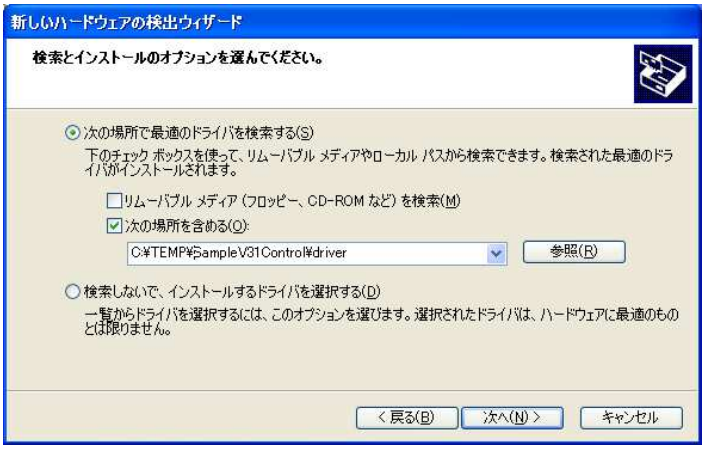
### 4-2. WindowsXP の場合

#### 4-2-1. リーダライターの電源を ON にして、デバイスドライバーをインストール

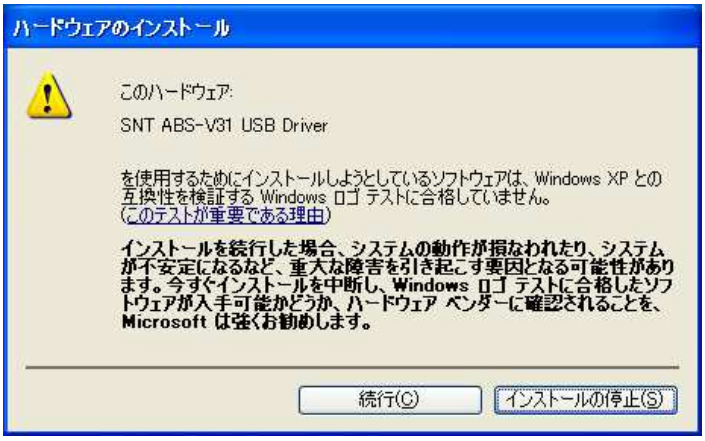
リーダライターの電源をONにして下さい。コンピューターがリーダライターを認識して以下のダイアログが表示されます。


「いいえ、今回は接続しません(T)」にチェックを入れて、「次へ(N)」ボタンをクリックします。


「一覧または特定の場所からインストールする(詳細)(S)」にチェックを入れて、「次へ(N)」ボタンをクリックします。




「次の場所で最適のドライバを検索する(S)」と「次の場所を含める(O)」にチェックを入れて、「参照(R)」ボタンをクリックして、添付のサンプルソースを展開したフォルダを指定します。
デバイスドライバー⚠は、展開したフォルダの「SampleV31Control¥driver」に保存されています。
フォルダを指定したら、「次へ(N)」ボタンをクリックします。




本デバイスドライバーは、Windowsロゴテストを受けておりませんが、動作に問題がないことを確認済みです。
「続行(C)」ボタンをクリックして、インストールを継続して下さい。


デバイスドライバーのインストールが完了するまで、しばらく待ちます。


インストールが完了すると左記のダイアログが表示されます。

ABS-V31のUSBタイプを使用する場合、Windowsの再起動は必要ありません。
これで、ABS-V31のUSBタイプを使用する準備が整いました。
「完了」ボタンをクリックしてデバイスドライバーのインストールを終えて下さい。

## 4-3. Windows7 (32bit 版) ⑥の場合

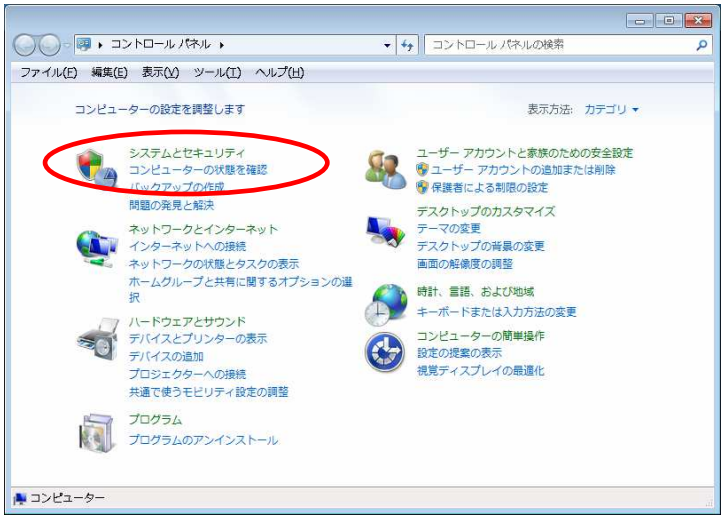
### 4-3-1. リーダライターの電源を ON

リーダライターの電源をONにして下さい。以下のように画面右下にメッセージが表示されます。

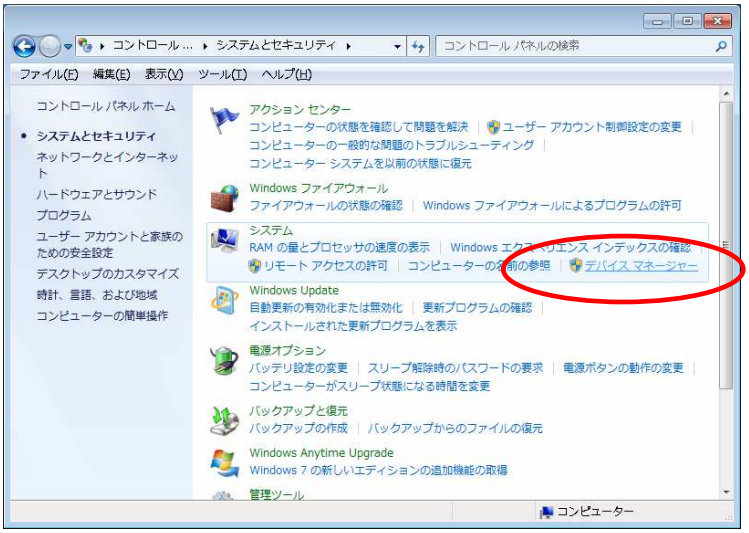

その後、以下のように表示されます。


ここではインストールされません。

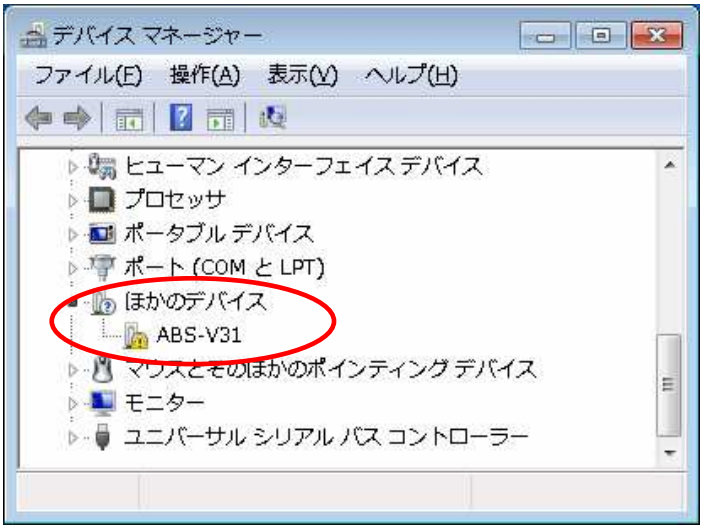
### 4-3-2. デバイスマネージャーを開いてデバイスドライバーをインストール



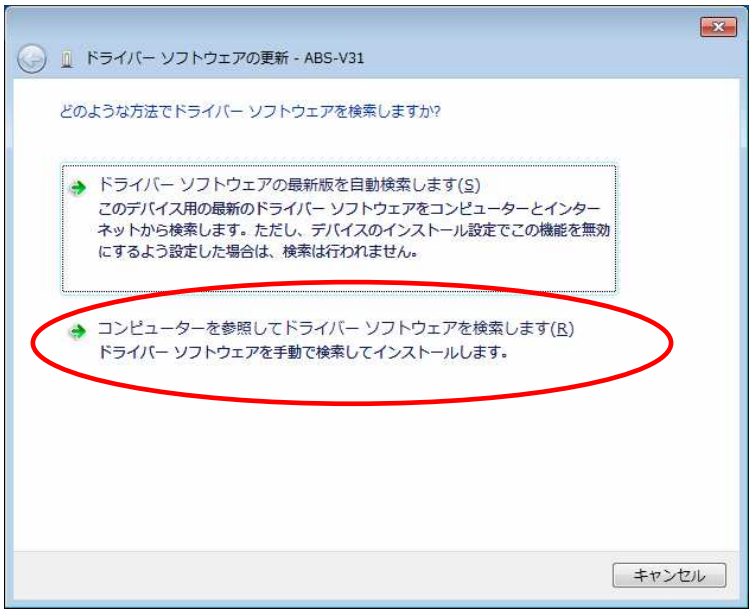

コントロールパネルを開いて、「システムとセキュリティ」を選択します。



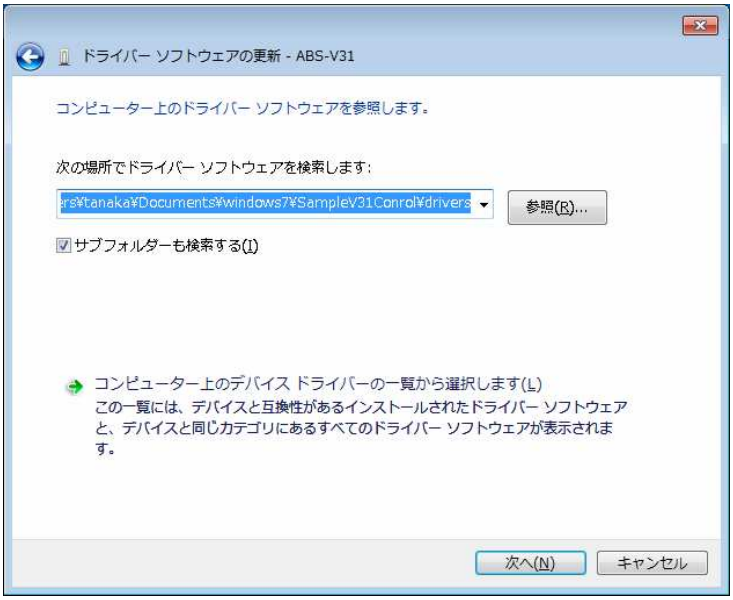

「デバイスマネージャー」を選択します。



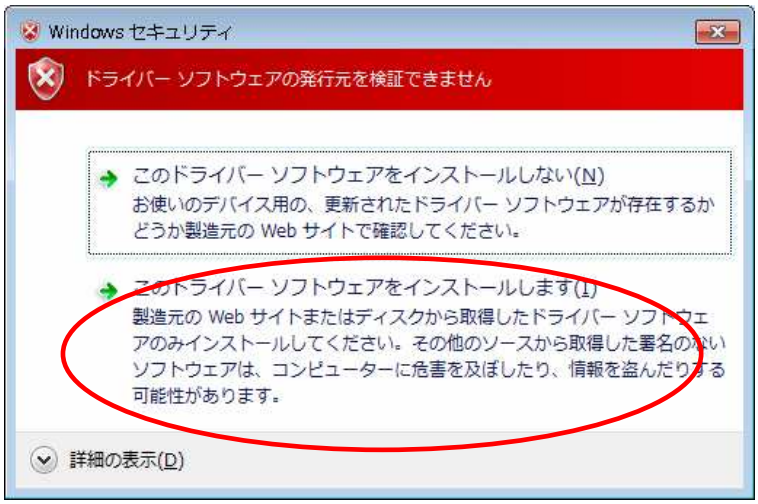

「ほかのデバイス」に「ABS-V31」と表示されます。ここを右クリックして「ドライバーソフトウェアの更新」を選択します。



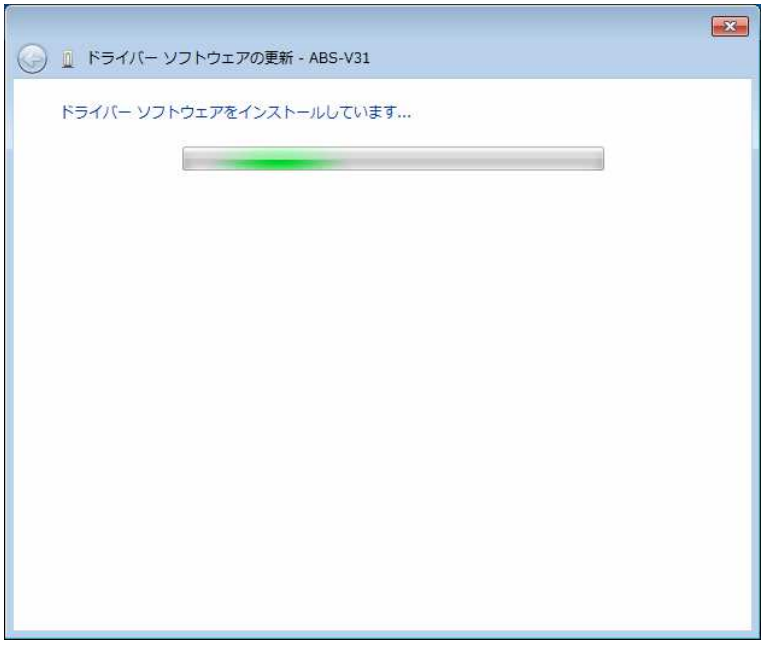

「コンピューターを参照してドライバーソフトウェアを検索します」を選択します。



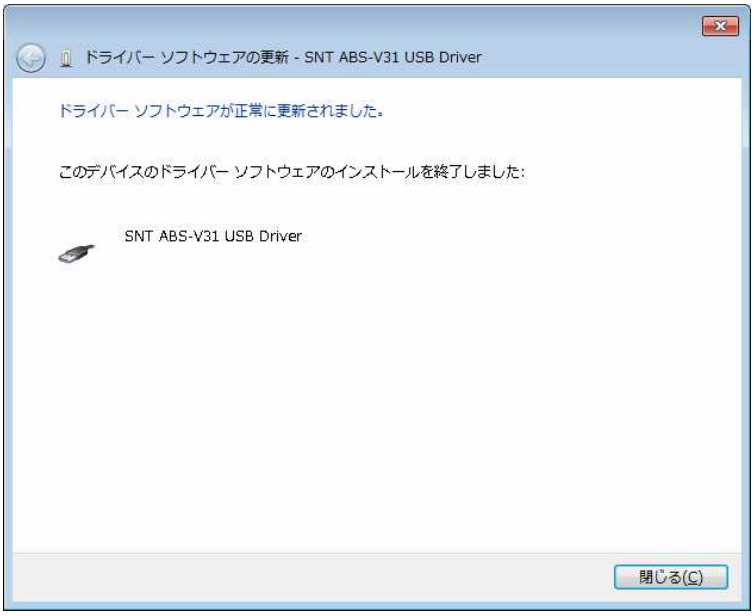

デバイスドライバーは、展開したフォルダの
「SampleV31Control¥driver」
に保存されています。
フォルダを選択したら、「次へ」を選択します。




「このドライバーソフトウェアをインストールします」
を選択します。




インストール中です。




インストールが完了すると左記のダイアログが表示されます。

ABS-V31のUSBタイプを使用する場合、Windowsの再起動は必要ありません。
これで､ABS-V31のUSBタイプを使用する準備が整いました。
「閉じる」ボタンをクリックしてデバイスドライバーのインストールを終えて下さい。

## 4-4. Windows7 (64bit 版)の場合 ⑥

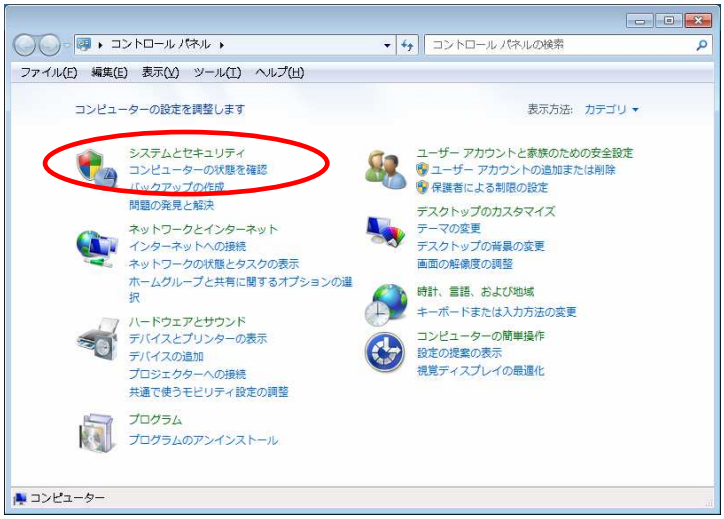
### 4-4-1. リーダライターの電源を ON

リーダライターの電源をONにして下さい。以下のように画面右下にメッセージが表示されます。

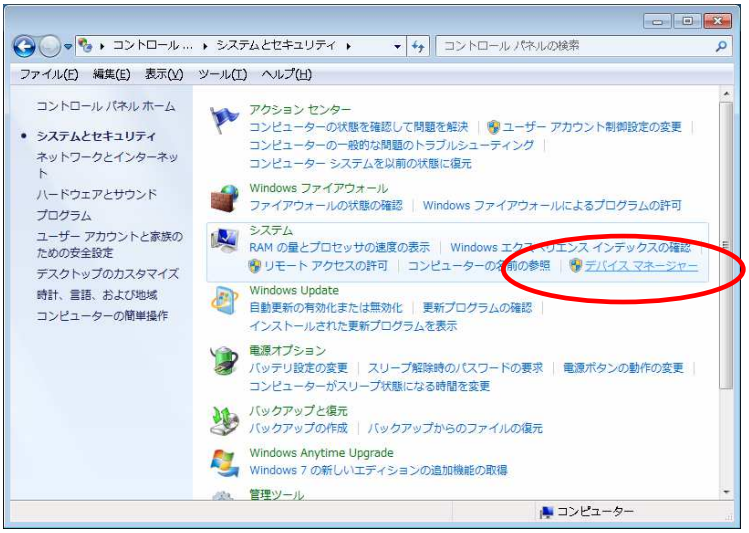

その後、以下のように表示されます。


ここではインストールされません。

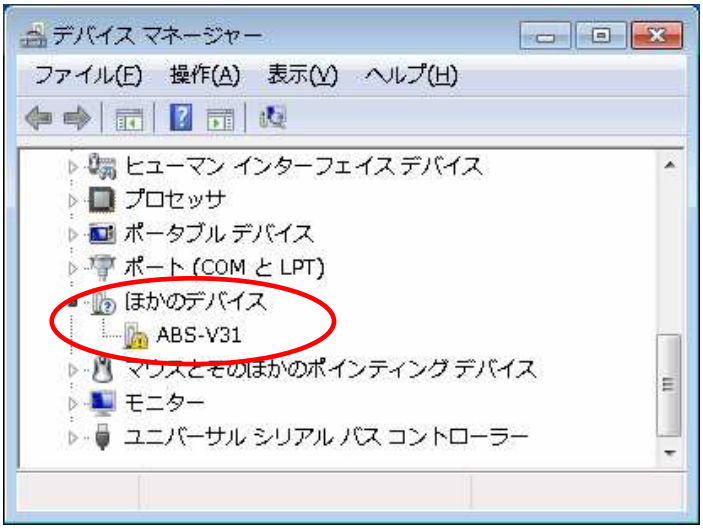
### 4-4-2. デバイスマネージャーを開いてデバイスドライバーをインストール



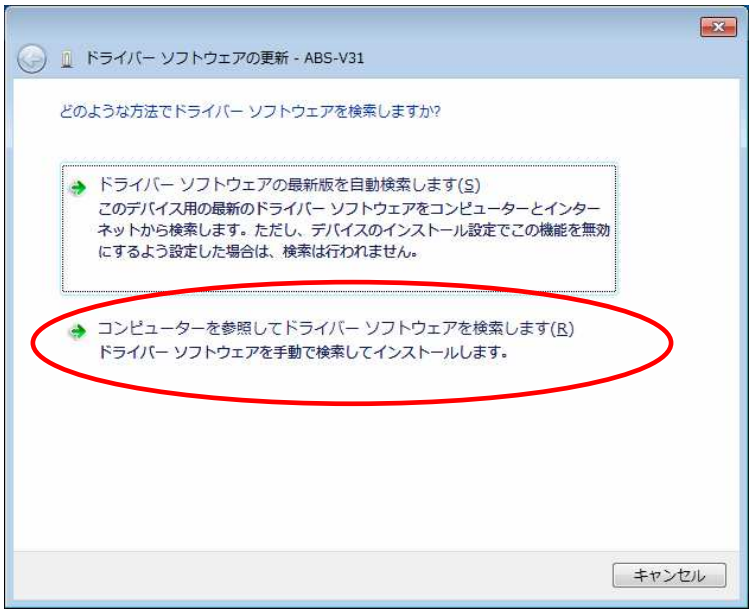

コントロールパネルを開いて、「システムとセキュリティ」を選択します。



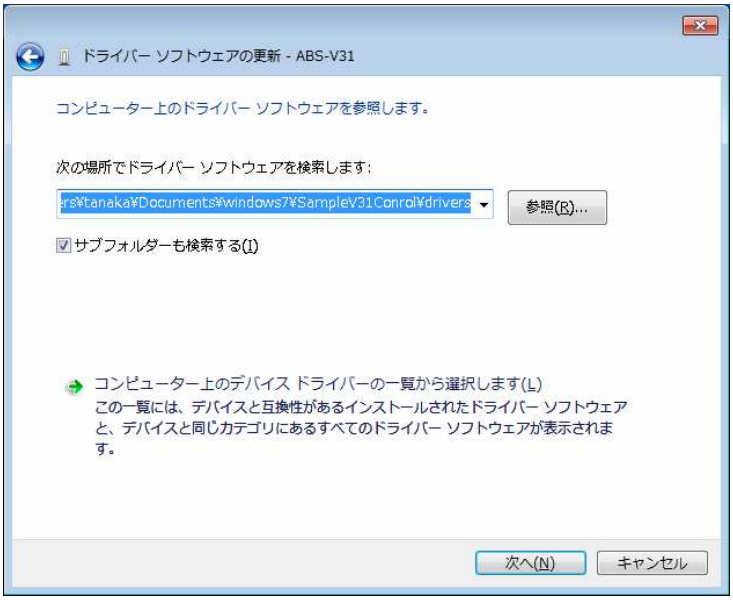

「デバイスマネージャー」を選択します。



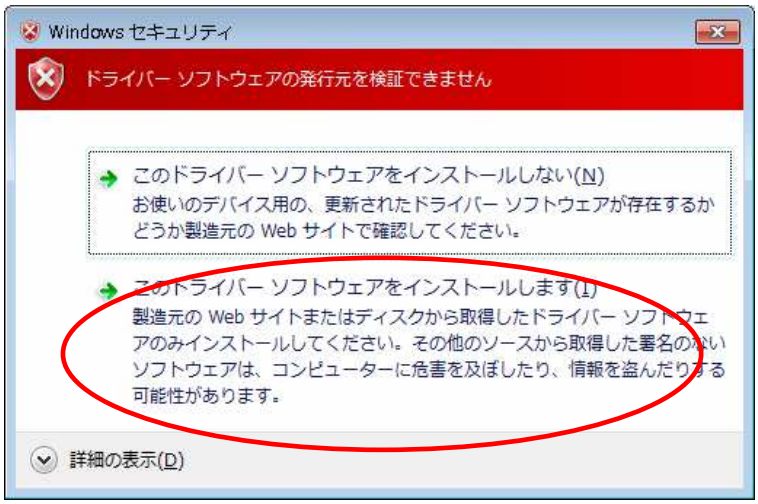

「ほかのデバイス」に「ABS-V31」と表示されます。ここを右クリックして「ドライバーソフトウェアの更新」を選択します。



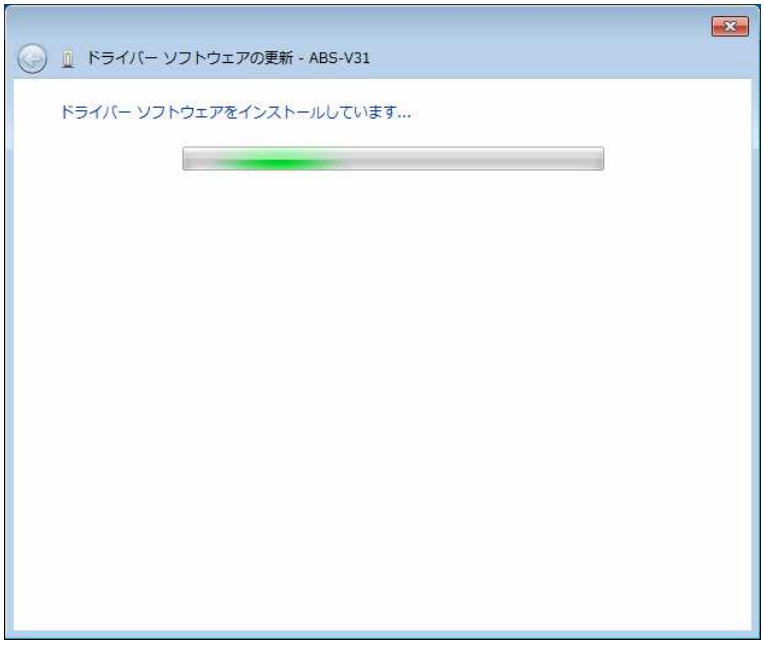

「コンピューターを参照してドライバーソフトウェアを検索します」を選択します。



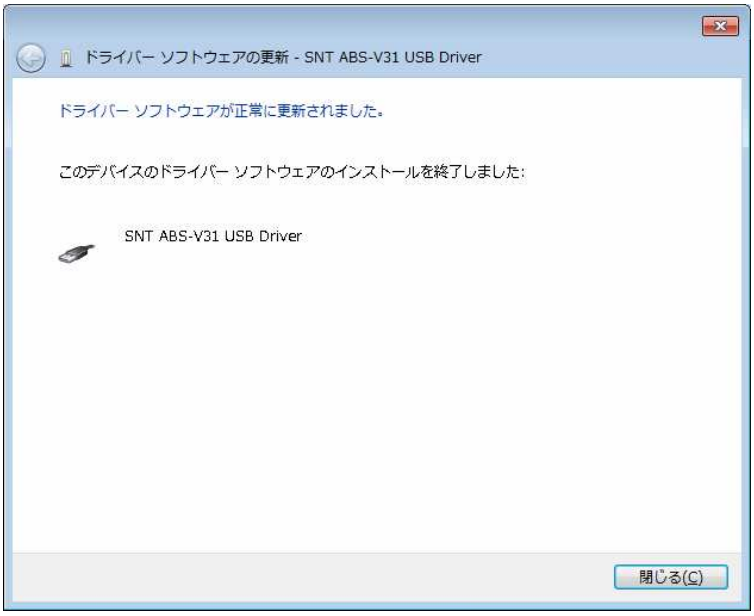

デバイスドライバーは、展開したフォルダの
「SampleV31Control¥driver¥Win7 x64」に保存されています。
フォルダを選択したら、「次へ」を選択します。




「このドライバーソフトウェアをインストールします」
を選択します。




インストール中です。



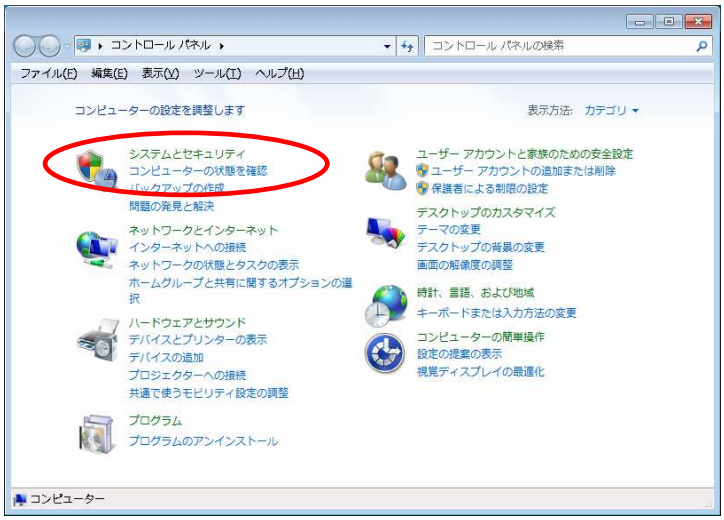

インストールが完了すると左記のダイアログが表示されます。

ABS-V31のUSBタイプを使用する場合、Windowsの再起動は必要ありません。
これで､ABS-V31のUSBタイプを使用する準備が整いました。
「閉じる」ボタンをクリックしてデバイスドライバーのインストールを終えて下さい。

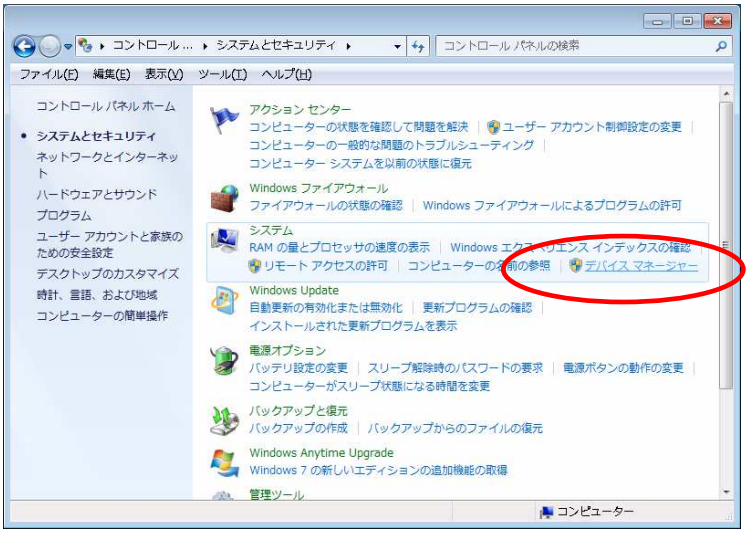
## 5. USB デバイスドライバーのアンインストール方法㉕

### 5-1. Windows7 (32bit 版) ㉖の場合

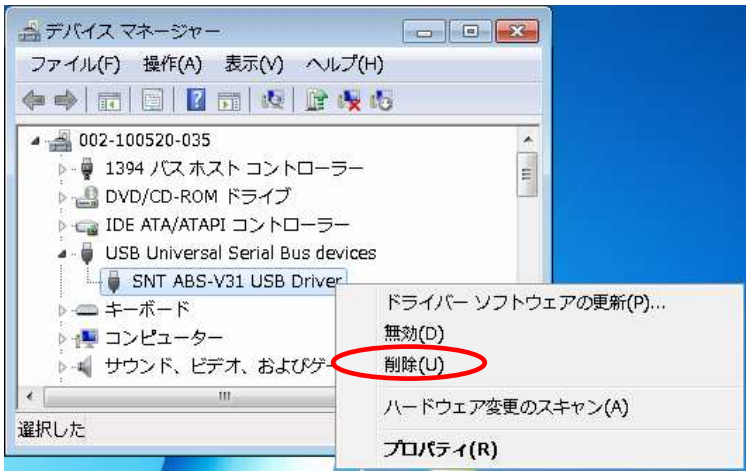
#### 5-1-1. デバイスマネージャーを開いてデバイスドライバーをアンインストール




コントロールパネルを開いて、「システムとセキュリティ」を選択します。



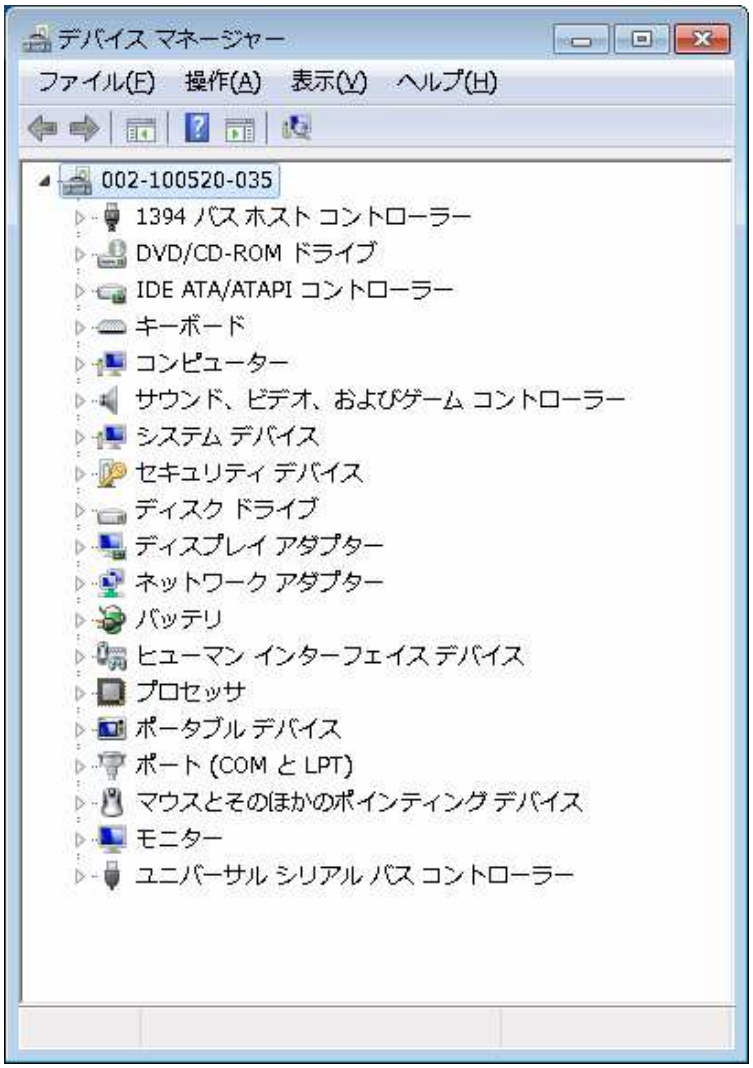

「デバイスマネージャー」を選択します。




「USB Universal Serial Bus devices」
の
「SNT ABS-V31 USB Driver」
を右クリックして
「削除」
を選択します。


「このデバイスのドライバーソフトウェアを削除する」
にチェックをしてOKボタンを押下します。

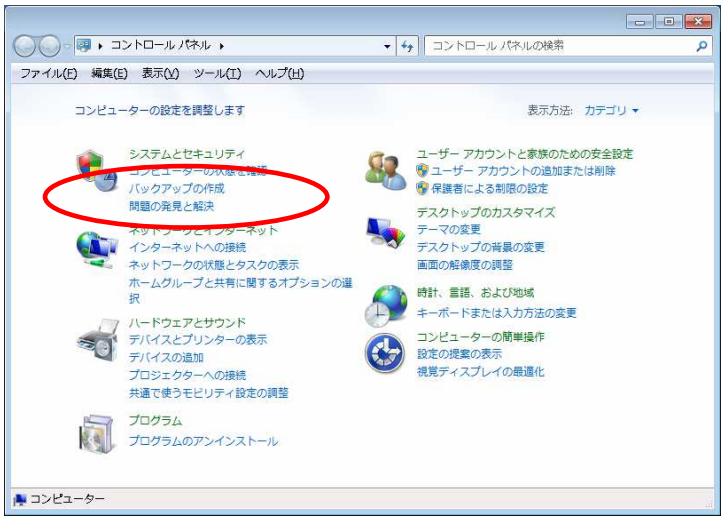

アンインストール中です。



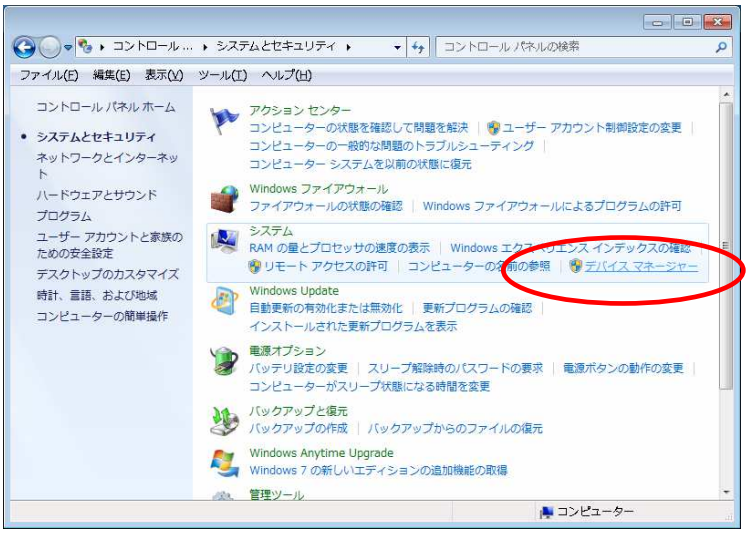

「SNT ABS-V31 USB Driver」が削除されています。

## 5-2. Windows7 (64bit 版)の場合 ⑥

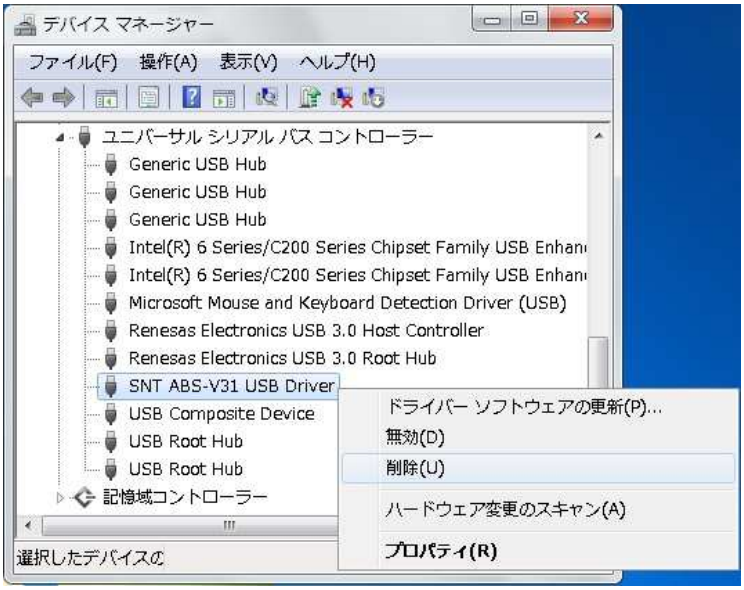
### 5-2-1. デバイスマネージャーを開いてデバイスドライバーをアンインストール




コントロールパネルを開いて、「システムとセキュリティ」を選択します。




「デバイスマネージャー」を選択します。




「ユニバーサルシリアルバスコントローラー」の
「SNT ABS-V31 USB Driver」
を右クリックして
「削除」
を選択します。


「このデバイスのドライバーソフトウェアを削除する」
にチェックをしてOKボタンを押下します。


アンインストール中です。


「SNT ABS-V31 USB Driver」
が削除されています。

－以上－