

Rcb4 ライブラリマニュアル

Ver.2.00 Rev.20120420

はじめに

Rcb4 ライブラリは RCB-4HV の内部コマンドを簡単に呼び出せるように、クラス、定義、メソッドなどをまとめたものです。本マニュアルはある程度のプログラミング経験者を対象としていますので、細かな説明はしていません。あらかじめご理解の上でご利用ください。

使用条件

- ・ Rcb4.dll と Extensions.dll (以降「本ソフトウェア」と呼ぶ) は近藤科学株式会社製品を使うという条件において複製が可能です。
- ・ 本ソフトウェアで生じたいかなる不具合や問題について近藤科学株式会社は一切の責任を持ちません。
- ・ 不特定多数への再配布はできません。
- ・ 本ソフトウェアに関する質問は受け付けておりません。

使用環境

- ・ Windows XP(SP2)/Windows Vista/Windows 7 の各 32bit/64bit
- ・ シリアル USB アダプター、シリアル USB アダプターHS、Dual USB Adapter HS
- ・ RCB-4HV
- ・ Visual Studio 2005、2008、2010 または Visual C# 2005、2008、2010 (基本的に Visual C++、Visual Basic などでも利用可)
- ・ .NET Framework 2.0 を使用しています

ライブラリの構成

Extensions.dll

*マークのついているクラスは通常は直接使いません

| Namespace | | クラス | 説明 |
|------------|-------------|-----------------------|---|
| Extensions | Collections | ByteList | RCB-4 のコマンドをバイト列で扱うためのクラス |
| | | NameObjectCollection* | String 型の Key と Object 型の Value をペアで扱う Dictionary クラス |
| | | QuickSort* | QuickSort アルゴリズムを利用するためのクラス |
| | | StringList* | コマンドなどを文字列で扱うためのクラス |
| | Converter | ByteConverter | バイト列と Int 型の相互変換をするクラス |
| | | StringConverter* | 文字列を Int 型に変換するクラス |
| | IO | FileEx | 指定したパスから指定したファイルを検索する。サブディレクトリも検索する |

Rcb4.dll

| Namespace | クラス | 説明 |
|-----------|---------|---|
| Rcb4 | Command | RCB-4 のコマンドをメソッドとして持つクラス。メソッドはほとんど Static で定義しているので、インスタンス化する必要はない |
| | Config | RCB-4 のシステムスイッチを扱うクラス |
| | Krc | KRC-1/2/3 の低速シリアルコード、拡張低速シリアルコードを定義しているクラス |
| | Rcb4 | RCB-4 の RAM および ROM アドレスを定義しているクラス。また AD/PIO 読み込みやモーション再生などのメソッドも持つ |

Visual Studioでプログラムを作成する

Microsoft 社製の Visual C# 2010 Express または Visual Studio 2010 が使用できます。下記例ではバッテリーの値を読み込んで、画面に表示するプログラムを作成します。

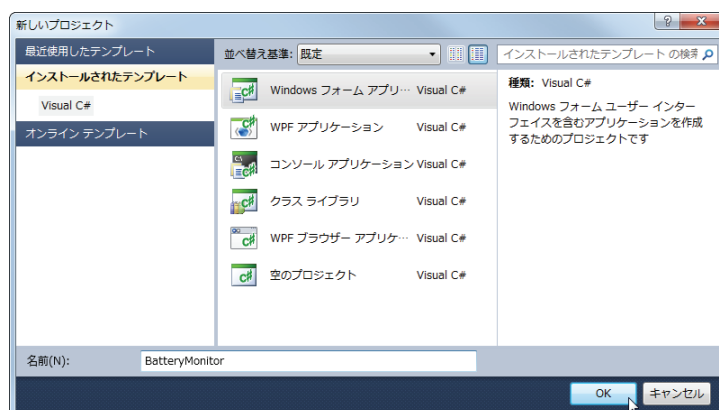
準備するもの

- RCB-4HV
- シリアル USB アダプター、シリアル USB アダプターHS、Dual USB アダプターHS のいずれか

新規フォームの作成

バッテリー電圧を読み込んで表示させるフォーム（ウィンドウ）を新規作成します。

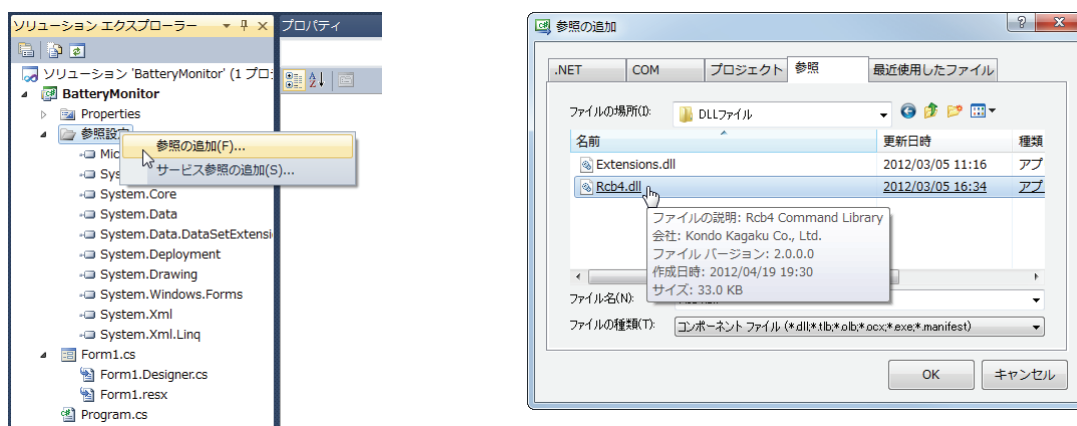
- Visual C# 2010 を起動します
- ファイルメニューから「新しいプロジェクト」を選び、「Windows フォームアプリケーション」を新規作成します。今回は BatteryMonitor という名前にします
- 画面にフォームが表示されます。



ライブラリの追加

Extension.dll と Rcb4.dll をプログラムに追加して、DLL ファイルの持つメソッドが使えるようにします。

- ソリューションエクスプローラーウィンドウを開きます。
- BatteryMonitor プロジェクトの「参照設定」フォルダの上で右クリックし、「参照の追加」を選びます。
- 参照の追加ダイアログで、「参照」タブを選択して、Extensions.dll と Rcb4.dll を 2 つ選択して OK を押します。
- ソリューションエクスプローラーウィンドウの参照設定に Extensions と Rcb4 が追加されます。

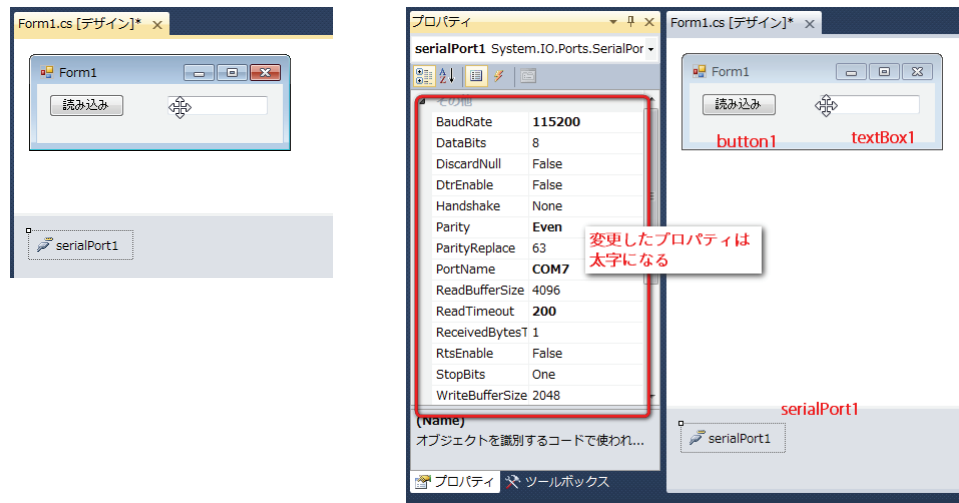


フォームにボタンなどのコントロールを追加する

- 表示されたフォームに以下のコントロールを取り付け、プロパティを設定します

| コントロール | 名前 | プロパティ | プロパティの値 |
|------------|-------------|---------------|---|
| Button | button1 | Text | 読み込み |
| TextBox | textBox1 | 変更するプロパティは特にな | |
| SerialPort | serialPort1 | Baudrate | 115200 (RCB-4 に保存されている通信速度に合わせます。1.25Mbps ならば 1250000 |

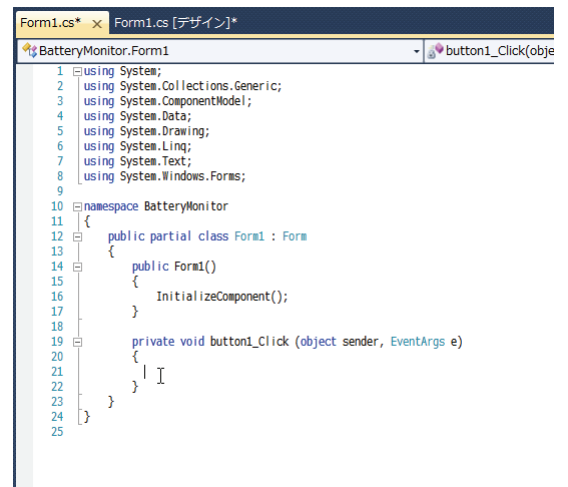
| | | | |
|--|--|-------------|--|
| | | | とします) |
| | | ReadTimeout | 500 (RCB-4 から 500ms 返事がこない場合はエラーとみなします) |
| | | PortName | シリアル USB アダプターの COM ポート名を指定します。例えば COM ポートが COM7 の場合は「COM7」と半角英数字で入力してください |
| | | Parity | Even |



9. デザイン画面で button1 をダブルクリックして、button1_Click イベントを追加します。画面は右図のようにソースコード表示になります。

10. button1_Click にコードを追加して、ボタンを押したときに RCB-4 からデータを読み込むプログラムを書き込みます。プログラムは下の通りになります。ただし緑色の//部分はコメントですので、プログラムとは関係ありません。メモ代わりに使うものですので、特に記載する必要はありません。

```
private void button1_Click (object sender, EventArgs e)
{
    if (serialPort1.IsOpen == false) // シリアルポートが開いていないとき
    {
        return; // なんにもせずにこのイベントを終了
    }
}
```



```
// バッテリーを読み取るコマンドを保存するためのクラスを作成
Extensions.Collections.ByteString cmd = new Extensions.Collections.ByteString ();
// 0 番ポート=バッテリーから AD 値を読み込むコマンドを作成し、cmd クラスに保存
cmd.Bytes = Rcb4.Rcb4.AdRead (0);
// データ受信用の変数を準備する。受信データは AD 変換値 10 ビットデータなので、2 バイトデータ (DATA_L、DATA_H) が戻る
// さらにヘッダ (SIZE CMD) と CHECKSUM が追加されるので受信データ内容は右の 5 バイトになる> SIZE CMD DATA_L DATA_H SUM
byte[] rx = new byte[5];
// 送信する、データの送受信に失敗したら false が返る
if (Rcb4.Command.Synchronize (serialPort1, cmd.Bytes, ref rx) == false)
{
    // データを正しく送受信できなかった場合はエラー表示を行い、イベント終了
    MessageBox.Show ("送受信エラー");
    return;
}
```

```

}
else
{
    // データを正しく送受信できた場合は受け取ったバッテリーデータを電圧に変換して textBox1 へ表示する
    // 受信したバイト列の 2、3 番目 (DATA_L、DATA_H) を int 型に直す
    int adv = Extensions.Converter.ByteConverter.ByteArrayToInt16 (rx[2], rx[3]);
    // バッテリー電圧は安全のため RCB-4 では 1/5 されている。通常は 5V 基準で AD 値 1024 が出力されるので、その値を 5 倍しておく
    float battery = (25.0f / 1024.0f) * ((float)adv);
    // 値を文字列に変換して、textBox1 に表示する
    textBox1.Text = string.Format ("{0} [V]", battery);
}
}

```

11. まだシリアルポートを接続するコードを書いていないので、Form1 コンストラクタに COM ポートをオープンするコードを追加します。

```

-----
public Form1 ()
{
    InitializeComponent ();

    // 起動時 COM ポートを接続（オープン）する
    serialPort1.Open (); // この部分を追加
}
-----

```

12. これでバッテリー電圧の読み取りプログラム作成は完了です。プログラムを実行して、読み取りボタンを押すたびにバッテリー電圧が表示されれば成功です。

まとめ

Rcb4 クラスを使うポイントは、Rcb4.Commandクラスから使いたいコマンドを呼び出し、作成されたコマンドをByteListクラスのBytes変数¹に代入してしまうことです。それから、Rcb4.Command.Synchronizeメソッドに接続済みのシリアルポートと、コマンド、受信用の配列を送ると、正しくコマンドを実行できた場合はtrueを返し、受信用配列にデータを保存します。シリアルポートが接続されていない場合、受信データの配列の大きさが間違っている場合、RCB-4 からの返事が届かなかった場合などはfalseを返します。

RCB-4 から返事がこない場合は、COM ポートのポート名や通信速度について確認してください。

これで Rcb4 ライブラリの簡単な使い方について説明は終わります。より詳しいクラス構成についてはクラスマニュアル (chm ファイル) を参照してください。また、例題として RCB4CommandGeneratorLE のソースコードを参照してください。

¹ 実際はプロパティ