ポケットコンピュータ、ＲＢＩＯ－１接続マニュアル

■ＲＢ－ＰＣＩＦ、簡易ケーブルご使用の方にお願い

１、RB-PCIFケーブルは、ポケットPCと、RBIO-1を接続して、外部制御の学習を行う目的で設計された簡易ケーブルです。業務用途等、一般用途には使用しないでください。この様な用途には、シャープ製純正ケーブル「CE-T800」をご利用ください。

２、RB-PCIFケーブルはRBIO-1専用品です。RBIO-1以外の機器には接続しないでください。他の機器に接続した場合、ポケットPCを破損する恐れがあります。

■概要

説明書はシャープ社ポケットPC、PC-G850VにRBIO-1を接続した状態での動作に基づいています。

ポケットPCのシリアルポート（11ピン端子のSIOモード）と、RBIO-1を接続する事により、ポケットPCからRBIO-1をコントロールする事ができます。接続には、弊社RB-PCIF簡易ケーブルまたはシャープ社の純正ケーブルを使用して行います。制御用のソフトはポケットPC内蔵のC言語を使用します。また一部機能制限がありますが、BASICも利用できます。

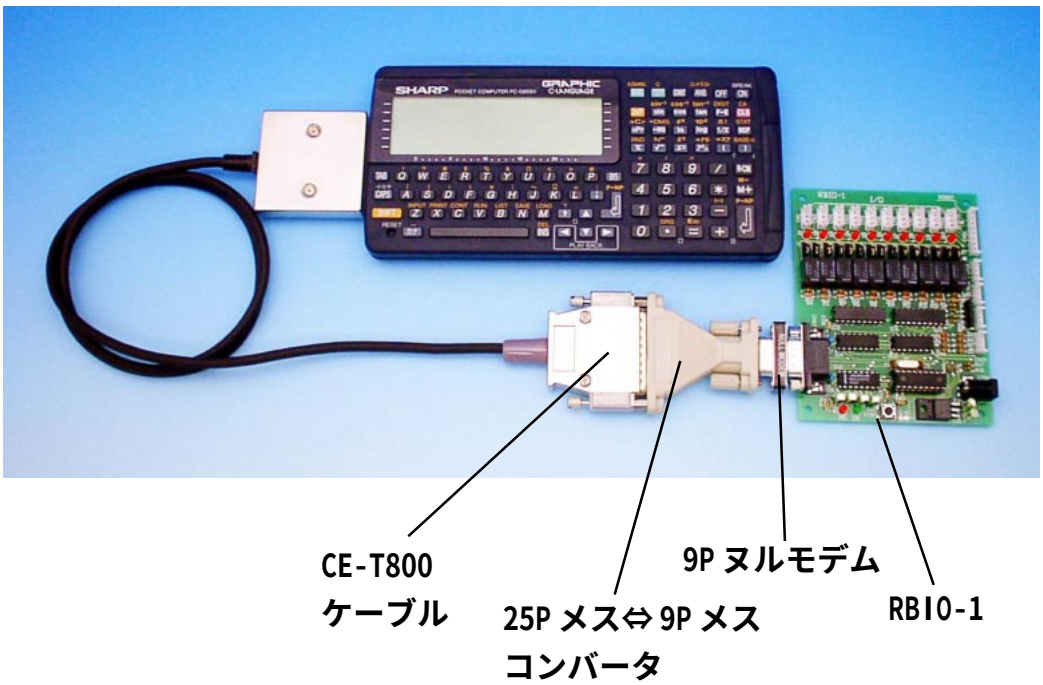
基本的に機械語（アセンブラ）も利用できるはずですが、ポケットPCの説明書に、機械語からシリアルをコントロールする方法が記載されていないため、操作できません。

■接続

●純正ケーブルをご利用場合

ポケットPC側の接続は、ポケットPCの取扱説明書内に説明があります。ケーブルの反対側、機器接続用の232Cコネクタは、25Pオスとなっています。一方、多くのDOS/Vマシンでは9Pコネクタを採用しています。RBIO-1も9Pコネク

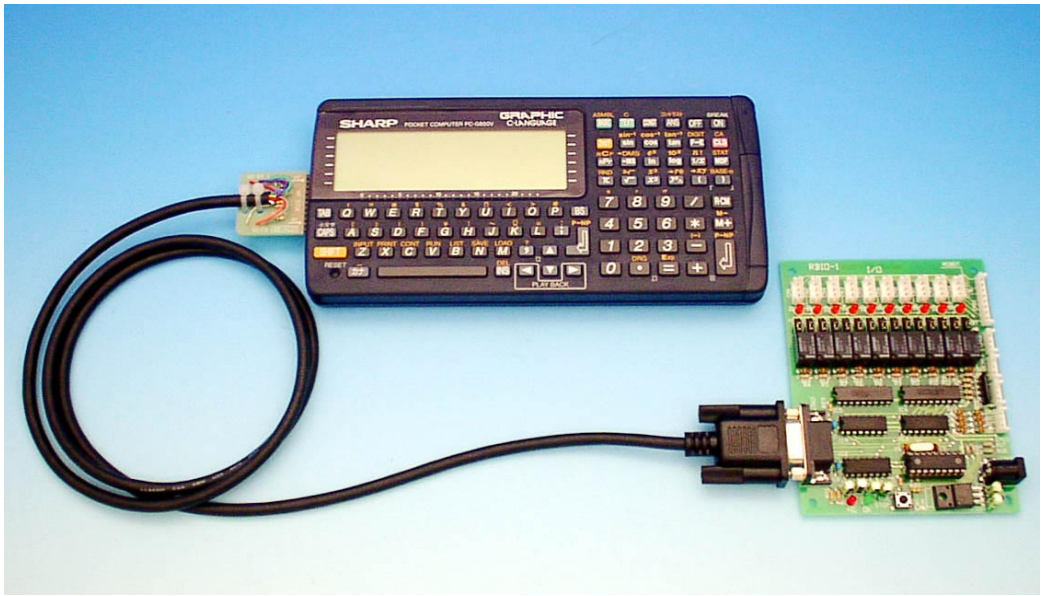
RBIO-1とポケットPCをCE-T800で接続する場合



タとなっています。25Pから9Pには、変換コネクタが利用できます。さらにCE-T800ケーブルのコネクタ配列は、他のWindowsマシンの様な本体を接続するための配列（対DCE用）となっています。モデムなどの受身になる機器（DTE機器）を接続する場合はヌルモデムと呼ばれる送信線と受信線を互いに反転（クロス結線）させるアダプタが必要になります。
RBIO-1もモデムと同じ種類の端子配列になっていますので、純正ケーブルでポケットPCと接続する場合はヌルモデムが必要となります。
25Pから9Pへの変換とヌルモデムの種類により、次の３種の接続法が考えられます。
・25Pメスと25Pオスが付いたヌルモデムに25Pメスから9Pメスに変換するアダプタを直列接続する。
・25Pメスから9Pメスに変換するアダプタに9Pオスと9Pメスが付いたヌルモデムを直列接続する。
・ヌルモデム機能の付いた25Pメスから9Pメスに変換するアダプタを使用する。
＊ヌルモデムはクロスアダプタやリバースアダプタと呼ばれる場合もあります。

●弊社RB-PCIF簡易ケーブルを使用する場合
ポケットPCの取扱説明書を参照して、11ピンコネクタのカバーを外してください。
ケーブル端に基板がある側がポケットPCと接続するコネクタになります。
基板の端にある11本のピンがポケットPCと接続するコネクタになります。等間隔にならんでおり、曲がったピンが無いか確認してください。
接続は基板に部品が実装され、番号が印刷されている面を上にしてポケットPCの11個の穴と基板端にある11本のピンが一致しているか確認の上、挿入してください。

RBIO-1とポケットPCをRB-PCIFで接続する場合



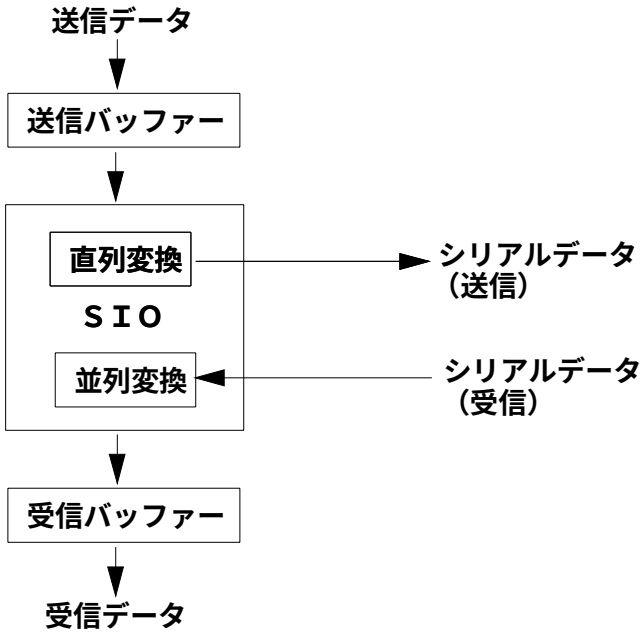
11 ピン全てがポケット PC 側の穴に入れば接続完了です。1 本ずれた状態では、通常挿入できませんが、無理に強行すると端のピンを曲げて挿入されてしまいます。
RBIO-1 側の接続はケーブルの他方、9P メスを RBIO-1 のシリアル端子に接続するだけです。
（専用ケーブルですので、ヌルモデム等のアダプタは必要ありません）

■ポケット PC での設定

シリアル通信では接続機器の通信条件に合わせてポケット PC の初期設定を行う必要があります。RBIO-1 の通信条件は RBIO-1 取り扱い説明書の 3 ページにある「ターミナル設定条件」の表を参照してください。
設定すべき項目は、伝送速度、ビット長、パリティー、フロー制御です。ポケット PC では TEXT キーに続いて S、さらに F を押す事で、しばらくすると、設定状態が表示されます。
ポケット PC では設定項目がアルファベットで表示されます。RBIO-1 との対応は、
伝送速度⇔ baud rate　(9600bps)
ビット長⇔ data bit　(8bit)
パリティー⇔ party　(none)
RBIO-1の設定条件にストップbitはありませんがポケット PC 上の stop bit の項目は 1 に設定してください。
残りの end of line =と end of file =、line number =の項目はそのままで結構です。
最後の flow の項目は RBIO-1 ではフロー制御の項目に該当します。ポケット PC では RS/CS に設定してください。

■シリアルでコントロールしよう

ポケット PC のシリアル信号（以下 SIO）のブロックは下の図の様になります。
ポケット PC からの送信では、アプリケーションの送信データ→送信バッファー→ SIO デバイスによる直列変換→ケーブルの順に送られて行き、最後に相手機器に到着します。
一方、相手機器がポケット PC に対して送信されて来たデータは、送信とは逆の手順により、ケーブル→ SIO デバイスによる並列変換→受信バッファー→アプリケーションの順に伝わって行きます。
ここでのアプリケーションは BASIC や C で書か



れたプログラムです。
直列変換は、バイトデータを順次ビットデータに分解して送り出す操作です。これにより、１本の電線でバイトデータを送信する事ができます。一方並列変換は受信用の変換で順番に送られてきたビットデータを元のバイトデータに変換する作業を行います。
ここから先はRBIO-1対ポケットPCに限定した説明です。と言うのも、相手機器の仕様によって通信内容や振る舞いが異なるからです。
細かい理屈は別にして、データ（文字または文字列）は指定の手順に従ってBASIC文を記述する事でシリアルデータに変換されて相手機器に送る事が出来ます。
RBIO-1のリレーをコントロールするには、SIOデバイスに対して制御文字を送信するれば良い事になります。
ポケットPCのBASICから

```
10:OPEN “COM1:”
20:PRINT #1,“ ”
30:PRINT #1,“PCT0005”
```

（行番号の後の:は入力不要です LISTを出すと勝手に付きます）
20番と30番にPRINT文がありますが、30番が本来のRBIO-1に対する動作指示です。20番は「おまじない」です。ここでは、そのまま入力してください。
（「おまじない」の解説は、この項の最後にあります）
RBIO-1と接続が完了していれば、RUNしてください。
RBIO-1の一番左のリレーが0.5秒間ONし、リレー動作確認用のLEDが0.5秒間点灯したはずです。
一瞬RBIO-1のエラーが点灯する場合もありますが、動作すればOKです。動かない場合は接続およびBASIC文を点検してください。
10番のステートメントは、SIOを使用する旨を宣言しています
30番でRBIO-1に指令を送り出していますが、指令の中身は”” の中にあるPCT0005です。PCT0は0番リレーに対する、タイマー付きのON指令です。
続く005でON時間を0.5秒として指示しています。
PRINT#1は画面に出力するPRINTと同じですが、出力先が#1デバイス、ここでは10番で宣言したSIOになる点が異なります。
うまく動作した場合は、BASICを少し改造してみましょう。
追加として..

```
40:FOR I=0 TO 2000
50:NEXT
60:GOTO 30
```

40番と50番は時間つぶしです、この2行を実行するのに2秒程度かかります。60番のGOTOでRBIO-1にON指令を出していた行に飛びます。
RUNするとRBIO-1に対して、約2秒間隔で0.5秒間ONする指令が永久に送信される事になります。結果的にRBIO-1のリレーがカチカチON/OFF動作をする事になります。
しばらく、このまま、走らせておいてください。
20回位ON/OFF動作した所で動作が停止してしまったと思います。
この現象は、シリアル受信用のバッファーが満杯になってしまったために起こる現象です。
もう少し詳しく見ていきましょう。
30番のPRINT#1文が送り出した指令文字はRBIO-1に届きます。RBIO-1はコマンドが正しい場合、OKの文字をポケットPCに対して送り返します。OKの文字はポケットPCのSIOで受信され、受信バッファーに溜まります。
（前ページの図では下半分の部分です）
ところが、実験に使用したBASIC文内には、溜まった文字（ここではOKの文字）を読み出す命令を記述していません。このままでは、いつか受信用のバッファーが満杯になってしまいます。
ここに何らかの工夫が無いと、相手が送信したデータが、受け取る前に消えて無くなります。
そこで導入されたのがフロー制御です。
ポケットPCのSIO設定でflow =の項目をRS/CSに設定しましたが、このRS/CSは制御専用の線を使って受信バッファーが溢れそうになると相手側の送信を停止させて、溢れるのを防ぐ機構です。
専用の線（ハードウエア）を利用するため、フロー＝ハードと表現される場合もあります。
シリアルデータを送受信するだけなら2本の電線があれば、足りますが、フロー制御が入るため、もう２本の電線が必要になります。
実際、ポケットPCのシリアルコネクタには４本

の信号線が出ています（共通線があるのでもう少しコネクタのピン数は多い）
さて、RBIO-1のリレーON/OFFが停止したのはフロー制御とどうかかわっているのでしょうか。実は、RBIO-1は、送信データ（この場合OKの文字）を送り出せずに、フロー制御が解除されるのを待ち続けているのです。
ここで、さらに、実験を行います。
走っているBASICを停止させてください。そして、再度RUNしてください。
しばらく時間が掛かると思いますが、リレーのON/OFFが再開されます。
これはBASICがRUN命令を実行する際にバッファーの中を空にするためで、空きになればフロー制御が解除され、RBIO-1の送信が再開、動作も再開される事になります。
今回の実験の様に、RBIO-1に送信した命令の結果を調べる必要のないアプリケーションでは、フロー制御はじゃまな存在になります。
確実な動作を保障するには,RBIO-1が応答する文字を受信して検査すべきですが、実際には送るだけのコントロールでも、さほど問題は発生しません。
フロー制御を止めるには二つの方法があります。
１：ポケットPC側で、フローを切る。
ポケットPCのSIO設定でflow =の項目をnoneに設定する事で、受信バッファーが溢れても、相手を送信禁止にしません。
受信バッファーから溢れた文字は捨てられる事になります。先の実験方法でもflow =noneの効果を確かめる事ができます。
２：RBIO-1の返答を止める。
RBIO-1は通常状態では、命令に対して必ず返事を送信します。
これを禁止するには、RBIO-1使用説明書の10ページ、特殊モードを利用してS4パラメータの内容を1に書き換える事で行う事ができます。
ただ、設定にはターミナルが必要になります。
Windowsマシンをお持ちの方はRBIO-1と繋いで設定できますが、環境をお持ちで無い場合は、次の章、簡易ターミナルを作るを参照してターミナルを製作して、RBIO-1の設定を行ってください。

それでは、フロー制御を止めたり、RBIO-1を設定替えせずに、利用する方法が無いのでしょうか。
正しいやり方は、RBIO-1が送り出す文字列をBASIC文内で読み出す事です。
そのためにBASICにはINPUT（INPUT#）やLNINPUT#の命令があります。
これを使えば、受信バッファーに溜まった文字を読み出す事ができそうです。
受信文字を確認するためPRINT文も一緒に挿入する事にして..

```
32:LNINPUT#1,D$
34:PRINT D$
36:LNINPUT#1,D$
38:PRINT D$
```

を追加してください。
２回入力を追加したのは、RBIO-1は一つの命令に対して２行分の返答を行っているためです。
RUNすると、今度は動き続ける事が確認できると思います。また画面には１行毎にOKの文字が出力されていると思います。
一見成功したかに見えますが、ちょっと問題を含んでいます。
動作中にRBIO-1のSTOPボタンを押して見てください。
STOPボタンは、動作中の全てのリレーをOFFにして「RBIO-1 I/O Control Ver1.0」をシリアル線に送り出す、緊急停止ボタンです。
もしリレーがONした直後なら、STOPボタンでリレーがOFFするのが判ります。そして、実行中の画面にはRBIO-1の文字が出ます。
さらにSTOPを押してください。
元々実行中はOKの文字が表示され、動作の度にスクロールされて消えて行きますが、この中に混じるRBIO-1の文字が出るタイミングがSTOPを押す度に遅れて行くのが判ります。
１０回程度STOPを押すと停止してしまいます。
ここで起こっている現象を整理すると以下の様になります。
BASICの30番で0.5秒間ONにする指令をRBIO-1に送る
RBIO-1は２行の返答を返す
追加した32番と36番の入力文がRBIO-1の返答２行を読み取る。
ここまでは問題ありません。返答２行に対して、読み出し２行。
しかしRBIO-1でSTOPボタンを押すと、
命令が来なくても、RBIO-1... の文字列を送り

出します。
この文字列は予定外の行になります。
すなわち、この瞬間だけ、３行の返事に対して２行しか読み出しが行われない事になります。
１行分が読み出してもらえずに受信バッファーに残ります。しかしBASICの実行が次の指令タイミングになると残った１行と新しい２行の内の最初の１行を読み出す事になります。
このまま、永久に１行分が順次残り続けます。
残った１行が読まれるのは次の操作指令の時ですので、この時点で２秒の遅れが出た事になります。
さらに、STOPを押し重ねると押す度に遅れて読まれる行が増え続ける事になります。
これは、倉庫に商品を積み上げると同じで、売れる量より、仕入れる量が上回ると、倉庫に在庫の山が増え続け、売り出す商品の日付が段々古くなってしまうと同じ理屈です。
最終的に停止したのは、受信バッファー内に余分な行がたまったため、フロー制御が働いて、RBIO-1が送信を止めたためです。
それでは、余分なLNINPUT#1の行を挿入すればどうでしょうか。
実験すれば判ると思いますがすぐに停止してしまいます。
今度は受信２行に対して、読み出しが3行になり、BASICは３行目のLNINPUT#1に文字が到着するのを待ち続ける事になります。
しかし、RBIO-1には指令を送らない限り、返答が得られないため、この時点で永久停止になります。
さらに、読み取りは行単位の制約があります。
相手が必ず、行としての要件を満たしていないと、BASICが行として認識できない事になります。
結論として、BASICではSIOの送信と受信の双方向を同時に行うプログラムは、かなり制限を受ける事になります。
これは、ポケットPCのBASIC仕様上の問題なので簡単には回避する方法がありません。
通常のPC用のBASICには、シリアル受信バッファーに溜まった文字の個数を調べる命令や１文字単位の読み出し機能があり、溜まった事を調べた後に読み出し操作が行えるので、この様なずれが発生する事を避ける事ができます。
もともとBASICは対人間用として会話形式の入出力を提供する様な仕様で製作された言語です。入力を人間が行う場合は、表示画面に対して柔軟に操作できるため、問題は起こりません。しかし、相手が融通の利かない機械の場合は、相当な配慮を必要とします。

＊おまじないに付いて
ポケットPCのSIOは他の11ピン用のデバイス（PIOなど）と共通の端子に配置されています。
BASICなどでSIOのOPEN命令が入って、初めて、端子に割り当てられた機能が有効になります。
しかし、SIO機能が働き出す前に、すでにRBIO-1が接続され、電源も投入されていますので、RBIO-1には不要な文字が受信されています。
一回目の改行は不要な文字に続けて改行を送る事で、RBIO-1が受信している文字（この場合は不正な文字）を命令として実行させています。
当然意味の無い文字が命令にされるのですから、エラーとなります。
初回がエラーですが、次の命令から正常に処理できる様になります。
この様な操作はOPEN直後の一回だけ行えば実行中は必要ありません。またOPENした後からRBIO-1の電源を投入する場合は、不正な文字を受信しないため、おまじないは不要になります。

■簡易ターミナルを作る

RBIO-1はPCの拡張として制御する以外に、直接、人間が操作した場合でも、なるべく分かりやすいコマンド体系としています。
ポケットPCから、RBIO-1を人手で操作するには、
１：キー入力がればそれを読み出し、SIOに送り出す。
２：SIOに受信文字が到着すればそれを読み出し、画面上に表示する。
３：以上の操作を停止させる事なく、永久に続ける。
この様な機能をダムターミナル（又は単にターミナル）と呼びます。
Windowsをお持ちの方なら、ハイパーターミナルを使った事が有ると思います。
ポケットPCで作るターミナルはそれを極限まで簡単にした物です。
要求は非常に単純ですが、前の章で説明した様に、BASICではSIOから受信した時のみ読み出す操作が困難なため、単純には製作できませ

ん。
ここではC言語を使って製作する事にします。C言語には、キーが押された事を調べるkbhitとSIOの受信バッファーに文字が入っているかを調べるfeofの命令（関数）があります。
これらを使って、データが用意されている時のみキー入力や、SIOの読み出しを行えば、入力停止にならないプログラムを作る事ができます。
本来ならkbhit()でキー押しを検出した場合のみgetchrを使って押されたキーのコードを１文字づつ取得するのが理想です。しかし、PC-G850Vでは関数がうまく動作せず、一行まとめてキー入力を行うgetsを使用しました。この部分はBASICのINKEY$の考え方の方がすっきりしています。

```
10 main()
20 {
30   int *f;
40   char c[50];
50   f=fopen("stdaux1","a+");
60   for(;;) {
70     if (!feof(f)) {
80       printf("%c",getc(f));
90     }
100     if (kbhit()!=0) {
110       gets(c);
120       fprintf(f,"%s\n",c);
130       while(kbhit());
140     }
150   }
160   fclose(f);
170 }
```

番号50はSIOを使用する宣言です。160番でSIOの使用終了を行っていますが、それより前に永久ﾙｰﾌﾟ（60番のfor文）があるため、ここが実行される事はありません。130番は行最終の改行キーが放されるのを待つループです。
１行まとめて入力する形式のため、一度入力を始めると、最終の改行キーを押すまで、RBIO-1からのメッセージを表示できません。
入力中は画面にカーソルが出るので、判断できます。
コンパイルしてエラーが出なければGで実効させてください。
最初は入力状態になって画面上にカーソルが出ると思います。
（実行用に押したGキーを放す前にプログラムが動作してしまうためで、この時点で最初のkbhit()はキーが押された物として入力に入っています。Gを押す時間が短いと入力に入らない場合もあります）
最初は改行を押して前章の「おまじない」に相当する作業を行ってください。当然ながら画面にはERRORが表示されると思います。
改行入力で、キー入力（行入力）からも抜け出せたため、画面上のカーソルが消えていると思います。
この状態でRBIO-1のSTOPボタンを押してみてください。画面にはRBIO-1 I/O...の文字が表示されます。
後は色々な指令をRBIO-1に送信してみてください。PCDJUと指令を送るとリレーが一つ置きに点灯します。
永久にループするソフトですので、停止はポケットPCのBREAKキーで行ってください。

●RBIO-1の設定パラメータを書き換える
前のフロー制御を回避するための方法にある様にRBIO-1には動作時の振る舞いを設定できる機能があります。
実行前にRBIO-1取扱説明書の9ページ以降、特殊モードの項を参照してください。
例として、RBIO-1のSTOPボタンを押した際に出力される文字列「RBIO-1 I/O C..」を出力しない様にしてみましょう。
１：製作した簡易ターミナルを起動します。
２：一度RBIO-1の電源を抜きます。
３：RBIO-1のSTOPボタンを押しながら、電源を挿入します。
４：RBIO-1上の赤（NG）と緑（OK）の表示が同時に点灯すれば準備完了です。
もし、緑だけ点灯する様なら、再度２からやり直してください。
５：ポケットPCで「SO=1」をキー入力して、改行キーを押します。
６：RBIO-1の緑ランプだけ点灯すれば、受け付け完了です。赤が点灯してしまった場合は、RBIO-1のSTOPボタンを押して赤、緑ランプの同時点灯状態にした後、５番からやり直してください。
７：RBIO-1の緑ランプが点灯している状態でSTOPボタンを押す事で設定が記憶されます。

８：RBIO-1の電源を一度抜き、再度入れてください。
以上で設定が完了しました。
RBIO-1のSTOPボタンを押しても、ポケットPC上には、何も表示されなくなったはずです。
ためしに、PCとタイプして改行ボタンを押してください。
OKの文字がポケットPCに表示されるはずです。
STOPを押した場合に出力されるメッセージを元の様に出力させるには、前ページの操作を行い、「S0=0」を設定してください。

■おまけのデモ
端から順に動作位置が移動します。
SIOの送受信を停止しないループ上で行っていますので、多少複雑になっています。
「おまじない」を記述していませんので、走らせた直後の１回目だけエラーが出るだけで以後は正常動作します。
（すぐに次の指令が送られるため、ERRORが出てもほとんど影響ありません）
永久にループするソフトですので、停止はポトットPCのBREAKキーで行ってください。
おまけのプログラムではRBIO-1からの返答は、単に画面に表示しているだけです。
しかし、ループを停止させずに読み取る考え方は生かされており、RBIO-1からの返答の読み出しは、受信バッファーに溜まった事を検査して行っています。
意地悪をして、RBIO-1のSTOPボタンを押しても、動作中のリレーはOFFしますが、返送された文字を画面に表示し、動作自体は影響なく継続されます。
より、完成度の高いソフトにするためには、RBIO-1から受信した文字を文字列として組み立て、OKの文字が正しく送られてきているかを検査する必要があります。

```
10 main()
20 {
30   int a,b,x,y1,y2,*f;
40   f=fopen("stdaux1","a+");
50   for(b=0,x=0;;x++) {
60     if (!feof(f)) {
70       a=getc(f);
80       printf("%c",a);
90     }
100     if (x>100) {
110       y1=0; y2=0;
120       if (b>=5)
130         y1=1<<(b-5);
140         else y2=1<<b;
150         fprintf(f,"%s%c%c\n","PCD",
y1+64,y2+64);
160       x=0; b++;
170       if (b>9)
180         b=0;
190     }
200   }
210   fclose(f);
220 }
```


＊当マニュアルの補足等は下記ＵＲＬにて公開します
http://www.kyohritsu.com/CATALOG/KIT_CTRL/rbio1.html

本製品のお問い合わせは
〒556-0004　大阪市浪速区日本橋西２－５－１
共立電子産業株式会社、ケイシーズ担当までお願いします
TEL (06)6644-0021
FAX (06)6644-0824
Email:keiseeds@kyohritsu.com

＊KEISEEDSの新製品ﾆｭｰｽは共立電子のﾎｰﾑﾍﾟｰｼﾞ
「http://www.kyohritsu.com」でご覧いただけます。