LibreOffice Calc BASIC による配管圧力損失ソフトウェア プログラム説明書

1.　はじめに

LibreOffice Calc の機能のなかで、最も強力な機能のがマクロプログラム　LibreOffice BASIC です。
LibreOffice BASIC は、数あるプログラム言語のひとつで、Microsoft の VBA と同等のものです。
私は古い教育を受けたので、最初に教わった言語は FORTRAN でした。
FORTRAN は、科学技術計算に特化したプログラム言語で、手続きを自動化して決められた計算を実行していくプログラムを作成するのに適しています。FORTRAN は、30 年前にもすぐになくなるといわれていましたが、過去の遺産もあるし、科学技術計算は記述しやすいので、今でもしっかり生き残っています。
私は、FORTRAN のコンパイラがなかなか無料にならなかったので、その後は、同じような計算が比較的簡単にできる C 言語に乗り換えましたが、いまでも FORTRAN と同じような作法で C（または C++）プログラムを書いています。FORTRAN は、いまでは GCC コンパイラに取り入れられており、無償で使用できるようになりました。もう少し早く GCC に取り入れられたら、いまでも FORTRAN を使っていたかもしれません。

さて、FORTRAN や C のような原始的なプログラム言語を使用してプログラミングする際に、最も面倒なのは、自分の使っている変数の数値の一時保存場所の指定方法です。原始的なプログラム言語では、その都度プログラム上で定義しなければならず、また、それらの数値を見るためには記述を追加しなければ確認できないのが面倒です。
ところが、LibreOffice BASIC や Excel VBA のような高度な言語は、スプレッドシート上のセルの位置を直接指定してそこに書込めるので、確認が容易です。
この機能により、定数の値を、スプレッドシート上から読み込んだり、計算結果をスプレッドシート上に書き込んだりできます。このため、従来の言語では直接見えなかった内部の一時変数の値等を直接見られるように簡単にプログラミングでき、Calc 本来の機能により、体裁を整えたレポートとして出力することもできます。
FORTRAN や C のような原始的な言語よりも劣るのは実行速度で、便利な反面、計算時間が数万倍とか数百万倍になると考えるほうがいいです。
計算のステップ数が多ければ、FORTRAN や C でしっかりプログラムを作り、計算のステップ数がすくなければ LibreOffice BASIC でプログラムするといった使い分けが必要です。
私が別のサイトで紹介している多自由度バスレフ型スピーカーシステムのシミュレーション等は、C++（使っているのは C 言語の文法のみ）で記述しています。計算ステップ数が多いので、LibreOffice BASIC を使用しても、遅すぎて実用にはならないでしょう。

というわけで、ここでは、計算そのものは簡単ですが、手計算では大変で、また、C や FORTRAN では、コーディングするのが面倒な圧力損失計算プログラムを紹介します。

2. 使用許諾条件

このプログラムは、General Public License (GPL)のライセンス形態で配布します。GPL については、説明を省略しますので、ウェブサイトなどで調べてください。
このプログラムの内容を理解せずに使用すると、誤った結果を導きます。また、ソースコードが正しいことは証明されていません。バグのないソフトウェアが存在しないことは、既知の事実であり、このプログラムもその例外ではありません。
このプログラムを使用したことによる直接・間接のいかなる損害についても保証することはできません。このプログラムは、ユーザーの責任で使用して下さい。

このプログラムは、著作権でも保護されているため、このプログラムやこれを元に修正したプログラムを販売することを禁じます。無償で配布することは、GPL により許諾されています。

このプログラムに関するご質問は、ウェブサイトのフォームよりご連絡ください。可能であれば対応致します。
場合によっては、ブログでの回答になることをご了承ください。

3. プログラムのバグについて

このプログラムは、何例かの計算では問題のないことを確認しましたが、計算していない条件では、誤った数値が入っていないことは、確認できていません。計算した結果、**何かおかしいと感じた場合には、各要素ごとに損失値を確認してください**。必要に応じ、データベース内の数値やプログラムの中の定数を修正してください。**入力ミスも必ず確認してください**。
データベース内の数値やプログラム中の定数を修正する前に、必ず、元のファイルをバックアップしておいてください。

4. プログラムに使用される定理と計算式の概要

4.1 ベルヌイの定理

水の配管経路での損失は、ベルヌイの定理にもとづいて計算します。
ベルヌイの定理は、水路のどの位置でも、全水頭が等しいという定理です。

たとえば、同じ水路上の点 A と点 B において、それぞれの点での静圧を p_A、p_B、速度を v_A、v_B、（どこかの基準点からの）高さを h_A、h_B とすると、下記の計算式が成り立ちます。

$$\frac{p_A}{\rho g}+\frac{v_A^2}{2g}+h_A=\frac{p_B}{\rho g}+\frac{v_B^2}{2g}+h_B+\Delta h=\mathrm{constant}$$

ここで、第一項を圧力水頭、第二項を速度水頭、第三項を位置水頭といい、点 A から点Bまでの水路の損失を損失水頭(Δh)といいます。
このままで計算しても良いのですが、プログラムでは、圧力の単位としてパスカル(Pa)を使用しているので、各辺に ρg を掛けて次の形にして使用します。

$$p_A+\frac{1}{2}\rho v_A^2+\rho g h_A=p_B+\frac{1}{2}\rho v_B^2+\rho g h_B+\Delta p=\mathrm{constant}$$

ここで、Δp の値を計算して、管路にわたり加算してゆくのが、ここで紹介するプログラムです。
ここで、もうひとつ必要になる式が、連続の式と呼ばれるもので、流体の密度が変化しない場合（非圧縮性流体）の連続の式は下記のとおりです。ただし、A は、管の断面積とします。

$$\rho v A=\mathrm{constant}$$

連続の式は、質量保存則の式なので上記のように常識的な形になります。普通にコーディングすれば、連続の式として意識していなくても間違うことはないでしょう。

4.2 圧力損失の係数

ベルヌイの定理を使用するにあたり、簡単に計算できないのは、損失である Δp の値だけです。
もちろん、ナビア・ストークスの式を使えば計算できるのですが、ナビア・ストークスの式は、非線型方程式なので、簡単には計算することができません。円管だけならナビア・ストークスの式でもそこそこ計算できますが、曲がりなどが出てくると簡単ではありません。
したがって、損失部分を計算するには、一般に使用されている簡便なモデルを使用します。
直管の圧力損失は、下記のようなモデル式で表されます。

$$\Delta p=f\cdot\frac{L}{D}\cdot\frac{\rho V^2}{2}$$

ただし、
- f　係数
- L　管路の長さ
- D　管の内径
- V　流速

ここで、直管については、管径ごとに損失係数を計算してデータベース化[1]しておけば良いので、損失の式は、単位長さあたりの損失を下記のように表現します。

$$\frac{\Delta p}{L}=\frac{f}{D}\cdot\frac{\rho V^2}{2}$$

これは、
　単位長さあたりの圧力損失＝配管径あたりの損失係数×流体の動圧

という形ですから、簡単に計算できる動圧に、別途求めてデータベース化しておいた損失係数を掛ける形となります。
また、継手や弁などの配管要素についての圧力損失は、配管径ごとに次式の係数を予め計算しておいてデータベース化しておけば、次式の形で圧力損失が計算できます。

$$\Delta p=\xi\cdot\frac{\rho V^2}{2}$$

これら一般の配管要素の他、例えば、熱交換器は、別に計算の定義式をプログラムに組込むとか、係数や損失値を直接指定できるようにしておけば、計算が可能になります。計算法によっては、曲がり等の配管要素を直管での相当長さとして計算する場合もありますが、このプログラムは、配管要素それぞれの損失値を計算するようにコーディングしています。
同じように、高低差もユーザーが直接指定します。
なお、配管内面の粗度による修正には、下記のスワミー・ジャインの式を使用しています。他の粗度を使用する場合には、プログラム中の係数を修正してください。

$$f=0.25\left[\log_{10}\left(\frac{\epsilon}{3.7D}+\frac{5.17}{R_e^{0.9}}\right)\right]^{-2}$$

工学的な説明はこれだけしかありません。

1　データベースにある値は、引用元が複数あるので、明示していませんが、ユーザー自身が妥当性を確認し、必要な場合は値を修正してください。

このソフトウェアに条件を入力するためには、施工図や設計図など（もしくは寸法がわかる資料）が必要です。配管経路をアイソメ図などで表現しておけば、ソフトウェアに入力する配管要素が定まるので、配管経路に亘っての圧力損失を計算できます。
このプログラムを使用して計算するのは一瞬ですが、アイソメ図等を描き、配管要素に分割して入力するのは従来通り時間がかかります。但し、報告書の作成時間などは短縮されるでしょう。また、報告書式は、ユーザーが使いたい書式に修正できるので、その場合は、書き込む変数を再選択し、セルのアドレスを変更してください。
同種の機能のソフトウェアも販売されていますが、これだけの計算のために、高価なソフトウェアを人数分購入するのではコストが跳ね上がるので、LibreOffice BASIC という無償のソフトウェアでアプリケーションを作ってしまおうというのがこの企画です。

5. ソフトウェアの詳細

それでは、実際のソフトウェアの詳細に入ります。
最初からプログラミング手法の講義を受けて知識を得たという人は稀だと思います。私もそうですが、文法の説明を少しだけ受けた後は、勝手に演習し、サンプルプログラムを実行し、解釈し、少しずつ変更して、結果を確認し. . . という過程を繰り返して覚えてきました。
私のように自己流の人も多いでしょう。
とはいっても、プログラムを美しく書けるよりも、実用的なプログラムを書くことを優先すべきだと思います。
私は職業プログラマーではありませんので、自己流の見苦しい表現はご容赦ください。
また、シート名の一部や、変数名には、ドイツ語を使用しているのでご留意ください。これは、プログラム言語が英語に近いので、予約語などを使用してバグが出ることを恐れたため[2]です（お遊びの要素が強いですが）。

LibreOffice BASIC や Excel VBA を使用したことのない人が、このプログラムを使用すると、その機能に驚くと思います。プログラマーでない人でもこれだけの機能を、短いプログラムで実現できる[3]優位性を実感してください。
計算処理は簡単ですが、レポート出力は、LibreOffice BASIC ならでは簡単に実現できた機能[4]です。スプレッドシートで計算レポートを作成することは多いと思いますが、後から行を挿入したり、変更を加えたりすると、計算式が破綻して何だか分からなくなります[5]。これに対してマクロプログラムで出力したレポートには、計算式を加えないことができるので、行を挿入して体裁を整えたり、文字を修正しても、レポートシートが破綻することはありません。

また、プログラムを理解すると、自分の目的に少しだけ足りない場合にでも、多少のバグがあっても、自分で修正できます。修正する際には、自分流に変更していってください。ただし、計算式を変更してわけが分からなくならないよう注意してください。
たったこれだけのプログラムですが、機能は市販のプログラムに引けをとらないと思うので、実際に使用する場合には手を加えて完成度を上げて下さい。

5.1 プログラムの構成設計

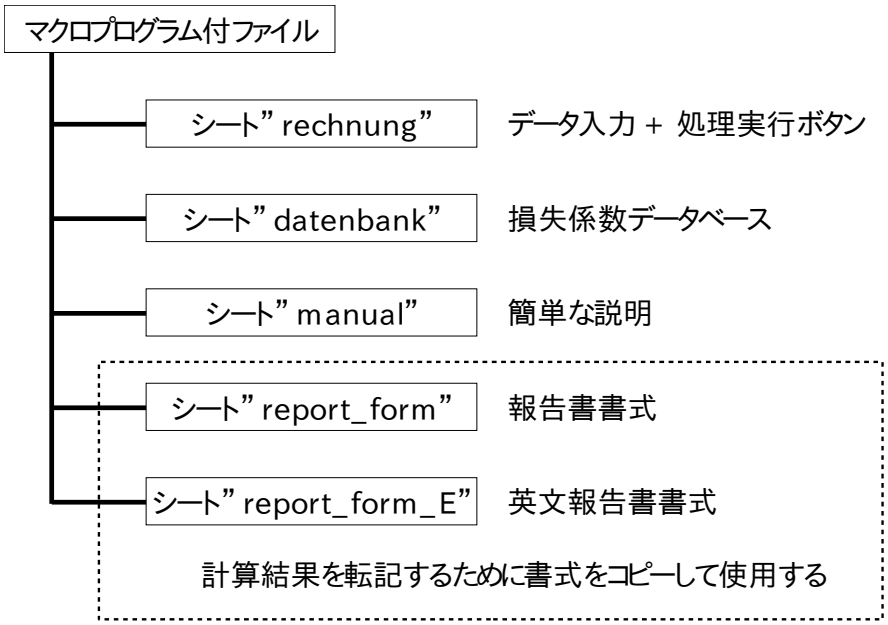
まず、プログラムに入る前に、ソフトウェア（実際は、マクロプログラム付のスプレッドシートファイル）としての構成を設計します。LibreOffice BASIC でソフトウェアをつくる場合には、シートを構成して造り込むことがコードと同じくらい重要です。
入力フォームは、プログラムの一部に相当します。FORTRAN や C 等の原始的な言語では、こういう部分も自分で作る必要がありましたが、LibreOffice BASIC では、スプレッドシート上にスペースを作るだけの話です。
計算結果は、同じファイルにも出力できるし、新たにファイルを作ってそこに書き込むこともできます。
このプログラムでは、報告書をいきなり作成せず、結果を確認してから所定の様式に転記するようにしています。

BASIC プログラムだけでなく、標準機能で作り込んだシートの内容もプログラムの一部であると考えてください。**シートの内容を変更すると、BASIC プログラムが意図したとおりに動作しない場合があります**。

プログラムは、下記のシートで構成されています。

---

2 意図せず予約語を使用して生じる不具合を防止するため、英語をそのまま使うのは避けたほうが良いようです。このため、日本語ローマ字を使用する人が多いです。
3 計算プログラムの経験がまったくない場合にはとっつきにくいかもしれませんが、ご了承ください。
4 Excel VBA も同じことができます。
5 スプレッドシートの計算式は、行や列を挿入した場合には、参照するセルの相対位置が変わってしまうので、何だか分からなくなります。



5.1.1 各シートの機能

rechnung（計算表）

ユーザーのデータを入力するためのシートです。色、枠、フォントで入力する部分を区別し、処理を実行するためのボタンを付けました。
入力する内容は、配管要素、要素記号（空白以外で任意）、管の内径、流量、その他高低差などの条件です。配管要素は、Calc の機能を使用してプルダウンで選択できるようになっていますが、同じものの場合には、コピー＆貼付しても同じです。プログラム内部で、ここで指定した文字列とデータベースに登録された文字列とを比較して係数を決めるようになっています。
後から途中に行を追加して項目を追加しても計算には問題がありません。ただし、要素記号に空白があるとその手前までしか計算しません。

datenbank（データベース）

損失係数の事前計算データベースです。係数の値のほとんどは、入手可能なデータを使用しています。実際に計算するにあたって、係数の値に疑問のある方は、普段使用されている値に変更することをお勧めします。
このデータベースには、ユーザーが必要な要素を追記することができます。値を直接指定できる場合には簡単ですが、計算が必要な場合には、C 列の値に、3 桁程度の整数を振って、プログラムの中に場合分けして計算できるようにします。なお、このプログラムでは、データとして読み込む最後の行数を、C1 セルに明示的に指定するようにしてあります。このため、要素を追加した場合には、C1 セルの値を変更する必要があります。
また、余分な機能かもしれませんが、このプログラムは、英語でレポートを出力できるようにしてあります。このため、このシートの B 列には日本語での要素名を記し、D 列には、要素名の英訳例を付けてあります。D 列とレポート書式シートを変更すれば、どのような言語でレポートを出力することも可能です。とりあえず、英語だけを例として付けてありますが、プログラムを修正すれば、何か国語にでも対応させることが可能です。

manual（マニュアル）

簡単な説明を記しています。（プログラムには関係ありません）

report_form（報告書書式）

報告書の書式です。ユーザーが変更しても構いませんが、セルのアドレスが変わるような変更の場合には、プログラムを修正する必要があります。
プログラムが計算した結果は、rechnung シートに書き込まれますが、同シートの『報告書作成』ボタンをクリックすると、プログラムがこのシートを指定した名前のシートとしてコピーし、そこに、計算結果を転記してゆきます。
計算結果が複数ページに及ぶ場合は、計算結果に誤りがないとことを確認したら、最終的にレポートシートを修正してください。

report_form_E（英語版報告書式）

英語版レポートの書式です。これもユーザーが変更することが可能です。英語版の報告書を提出しなければならない場合には、役立ちます。rechnung シートの”Report In English”ボタンをクリックすると、このシートが指定した名称にコピーされ、そこに計算結果を転記されます。

英訳部分が気に入らない場合には、計算後にレポート上で修正することもできますが、datenbank シートの D 列を修正することで、その後一貫した報告書に仕立てることができます。

以上に、各シートの役割を記述してきました。LibreOffice BASIC（Excel VBA も同様）の良いところは、Calc の機能を使用できることと、セルのアドレスを直接指定できるところです。
上記のような機能を持たせたデータシートを FORTRAN や C で作り込むのは、私には一生無理だと思いますが、LibreOffice BASIC なら簡単にできてしまいます。

5.2 プログラムの記述を理解するための付記事項

プログラムを配布することが目的ではありませんので、プログラムの各パートの詳細は省略し、ここでは、知っておくと便利なことを紹介します。一般的なプログラミング手法と類似の部分は、解説しませんので、コード中のコメントを参照してください。最も重要なのは、数値や文字列の受け渡し方法なので、その部分を中心に解説します。Excel VBA を使用している方は、表記方法の違いに戸惑うかもしれませんが、慣れれば同じようなものだと感じられと思います。

LibreOffice BASIC の文法については、分かりやすく紹介しているサイトがいくつかありますので、それらを参照してください。LibreOffice BASIC と OpenOffice BASIC とは、文法は共通です。
Excel VBA との主な違いは、下記のとおりです。
Calc BASIC は、プロパティ等の省略ができず、また、記述が長めなところと、だと思います。また、セルのアドレスを列→行の順に記述することと、アドレスが、1 ではなく、0 から始まっているところが Excel VBA とは違います。

5.2.1 シートやセルの指定方法

プログラムをつくるにあたって、もっとも重要なのは、対象のシートやセルアドレスの指定方法を覚えることです。
対象シートは下記のように指定します。

セルのアドレスは下記のように指定します。

buch=ThisComponent
このファイルを”buch”という変数名（オブジェクト）に指定しました。
なお、プログラム中にオブジェクト変数の宣言が必要になります。
このプログラムでは、文頭の宣言文の中に下記のオブジェクト変数宣言があります。

```
Dim buch, bankname, formular, rechnung, aSheet as Object
```

bankname=buch.getSheets().getByName("datenbank")
datenbank シートを、このファイルの bankname という変数名（オブジェクト）に指定しました
buch と同様に上記の宣言文で、bankname をお b ジェクト変数として宣言しています。

rechnung=buch.getSheets().getByName("rechnung")
bankname の項と同様です

aSheet=buch.CurrentController.ActiveSheet
このファイルのアクティブシートを aSheet という変数名で表します。
これは、都度、”buch.CurrentController.ActiveSheet”と記述しても構いませんが面倒です。

セルのアドレスは下記のように指定します。左辺は、配列を用いた変数名です。

auflisten(i, 1) = bankname.getCellByPosition(1,i).string
上述の bankname シートの i 行（一番上の行を 0 と数えるので、i-1 行と思えても良いでしょう）、1 列（一番左の列を 0 と数えるので、2 列と覚えても良いでしょう）にある文字列、という意味です。
ここで、bankname の代わりに、buch.getSheets().getByName("datenbank")と記述してもかまいません。
この文は、配列の要素を外部から見えるセルから読み込むもので、Calc BASIC のプログラムならではの機能です（Excel VBA も同様）。
配列での文字列変数 auflisten は、プログラムの文頭で下記のように宣言しています。

```
Dim aspekt, auflisten(400, 2) As String
```

auflisten(i, 2) = bankname.getCellByPosition(3,i).string
同上の内容です。

friktion(i) = bankname.getCellByPosition(2,i).value
上述の bankname シートの i 行（一番上の行を 0 と数えるので、i-1 行と思えても良いでしょう）、2 列（一番左の列を 0 と数えるので、3 列と覚えても良いでしょう）にある文字列、という意味です。

ここで、bankname の代わりに、buch.getSheets().getByName("datenbank")と記述してもかまいません。

5.3 プログラムルーチンの構成

共通の変数宣言部

Calc BASIC では、C や FORTRAN と同様、変数を宣言しなければなりません。FORTRAN では、宣言しないで使用できる変数名もありますが、BASIC では、宣言されていない変数はエラーとなります。
プログラム外部である文頭で変数宣言すれば、全てのサブルーチンで使用することができます。
各サブルーチンの中で変数宣言すると、変数名を使い回すことができません。

実行ボタン

errechnen_Click

『計算実行』ボタンをクリックすると”auffuellen”サブルーチンを実行する。
ボタンは、Calc のツールを使って作成し、ボタンのプロパティで、実行するサブルーチンを指定します。
この例では、ボタンにサブルーチンの”auffuellem”を直接割り当ててても良いのですが、
”errechnen_Click”から”auffuellen”を呼び出すようにしています。
この書き方が、一般的かどうか分かりませんが、私はこのようにボタンに割り当てたサブルーチンから実行したいサブルーチンを呼び出す書き方をしています。

report_Click（日本語書式の報告書作成）

『報告書作成』ボタンをクリックすると、作成言語を日本語に指定して、”bericht”サブルーチンを実行する。
“report_Click”と、下記の”report_Click_e”とは、同じサブルーチン”bericht”を呼び出すのですが、ボタンに割り当てたサブルーチンの中に、言語スィッチとして用いた変数”sprache”の値を”0”にしてから”bericht”を呼び出して、日本語報告書を出力させています。

report_Click_e（英語書式の報告書作成）

『Report In English』ボタンをクリックすると、作成言語を英語に指定して、”bericht”サブルーチンを実行する。
このサブルーチンでは、言語スィッチとして用いた変数”sprache”の値を”1”にしてから”bericht”を呼び出して、英語報告書を出力させています。

streichen_Click（計算値消去）

『計算値消去』ボタンをクリックすると”streichen”サブルーチンを実行する。
計算は一度で間違いなく完了することは稀で、数値の入力ミスや、条件の指定等が必ず起こりえます。
また、計算した結果が予想と大きく違っていることもあるため、条件を入力しなおしたり、考慮されていなかった配管要素を加えたり、余分に入れてしまった配管要素を削除したりして、最終的に、納得ゆく計算結果になってから報告書にします。このときに、前の計算結果が残っていると間違いの元なので、前の計算結果は、全て消去するようにしています。
実際に計算するユーザ以外の専門のプログラマがこうした計算プログラムを作ると、データを全部入力し直さなければならなかったりしますが、Calc BASIC で、このようにプログラムを構成すると、あとからデータ行を挿入したりしても問題ありません。

処理ルーチン

auffuellen（圧力損失計算実行モジュール）

圧力損失計算を実行する。
“rechnung”シートに記入した数値と条件に基き、計算ステップごとに”datenbank”シートから損失係数を読み取るか、”datenbank”シートの情報から損失を個別に転記または計算し、個々の損失を集計します。

streichen（計算値消去モジュール）

入力値はそのまま残し、自動計算した結果だけを消去する。
入力や条件に間違いがないか、計算をやり直すときに使用します。

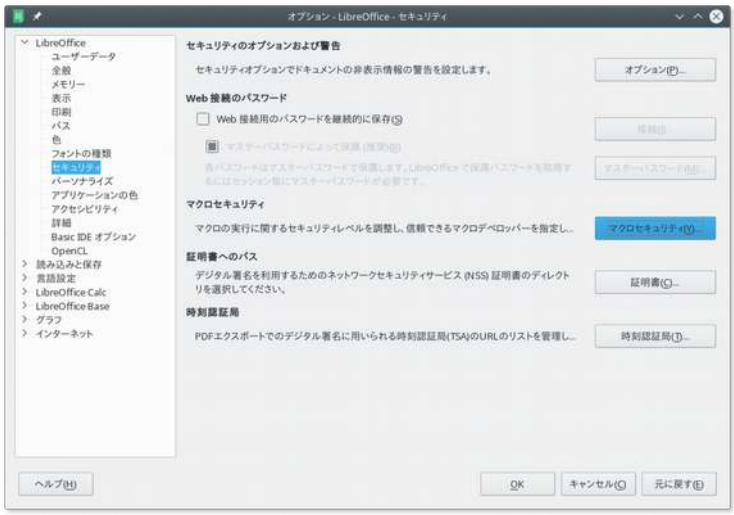
bericht（報告書作成モジュール）

レポート書式を別なシートとしてコピーし、ユーザの入力値と自動計算結果をコピーしたシートに転記する。この処理を実行すると、まず、出力するシート名を入力するためのダイアログが表示され、報告書式シートを新しいシートとしていちばん手前のシートとしてコピーします。
そして、”rechnung”シートに記入された計算結果を新しいシートに転記します。
auffuellen モジュールでは、圧力損失の計算値そのものを表示していましたが、このモジュールでは、それに加えて、余裕係数を掛けた計算結果も表示します。
報告書言語は、前述したとおろ処理ボタンを押すと自動に言語が選択されます。

6. プログラムの修正
このプログラムは、そのままでもそこそこ使えますが、ユーザーの条件により更に拡張してゆくことが可能です。
拡張が必要なのは、主に配管要素の追加と係数の修正でしょう。損失の係数は、SHASE の値を主に使用していますが、文献やデータベースにより多少違うし、転記のミスもあり得るので、係数を全てチェックして、変更してください。軽微な係数の差は、気にする必要はないものと思います。
長い配管経路の場合は、管内面の表面の状態による差が大きくなるので、普段使っている値と差がある場合に、普段の計算アルゴリズムが信頼できる場合には、このプログラムの中にある係数やアルゴリズムを修正してください。
このプログラムは、計算シートによる手書き計算表を自動化しただけなので、こうした修正は有効でしょう。
また、配管のサイズや継手等の種類がこのデータベースにない場合にも追加が必要になります。特別な圧力損失要素は、データベースに適当な番号をふることにより追加できます。
このプログラムでは、配管の内径は直接指定するようにしています。このことにより、管の種類を変更しても対応できるのですが、ここも自動計算にすることは可能です。機能を増やすとバグの元になるので、内径は、都度入力することをお勧めします。規格表を調べて入力するだけだし、古い配管などは、付着したスケールを適切に見積ることにより、市販の便利なプログラムよりも却って良い結果を出せるかもしれません。
個人的には、マニュアル計算っぽい要素を残しておくことで、結果を盲目的に信てしまうような失敗がしにくくなると思います。
また、あまり、ガチガチに作ってしまうと、少し違うだけで、その都度プログラムを変更しなければならないので、これ以上の便利さは追求しないほうが良いかもしれません。
変数名が気に入らない場合には、検索して置換する機能を使用すればよいと思います。ドイツ語の変数名で作ってみたのは良いのですが、実際にプログラムを読むと見にくいです。

7. 使用方法

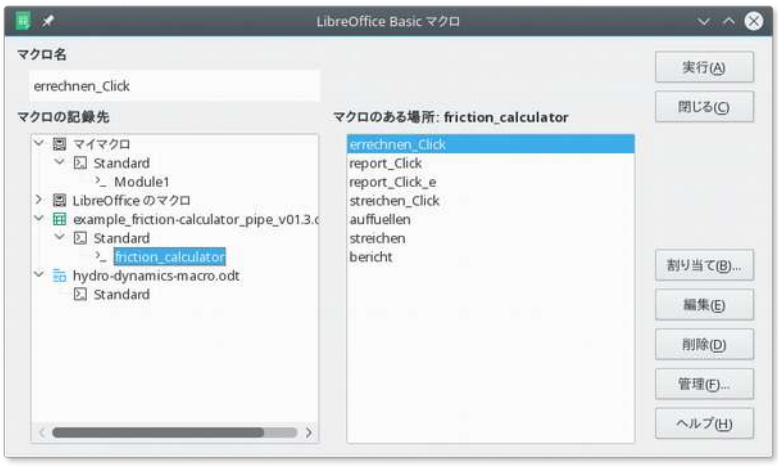
7.1 準備



このファイルを開いて使用するには、マクロが有効になる設定にしておかなければなりません。マクロの設定を有効にするには、プルダウンメニューからツール−オプションを選択します。ここで開かれるウィンドウで LibreOffice の下のセキュリティを選択し、マクロセキュリティを開きます。

マクロセキュリティを開いたら、セキュリティレベルが表示されるので、ここから、『中』または『低』を選択します。私は、中を選択しており、マクロのあるファイルを開くときは、確認してから開くようにしています。

マクロの設定が完了したら、ファイルを開きます。警告表示が出ますが、私を信頼して下さる方は、『マクロの有効化』をクリックして開いてください。この場合、プログラムが使用できる状態になります。
信頼しない方は、『マクロの無効化』を選択してマクロなしで開いてください。開いた後に、プルダウンメニューから、ツール−マクロ−マクロの管理−LibreOffice を選択してください。



開いたファイル名のファイルの下の”Standard”の下に”friction calculator”というものがあるので、その画面で、『編集』をクリックしてください。そうすると、ソースコードが表示されるので、怪しい記述がないことを確認したら、一旦プログラムを閉じて、再度、『マクロを有効化』して開いてください。
マクロのあるファイルを受け取った場合には、このように確認すれば、悪意あるファイルを排除することができます。

7.2 条件の入力

“rechnung”シート

このプログラムを使用するには、通常は、”rechnung”シートに条件を入力します。
このシートには、入力すべき部分、計算値がプログラムによって書き込まれる部分とそれ以外に分かれています。
入力する条件は下記のとおりです。

要素記号

配管系統図にユーザー自身が付けた配管の記号。数字でも文字でも良い。空白にすると、その手前までしか計算しない。

数量

同一配管要素の数量。例えば、区間の中にエルボが 2 個あった場合は、2 を入力する。ひとつしかない場合には、1 を入力する。同一要素をこのようにまとめて入力する場合には、その区間内の配管長は合計値を入力してください。細かな計算であることを見せたい場合には、同じ要素が近くにあっても、別々の要素として入力すると、計算表の行数が増えます。本質的には、近傍の要素をひとつのグループとしても別々に入力しても計算結果は同じです。
ただし、径違い継手がある場合には、その前後を一緒にしてはいけません。

配管要素選択

“datenbank”シート（データベース）にリストアップした配管要素を選択する。セルをクリックすると、右側に下向き矢印が現れるので、そこをクリックするとデータベースにある配管リストを選択できます。
データベースに配管要素を追加した場合には、ドロップダウンリストのセルの範囲を修正してください。
ドロップダウンリストを使わずに、コピー＆貼付しても同じです。これは、BASIC プログラム中で、同じ文字列を探すようになっているためです。データベースにない文字列を入れても計算できません。

流量

F11 セルに源の流量をリットル毎分単位で入力します。この列の12行以下はプログラムが自動で書き込みます。
G12 以下のセルに系から流出する流量または系に流入する流量を記入します。流出は正の数値、流入は府の数値を入力します。

相当長

直管の長さ、または、配管要素名に『相当長』と示されている場合には、直管相当の長さを入力します。

添付資料

A1. 変数一覧

変数名	型	内容	備考
i,j	integer	カウンタ	
reihen_auflisten	integer		使用せず
reihe_rechnung	integer	損失要素の数量	
nom_intestazione	integer	レポートシートのヘッダ行数	
reihe_erst	integer	最初のデータ行	
num_list	integer	リスト要素の最大数	bankname.getCellByPosition(2, 0)
reihe_rechnung	integer	現在計算中の行数	rechnung.getCellByPosition(3, 0) 最終的には計算する全行数
aspekt	string	圧損要素の名称	rechnung シートの２列目
auflisten(400,2)	string	データベースの内容	auflisten(*,1)=要素の名称 auflisten(*,2)=データベースの数字
reibung	double	単位長当りの圧損(kPa/m)	- rechnung.getCellByPosition(10,*) - faktor*druckbewegung / durchmesser*1000
friktion	double	損失係数（中間変数）	
verlust	double	部分圧力損失値	flesen*reibung*anzahl または reibung*laenge*anzahl
fluss	double	流量(l/min)	rechnung.getCellByPosition(5,*)
laenge	double	管長(m)	rechnung.getCellByPosition(7,*)
durchmesser	double	円管の直径	rechnung.getCellByPosition(8,*)
flaeche	double	断面積	3.14 / 4 * durchmesser ^ 2
schnellkeit	double	流速	(fluss / 60 / 1000) / flaeche * 1000000
druckbewegung	double	動圧	schnellkeit ^ 2 * 1000 / 2 / 1000
flesen	double	損失係数のデータベースからの読取値	ユーザ定義等損失値以外の場合があるので注意
anzahl	double	数量	schnellkeit * durchmesser / 1000 / nu
faktor	double	損失係数（直管のみ）	
reynolds	double	レイノルズ数	
pa, maq, ftaq	double	合計圧力損失値	単位違い(Pa, m-H_2O, ft-H_2O)
eps	double	スワミー・ジャインの式の係数 ε	
rohr	integer	管種	0:VLP,1:SGP,2:SUS
buch	object	このファイル	ThisComponent
bankname	object	“datenbank”シート	buch.getSheets().getByName("datenbank")
formular	object		使用せず
rechnung	object	“rechnung”シート	buch.getSheets().getByName("rechnung")
aSheet	object	アクティブシート	buch.CurrentController.ActiveSheet
objController	object		
objSheet	object	対象シート	
aufsatz	string	レポートのシート名	
sprache	int	言語選択スィッチ	0: 日本語,　: 英語

注：行と列の番号は、LibreOffice BASIC の記述法に倣い、0 から始めています。例：B 列＝列番号 1

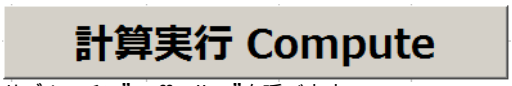
A2. プログラムの解説

```
REM  ***** LIBREOFFICE/OPENOFFICE BASIC  *****
REM     配管圧力損失計算プログラム
REM     COPYRIGHT SHIGERU SUZUKI
REM     商用利用を禁ずる。
REM     配布する場合には無償で配布すること。
REM     ライセンス　GPL

Dim i, j, reihen_auflisten, reihe_rechnung, nom_intestazione As Integer
Dim reihe_erst, num_list As Integer
Dim aspekt, auflisten(400, 2) As String
Dim reibung, friktion(400) As Double
Dim verlust, fluss, laenge, durchmesser As Double
Dim sostanziale, flaeche, schnellkeit, druckbewegung, flesen As Double
Dim anzahl, faktor, reynolds, nu As Double
Dim pa, maq, ftaq As Double
Dim eps As Double
Dim rohr As Integer
Dim buch, bankname, formular, rechnung, aSheet as Object
Dim sprache As Integer
```

グローバル変数宣言部
REM または ' で始まる行はコメント
' で始まるコメントは行の途中以降任意の位置に付けられる

```
Sub errechnen_Click()
'     『計算実行』ボタン
    Call auffuellen
End Sub
```

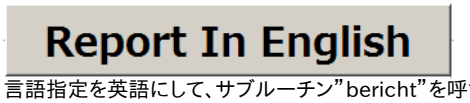


サブルーチン”auffuellen”を呼び出す

```
Sub report_Click()
'     『報告書作成』ボタン
'     sprache : 言語選択スイッチ(0:日本語)
      sprache=0
' 日本語レポート出力ボタン
    Call bericht
End Sub
```

言語指定を日本語にして、サブルーチン”bericht”を呼び出す。

```
Sub report_Click_e()
'     『Report In English』ボタン
'     sprache : 言語選択スイッチ(1:英語)
      sprache=1
' 英語レポート出力ボタン
    Call bericht
End Sub
```



言語指定を英語にして、サブルーチン”bericht”を呼び出す。

```
Sub streichen_Click()
'     『計算値消去』ボタン
    Call streichen
End Sub
```

サブルーチン”streichen”を呼び出す

```
Sub auffuellen()
'     圧損計算モジュール

'     オブジェクト指定
buch=ThisComponent
bankname=buch.getSheets().getByName("datenbank")
rechnung=buch.getSheets().getByName("rechnung")
aSheet=buch.CurrentController.ActiveSheet
'
' 圧損要素開始行の 1 行前

reihe_erst = 9     '最初の行
num_list = bankname.getCellByPosition(2, 0).value     'リスト要素の最大数
nu = 0.000001308   '水の動粘性係数

'   verlust              部分圧力損失値
'   flesen               損失係数のデータベースからの読取値
'   friktion             損失係数（中間変数）
'   faktor               損失係数（直管のみ）
'   anzahl               数量
'   rohr                 管種(0:VLP,1:SGP,2:SUS)
```

プログラムのメインモジュール
圧力損失計算を実行する

buch：このファイル
bankname：データベースシート
rechnung：計算シート
aSheet：現在アクティブなシート

プログラムの行番号や列番号を数字で指定すると、修正が面倒なので、データの書込を始める行の手前までの行数や行番号を、定数として指定している。
圧損要素のリストの要素数は、データベースシートの C1 セルに書き込まれている（大きな数字にしても良いが処理に時間がかかるためここで指定している）

nu(ν)：水の動粘性係数をここで指定している。10℃のときの動粘度(m^2/s)

各変数はコメント参照

```
' 要素の行数を求め、D1 セルに書込む

For i = reihe_erst+1 To 1000
    If aSheet.getCellByPosition(0,i).string = "" Then
        reihe_rechnung = i - reihe_erst - 1
        GoTo Z10010
    End If
Next i
Z10010:
'     D1 セルに要素数を書込む
    rechnung.getCellByPosition(3, 0).value = reihe_rechnung
```

要素の行数を求める部分
要素記号が始まる行から、1000 行までの間に A 列セルの空白を発見したら、その手前までを研鑽対象とし、その値を D1 セルに書き込む。

' 流量を自動計算する For i = 2 To reihe_rechnung 　rechnung.getCellByPosition(5,reihe_erst+i).value = rechnung.getCellByPosition(5,reihe_erst+i-1).value - rechnung.getCellByPosition(6,reihe_erst+i).value Next i	系の始まりの流量は、ユーザーが F11 セルに書き込む。 G12 セル以降には、流出する流量（正の値）、流入する流量（負の値）をユーザが指定し、計算対象の要素での流量を、F 列のセルに書き込む
' "datenbank"タブ内の損失係数表を読込みメモリ内に保存する For i = 1 To num_list 　auflisten(i, 1) = bankname.getCellByPosition(1,i).string 　auflisten(i, 2) = bankname.getCellByPosition(3,i).string 　friktion(i) = bankname.getCellByPosition(2,i).value Next i	計算を高速化するために、データベースに書かれているセルの内容（文字列と損失値）

' A 列が空白になるまで、計算を継続する(最大で要素数まで)
' A 列セルに空白があるとその下にデータがあっても計算しないので注意
 	
For i = 1 To reihe_rechnung
　If rechnung.getCellByPosition(0,reihe_erst+i).string = "" Then
　　GoTo Z100
　End If
 	
　aspekt = rechnung.getCellByPosition(2,reihe_erst+i).string　'圧損要素の名称
　durchmesser = rechnung.getCellByPosition(8,reihe_erst+i).value　'円管の直径
　fluss = rechnung.getCellByPosition(5,reihe_erst+i).value　'流量
　laenge = rechnung.getCellByPosition(7,reihe_erst+i).value　'管長
　anzahl = rechnung.getCellByPosition(1,reihe_erst+i).value　'数量
　rohr = rechnung.getCellByPosition(17,reihe_erst+i,).value　'管種
 	
' 円管直径の値がある場合には、K 列に円管直径を書込む
　rechnung.getCellByPosition(11,reihe_erst+i).value = durchmesser
 	
' 直径は相当直径に統一し、L 列の値を計算に使う
　durchmesser = rechnung.getCellByPosition(11,reihe_erst+i).value
 	
' 断面積 -flaeche
' 円管の場合
' M 列に面積を書込む
　flaeche = 3.14 / 4 * durchmesser ^ 2
　rechnung.getCellByPosition(12,reihe_erst+i).value = flaeche
 	
' 流速,動圧,レイノルズ数 -schnellkeit, druckbewegung, reynolds
'　流速
　schnellkeit = (fluss / 60 / 1000) / flaeche * 1000000#
　rechnung.getCellByPosition(13,reihe_erst+i).value = schnellkeit
'　動圧
　　druckbewegung = schnellkeit ^ 2 * 1000 / 2 / 1000
　rechnung.getCellByPosition(14,reihe_erst+i).value = druckbewegung
'　レイノルズ数
　reynolds = schnellkeit * durchmesser / 1000 / nu
 	
' f 読取値 - flesen
' B 列内の選択した要素を読込み、メモリ内の損失係数表から損失係数を参照する
 	
For j = 1 To num_list
　If aspekt = auflisten(j, 1) Then
　　flesen = friktion(j)
　　GoTo Z200
　End If
Next j
 	
Z200:
 	
'　参照した損失係数を P 列にそのまま記入する
　rechnung.getCellByPosition(15,reihe_erst+i).value = flesen
'　レイノルズ数を Q 列に書込む
　rechnung.getCellByPosition(16,reihe_erst+i).value = reynolds
 	
' f - verlust
' 損失係数を修正する（直管のみ）
' 直管の場合は損失係数計算に、管長を含めるため、別処理にする
 	
If flesen = 999 Then
 	
'　"datenbank"タブからの損失係数の読値で 999 となっている場合(直管)
'　999 ではなく、計算式で損失係数を求め、損失値を計算する
 	
'　損失係数：スワミー・ジャインの式
'　管種によって、相対粗度を指定する（相対粗度は逆算⇒確認して使用すること）
'　管種を増やす場合には、下記管種選択条件文を修正する
 	
'　管種条件
　If rohr = 1 Then
　eps = 0.0000944　　'SUS 管(鏡面仕上)
　ElseIf rohr = 2 Then
　eps = 0.0001245　　'SGP 管
	計算シートに指定した圧力損失要素名とデータベースの要素名とを比較し、同じ要素を発見したら、その要素の損失値（または損失記号）を計算シートに転記するモジュール。
	A 列が空白になったら処理を打ち切る。
	
	aspekt : 圧損要素の名称
	
	durchmesser : 円管の直径（内径）（この部分では、計算シートに指定した mm 単位で読み込む）
	fluss : 流量（この部分では、計算シートに指定した L/M（リットル毎分）で読み込む）
	laenge : 管長(m)
	rohr : 管種（単位なし）
	
	この部分は、ダクトの圧力損失計算を元に作成したため、不要な処理を含んでいます。
	ダクトの場合は、円管以外に矩形断面のものがあるため、相当直径を使用していました。
	水路の配管は、殆どに場合円管のみなので相当直径という概念は使いません。
	
	流速(schnellkeit)は、m/s 単位
	動圧(druckbewegung)は、kPa 単位
	レイノルズ数(reunolds)は、単位なし
	
	損失値変数(flesen)に一旦、データベースからの読み取り値をそのまま格納する
	
	データベースの損失読取値が 999 となっている場合には、円管指定なので、損失係数を別途計算する
	
	損失係数は、スワミー・ジャインの式を指定して計算する。管種による相対粗度の違いは、SHASE のデータから逆算して求めている
	
	管種（表面の粗度だけが問題で、内径の違いは直接指定のため、問題にしない）は、ここでは、鏡面仕

```
    Else
    eps = 0.0001134        '上記以外（デフォルトでは VLP（塩ビライニング管））
    End If

'    スワミー・ジャインの式
'    別のモデルを使用する場合には、下記の式を修正する
   faktor = 0.25 * (Log(eps / 3.7 / durchmesser * 1000 + 5.74 / (reynolds ^ 0.9)) /
Log(10)) ^ (-2)
'
   reibung = faktor * druckbewegung / durchmesser * 1000
   verlust = reibung * laenge * anzahl

'    スワミー・ジャインの式によって計算した損失係数を S 列に書込む（デバグ用）
   rechnung.getCellByPosition(18,reihe_erst+i).value = faktor

'    損失係数を一般的に決められない場合に、条件を定める
'    datenbank シートの C 列の値
'    998          高低差をメートル単位で指定する
'    997          損失値を Pa 単位で接指定する
'    996          損失係数を直接指定する場合
'    その他   損失係数データベース(datenbank シート）の値を直接使用する
'    損失値の指定方法は、適宜拡張可能
'    ここでは、996, 997, 998 を使用しているが、Integer の範囲の数字で、紛らわしくなけ
れば使用可能

ElseIf flesen = 998 Then
'    損失値を mH2O 単位でユーザ指定した場合（高低差）
   verlust = rechnung.getCellByPosition(9,reihe_erst+i).value * 9.81 * anzahl

ElseIf flesen = 997 Then
'    損失値を kPa 単位でユーザ指定した場合（空調機、冷凍機、冷却塔）
   verlust = rechnung.getCellByPosition(9,reihe_erst+i).value * anzahl

ElseIf flesen = 996 Then
'    損失係数をユーザ指定した場合
   verlust = rechnung.getCellByPosition(9,reihe_erst+i).value * druckbewegung *
anzahl

Else

' 直管、高低差以外の場合は、損失係数はそのまま使用する

   verlust = flesen * reibung * anzahl

End If

' 損失係数は D 列に記入する

'    損失値(kPa)
  rechnung.getCellByPosition(3,reihe_erst+i).value = verlust
'    損失係数(-)
If flesen<996 then
  rechnung.getCellByPosition(4,reihe_erst+i).value = flesen
Else
  rechnung.getCellByPosition(4,reihe_erst+i).value = faktor
End If
'    単位長圧損(kPa/m)
  rechnung.getCellByPosition(10,reihe_erst+i).value = reibung

Next i
```

上げでないステンレス管、SGP 管、塩ビライニング管だけにしている。
管種を増やす場合には、ElseIf 文を付けて足してゆく。

スワミー・ジャインの式

$$f = 0.25\left[\log_{10}\left(\frac{\epsilon}{3.7\,D}+\frac{5.17}{R_e^{0.9}}\right)\right]^{-2}$$

faktor : スワミー・ジャインの式の f
reibung : 単位長当りの圧損(kPa/m)
durchbewegung : 動圧(kPa)
verlust : 区間の圧力損失値(kPa)
anzahl : 数量（単位なし）
区間の圧力損失=f×動圧÷直径×長さ×数量

データベース中の f の読取値(flesen)の値によって、区間の圧力損失の係数を変更する。
条件を増やす場合には、995, 994,...等を追加して If 文に ElseIf を追加して増やしてゆけば良い。

flesen=998（損失値を mH_2O で指定）

$$\Delta p = 入力値 \times 9.81 \times 数量$$

flesen=997（損失値を Pa で指定）

$$\Delta p = 入力値 \times 数量$$

flesen=996（損失係数を直接指定）

$$\Delta p = 入力値 \cdot \frac{\rho V^2}{2}$$

flesen が上記以外の場合（直管の場合）

flesen=ξ,　$\Delta p = \xi \cdot \frac{\rho V^2}{2}$

損失値と係数を、計算表(recnung シート)に書き込む。

以下は、プログラム中のコメント参照

```
Z100:

' 要素毎の圧力損失値(Pa)を集計する

pa = 0        ' 損失集計値の初期値
For i = 1 To reihe_rechnung
   pa = pa + rechnung.getCellByPosition(3,reihe_erst+i).value
Next i
```

```
' 合計圧力損失値(Pa)を書込み、Pa 値を、mAq 単位とインチ Aq 単位に変換した値も書込
む

   maq = pa / 9.81
   ftaq = maq / 0.3048

   rechnung.getCellByPosition(3, 4).value = pa
   rechnung.getCellByPosition(3, 5).value = maq
   rechnung.getCellByPosition(3, 6).value = ftaq

'    計算ステップ数が多く計算時間が長い場合を想定し、計算終了メッセージを出す（通常
は数秒で計算終了）
Msgbox "計算完了 Calculation Complete!",0,"配管損失計算ツール PIPING
PRESSURE LOSS CALCULATOR"

End Sub
```

```
Sub streichen()

' 計算結果消去モジュール
' 自分の入力は消去せず、計算結果部分のみ消去する
buch=ThisComponent
rechnung=buch.getSheets().getByName("rechnung")

reihe_erst = 9     '最初の行
reihe_rechnung=rechnung.getCellByPosition(3,0).value

For i = 1 To reihe_rechnung

    rechnung.getCellByPosition(3,reihe_erst+i).string = ""
    rechnung.getCellByPosition(4,reihe_erst+i).string = ""
    rechnung.getCellByPosition(5,reihe_erst+1+i).string = ""
    rechnung.getCellByPosition(10,reihe_erst+i).string = ""
    rechnung.getCellByPosition(11,reihe_erst+i).string = ""
    rechnung.getCellByPosition(12,reihe_erst+i).string = ""
    rechnung.getCellByPosition(13,reihe_erst+i).string = ""
    rechnung.getCellByPosition(14,reihe_erst+i).string = ""
    rechnung.getCellByPosition(15,reihe_erst+i).string = ""
    rechnung.getCellByPosition(16,reihe_erst+i).string = ""
    rechnung.getCellByPosition(18,reihe_erst+i).string = ""
    rechnung.getCellByPosition(19,reihe_erst+i).string = ""

Next i

    rechnung.getCellByPosition(3,4).string = ""
    rechnung.getCellByPosition(3,5).string = ""
    rechnung.getCellByPosition(3,6).string = ""
    rechnung.getCellByPosition(3,0).string = ""

Msgbox "計算値を消去しました Computed Values Are Removed!",0,"配管損失計算ツール PIPING PRESSURE LOSS CALCULATOR"

End Sub
```

このモジュールは、計算した結果に疑問があり、計算を再実行する場合に、前回の計算結果を消去します。

消去しなくても計算に支障はありませんが、このようなモジュールを作成しておくと、間違いにくくなります。

入力データは消去されないので、何度でも消して書くことができます。

```
Sub bericht()
' レポート作成モジュール

Dim aufsatz As String              '    レポートのシート名
Dim aSheet As Object               '    アクティブシート（使用していない）
Dim objController As Object
Dim objSheet As Object             '    対象シート
```

```
'    記述を簡素にするためのオブジェクト名指定
     buch=ThisComponent
     bankname=buch.getSheets().getByName("datenbank")
     rechnung=buch.getSheets().getByName("rechnung")
     aSheet=buch.CurrentController.ActiveSheet

'    リスト要素の最大数を datenbank シートの C1 セルから読む
     num_list = bankname.getCellByPosition(2,0).value-1
```

データベースにリストを追加した場合には、リスト要素の最大数をC1 セルに記入してください。

```
'    ダイアログボックスでシート名を入力
'    sprache : 日本語:0, 英語:1, Japanese:0, English:1, ボタンで自動指定
'    実行ボタンをクリックした際に言語を自動指定されるので、このモジュールを直接実行するとエラーになる

If sprache=0 then
     aufsatz = InputBox("出力するシート名を英数字で入力してください", "出力先シート名入力", "")
Else
     aufsatz = InputBox("Input Output Sheet Name", "SET OUTPUT SHEET", "")
End If
```

プログラムをボタンから実行しない場合には、言語指定の文を追加してください。

```
'    押されたボタンによってレポート書式を自動選択する
'    レポート書式をコピーし、コピーしたレポート様式の名称を指定のものに変更する
'    言語により報告書のひな型シートを使い分ける

If sprache=0 then
     buch.Sheets.copyByName("report_form",aufsatz,0)
Else
     buch.Sheets.copyByName("report_form_E",aufsatz,0)
End If
```

```
With ThisComponent
     objController=.getCurrentController()
     objSheet=.Sheets.getByName(aufsatz)
     objController.setActiveSheet(objSheet)
End With
```

```
' コピーする行数を数える
' reihe_erst : "rechnung"シートのヘッダ行数
' reihe_rechnung : 計算する行数
' nom_intestazione : レポートシートのヘッダ行数

reihe_erst = 9              ' 計算シートの最後のヘッダ行
nom_intestazione = 6   ' レポート書式の最後のヘッダ行
```

reihe_rechnung = rechnung.getCellByPosition(3,0).value	

' プロジェクト名と系統名をコピー
objSheet.getCellByPosition(2,2).string = rechnung.getCellByPosition(2,7).string
objSheet.getCellByPosition(2,3).string = rechnung.getCellByPosition(2,8).string
 	
'　余裕係数をコピー
objSheet.getCellByPosition(10,4).value = rechnung.getCellByPosition(3,3).value
 	
' 結果コピー
' 合計損失値を書込む
objSheet.getCellByPosition(2,4).value = rechnung.getCellByPosition(3,5).value
*rechnung.getCellByPosition(3,3).value
objSheet.getCellByPosition(4,4).value = rechnung.getCellByPosition(3,6).value
*rechnung.getCellByPosition(3,3).value
objSheet.getCellByPosition(6,4).value = rechnung.getCellByPosition(3,4).value
*rechnung.getCellByPosition(3,3).value
 	
' 計算中の詳細数字を書込む
For i = 1 To reihe_rechnung
objSheet.getCellByPosition(0,nom_intestazione+i).string =
rechnung.getCellByPosition(0,reihe_erst+i).string
objSheet.getCellByPosition(2,nom_intestazione+i).value =
rechnung.getCellByPosition(8,reihe_erst+i).value
objSheet.getCellByPosition(3,nom_intestazione+i).value =
rechnung.getCellByPosition(1,reihe_erst+i).value
 	
If rechnung.getCellByPosition(7,reihe_erst+i).string="" then
　objSheet.getCellByPosition(4,nom_intestazione+i).string="-"
Else
　objSheet.getCellByPosition(4,nom_intestazione+i).value =
rechnung.getCellByPosition(7,reihe_erst+i).value
End If
 	
objSheet.getCellByPosition(5,nom_intestazione+i).value =
rechnung.getCellByPosition(4,reihe_erst+i).value
objSheet.getCellByPosition(6,nom_intestazione+i).value =
rechnung.getCellByPosition(5,reihe_erst+i).value
objSheet.getCellByPosition(7,nom_intestazione+i).value =
rechnung.getCellByPosition(13,reihe_erst+i).value
objSheet.getCellByPosition(8,nom_intestazione+i).value =
rechnung.getCellByPosition(14,reihe_erst+i).value
objSheet.getCellByPosition(9,nom_intestazione+i).value =
rechnung.getCellByPosition(3,reihe_erst+i).value
	プロジェクト名、系統名、余裕係数は、計算シートにユーザが書き込みます。

For j = 1 To num_list
 	
　auflisten(j,1) = bankname.getCellByPosition(1,j).string '日本語
　auflisten(j,2) = bankname.getCellByPosition(3,j).string '英語
 	
　If rechnung.getCellByPosition(2,reihe_erst+i).string = auflisten(j,1) Then
　　If sprache=0 then
　　　objSheet.getCellByPosition(1,nom_intestazione+i).string = auflisten(j,1)
　　Else
　　　objSheet.getCellByPosition(1,nom_intestazione+i).string = auflisten(j,2)
　　End If
　GoTo Z10030
　End If
Next j
 	
Z10030:
 	
Next i	

'　損失値積算計算 '　最初の行は累積損失値と部分損失値が等しい objSheet.getCellByPosition(10,nom_intestazione+1).value = objSheet.getCellByPosition(9,nom_intestazione+1).value For i = 2 To reihe_rechnung If objSheet.getCellByPosition(10,nom_intestazione+i).value = "" Then　　' A 列に記号がなければ止める 　GoTo Z1110 End If ' 前回値と今回値を加算する objSheet.getCellByPosition(10,nom_intestazione+i).value = _ objSheet.getCellByPosition(10,nom_intestazione+i-1).value + objSheet.getCellByPosition(9,nom_intestazione+i).value Next i	損失値の累計を見ることにより、計算結果に問題点がある場合発見しやすくなります。
Z1110:　' 記号がなくそれ以下にデータがないと認められる場合 Msgbox "報告書出力完了 Report Complete!",0,"配管損失計算ツール PIPING PRESSURE LOSS CALCULATOR" End Sub	