作ってみようTRNSYSコンポーネント

このドキュメントは以下のライセンスで提供されます。

[クリエイティブ・コモンズ・ライセンス](http://creativecommons.org/licenses/by-nc-sa/4.0/)

quattro corporate design Co., Ltd. 作『TRNSYS.JP Library』は[クリエイティブ・コモンズ 表示 - 非営利 - 継承 4.0 国際 ライセンス](http://creativecommons.org/licenses/by-nc-sa/4.0/)で提供されています。

内容

[1 準備編 4](#_Toc420589374)

[1.1 用意するもの 4](#_Toc420589375)

[1.2 作り方 4](#_Toc420589376)

[2 プロフォルマ 6](#_Toc420589377)

[2.1 プロフォルマを作成する 6](#_Toc420589378)

[2.2 インターフェースを定義する 7](#_Toc420589379)

[2.3 プロフォルマを保存する 12](#_Toc420589380)

[3 ソースコードの生成 14](#_Toc420589381)

[3.1 ソースコードの生成 14](#_Toc420589382)

[3.2 プロジェクトの設定 17](#_Toc420589383)

[3.2.1 プロジェクト項目の抽出 17](#_Toc420589384)

[3.2.2 Fortranコンパイラの設定 18](#_Toc420589385)

[3.2.3 Debug/Release設定 20](#_Toc420589386)

[3.2.4 ライブラリー 22](#_Toc420589387)

[3.2.5 ランタイム・ライブラリーの配布 23](#_Toc420589388)

[3.3 ビルド 25](#_Toc420589389)

[4 ソースコードの編集 26](#_Toc420589390)

[4.1 ソースコードの確認 26](#_Toc420589391)

[4.2 処理の追加 28](#_Toc420589392)

[4.2.1 出力用の変数の追加 28](#_Toc420589393)

[4.2.2 計算式の追加 29](#_Toc420589394)

[4.3 ビルドする 30](#_Toc420589395)

[5 Simulation Studioで実行 31](#_Toc420589396)

[5.1 プロジェクトの準備 31](#_Toc420589397)

[5.1.1 Equationを追加する 32](#_Toc420589398)

[5.1.2 Online Plotter 33](#_Toc420589399)

[5.2 コンポーネントの接続 33](#_Toc420589400)

[5.3 実行 35](#_Toc420589401)

[5.4 まとめ 35](#_Toc420589402)

[6 デバッグ 36](#_Toc420589403)

[6.1 TRNDLL.DLLのソリューションを開く 36](#_Toc420589404)

[6.2 コンポーネントのプロジェクトを追加する 37](#_Toc420589405)

[6.3 プロジェクトの依存関係を設定する 38](#_Toc420589406)

[6.4 プロジェクトの設定を変更する 39](#_Toc420589407)

[6.5 デバッグモードで実行する 41](#_Toc420589408)

[7 補足：FORTRANの環境設定 43](#_Toc420589409)

[7.1 設定方法 43](#_Toc420589410)

[7.1.1 Windows7(32bit) 44](#_Toc420589411)

[7.1.2 Windows7(64bit) 44](#_Toc420589412)

[7.1.3 Shellで起動 44](#_Toc420589413)

[8 補足：関連ファイル 45](#_Toc420589414)

[8.1 フォルダ構成 45](#_Toc420589415)

[8.2 ソースコード 46](#_Toc420589416)

[8.3 プロフォルマ 47](#_Toc420589417)

[8.4 DLL 48](#_Toc420589418)

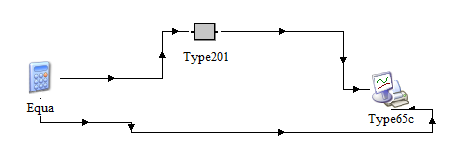
[9 補足：Fortran77/90形式の切り替え 49](#_Toc420589419)

※建築環境工学系日記（[http://kankyoukei.blogspot.jp/](http://kankyoukei.blogspot.jp/2012/08/trnsys1.html)）

2012/08/03～2012/08/15の掲載記事より転載。

# 準備編

夏休みなので、TRNSYSのコンポーネントを作ってみます。（普通に仕事しています。まあ、気分的にそんな時期だなと）



完成メージ

TRNSYSには標準でたくさんのコンポーネントが用意されています。一般的な計算はコンポーネントをつなげるだけでできてします。そうは言っても、研究のテーマによっては、ちょっと違った計算を試したいケースもありますよね？  
TRNSYSには、そんな時のために独自のコンポーネントを作成する仕組みが用意されています。これを覚えるとTRNSYSの使い勝手の幅がグッと広がります。  
  
※TRNSYSのドキュメント、Vol.1　Getting　Startedの内容に沿って作ってみます。

* 1. 用意するもの

TRNSYS17.1　最新版にアップデートしておきましょう。  
  
コンパイラ：　インテル(R) Visual Fortran Composer XE 2013 Windows版  
↓ここから評価版がダウンロードできます。  
<http://software.intel.com/en-us/articles/intel-software-evaluation-center/> 

* 1. 作り方

コンポーネントの作成は、だいたい以下のような流れで行います。

１）コンポーネントのプロフォルマを定義する  
作成するコンポーネントのインターフェースを定義します。具体的に言うと、コンポーネントのParameterとかInput、Outputの項目を定義します。

２）ソースコードの生成とコンパイラの設定  
プロフォルマが出来上がったら、ソースコードのスケルトン（雛形）を作成します。このあたりの処理はSimulation Studioがやってくれます。

３）ソースコードの編集とビルド

ソースコードが出力されたら開発ツールをつかって、計算内容を記述します。

４）Simulation Studioで実行

出来上がったコンポーネントで計算実行。

一回じゃまとめ切れないので、順番に書いていこうと思います。

つづく。

# プロフォルマ

コンポーネントのプロフォルマを定義する

はじめにプロフォルマ（Proforma）を用意します。これなにかというと、コンポーネントのインターフェースを定義しているファイルです。後述する例ではInputsで値を2個受け取って、合計した値を任意の倍数にしてOutputsへ返すコンポーネント作成しています。

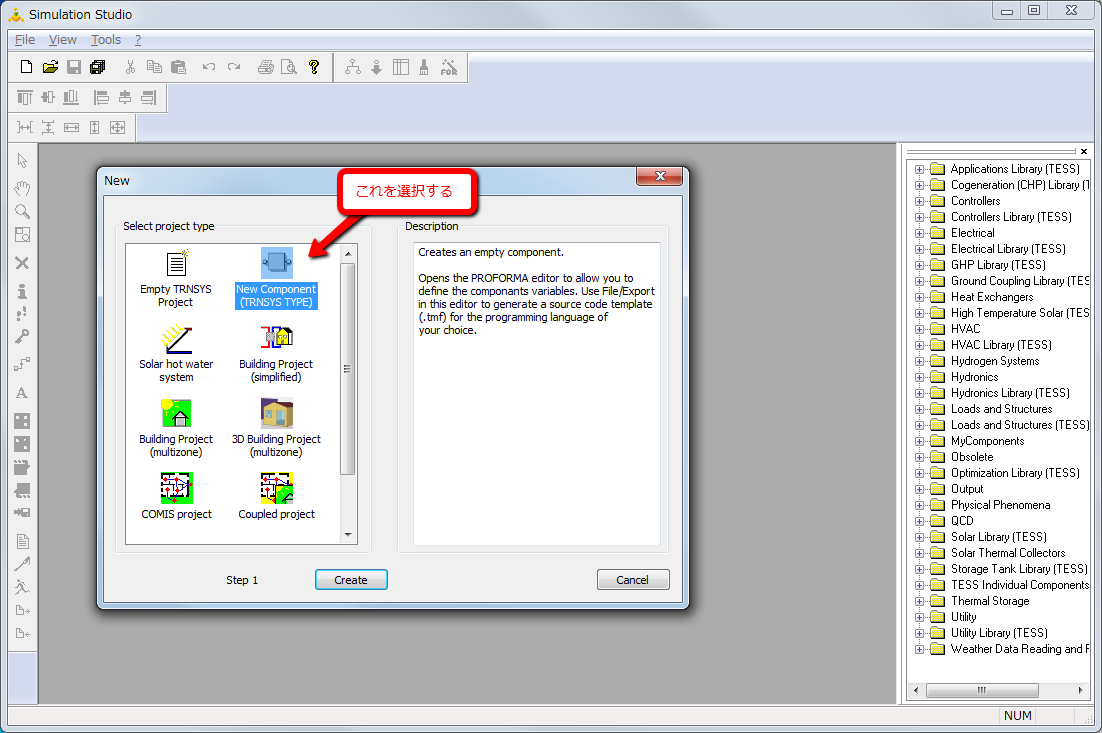
式で書くと、次のようになります。

Out1 = (Inp1 + Inp2) \* Mult

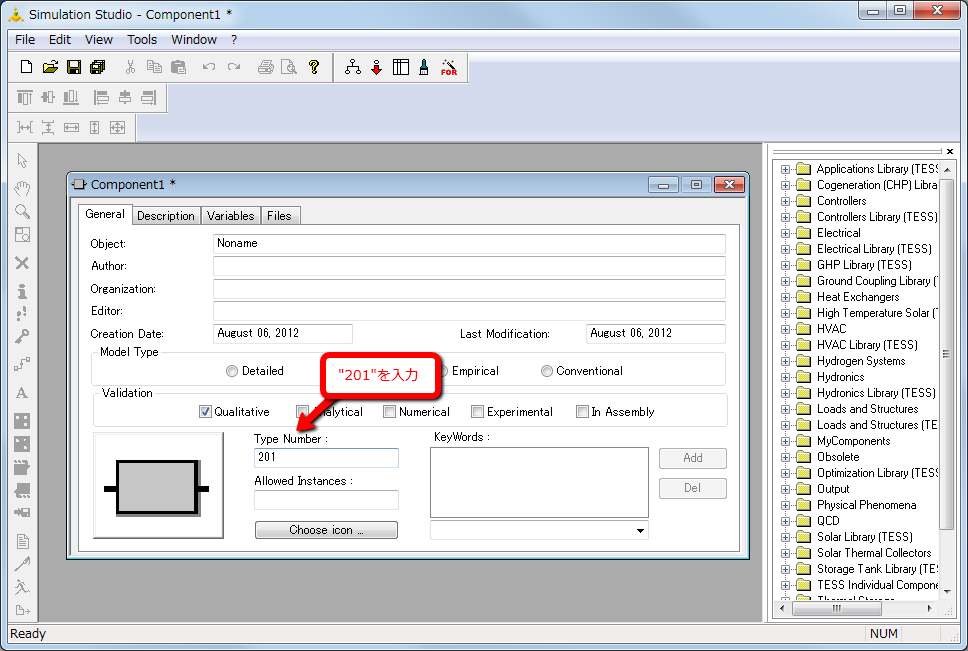
この式をプロフォルマでコンポーネントのParameters、Inputs、Outputsの項目として定義します。

* 1. プロフォルマを作成する

Simulation Studioを起動して、メニューから[File]-[New...]の順で選択します。表示されたダイアログで「New Component」のアイコンをダブルクリックします。



ダイアログが表示されたらコンポーネントの基本情報を入力します。ここではコンポーネントの「Type Number」に識別用の番号を入力します。番号はユーザー用に201～300~~250~~が割り当てられているので、この範囲で指定します。今回は201を指定します。



* 1. インターフェースを定義する

つづいて、インターフェースの設定です。今回作成するコンポーネントは以下のような構成です。

Parameters

　　　Mult (デフォルト:1)

Inputs：

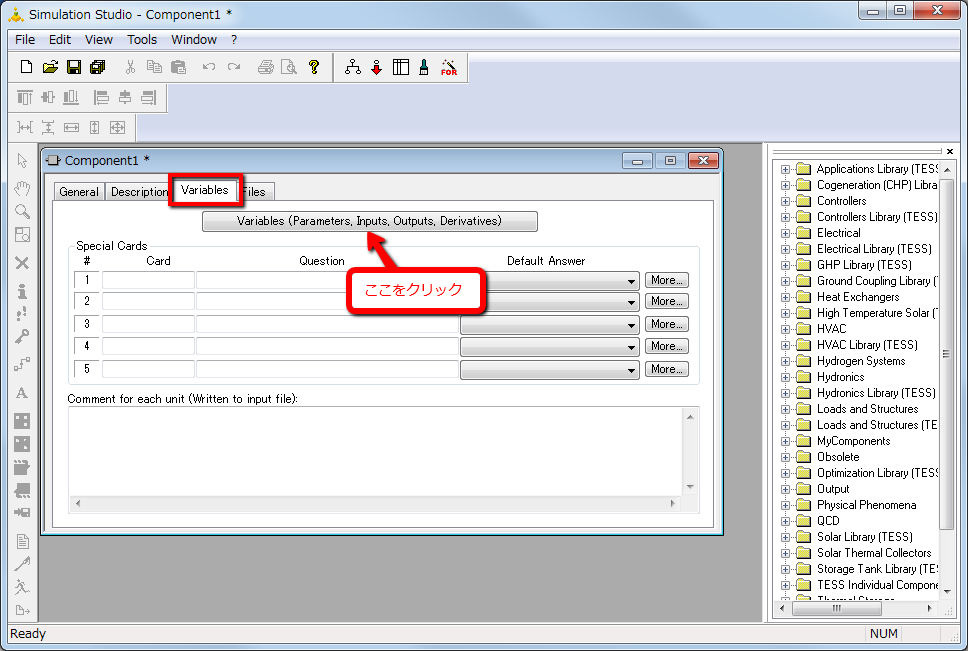
　　　Inp1

　　　Inp2

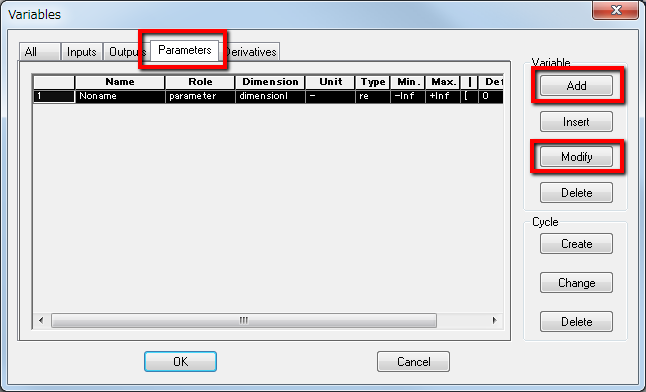
Outputs:

　　　Out1

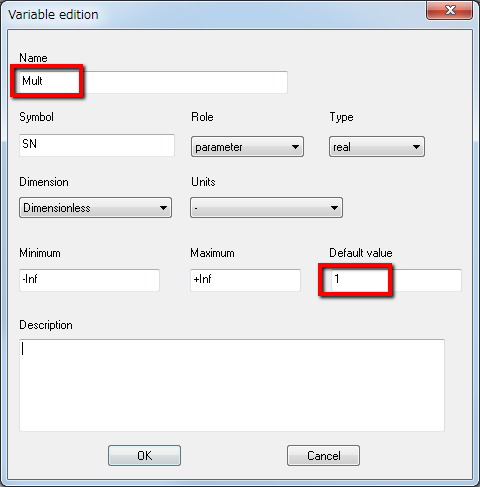
では、実際に設定してみましょう。「Variable」タブを選択して「Variables（Parameters, Inputs, ～」ボタンをクリックしてVariablesダイアログを表示します。



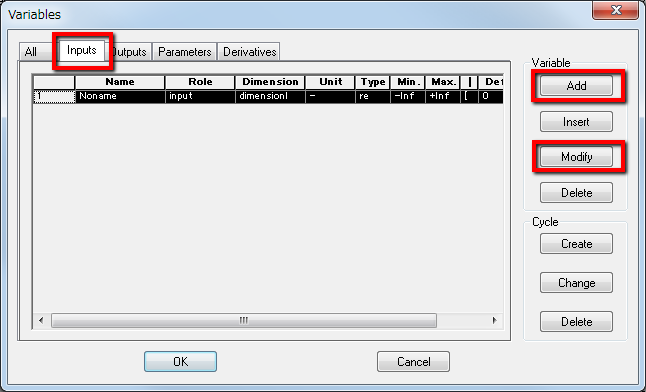
まず、はじめに「Parameters」タブを選択して、「Add」ボタンをクリックします。これで項目が追加されるので、つづけて「Modify」ボタンをクリックして、内容を定義します。



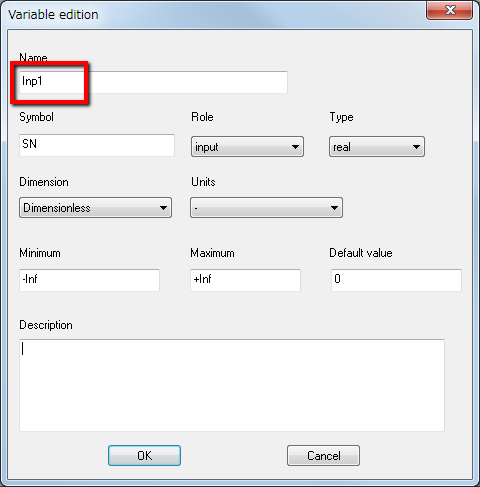
「Name」の項目にMultと入力します。デフォルト値は１にしたいので「Default value」に1を入力したら「OK」ボタンを押してダイアログを閉じます。



 つづいて、今度はInputsの追加です。「Inputs」タブを選択して、先ほどと同じように「Add」、「Modify」の順でクリックしていきます。

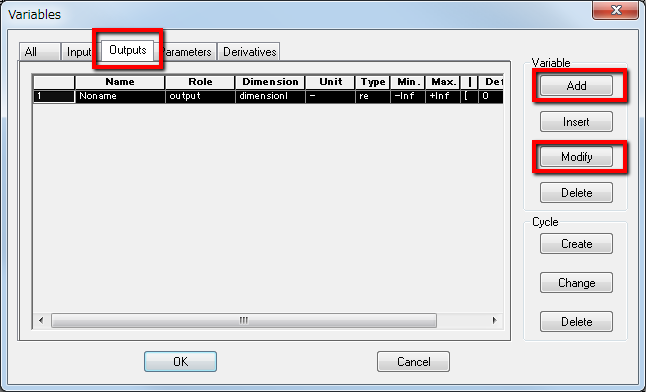


「Name」の項目にInp1を入力して「OK」ボタンをクリックしたら一つ目のInputsの項目の出来上がりです。

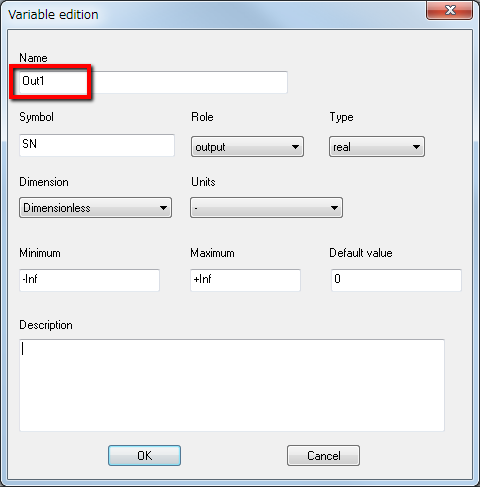


上記の作業を繰り返して、Inp2も同じように追加します。

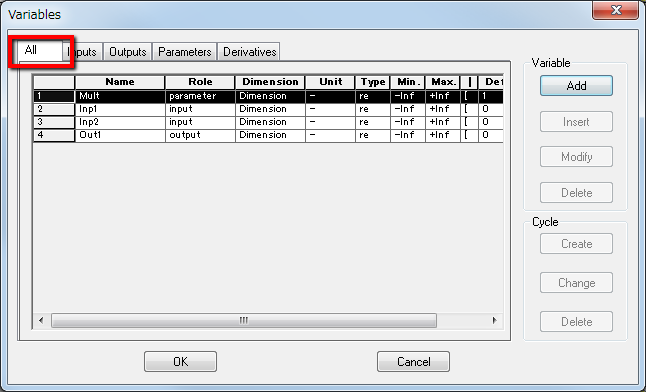
そしてOutputsの追加です。「Outputs」タブを選んで、「Add」、「Modify」をクリックして項目を追加します。



「Name」の項目にOut1を入力して、「OK」ボタンをクリックしてダイアログを閉じます。



最後に「All」タブを選んで図のような表示になっていれば完成です。「OK」ボタンを押してダイアログを閉じます。

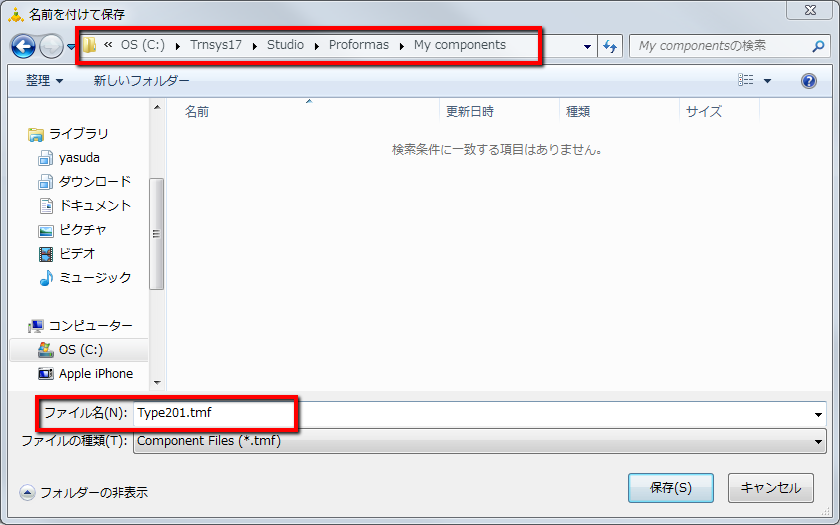


* 1. プロフォルマを保存する

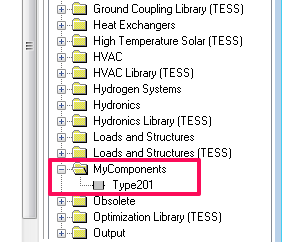
作成したプロフォルマを保存します。メニューから[File]-[Save...]を選択してファイルを保存します。

保存先のフォルダはSimulation Studioから認識できるように必ず以下のフォルダにする必要があります。

「C:\Trnsys17\Studio\Proformas\」  
今回はこのフォルダに「My Components」というフォルダを作成して、そこに「Type201.tmf」という名前で保存します。



保存した後、TRNSYSのプロジェクトを新規に作成すると、Simulation Stuidoの画面右側のDirect Access ToolbarにType201が表示されるようになります。（表示されていない場合はメニューから[Direct Access]-[Refresh tree]を選んで下さい）



プロジェクトへ配置することも出来ますが、この時点では中身が何もない状態なので、実行してもエラーになります。

以降では、ソースコードを生成して、実際に動作するようにします。

つづく。

# ソースコードの生成

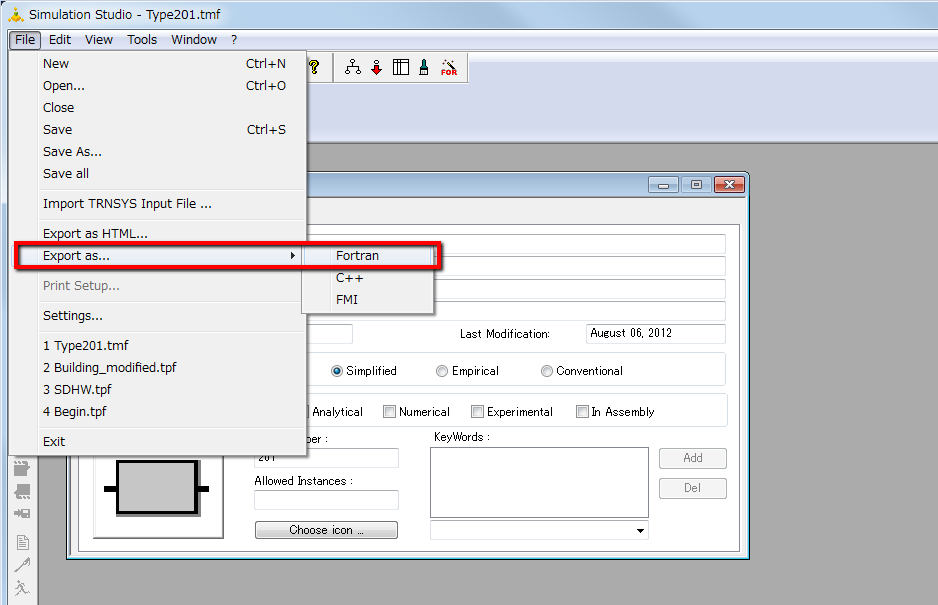
さて、前回までにプロフォルマまで出来上がったので、今回はソースコードの書き出しです。

Simulation studioにはプロフォルマで設定された内容から、ソースコードを書きだしてくれる便利なコマンドがあります。これを使ってソースコードのスケルトン（雛形）の書き出しを行います。

また、書きだされたソースコードをビルド（コンパイル、リンク）するためのプロジェクト設定もあわせて行います。

* 1. ソースコードの生成

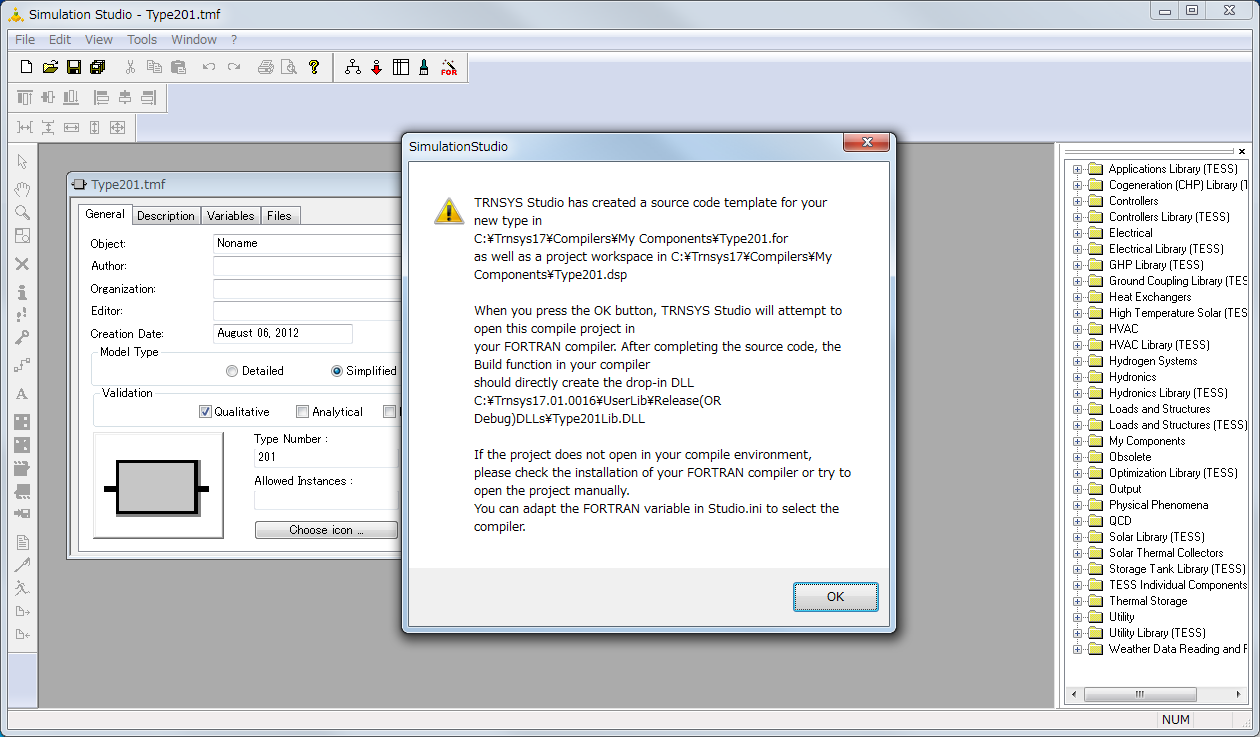
メニューから[File]-[Export as]-[Fortran]の順で選択します。



保存先は、基本的にどこでもOKですが、個人的にはコンパイラ関係の例が収められている"C:\Trnsys17\Compilers\"フォルダがオススメです。このフォルダに分かりやすいように"My Components"というフォルダを作って保存します。

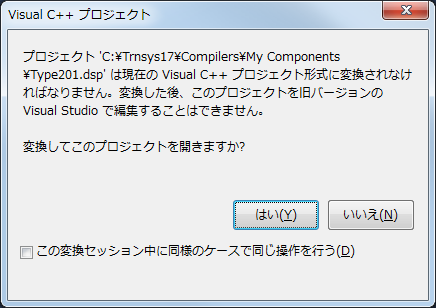
保存先：　C:\Trnsys17\Compilers\My Components  
ファイル名：Type201.for

ファイルが保存されるとメッセージが表示されます。保存先などの情報を確認したら「OK」をクリックします。

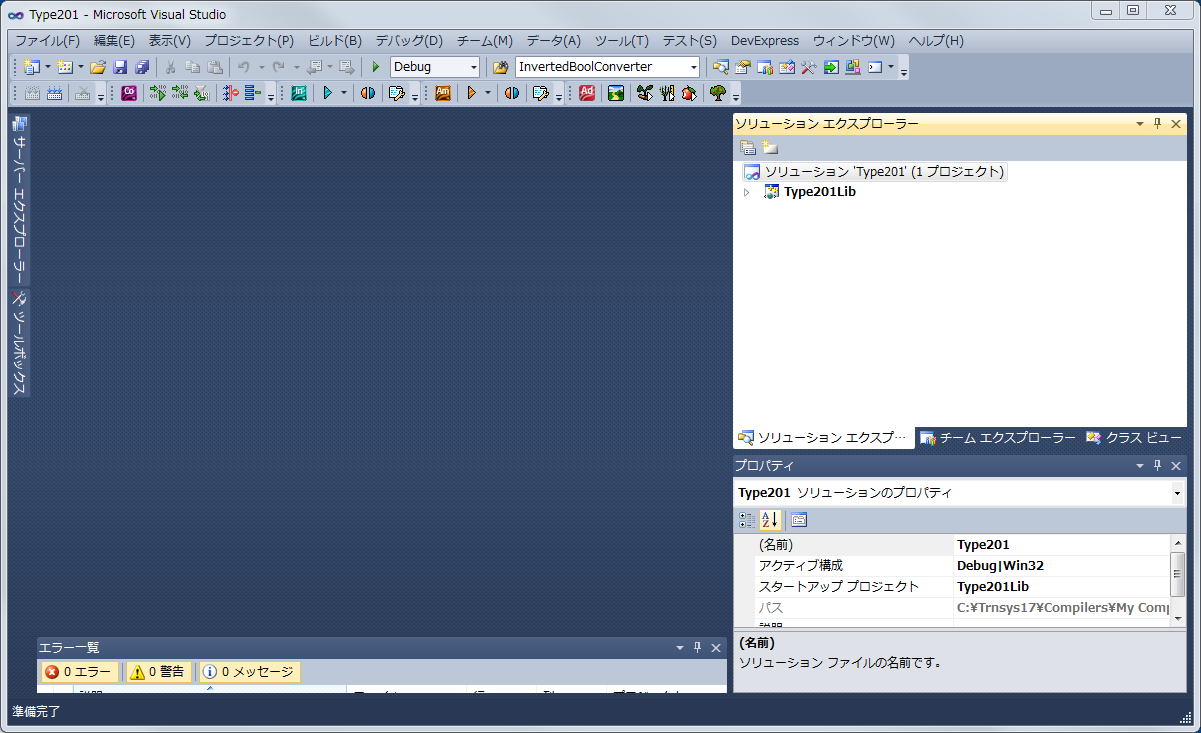


 つづけて以下のメッセージが表示されます。これはSimulation Studioが書き出すプロジェクトファイル形式が、以前のバージョンの形式に対応のため更新を促すメッセージです。

ここは素直に「はい」をクリックして次へ進みます。



変換が終わるとVisual Fortran Composer XE 2011が起動します。（見た目はというか、Visual Studioと同じShellを使っているので、画面はまったく同じになります）



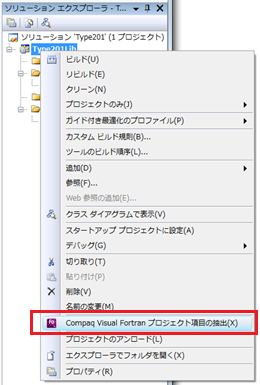
* 1. プロジェクトの設定

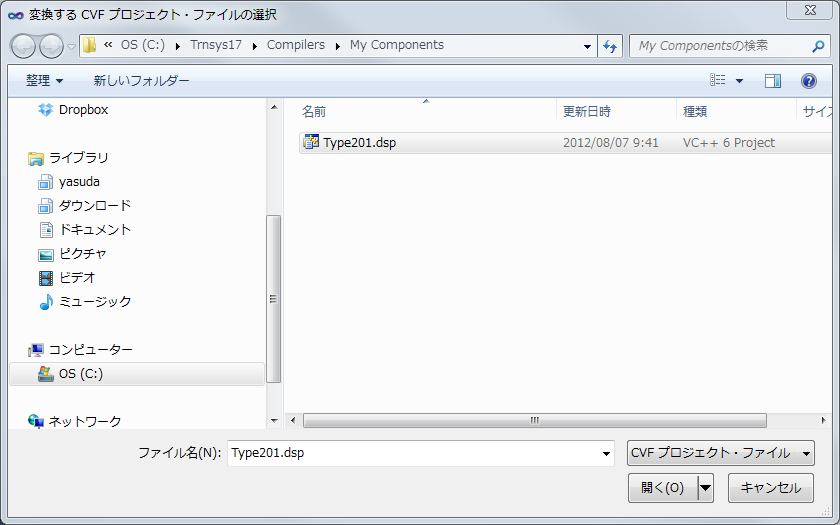
プロジェクトの変換作業を行います。Simulation Studioが書き出したFORTRANのソースコードはCompaq Visual Fortran(以下CVF）形式になっています。使用する開発環境に合わせて設定を変更します。

* + 1. プロジェクト項目の抽出

使用するIntel Parallel Studio XE 2013(Intel Fortran)に合わせてプロジェクトを変換します。

ソリューションエクスプローラーでプロジェクト（Type201lib）を選んで、右クリックで表示されるメニューから「COMPAQ Visual Fortran プロジェクト項目の抽出」を選択します。

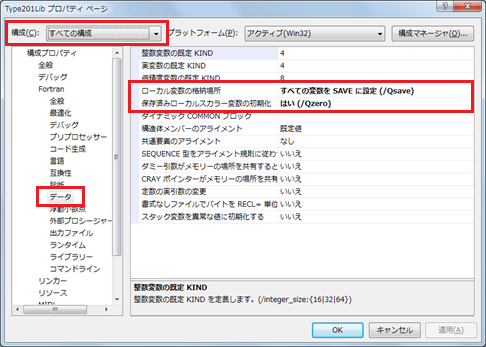


ここで変換対象のファイルを聞いてきます。先ほどエクスポートしたプロジェクトファイル「Type201.dsp」を選択して「OK」ボタンをクリックします。

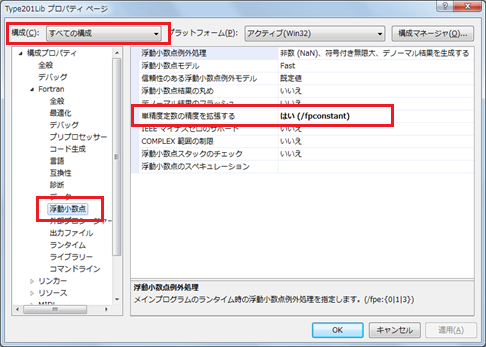
* + 1. Fortranコンパイラの設定

つづいてコンパイラ関係の設定です。メニューから[プロジェクト]-[プロパティ]で選択されるダイアログで「構成」で「すべての構成」を選択してFORTRANコンパイラの設定を行います。赤枠内の項目を設定します。

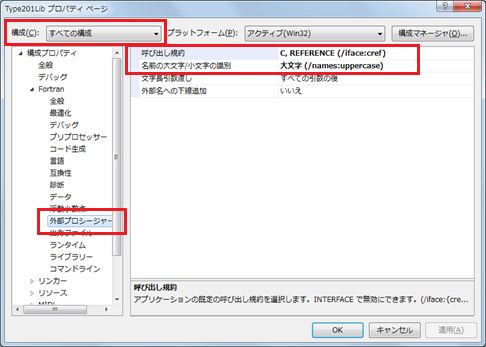
* データ



* 浮動小数点



* 外部プロシージャ

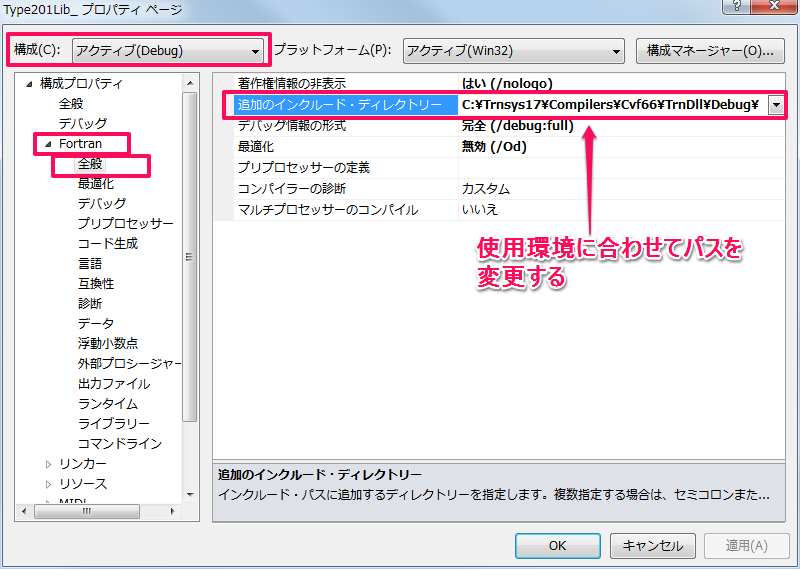


* + 1. Debug/Release設定

Debug/Releaseモードでは参照するインクルードファイル、Objファイルの参照先が変わります。

以下、Debugモードを例に説明しています。**Releaseモードでも再度、同様の設定を行なって下さい。**

* Fortranのインクルードファイル



Fortranの「全般」、「追加のインクルード・ディレクトリ」のパスを開発環境に合わせて書き換えます。以下はDebugモードの例です。Releaseモードでは対応するフォルダに読み替えて下さい。

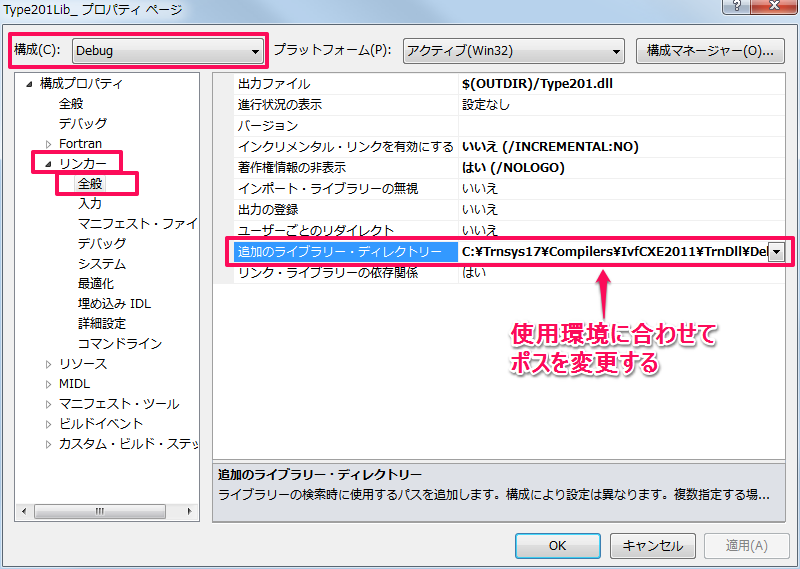
**Intel Fortran**

C:\Trnsys17\Compilers\IvfCXE2011\TRNDll\Debug

**Compaq Visual Fortran**

C:\Trnsys17\Compilers\Cvf66\TRNDll\Debug

* リンカー



リンカーの「全般」、「追加のインクルード・ディレクトリ」のパスを開発環境に合わせて書き換えます。以下はDebugモードの例です。Releaseモードでは対応するフォルダに読み替えて下さい。

**Intel Fortran**

C:\Trnsys17\Compilers\IvfCXE2011\TRNDll\Debug

**Compaq Visual Fortran**

C:\Trnsys17\Compilers\Cvf66\TRNDll\Debug

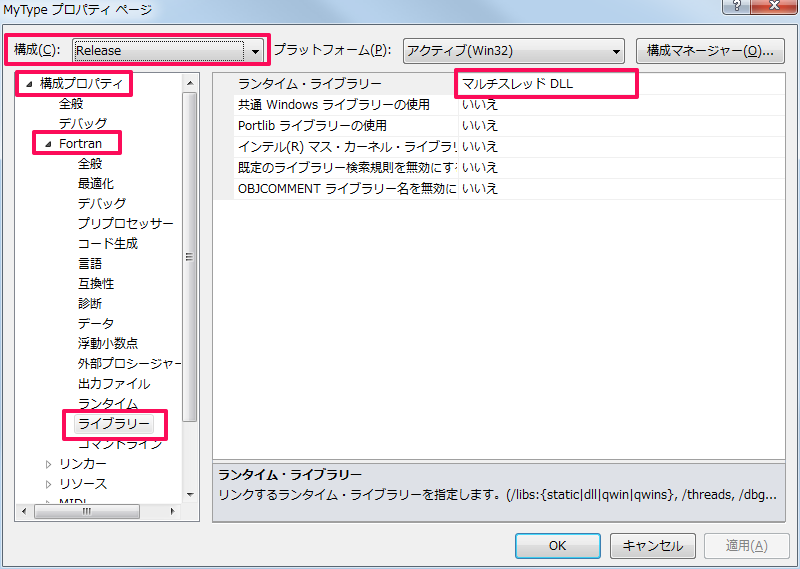
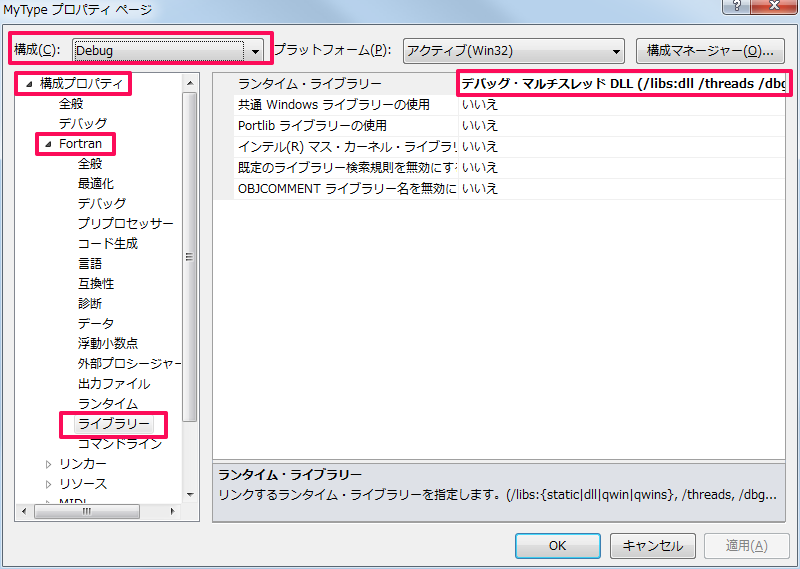
* + 1. ライブラリー

コンポーネント内からデータファイルなどにアクセスする場合は、ランタイム・ライブラリーの設定を「DLL」のタイプへ変更します。Release/Debugでそれぞれ設定を行います。（「マルチスレッド DLL」、「デバッグ・マルチスレッドDLL」）

以下参照。

TRNDLLも含め関連するプロジェクトも同じ設定に合わせる。

※ランタイム・ライブラリーでStatic（例：「マルチスレッド（/libs:static /threads）」）を選んだ場合、ファイルのLUが共有できなくなるためDLL（例：「マルチスレッド DLL」）へ変更する。



* + 1. ランタイム・ライブラリーの配布

開発環境によって必要になるランタイム・ライブラリーが異なる。開発環境、構成に合わせて適切なDLLを使用する。

|  |  |
| --- | --- |
| DLL | 開発環境 |
| MSVCRT.DLL | VS2008?  TRNDLL.DLLが既定で参照しているDLL |
| MSVCR100.DLL | VS2010 |
| MSVCR110.DLL | VS2012  例）VS2012では以下のフォルダに格納されています。  C:\Program Files (x86)\Microsoft Visual Studio 11.0\VC\redist\x86\Microsoft.VC110.CRT |

本体側（TRNDLL.DLL）が参照するランタイムと、コンポーネントが参照するランタイムが異なると、処理に齟齬を発生するケースがあります。

具体的には、データファイルを参照しているコンポーネントではTRNSYS本体が割り当てたLUがコンポーネント側から見えないといった問題となって現れます。

ファイルを参照するコンポーネントの場合は、実行環境でTRNDLL.DLLも含め、すべてのコンポーネントをビルドすれば、この問題は発生しません。

ファイルを参照しないコンポーネントのみ配布するケースでは、ランタイム・ライブラリーも含めてインストーラーでExeフォルダへ配置するようにします。

※重要：まとめると配布は以下の点に注意が必要。

|  |  |
| --- | --- |
| コンポーネント | 対処 |
| 外部ファイル参照型コンポーネント | TRNDLL.DLLを含めすべてのコンポーネントが同じランタイム・ライブラリーを参照する必要がある。（TESSも含む） |
| 外部ファイルを参照しないコンポーネント | コンポーネントに必要なDLLを同時配布すればOK |

※VSのバージョンに関わらず、同じランタイム・ライブラリーを参照すれば問題ないが、設定方法等未確認。

参考：MSDNの記載

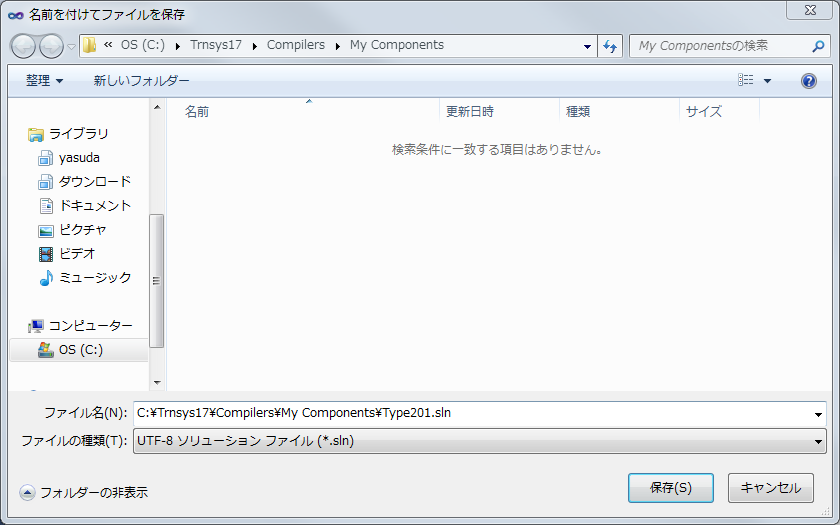
*「複数のバージョンの CRT を使用するプロジェクトでは、DLL の境界を超えて特定の CRT オブジェクト (ファイル ハンドル、ロケール、環境変数など) を渡す場合には注意を要します。 発生する可能性のある問題とその対処法の詳細については、「DLL の境界を越えて CRT オブジェクトを渡す場合に発生する可能性のあるエラー」を参照してください。」*

メモ

DESIGNNATEでファイルを処理すればOKとか？

* 1. ビルド

とりあえずビルドできるか試してみます。メニューから[ビルド]-[ソリューションのビルド]を選択してビルドします。ここでソリューションの保存先を聞いてくるので、コンポーネントと同じフォルダを指定します。



ビルドがエラーなく終わったらひとまず終了です。

次回はソースコードの編集です。

# ソースコードの編集

前回まででソースコードのスケルトン（雛形）の書き出しとコンパイラの設定が終わったので、今回はソースコードを書き換えて計算式を記述していきます。

はじめる前にちょっとプロフォルマで設定した内容のおさらいです。プロフォルマでは以下の様な設定を行なっていました。

Parameters

　　　Mult (デフォルト:1)

Inputs：

　　　Inp1

　　　Inp2

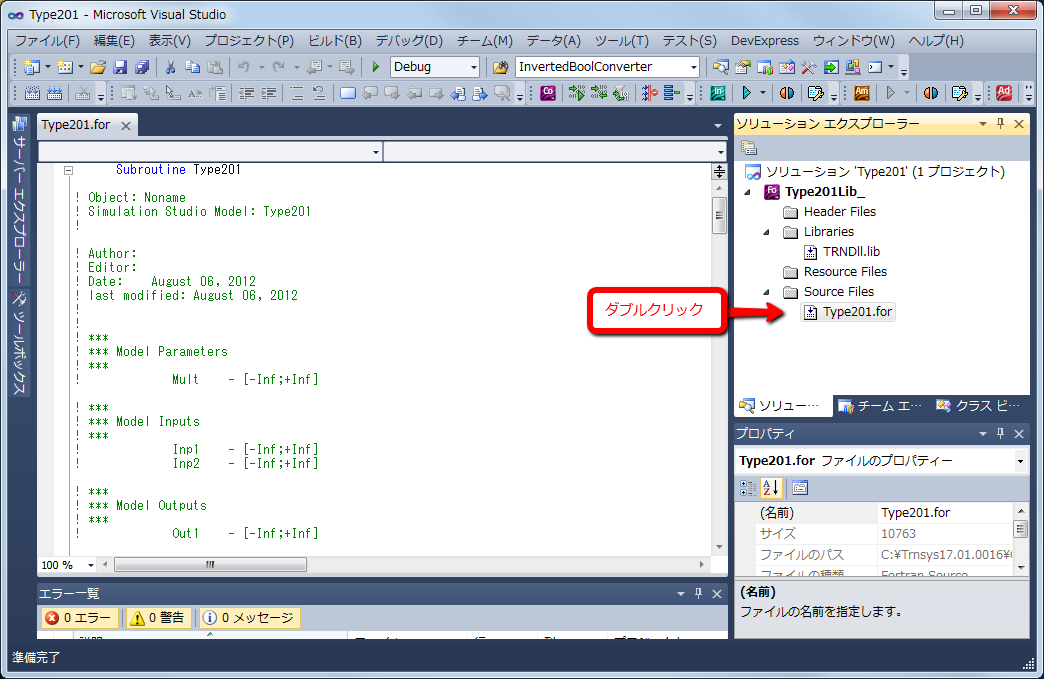
Outputs:

　　　Out1

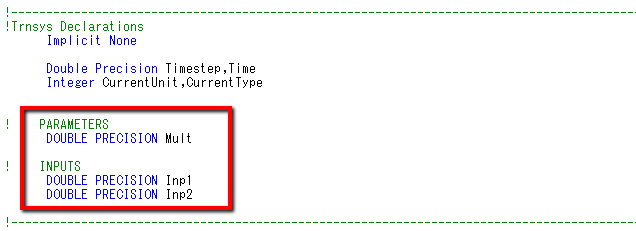
この設定を頭に入れてソースコードを見て行きます。

* 1. ソースコードの確認

まず、ソリューションエクスプローラーからType201.forをダブルクリックしてソースコードを表示します。

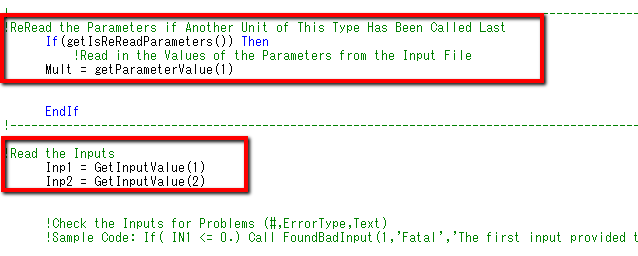


ソースコードを上から順に見ていくと60行目付近に変数の宣言箇所があります。（Ctrl+Gで直接指定行にジャンプできます。これ使うと探しやすいです）



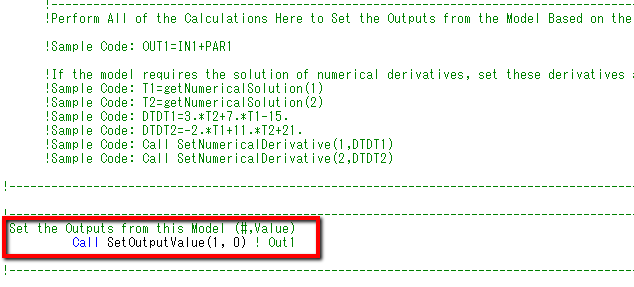
プロフォルマの設定と見比べると分かりやすいですが、Parameters, Inputsの項目として登録したものが宣言されているのがわかります。

さらに下の方を見ていくと、150行目付近にParametersとInputsに値を設定している箇所があります。この分でSimulation Studioで設定した値や他のコンポーネントから受け取った値（Inputsの値）を変数に設定しています。



でもって、最後は200行目付近にOutputを処理している箇所があります。ここで、コンポーネントの中で計算した結果をOutputsの値して書き出しています。

生成されたばかりのソースコードでは、とりあえず"0"を書きだすように指定されています。



* 1. 処理の追加

それでは、実際に処理を記述してみましょう。処理内容はInputsの値2つを足してParameterの値を掛けるシンプルなものです。

Out1 = (Inp1 + Inp2) \* Mult

* + 1. 出力用の変数の追加

まず計算結果を収める変数を用意しておきます。50行目付近に戻って計算結果を収める変数の宣言を追加します。



* + 1. 計算式の追加

200行目付近に先ほどの式を記述します。つづけてOutputの値の指定を変数に変更します。



以上で、ソースコードの変更は終了です。

コンポーネントの処理内容は、

　　　　　Parameters, Inputsの値を受け取る

　　　　　　　　　　　　　↓

　　　　　　　　　　 計算する

　　　　　　　　　　　　　↓

　　　　　　　　　   出力する

の流れの繰り返しです。簡単ですよね？

いろいろ複雑なことをやり始めると大変になってきますが、コンポーネントは基本的にはこのようにシンプルな仕組みで動いています。

* 1. ビルドする

ソースコードの変更が終わったらビルドします。

この時、メニューから[ビルド]-[構成マネージャー]を選択して、「アクティブソリューションの構成」で「Release」を選択しておきます。（DebugモードだとTRNSYSが認識してくれないので必ずReleaseでビルドしてください）

[ビルド]-[ソリューションのビルド]を選択してビルドします。エラーにならずビルドができたらコンポーネントの完成です。

次回は、実際にこのコンポーネントの動作確認をしてみます。

つづく。

# Simulation Studioで実行

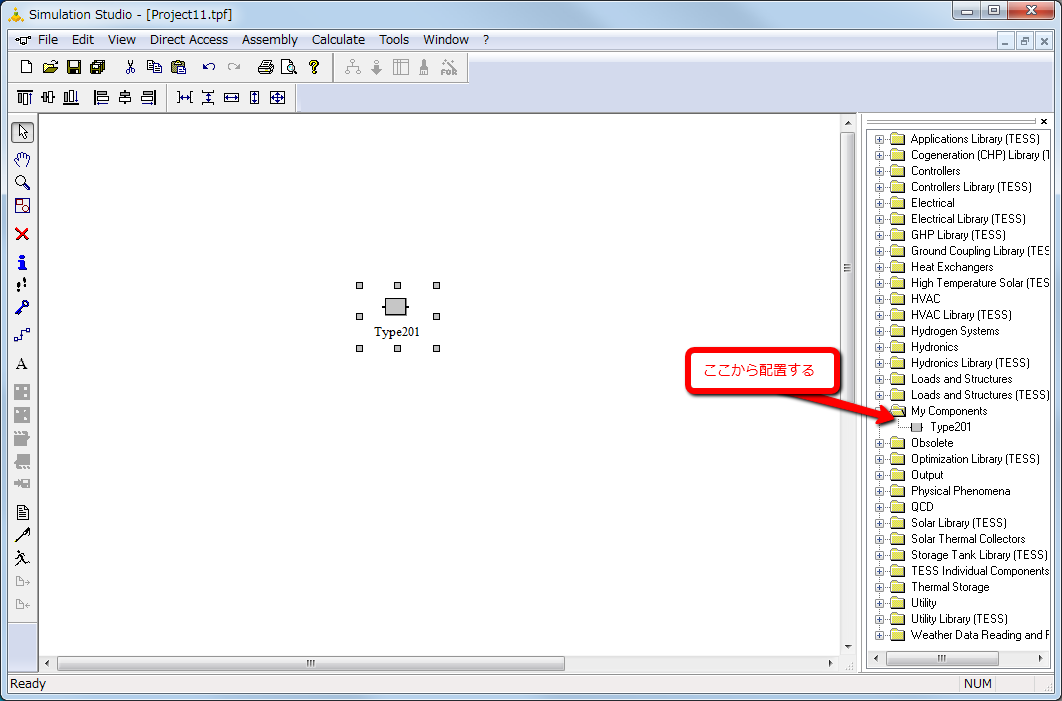
前回まででコンポーネントのビルドが済みました。これでももう計算に使える状態です。

Simulation Studioを起動して動作を確認します。

* 1. プロジェクトの準備

新しいプロジェクトを用意して、画面右側のツリーから作成したコンポーネント、Type201を配置します。（ちなみにこの右側のツリーですが、正式にはDirect Access Toolbarという名前が付いています）

もし、ツリーにType201が表示されていなければメニューから[Direct Access]-[Refresh tree]を選んで、表示を更新して下さい。



* + 1. Equationを追加する

コンポーネント単体では動作確認できませんから、Inputs側のデータを作るためのEquationを用意します。

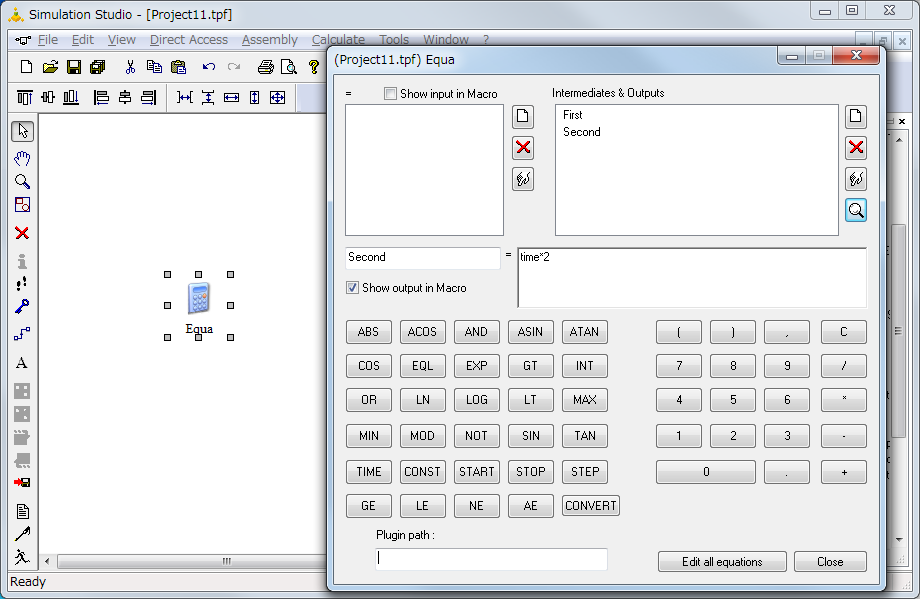
メニューから[Assembly]-[Insert new equation]を選んで、新しいEquationを作成して、以下の2つの変数を追加します。

変数             設定

-------　　　------------

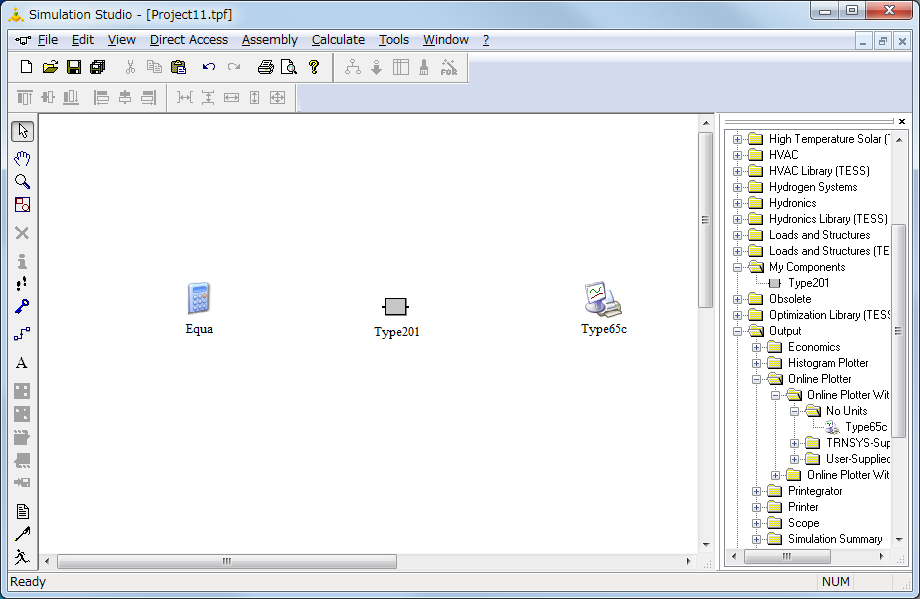
First            time

Second        time\*2



* + 1. Online Plotter

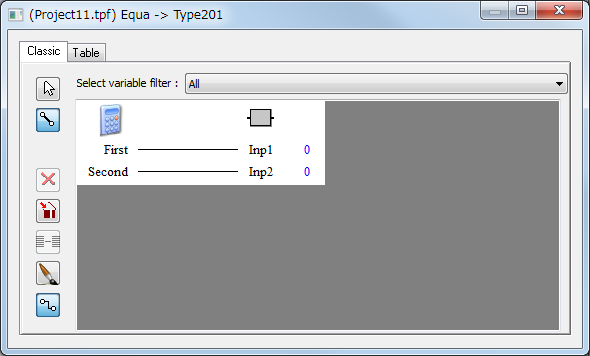
出力の確認用に、ツリーからOnline Plotterを配置しておきます。



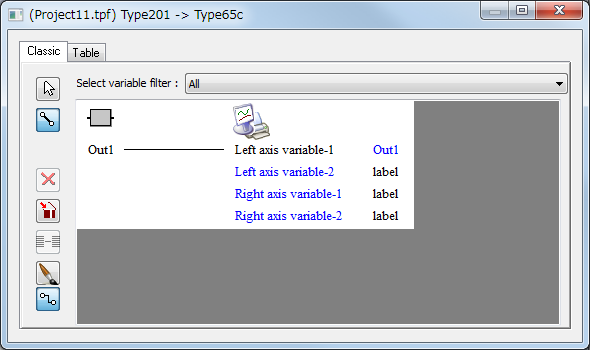
* 1. コンポーネントの接続

以下の画面のように接続します。

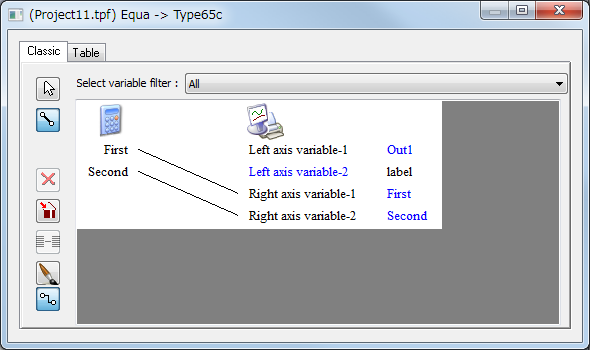
Equa->Type201



Type201->Type65c

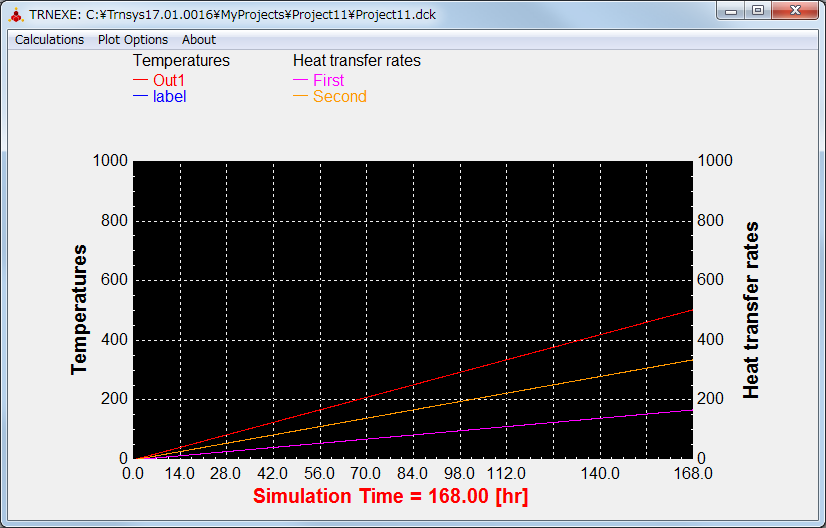


Equa->Type65c



* 1. 実行

実行して以下のようなグラフが表示されたらOKです。



* 1. まとめ

シンプルなコンポーネントを作成しましたが、複雑な処理でも基本的には同じ方法で作り込めます。コンポーネントが作れると、独自の複雑な計算や、条件判定を含んだ処理を組み込むことができるようになります。TRNSYSの計算の幅がグ、グ、グッと広がりますので、ぜひお試しください。

# デバッグ

作ってみようTRNSYSコンポーネントシリーズのおまけです。

コンポーネントの作成といっても、これはいわゆるプログラミングです。プログラミングといえば欠かせないのはデバッグ作業。という事で新しく作ったコンポーネントをデバッグしてみましょう。

コンポーネントをデバッグする場合、TRNSYS側もデバッグモードでビルドされている必要があります。具体的にはTRNDLL.DLLがデバッグモードじゃないとデバッグができません。

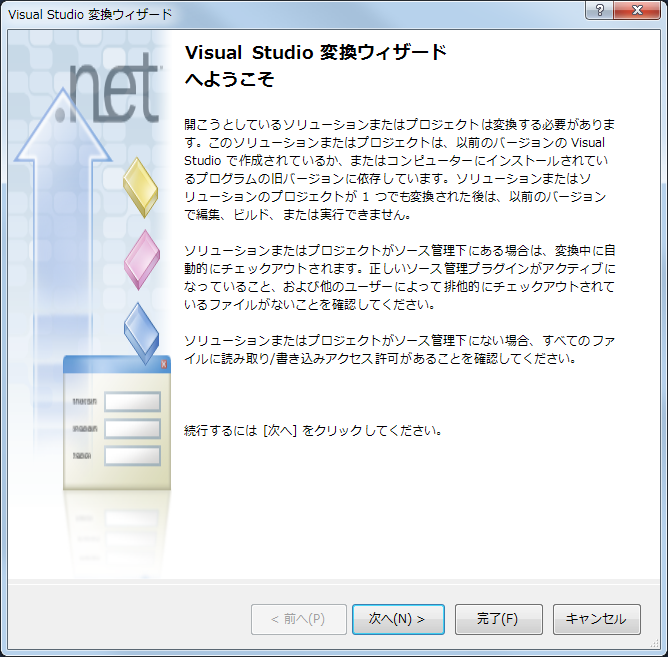
で、どうするかですが、幸いにもTRNSYSには、はじめからTRNDLL.DLLをビルドするためのソリューションが用意されています。これを使ってデバッグの準備をします。

* 1. TRNDLL.DLLのソリューションを開く

Intel Visual Fortran Composer XE 2011を起動して、以下のソリューションを開きます。

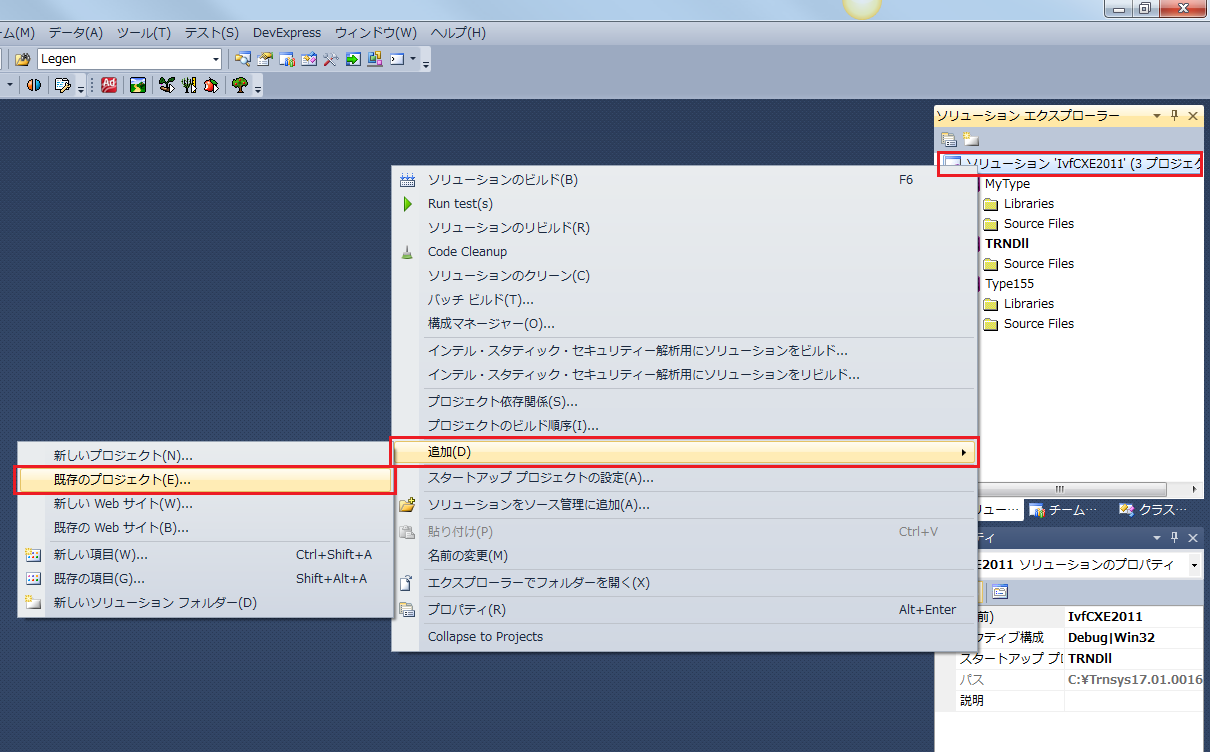
C:\Trnsys17.01.0016\Compilers\IvfCXE2011\IvfCXE2011.sln

VS2008用なのでVS2010を使われている場合、開くと変換ウィザードが立ち上がります。ここは素直に「次へ」を選択して変換を行います。

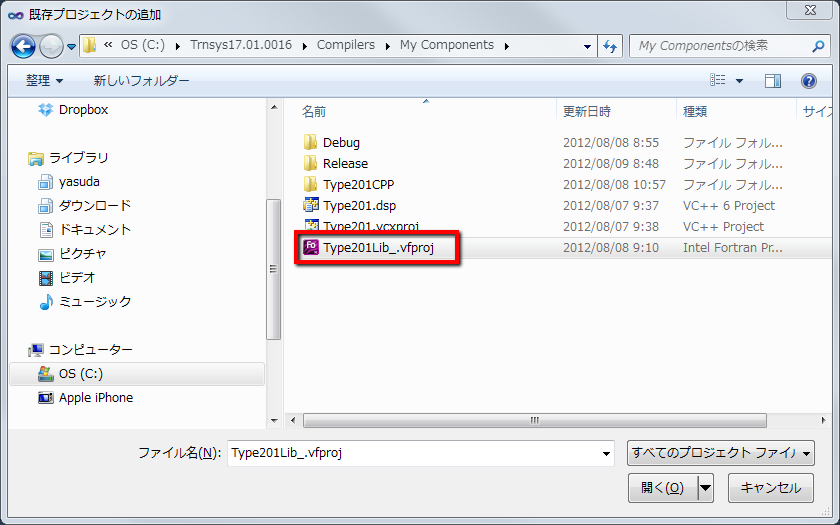


* 1. コンポーネントのプロジェクトを追加する

ソリューションが無事に開けたら、新しく作成したコンポーネントのプロジェクトを追加します。ソリューションエクスプローラーで「IvfCXE2011」を選んで右ボタンクリックで表示されるメニューから[追加]-[既存のプロジェクト]の順で選択します。



 ここで前回までに作成したコンポーネントのプロジェクトを指定します。

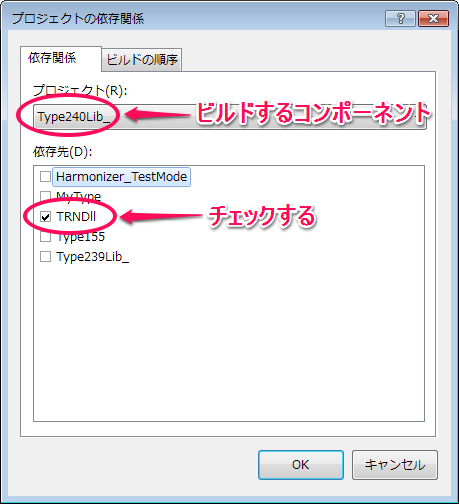
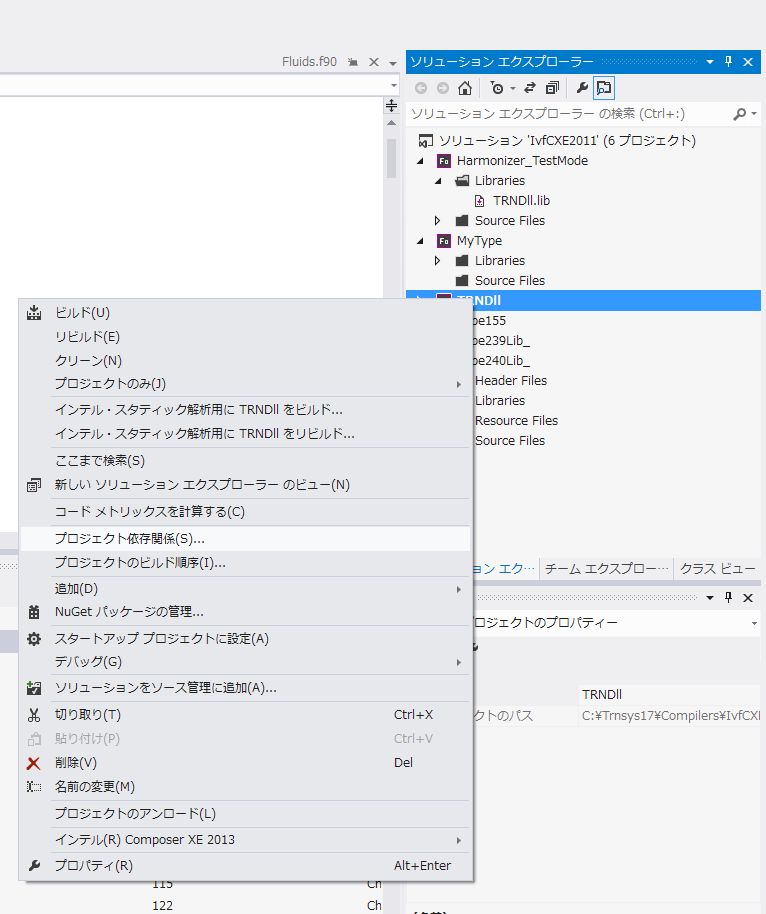


これで新しいコンポーネントとTRNDll.DLLをまとめてビルド出来るようになりました。

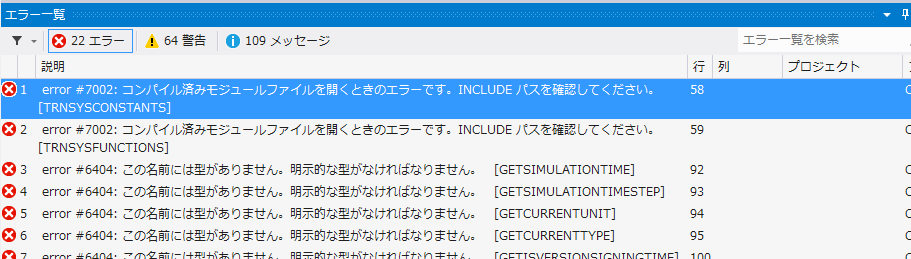
* 1. プロジェクトの依存関係を設定する

TRNDLLと追加したコンポーネントのプロジェクトに依存関係を設定します。

ソリューションを選んで右クリックで「ソリューションの依存関係」を選び、表示されるダイアログでプロジェクトの依存関係を設定します。

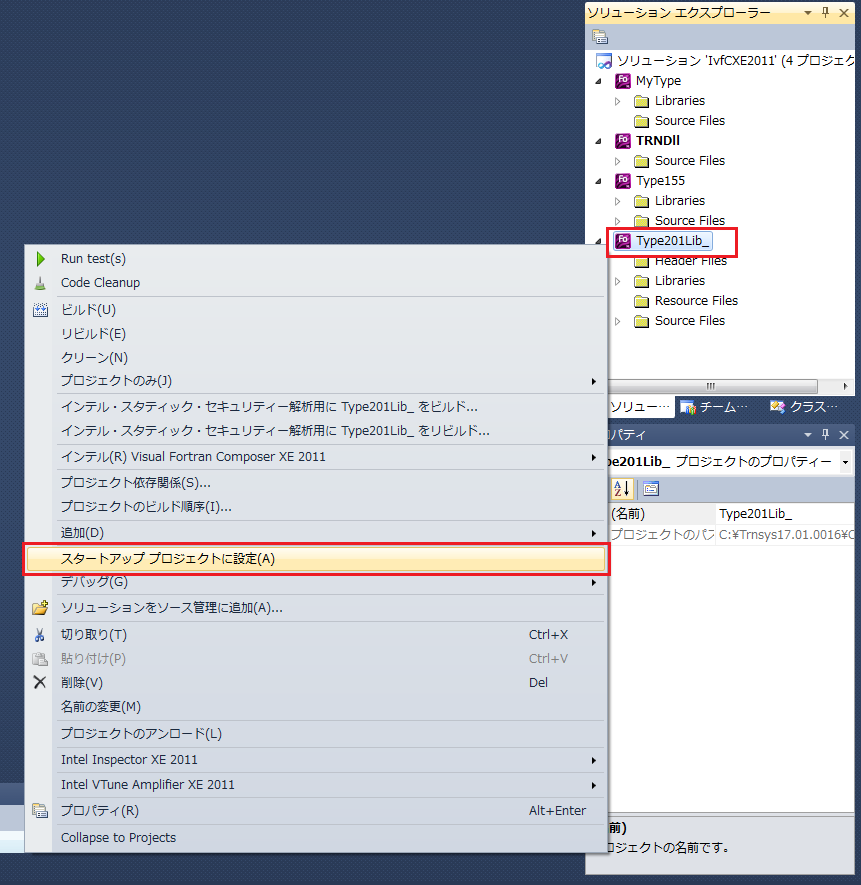


※設定しないとビルド時に次のようなエラーの原因になります。



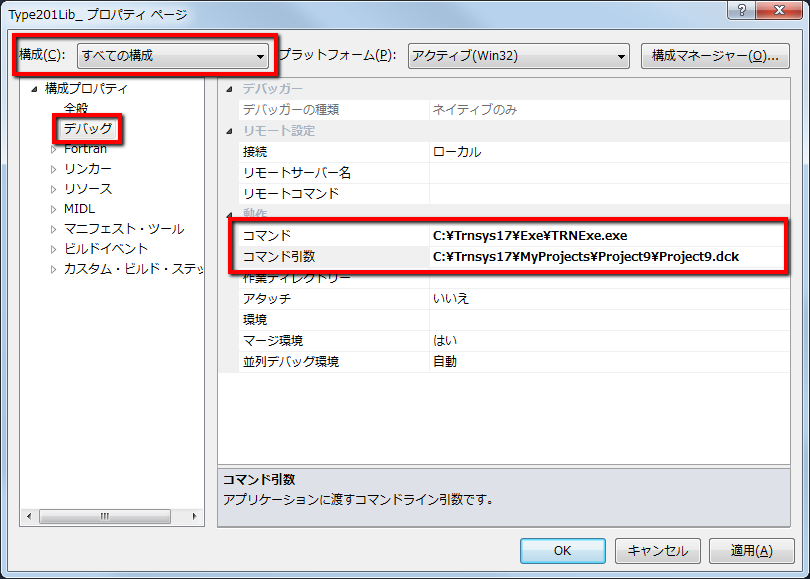
* 1. プロジェクトの設定を変更する

プロジェクトに追加されたら、右ボタンクリックで表示されるメニューから[スタートアッププロジェクトに設定]を選択します。



さらに右ボタンクリックで表示されるメニューから[プロパティ]を選んで、デバッグの設定を行います。

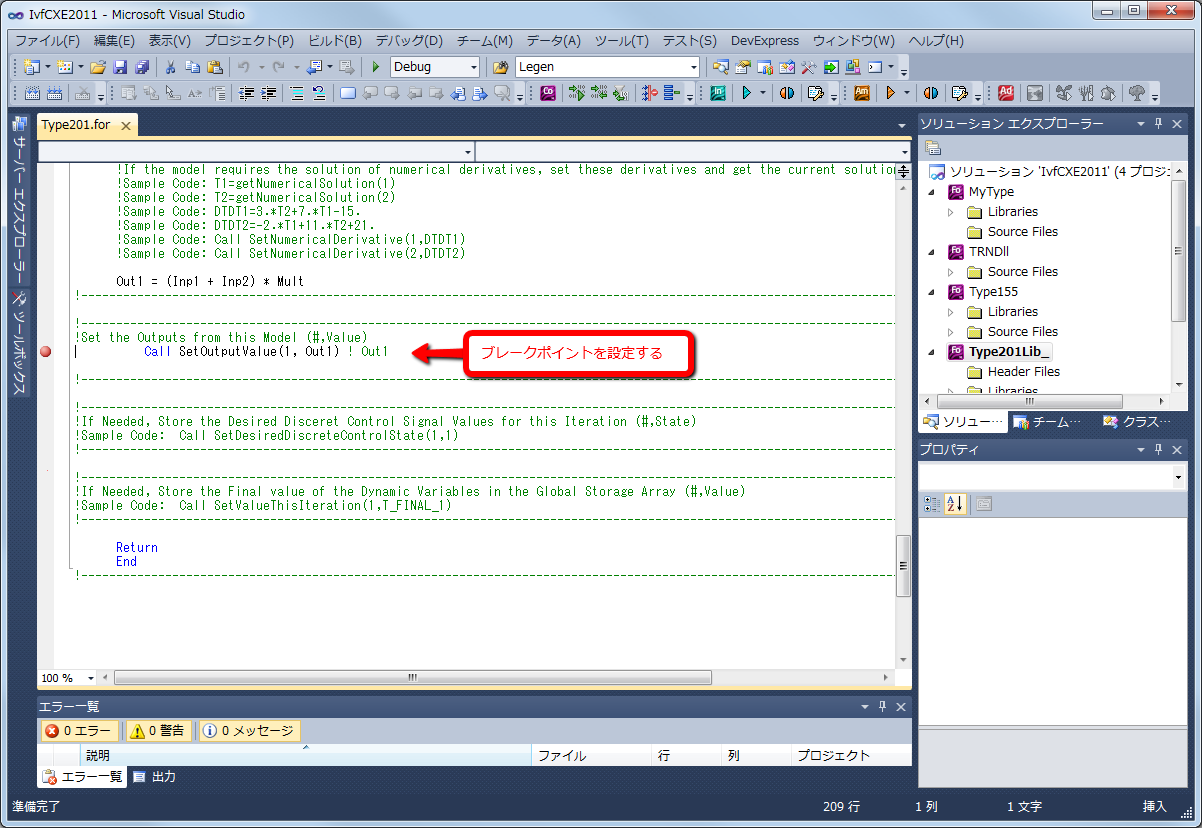
下の図のように実行用の指定を行います。  
「コマンド」の項目にはTRNSYSを実行するためのTRNexe.exeを指定します。  
「コマンド引数」の項目は、新しいコンポーネントを実行するためのDckファイルを指定します。ここでは「(5) Simulation Studioで実行」で作成したプロジェクトの\*.dckファイルを指定します。



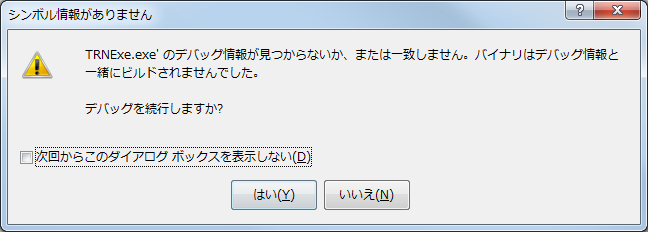
* 1. デバッグモードで実行する

[ビルド]-[構成マネージャー]を選んで、表示されるダイアログで「アクティブソリューション構成」の項目を[Debug]にしてビルドします。

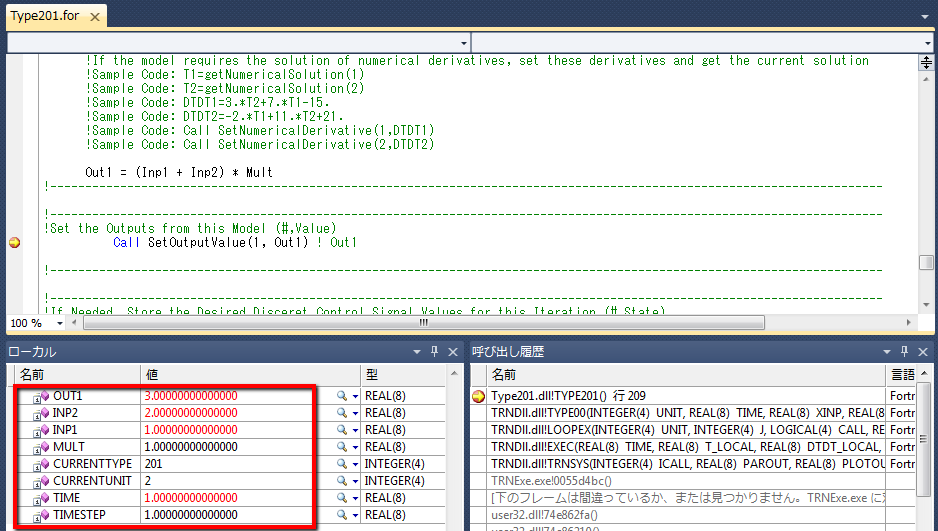
デバッグの状態を確認するため、出力値を設定している行にブレークポイントを設定します。



で、実行してみます。ここでいきなり警告メッセージが表示されます。これはTRNEXE.exeがデバッグモードでビルドされていないのが原因です。ここは警告を無視して「はい」を選んで続行します。



TRNSYSが起動して、ブレークポイントでプログラムの実行が一時停止したら、変数の内容が適切な値になっているか確認します。以下の図のような内容であればOKです。



これで何か問題があってもデバッグモードで処理内容を確認することができます。

※動作が確認できたら、必ずReleaseモードでビルドし直して下さい。

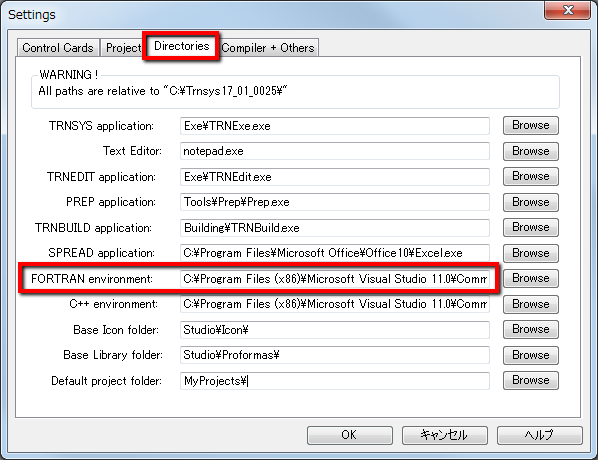
# 補足：FORTRANの環境設定

TRNSYSの設定によっては、うまく記載通りに開発環境（Intel Visual Fortran)が起動しないケースが有るようです。

その場合、FORTRANの開発環境の設定が必要になります。

* 1. 設定方法

Simulation Studioを起動して、[File]-[Settings...]で表示されるダイアログでDirectoriesタブのFORTRAN environmentの項目に開発環境を起動するパスを設定すればOKです。



以下、Microsoft Visual Studioを使用する際の設定例です。使用するVSのバージョンに合わせて設定します。インストールフォルダを変更されている場合は、適宜読み替えて下さい。

* + 1. Windows7(32bit)

VS2008

"C:\Program Files\Microsoft Visual Studio 9.0\Common7\IDE\devenv.exe"

VS2010

"C:\Program Files\Microsoft Visual Studio 10.0\Common7\IDE\devenv.exe"

VS2012

"C:\Program Files\Microsoft Visual Studio 11.0\Common7\IDE\devenv.exe"

* + 1. Windows7(64bit)

VS2008

"C:\Program Files (x86)\Microsoft Visual Studio 9.0\Common7\IDE\devenv.exe"

VS2010

"C:\Program Files (x86)\Microsoft Visual Studio 10.0\Common7\IDE\devenv.exe"

VS2012

"C:\Program Files (x86)\Microsoft Visual Studio 11.0\Common7\IDE\devenv.exe"

* + 1. Shellで起動

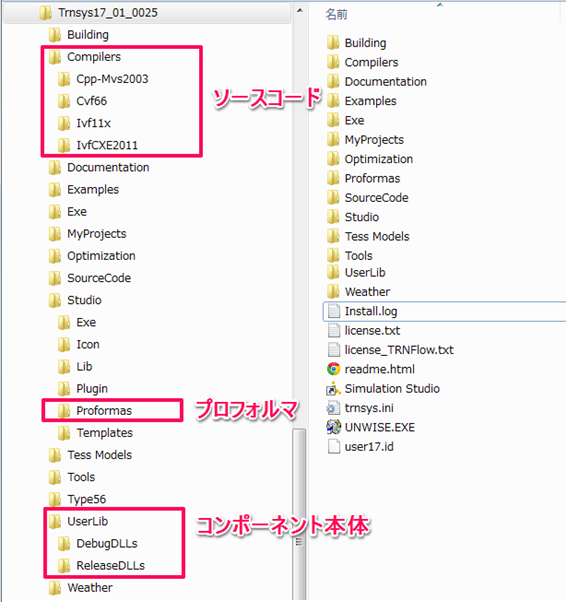
VSのShellで利用されている場合については、手元に環境がないので確認できませんでした。

おそらくスタートメニューにIntel Visual Fortranの起動用の項目が追加されていると思うので、それのリンク先をコピペすれば大丈夫だと思います。←くどいようですが手元に環境がないのでうまく行かなかったらごめんなさい。

# 補足：関連ファイル

* 1. フォルダ構成

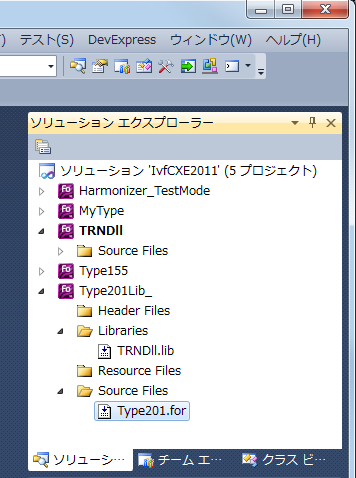
コンポーネントの開発では以下の3つのフォルダで作業を行います。



* 1. ソースコード

コンポーネントのソースコード。インテル(R) Visual Fortran Composer XE 2013 Windows版で作成され、以下のファイルを含みます。

* ソリューション(\*.sln)
* プロジェクト(\*.vfproj)
* ソースコード(\*.for)



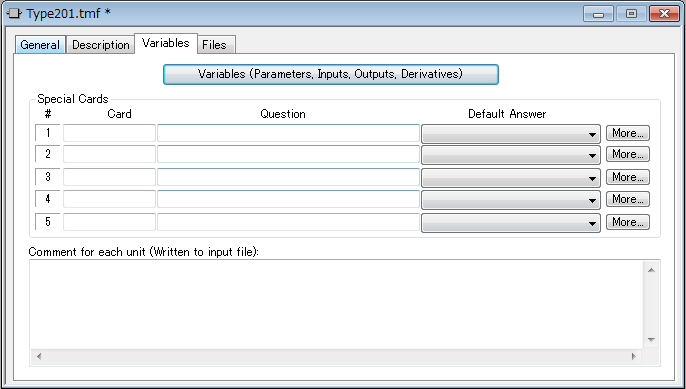
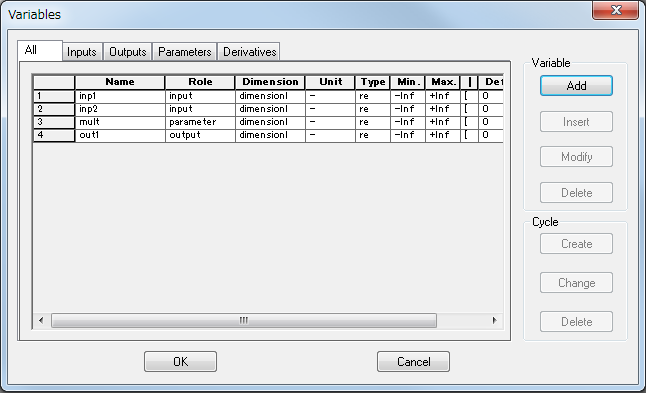
プロジェクト(\*. vfproj)

ソースコード(\*.for)

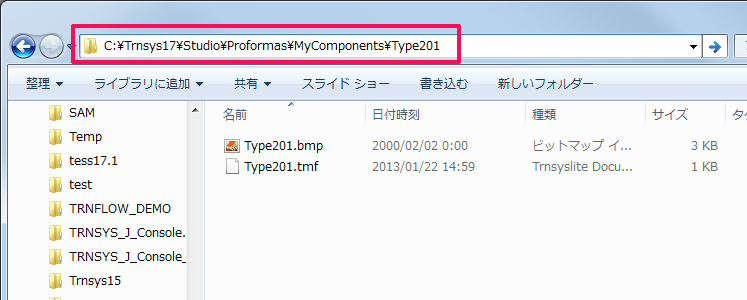
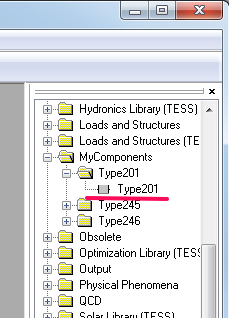
ソリューション(\*.sln)

* 1. プロフォルマ

「C:\TRNSYS17\Studio\Proformas」フォルダに配置され、Simulation Studioから利用されます。



配置先のフォルダのパスがそのままSimulation Studioの画面左側のツールバー（Direct Access Toolbar）に表示されます。



* 1. DLL

コンポーネントの実態。DLL（Dynamic Link Library形式のファイル）としてTRNSYSから呼び出されます。

ビルドされたモード（Release/Debugモード）によってDLLの配置先は以下のようになります。

|  |  |
| --- | --- |
| ビルドモード | フォルダ |
| Release | C:\Trnsys17\UserLib\ReleaseDLLs |
| Debug | C:\Trnsys17\UserLib\DebugDLLs |

参考：TRNSYSのシステム構成

TypeNNN.dll

TRNEXE.exe

TRNDLL.dll

Kernel

標準コンポーネント

TypeNNN.dll

TypeNNN.dll

カスタムコンポーネント

このファイルがReleaseモードなのか、Debugモードなのかによって参照先のフォルダがReleaseDLLs、もしくはDebugDLLsに変わる。

# 補足：Fortran77/90形式の切り替え

FORTRANのソースコードの形式には固定形式と自由形式があります。前者はいわゆるFortran77互換の形式、後者はFortran90です。

Intel Fortranではどちらの形式にも対応しており、デフォルトではファイル拡張子で区別しています。

【拡張子による判定】

|  |  |
| --- | --- |
| 形式 |  |
| 固定形式（Fortran77） | \*.for、\*.f |
| 自由形式（Fortran90） | \*.f90 |

TRNSYSが書きだすコンポーネントの雛形は拡張子\*.forで書きだされます。つまり、Fortran77形式です。

拡張子によって形式が区別されるため、あるソースコードではコンパイルできるのに、同じ書き方を別のソースコードではエラーになる事があります。例えば、拡張子によって、一行72文字を超えたコードがソースコードによってビルドできたり、できなかったりします。

自由形式の方が書き方の自由度が高いのでオススメです。

**ヒント！！**

新旧のソースコードが混在するときは、拡張子で区別されるので通常はデフォルトの設定で問題ありません。

一括してどちらかの形式を指定したい場合は、プロパティダイアログの以下の項目で設定します。

