# C++版TRNSYSコンポーネントのサンプル

* 1. 内容

TypeStudioのテキストと同じコンポーネントのC++版の作成例。

「C:\TRNSYS18\Compilers」フォルダへ展開して下さい。

また、試用する場合は以下のリポジトリのプロフォルマ、プロジェクトを使用してください。

* プロフォルマ

https://github.com/TRNSYSJP/TRNSYS.JP/tree/master/TRNSYS18/Studio/Proformas/MyComponents/Type201Heater

* プロジェクト（動作確認用）

https://github.com/TRNSYSJP/TRNSYS.JP/tree/master/TRNSYS18/MyProjects/Type201Heater

* 1. 参考

TRNSYS TypeStudioのテキストを公開

<https://www.kankyoukei.com/2018/11/trnsystypestudio.html>

TypeStudioで作るTRNSYSコンポーネント（１）

<https://www.kankyoukei.com/2018/11/typestudiotrnsys-2.html>

* 1. 注意事項
     1. TypeStudio版との競合

すでにTypeStudioのサンプルを試されている場合は、C:\TRNSYS18\UserLib\ReleaseDLLs、およびDebugDLLsフォルダにType201\_x64.dllが作成されています。

C++版とType番号が重複しますので、予め削除して下さい。

|  |  |
| --- | --- |
| サンプル | DLLファイル名 |
| TypeStudio版 | Type201\_x64.dll |
| C++版（Visual Studio） | Type201HeaterCpp.dll |

* 1. 動作環境

以下の環境で動作を確認しています。

Windows10 Pro(64bit, 1909)

TRNSYS18.01.0001(64bit)

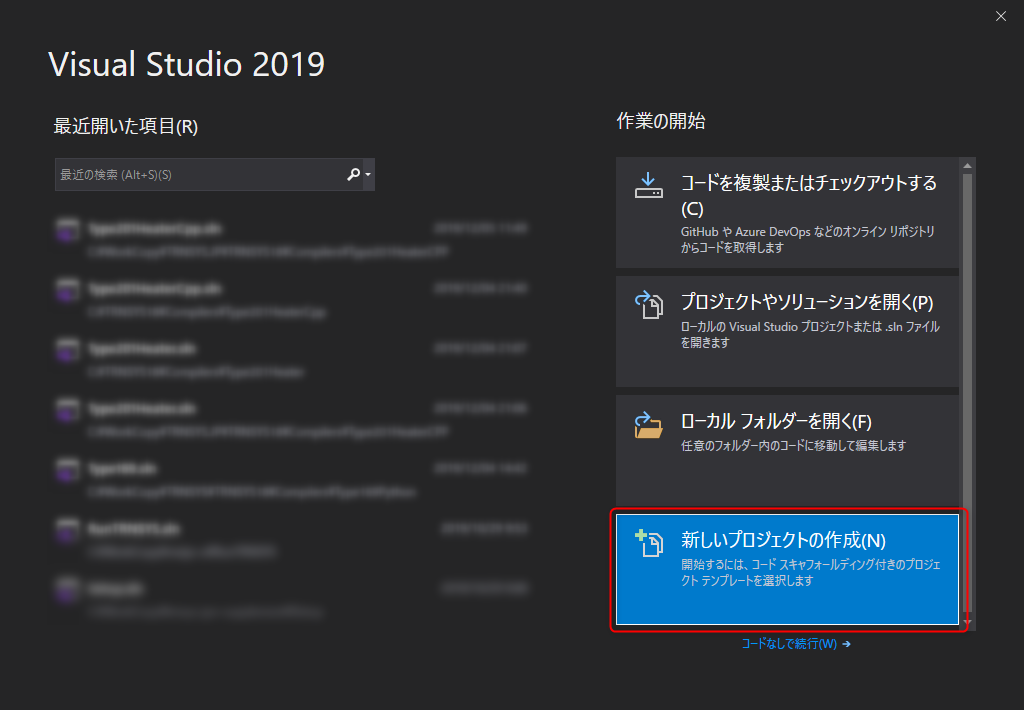
Microsoft Visual Studio 2019 Professional ver16.3.8

* 1. 新規プロジェクトの準備

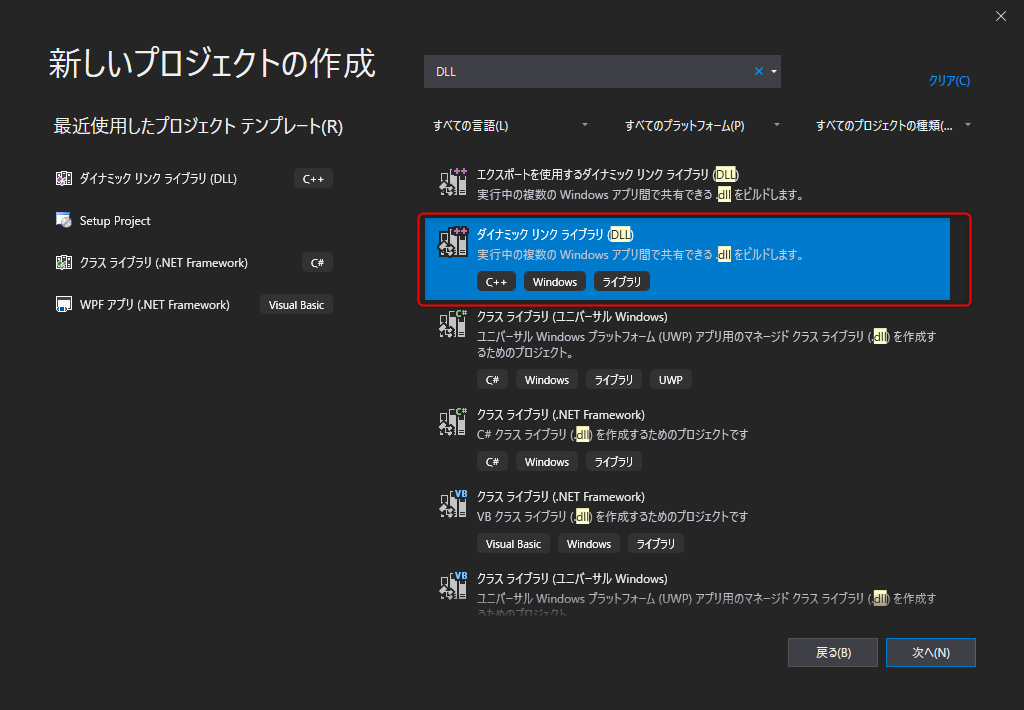
コンポーネントの開発用にDLL(Dynamic Link Library)のプロジェクトを作成して、コンポーネントのソースコードを追加する。

* + 1. 新しいプロジェクトの作成

1. Visual Studioを起動して、「新しいプロジェクトの作成」を選択する。



1. 「ダイナミックリンクライブラリ」を選択して「次へ」



1. プロジェクト名、場所を指定して「作成」をクリック。



* + 1. ソースコードの削除と追加

1. 不要ファイルの削除

デフォルトで作成される以下のファイルは不要なので、すべて削除する。

* 「ソースファイル」フォルダ

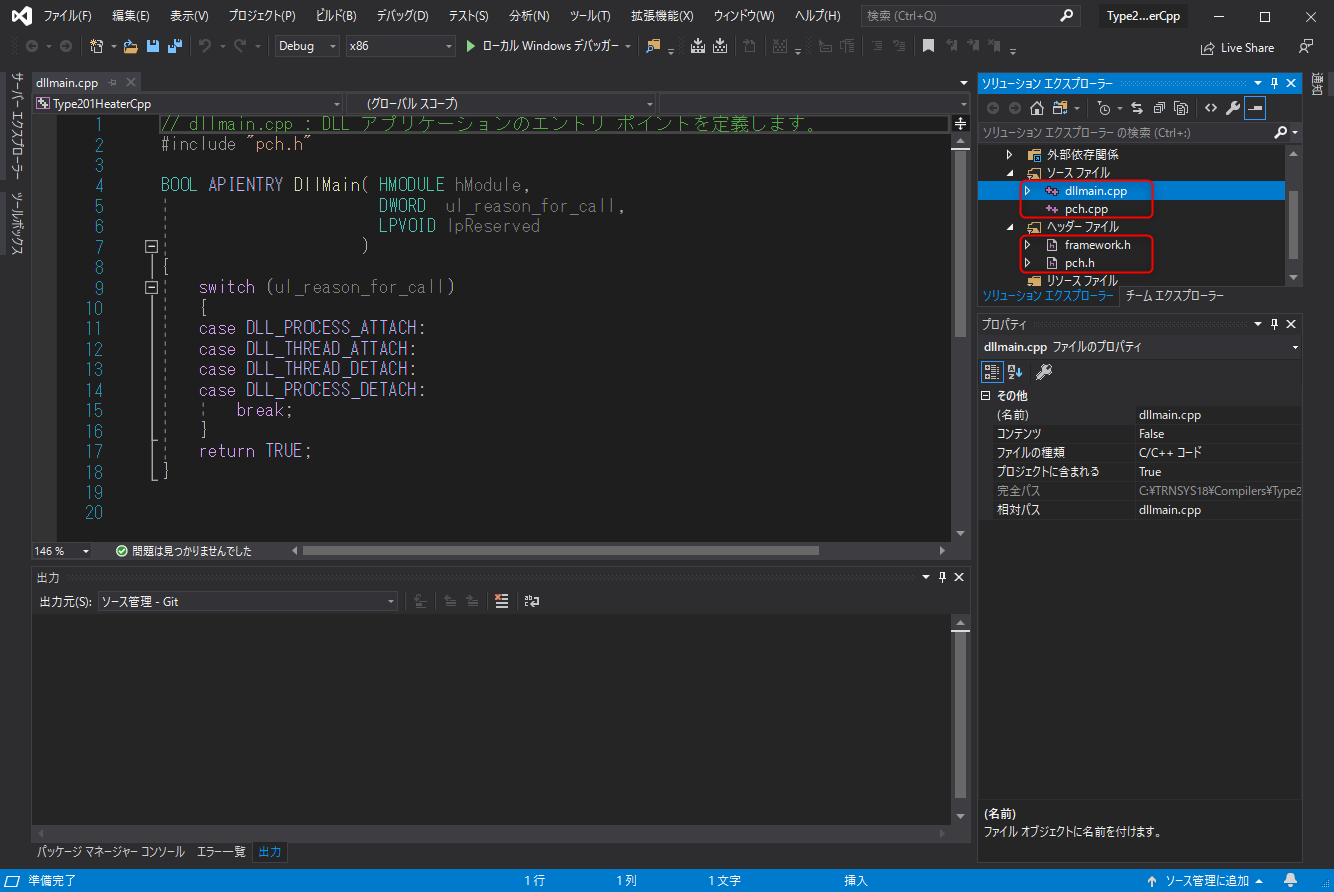
dllmain.cpp

pch.cpp

* 「ヘッダーファイル」フォルダ

framework.h

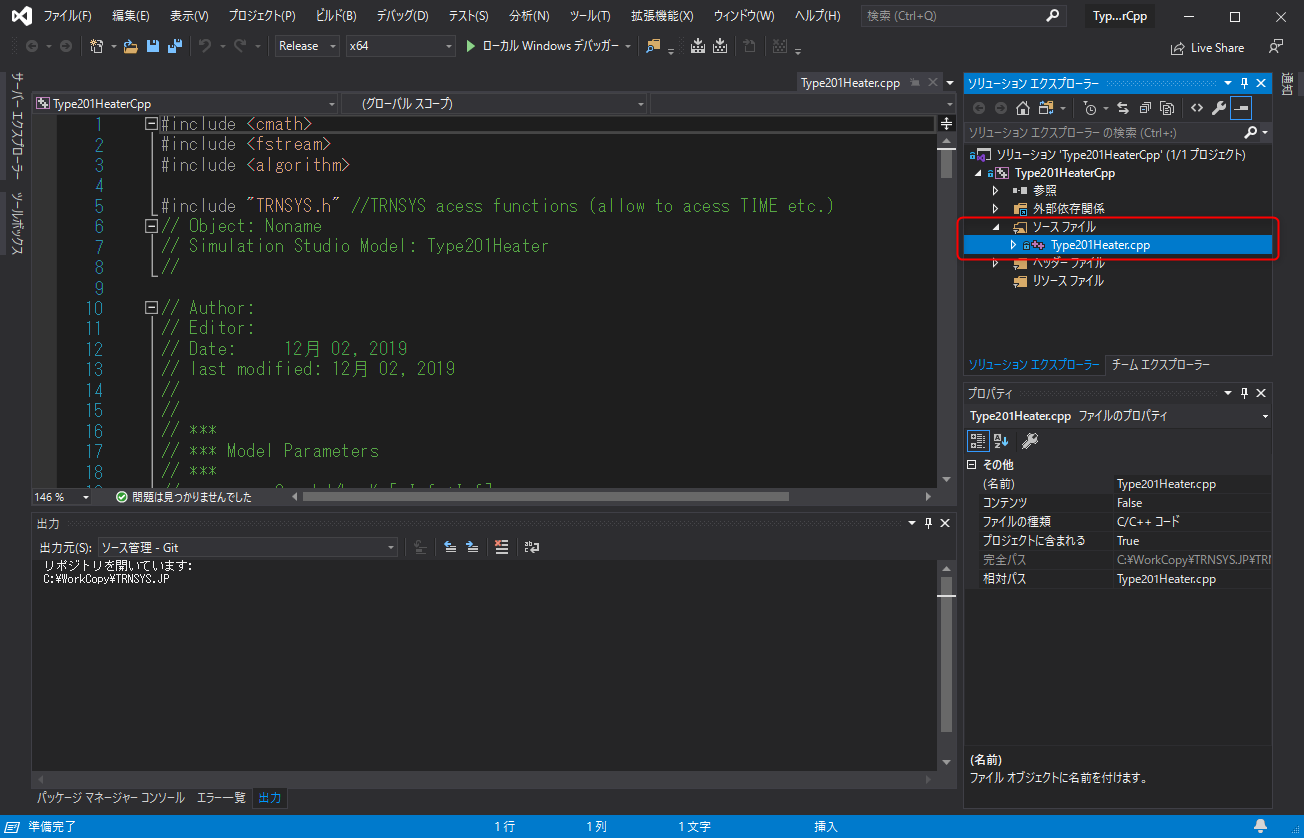
pch.h



1. ソースコードの追加

Simulation Studioで作成したプロフォルマからエクスポートしたソースコード（この例では「Type201Heater.cpp」をプロジェクトフォルダへコピーする。※

「ソースフォルダ」で右クリックして、[追加]-[既存の項目]でType201Heater.cppを追加する。



ビルド関連の設定、ヘッダーファイル、ライブラリの指定については次頁移行で解説します。

※ソースコード

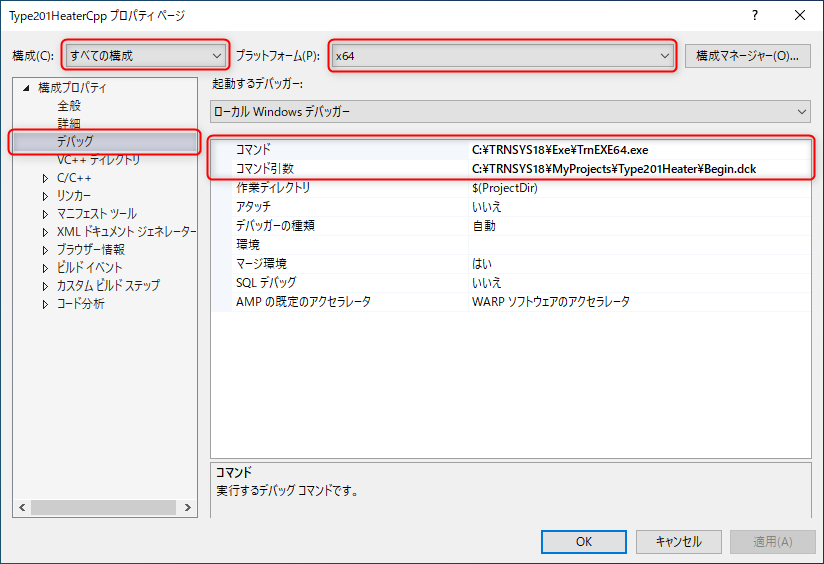
ソースコードの作成については以下を参照

TRNSYS18 - TypeStudioで作るTRNSYSコンポーネント.docx

<https://github.com/TRNSYSJP/TRNSYS.JP/blob/master/Docs/TRNSYS18%20-%20TypeStudio%E3%81%A7%E4%BD%9C%E3%82%8BTRNSYS%E3%82%B3%E3%83%B3%E3%83%9D%E3%83%BC%E3%83%8D%E3%83%B3%E3%83%88.docx>

* 1. プロジェクトプロパティ（設定）
     1. デバッグ

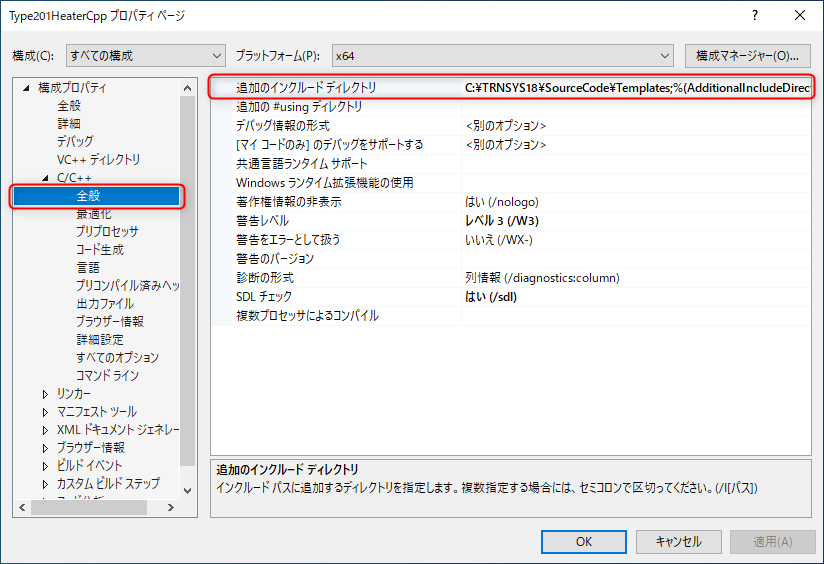
Visual Studioから直接実行する場合は、[プロジェクト]-[プロパティ]で、「構成プロパティ」の「デバッグ」の項目で、「コマンド」、「コマンド引数」を環境に合わせて修正して下さい。



以下、赤枠の部分に注意して変更を行って下さい。

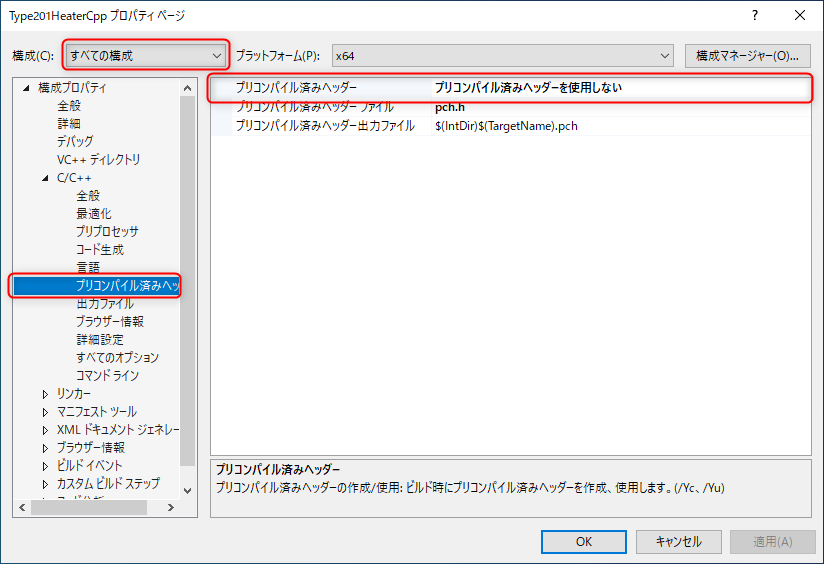
* + 1. C/C++
* 全般

「追加のインクルードディレクトリ」へ「C:\TRNSYS18\SourceCode\Templates」を追加



* プリコンパイル済みヘッダー

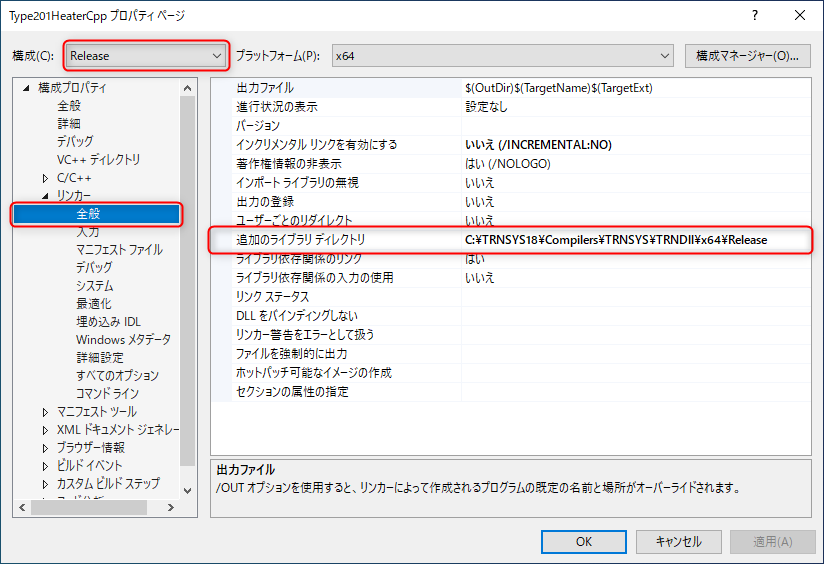
「プリコンパイル済みヘッダーを使用しない」を選択



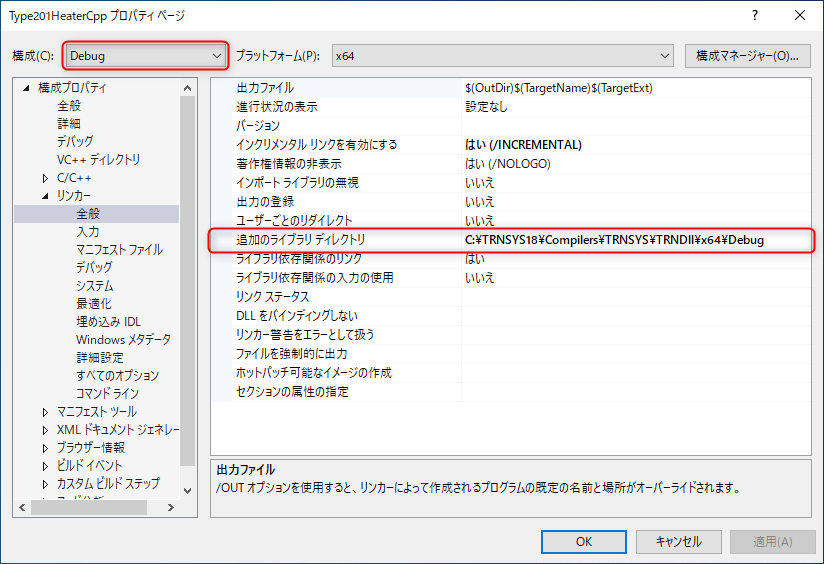
* + 1. リンカー
* 全般

「Release」の構成で「追加のライブラリディレクトリ」へ

「C:\TRNSYS18\Compilers\TRNSYS\TRNDll\x64\Release」を追加

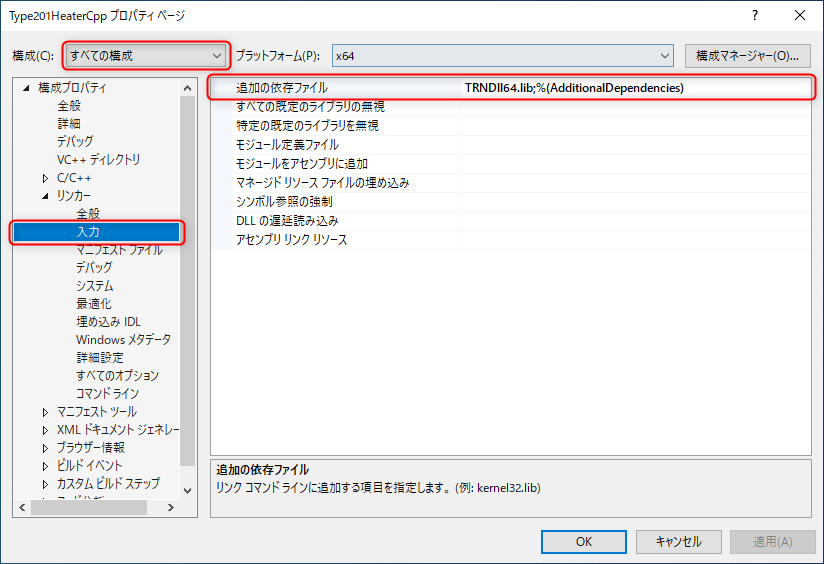


「Debug」の構成へは「C:\TRNSYS18\Compilers\TRNSYS\TRNDll\x64\Debug」を追加



* 入力

「追加の依存ファイル」へTRNSYSのライブラリ「TRNDll64.lib」を追加



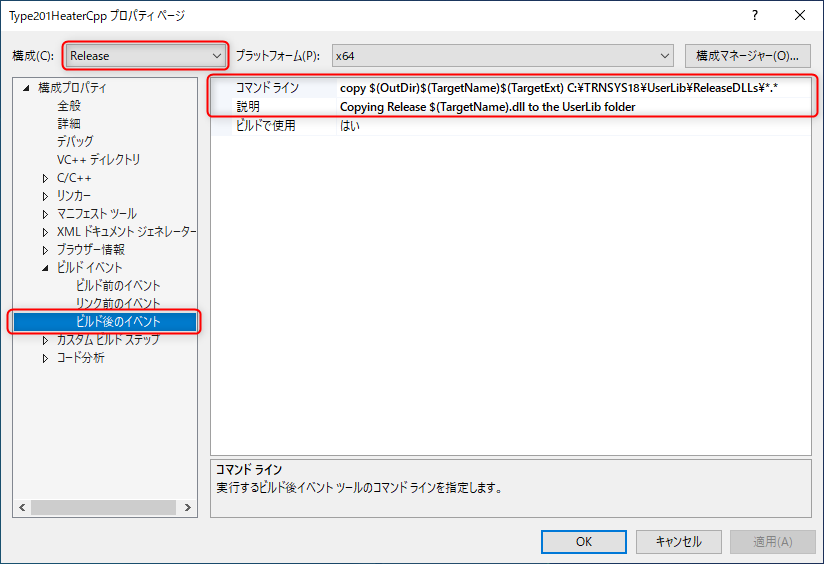
* + 1. ビルドイベント

ビルド後にDLLをTRNSYSのフォルダへコピーします。

「Release」の構成で「コマンドライン」、「説明」へ以下を追加する。

copy $(OutDir)$(TargetName)$(TargetExt) C:\TRNSYS18\UserLib\ReleaseDLLs\\*.\*

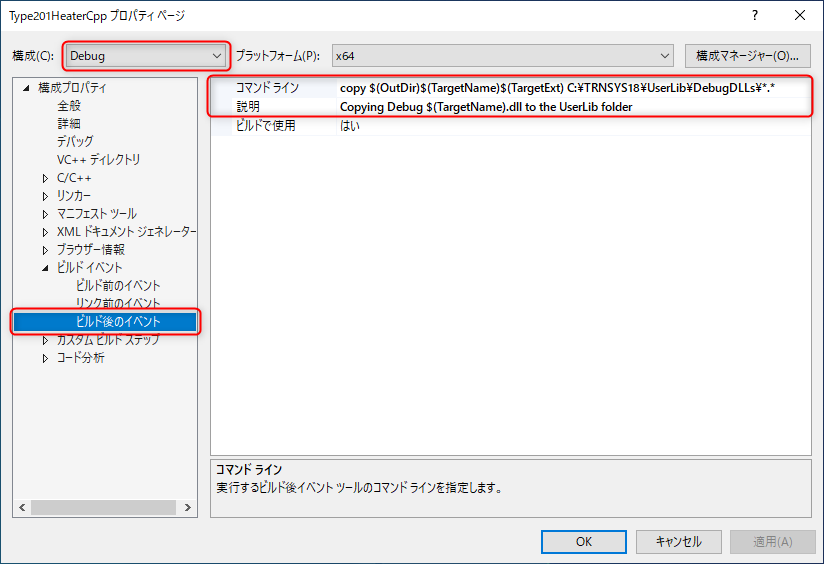
Copying Release $(TargetName).dll to the UserLib folder



「Debug」の構成で「コマンドライン」、「説明」へ以下を追加する。

copy $(OutDir)$(TargetName)$(TargetExt) C:\TRNSYS18\UserLib\DebugDLLs\\*.\*

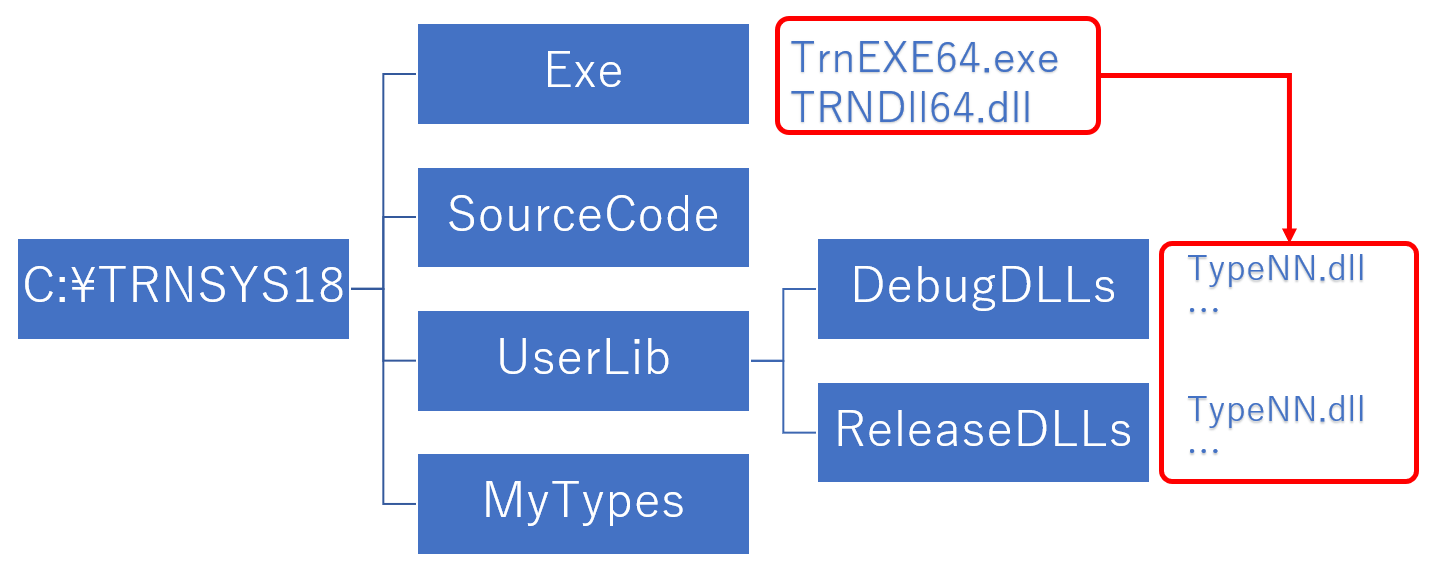
Copying Debug $(TargetName).dll to the UserLib folder



* 1. デバッグ
     1. DLL（コンポーネント）の配置先

TRNSYSのコンポーネントはビルドされたのがReleaseかDebugによってDLLの配置先が異なります。

|  |  |
| --- | --- |
| ビルド | 配置先 |
| Release | C:\TRNSYS18\UserLib\ReleaseDLLs |
| Debug | C:\TRNSYS18\UserLib\DebugDLLs |



どちらのフォルダのDLLが使用されるかは、ExeフォルダのTRNDll64.dllがRelease、Debugのどちらでビルドされているかによります。

TRNDll64.dll がReleaseでビルドされていればReleaseDLLsフォルダが、DebugでビルドされてればDebugDLLsフォルダが参照されます。

TRNSYSに添付されるTRNDll64.dllはReleaseでビルドされています。このため、標準の状態では常にReleaseDLLsフォルダが参照されます。

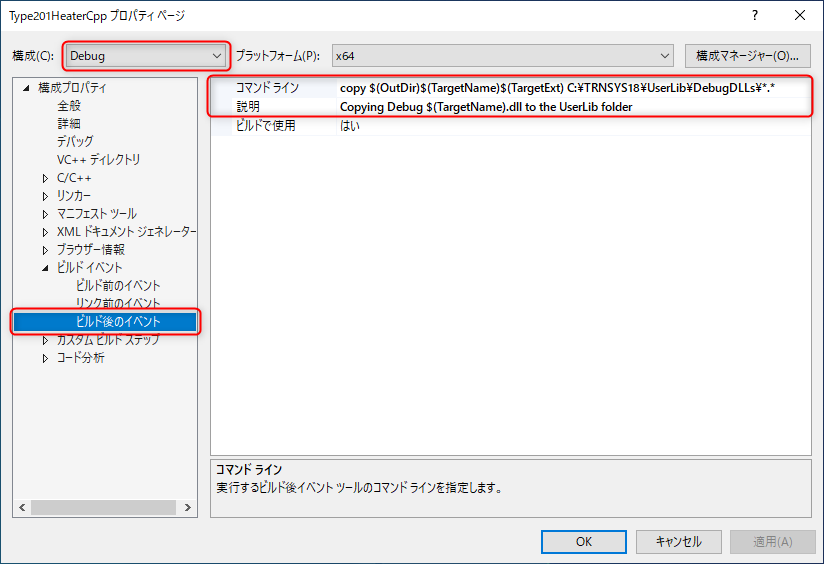
* + 1. デバッグ方法

C++のコンポーネントをデバッグする場合、以下のいずれかの対応が必要になります。

* TRNDll64.dllをDebugでビルド

Intel FORTRANを使ってTRNDll64.dllをDebugでビルドする。

* DebugでビルドしたDLLをReleaseDLLsフォルダへ配置する。



注意：TRNDll64.dllはReleaseのままなので、TRNSYS本体のデバッグは出来ません。（希に、TRNSYS本体とコンポーネントのデータのやり取りを調べたいケースがありますが、その場合はDebugでビルドしたTRNDll64.dllが必須です）