



pure::variants - MATLAB/Simulink コネクタ

pure::variants はバリエーションの定義とバライアビリティのモデリングを開発全フェーズでサポートし、既存のツールはバライアビリティとバリエーションを効率的に扱えるように増強されます。

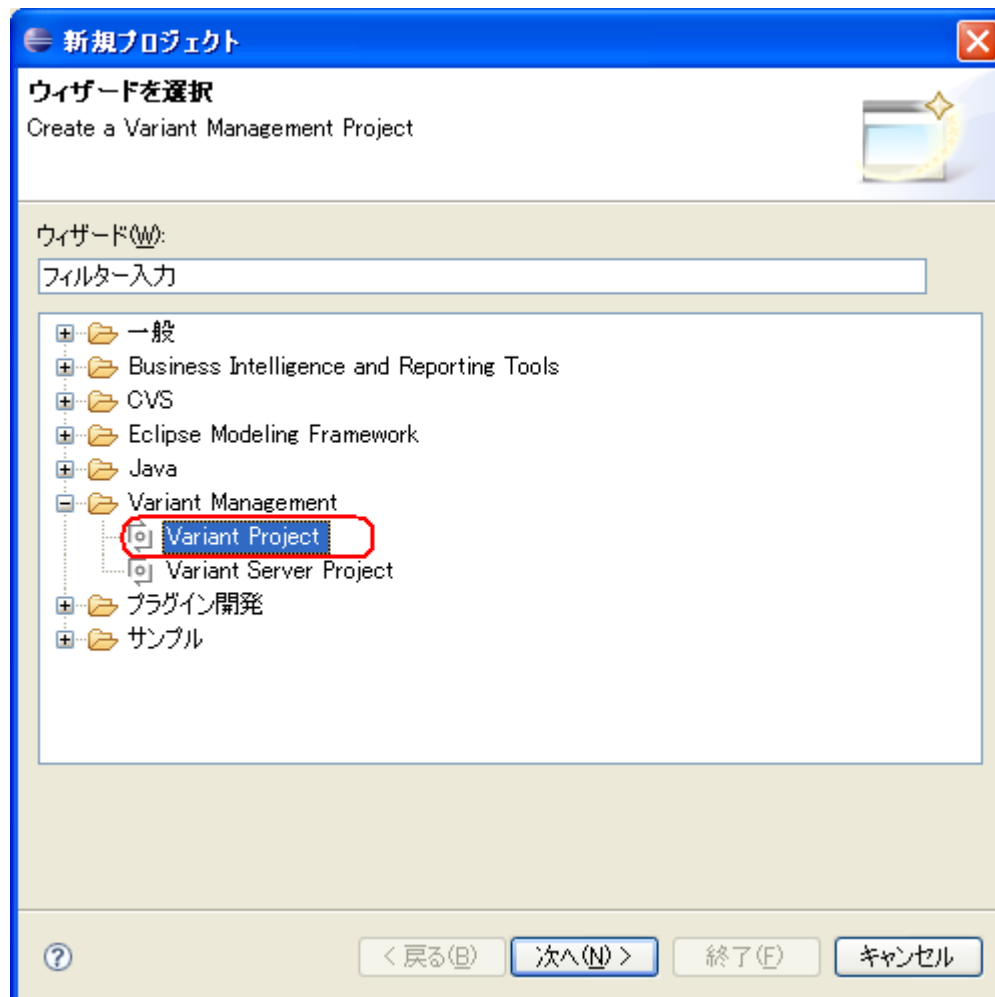
このドキュメントでは、pure::variants に MATLAB/Simulink モデルをファミリーモデルとしてインポートし、これらをフィーチャモデルの各バリエーションに関連付けさせて、プロダクトラインのコア資産として再利用できるようにするまでの過程を、例を用いて紹介します。

The screenshot displays the Eclipse IDE interface for Variant Management. The main window shows a project named "Sample Config Space with Transformation.vdm" with a tree view of feature models. The "Feature Models" tab is active, showing a hierarchy of features like "Safety Functions", "Brakes", "Engine", "Diesel", and "Gasoline". The "Relations" tab shows a list of features and their dependencies. A "Properties" window shows 2 errors, 0 warnings, and 0 infos. A "Scope" window shows a graph of a signal generator and a gain block. A "Scope" window shows a plot of a signal over time. A "Scope" window shows a Simulink block diagram with a signal generator, gain, and scope block. A blue arrow points from the Simulink block diagram to the Scope window showing the plot.

Description	Resource	In Folder
'ESP' require(s) 'ABS'	Sample Confi...	Simple Car Example/S
open alternatives are 'Diesel', 'Gasoline'	Sample Confi...	Simple Car Example/S



- 1 . `pure::variants` を起動し、[ファイル]メニューから、[新規] - [プロジェクト]を選択します。
- 2 . **Variant Management** から **Variant Project** を選択し、[次へ]ボタンを押下します。



- 3 . **Project name** を入力します。 **Project contents** では、 **Use default** にチェックが入っています。この場合、マイドキュメント配下の **pure-variants-workspace-2.4** フォルダ内にプロジェクトが生成されます。プロジェクトの場所を任意で設定する場合は、 **Use default** のチェックを外してディレクトリを指定します。

Project type を選択します。

- ・ **Standard** は、フィーチャモデル、ファミリーモデル、コンフィグレーションスペース（標準変換に接続されるバリエーション記述モデル）を含んでいるプロジェクトを作成します。全てのモデルは、プロジェクト名と同じ名前になります。
- ・ **Custom** は、名前、型、生成されるモデルの数などの定義を許可します。
- ・ **Empty** は、どのモデルもない空のプロジェクトを生成します。

ここでは、 **Empty** を選択します。 [終了] ボタンを押下します。

New Variant Management Project

Variant Project
Create new variant project

Project name

Project contents

Use default

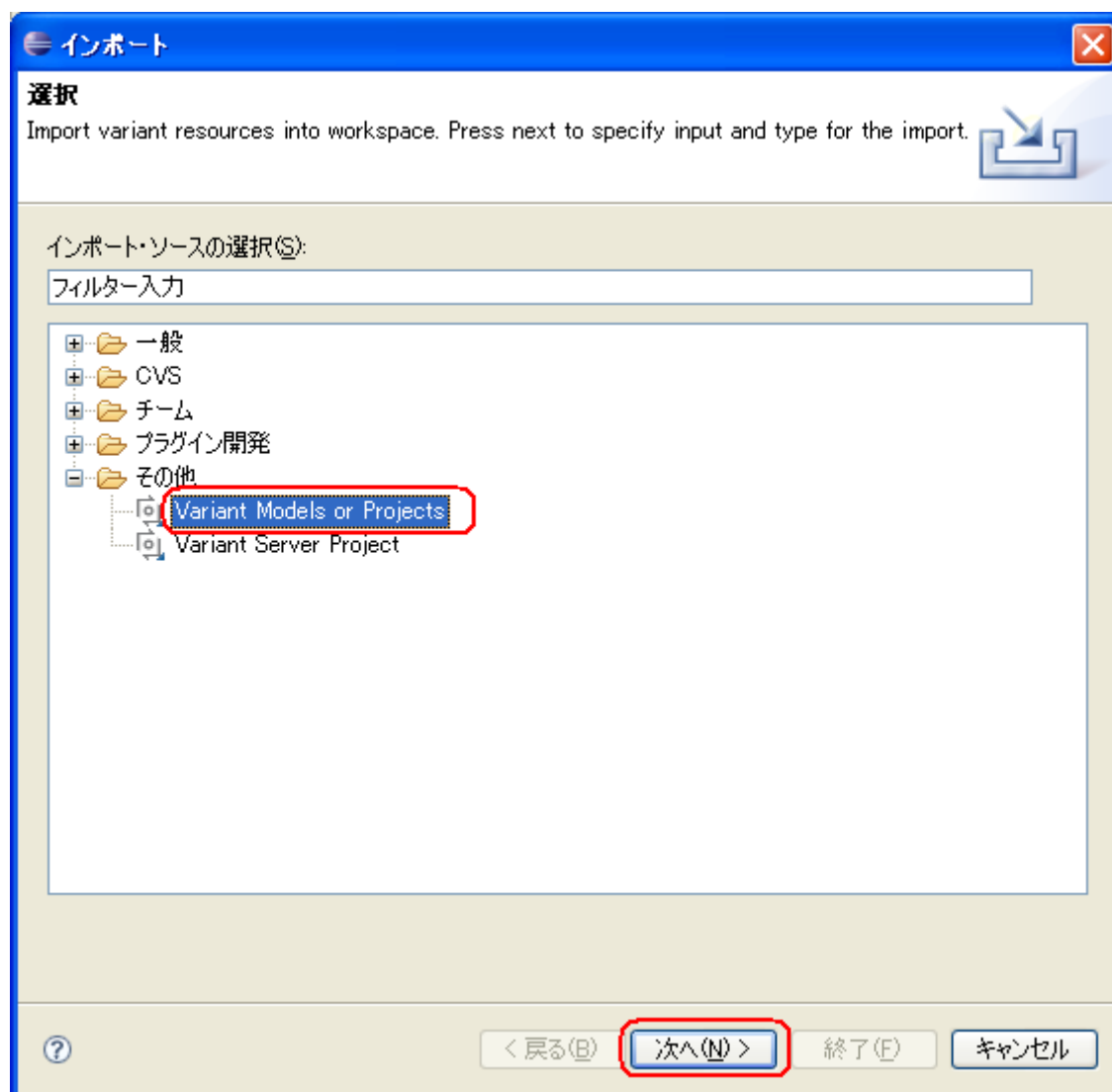
Directory

Project type

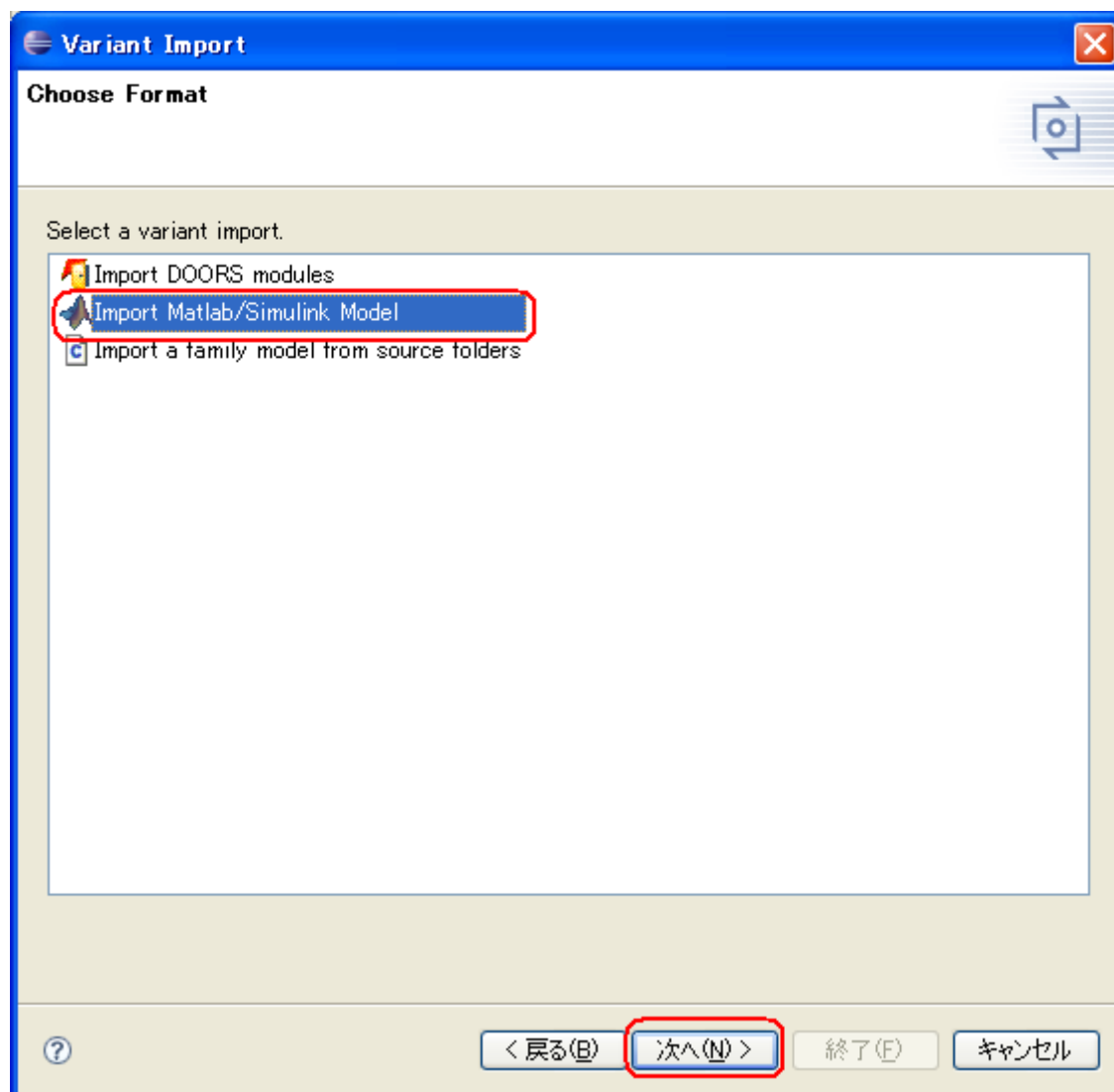
Standard Custom Empty

Description
Creates an empty project without any models.

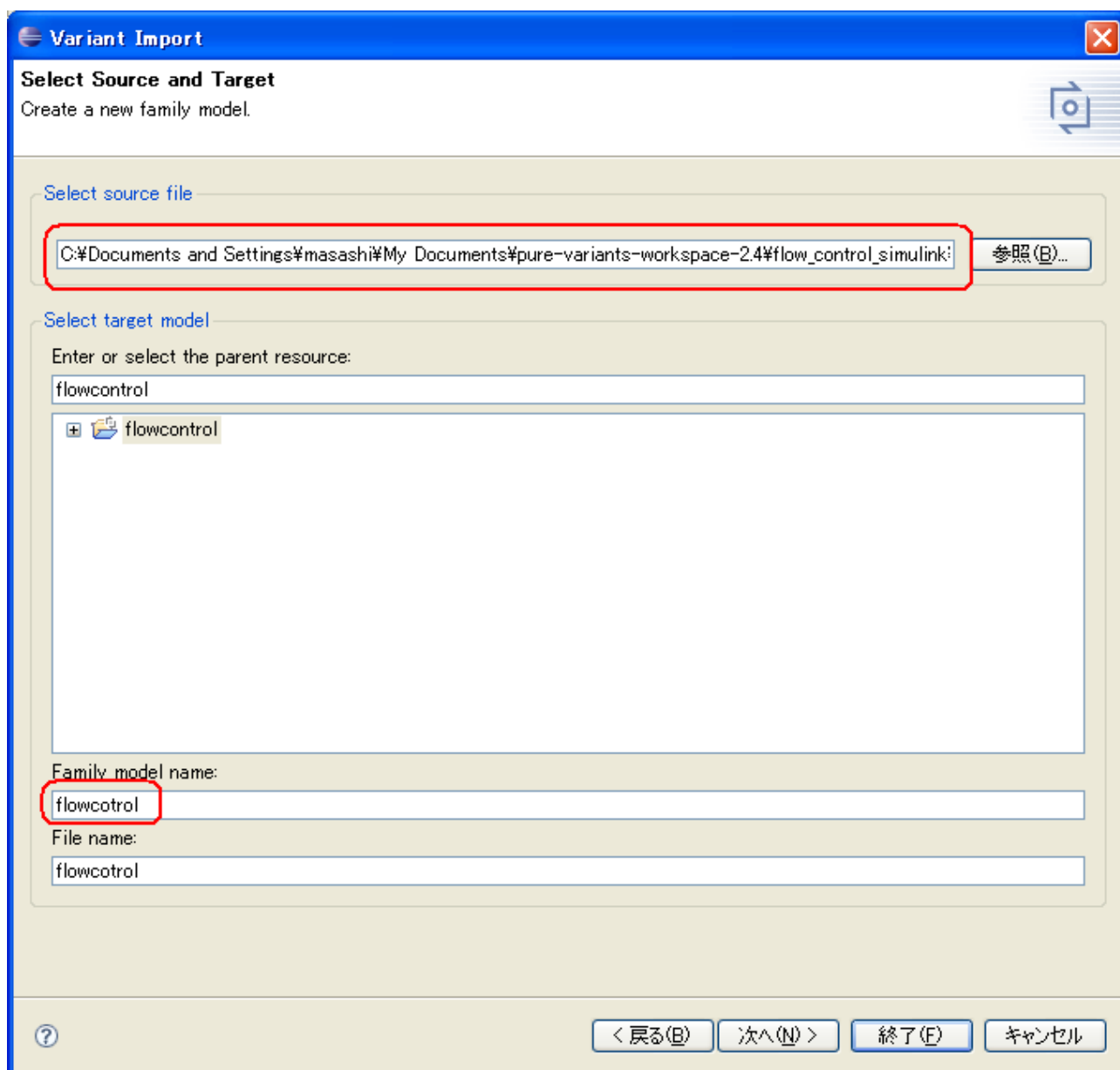
- 4 . pure::variants GUI の左にある **Variant Project** で、先程入力したプロジェクト名の上で右クリックし、インポートを選択します。
- 5 . **その他** から **variant Models or Projects** を選択し、[次へ]ボタンを押下します。



6 . **Import Matlab/Simulink Model** を選択し、[次へ]ボタンを押下します。

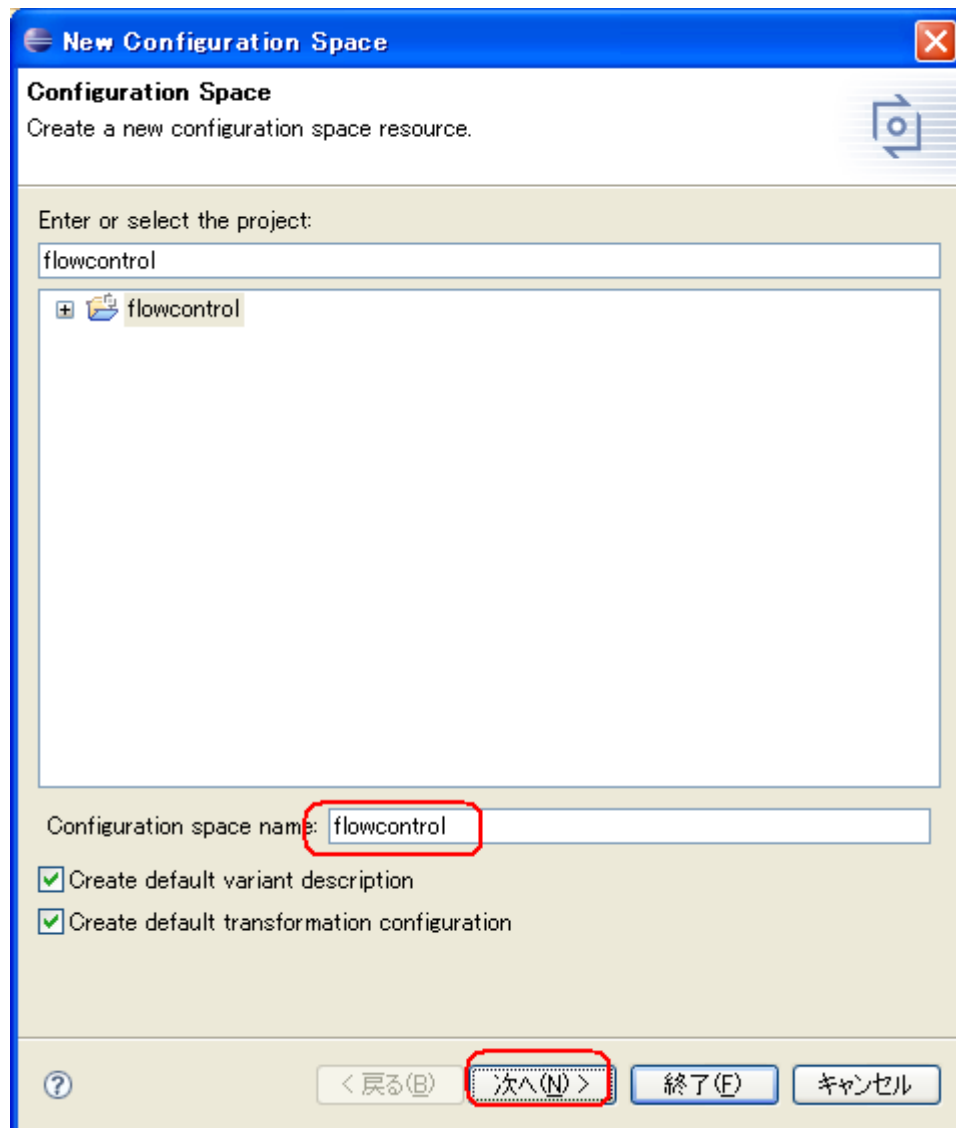


- 7 . Select source file で、**Matlab/Simulink モデル (.mdl)** を選択します。**Family model name** に任意の名前を入力します (File name には同じ名前が入力されます) 。[終了]ボタンを押下します。

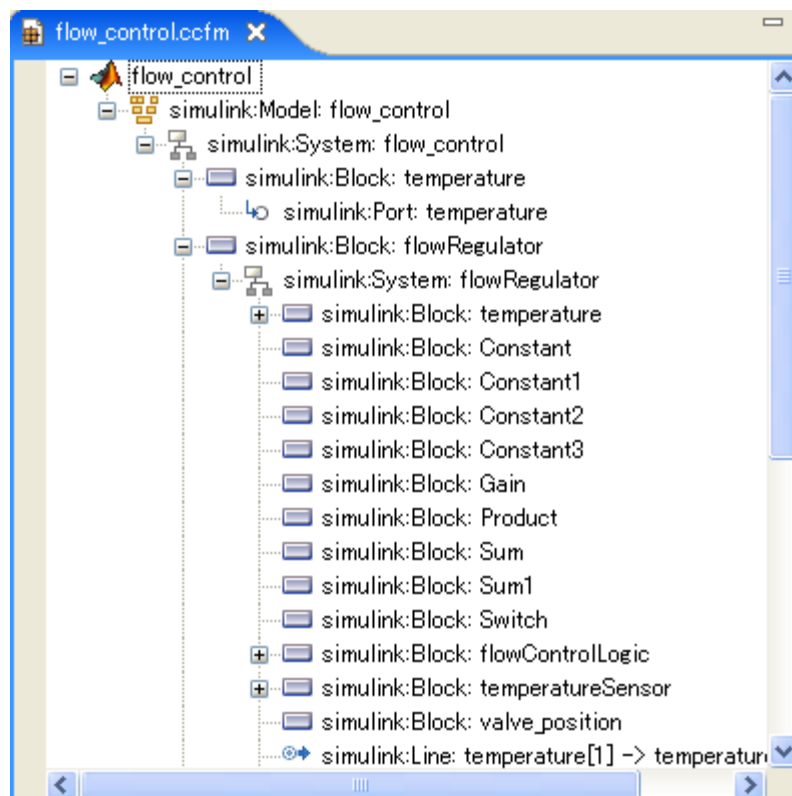


- 8 . 指定した Matlab/Simulink モデルファイルが、**pure::variants** にインポートされます。

- 9 . **Variant Project** で、プロジェクト名の上で右クリックし、[New]から[Configuration Space]を選択します。**Configuration space name** を任意の名前で入力し、[終了]ボタンを押下します。



以下のように、MATLAB/Simulink モデルがファミリーモデルにインポートされました。



ファミリーモデル内は、サブシステムや Simulink ブロック、ブロック同士を結ぶ線として分類されます。上記と同様の手順で、1つのプロジェクトに複数のモデルをインポートすることも可能です。

< バリエーションポイント（変動点）の作成 >

フィーチャモデルにバリエーションポイントを作成し、インポートしたファミリーモデル内のコンポーネントとリンクさせる方法について説明します。ここでは、以下のサンプルモデル（flow_control.mdl）を使って説明します。対象は、flow_control システムの temperatureSensor サブシステムです。

バリエーションポイントの作成には、2種類の方法があります。

1. 別の Simulink モデルをインポートして行う方法（P.11）

プロジェクトに、個々の Simulink モデルをインポートし、そのバリエーションポイントを登録する方法。（インポートの方法は、flow_control.mdl と同様）

2. シンクロナイザー機能を使用する方法（P.39）

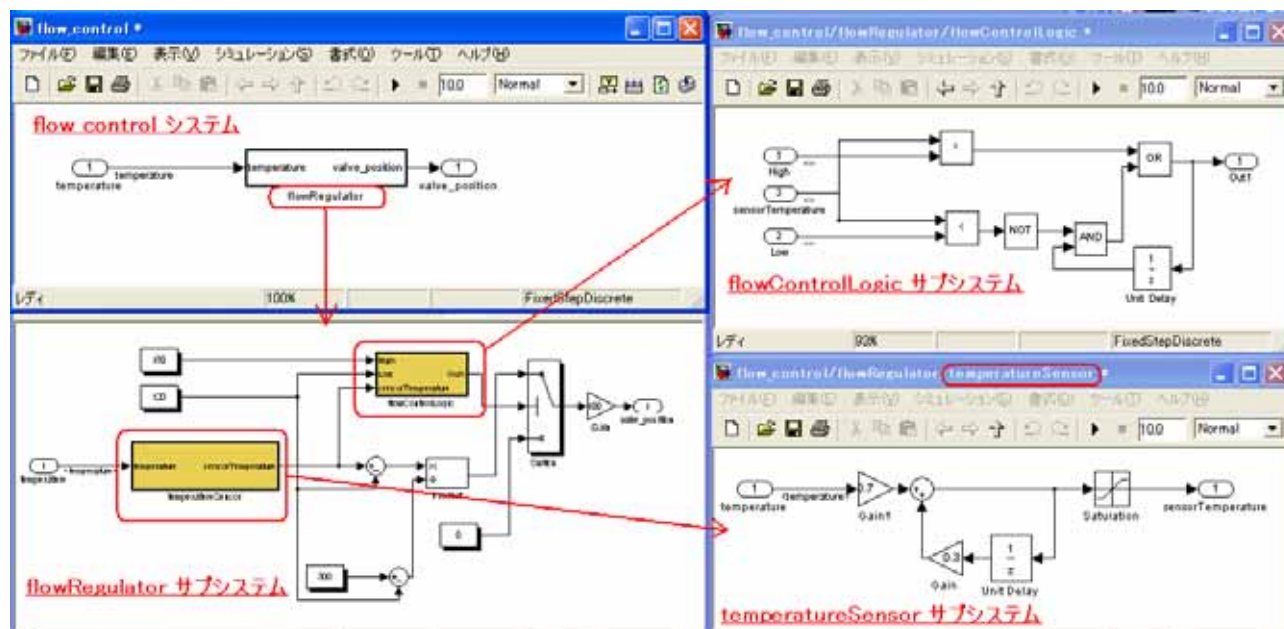
個々の Simulink モデルに対して、ファミリーモデルをシンクロナイズさせて、バリエーションポイントとなる部品（ブロック、信号線）を Simulink モデルからファミリーモデルに追加する方法。

[flow_control.mdl]

flow_control - flowRegulator サブシステム

| - temperatureSensor サブシステム

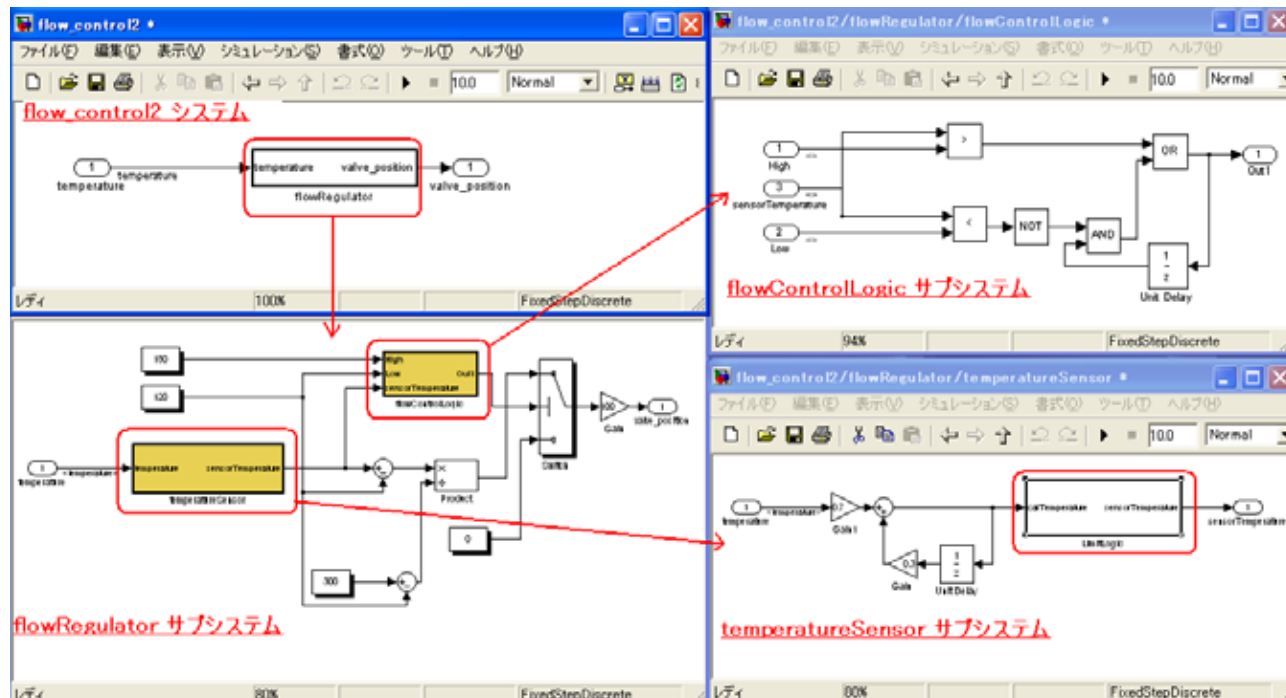
| - flowControlLogic サブシステム



上記右下が temperatureSensor サブシステムです。

[flow_control2.mdl]

flow_control.mdl と同等ですが、バリエーションポイントとして temperatureSensor サブシステム内の Saturation ブロックの代わりに、サブシステム (LimitLogic) を使用しています。



これらのモデルを使用して、以下のような変動性を作成してみたいと思います。

- フィルタ処理 (Gain ブロック、UnitDelay ブロックの処理部分) の有無
- フィルタ処理ありの場合の Gain、Gain1 ブロック値を可変にする
- リミット処理の選択 (Saturation ブロック or LimitLogic サブシステム)
- リミット処理の上下限值を可変にする

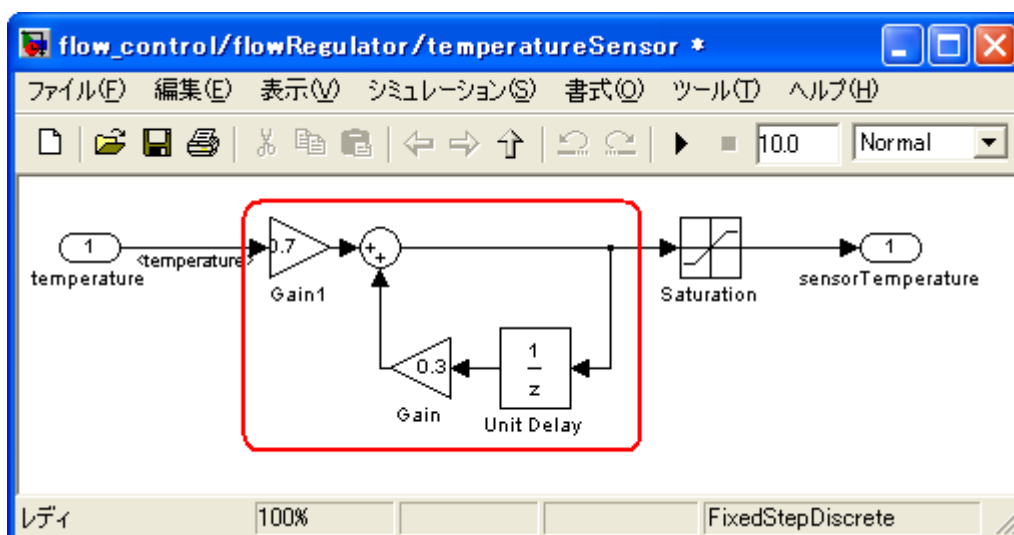
以下、上記 4 つの作成方法について説明します。

1. 別の Simulink モデルをインポートして行う方法

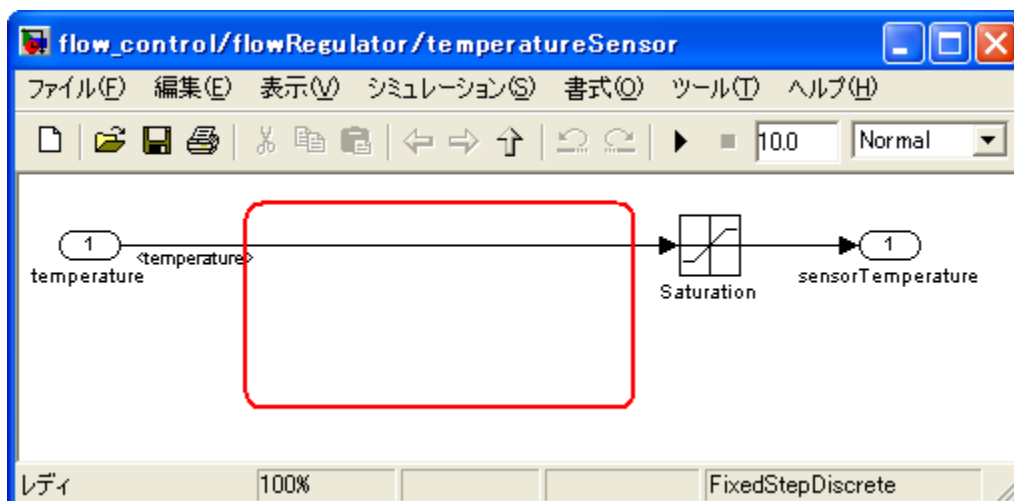
フィルタ処理 (Gain ブロック、UnitDelay ブロックの処理部分) の有無

temperatureSensor サブシステムのフィルタ処理部分(あり or なし)を変動部分として扱います。pure::variants で、以下のような [フィルタ処理あり/なし] のモデルを生成できるようにします。

[フィルタ処理あり]

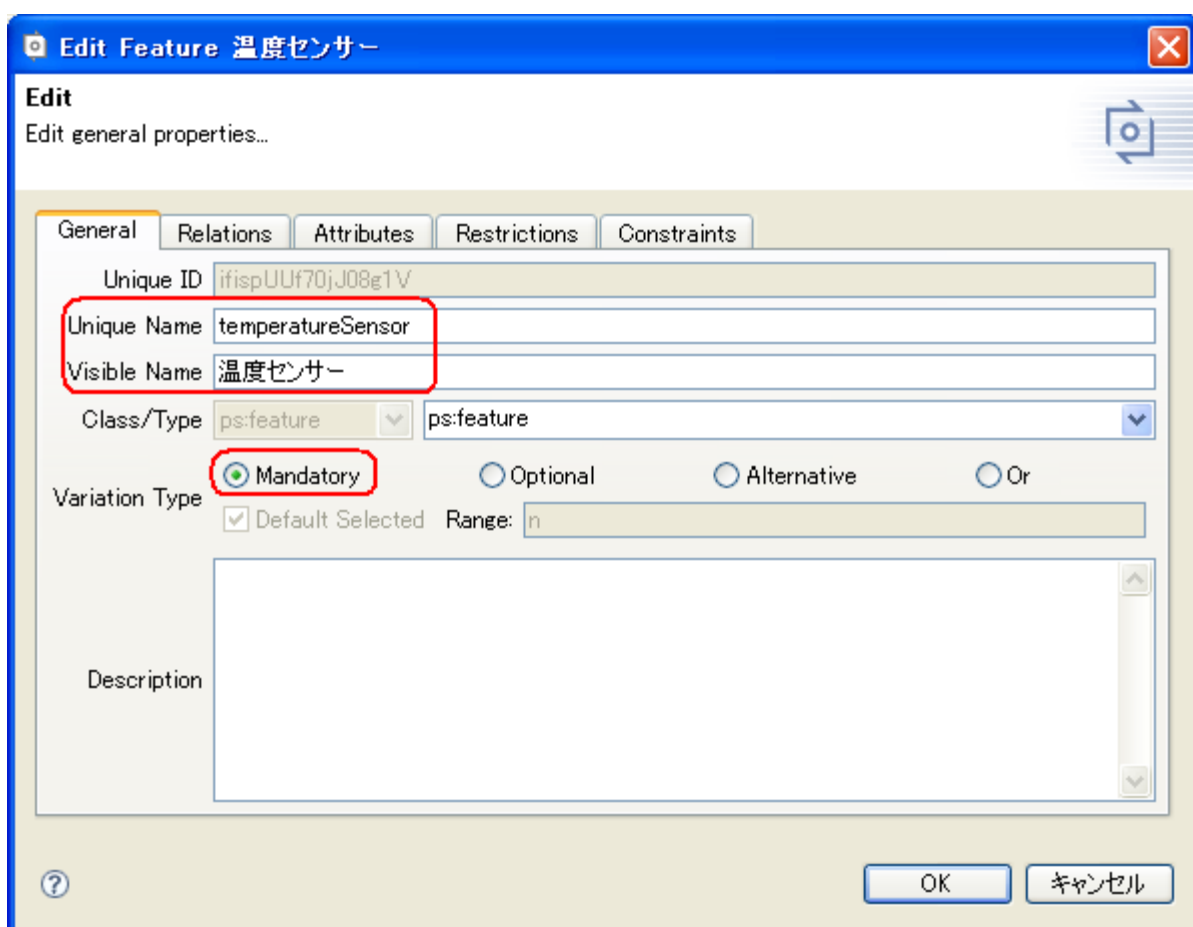


[フィルタ処理なし]



[手順]

1. フィーチャモデル (flow_control.xfm) の flow_control 上で右クリックし、[New]から[Generic Feature]を選択します。New Feature のダイアログが開くので、以下のように [温度センサー] (temperatureSensor) を作成し、[OK]ボタンを押下します。Visible Name は、日本語で記述可能です。

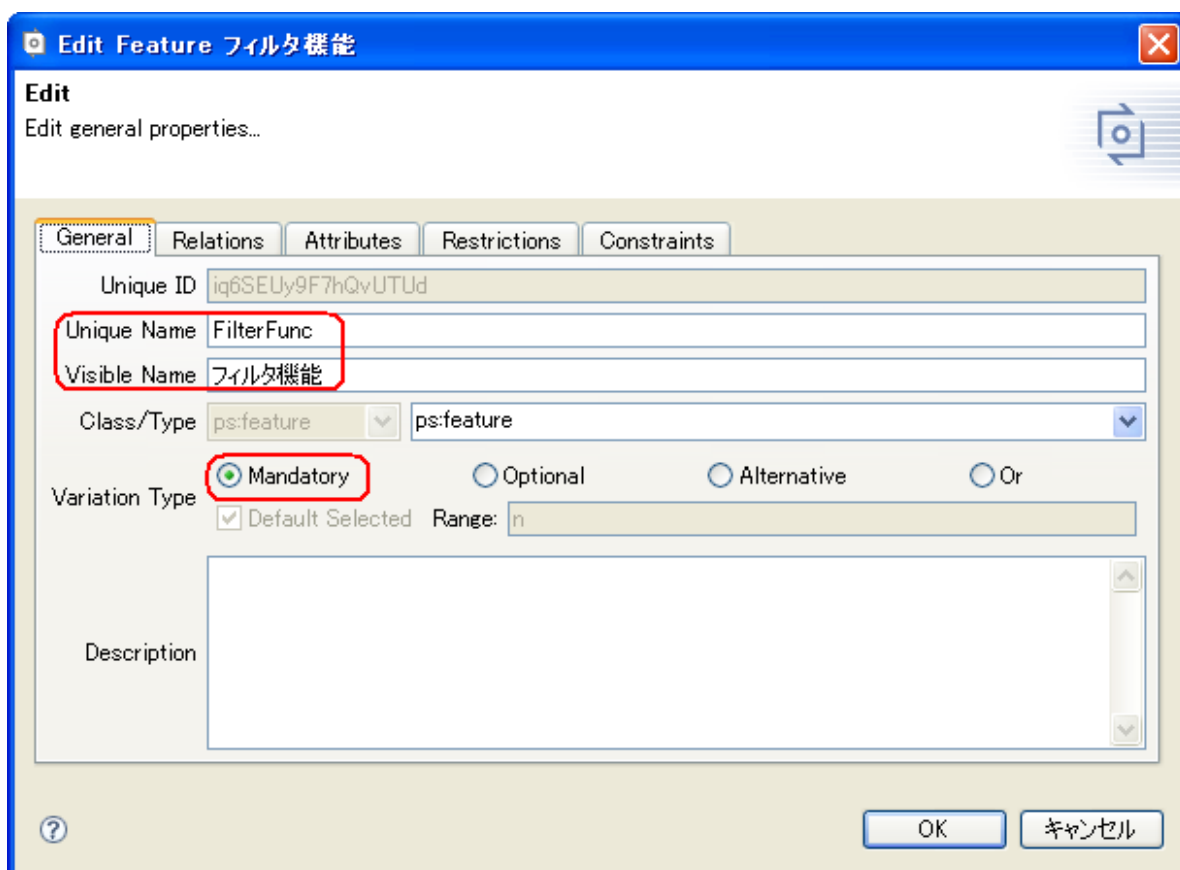


Variation Type では、**Mandatory** を選択します。

[Variation Type]

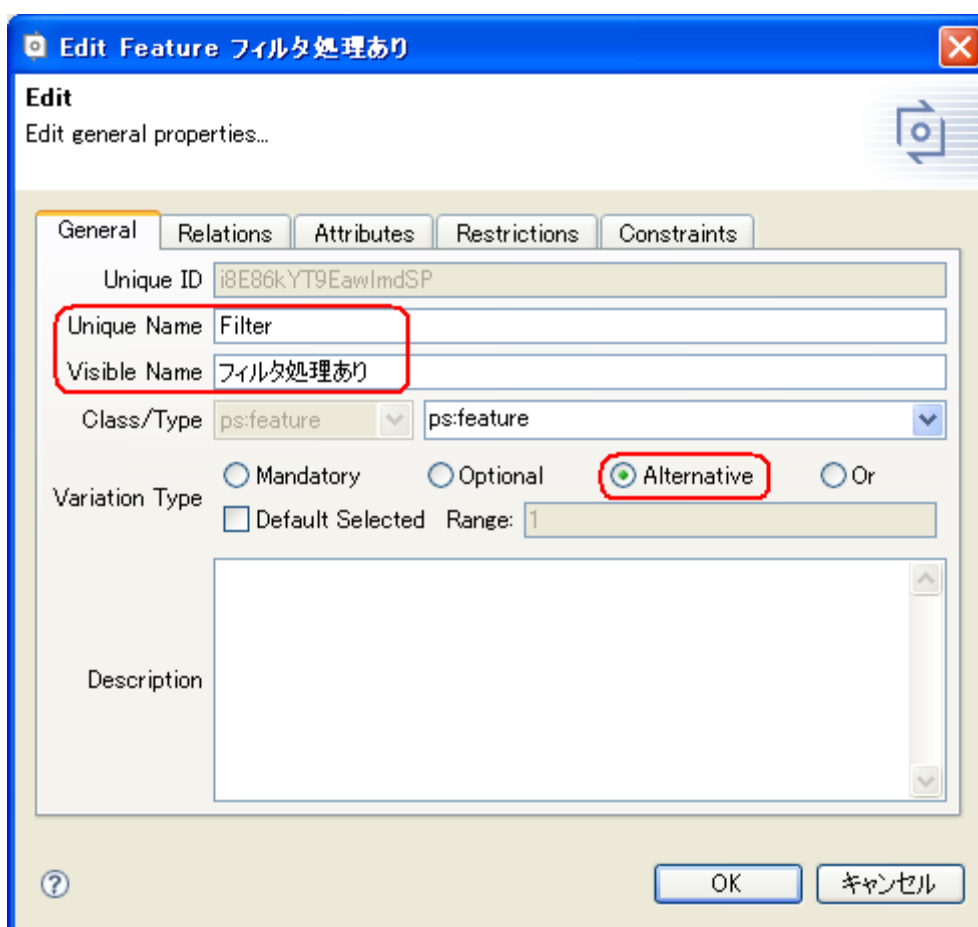
- ・ Mandatory - 必ず選択されるフィーチャとして登録
- ・ Optional - 選択可能なフィーチャとして登録
- ・ Alternative - どれか1つだけ選択されるフィーチャとして登録
- ・ Or - 1つ以上選択されるフィーチャとして登録

- 2 . フィーチャモデルの[温度センサー]上で右クリックし、[New]から[Generic Feature]を選択します。New Feature のダイアログが開くので、以下のように[フィルタ機能] (FilterFunc) を作成し、[OK]ボタンを押下します。



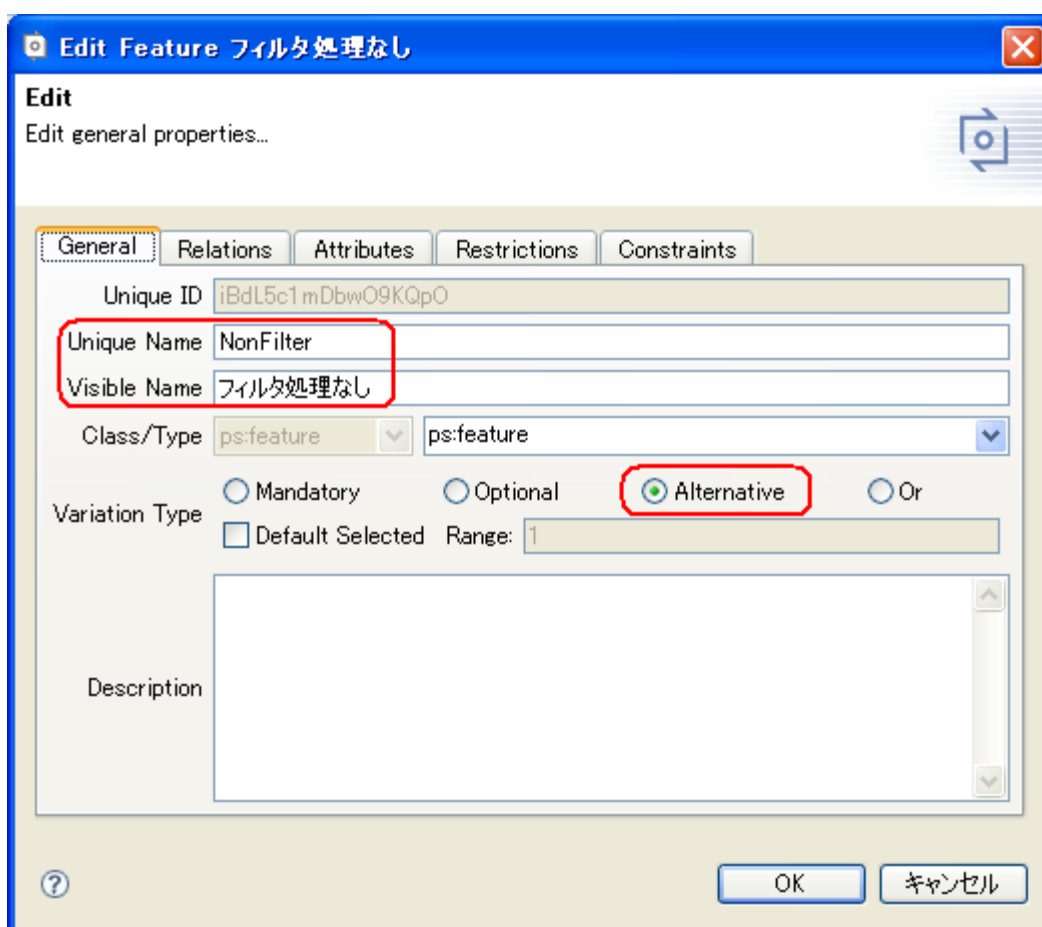
Variation Type では、Mandatory を選択します。

3. フィーチャモデルの[フィルタ機能]上で右クリックし、[New]から[Generic Feature]を選択します。New Feature のダイアログが開くので、以下のように[フィルタ処理あり] (Filter) を作成し、[OK]ボタンを押下します。



Variation Type では、**Alternative** を選択します。

- 4 . 同様に、[フィルタ機能]上で右クリックし、[New]から[Generic Feature]を選択します。New Feature のダイアログが開くので、以下のように[フィルタ処理なし] (NonFilter) を作成し、[OK]ボタンを押下します。

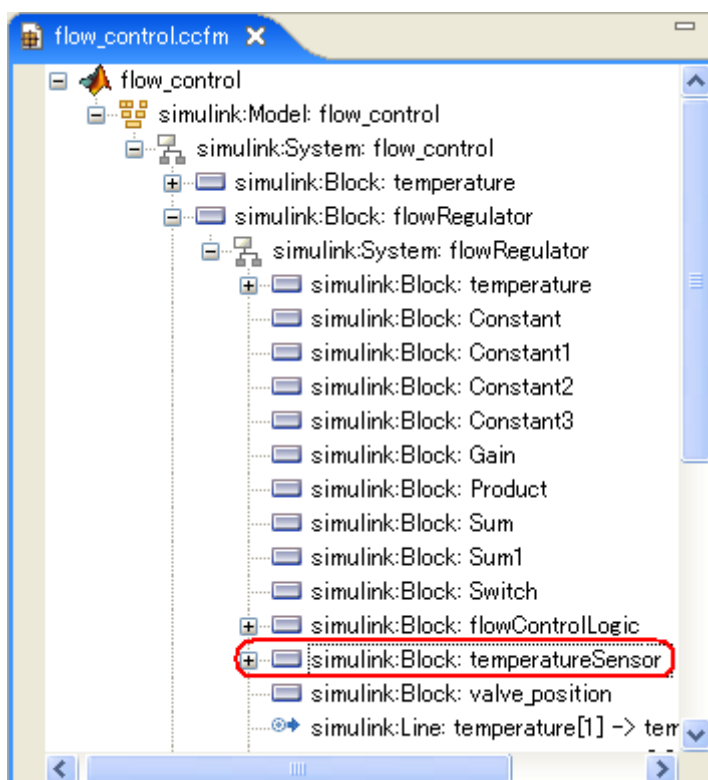


Variation Type では、**Alternative** を選択します。

これで [フィルタ機能] は、[フィルタ処理あり] か [フィルタ処理なし] のどちらか1つを選択することになります。

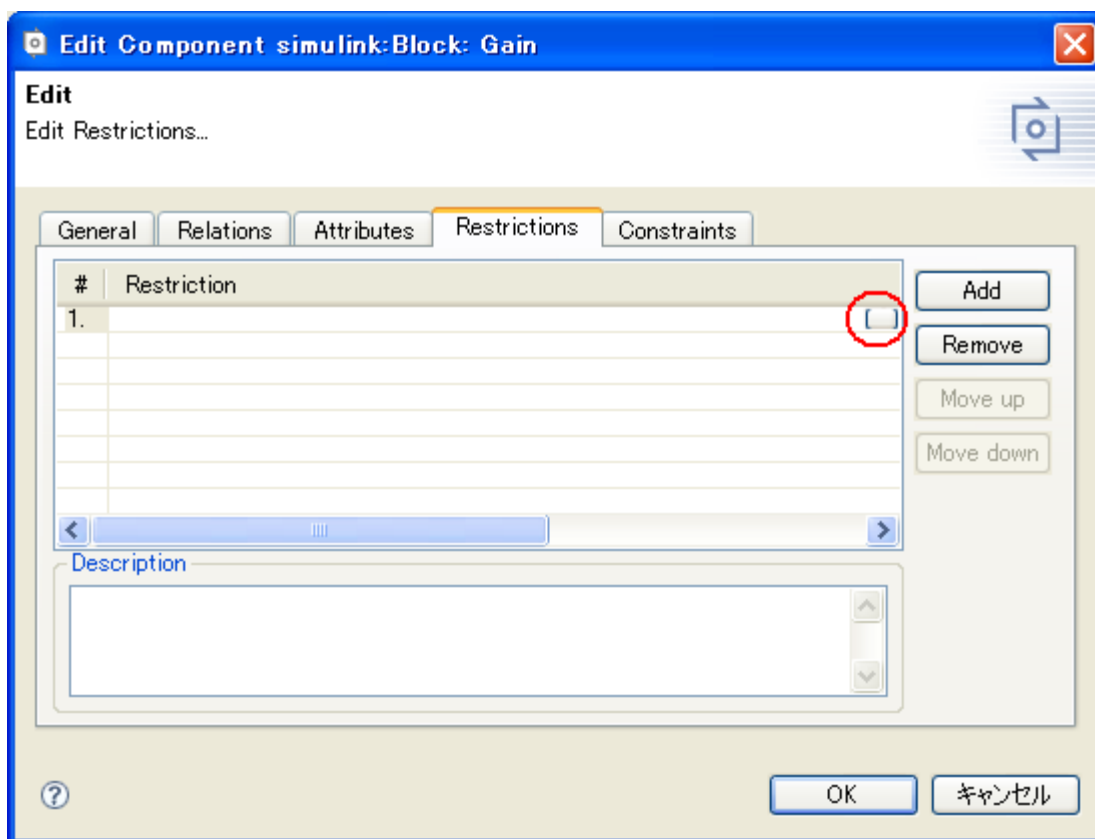


5. 作成したフィーチャモデルとファミリーモデルをリンクします。ファミリーモデルから temperatureSensor の部分を開きます (temperatureSensor の上位層である Simulink:Block: flowRegulator から、simulink:Block: temperatureSensor を開きます)。そこに、インポートされた temperatureSensor サブシステムのブロックや信号線などの部品が置かれています。

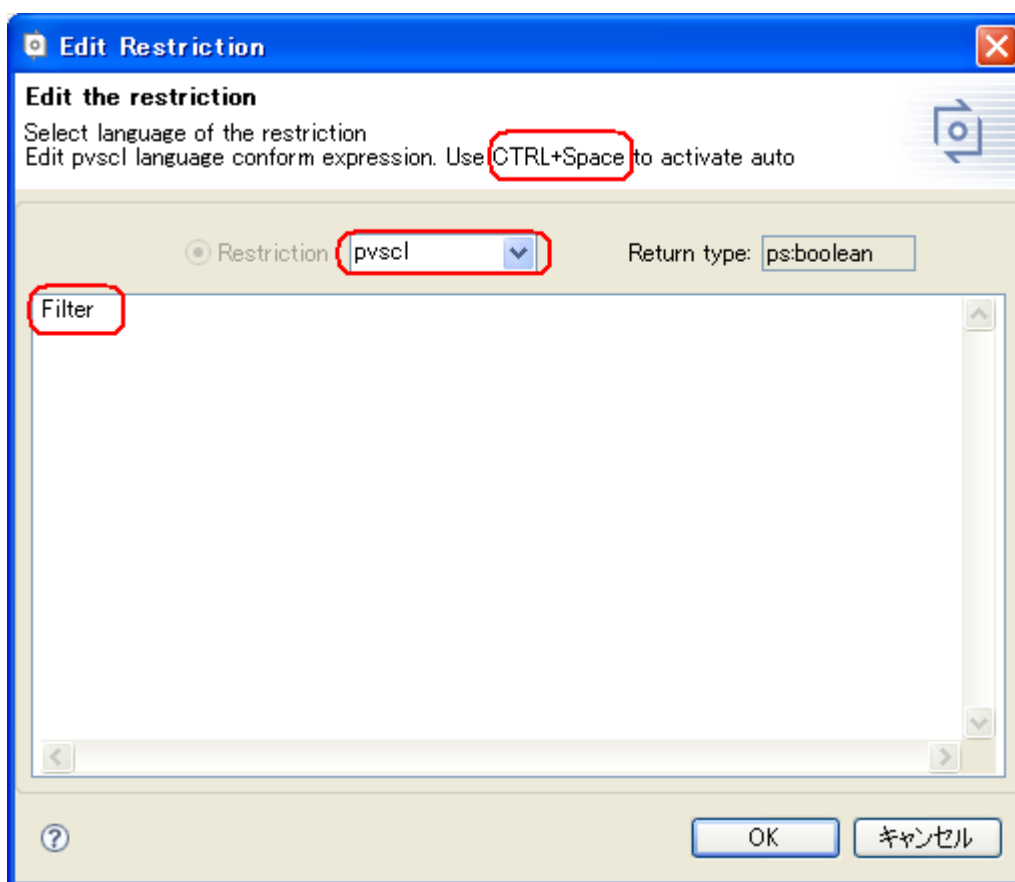


まずはファミリーモデルに対して、[フィルタ処理あり] (Filter) の設定を行います。

Simulink:Block: Gain をダブルクリックします。Edit 画面が表示されるので、**Restrictions タグ**を選択し、[Add]ボタンを押下します。Restriction フィールドをクリックすると、右端にボタンが現われるのでそれをクリックします。



Restriction が pvscl になっていることを確認し、白紙の部分をクリックして CTRL+Space を押下します。いくつか項目が現われるので、その中から [Filter - フィルタ処理あり] をダブルクリックします。すると白紙の部分に **Filter** と入力されるので、[OK] ボタンを押下します。これで、この **Gain** ブロックは、フィルタ処理ありの時だけ使用されるようにリンクされました。

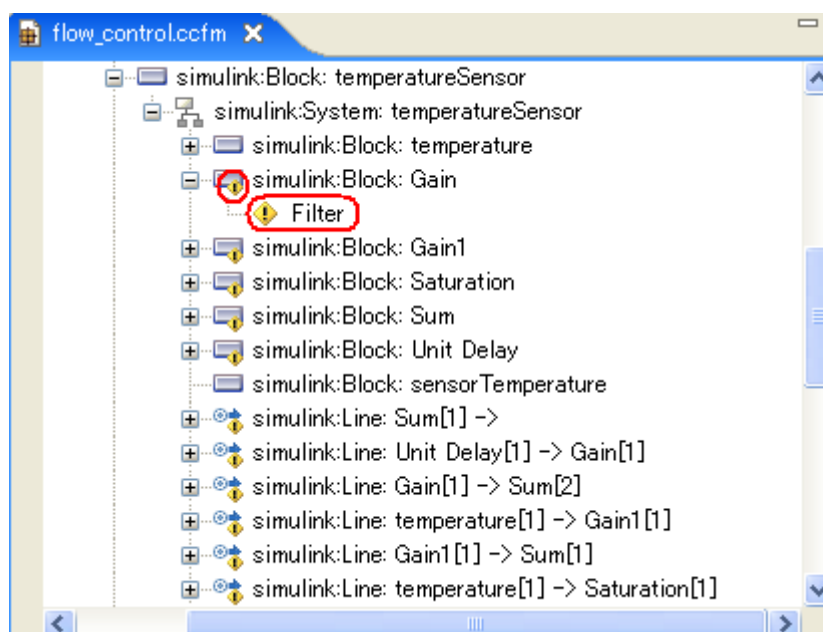


同様に、以下のブロックや信号線に対しても設定を行います。

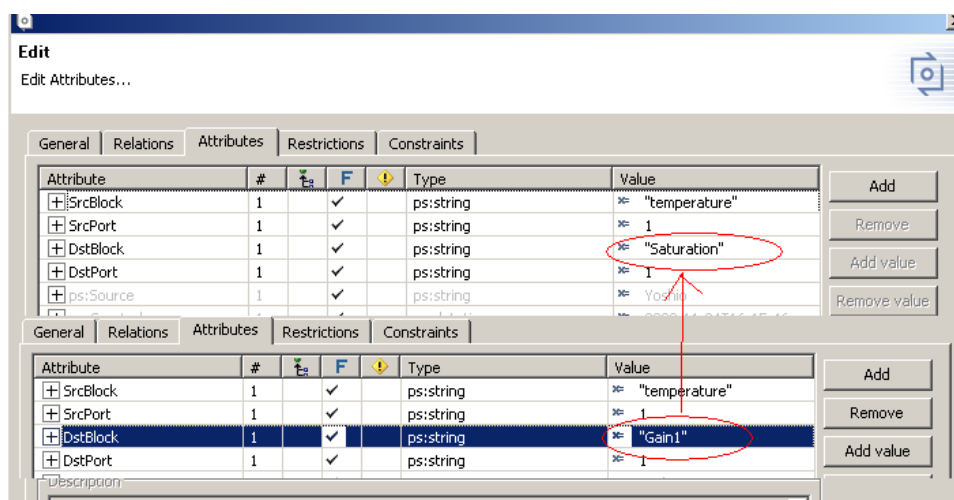
simulink:Block: Gain1、
 simulink:Block: Sum、
 simulink:Block: Unit Delay、
 simulink:Line: Sum[1] ->、
 simulink:Line: Unit Delay[1] -> Gain[1]、
 simulink:Line: Gain[1] -> Sum[2]、
 simulink:Line: temperature[1] -> Gain1[1]、
 simulink:Line: Saturation[1] -> sensorTemperature[1]、
 simulink:Line: Gain1[1] -> Sum[1]

1つずつ Restriction を設定していますが、まとめて設定することも可能です。その場合、ファミリーモデルの要素を複数選択し、右クリック->New->Restriction をクリックします。後は、上記 Edit Restriction で設定します。

ファミリーモデル上で右クリックし、[Show InTree]から[Restrictions]を選択すると、先程設定した **Filter** が参照できるようになります。各ブロックや信号線に何が設定されているかが確認できます。

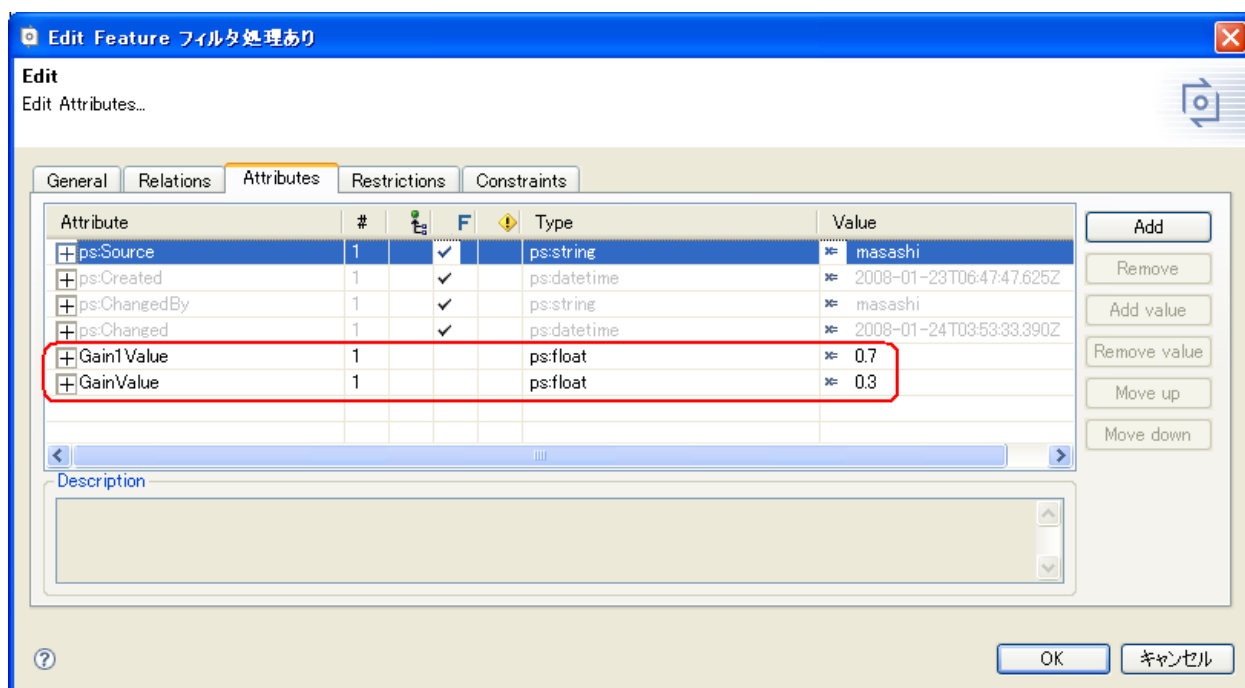


6. 次に、[フィルタ処理なし] (NonFilterer) の設定を行います。インプットの **temerature** ブロックから **Saturation** ブロックへ信号線が新たに必要になるので、**simulink:Line: temperature[1] -> Gain1[1]** 上で右クリックして **Copy** を選択し、**simulink:System: temperatureSensor** 上で **Paste** します。すると、同じ信号線が **simulink:System: temperatureSensor** の一番下に追加されます。このコピーで新しく作成したものをダブルクリックし、**General** タグの **Visible Name** を **temperature[1] -> Saturation[1]** に変更します。さらに、**Restrictions** タグをクリックし、手順4と同様に、Restriction に[**NonFilterer - フィルタ処理なし**]をダブルクリックで設定します。すると白紙の部分に **NonFilter** と入力されるので、[OK]ボタンを押下します。これで、フィルタ処理なしの時だけ使用されるようにリンクされました。更に以下のように、Attributes の **Gain1** の名前を **Saturation** に変更します。



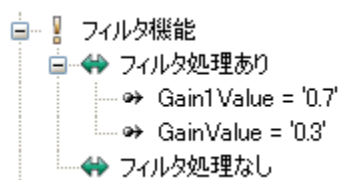
フィルタ処理ありの場合の Gain、Gain1 ブロック値を可変にする

temperatureSensor サブシステムの[フィルタ処理あり]に属性を設定します。[フィルタ処理あり]をダブルクリックし、プロパティを開きます。**Attributes タグ**をクリックし、[Add]ボタンを押下し Attribute を追加します。以下のように、Attribute 欄に **GainValue** と入力、Type 欄から **ps:float** を選択し、Value 欄にデフォルト値(0.3)を入れておきます。F 欄のチェックは外しておきます(フィーチャモデルやバリエーションモデル上で値を変更できるようにする為)。同様に、[Add]ボタンを押下して、デフォルト値(0.7)の **Gain1Value** を作成します。



[OK]ボタンを押下します。

フィーチャモデル上で右クリックし、[Show InTree]から[Show All]を選択すると、先程設定した **Attribute** が参照できます。

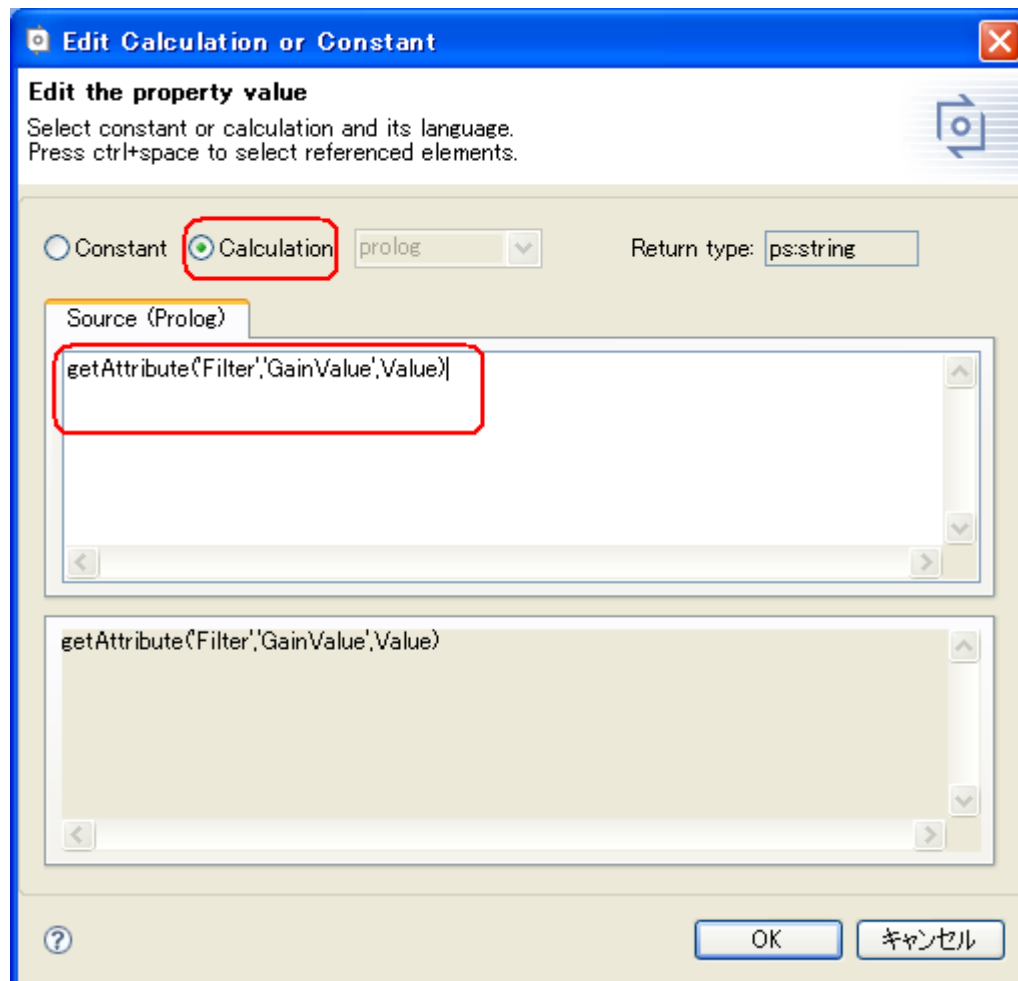


次に、これらの変数がファミリーモデル側に反映されるように設定します。

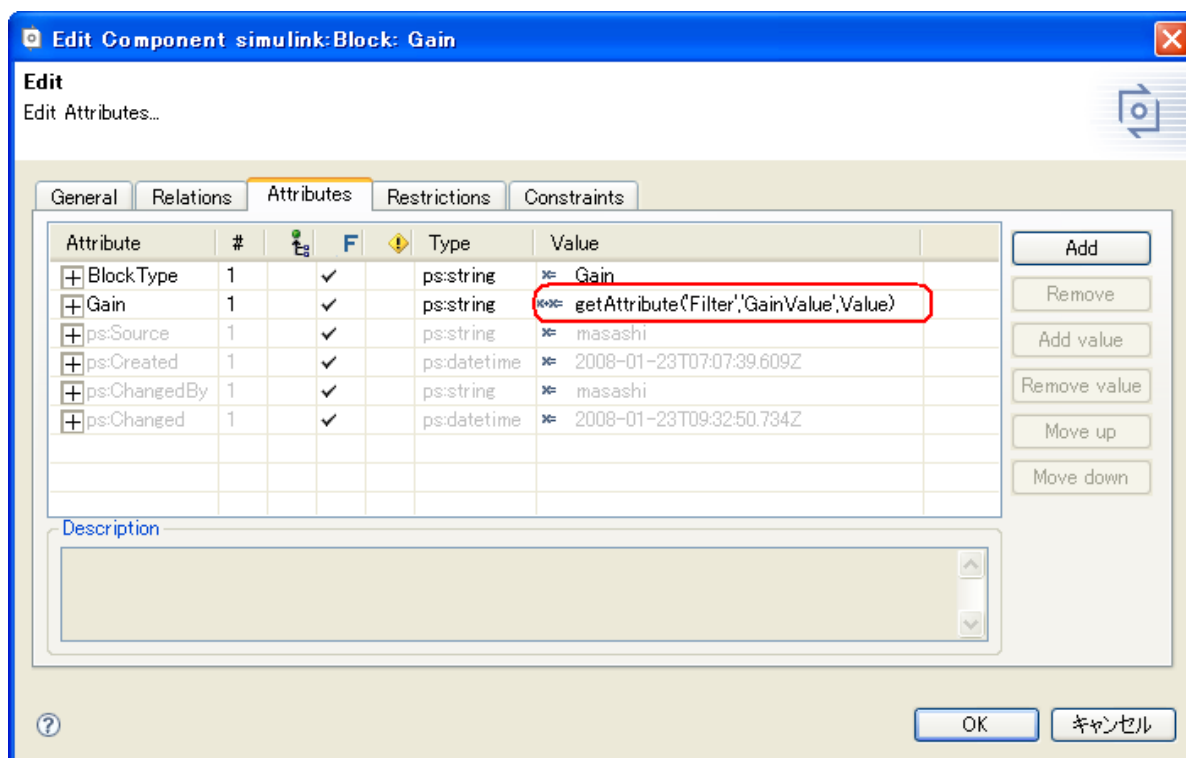
ファミリーモデルの simulink:Block: Gain をダブルクリックし、プロパティを開きます。**Attributes タグ**をクリックし、Attribute である **Gain** の Value 欄を以下のように変更します。

```
Gain : getAttribute('Filter','GainValue',Value)
```

Value 欄をクリックして、右端に出るボタンを押下して Edit 画面を表示します。以下のように設定します。



これは、フィーチャモデルの Filter(フィルタ処理あり)の Attribute である GainValue の Value 欄の値を取得するという式になります。



[OK]ボタンを押下します。

同様に、Gain1 ブロックのプロパティから Attribute である **Gain** の **Value** 欄を以下のように変更します。

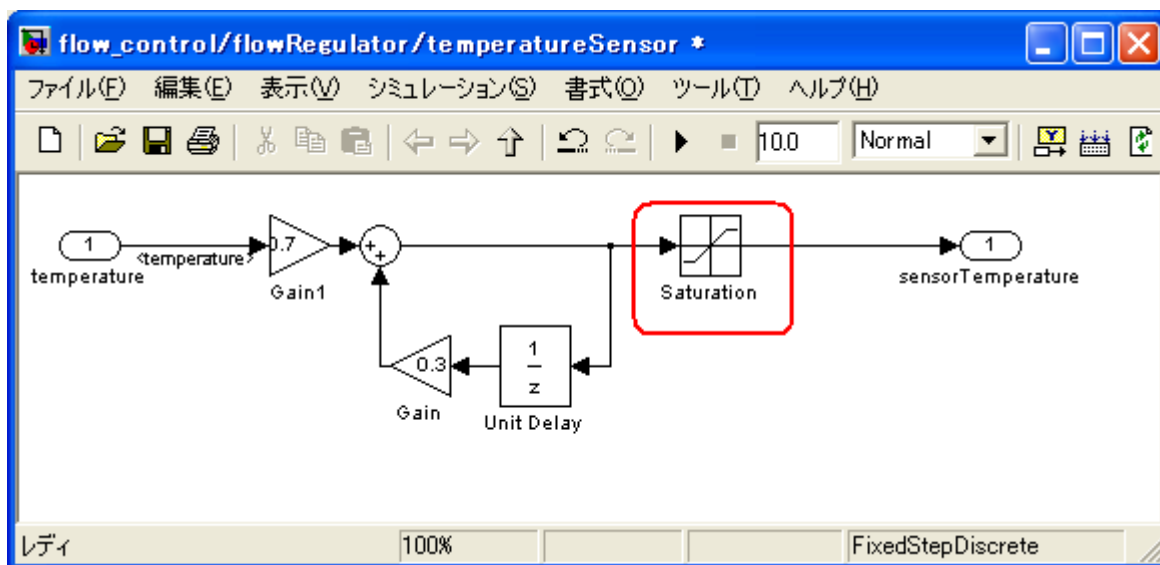
Gain : `getAttribute('Filter', 'Gain1Value', Value)`

これで、フィーチャモデルで設定した Gain、Gain1 の値が、ファミリーモデル (Simulink モデル) に反映されるようになります。

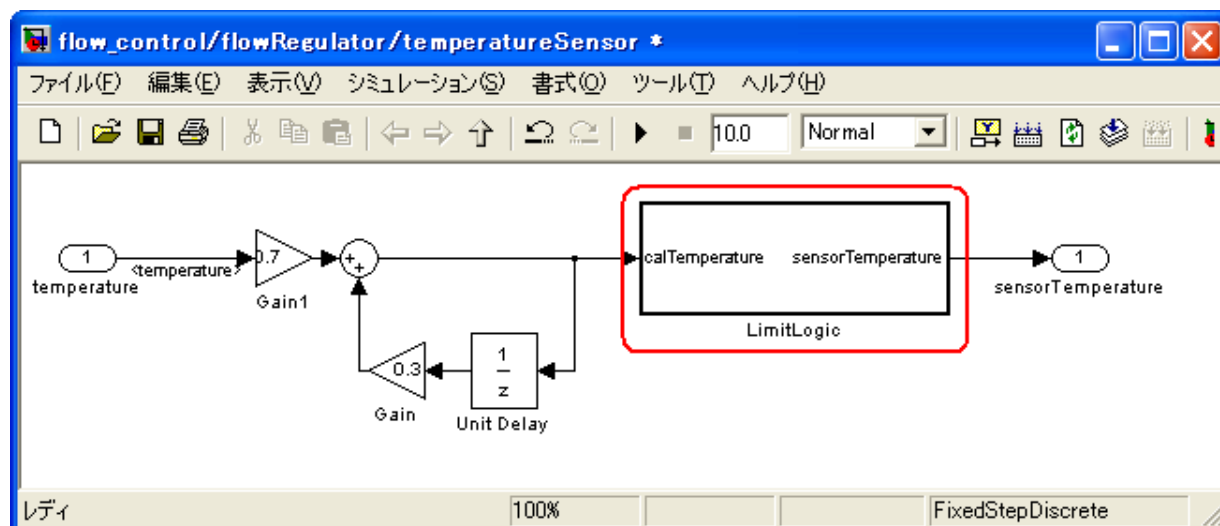
リミット処理の選択 (Saturation ブロック or LimitLogic サブシステム)

temperatureSensor サブシステムのリミット処理部分 (Saturation or サブシステム) を変動部分として扱います。pure::variants で、以下のような [リミット処理] のモデルを生成できるようにします。

[リミット処理 : Saturation]

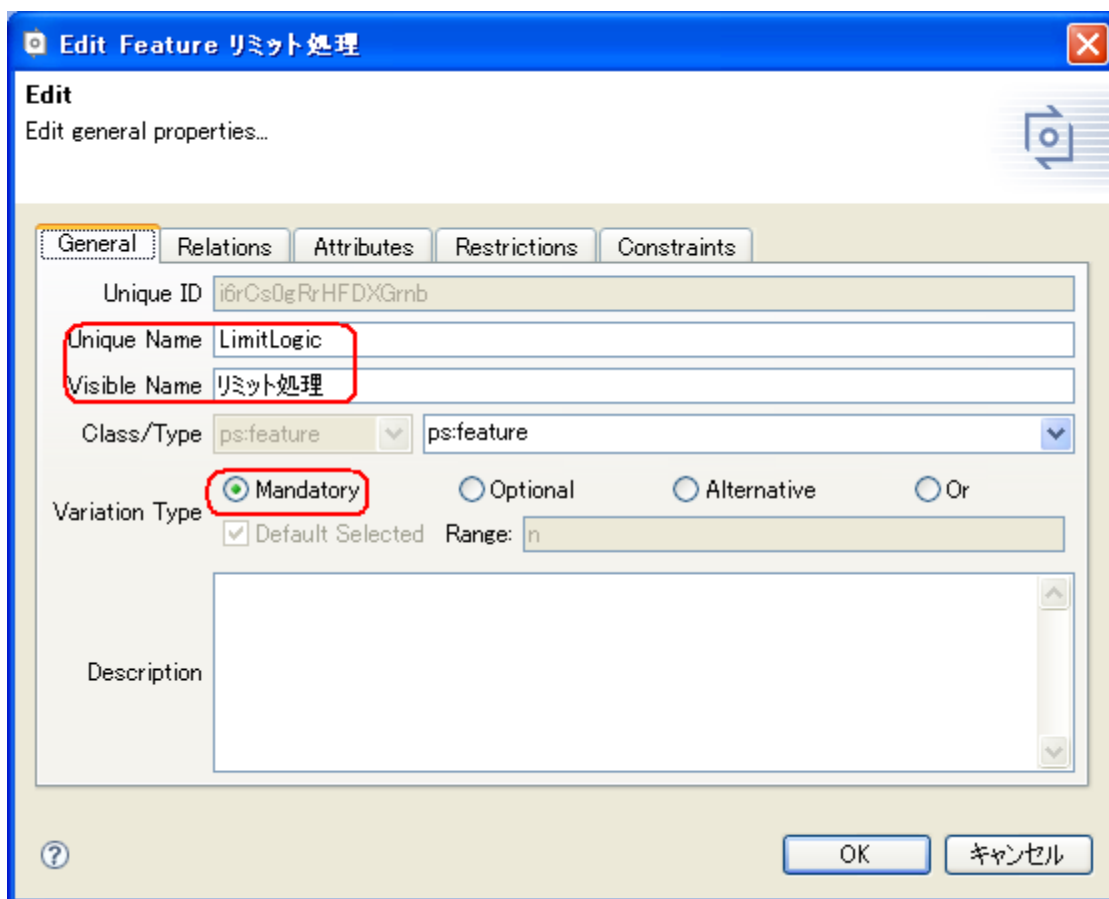


[リミット処理：サブシステム]



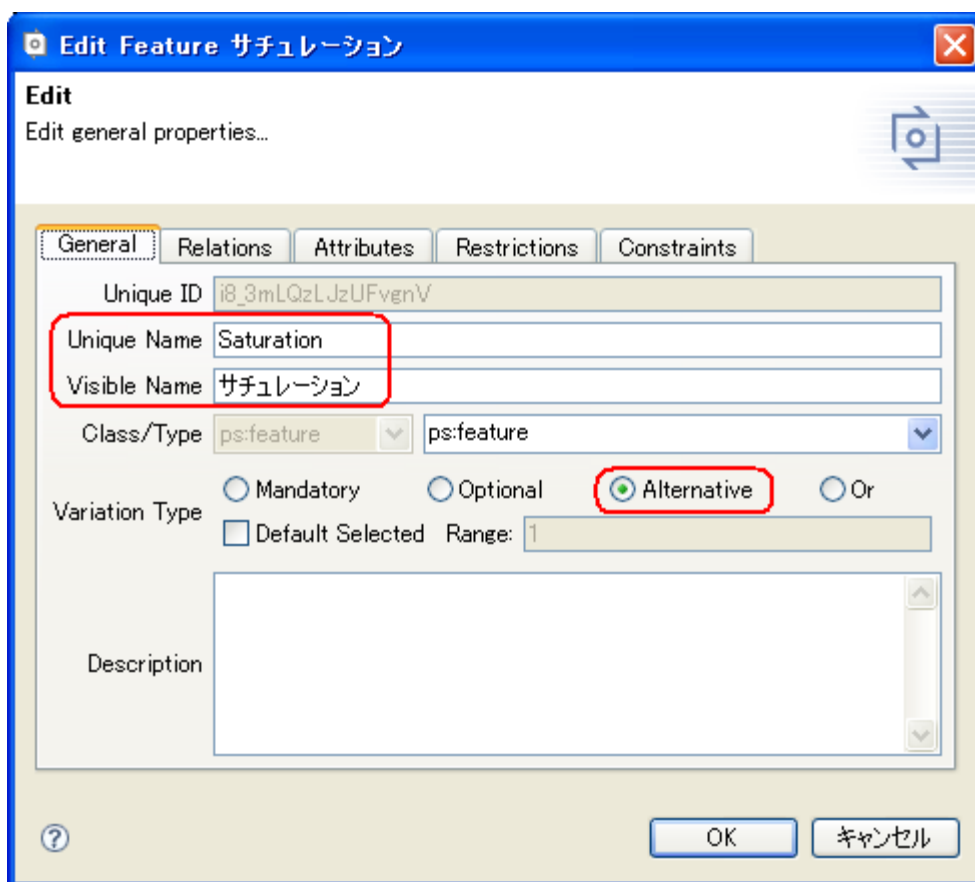
[手順]

2. フィーチャモデル(flow_control.xfm)の[温度センサー]上で右クリックし、[New]から[Generic Feature]を選択します。New Feature のダイアログが開くので、以下のように [リミット処理] (LimitLogic) を作成し、[OK]ボタンを押下します。



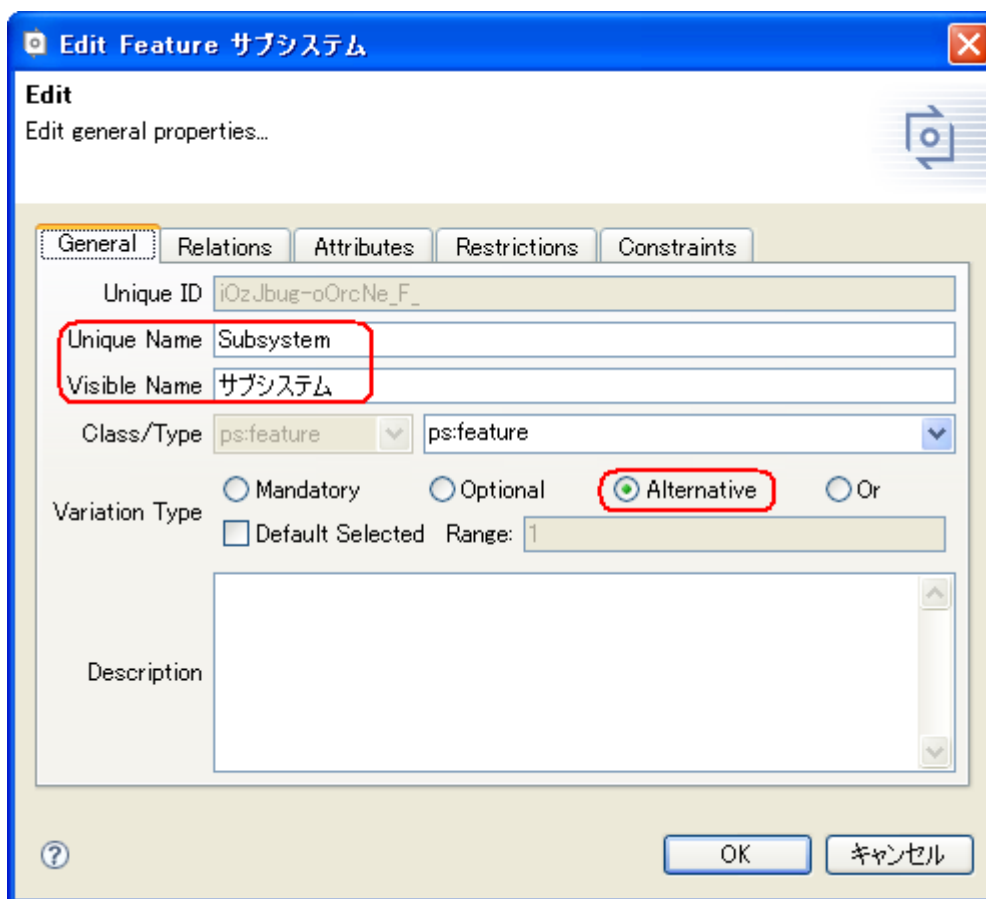
Variation Type では、**Mandatory** を選択します。

3. フィーチャモデルの[リミット処理]上で右クリックし、[New]から[Generic Feature]を選択します。**New Feature** のダイアログが開くので、以下のように[サチュレーション](Saturation)を作成し、[OK]ボタンを押下します。



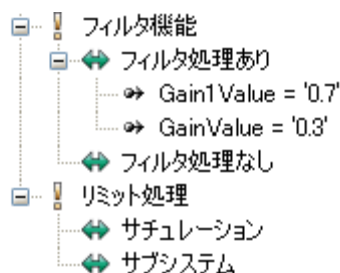
Variation Type では、Alternative を選択します。

- 同様に、フィーチャモデルの[リミット処理]上で右クリックし、[New]から[Generic Feature]を選択します。New Feature のダイアログが開くので、以下のように[サブシステム] (Subsystem) を作成し、[OK]ボタンを押下します。



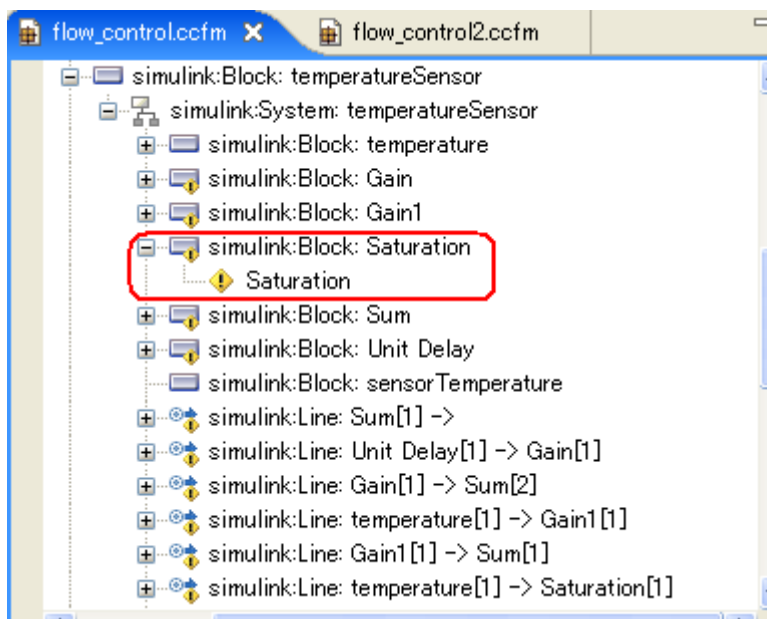
Variation Type では、Alternative を選択します。

これで [リミット処理] は、[サチュレーション] か [サブシステム] のどちらか1つを選択することになります。



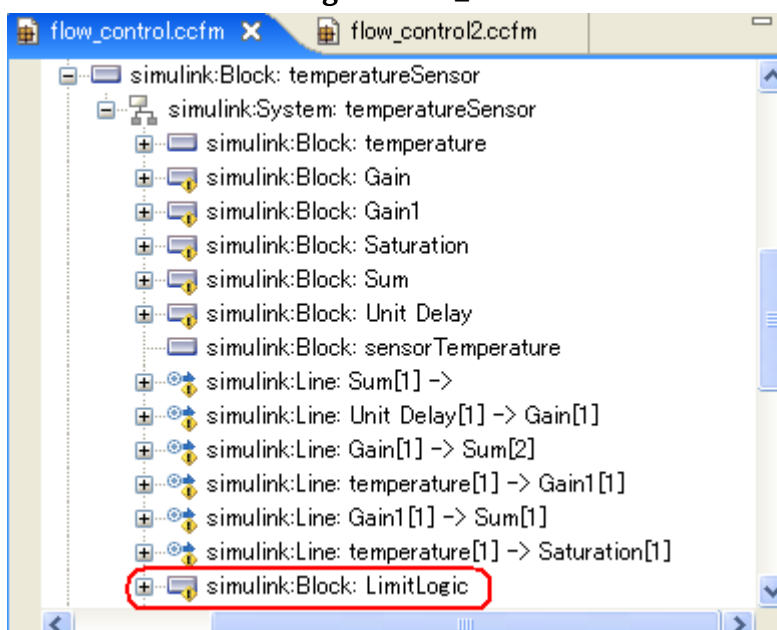
- 作成したフィーチャモデルとファミリーモデルをリンクします。flow_control.mdl のファミリーモデル (flow_control.cfm) から、 temperatureSensor の部分を開きます (temperatureSensor の上位層である Simulink:Block: flowRegulator から、 simulink:Block: temperatureSensor を開きます)。その中の simulink:Block:Saturation をダブルクリックします。プロパティで Restrictions タグをクリックし、[Add]ボタンを押下します。Restriction フィールドをクリックすると、右端にボタンが現われるのでそれをクリックします。Restriction が pvscl になっていることを確認し、白紙の部分をクリックして CTRL+Space を押下します。いくつか項目が現われるので、その中から [Saturation - サチュレーション] をダ

ブルクリックします。すると白紙の部分に **Saturation** と入力されるので、[OK]ボタンを押下します。これで、この **Saturation** ブロックは、[サチュレーション]の時だけ使用されるようにリンクされました。



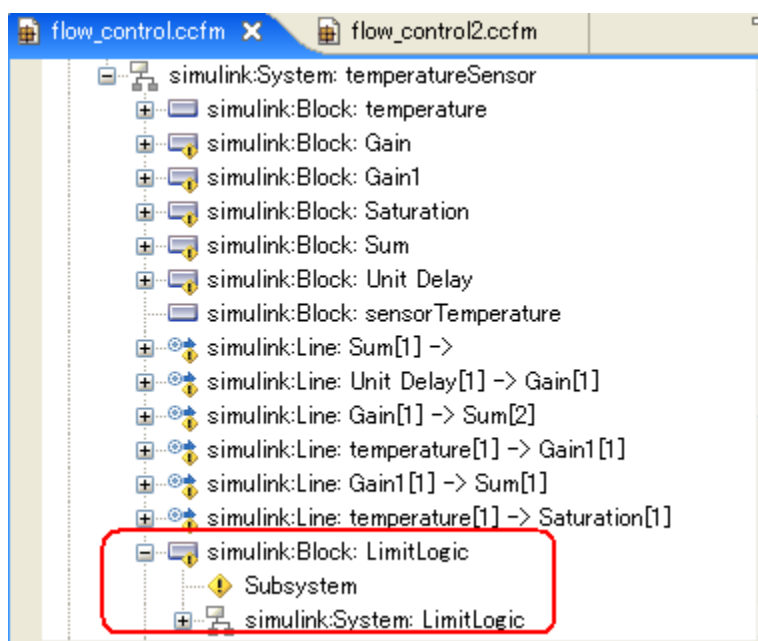
- 6 . flow_control2.mdl のファミリーモデル (flow_control2.ccfm) から、temperatureSensor の部分を開きます (temperatureSensor の上位層である Simulink:Block: flowRegulator から、simulink:Block: temperatureSensor を開きます)、その中の simulink:Block: LimitLogic を右クリックして Copy を選択します。

flow_control.mdl のファミリーモデル (flow_control.ccfm) に戻り、simulink:System:temperatureSensor を右クリックし、Paste を選択します。これで、simulink:Block: LimitLogic が flow_control.ccfm にコピーされました。



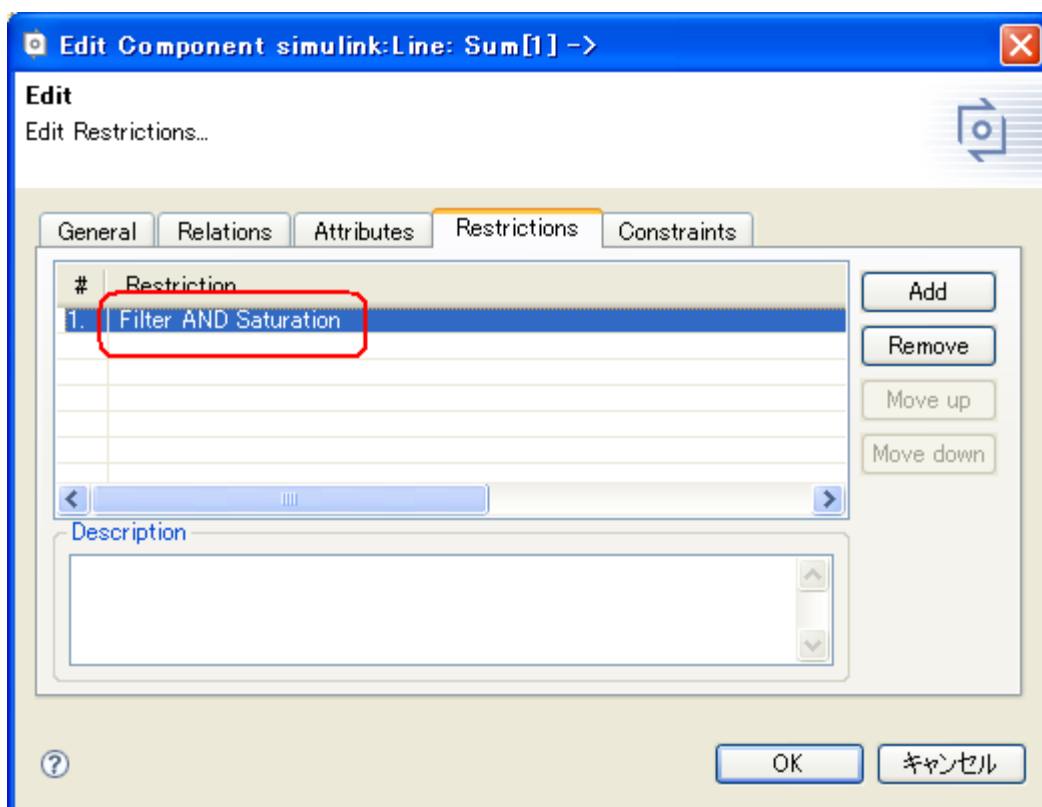
simulink:Block: LimitLogic をダブルクリックしてプロパティを開きます。**Restrictions** タグをクリックし、[Add]ボタンを押下します。Restriction フィールドをクリックすると、右端にボタンが現われるのでそれをクリックします。

Restriction が pvscl になっていることを確認し、白紙の部分をクリックして CTRL+Space を押下します。いくつか項目が現われるので、その中から [**Subsystem - サブシステム**]をダブルクリックします。すると白紙の部分に **Subsystem** と入力されるので、[OK]ボタンを押下します。これで、この **LimitLogic** サブシステムは、[**サブシステム**]の時だけ使用されるようにリンクされました。

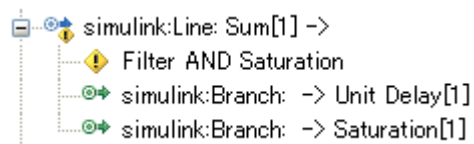


7. フィルタ機能とリミット処理の組合せに対応できるように、信号線を準備しなければなりません。

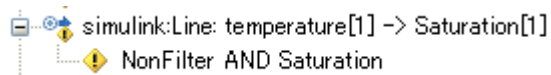
simulink:Line:Sum[1]-> は、UnitDelay と Saturation に信号線が出ているので、**simulink:Line:Sum[1]->** のプロパティの **Restrictions** タグで設定した **Filter** に対して **Saturation** の条件も追加します。Edit の GUI を表示して、Filter の後ろに半角スペースを入力し、CTRL+Space を押下します。選択肢の中から AND を選びダブルクリックします。さらに半角スペースを入力し、CTL+Space を押下し、選択肢の中から [**Saturation - サチュレーション**]をダブルクリックします。



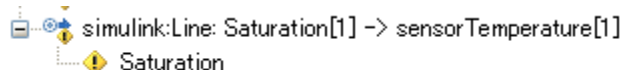
これで **simulink:Line:Sum[1]->** は、フィーチャモデルで[フィルタ処理あり]かつ[サチュレーション]が選択された時だけ Simulink モデルとして生成されるようリンクされました。



同様に、**simulink:Line:temperature[1]-> Saturation[1]**の **Restrictions タグ**で、“**NonFilter AND Saturation**”と設定します。



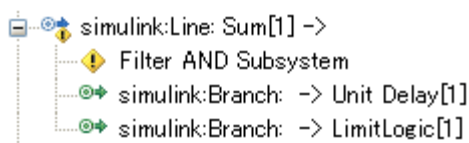
同様に、既存の **simulink:Line:Saturation[1] -> sensorTemperature[1]**の **Restrictions タグ**で、“**Saturation**”と設定します。



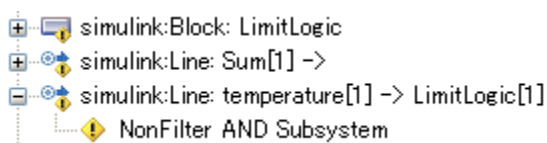
ここからは、新規の信号線になります。既存の信号線をコピーして修正していきます。

flow_control2.ccfm の **simulink:Line:Sum[1]->** を Copy し、flow_control.ccfm の

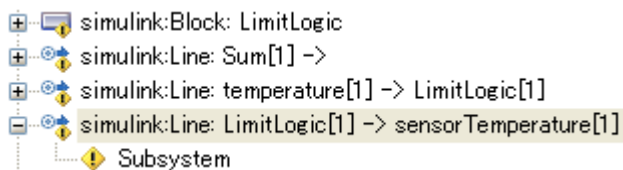
simulink:System:temperatureSensor に Paste します。この信号線の **Restrictions タグ**で、“**Filter AND Subsystem**” と設定します。



次に、**flow_control.ccfm** の **simulink:Line:temperature[1]->Saturation[1]** を Copy し、**flow_control.ccfm** の **simulink:System:temperatureSensor** に Paste します。この信号線の **Restrictions タグ**で、“**NonFilter AND Subsystem**” と設定します。

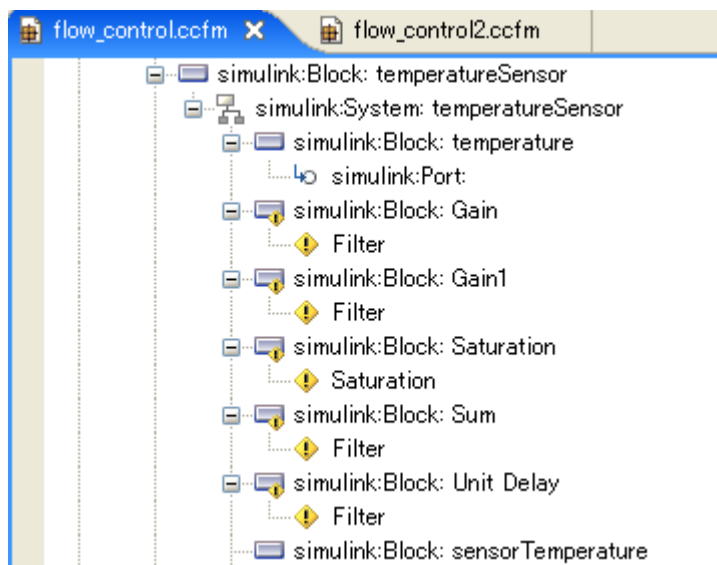


次に、**flow_control2.ccfm** の **simulink:Line:Limit[1]-> sensorTemperature[1]** を Copy し、**flow_control.ccfm** の **simulink:System:temperatureSensor** に Paste します。この信号線の **Restrictions タグ**で、“**Subsystem**” と設定します。

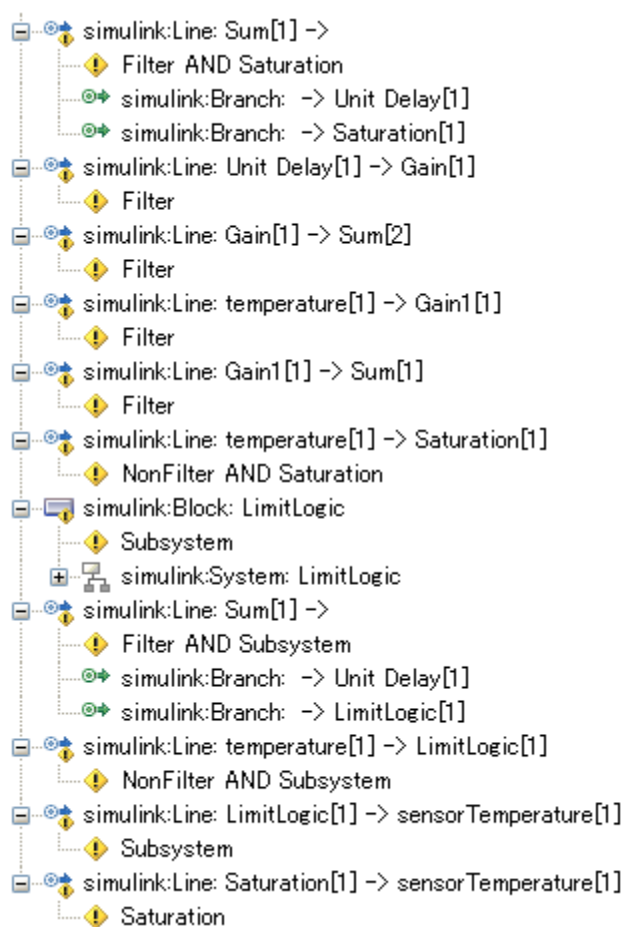


これで、[リミット処理]の設定が完了しました。ブロックと信号線は以下のようにになっているかと思います。

[ブロック]

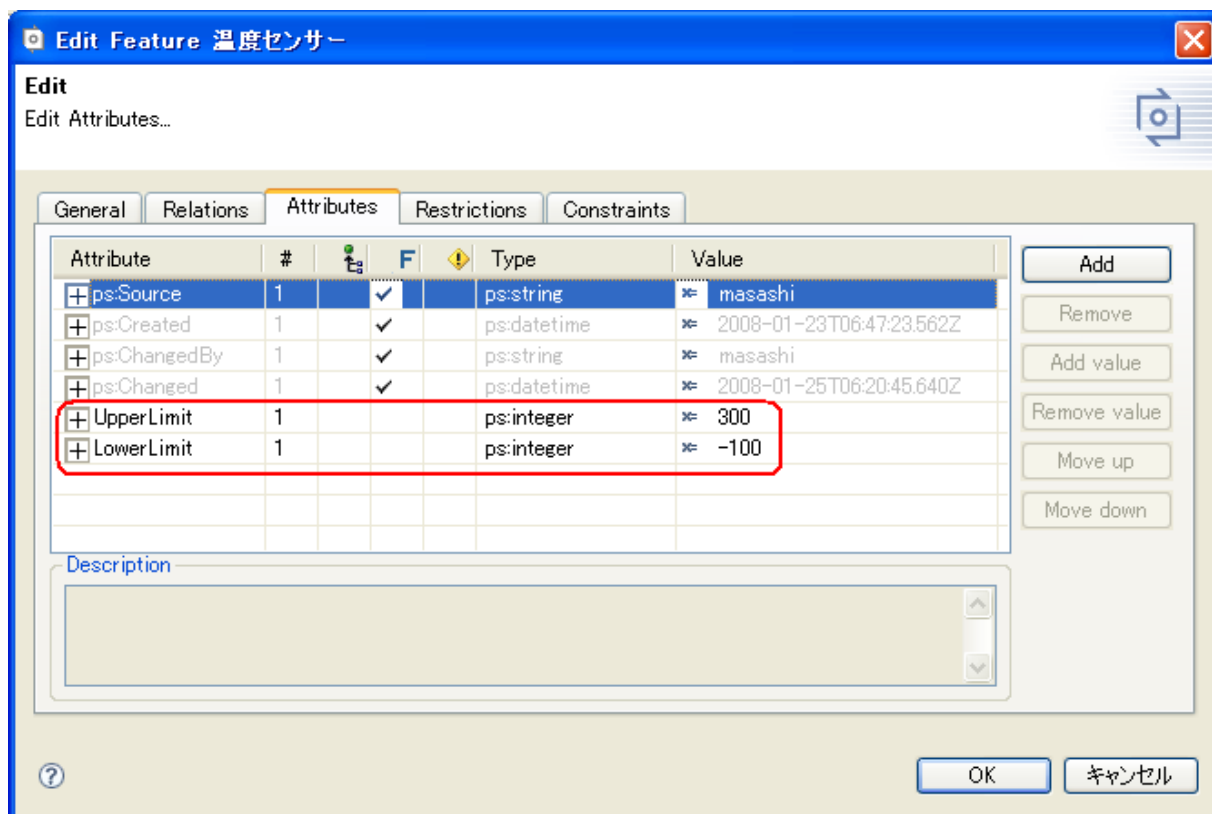


[信号線]



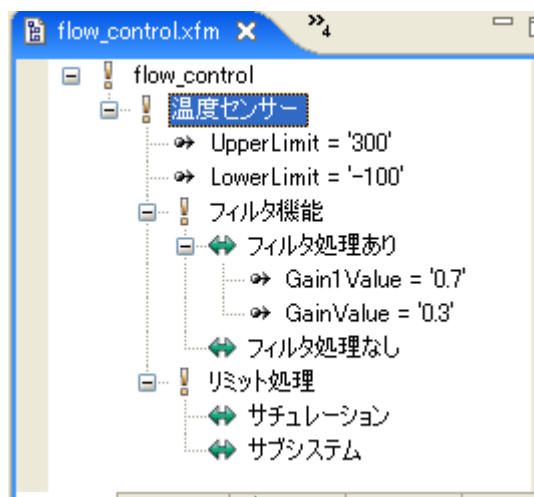
リミット処理の上下限值を可変にする

temperatureSensor サブシステムの[温度センサー]に属性を設定します。[温度センサー]をダブルクリックし、プロパティを開きます。Attributes タグをクリックし、[Add]ボタンを押下し Attribute を追加します。以下のように、Attribute 欄に UpperLimit と入力、Type 欄から ps:integer を選択し、Value 欄にデフォルト値 (300) を入れておきます。F 欄のチェックは外しておきます (フィーチャモデルやバリエーションモデル上で値を変更できるようにする為)。同様に、[Add]ボタンを押下して、デフォルト値 (-100) の LowerLimit を作成します。



[OK]ボタンを押下します。

フィーチャモデル上で右クリックし、[Show InTree]から[Show All]を選択すると、先程設定した **Attribute** が参照できます。

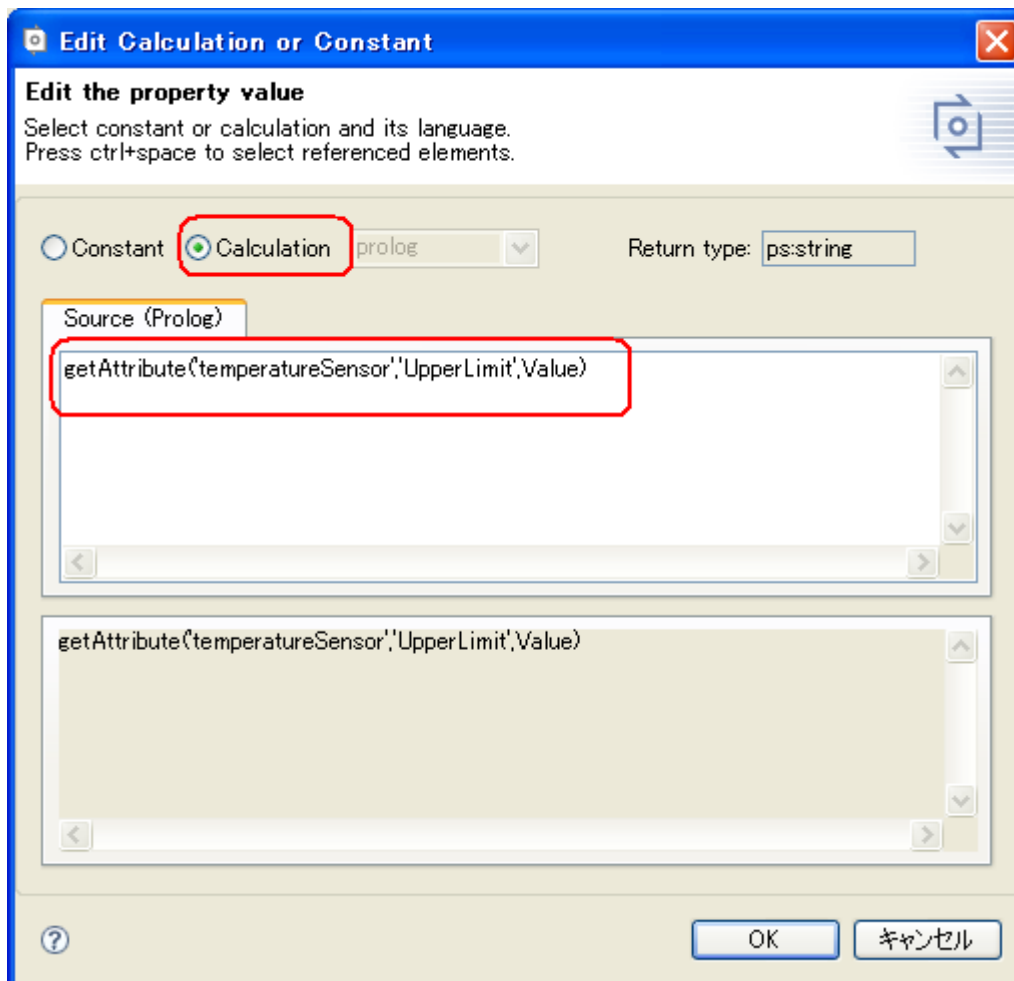


次に、これらの変数がファミリーモデル側に反映されるように設定します。

ファミリーモデルの **simulink:Block: Saturation** をダブルクリックし、プロパティを開きます。**Attributes タグ**をクリックし、Attribute である UpperLimit の **Value** 欄を以下のように変更します。

```
UpperLimit : getAttribute('temperatureSensor', 'UpperLimit', Value)
```

Value 欄をクリックして、右端に出るボタンを押下して Edit 画面を表示します。以下のように設定します。

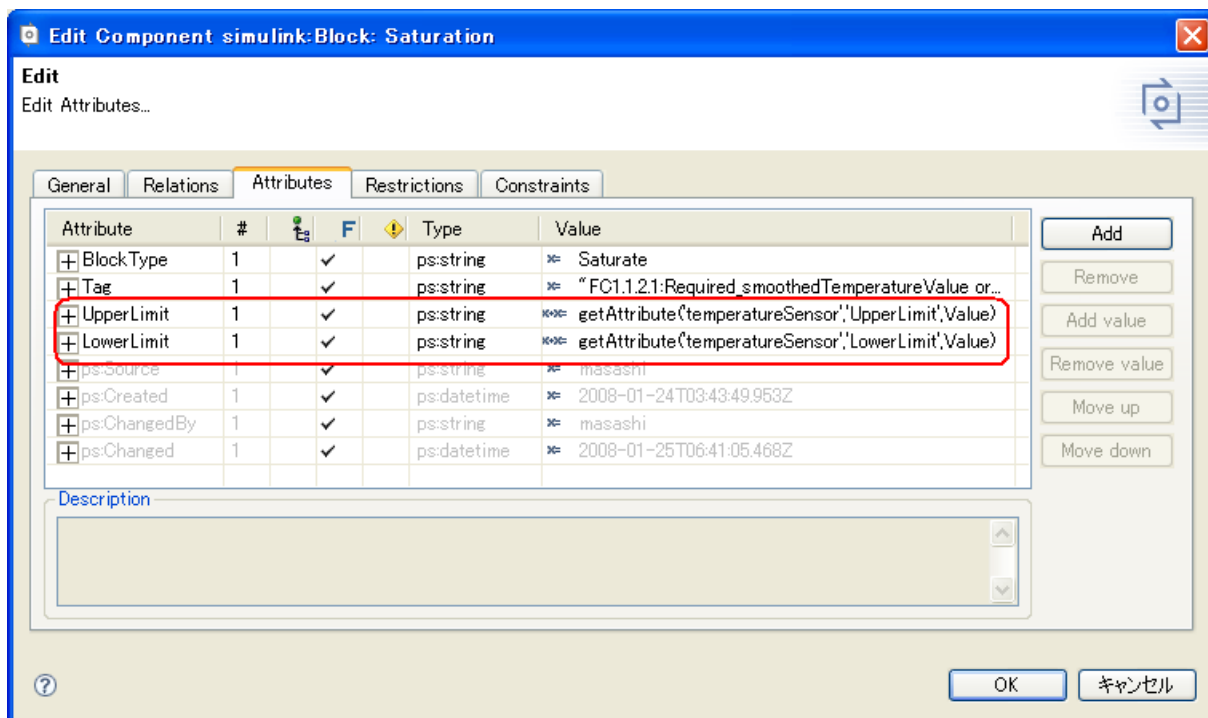


これは、フィーチャモデルの `temperatureSensor`(温度センサー)の Attribute である `UpperLimit` の Value 欄の値を取得するという式になります。

同様に、`LowerLimit` の Value 欄を以下のように変更します。

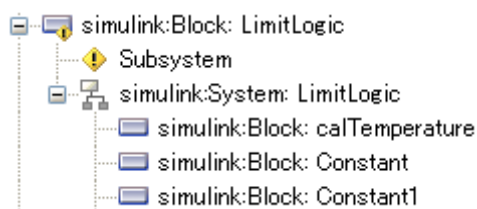
`LowerLimit` : `getAttribute('temperatureSensor', 'LowerLimit',Value)`

これで、フィーチャモデルで設定した `UpperLimit`、`LowerLimit` の値が、ファミリーモデル (Simulink モデル) に反映されるようになります。



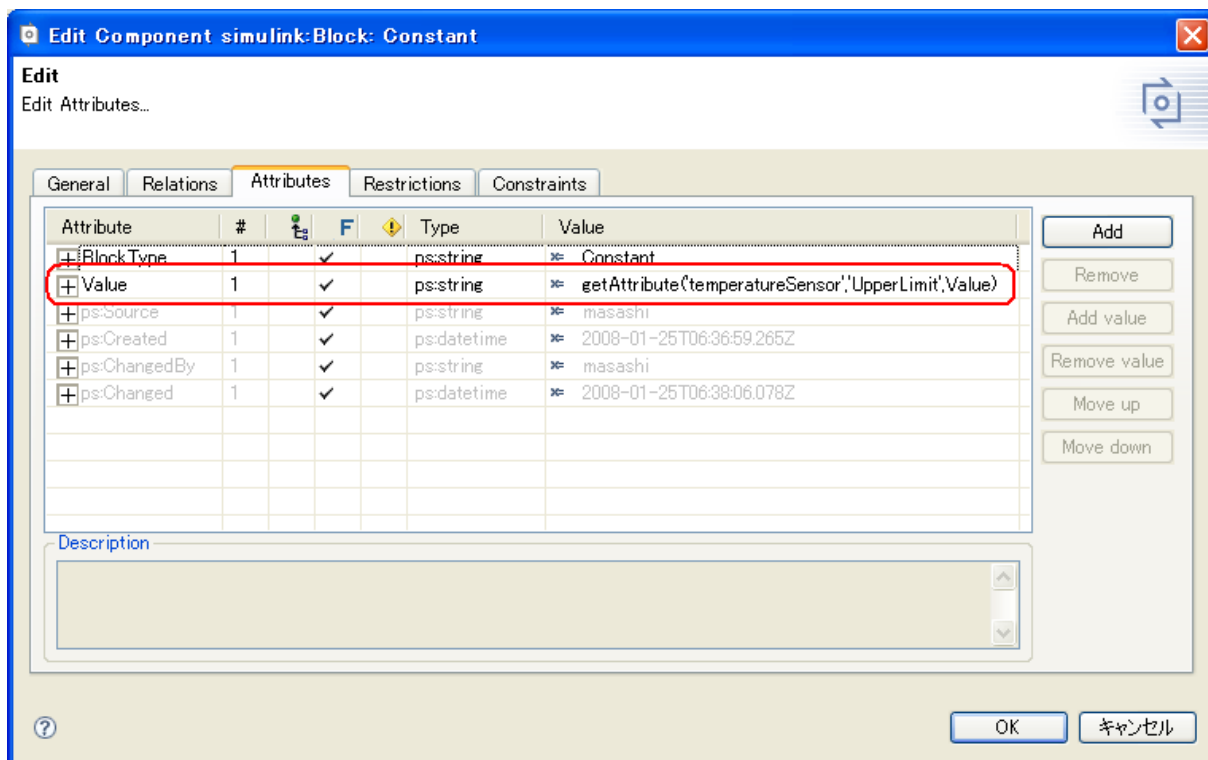
さらに、サブシステムでも UpperLimit、LowerLimit は使用しているので、こちらも Saturation ブロックと同様に設定する必要があります。

simulink:Block:LimitLogic 内の **simulink:System:LimitLogic** の **simulink:Block:Constant** と **simulink:Block:Constant1** です。

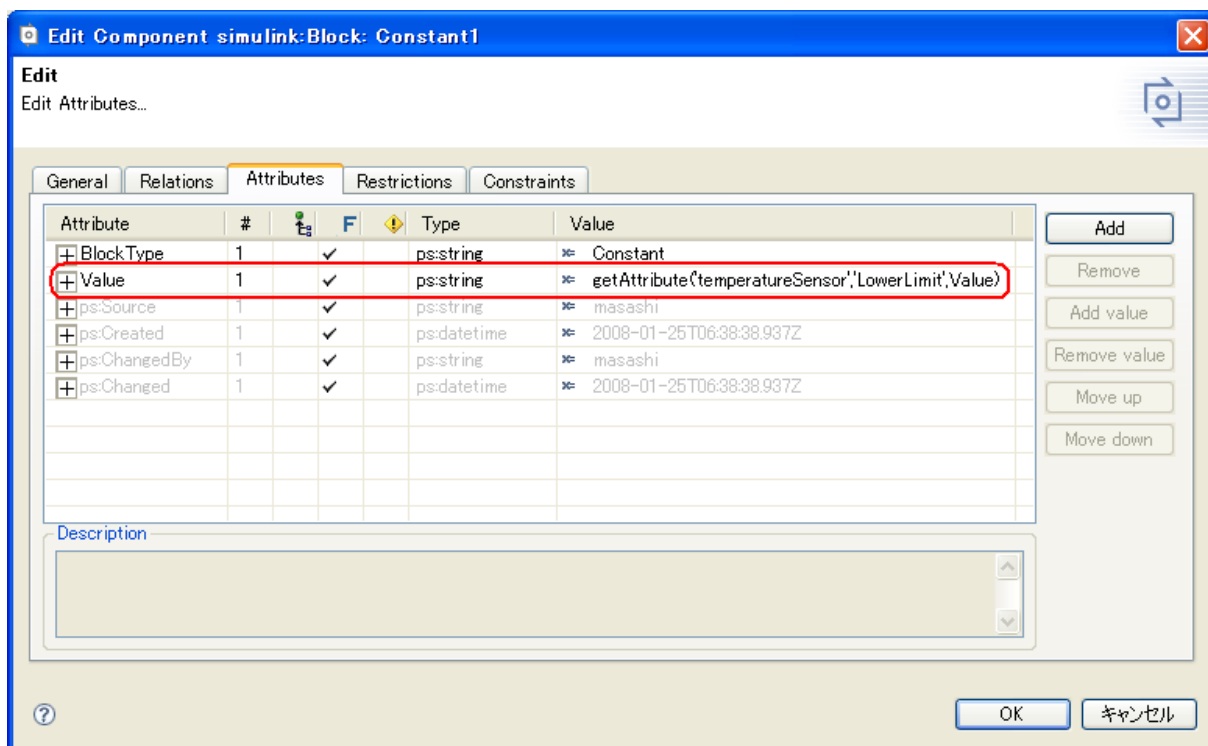


simulink:Block:Constant には UpperLimit、**simulink:Block:Constant1** には LowerLimit を設定します。(getAttribute 式の設定は今までと同じです)

[Constant]

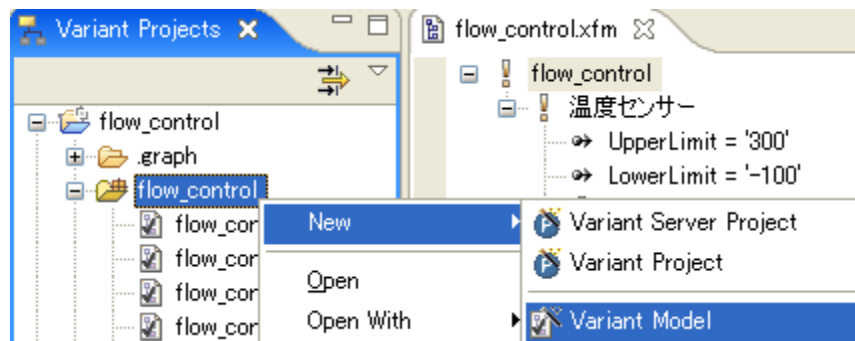


[Constant1]

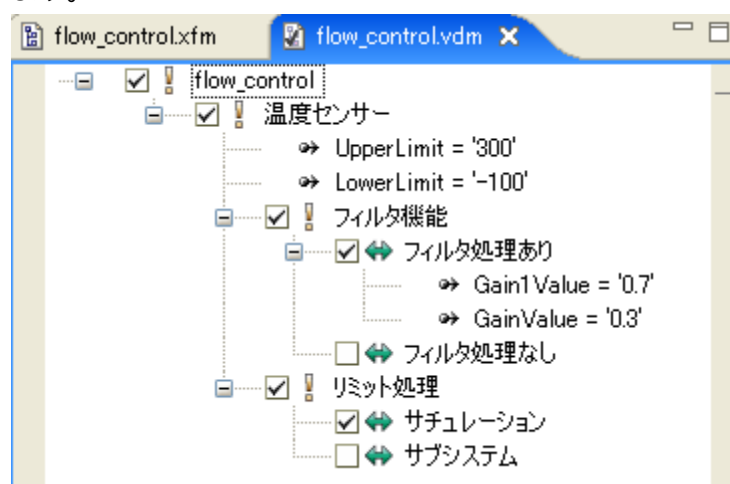


以上で、変動部分の作成は完了しました。

以下、flow_control から[New] - [Variant Model]をクリックします。



flow_control.vdm が作成されます。ここで右クリックから[Show In Tree] - [Show All]を選択します。



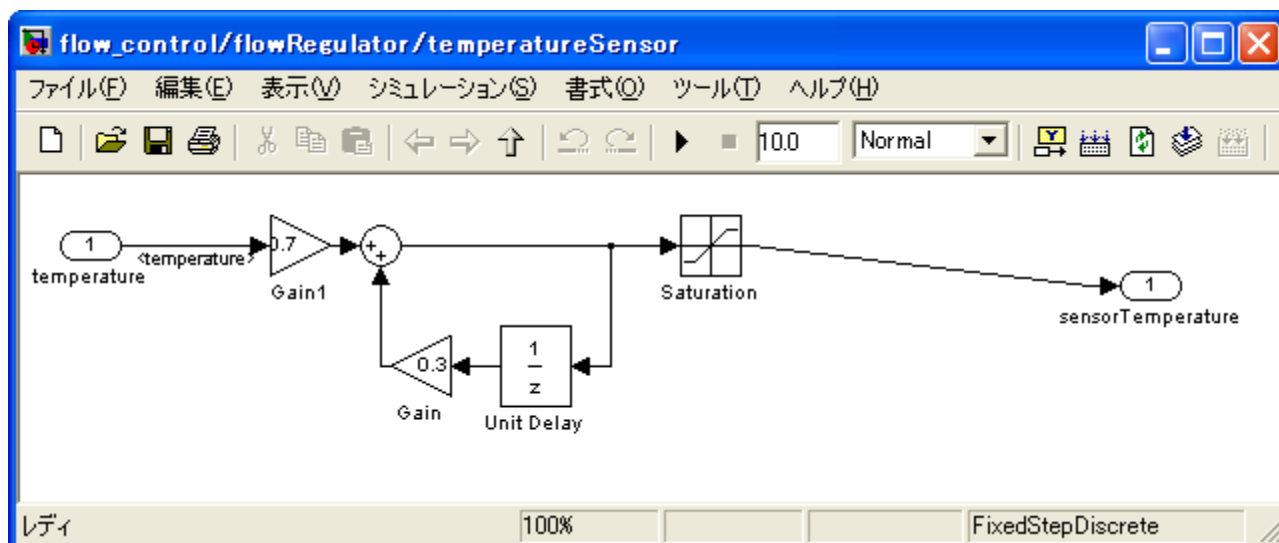
組合せとしては、下記の 4 パターンが生成されます。[フィルタ処理あり]と[フィルタ処理なし]は同時に選択不可、[サチュレーション]と[サブシステム]は同時に選択不可になっています。

(UpperLimit、LowerLimit、GainValue、Gain1Value は可変です)

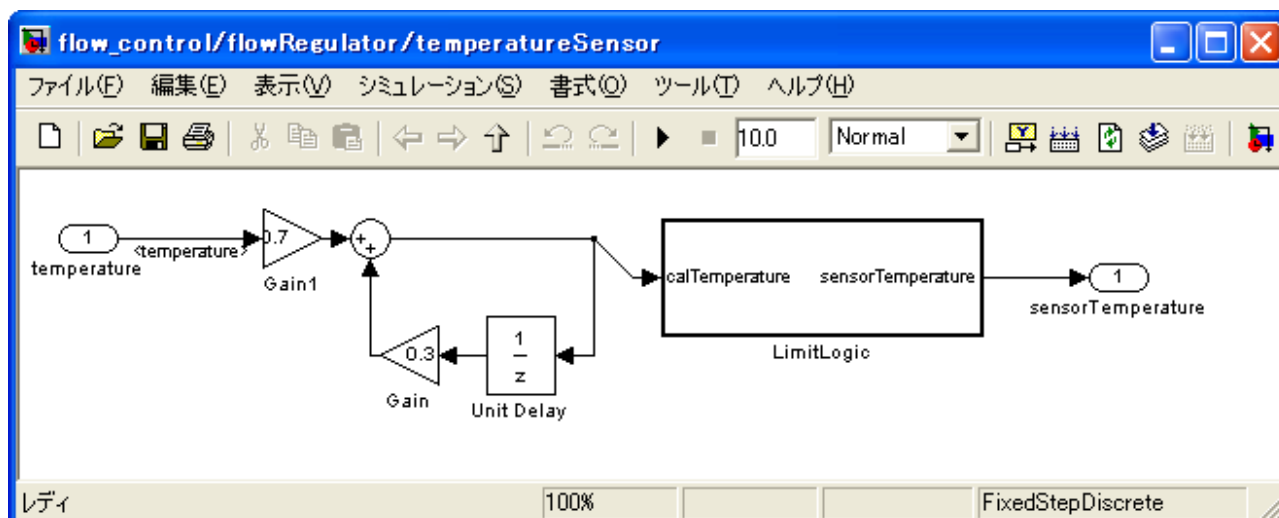
1. フィルタ処理あり - リミット処理 (サチュレーション)
2. フィルタ処理あり - リミット処理 (サブシステム)
3. フィルタ処理なし - リミット処理 (サチュレーション)
4. フィルタ処理なし - リミット処理 (サブシステム)

上記 4 パターンの temperatureSensor サブシステムは、以下のようになります。

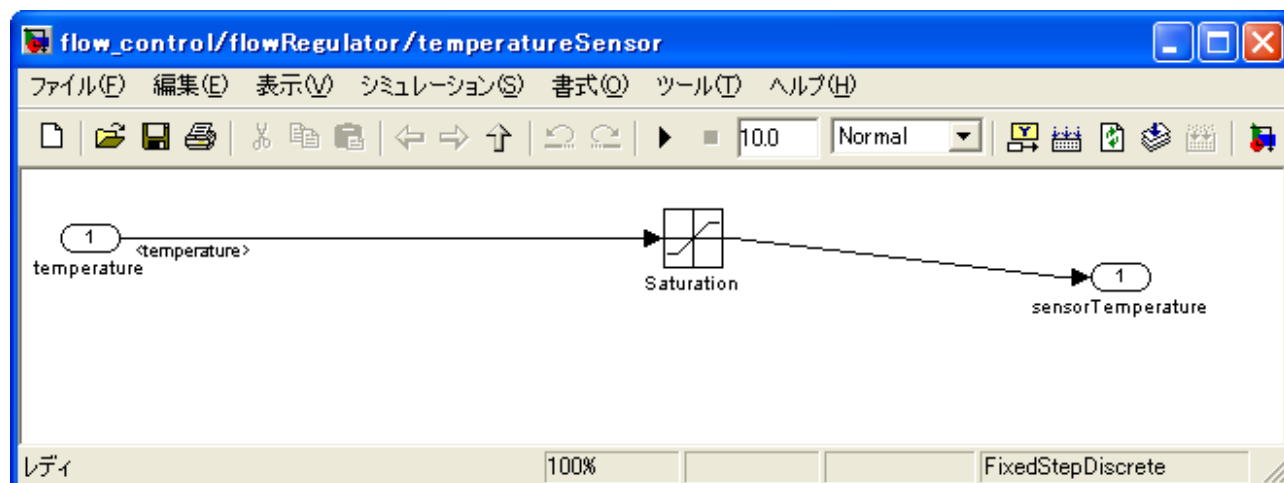
[1 . フィルタ処理あり - リミット処理 (サチュレーション)]



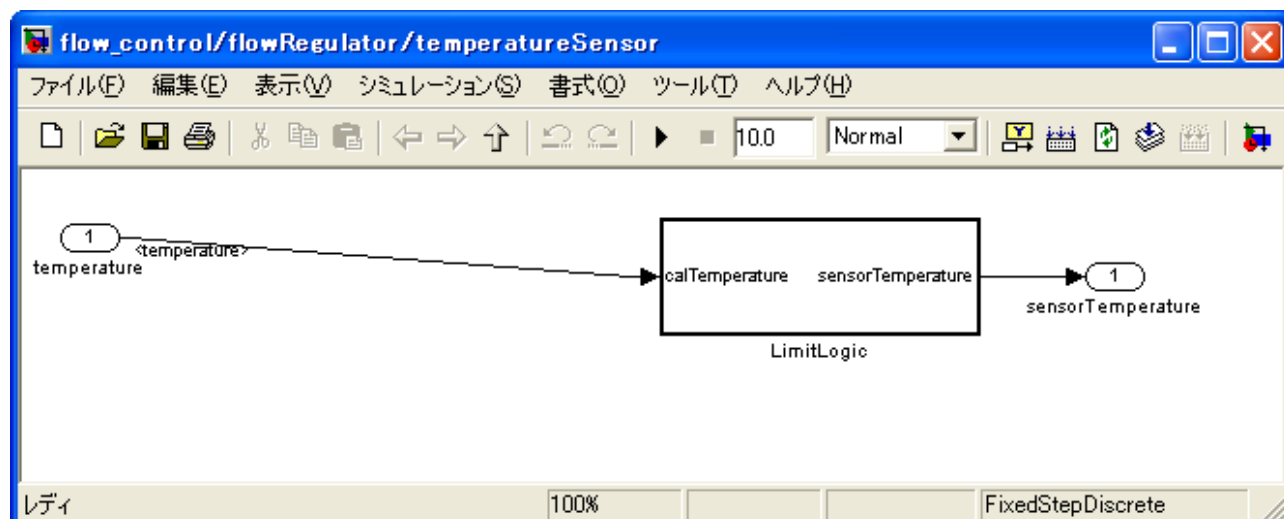
[2 . フィルタ処理あり - リミット処理 (サブシステム)]



[3 . フィルタ処理なし - リミット処理 (サチュレーション)]

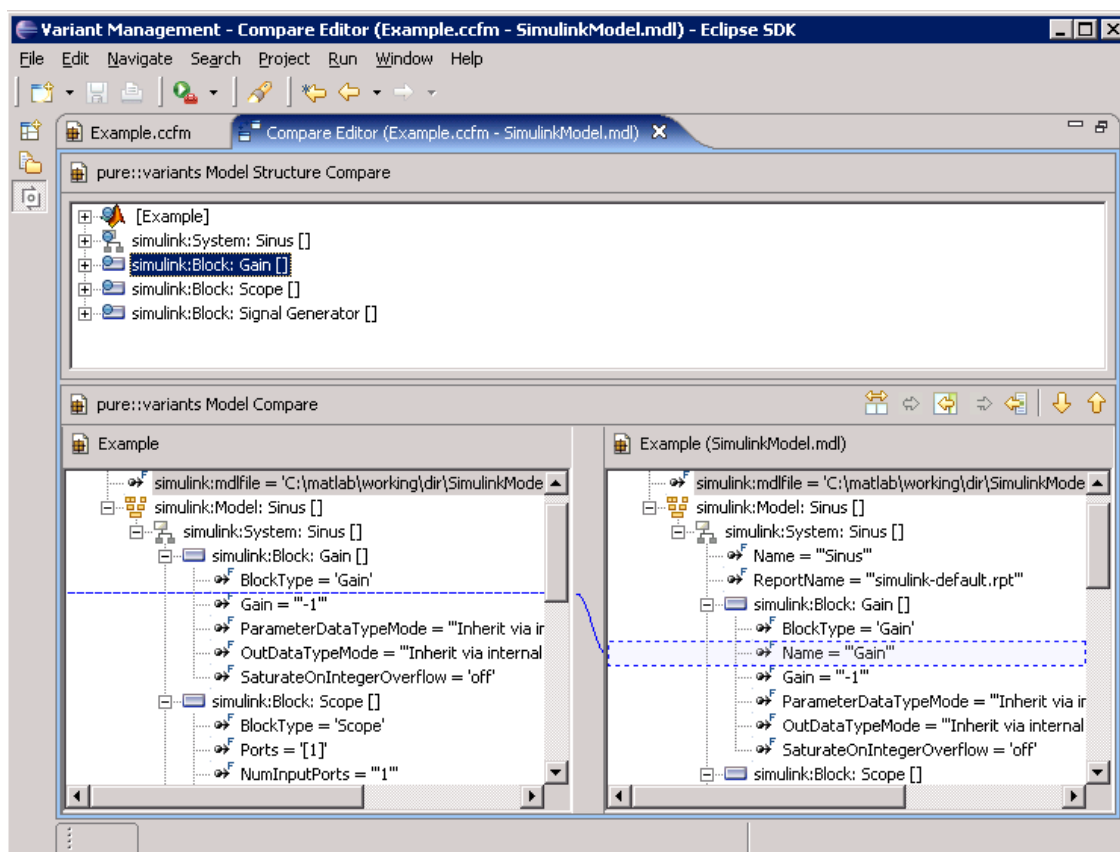


[4 . フィルタ処理なし - リミット処理 (サブシステム)]



2 . シンクロナイザー機能を使用する方法

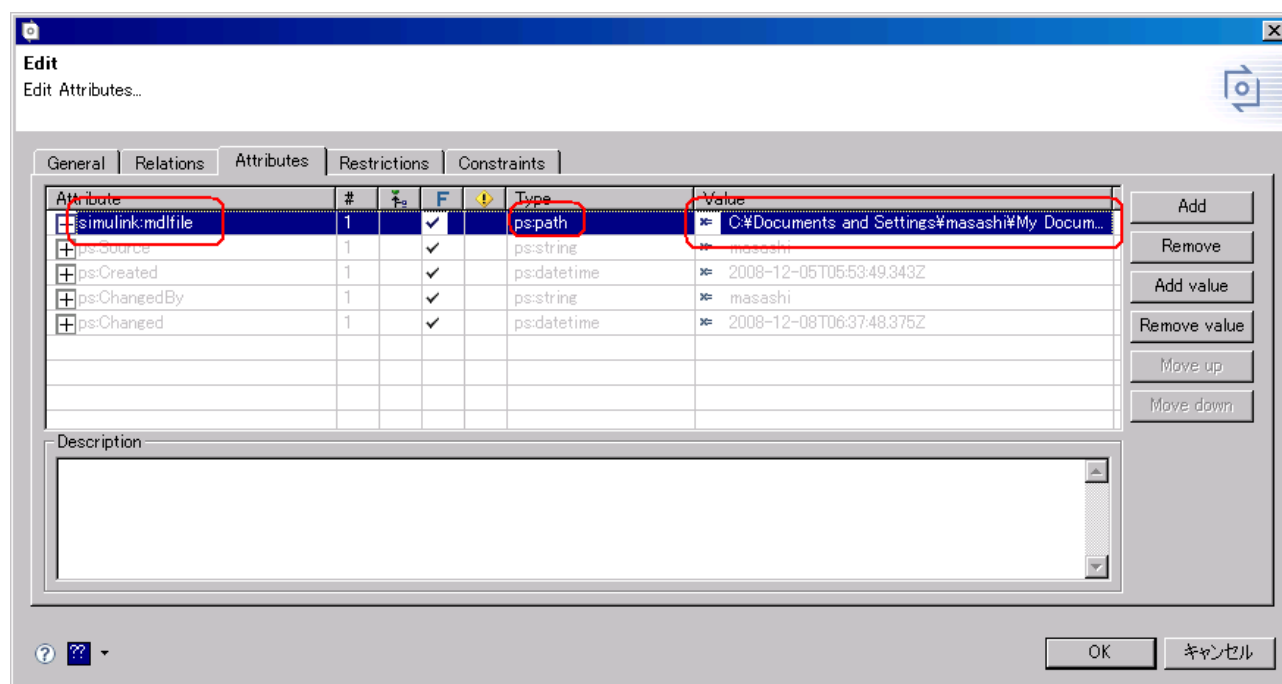
シンクロナイザー機能で、インポートされたファミリーモデルと別の Simulink モデル間で同期をとります。Compare Editor を開いてファミリーモデルと Simulink モデルを比較し、エディタ上部で異なる部分をリストで表します。このリストで選択すると、両方のモデルでそれぞれの場所がハイライトされます。Compare Editor を使って、変動部分といった部品などを Simulink モデル（右側）からファミリーモデル（左側）に追加することができます。




ファミリーモデル内に、Simulink モデルへのパス情報が格納されています。このパス情報を変更することで、様々な Simulink モデルとファミリーモデルを比較し、Simulink モデルからファミリーモデルに部品（変動部分）を追加することができます。ただし、ファミリーモデルと比較する Simulink モデルのモデル名は、ファミリーモデルでインポートした Simulink モデル名と同じでなければなりません。モデル名が異なると、比較した時に全体が丸々異なると判断されてしまい、部品を選択して追加することができません。

ここでは、シンクロナイザー機能の手順を紹介します。変動部分の設定などの詳細については、上述の『1 . 別の Simulink モデルをインポートして行う方法』をご覧ください。

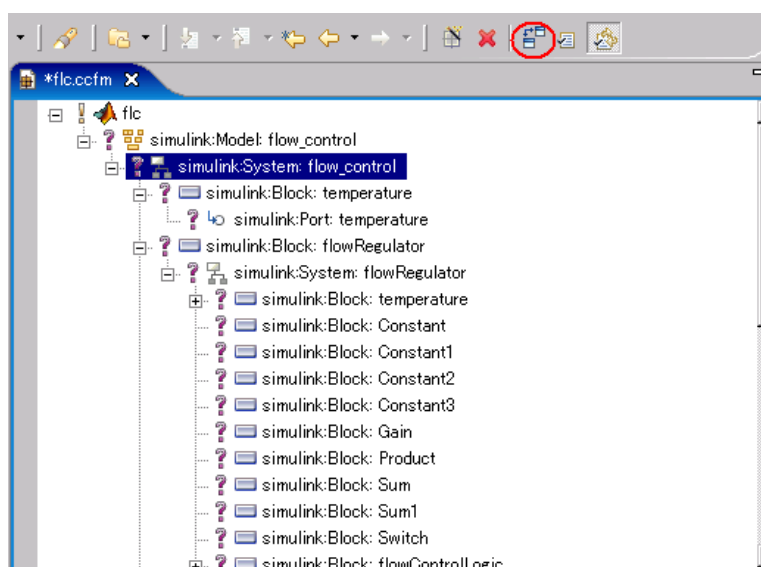
Simulink モデルへのパス情報の変更方法は、ファミリーモデルの最上位をクリックして右クリック、コンテキストメニューから Properties を選択、Attributes タグをクリックします。



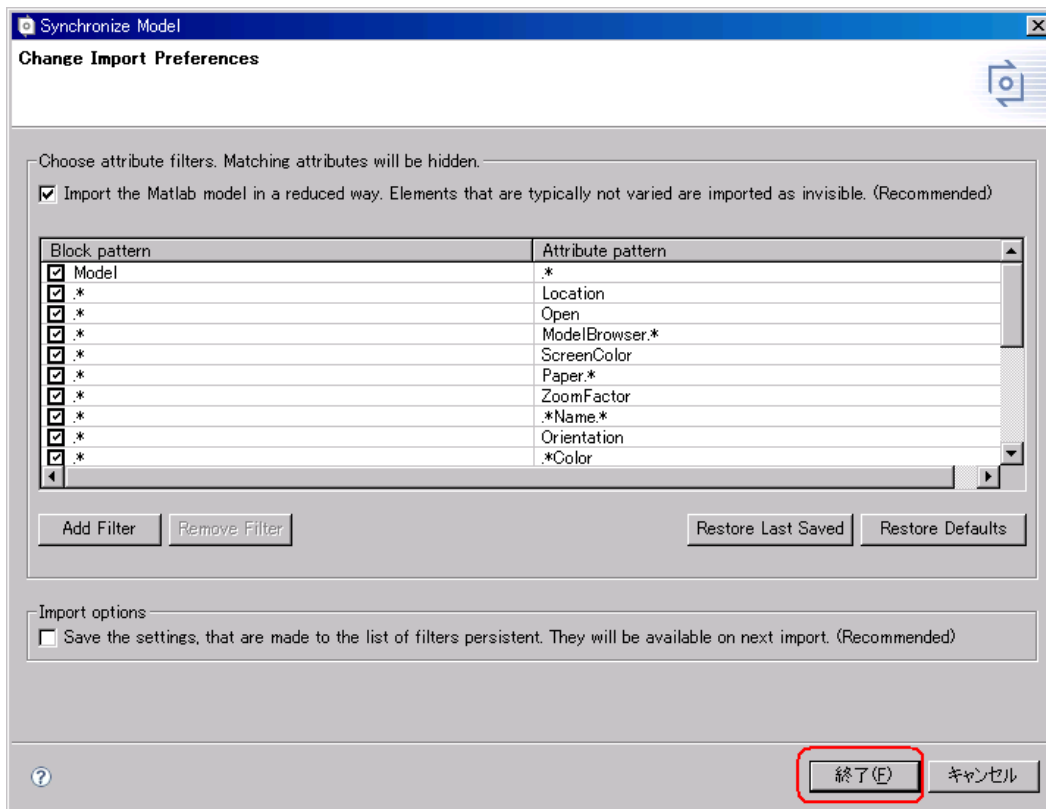
一番上の simulink:mdlfile の Value 欄をクリックすると、右端にボタン  が出てくるのでクリックします。現在設定されているモデルパス情報が確認できるので、比較対象モデルを変更したい場合はここを変更し、OK をクリックします（上記エディタでも OK をクリックします）。

<手順>

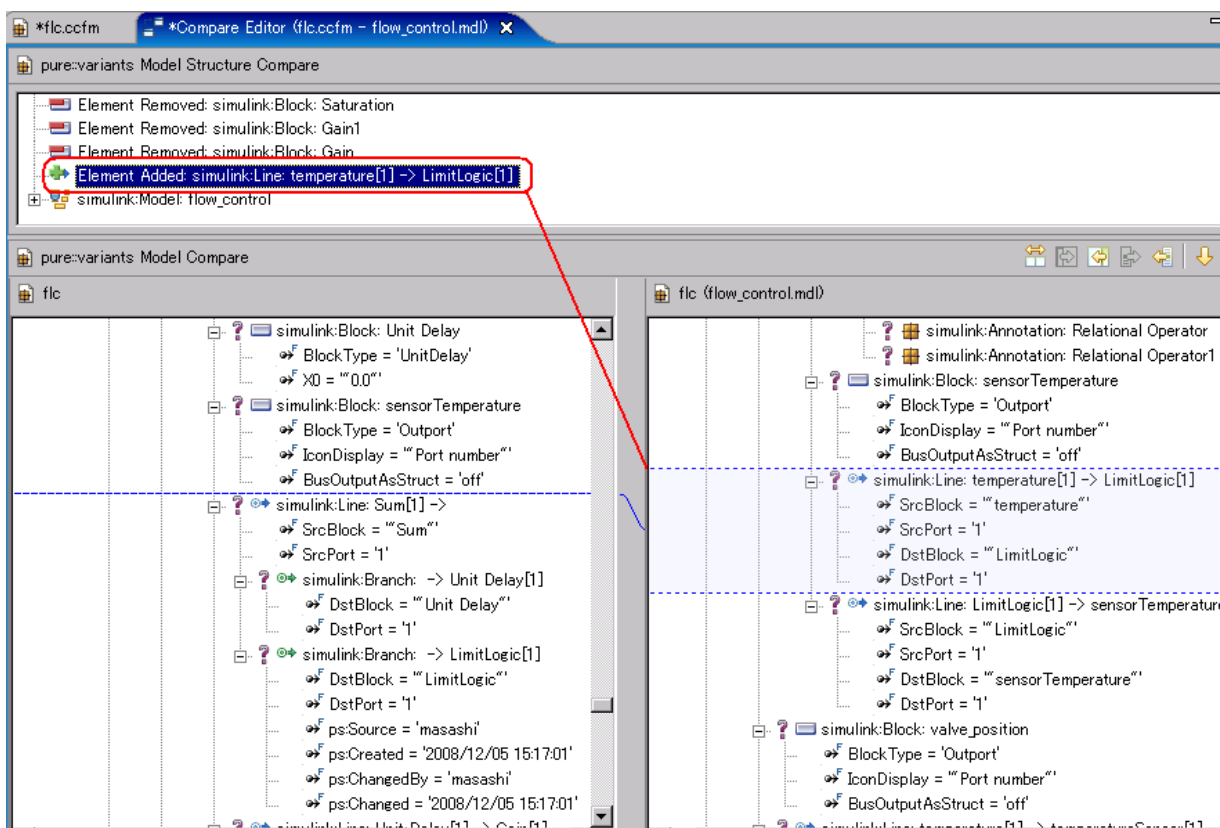
ファミリーモデルと Simulink モデルを同期するために、ファミリーモデルの何か要素をクリックすることで、シンクロナイザーボタンがアクティブになるので、それをクリックします。



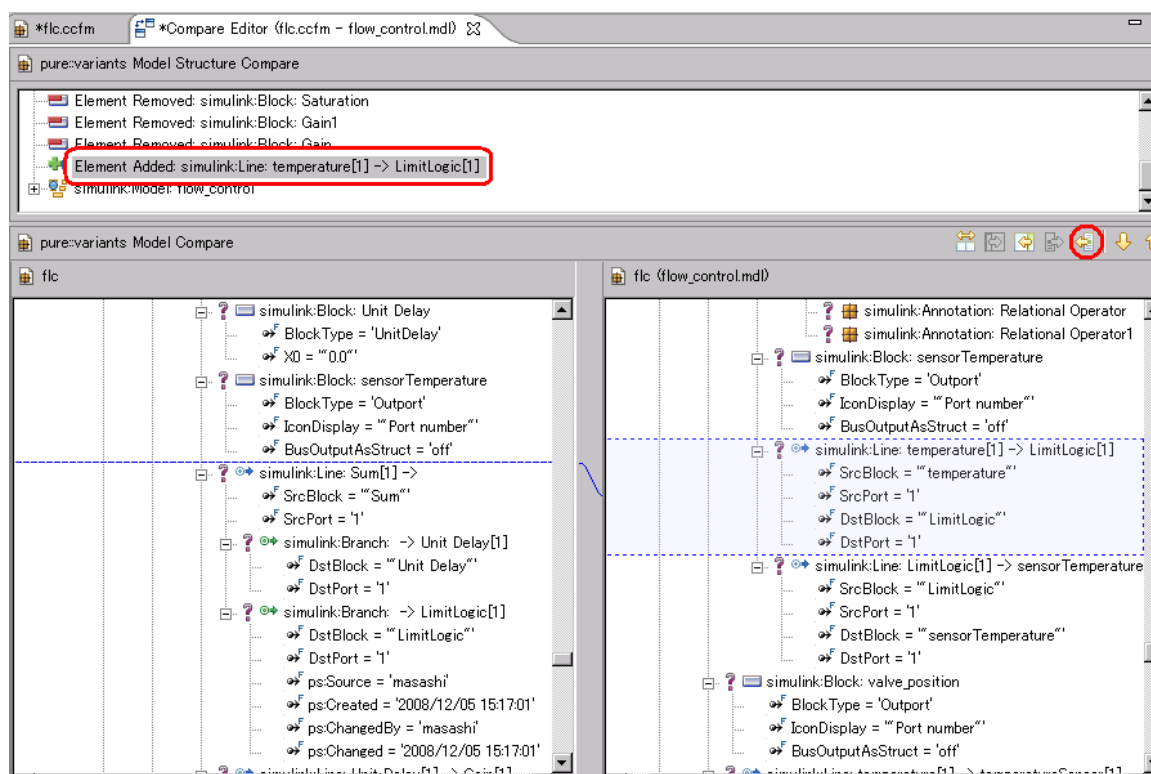
Synchronize Model ダイアログが開きますので、終了ボタンをクリックします。



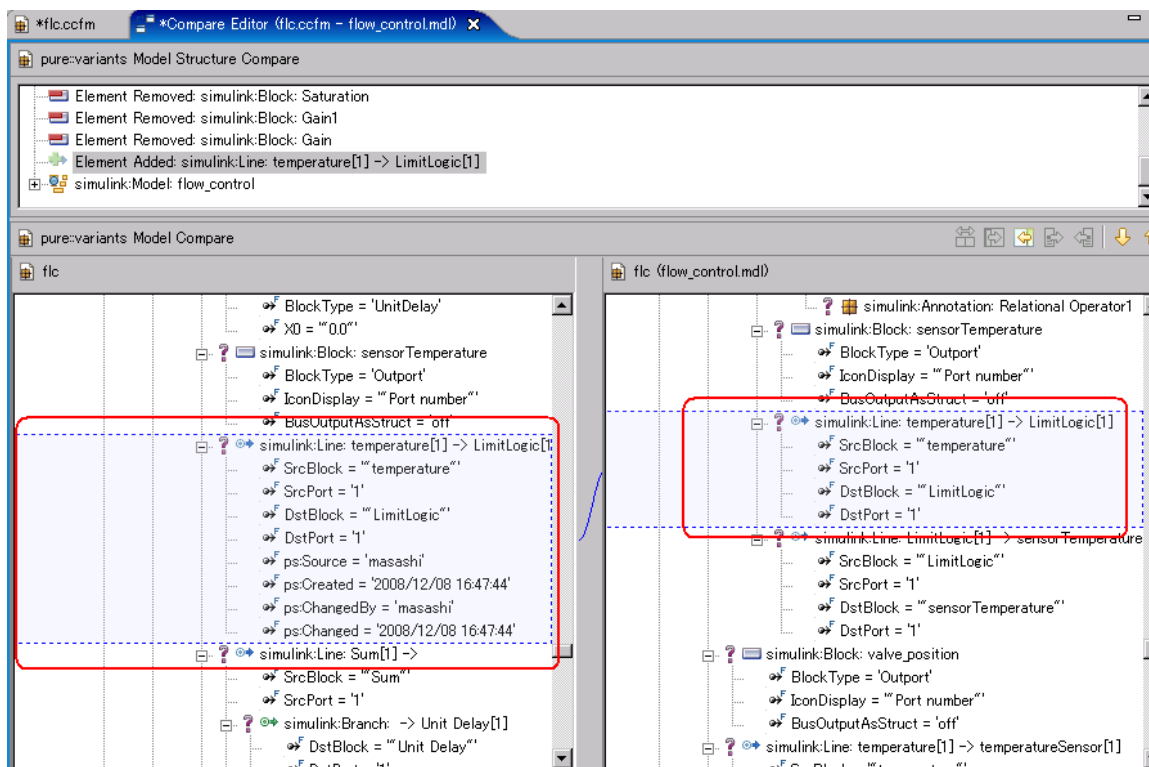
Compare Editor が開きます。(以下の場合、Simulink モデルにあってファミリーモデルにないものが選択されています)



この信号線を Simulink モデル（右側）からファミリーモデル（左側）に追加します。リストで選択された状態で、Compare Editor の “ Copy Current Change from Right to Left ” ボタンをクリックします。



そうすると、Simulink モデルからファミリーモデルに信号線が追加されます。



このように、ファミリーモデルに部品を追加したい Simulink モデルを比較対象モデルとして設定し、部品を追加するといった作業を繰り返し行います。

以上のサンプルを評価用に希望される場合は、以下の連絡先まで要求いただくと幸いです。



富士設備工業株式会社 電子機器事業部

〒591-8025 大阪府堺市北区長曾根町1928-1

Tel : 072-252-2128 www.fuji-setsu.co.jp