CONTROL ALGORITHM MODELING GUIDELINES USING MATLAB®, Simulink®, and Stateflow® Version 5.1(日本語版)

Japan MBD Automotive Advisory Board (JMAAB)

2018 年 03 月 30 日

（誤字訂正 2018 年 8 月 31 日）

■ 著作権について

- 本ドキュメントの著作権は JMAAB に帰属します。
- JMAAB は、本文書の内容に関し、いかなる保証もするものではありません。万一本文書を利用して不具合等があった場合でも JMAAB は一切責任を負いかねます。また、本文書に記載されている事項は予告なしに変更または廃止されることがありますので、あらかじめご了承ください。

■ 本ドキュメントの取扱いについて

- 本文書は、非営利目的、または利用者内部で使用する場合に限り、複製が可能です。また、本文書を引用する場合は、本文書からの引用であることを明示し、引用された著作物の題号や著作者名を明示する等の引用の要件を満たす必要があります。
- 本成果物については JMAAB ホームページ(http://jmaab.mathworks.jp/)を参照下さい。
- その他のお問い合わせは、JMAAB 事務局(jmaab-office@mathworks.co.jp)へご連絡下さい。

目次

1. はじめに

1.1. ガイドラインとは…

自動車用制御装置のコントローラモデルを運用する上で、作成者と使用者の間で容易に共通の理解が得られるように、Simulink / Stateflow モデルの記述について重要な基本的なルールを規定したものです。

主に、以下の事を目的として制定しました。

- 可読性の向上
  - モデルのグラフィカルな理解度向上
  - モデルの機能を解析する可読性向上
  - 接続ミスの防止
  - コメントなど
- シミュレーション、検証
  - シミュレーションが実行できる仕組み
  - テストの容易化
- コード生成
  - コード効率の向上(ROM、RAM 効率)
  - 生成コードのロバスト性確保
- その他

尚、Ver 5.0 では、MATLAB バージョン R2010b から R2015a を対象としてルールが作られています。モデル実行時にエラーとなり、実装できない事項はルール化していません。

1.2. ガイドラインのテンプレート

ガイドラインは以下のテンプレートに従って作成されています。独自でガイドラインを作成する場合も、このテンプレートの使用を推奨します。

注意：このテンプレートは、ガイドラインを正しく理解する為、最低限必要な項目を表にしたものです。既存の項目のいずれにも重複しない限り、新たな項目をこのテンプレートに追加することができます。

ルール ID：タイトル	**xx_nnnn：ガイドラインのタイトル(短く、他と重複しない)**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
xn	(ガイドラインの説明)	(パラメーター名)
	【正】 (記述内容に正しいイメージ及びコメント)	
	【誤】 (記述内容に誤りのイメージ及びコメント)	
根拠		
サブ ID	**記述内容**	
xn	(根拠内容)	

1.2.1. ルール ID

ルール ID は、アルファベットの小文字 2 文字(ガイドライン著者の識別)と、4 桁の数字をアンダースコアでつないだものを使用します。ID は、永久固定とし変更されることはなく、ガイドラインを参照する時に使用します。

db、jm、hd、ar は Ver1.0 制定メンバーが使用した ID です。Ver2.0 以降で **na、jc** の ID が使われています。

1.2.2. サブ ID

サブ ID は、ルール ID 一つに対して複数存在し、「採否選択式」と「排他選択式」があります。採否選択式は、アルファベットの小文字 1 文字（アルファベット順）で使用します。採否選択式のサブ ID の採用可否は使用者の任意となります。排他選択式は、アルファベットの小文字 1 文字（アルファベット順）と、1 桁の数字を結

合したものを使用します。排他選択式は、採否を判断し採用とした場合にいずれかを選択するものとなります。

例
xy_0000 → xy_0000a 採否選択式（採用可否は使用者の任意）
→ xy_0000b1 排他選択式（採用する場合、xy_0000b1 か xy_0000b2 どちらかを選択）
→ xy_0000b2 排他選択式（採用する場合、xy_0000b1 か xy_0000b2 どちらかを選択）

1.2.3. タイトル

タイトルは、そのガイドラインを短く説明したユニークなものを設定します。

1.2.4. 記述内容

記述内容は、図表等を用いてガイドラインルールの詳細な内容を記述しています。

記述内で使用している記号について

記述内の表記	説明	記述例
[ ] (角括弧)	ブロック名	[Outport]
{ } (波括弧)	ブロックパラメーター名 Stateflow パラメーター名 コンフィギュレーションパラメーターの設定項目	{伝播信号の表示}
“ ” (ダブルクォーテーション)	パラメーター等の設定値	“0”

1.2.5. カスタムパラメーター

カスタムパラメーター欄に記述のあるルールは、記述されている項目についてプロジェクトの特性に合わせて対象範囲を決定する必要があります。

記述内容には対象や値の例を記載していますが、プロジェクトのプロセスや制御対象の条件、携わるエンジニアの平均的なレベルから総合的に判断し、チェック時に適用する内容を決定してください。

1.2.6. 根拠

ガイドラインルールの記述内容について、可読性や検証効率、コード生成後のコード効率等の観点から利用を推奨する理由を記述しています。

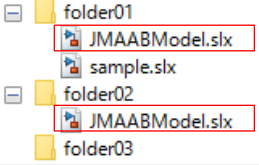
1.3. ガイドラインの構成

1 章は、本書の解説、2 章から 5 章にガイドラインルールを掲載しています。
6 章は、ガイドラインルールの根拠分類と関連ガイドラインのリストを掲載しています。
7 章は、用語についての説明を記載しています。
8 章から 9 章は、上級者が必要とする運用、モデルアーキテクチャについて記載しています。
10 章は、本ガイドラインの改定内容や履歴をまとめています。
11 章は、Simulink/Stateflow 機能の解説やモデリングノウハウについて記載しています。

2. Simulink / Stateflow 共通

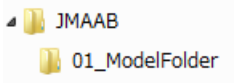
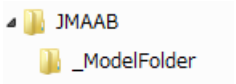
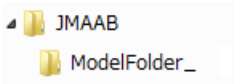
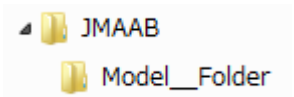
2.1. 命名規則(一般)

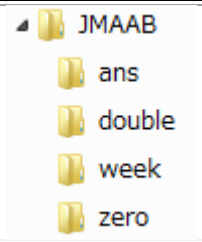
2.1.1. ar_0001：ファイル名に使用できる文字

ルール ID：タイトル	ar_0001：ファイル名に使用できる文字	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	ファイル名で使用可能な文字の種類は、下記のみとします。 ・半角英数字 ・半角アンダースコア 備考：改行、半角スペース、全角文字、制御文字は使用しません。 モデルファイル、MATLAB スクリプト等、確認対象とするファイルの種別をプロジェクトで定めます。	ファイル(拡張子)
	【誤】 JMAAB Model.slx　半角スペースが使用されています。 JMAAB 設定.m　全角文字が使用されています。 JMAAB-Model.p JMAAB(Model).mdl　記号が使用されています。	
b	ファイル名は、先頭に数字を使用しません。	ファイル(拡張子)
	【誤】 001_JMAABModel.slx	
c	ファイル名は、先頭にアンダースコアを使用しません。	ファイル(拡張子)
	【誤】 _JMAABModel.slx	
d	ファイル名は、末尾にアンダースコアを使用しません。	ファイル(拡張子)
	【誤】 JMAABModel_.slx	
e	ファイル名は、連続したアンダースコアを使用しません。	ファイル(拡張子)
	【誤】 JMAAB__Model.slx	
f	ファイル名は、完全一致する MATLAB 予約語を使用しません。	ファイル(拡張子)
	【誤】 ans.slx double.slx week.slx zero.slx　etc.	
g	MATLAB のパスを通したファイル名は、同一にしません。	ファイル(拡張子)
	【誤】 MATLAB のパスを通ったフォルダーに同一名称のファイルを配置しています。 	
根拠		

サブ ID	記述内容
abcf	・可読性が損なわれます。 ・一般的な作法から外れることにより、想定外の問題が起こるリスクがあります。
de	・可読性が損なわれます。
g	・同一名称のファイルが複数ある場合、パスの優先順位が高い方のファイルが読み込まれます。これにより、意図しないファイルを読み込む可能性があります。 ・可読性が損なわれます。 ・一般的な作法から外れることにより、想定外の問題が起こるリスクがあります。

2.1.2. ar_0002：フォルダー名に使用できる文字

ルール ID：タイトル	ar_0002：フォルダー名に使用できる文字	
ルール		
サブ ID	記述内容	カスタムパラメーター
a	フォルダー名で使用可能な文字の種類は、下記のみとします。 ・半角英数字 ・半角アンダースコア 備考：改行、半角スペース、全角文字、制御文字は使用しません。	-
	【誤】  記号が使用されています。 半角スペースが使用されています。 全角文字が使用されています。	
b	フォルダー名は、先頭に数字を使用しません。	-
	【誤】 	
c	フォルダー名は、先頭にアンダースコアを使用しません。	-
	【誤】 	
d	フォルダー名は、末尾にアンダースコアを使用しません。	-
	【誤】 	
e	フォルダー名は、連続したアンダースコアを使用しません。	-
	【誤】 	
f	フォルダー名は、完全一致する MATLAB 予約語にしません。	-
	【誤】	



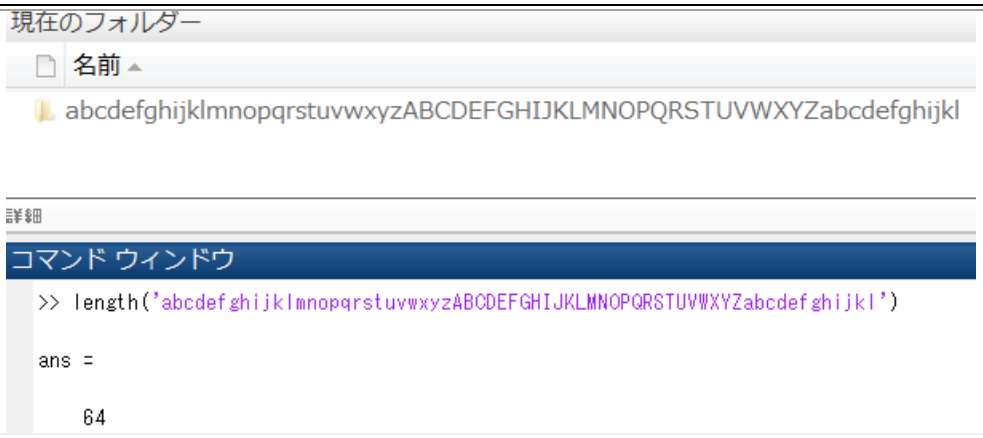
根拠	
サブ ID	**記述内容**
abcdef	・可読性が損なわれます。 ・一般的な作法から外れることにより、想定外の問題が起こるリスクがあります。

2.1.3. jc_0241 ： モデルファイル名の文字数制限

ルール ID : タイトル	**jc_0241 ： モデルファイル名の文字数制限**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	モデルファイル名(ドット、及び拡張子を除いた部分)の最大文字数は、63 文字以内とします。	モデルファイル名の最大文字数
根拠		
サブ ID	**記述内容**	
a	・ファイル名が長いとモデルリファレンスで参照不可となるおそれがあります。	

2.1.4. jc_0242 ： フォルダー名の文字数制限

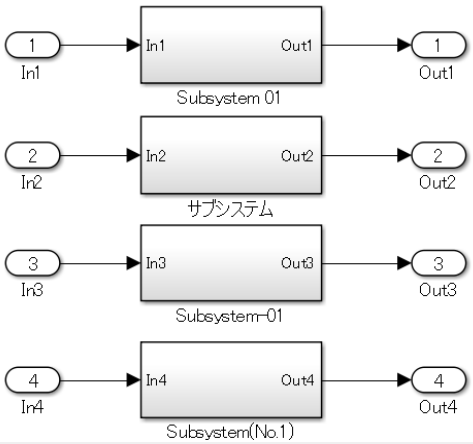
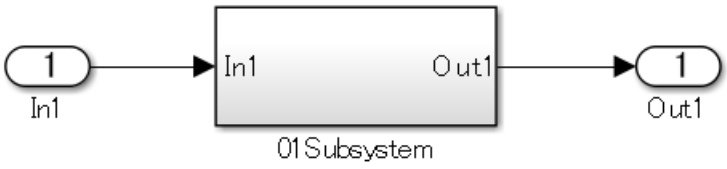
ルール ID : タイトル	**jc_0242 ： フォルダー名の文字数制限**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	フォルダー名の最大文字数は、63 文字以内とします。	フォルダー名の最大文字数
【正】 フォルダー名が 63 文字以内です。  【誤】 フォルダー名が 64 文字で、63 文字よりも多いです。		

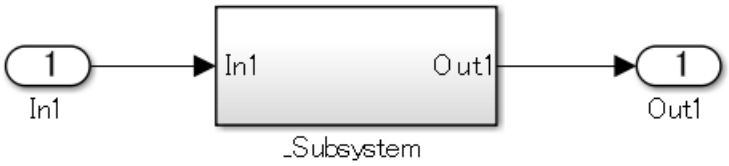
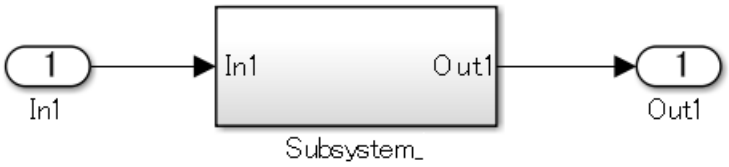
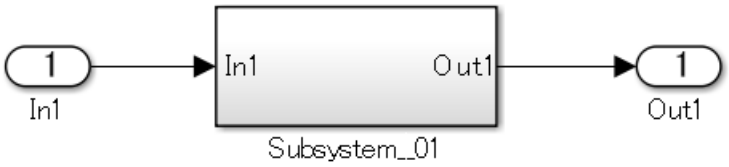


根拠	
サブ ID	**記述内容**
a	・逸脱した場合、GUI 等でパス名を表示しきれない場合があります。

2.2. 命名規則(モデル)

2.2.1. jc_0201 ： サブシステム名に使用できる文字

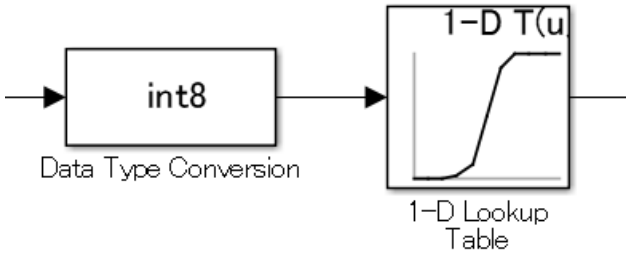
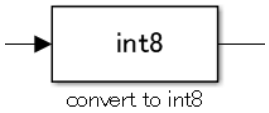
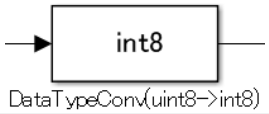
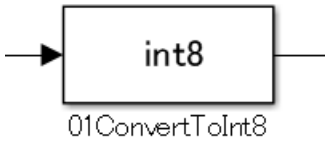
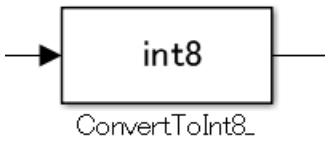
ルール ID：タイトル	**jc_0201 ： サブシステム名に使用できる文字**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	構造サブシステム名で使用可能な文字の種類は、下記のみとします。 ・半角英数字 ・半角アンダースコア 備考：改行、半角スペース、全角文字、制御文字は使用しません。	-
	【誤】  半角スペースが使用されています。 全角文字が使用されています。 記号が使用されています。	
b	構造サブシステム名は、先頭に数字を使用しません。	-
	【誤】  	
c	構造サブシステム名は、先頭にアンダースコアを使用しま	-

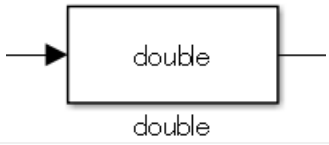
	せん。	
	【誤】  	
d	構造サブシステム名は、末尾にアンダースコアを使用しません。	-
	【誤】  	
e	構造サブシステム名は、連続したアンダースコアを使用しません。	-
	【誤】  	
f	構造サブシステム名は、完全一致する MATLAB 予約語にしません。	-
	【誤】  	

根拠	
サブ ID	**記述内容**
abf	・設定した構造サブシステム名でコード生成ができなくなります。
cde	・設定した構造サブシステム名でコード生成ができなくなる可能性があります。

2.2.2. jc_0231 : ブロック名に使用できる文字

ルール ID : タイトル	**jc_0231 : ブロック名に使用できる文字**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	[Inport]、[Outport]を除く基本ブロック名で使用可能な文字の種類は、下記のみとします。 ・半角英数字 ・半角アンダースコア 備考:ユーザーが新たにブロック名を付ける場合には、改行や半角スペースは使用しません。(Simulink ライブラリに登録されている初期状態のブロック名に使用されている場合は使用可能です) また、全角文字、制御文字等も使用しません。	-

	Simulink ライブラリに登録されている初期状態のブロック名の例  【誤】  半角スペースが使用されています。  全角文字が使用されています。  記号が使用されています。 	
b	[Inport]、[Outport]を除く基本ブロック名は、先頭に数字を使用しません。	-
	【誤】 	
c	[Inport]、[Outport]を除く基本ブロック名は、先頭にアンダースコアを使用しません。	-
	【誤】 	
d	[Inport]、[Outport]を除く基本ブロック名は、末尾にアンダースコアを使用しません。	-
	【誤】 	
e	[Inport]、[Outport]を除く基本ブロック名は、連続したアンダースコアを使用しません。	-
	【誤】 	
f	[Inport]、[Outport]を除く基本ブロック名は、完全一致するMATLAB 予約語にしません。	-

	【誤】 

根拠	
サブ ID	**記述内容**
ab	・逸脱した場合、モデルとコードの整合が取りづらくなります。
c	・逸脱した場合、可読性が悪くなります。 ・MISRA-C:2004 に逸脱する可能性があります。
d	・逸脱した場合、可読性が悪くなります。 アンダースコアは、一般的に単語の区切りとして用いるため、記述ミスのように見えます。
e	・逸脱した場合、可読性が悪くなります。
f	・逸脱した場合、可読性が悪くなります。 ・一般的な作法から外れることにより、想定外の問題が起こるリスクがあります。

2.2.3. jc_0211 : Inport ブロック / Outport ブロックに使用できる文字

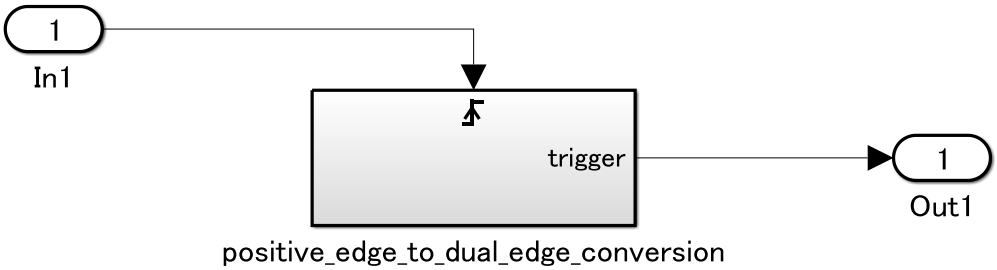
ルール ID : タイトル	**jc_0211 : Inport ブロック / Outport ブロックに使用できる文字**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	[Inport]、[Outport]のブロック名で使用可能な文字の種類は下記のみとします。 ・半角英数字 ・半角アンダースコア 備考:改行、半角スペース、全角文字、制御文字等は、使用しません。	-
	【誤】  半角スペースが使用されています。 全角文字が使用されています。 記号が使用されています。	
b	[Inport]、[Outport]のブロック名は、先頭に数字を使用しません。	-
	【誤】 	
c	[Inport]、[Outport]のブロック名は、先頭にアンダースコアを使用しません。	-
	【誤】 	

d	[Inport]、[Outport]のブロック名は、末尾にアンダースコアを使用しません。	-
	【誤】 	
e	[Inport]、[Outport]のブロック名は、連続したアンダースコアを使用しません。	-
	【誤】 	
f	[Inport] 、[Outport]のブロック名は、完全一致する MATLAB 予約語にしません。	-
	【誤】 	

根拠	
サブ ID	**記述内容**
ab	・逸脱した場合、モデルとコードの整合が取りづらくなります。
c	・逸脱した場合、可読性が悪くなります。 ・MISRA-C:2004 に逸脱する可能性があります。
d	・逸脱した場合、可読性が悪くなります。 アンダースコアは、一般的に単語の区切りとして用いるため、記述ミスのように見えます。
e	・逸脱した場合、可読性が悪くなります。
f	・逸脱した場合、可読性が悪くなります。 ・一般的な作法から外れることにより、想定外の問題が起こるリスクがあります。

2.2.4. jc_0243：サブシステム名の文字数制限

ルール ID：タイトル	**jc_0243：サブシステム名の文字数制限**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	構造サブシステム名の最大文字数は、63 文字以内とします。	サブシステム名の最大文字数
	【正】 サブシステム名が 63 文字以内です。	



根拠	
サブ ID	**記述内容**
a	・コード生成できない可能性があります。

2.2.5. jc_0247 : ブロック名の文字数制限

ルール ID : タイトル	jc_0247 : ブロック名の文字数制限	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	[Inport]、[Outport]を除く基本ブロック名の最大文字数は、63 文字以内とします。	ブロック名の最大文字数
根拠		
サブ ID	**記述内容**	
a	・コード生成できない可能性があります。	

2.2.6. jc_0244 : Inport ブロック名 / Outport ブロック名の文字数制限

ルール ID : タイトル	jc_0244 : Inport ブロック名 / Outport ブロック名の文字数制限	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	[Inport]、[Outport]のブロック名の最大文字数は、63 文字以内とします。	Inport ブロック名の最大文字数 Outport ブロック名の最大文字数
根拠		
サブ ID	**記述内容**	
a	・コード生成できない可能性があります。	

2.2.7. jc_0222 : 信号名 / バス名に使用できる文字

ルール ID : タイトル	jc_0222 : 信号名 / バス名に使用できる文字	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	信号名、バス名で使用可能な文字の種類は下記のみとしま	-

	す。 ・半角英数字 ・半角アンダースコア 備考:改行、半角スペース、全角文字、制御文字は使用しません。	
b	信号名、バス名は、先頭に数字を使用しません。	-
c	信号名、バス名は、先頭にアンダースコアを使用しません。	-
d	信号名、バス名は、末尾にアンダースコアを使用しません。	-
e	信号名、バス名は、連続したアンダースコアを使用しません。	-
f	信号名、バス名は、完全一致する MATLAB 予約語にしません。	-

根拠

サブ ID	記述内容
ab	・逸脱した場合、モデルとコードの整合が取りづらくなります。
c	・逸脱した場合、可読性が悪くなります。 ・場合によっては MISRA-C:2004 に逸脱する可能性があります。
d	・逸脱した場合、可読性が悪くなります。 アンダースコアは一般的に単語の区切りとして用いるため、記述ミスのように見えます。
e	・逸脱した場合、可読性が悪くなります。
f	・逸脱した場合、可読性が悪くなります。 ・一般的な作法から外れることにより、想定外の問題が起こるリスクがあります。

2.2.8. jc_0232 : パラメーター名に使用できる文字

ルール ID : タイトル	jc_0232 : パラメーター名に使用できる文字

ルール

サブ ID	記述内容	カスタムパラメーター
a	パラメーター名で使用可能な文字の種類は下記のみとします。 ・半角英数字 ・半角アンダースコア 備考:改行、半角スペース、全角文字、制御文字等は使用しません。	-
b	パラメーター名は、先頭に数字を使用しません。	-
c	パラメーター名は、先頭にアンダースコアを使用しません。	-
d	パラメーター名は、末尾にアンダースコアを使用しません。	-
e	パラメーター名は、連続したアンダースコアを使用しません。	-
f	パラメーター名は、完全一致する MATLAB 予約語にしません。	-

根拠

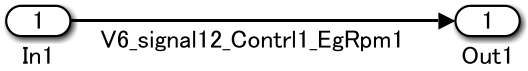
サブ ID	記述内容
ab	・逸脱した場合、モデルとコードの整合が取りづらくなります。
c	・逸脱した場合、可読性が悪くなります。 ・MISRA-C:2004 に逸脱する可能性があります。
d	・逸脱した場合、可読性が悪くなります。

	アンダースコアは一般的に単語の区切りとして用いるため、記述ミスのように見えます。
e	・逸脱した場合、可読性が悪くなります。
f	・逸脱した場合、可読性が悪くなります。 ・一般的な作法から外れることにより、想定外の問題が起こるリスクがあります。

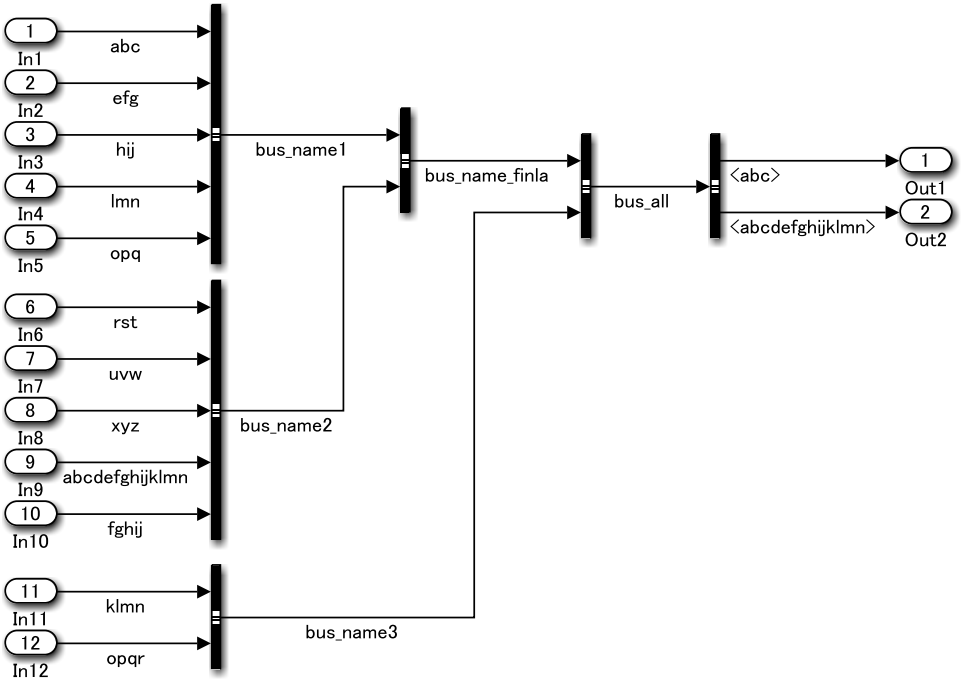
2.2.9. jc_0245 : 信号名 / バス名の文字数制限

ルール ID : タイトル	jc_0245 : 信号名 / バス名の文字数制限	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	信号名、バス名の最大文字数は、63 文字以内とします。	信号名の最大文字数 バス名の最大文字数

【正】
信号名が 63 文字以内です。



【正】
階層化されたバス信号の総文字数(フルパス)が 63 文字以内です。
bus_all.bus_name_finla.bus_name2.abcdefghijklmn



根拠	
サブ ID	**記述内容**
a	・コード生成できない可能性があります。

2.2.10. jc_0246 : パラメーター名の文字数制限

ルール ID : タイトル	jc_0246 : パラメーター名の文字数制限	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**

a	パラメーター名の最大文字数は、63 文字以内とします。	パラメーター名の最大文字数
根拠		
サブ ID	**記述内容**	
a	・コード生成できない可能性があります。	

2.2.11. jc_0795 : Stateflow データ名に使用できる文字

ルール ID : タイトル	**jc_0795 : Stateflow データ名に使用できる文字**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	Stateflow データの{名前}には、先頭にアンダースコアを使用しません。	-
b	Stateflow データの{名前}には、末尾にアンダースコアを使用しません。	-
c	Stateflow データの{名前}には、連続したアンダースコアを使用しません。	-
d	Stateflow データの{名前}には、完全一致する MATLAB 予約語にしません。	-
根拠		
サブ ID	**記述内容**	
abcd	・逸脱した場合、可読性が損なわれます。 ・逸脱した場合、意図したコードとならない可能性があります。	

2.2.12. jc_0796 : Stateflow データ名の文字数制限

ルール ID : タイトル	**jc_0796 : Stateflow データ名の文字数制限**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	Stateflow データの{名前}の最大文字数は、63 文字以内とします。	Stateflow データの名前の最大文字数
根拠		
サブ ID	**記述内容**	
a	・逸脱した場合、可読性が損なわれます。 ・逸脱した場合、意図したコードとならない可能性があります。	

2.2.13. jc_0791 : 定義データ名の重複

ルール ID : タイトル	**jc_0791 : 定義データ名の重複**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	ベースワークスペース、モデルワークスペース間でデータ名の重複は禁止します。	-
b	ベースワークスペース、データディクショナリ(sldd)間でデータ名の重複は禁止します。	データディクショナリの種類
c	モデルワークスペース、データディクショナリ(sldd)間でデータ名の重複は禁止します。	データディクショナリの種類

根拠	
サブ ID	**記述内容**
abc	・データ名が重複するとモデルが意図しない動作をする可能性があります。

2.2.14. jc_0792 : 未使用のデータ

ルール ID : タイトル	jc_0792 : 未使用のデータ	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	データディクショナリ(sldd)には、Simulink/Stateflow モデルで使用されるデータだけを定義します。	データディクショナリの種類
b	モデルワークスペースには、Simulink/Stateflow モデルで使用されるデータだけを定義します。	-
根拠		
サブ ID	**記述内容**	
ab	・未使用のデータが存在すると保守性、運用性に影響がある可能性があります。	

2.3. その他

2.3.1. db_0043 : モデルで使用するフォントとフォントサイズ

ルール ID : タイトル	db_0043 : モデルで使用するフォントとフォントサイズ	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	・ブロック名の{フォント}、{フォントスタイル}をプロジェクトで定めた設定にします。 ・信号名の{フォント}、{フォントスタイル}をプロジェクトで定めた設定にします。	フォント フォントスタイル
b	・ブロック名のフォントの{サイズ}をプロジェクトで定めた設定にします。 ・信号名のフォントの{サイズ}をプロジェクトで定めた設定にします。	フォントサイズ
c	・ステートラベルとボックス名の{フォント}、{フォントスタイル}をプロジェクトで定めた設定にします。 ・遷移ラベル、コメントの{フォント}、{フォントスタイル}をプロジェクトで定めた設定にします。	フォント フォントスタイル
d	・ステートラベルとボックス名のフォントの{サイズ}をプロジェクトで定めた設定にします。 ・遷移ラベル、コメントのフォントの{サイズ}をプロジェクトで定めた設定にします。	フォントサイズ
根拠		
サブ ID	**記述内容**	
ac	・フォントを統一することで、可読性が向上します。	
bd	・フォントサイズを統一することで、可読性が向上します。	

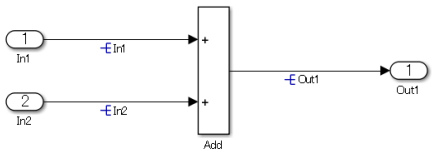
2.3.2. jc_0644 : 型の設定方法

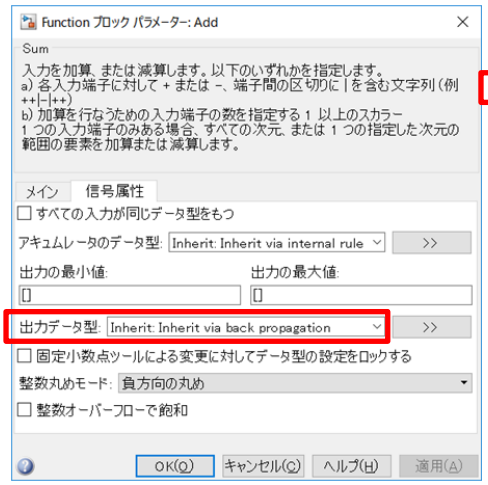
ルール ID : タイトル	jc_0644 : 型の設定方法

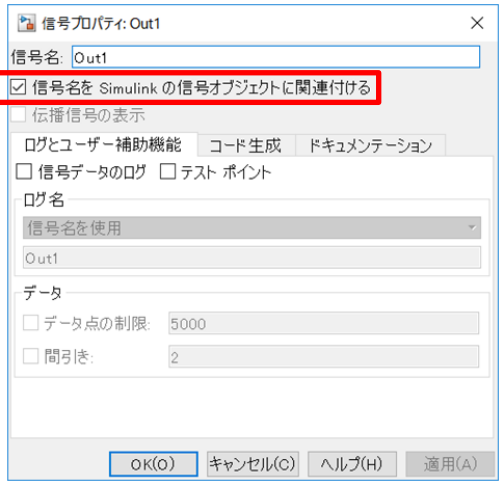
ルール

サブ ID	記述内容	カスタムパラメーター
a	データオブジェクトで型を設定している場合は、ブロックと Stateflow データでは型を設定しません。 ＜例外＞ ・再利用可能な関数内部 ・[Data Type Conversion] ・“fixdt”による型設定 ・double 型、boolean 型の指定	-

【正】
データオブジェクトで型を設定し、ブロックでは型を設定していません。







根拠

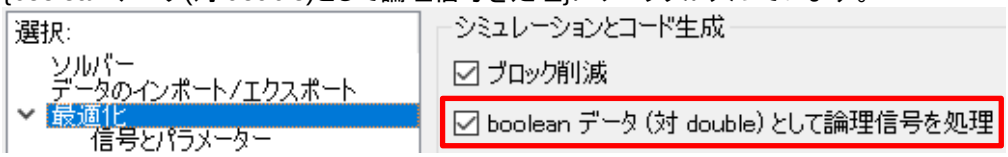
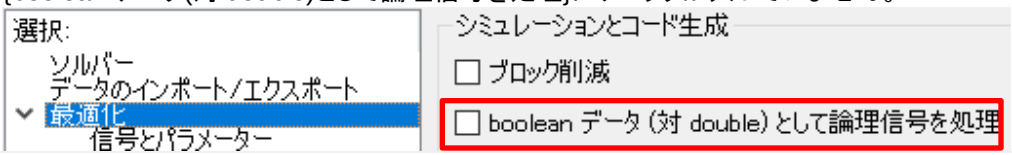
サブ ID	記述内容
a	・ブロックの型をデータオブジェクトでの設定と個別にブロックでの設定が混在している場合、どちらの設定が正しいか判断がつかず可読性が悪化します。 ・ブロックに型設定をすると、信号線の型が変更になったときの保守性が落ちます。 除外対象の説明 ・再利用可能な関数内部 中のブロック構成がすべて同一でも入出力の型が異なれば C ソースが別々にできてしまい、再利用になりません。再利用可能な関数は、少なくとも入出力ブロックの型を固定します。 ・[Data Type Conversion] [Data Type Conversion]は、型を埋め込む為に作られたブロックです。必要に応じてこのブロックを使って型の変更を明示的に指定します。 ・”fixdt”による型設定 固定小数点を指定した場合、ブロック毎に桁が変わるので、細かい指定が必要です。データオブジェクトだけではコントロールできません。 ・double 型、boolean 型の指定 boolean 型は、ブロック内部で直接指定しなければならないブロック種別が存在します。 double 型は通常プラントモデルや RCP で使われます。それらは対象外です。

	組み込みソフトの内部で double 型を使用する場合もありますが、そのような環境は特殊です。double 型の使用箇所を最小とする必要があるので、ブロック毎に細かく指定します。

3. Simulink

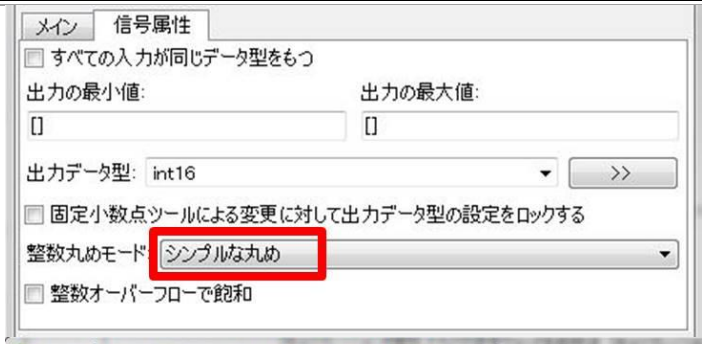
3.1. コンフィギュレーションパラメーター

3.1.1. jc_0011 : 論理信号に対する最適化パラメーター設定

ルール ID : タイトル	jc_0011 : 論理信号に対する最適化パラメーター設定	
ルール		
サブ ID	記述内容	カスタムパラメーター
a	論理信号に対する最適化パラメーターは有効にします。 コンフィギュレーションパラメーターの{最適化} - {シミュレーションとコード生成} - {boolean データ(対 double)として論理信号を処理}にチェックを入れます。	-
	【正】 {boolean データ(対 double)として論理信号を処理}にチェックが入っています。  【誤】 {boolean データ(対 double)として論理信号を処理}にチェックが入っていません。 	
根拠		
サブ ID	記述内容	
a	・boolean データを使用する事で C コード生成時の RAM 容量を削減する事ができます。	

3.1.2. jc_0642 : 整数丸めモードの設定

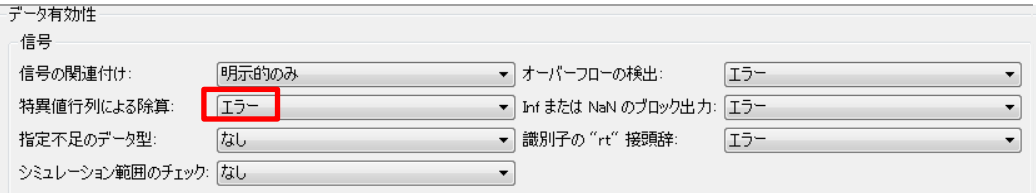
ルール ID : タイトル	jc_0642 : 整数丸めモードの設定	
ルール		
サブ ID	記述内容	カスタムパラメーター
a	{整数丸めモード}にて"シンプルな丸め"を選択する場合は、コンフィギュレーションパラメーターの{ハードウェア実行} - {製品ハードウェア} - {符号付き整数の除算の丸め}を"未定義"以外に設定します。	-
	【正】 {整数丸めモード}にて"シンプルな丸め"を選択し、コンフィギュレーションパラメーターの{ハードウェア実行} - {製品ハードウェア} - {符号付き整数の除算の丸め}を"ゼロ方向"に設定しています。	

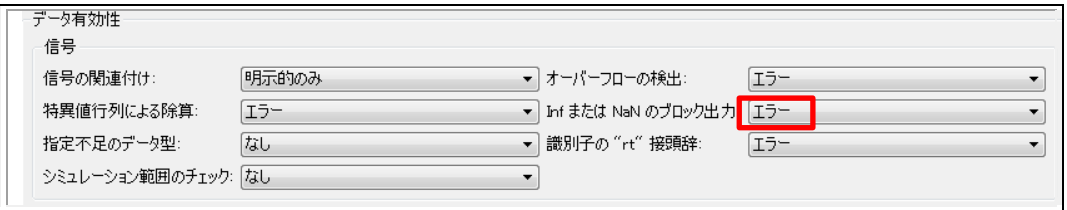
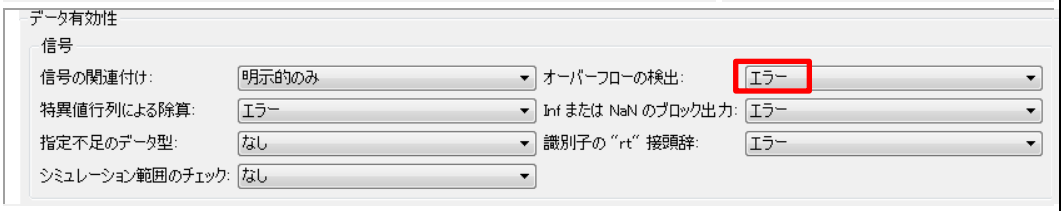


【誤】
{整数丸めモード}にて"シンプルな丸め"を選択し、コンフィギュレーションパラメーターの{ハードウェア実行} - {製品ハードウェア} - {符号付き整数の除算の丸め}を"未定義"に設定しています。

根拠	
サブ ID	**記述内容**
a	・符号付整数の除算結果が意図しない丸めになることを防止できます。

3.1.3. jc_0806 ： 不正な演算結果の検出

ルール ID : タイトル	**jc_0806 ： 不正な演算結果の検出**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	コンフィギュレーションパラメーターの{診断} - {データ有効性} - {信号} - {特異値行列による除算}を”エラー”に設定します。	-
		
b	コンフィギュレーションパラメーターの{診断} - {データ有効性} - {信号} - {Inf または NaN のブロック出力}を”エラー”に設定します。	-

		
c	コンフィギュレーションパラメーターの{診断} - {データ有効性} - {信号} - {オーバーフローの検出}を"エラー"に設定します。	-
		
根拠		
サブ ID	**記述内容**	
abc	・不正な値による演算を検出することができます。	

3.2. ダイアグラムの外観

3.2.1. na_0004：Simulink モデルの表示設定

ルール ID：タイトル	**na_0004：Simulink モデルの表示設定**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	Simulink モデルの表示オプションは、プロジェクトで定めた設定にします。	表示オプション
	例)	

表示オプション	設定
モデル ブラウザー	無効
画面の色	白
ステータス バー	有効
ツール バー	有効
ズーム倍率	標準 (100%)

ブロック表示オプション	設定
背景色	白
前景色	黒
実行コンテキスト インジケーター	無効
ライブラリ リンクの表示	なし
線形化インジケーター	有効
Model/ブロック I/O の不一致	無効
Model ブロックのバージョン	無効
サンプル時間の色分け	無効
並べ替え順序	無効

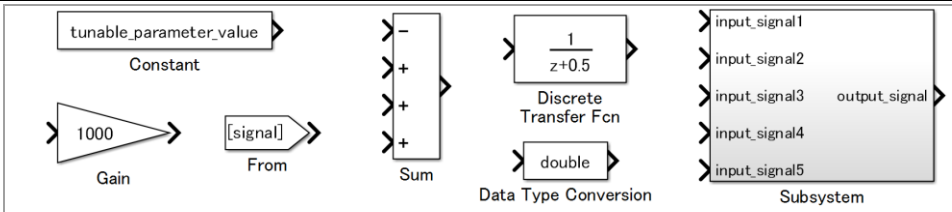
信号の表示オプション	設定
端子のデータ型	無効
信号の次元	無効
ストレージ クラス	無効
テスト ポイント インジケーター	有効
ビューアー インジケーター	有効
非スカラー ラインを太く表示	有効

根拠

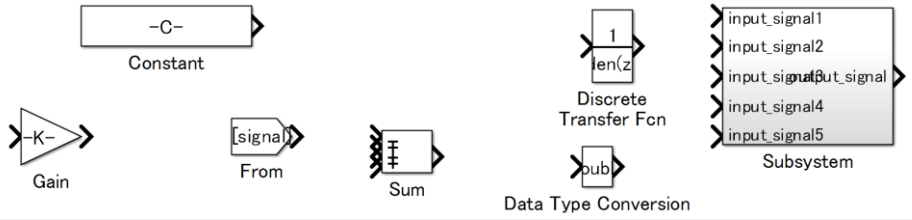
サブ ID	記述内容
a	・モデルの表示設定を統一することで、可読性が向上します。

3.2.2. jm_0002：ブロックのサイズ調整

ルール ID：タイトル	jm_0002：ブロックのサイズ調整	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	・ブロックはアイコン表示が確認できるサイズに調整します。	-
	【正】 ブロックはアイコン表示が確認できる大きさです。	



【誤】
ブロックの大きさが小さいため、アイコン表示が確認できません。

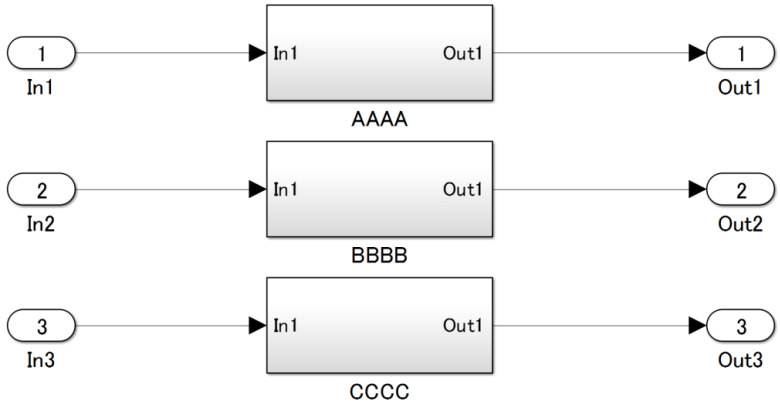


根拠	
サブ ID	**記述内容**
a	・ブロックサイズが小さすぎると、アイコン表示されるテキストやシンボルが見えず可読性が損なわれます。

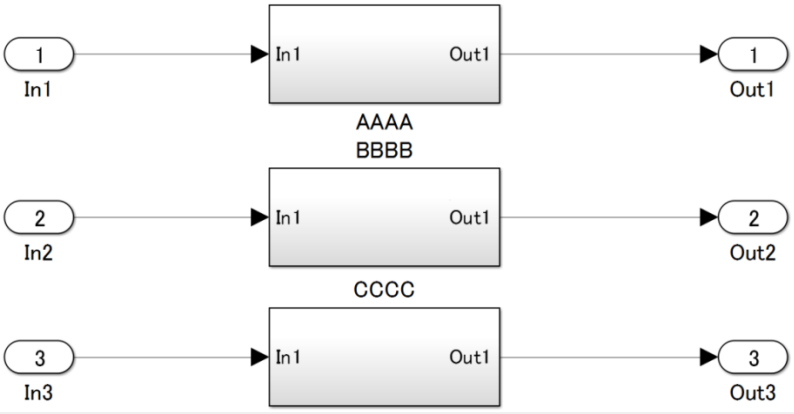
3.2.3. db_0142：ブロック名の位置

ルール ID：タイトル	**db_0142：ブロック名の位置**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	ブロック名はブロックの下側に表示します。	-

【正】



【誤】

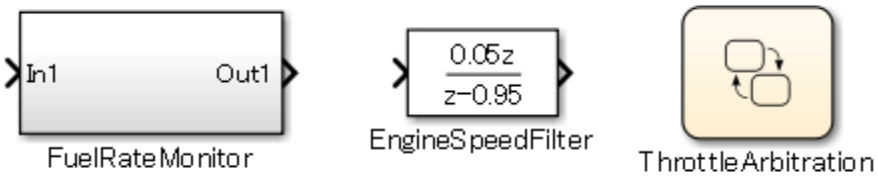


根拠	
サブ ID	**記述内容**
a	・上下にブロックが並ぶ場合、ブロック名の位置を上下で混在させると、どちらのブロック名であるかが判断できなくなる場合があります。

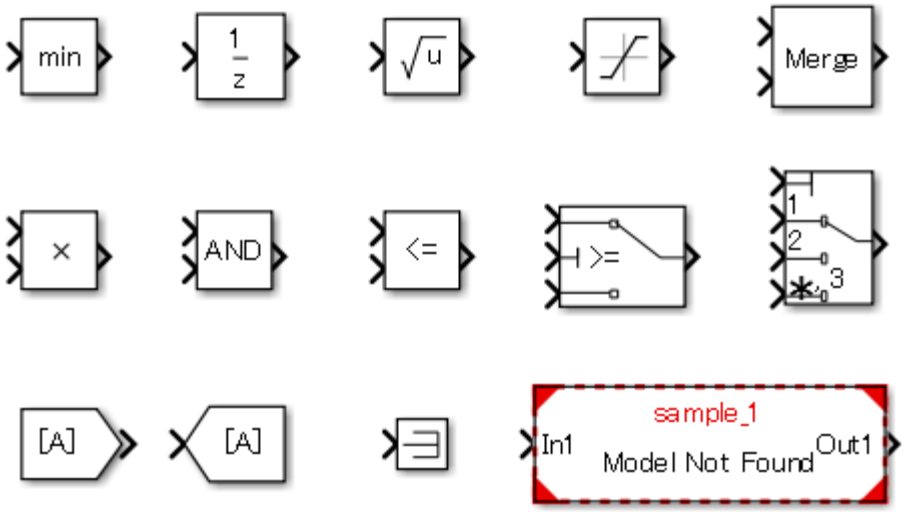
3.2.4. jc_0061 ：ブロック名の表示

ルール ID : タイトル	**jc_0061 ：ブロック名の表示**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	以下の全ての条件にあてはまるブロックはブロック名を非表示にし、いずれかの条件にあてはまらないブロックはブロック名を表示します。 ・ブロックの外観からブロックの種類が解る ・デフォルトブロック名のまま (末尾に数値だけ追加されている場合を含みます)	外観から種類が解るブロック

ブロック名表示例



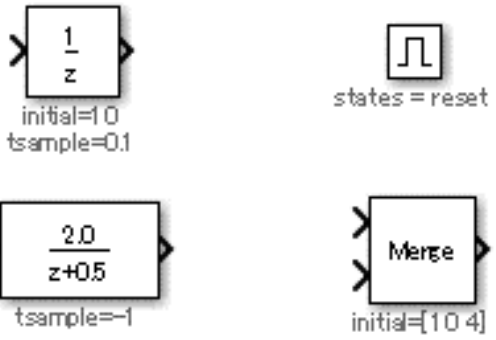
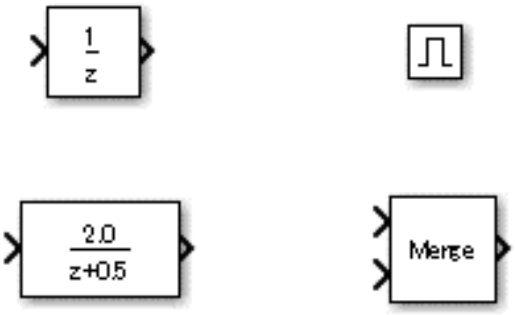
ブロック名非表示例



根拠	
サブ ID	**記述内容**
a	・可読性を向上させます。

3.2.5. db_0140 ：ブロックパラメーターの表示

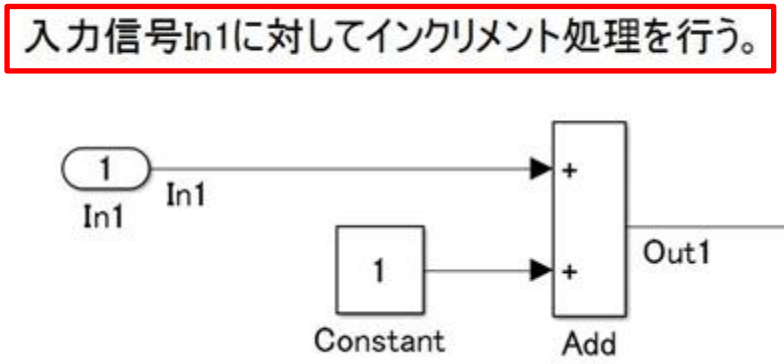
ルール ID : タイトル	**db_0140 ：ブロックパラメーターの表示**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**

a	ブロック注釈には、プロジェクトで定めたブロックパラメーターを表示します。	ブロックパラメーター
	【正】  【誤】  	

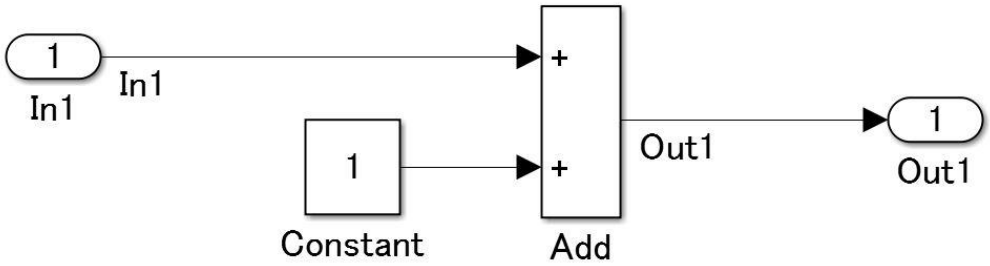
根拠	
サブ ID	**記述内容**
a	・ブロックパラメーターを外観で確認できることで、可読性が向上します。

3.2.6. jc_0603：モデルの説明

ルール ID：タイトル	jc_0603：モデルの説明	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	モデルのレイヤには、そのレイヤの説明を記載します。 説明を記載するレイヤはプロジェクトで定めます。 (機能やレイヤの種類で定義します。)	説明記載オブジェクト (ブロックの種類等) 説明記載対象レイヤ
	【正】 モデルの説明が記載されています。	



【誤】
モデルの説明が記載されていません。



b	レイヤの説明はモデル全体で共通のフォーマットを使用します。	モデル説明のフォーマット

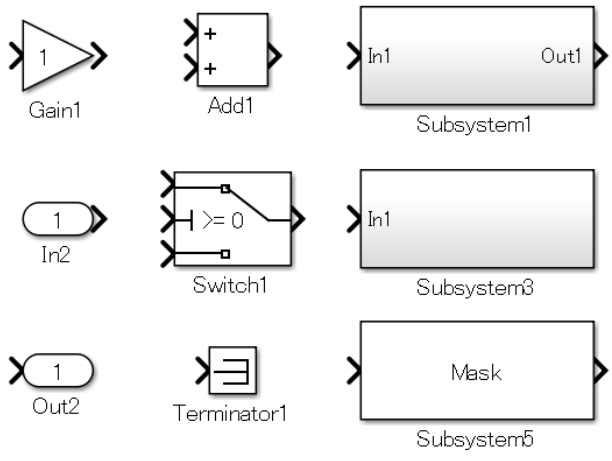
根拠

サブ ID	記述内容
a	・説明が無いと制御仕様の可読性が低下し、使用性・保守性・移植性が低下します。
b	・説明のフォーマットが統一されていないと可読性が低下します。

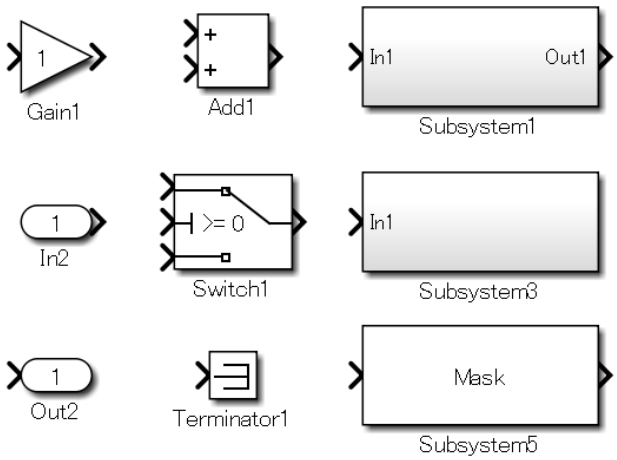
3.2.7. jc_0604 : ブロックの陰影

ルール ID : タイトル	jc_0604 : ブロックの陰影	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	ブロックには陰影を付けません。	-

【正】
ブロックに陰影が付いていません。

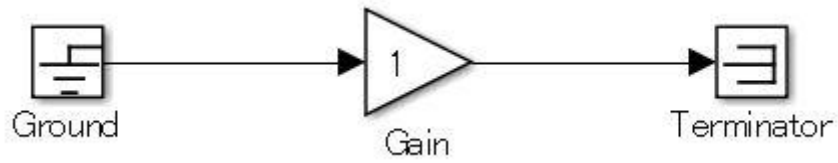
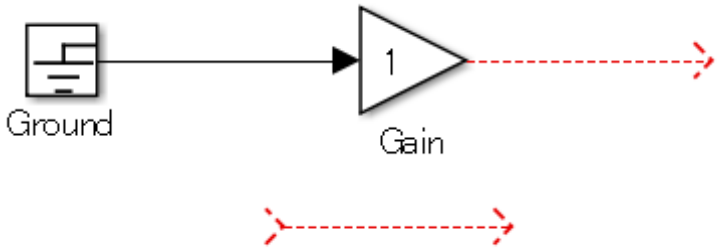


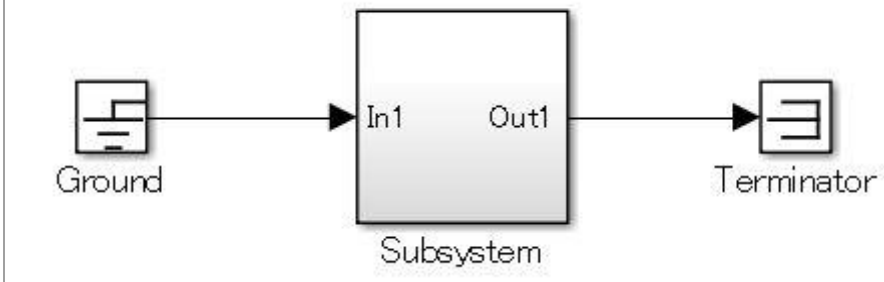
【誤】
ブロックに陰影が付いています。



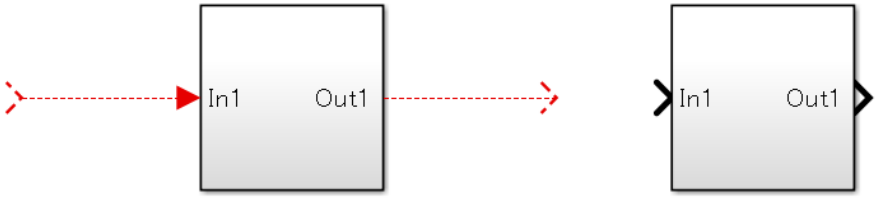
根拠	
サブ ID	**記述内容**
a	・ポートが陰影に隠れてポートの有無が解り辛いため可読性が低下します。

3.2.8. db_0081：未接続の信号 / ブロック

ルール ID : タイトル	**db_0081：未接続の信号 / ブロック**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	未接続の信号線を含みません。	-
	【正】  【誤】  	
b	未接続のサブシステム、基本ブロックを含みません。	-
	【正】	

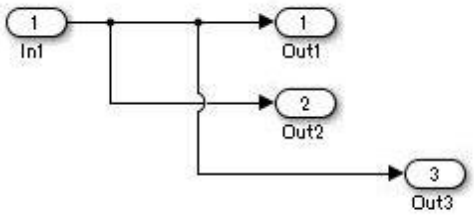


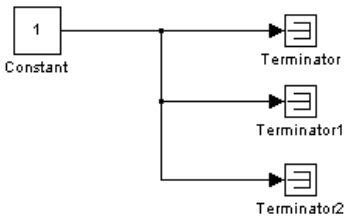
【誤】



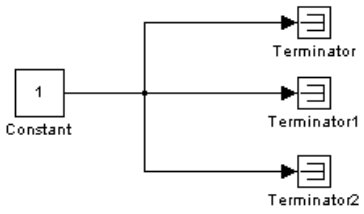
根拠	
サブ ID	**記述内容**
ab	・結線忘れによるシミュレーション結果の誤判定やコードが生成されないなどの弊害が懸念されます。

3.2.9. db_0032：信号線の結線

ルール ID：タイトル	**db_0032：信号線の結線**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a1	信号線は、交差させません。	-
a2	{ライン交差のスタイル}を”飛び越し”に設定します。	-
	【正】 交差した縦線が横線をよけるようにします。(R2014a 以降で設定できます) 	
b	信号線は、他の信号線と重ねません。	-
c	信号線は、ブロックを横切りません。	-
d	信号線は、１つの分岐点で２つ以上の分岐をしません。	-
	【正】	



【誤】



e	信号線は、垂直方向、もしくは水平方向に引きます。	-

根拠

サブ ID	記述内容
a1	・信号線が交差すると、ブロック間の接続関係が解りにくくなります。
a2	・R2014a 以降で、交差と分岐の違いを明確にすることができます。
b	・信号線が重なると、ブロック間の接続関係が解りにくくなります。
c	・信号線とブロックが交差すると、ブロック間の接続関係が解りにくくなります。
d	・ブロック間の接続関係が解りにくくなります。
e	・線の引き方を統一することで可読性が向上します。

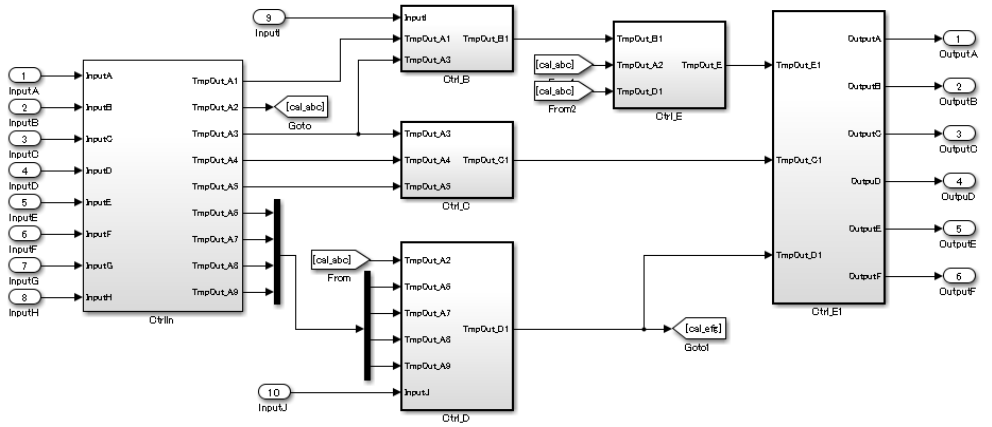
3.2.10. db_0141：Simulink モデルの信号フロー

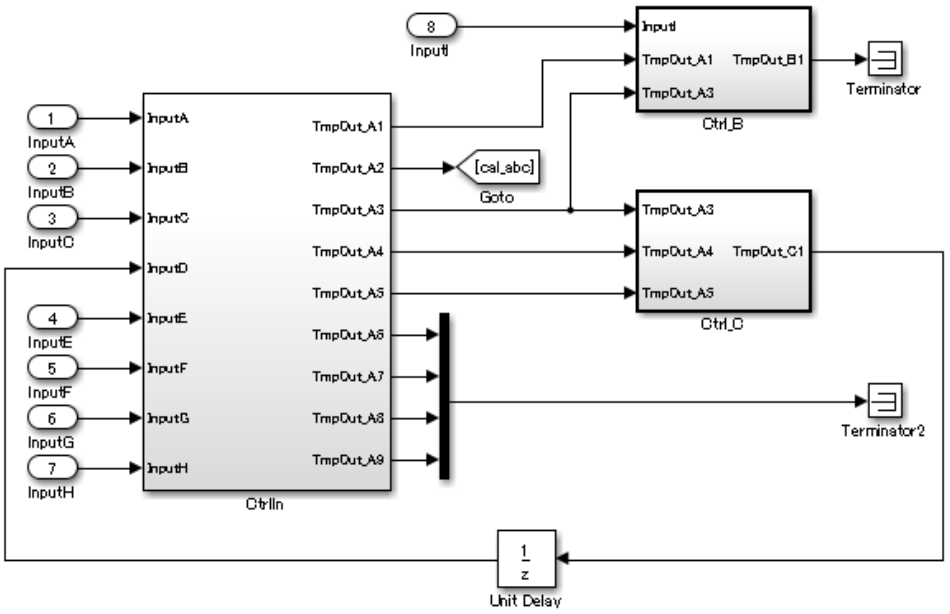
ルール ID：タイトル	db_0141：Simulink モデルの信号フロー

ルール

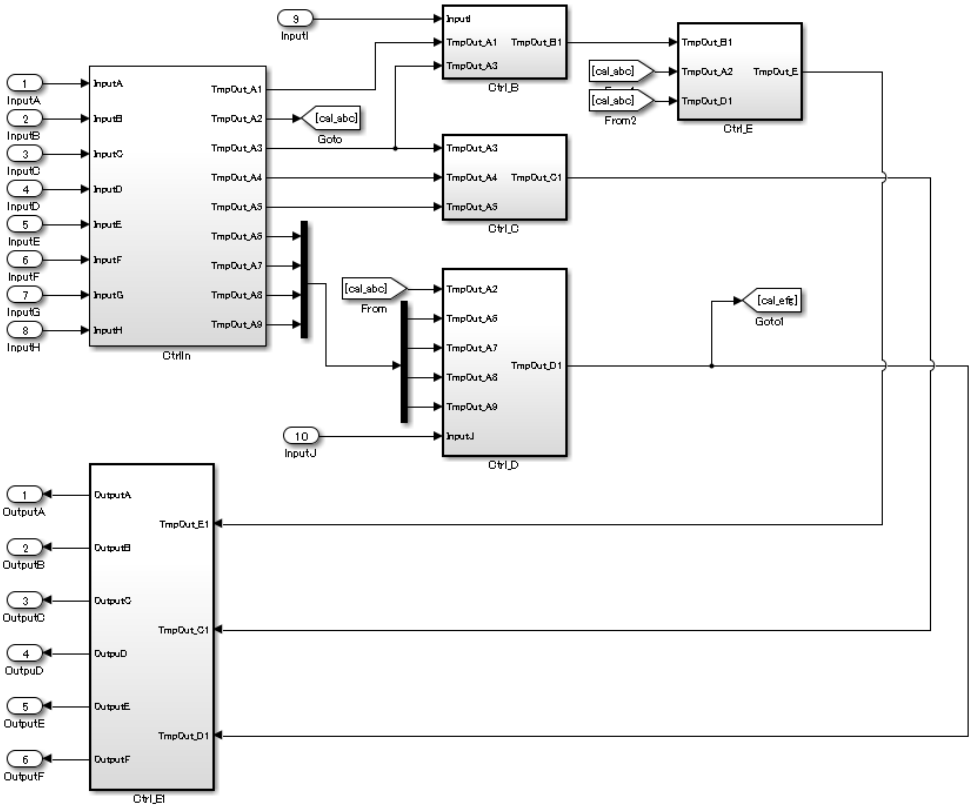
サブ ID	記述内容	カスタムパラメーター
a	信号は、左から右に流します。 ＜例外＞ フィードバックループは右から左に流します。	-

【正】



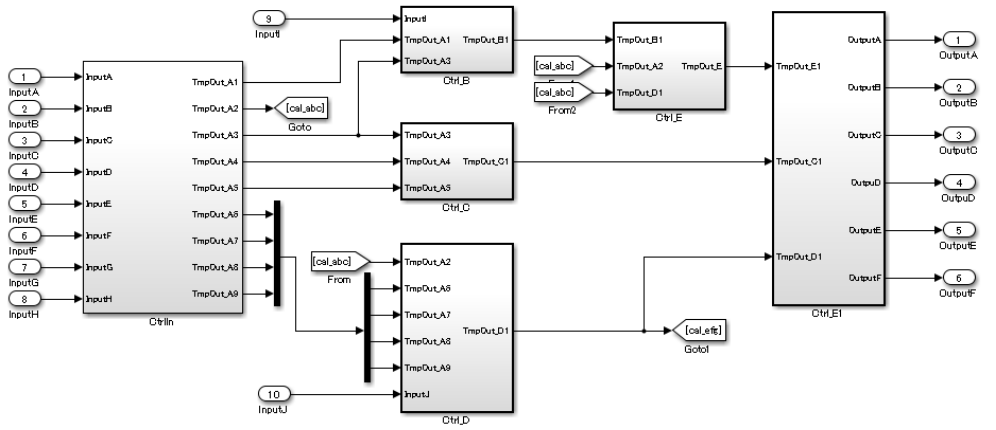


【誤】

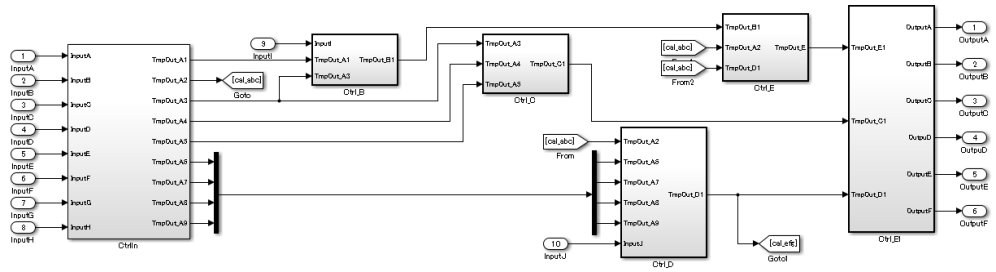


b	並列処理のブロックまたはサブシステムは上下に配置します。	-

【正】

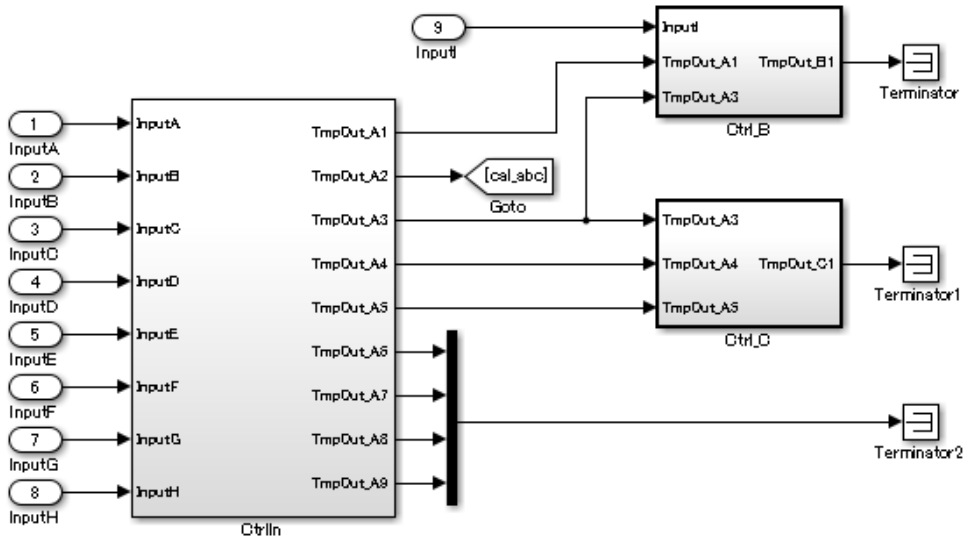


【誤】

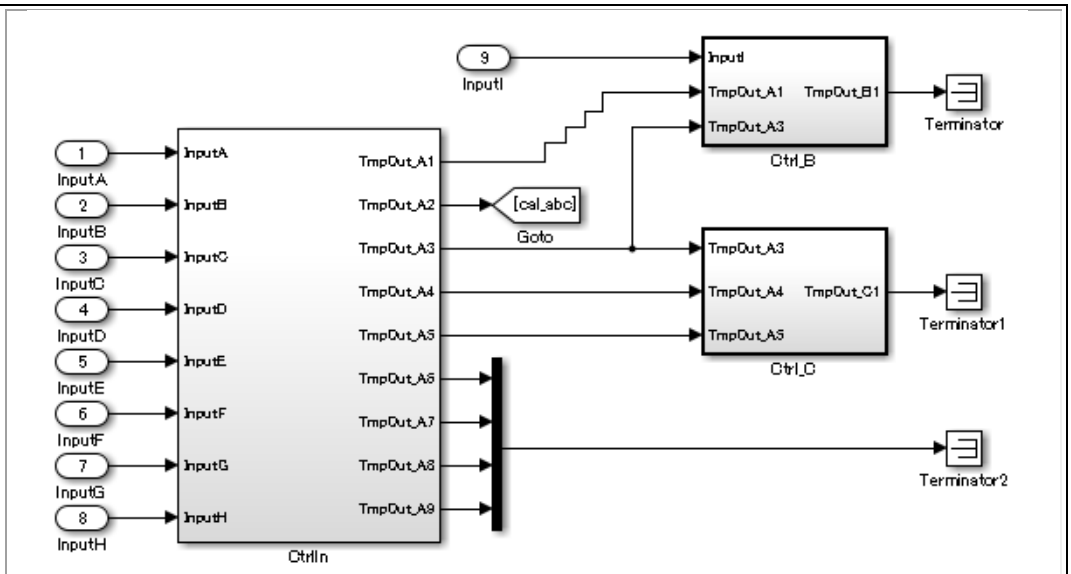


c	信号線は、不必要に何回も折り曲げません。	-

【正】



【誤】

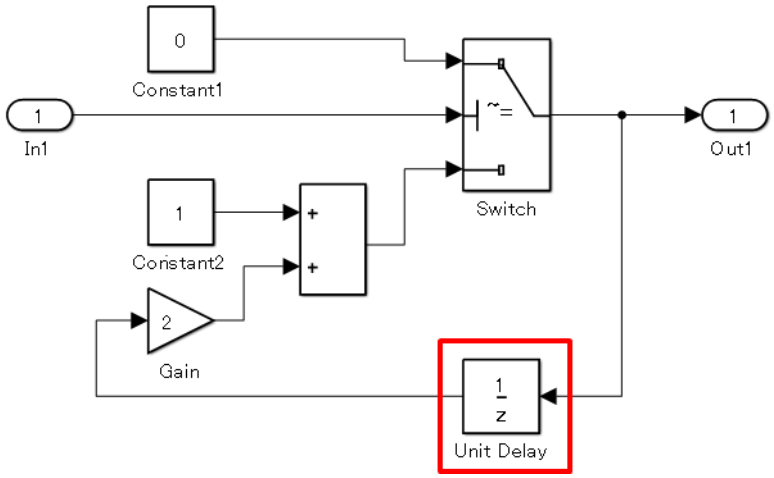


根拠	
サブ ID	**記述内容**
abc	・ 定められた信号線フロー記述規則に従うことで、可読性が向上します。

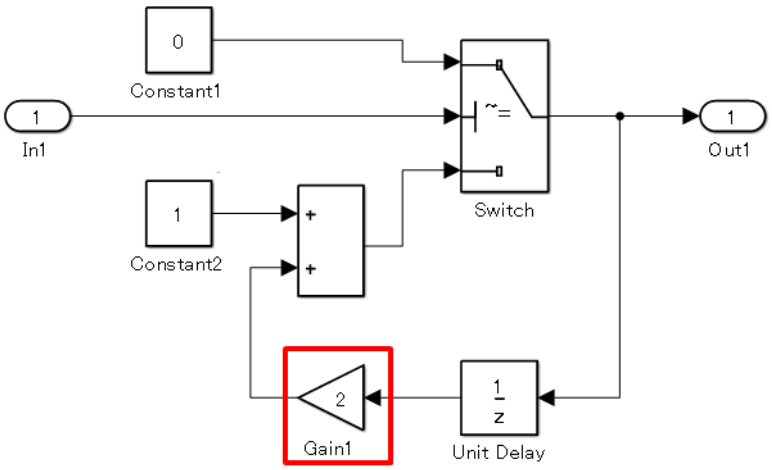
3.2.11. jc_0110：ブロックの向き

ルール ID：タイトル	jc_0110：ブロックの向き	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	ブロックは出力が右になるように配置します。 ＜例外＞ 遅延ブロックをフィードバックループに使用する場合は出力が左になるように配置できます。	-

【正】
出力が右になるようにブロックを配置しています。
フィードバックループ上なので、出力を左にして遅延ブロックを配置できます。



【誤】
出力が右になるようにブロックを配置していません。

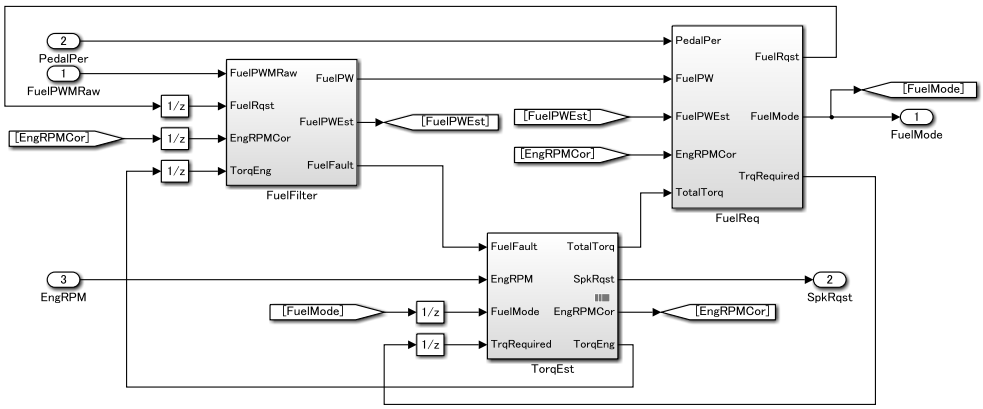


根拠	
サブ ID	**記述内容**
a	信号の向きが統一されていないと信号の流れが把握しにくくなります。

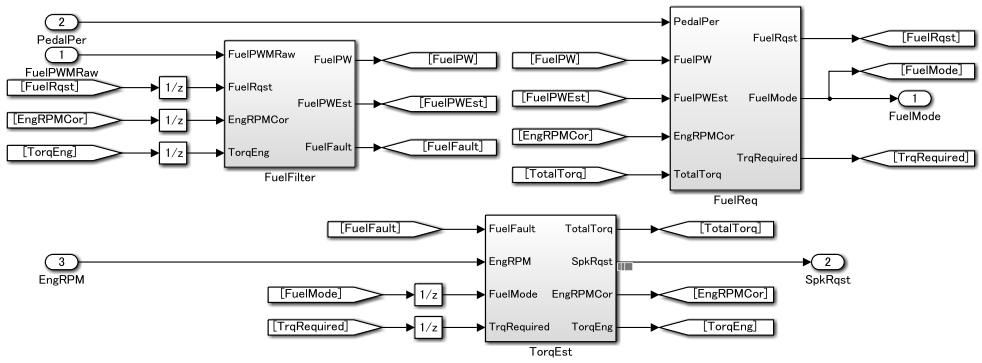
3.2.12. jc_0171：構造サブシステム間の接続関係の明確化

ルール ID：タイトル	**jc_0171：構造サブシステム間の接続関係の明確化**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	・構造サブシステムから構造サブシステムへ信号線を接続するときは、少なくとも 1 本は信号線で接続します。 ・2 つの構造サブシステム A、B で双方向の信号接続が存在する場合は、それぞれの方向につき、少なくとも 1 本は信号線で接続します。 ＜例外＞ 構造サブシステムから構造サブシステムへ[BusCreator]、[Merge]を経由して信号を接続する場合、これらのブロックには[Goto]、[From]のみで接続しても構いません。	-

【正】

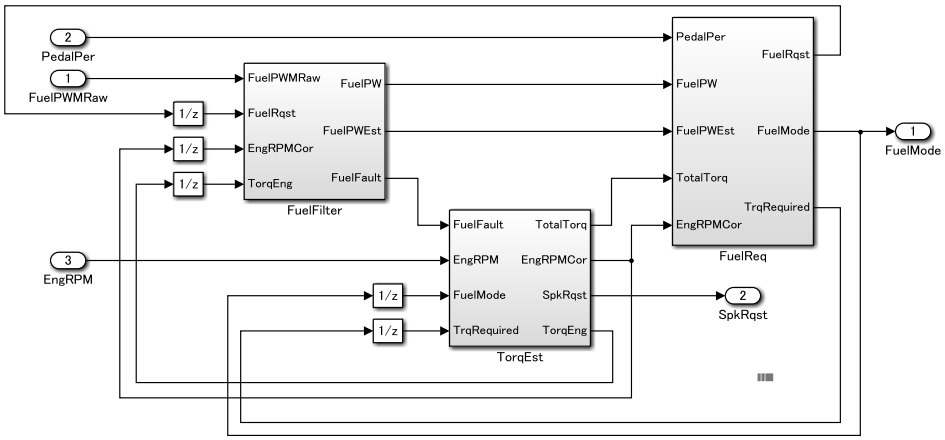


【誤】



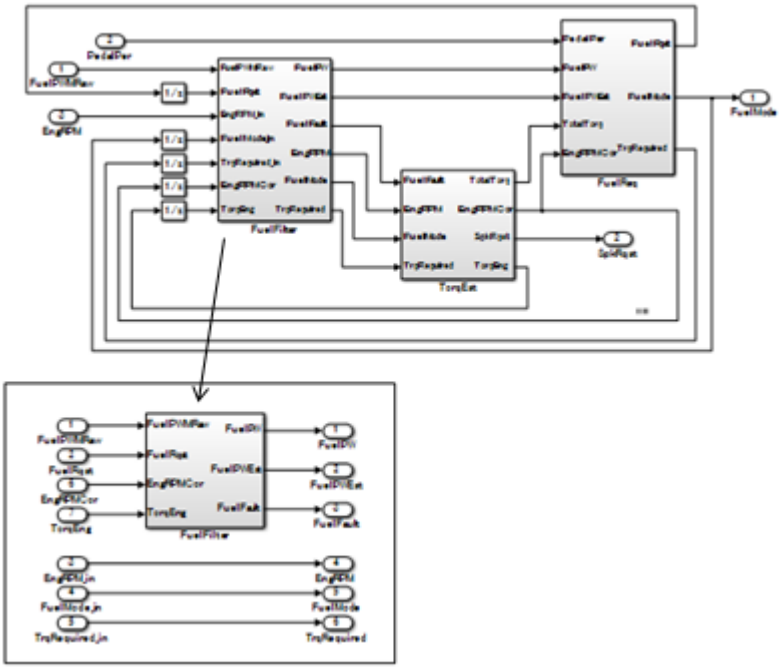
b	構造サブシステム内部で使用されない信号を構造サブシステムに入力し、その信号をそのまま別の構造サブシステムや基本ブロックへ向けて出力しません。	-

【正】



【誤】

信号線の交差を避けるために、サブシステムで使用しない信号を接続

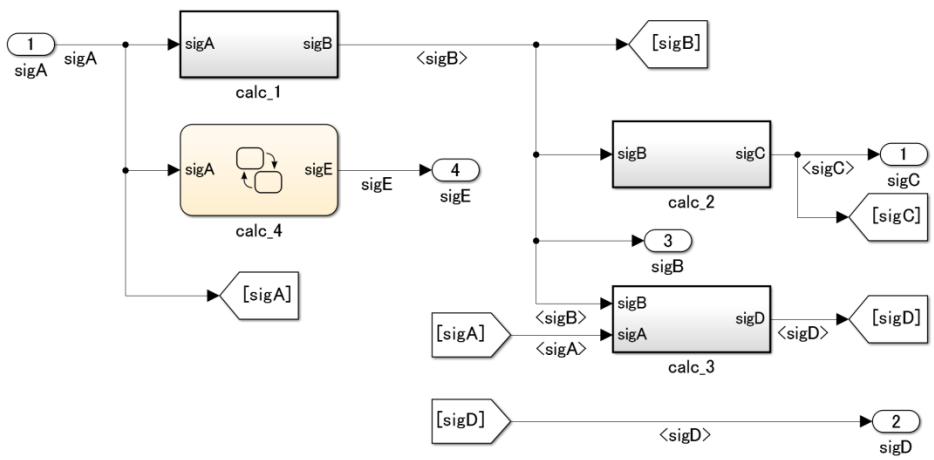


根拠	
サブ ID	**記述内容**
a	・構造サブシステム間の接続関係や実行順序が見やすくなります。
b	・不要な接続を減らすことで、接続関係が見やすくなります。 ・ルールを逸脱すると、サブシステムで未使用の入出力信号が使用されているように誤解を招きます。

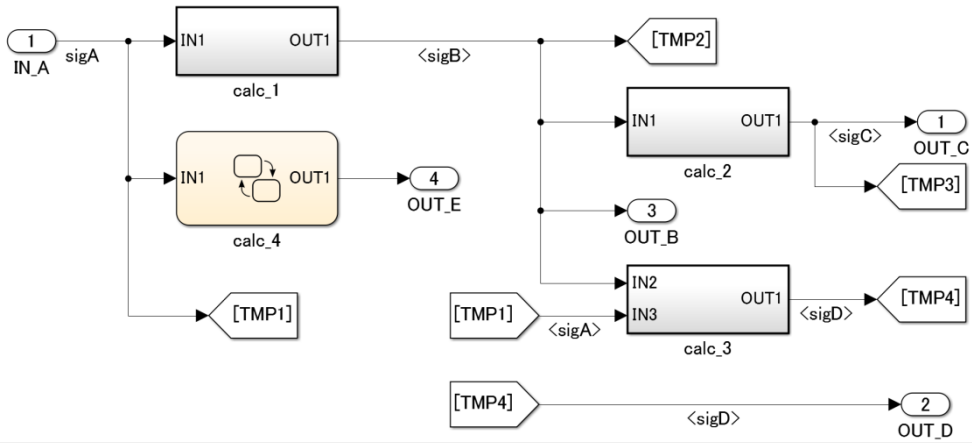
3.2.13. jc_0602：モデル要素の名前の一致

ルール ID：タイトル	**jc_0602：モデル要素の名前の一致**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	信号線で直接接続された以下の名前は一致させます。 ・[Inport]のブロック名 ・[Outport]のブロック名 ・構造サブシステムの入力ポートラベル名 ・構造サブシステムの出力ポートラベル名 ・[From]のタグ名 ・[Goto]のタグ名 ・信号線の信号名 ＜例外 1＞ 以下のサブシステムと接続する信号線の信号名には例外的にサブシステムのポートラベル名と異なる名前を設定できます。 ・ライブラリにリンクされているサブシステム ・再利用可能なサブシステム ＜例外 2＞ [Inport]、[Outport]、その他のブロックの組み合わせが同じブロック名の場合は、[Inport]と [Outport]にサフィックスまたはプレフィックスを使用します([Inport]、[Outport]のブロック名と信号名に異なる名前を設定します)。端子に使用するサフィックス、プレフィックスは任意ですが、一貫したサフィックス、プレフィックスを付与します([Inport]では”in”、[Outport]では”out”等)。	-

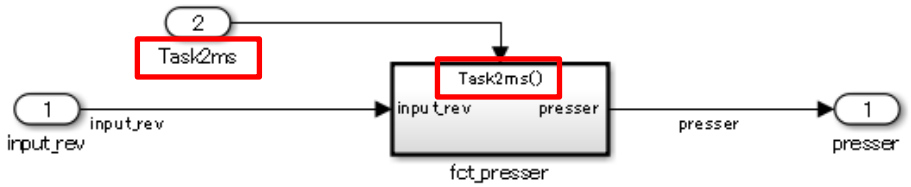
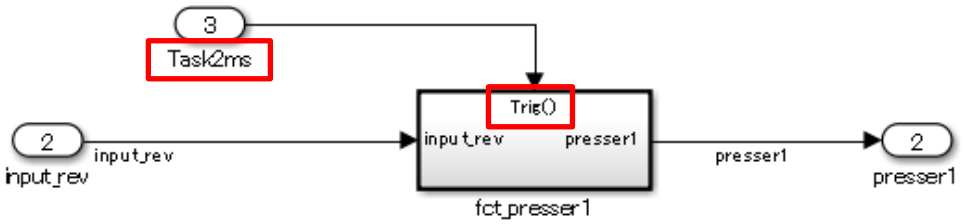
【正】
信号線で直接接続されたモデル要素の名前が一致しています。

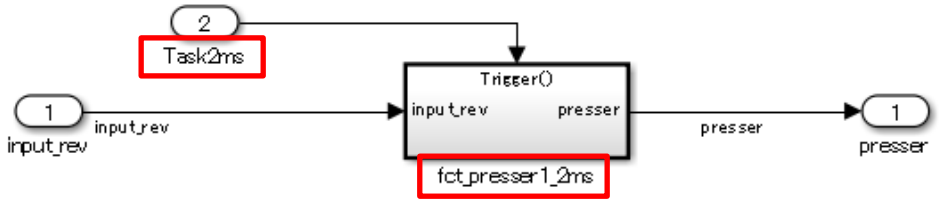


【誤】

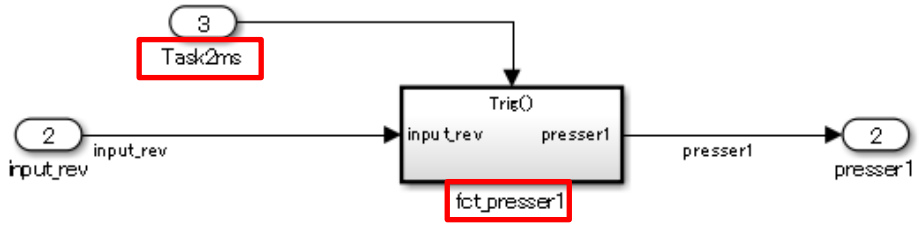
	信号線で直接接続されたモデル要素の名前が一致していません。 
根拠	
サブ ID	**記述内容**
a	・信号線の接続ミスが防止できます。 ・逸脱した場合、可読性が損なわれます。 ・逸脱した場合、モデルとコードの整合が取りづらくなります。

3.2.14. jc_0281 : トリガー信号の名前

ルール ID : タイトル	**jc_0281 : トリガー信号の名前**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a1	トリガー信号の接続元のブロック名と、接続先の条件入力ブロックのブロック名は、同じ名称が含まれるようにします。	-
	【正】  【誤】 	
a2	トリガー信号の接続元のブロック名と、接続先の条件付きサブシステムのサブシステム名は、同じ名称が含まれるようにします。	-
	【正】	

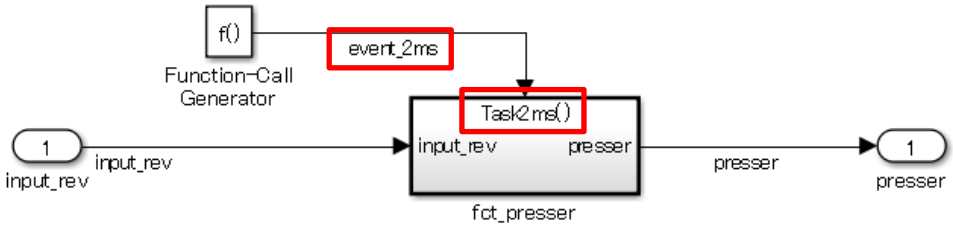


【誤】

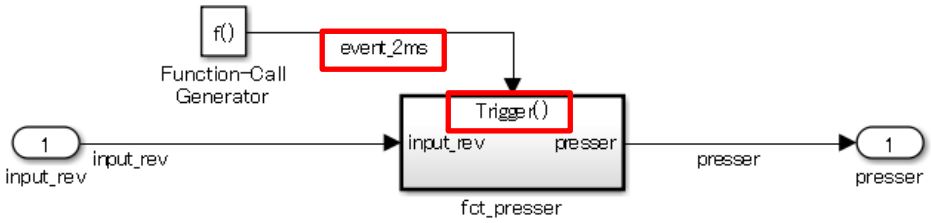


a3	トリガー信号の信号名と、接続先の条件入力ブロックのブロック名は、同じ名称が含まれるようにします。	-

【正】

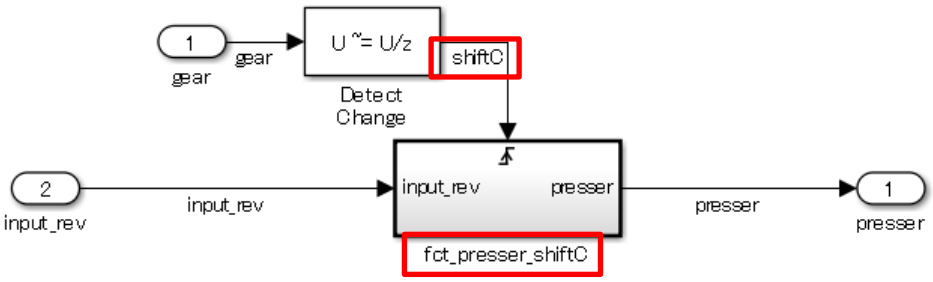


【誤】

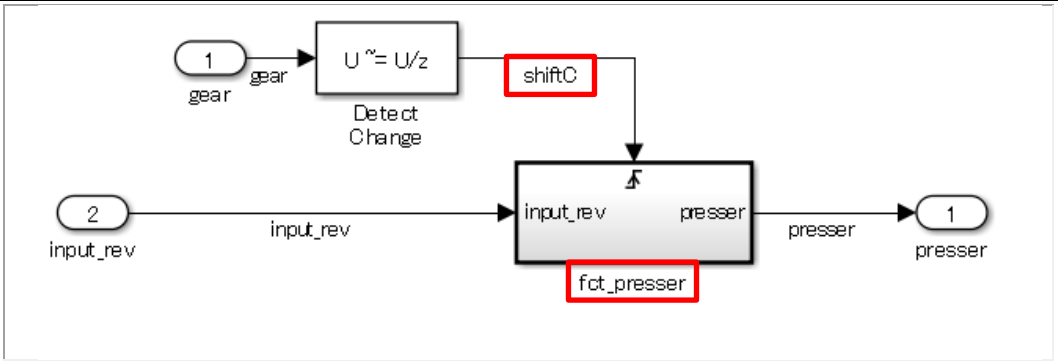
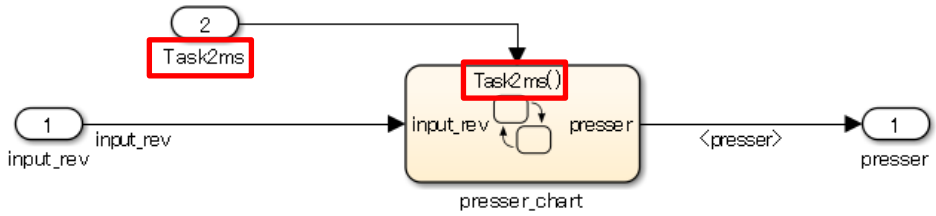


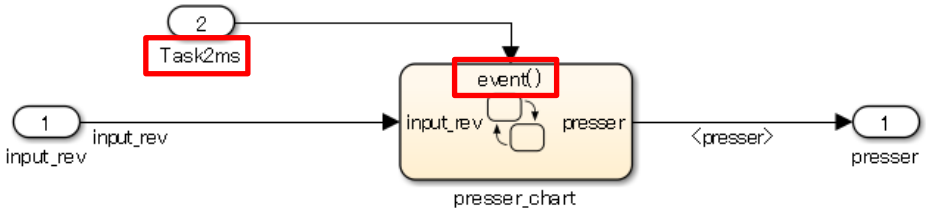
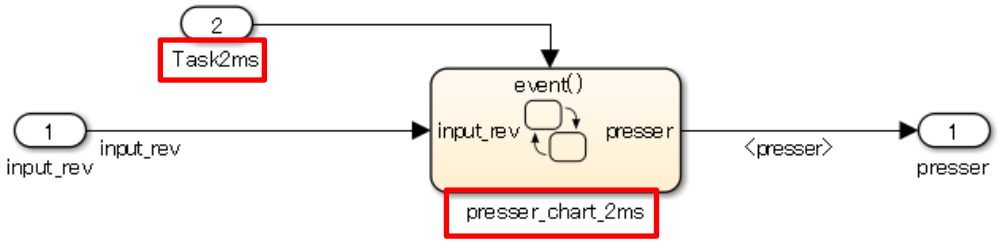
a4	トリガー信号の信号名と、接続先の条件付きサブシステムのサブシステム名は、同じ名称が含まれるようにします。	-

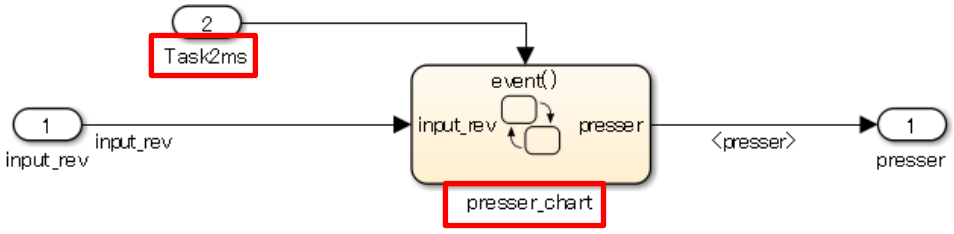
【正】



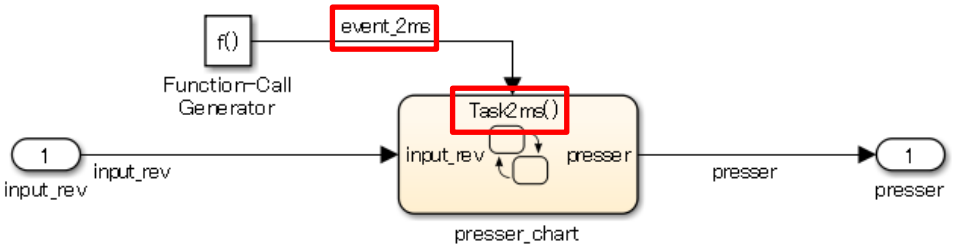
【誤】

<table>
<tr><td></td><td> 
</td><td></td></tr>
<tr><td>b1</td><td>トリガー信号の接続元のブロック名と、接続先の Stateflow ブロックのイベント名は、同じ名称が含まれるようにします。</td><td>-</td></tr>
<tr><td></td><td>【正】 

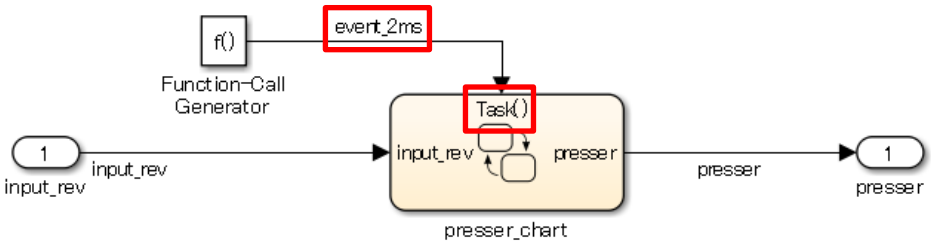
【誤】 
</td><td></td></tr>
<tr><td>b2</td><td>トリガー信号の接続元のブロック名と、接続先の[Chart]の名前は、同じ名称が含まれるようにします。</td><td>-</td></tr>
<tr><td></td><td>【正】 

【誤】 
</td><td></td></tr>
<tr><td>b3</td><td>トリガー信号の信号名と、接続先の Stateflow ブロックのイベント名は、同じ名称が含まれるようにします。</td><td>-</td></tr>
</table>

【正】

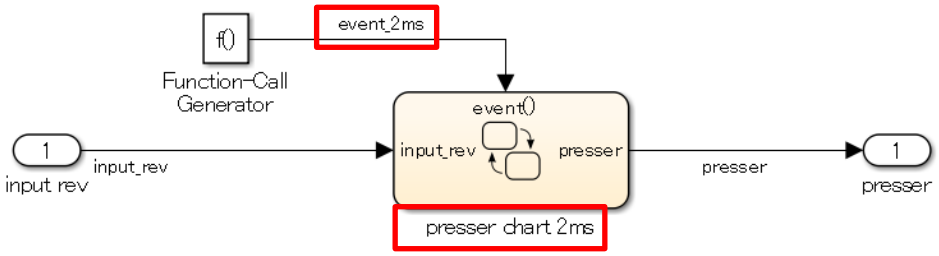


【誤】

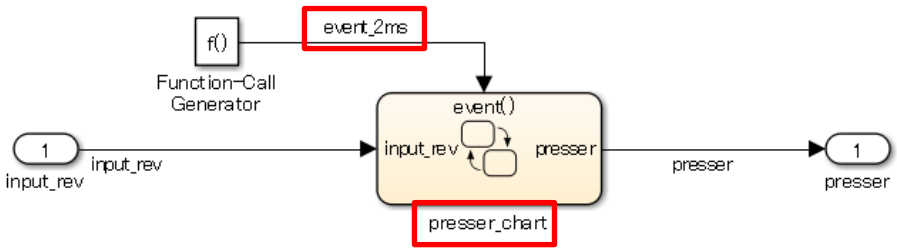


b4	トリガー信号の信号名と、接続先の[Chart]の名前は、同じ名称が含まれるようにします。	-

【正】



【誤】



根拠	
サブ ID	**記述内容**
a1a2a3 a4b1b2 b3b4	・トリガー信号の接続元と接続先の関連が解りやすくなり、接続ミスを低減できます。 (自動チェッカーで検出できるようになります。)

3.2.15. db_0143：各モデル階層で使用できるブロックタイプ

ルール ID：タイトル	**db_0143：各モデル階層で使用できるブロックタイプ**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	モデルの各階層は、レイヤの種類に応じて、プロジェクトで定めたブロックタイプのみを使用します。	レイヤの種類 ブロックタイプ

レイヤの種類については 9.2 コントローラモデルの階層構造を参照してください。レイヤの種類を細かく定義すると使用可能なブロックがより限定されます。

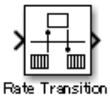
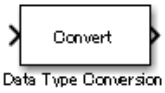
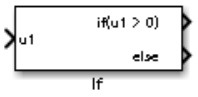
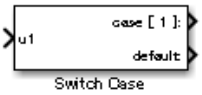
下記ブロックにおいてはツールの制約があります。
・R2011a 以前は、[Enable]をモデルのルート レベルで使用できません。
・Action 端子はモデルのルート レベルでは使用できません。

以下のような使い分けをします。
・データフローレイヤ
基本ブロックのみを使用

・データフローレイヤ以外のレイヤ
構造サブシステムと全レイヤで使用可能なブロックを使用

全レイヤで使用可能なブロック

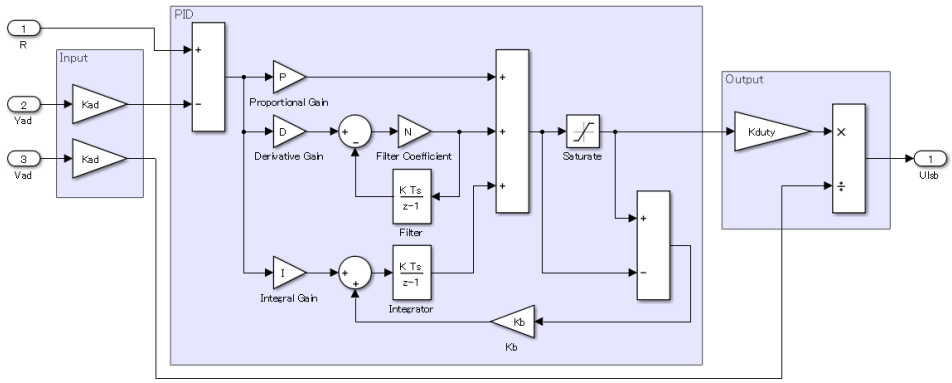
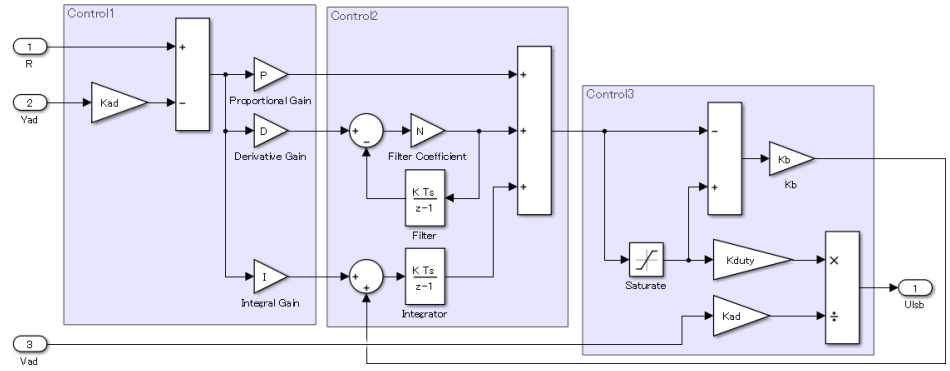
ブロックタイプ	ブロックアイコン例
Inport	
Outport	
Mux	
Demux	
Bus Selector	
Bus Creator	
Selector	
Ground	
Terminator	

From	
Goto	
Merge	
Unit Delay	
Rate Transition	
Data Type Conversion	
Data Store Memory	
If	
SwitchCase	
Function-Call Generator	
Function-Call Split	

根拠	
サブ ID	**記述内容**
a	・同じ階層にサブシステムと基本ブロックが混在すると可読性が低下します。

3.2.16. db_0144 :サブシステムの使用方法

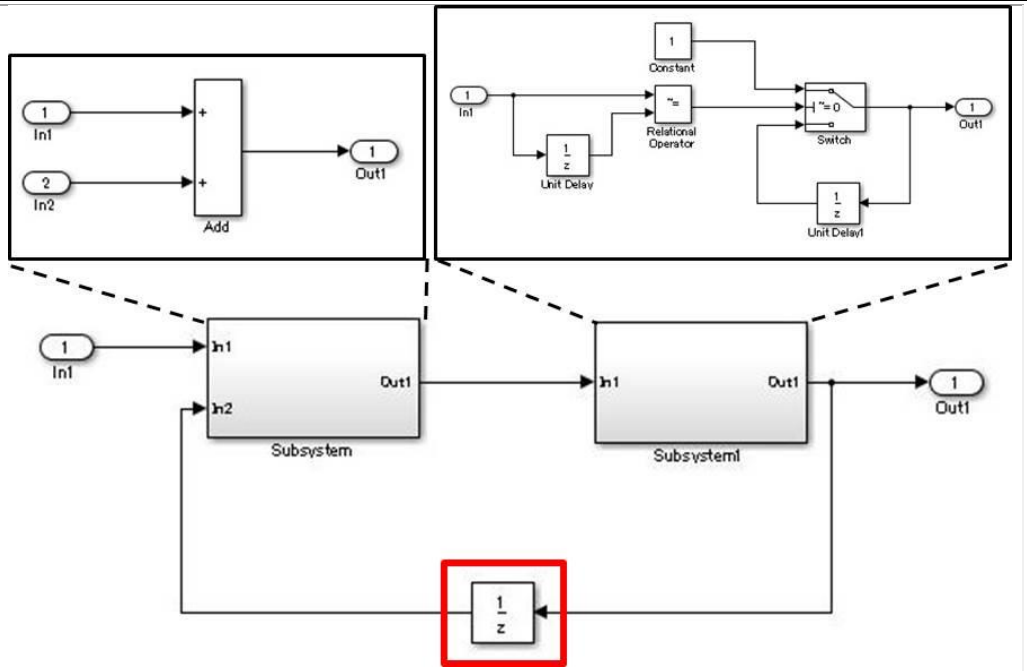
ルール ID : タイトル	**db_0144：サブシステムの使用方法**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**

a	ブロック線図の一部分をサブシステム化する際は、アルゴリズムを機能的に分解したグループをサブシステムとします。	-
	【正】 機能単位でサブシステム化しています。  【誤】 機能単位でサブシステム化していません。  	
b	処理順序やコード生成への影響を考慮せず、単純にサブシステムを作成する必要がある場合は、バーチャル サブシステムを使用します。	-

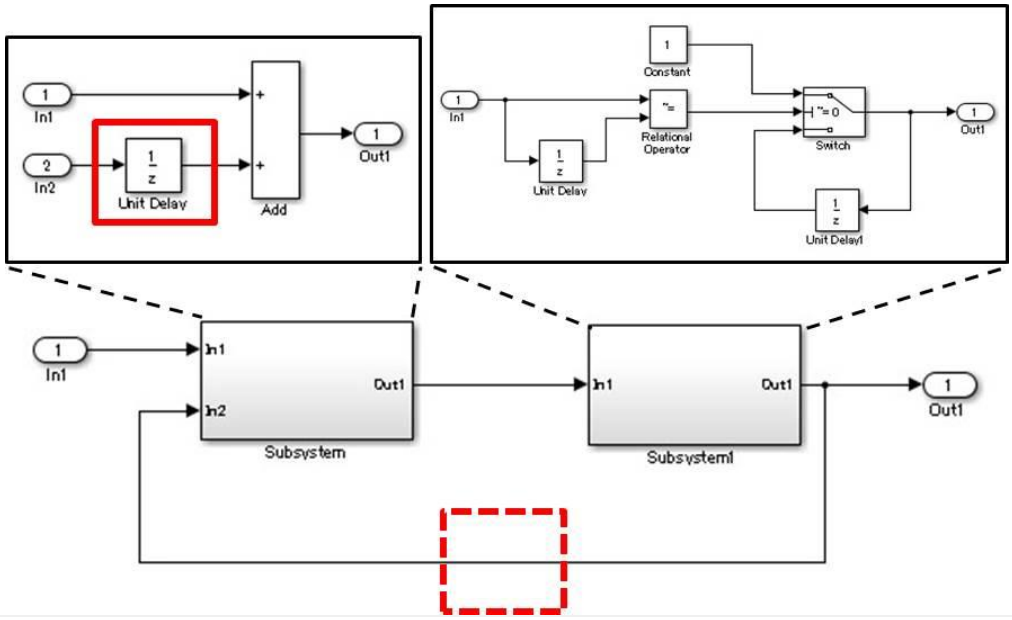
根拠	
サブ ID	**記述内容**
a	・描画スペースの都合でサブシステム化すると、その部分が機能としてまとまっているように誤解を与えます。また、サブシステムを再利用しにくくなります。
b	・Atomic サブシステムを使うと1つの処理としてみなされるため、処理順序やコード最適化に影響します。 その意図がないのに Atomic サブシステムを使用すると、その意図があるような誤解が生じます。

3.2.17. jc_0653 : フィードバックループにおける遅延ブロックの配置方法

ルール ID : タイトル	**jc_0653 : フィードバックループにおける遅延ブロックの配置方法**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	サブシステムをまたぐフィードバックループ において、遅延ブロックを使用する場合、フィードバックループを記載している階層に配置します。	-
	【正】 遅延ブロックがフィードバックループを記載している階層に配置されています。	



【誤】
　遅延ブロックがフィードバックループを記載している階層より下のサブシステム内部に配置されています。



根拠	
サブ ID	**記述内容**
a	・遅延ブロックを 2 重に配置することを防ぎます。 ・流用範囲を明確にすることで、再利用性が向上します。 ・遅延ブロックを含むサブシステムを単体でテストしたい場合に、過去値を直接入力できないためテストしにくくなります。

3.3. 信号

3.3.1. na_0010 : ベクトル信号 / バス信号の使用方法

ルール ID : タイトル	na_0010 : ベクトル信号 / バス信号の使用方法
ルール	

サブ ID	記述内容	カスタムパラメーター
a	ベクトルの生成、分解には[Mux]、[Demux]を使用します。	-
b	[Mux]の入力はスカラー、ベクトルにします。	-
c	バスの生成、分解には[BusCreator]、[BusSelector]を使用します。	-
d	バスはバス対応ブロックに接続します。	-

根拠

サブ ID	記述内容
abcd	・ベクトル/バスの混在問題を回避します。 (詳細はヘルプの「バスと Mux の混在の防止」を参照)

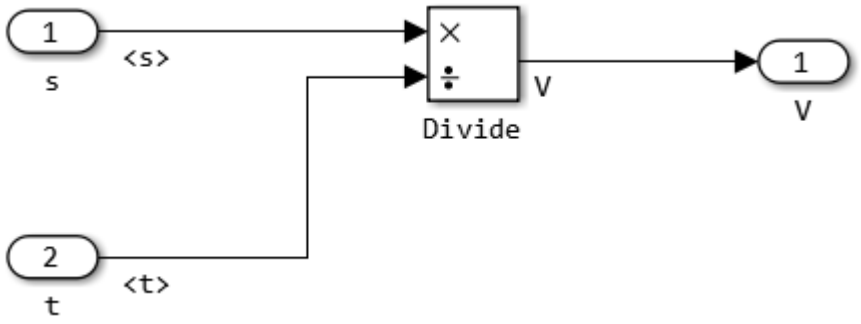
3.3.2. jc_0008：信号名の定義

ルール ID：タイトル	jc_0008：信号名の定義

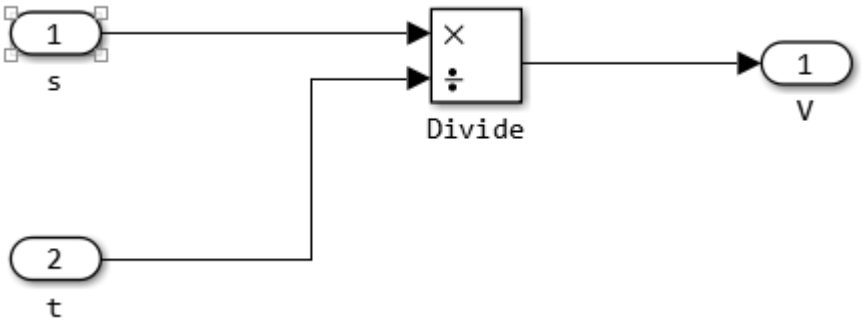
ルール

サブ ID	記述内容	カスタムパラメーター
a	重要なブロックから出力される信号線には信号名を定義します。 信号名を定義した場合、必ずラベルを表示します。 信号名は信号の発生箇所に一度だけ入力します。 重要なブロック: ブロックの種類で決まるものではなく、システムの入力や、意味のある結果を出力するブロックです。	重要なブロックの定義

【正】



【誤】



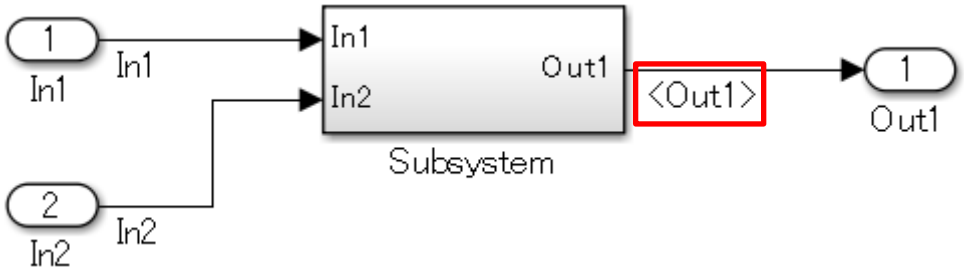
根拠

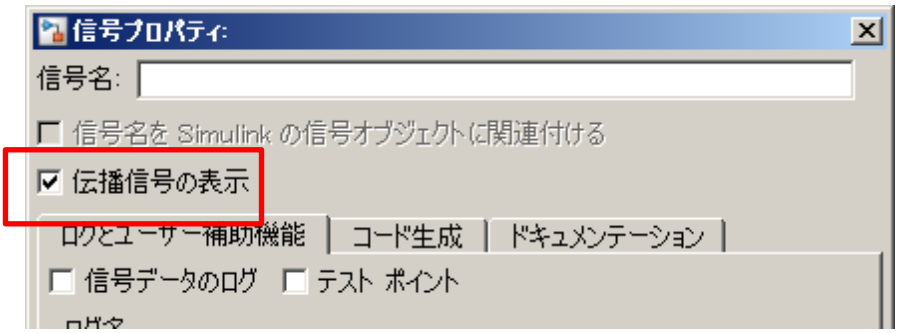
サブ ID	記述内容
a	・意味を持つ結果を出力する重要なブロックの出力に対して、信号名の定義とラベル表示を設定することで、モデルの制御的な可読性を向上します。

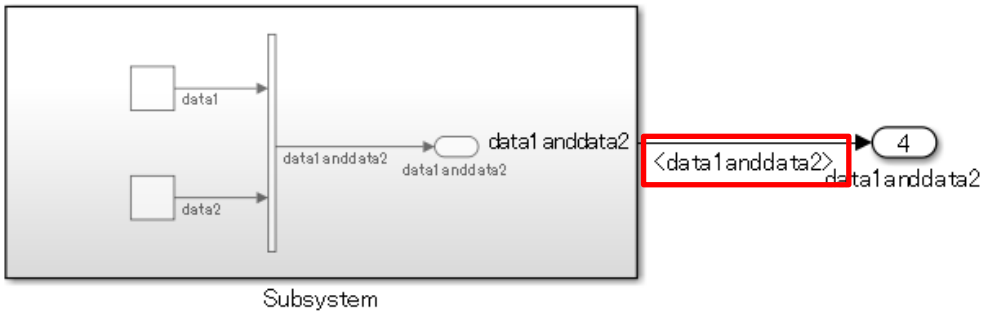
3.3.3. jc_0009 : 信号名の伝播表示

ルール ID : タイトル	jc_0009 : 信号名の伝播表示	
ルール		
サブ ID	記述内容	カスタムパラメーター
a	階層を跨いだ信号元に信号名が定義されている場合、{伝播信号の表示}にチェックを入れて伝播された信号を表示します。 ただし、以下のいずれかを満たす場合は、{伝播信号の表示}はチェックを外します。 ・ライブラリが設定されたサブシステムの内部 ・再利用可能な関数が設定されたサブシステムの内部 ・[Bus Creator]の出力信号に信号名が設定されていない	-

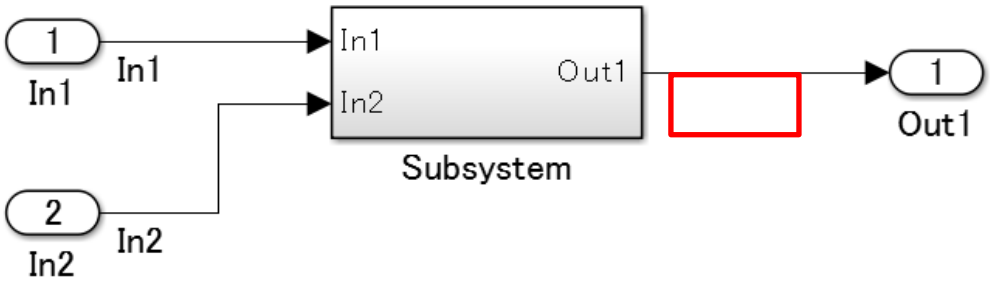
【正】
{伝播信号の表示}にチェックを入れて伝播された信号を表示しています。

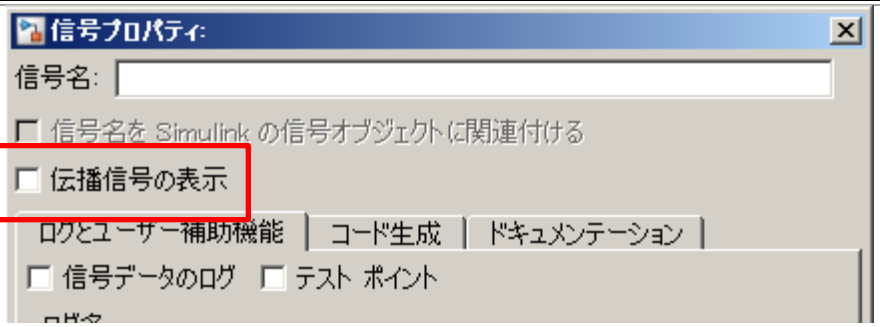




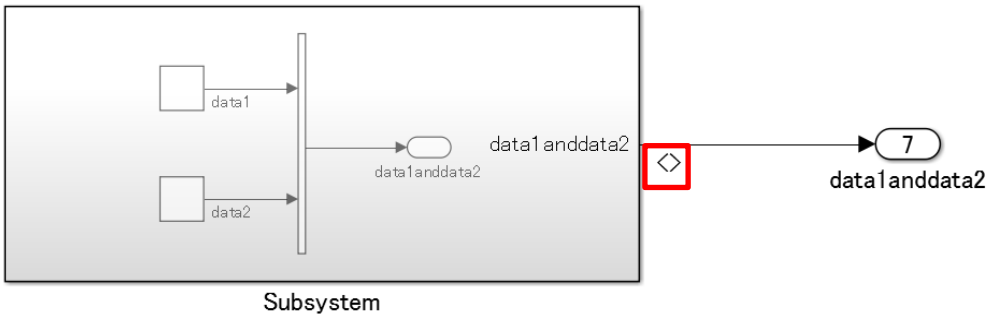


【誤】
{伝播信号の表示}にチェックを入れず信号を表示していません。

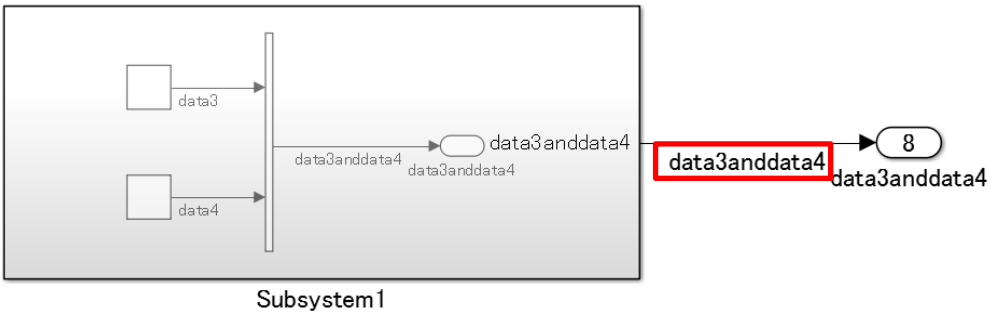




[Bus Creator]と[Outport]を結ぶ信号に名前が付与されていませんが、[Subsystem]と[Outport]を結ぶ信号の{伝播信号の表示}にチェックを入れています。

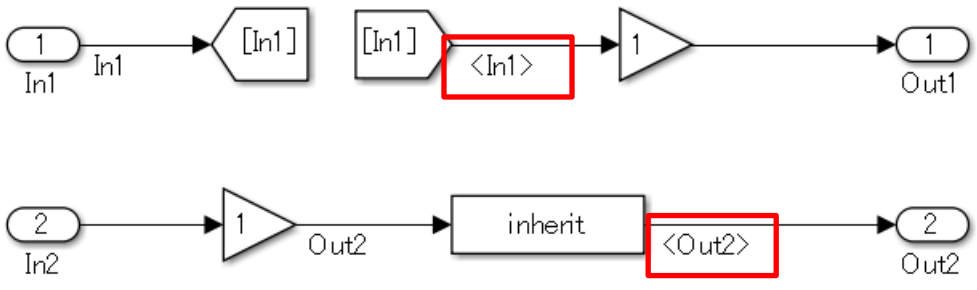


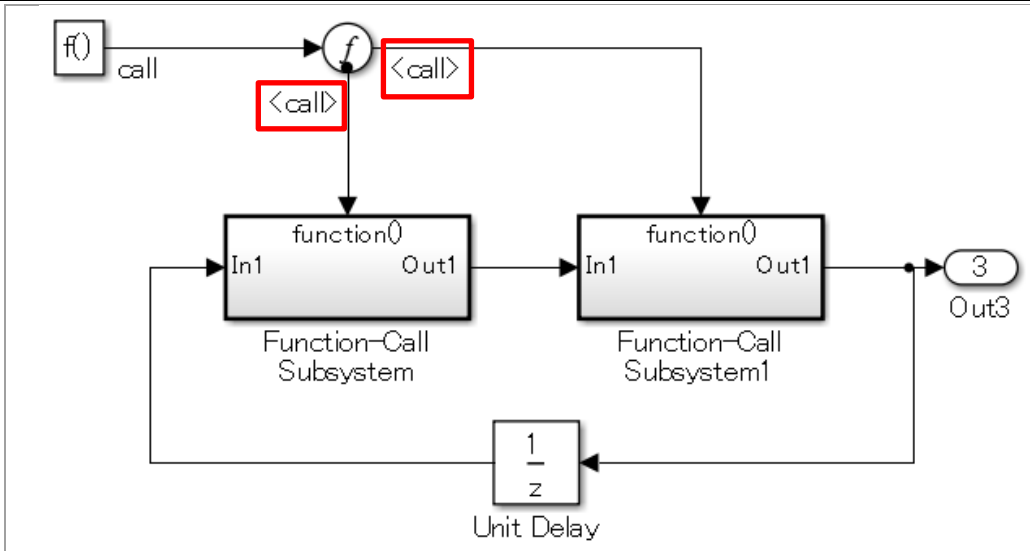
[Bus Creator]と[Outport]を結ぶ信号に名前が付与されていますが、[Subsystem]と[Outport]を結ぶ信号にも名前を付与しています。

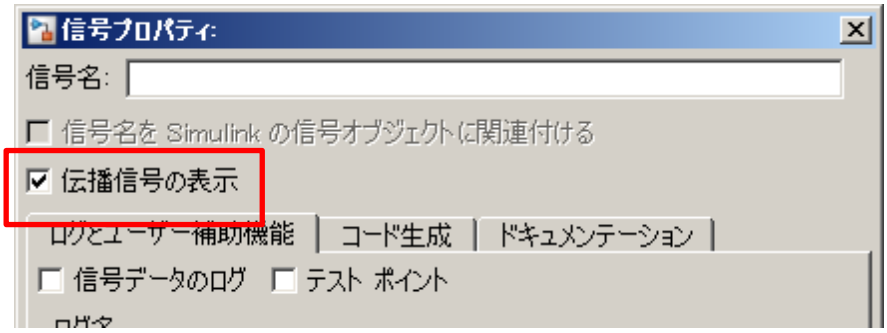


b	以下のブロックから出力される信号元に信号名が定義されている場合、{伝播信号の表示}はチェックを入れて伝播された信号を表示します。 ・[From] ・[Signal Specification] ・[Function-Call Split]	-

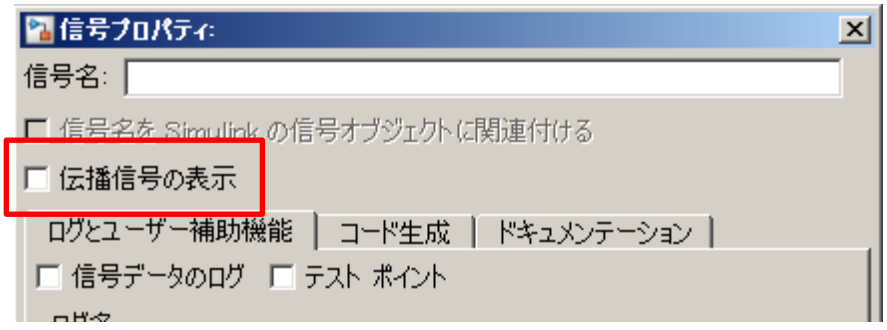
【正】
{伝播信号の表示}にチェックを入れて伝播された信号を表示しています。



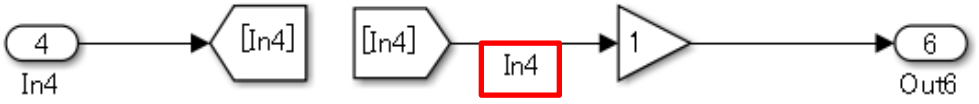


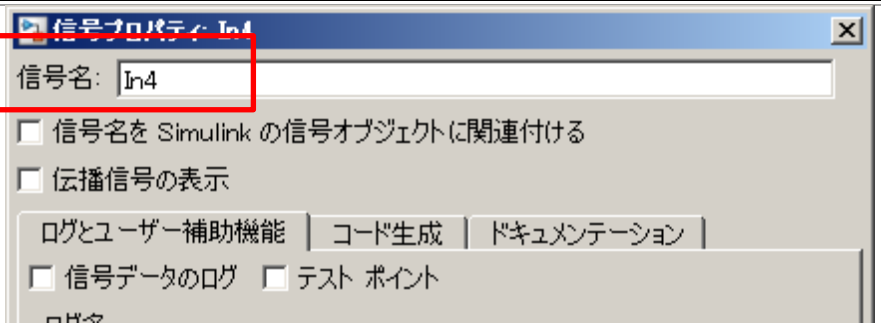


[Inport]と[Goto]を結ぶ信号に名前が付与されていないので、[From]と[Gain]を結ぶ信号の{伝播信号の表示}にチェックを入れなくても良い。



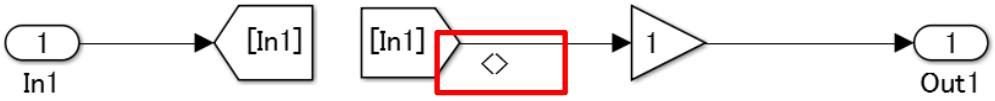
[Inport]と[Goto]を結ぶ信号に名前が付与されていないので、[From]と[Gain]を結ぶ信号に名前を付与しても良い。



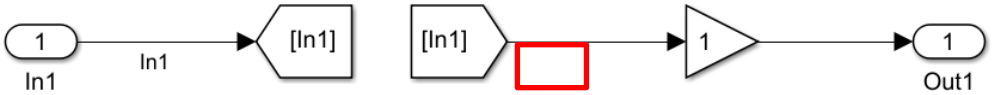


【誤】

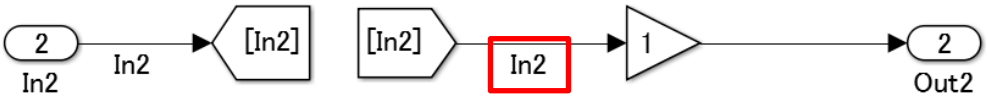
[Inport]と[Goto]を結ぶ信号に名前が付与されていませんが、[From]と[Gain]を結ぶ信号の{伝播信号の表示}にチェックを入れています。



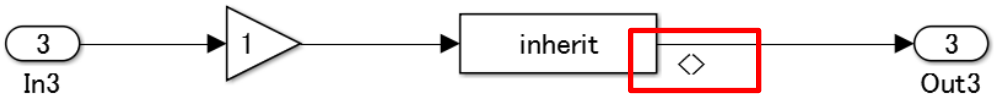
伝播する信号があるにも関わらず、{伝播信号の表示}にチェックを入れていません。



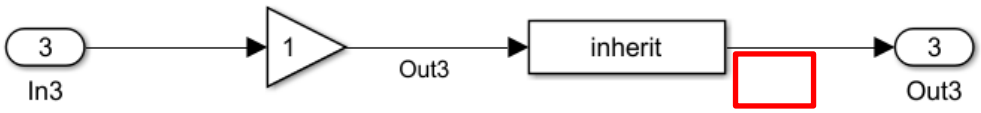
[Inport]と[Goto]を結ぶ信号に名前が付与されていますが、[From]と[Gain]を結ぶ信号にも名前を付与しています。



[Gain]と[Signal Specification]を結ぶ信号に名前が付与されていませんが、[Signal Specification]と[Outport]を結ぶ信号の{伝播信号の表示}にチェックを入れています。

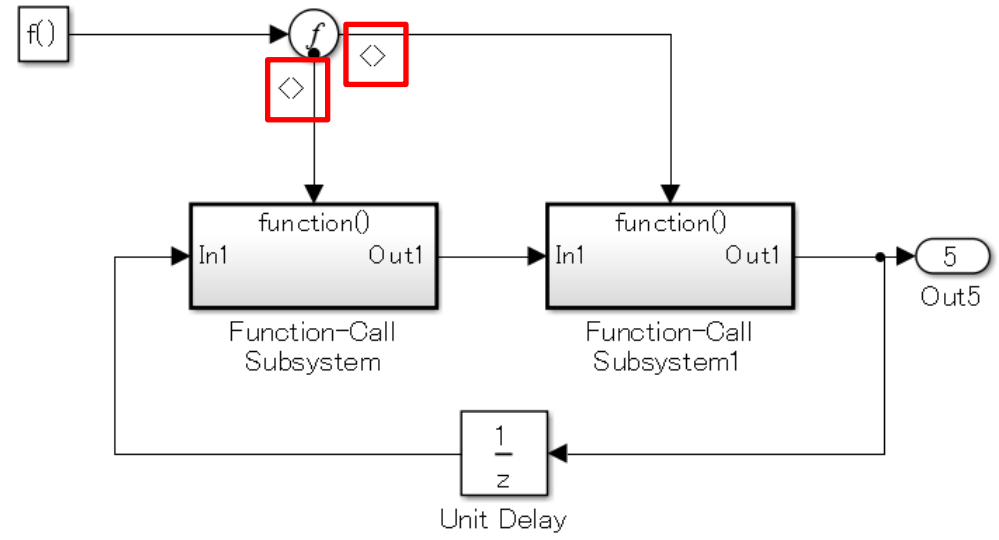


伝播する信号があるにも関わらず、{伝播信号の表示}にチェックを入れていません。

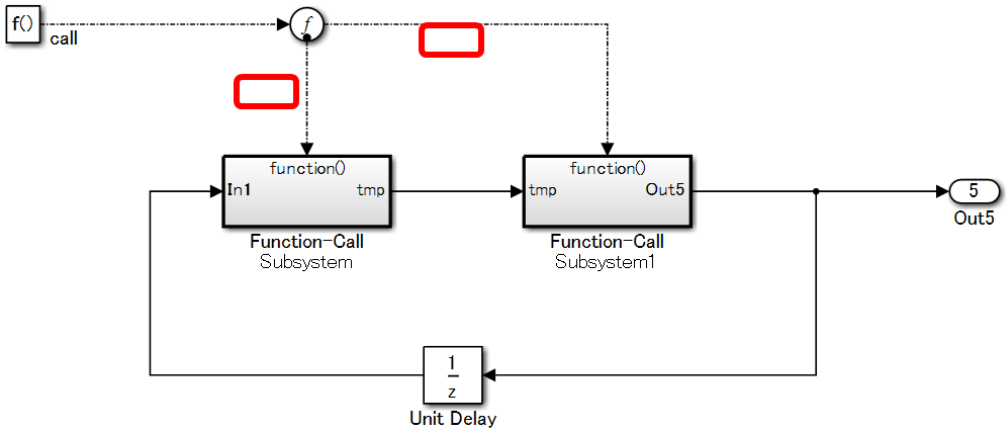


[Gain]と[Signal Specification] を結ぶ信号に名前が付与されていますが、[Signal Specification]と[Outport]を結ぶ信号にも名前を付与しています。

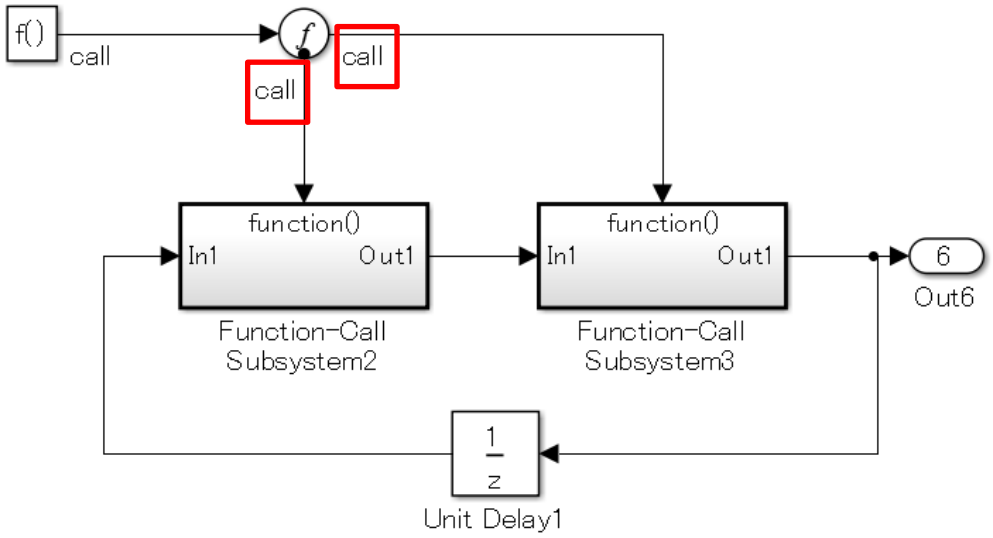
[Function-Call Generator]と[Function-Call Split]を結ぶ信号に名前が付与されていませんが、[Function-Call Split]と[Function-Call Subsystem]を結ぶ信号の{伝播信号の表示}にチェックを入れています。



伝播する信号があるにも関わらず、{伝播信号の表示}にチェックを入れていません。



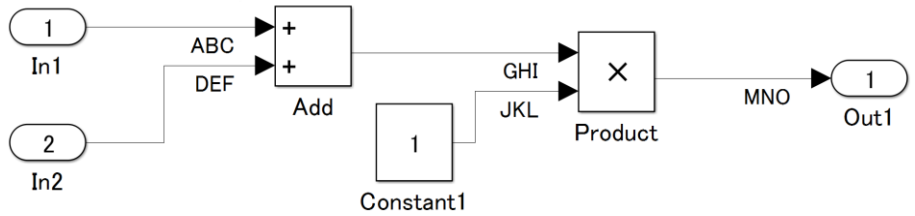
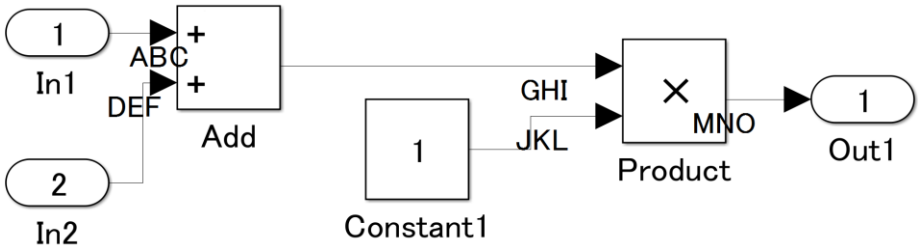
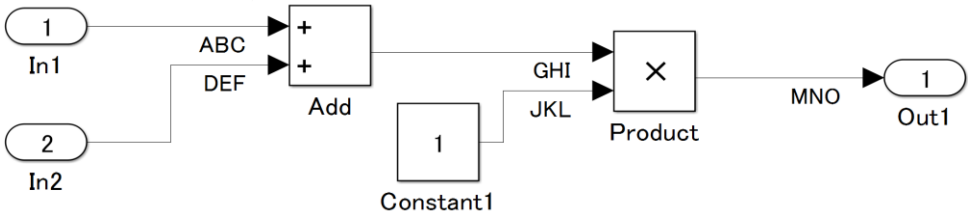
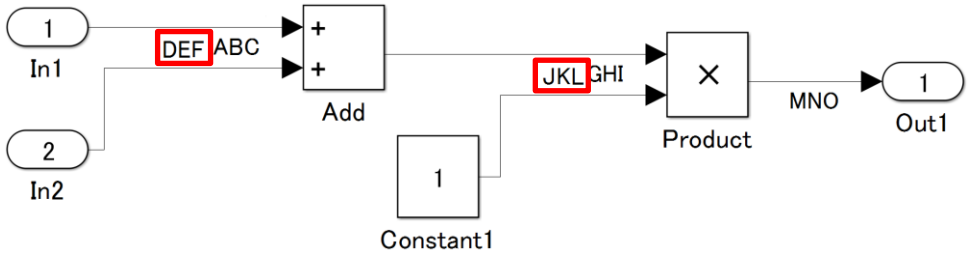
[Function-Call Generator]と[Function-Call Split]を結ぶ信号に名前が付与されていますが、[Function-Call Split]と[Function-Call Subsystem]を結ぶ信号にも名前を付与しています。

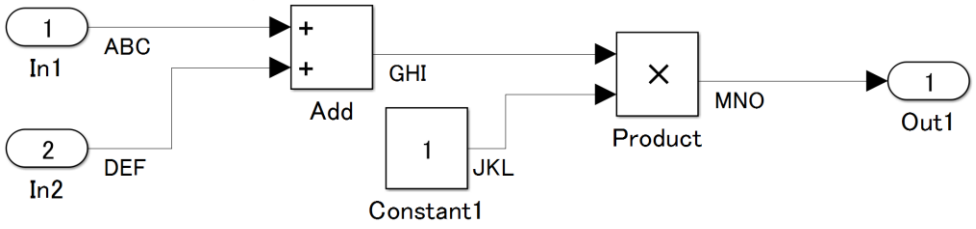


根拠	
サブ ID	**記述内容**
a	・信号線結線ミスを防止します。 ・信号線名のつけ間違いを防止します。

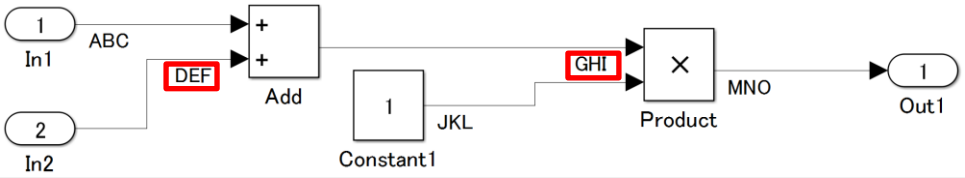
b	・信号線結線ミスを防止します。 ・信号線名のつけ間違いを防止します。 ・これらのブロックはデータの変換を行わないため、信号名の変更は不要です。

3.3.4. db_0097 : 信号とバスのラベルの位置

ルール ID : タイトル	db_0097 : 信号とバスのラベルの位置	
ルール		
サブ ID	記述内容	カスタムパラメーター
a	信号線やバスのラベルを、他のラベル、信号線、ブロックと重ならないようにします。	-
	【正】 信号線やバスのラベルが他のラベルや信号線、ブロックと重なっていません。  【誤】 信号線やバスのラベルが他のラベルや信号線、ブロックと重なっています。  	
b	信号線やバスのラベルは信号線の下に配置します。	-
	【正】 信号線やバスのラベルが信号線の下に配置されています。  【誤】 信号線やバスのラベルが信号線の上に配置されている箇所があります。  	
c	信号線やバスのラベルは接続元に配置します。	-
	【正】 信号線やバスのラベルが信号線の接続元に配置されています。	



【誤】
信号線やバスのラベルが信号線の接続先にも配置されています。

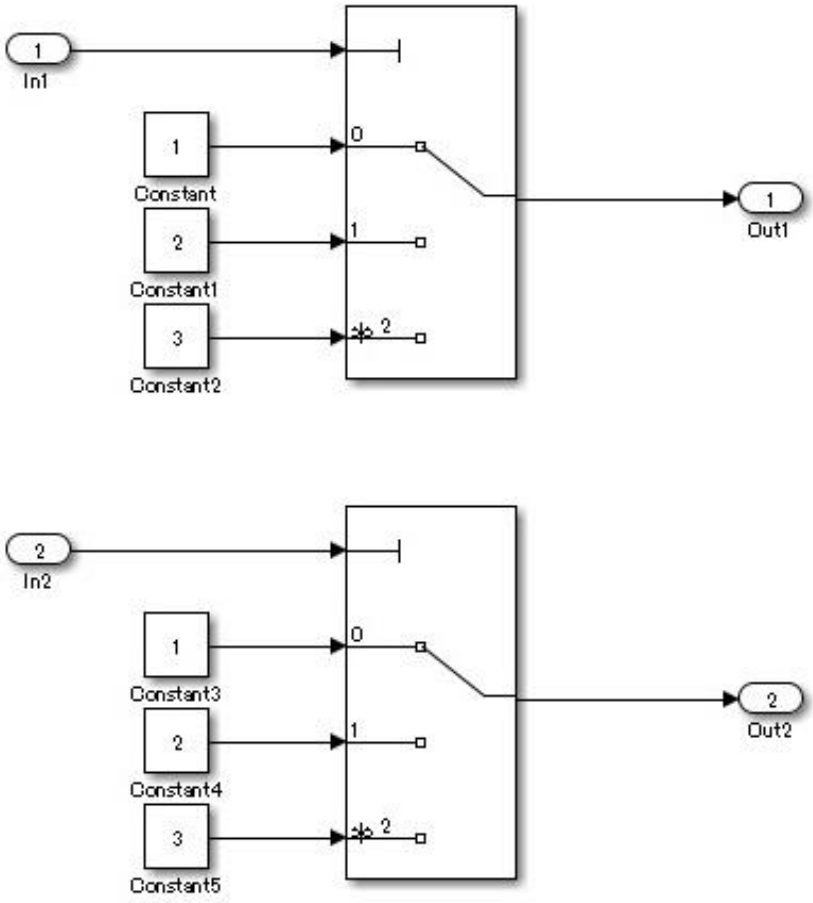


根拠	
サブ ID	**記述内容**
a	・信号線とバスのラベルを他のラベル、信号線、ブロックと重ねないことで、関連付くラベルと信号線・バスの誤認識を防ぎ、信号線名やバス名に対する可読性を向上します。
bc	・ラベルの表示位置を統一することで関連付くラベルと信号線・バスの誤認識を防ぎ、信号線名やバス名に対する可読性を向上します。

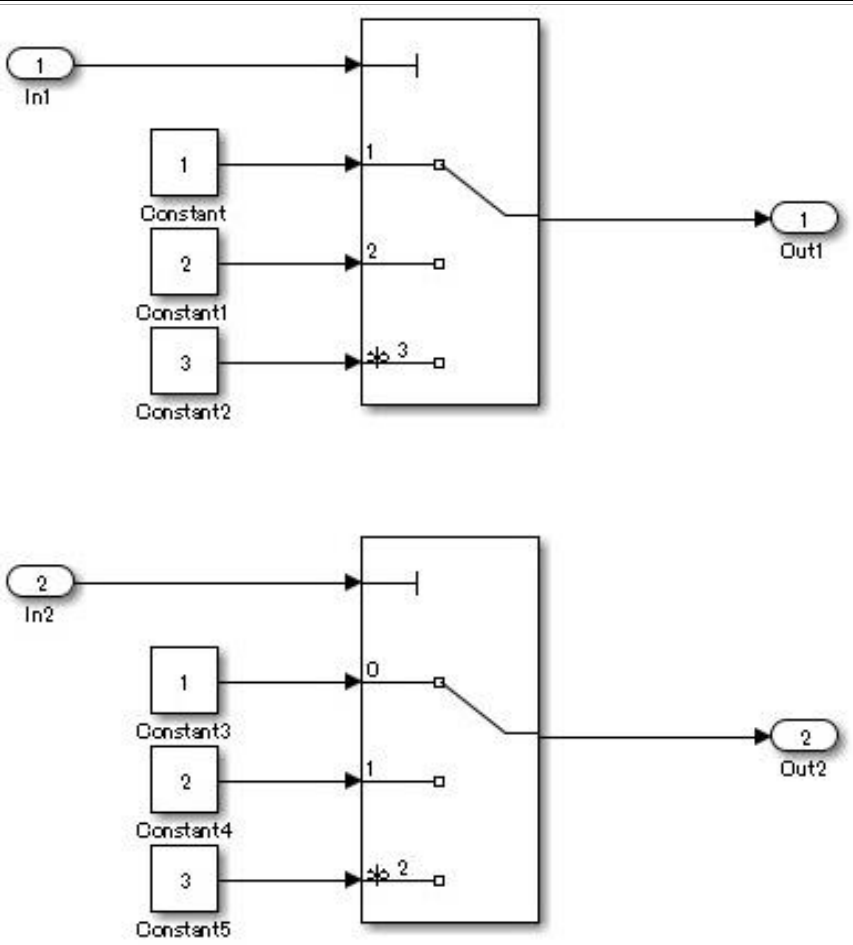
3.4. ブロック共通

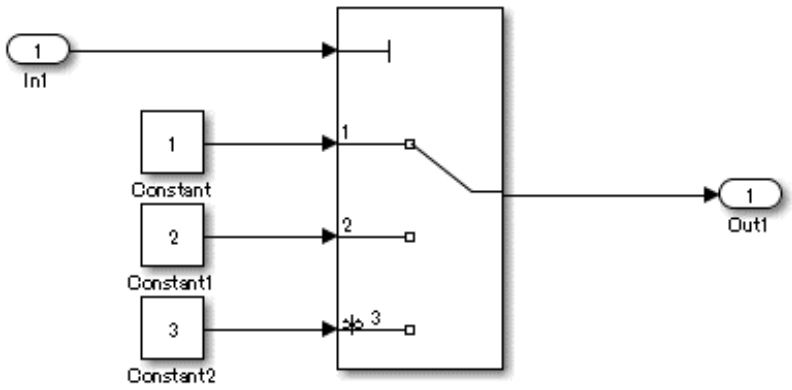
3.4.1. db_0112：インデックスの使用方法

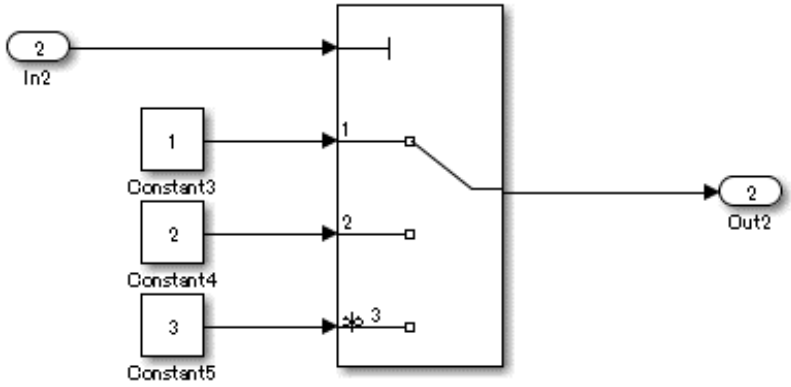
ルール ID：タイトル	**db_0112：インデックスの使用方法**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a1	ベクトル信号の{インデックス モード}は”0 ベース”にします。	-
	【正】 インデックスモードが 0 ベースに統一されています。	



【誤】
インデックスモードが統一されていません。

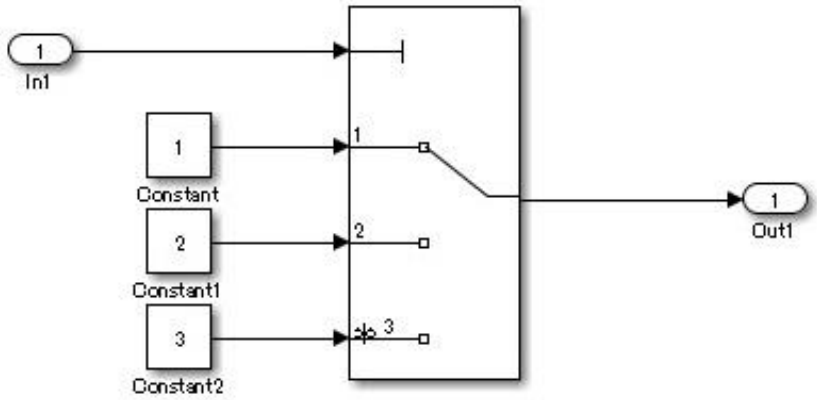
<table>
<tr><td></td><td> 
</td><td></td></tr>
<tr><td>a2</td><td>ベクトル信号の{インデックス モード}は”1 ベース”にします。</td><td>-</td></tr>
<tr><td></td><td colspan="2">【正】
インデックスモードが 1 ベースに統一されています。</td></tr>
</table>

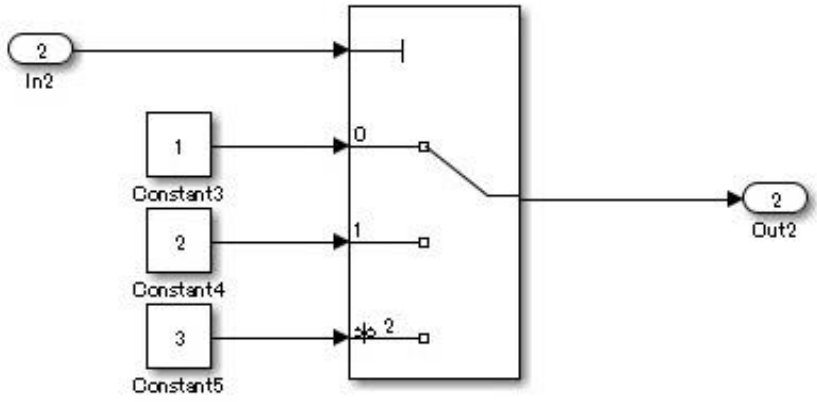




【誤】

インデックスモードが統一されていません。(db_0112_a1 と同じです。)



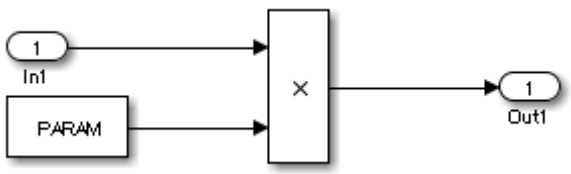
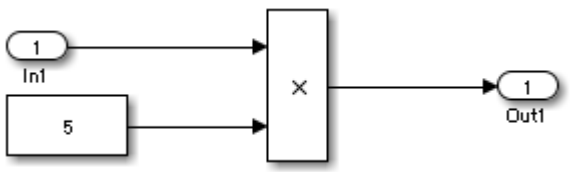


根拠	
サブ ID	**記述内容**
a1a2	・統一することでロジックが容易に理解できます。

3.4.2. db_0110 : ブロックパラメーターの記述方法

ルール ID : タイトル	**db_0110 : ブロックパラメーターの記述方法**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	ブロックパラメーターにおいて以下の記述をしません。 ・演算式の利用 ・データ型の変換 ・行または列の選択 ・MATLAB コマンド	-
根拠		
サブ ID	**記述内容**	
a	・演算、データ型変換、行または列の選択を記述すると、自動生成されたコード上ではマジックナンバーとなるので、モデルとコードの整合がとりづらくなります。また、パラメーター調整ができなくなります。 ・ブロック内に計算式を記述することで可読性が低下します。 ・MATLAB コマンドを記述するとコマンド実行結果がコードに反映されるため、モデルとコードの整合が取りづらくなります。	

3.4.3. jc_0645 : キャリブレーション対象の名前付き定数設定

ルール ID : タイトル	**jc_0645 : キャリブレーション対象の名前付き定数設定**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	キャリブレーション対象のブロックパラメーターは、名前つき定数として定義します。 キャリブレーション対象外の例 ・初期値パラメーターの 0 ・インクリメント・デクリメントの 1	-
	【正】  【誤】  	
根拠		

サブ ID	記述内容
a	モデルで直値にするとコードも直値になるためキャリブレーションできなくなります。

3.4.4. jc_0641 : サンプル時間の設定

ルール ID : タイトル	jc_0641 : サンプル時間の設定	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	ブロックの{サンプル時間}は”-1”(継承)に設定します。 ＜例外＞ ・[Inport]、[Outport] ・[Atomic Subsystem] ・[Unit Delay]、[Memory]等の状態変数を持つブロック ・[DataTypeConversion]、[Rate Transition]等の信号変換ブロック ・[Constant]等の外部入力を持たないブロック ・[Chart]	-
根拠		
サブ ID	**記述内容**	
a	・シミュレーション時間の違いによる処理結果の相違が起きる可能性があります。 ・特定のサンプル時間がブロックパラメーター個別に設定されると、メンテナンス性が低下します。	

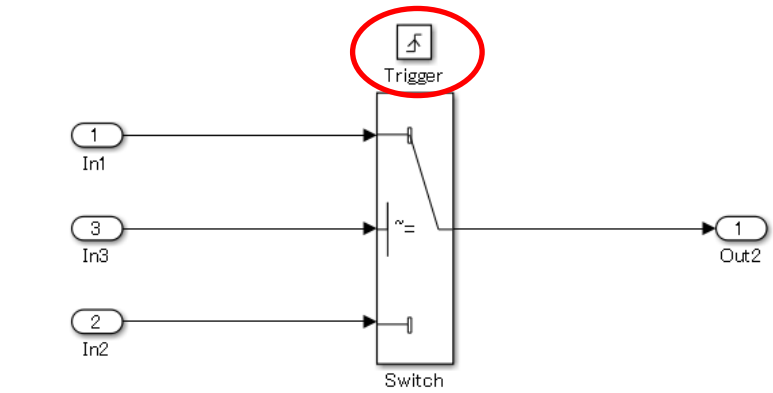
3.4.5. jc_0643 : 固定小数点設定

ルール ID : タイトル	jc_0643 : 固定小数点設定	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	{データ型}が固定小数点で、{スケーリング}で"勾配とバイアス"を選択した場合、{バイアス}を"0"にします。	-
根拠		
サブ ID	**記述内容**	
a	バイアスがモデル内で統一されていないと下記の問題が発生します。 ・見た目で動作をイメージできず、意図しないオーバーフロー/アンダーフローが発生します。 ・無駄な演算が発生し、コード効率/演算負荷が悪化します。	

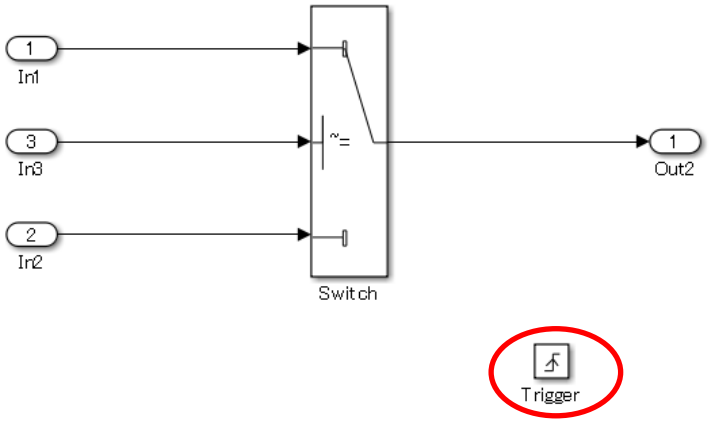
3.5. 条件付きサブシステム関連

3.5.1. db_0146 : 条件付きサブシステム内のブロック配置

ルール ID : タイトル	db_0146 : 条件付きサブシステム内のブロック配置	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	条件入力ブロックはサブシステム内の最上位に配置します。	-
	【正】	



【誤】



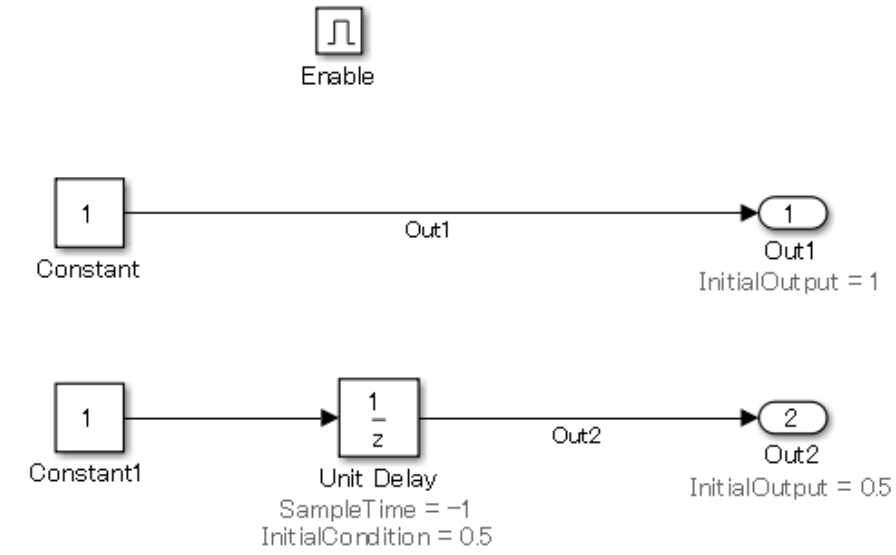
b	条件入力ブロックに近い扱いの下記ブロックはプロジェクトで定めた位置に配置します。 ・[ForEach] ・[ForIterator] ・[WhileIterator]	配置場所

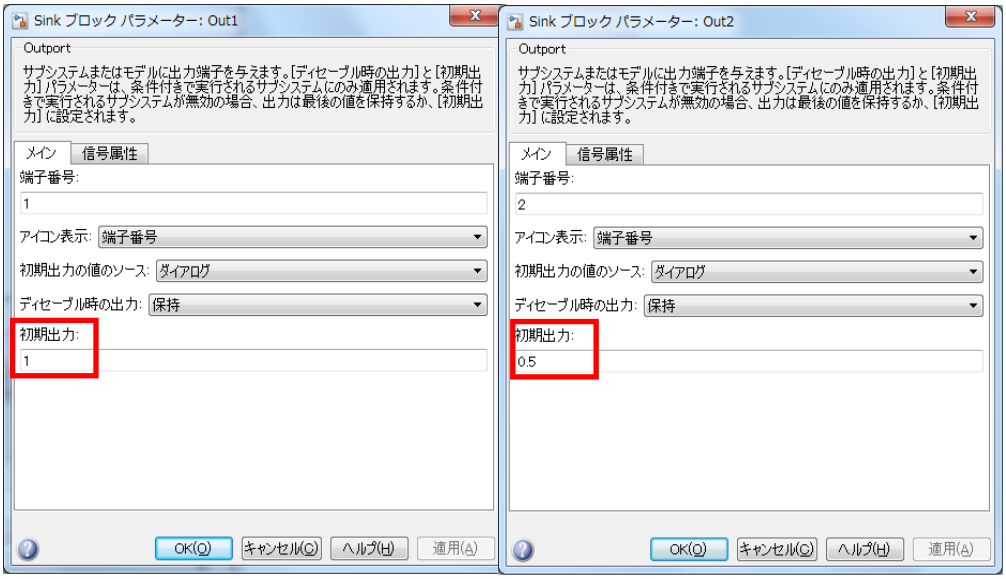
根拠

サブ ID	記述内容
ab	・条件付きサブシステム直下の階層なのかが判断しやすくなります。 ・条件付きサブシステムの外観と内部の配置を同一にすることで可読性を高める意図があります。

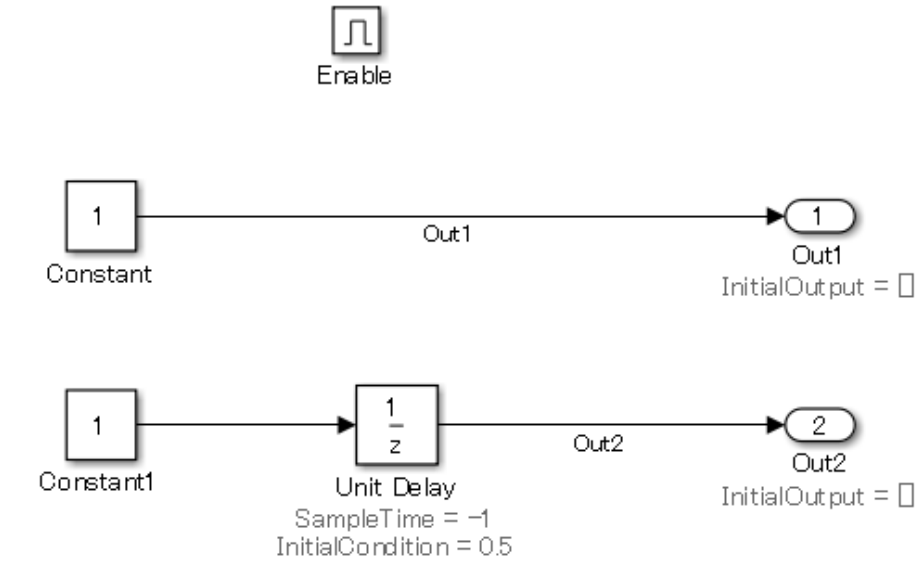
3.5.2. jc_0640 ： 条件付きサブシステムにおける Outport ブロックの初期値設定

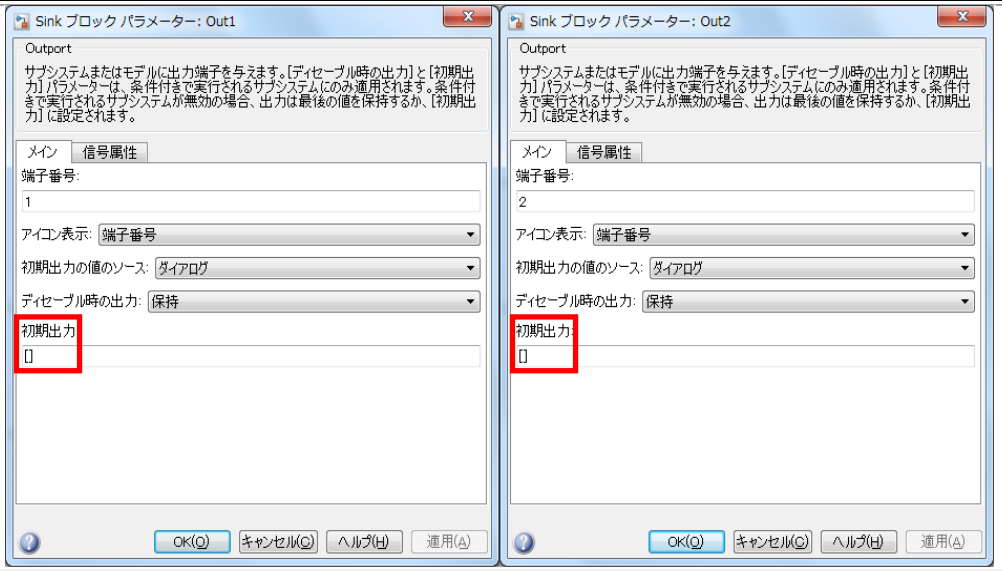
ルール ID：タイトル	jc_0640 ： 条件付きサブシステムにおける Outport ブロックの初期値設定	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	モデルに含まれている条件付きサブシステムにおいて、初期条件を設定したブロック ([Constant]、[Delay]など)を[Outport]に接続している場合、その初期値を[Outport]にも明示的に設定します。 　ただし、条件付きサブシステムの出力信号が[Merge]に接続されている場合は、その初期値は[Merge]に明示的に設定します。	-
	【正】 　初期値が定義されています。	





【誤】
初期値が定義されていません。



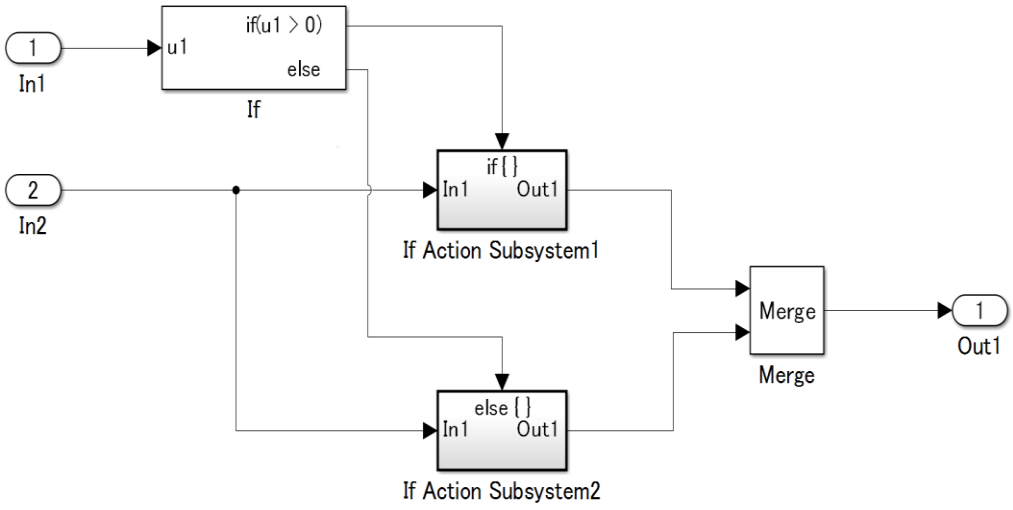


根拠	
サブ ID	**記述内容**
a	・初期値を明確にしないことで意図しない動作を起こす可能性があります。

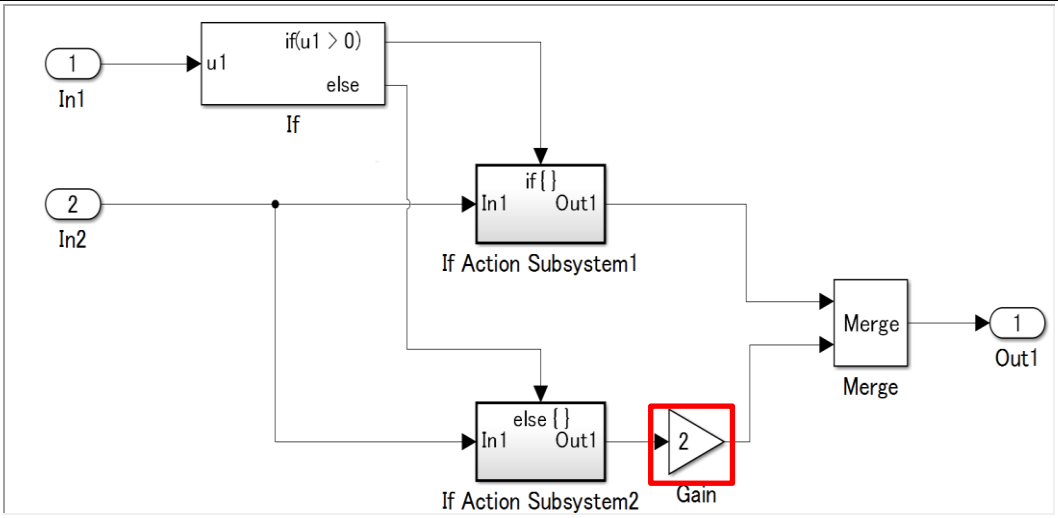
3.5.3. jc_0659 ： Merge ブロックへ入力する信号線の使用制限

ルール ID : タイトル	**jc_0659 ： Merge ブロックへ入力する信号線の使用制限**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	[Merge]には、条件付きサブシステムの出力信号のみを入力します。	-

【正】
[Merge]には、条件付きサブシステムの出力信号のみが入力されています。



【誤】
[Merge]に[Gain]の出力信号が入力されています。

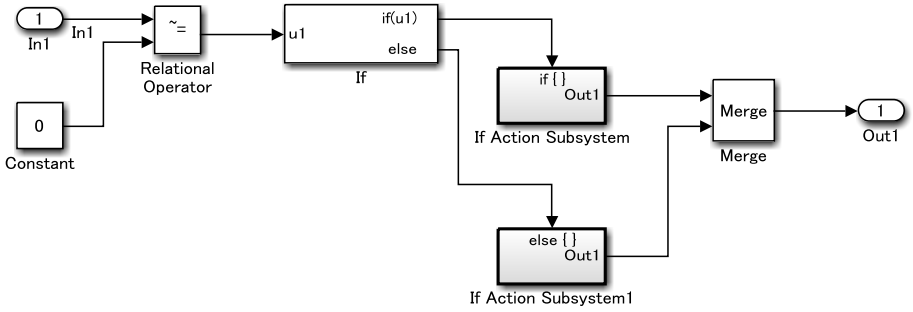


根拠	
サブ ID	**記述内容**
a	・意図したとおりにシミュレーションが動かなくなることを防止します。

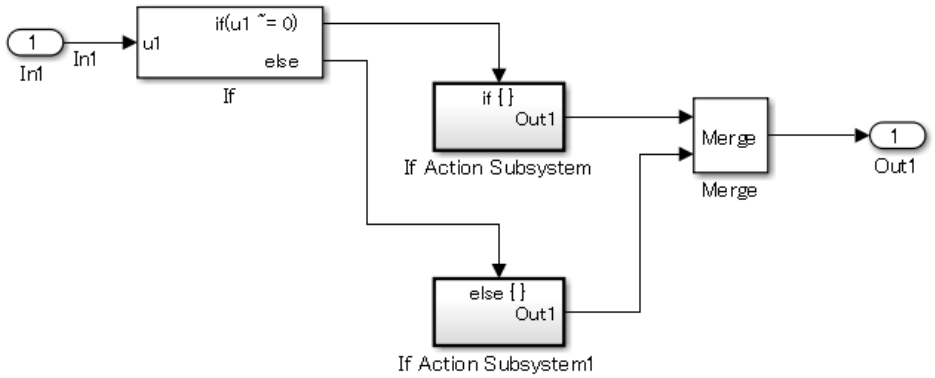
3.5.4. na_0003：If ブロックの使用方法

ルール ID：タイトル	**na_0003：If ブロックの使用方法**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	[If]の{If 式}、{Elseif 式}には入力信号以外を定義しません。	-

【正】
{if 式}には入力変数のみ定義されています。



【誤】
{if 式}に比較演算が定義されています。



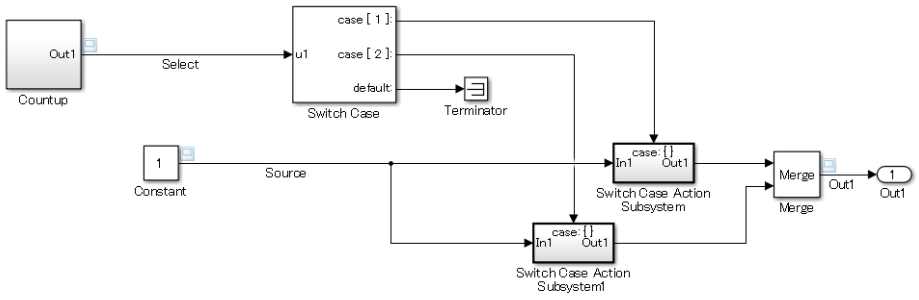
根拠

サブ ID	記述内容
a	・[If]の中に論理演算式を記述せず[If]の外に記述することで、見た目で制御仕様を理解し易くします。 ・また、[If]の外に記述することで、論理演算式に焦点をあてた検証を可能にします。

3.5.5. jc_0656 ： 条件付き制御フローブロックの使用方法

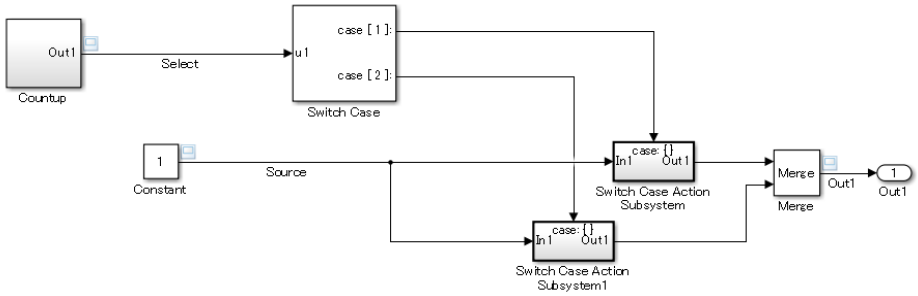
ルール ID：タイトル	jc_0656 ： 条件付き制御フローブロックの使用方法	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	[If]、[Switch Case]は、下記の設定により、全ての条件における動作を明示的にします。 ・[if]は、{else 条件の表示}にチェックを入れます。 ・[Switch Case]は、{default ケースを表示}にチェックを入れます。	-

【正】
default の動作あり



【誤】

default の動作なし



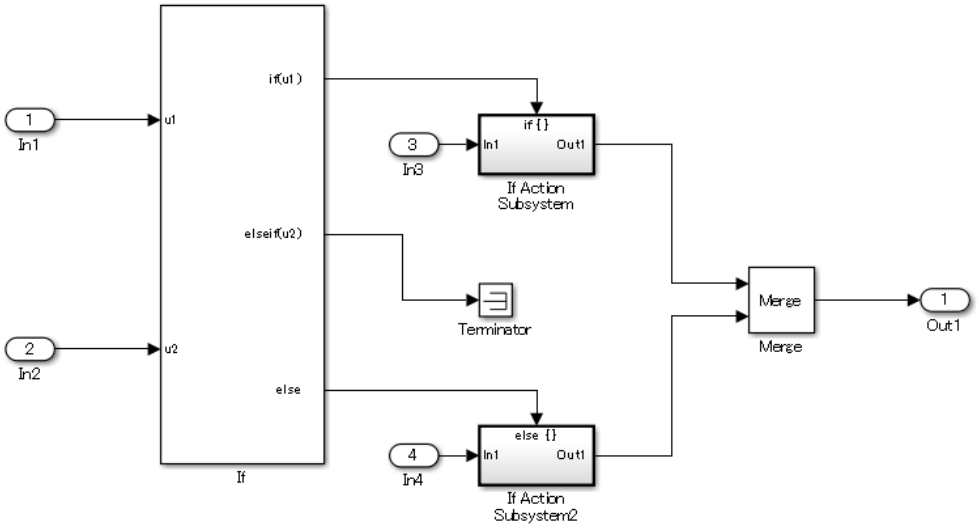
根拠	
サブ ID	**記述内容**
a	・モデル上で例外処理(else、default)を明示的に設定することで、何も行わないという処理なのか、設計漏れ(記述忘れ)なのか判断し易くなります。

3.5.6. jc_0657 ： 条件付き制御フローブロックと Merge ブロックによる出力値保持

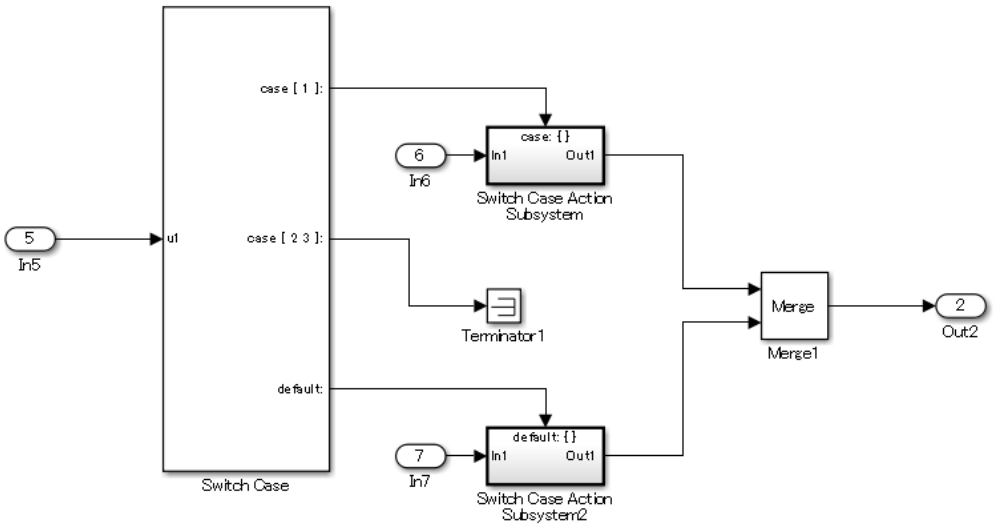
ルール ID：タイトル	jc_0657 ： 条件付き制御フローブロックと Merge ブロックによる出力値保持	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a1	[If]、[Switch Case]と[Merge]で機能を切り替える際に、過去の値を保持したい場合は、[Outport]に[Terminator]を接続します。	-

【正】

[If]の例

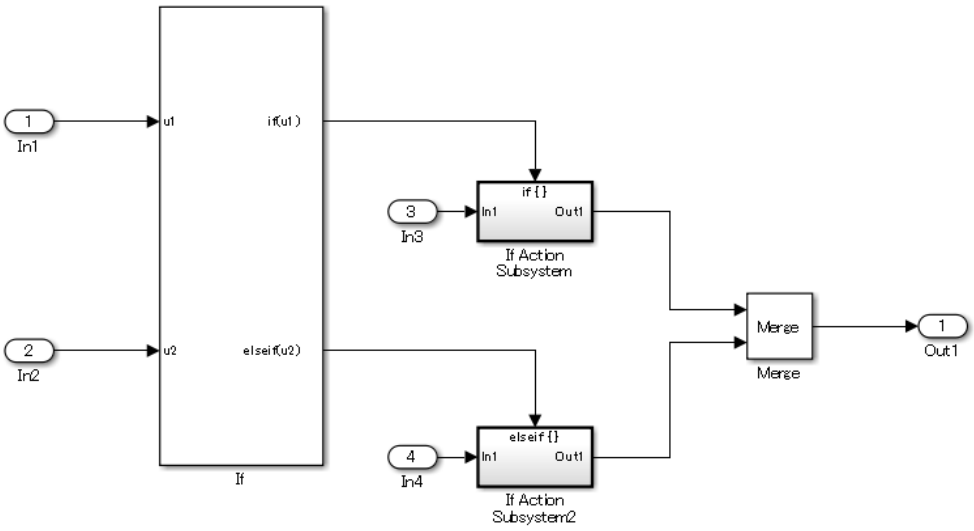


[Switch Case]の例

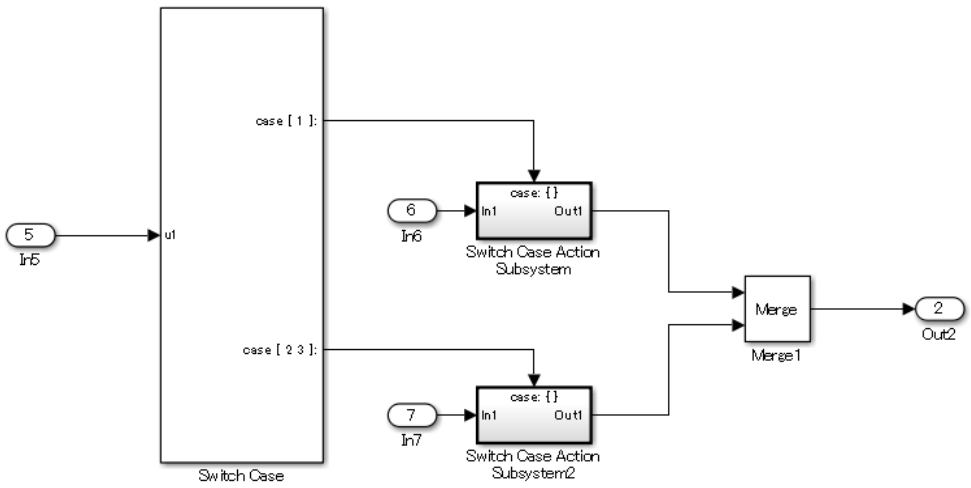


【誤】

[If]の例



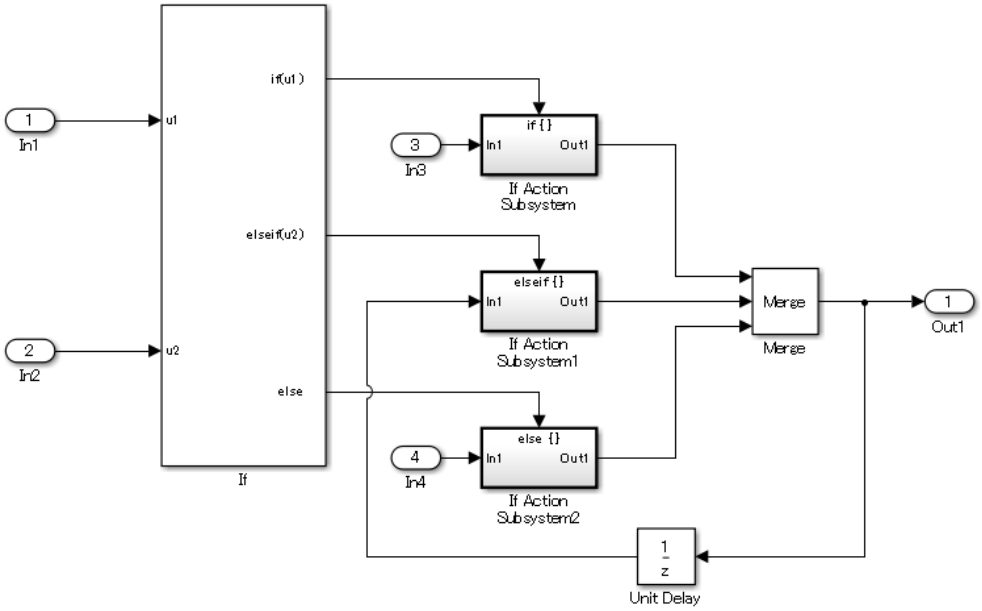
[Switch Case]の例



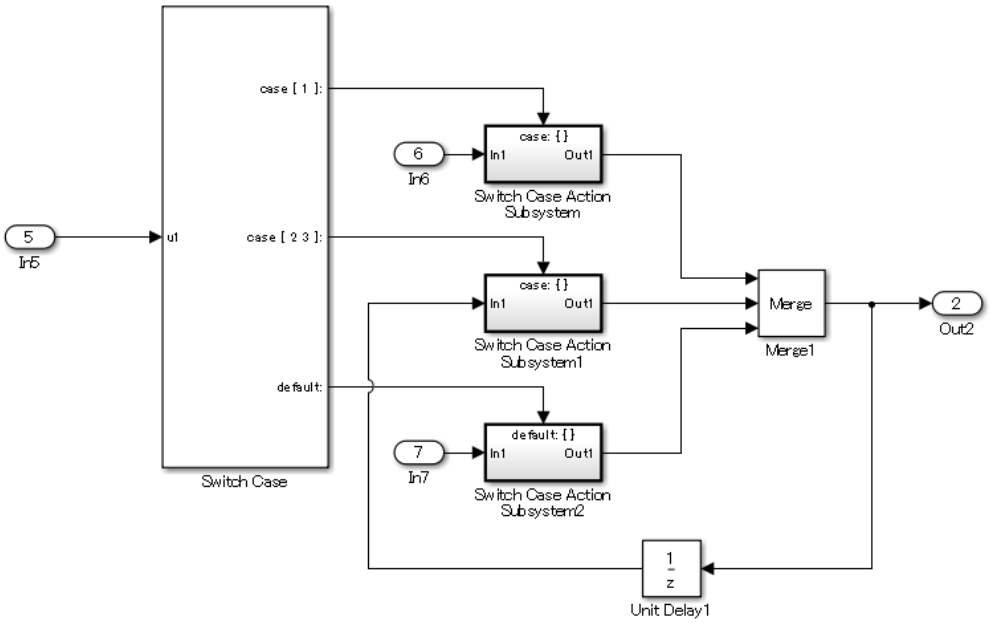
a2	[If]、[Switch Case]と[Merge]で機能を切り替える際に、過去の値を保持したい場合は、遅延ブロックを使用してフィードバックループにします。	-

【正】

[if]の例

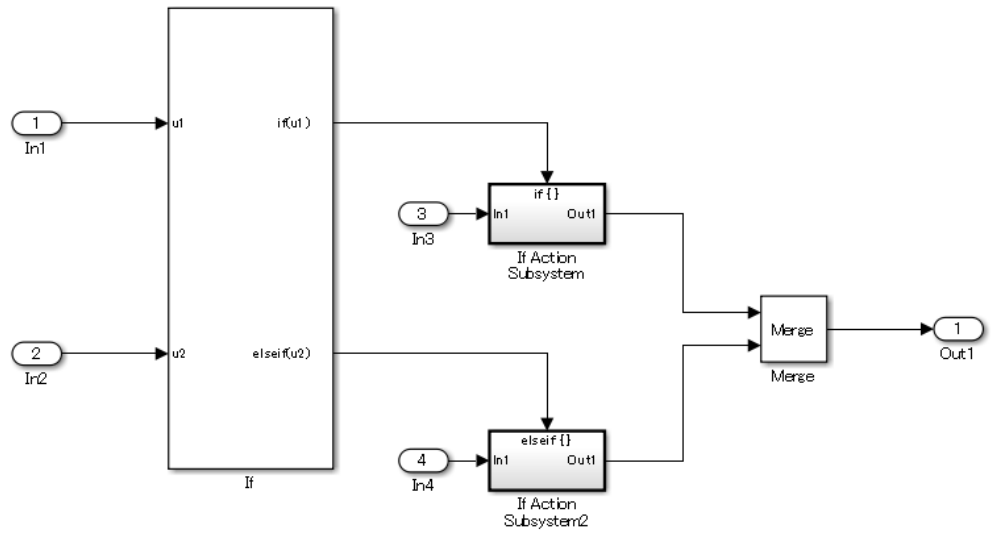


[Switch Case]の例

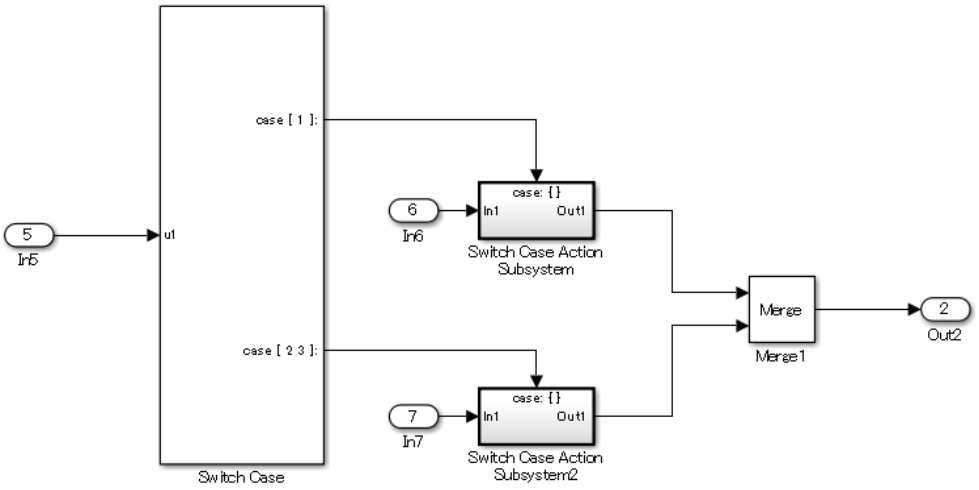


【誤】

[If]の例



[Switch Case]の例

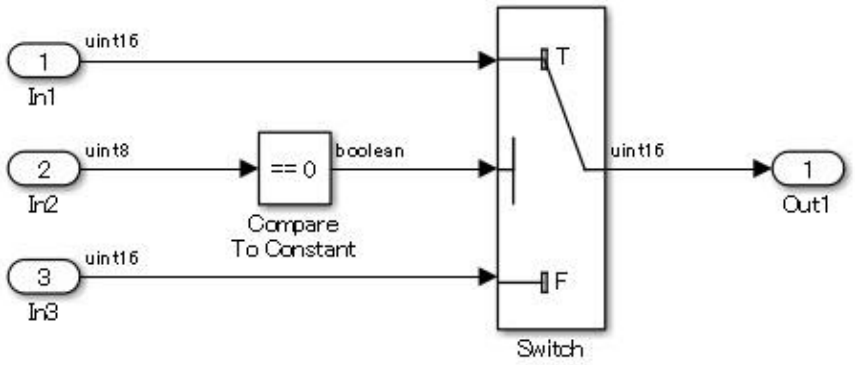


根拠	
サブ ID	**記述内容**
a1	・コード効率が向上します。 ・default(else)以外においても過去の値を保持する場合は、[Terminator]への接続を使用することができます。
a2	・過去の値を保持することが明示的になります。

3.6. 演算系ブロック

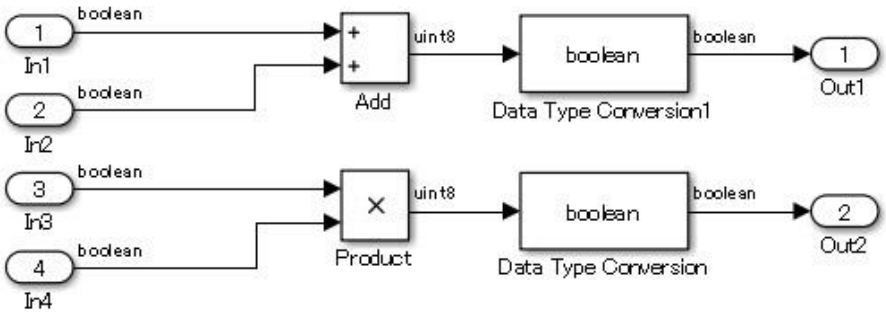
3.6.1. na_0002：基本的な論理演算と数値演算の適切な実装

ルール ID：タイトル	**na_0002：基本的な論理演算と数値演算の適切な実装**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	数値型信号の入力を期待するブロックには、論理型信号を入力しません。	数値型信号の入力を期待するブロック
	【正】 数値と数値の一致性を比較しています。	

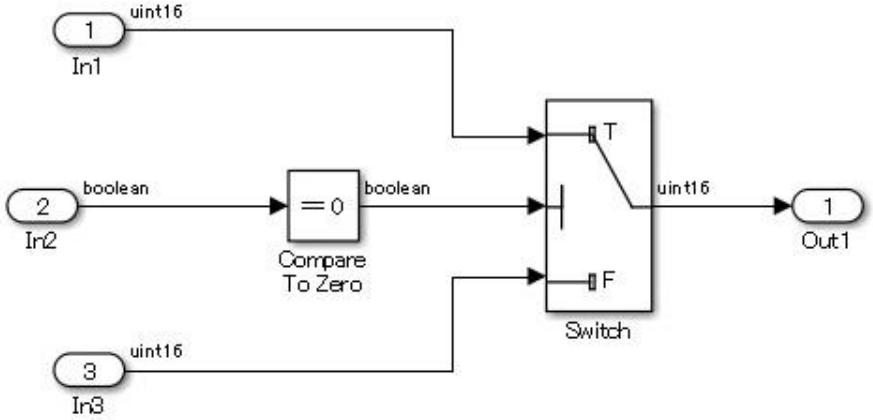


【誤】

論理出力を、数値入力を処理するブロックの入力に直接接続しています。

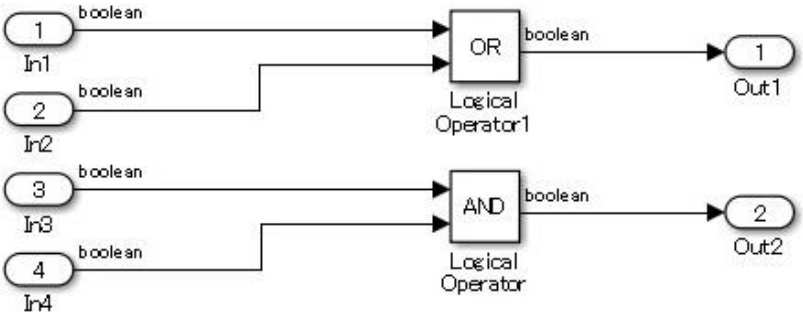


論理型の信号を数値と比較しています。

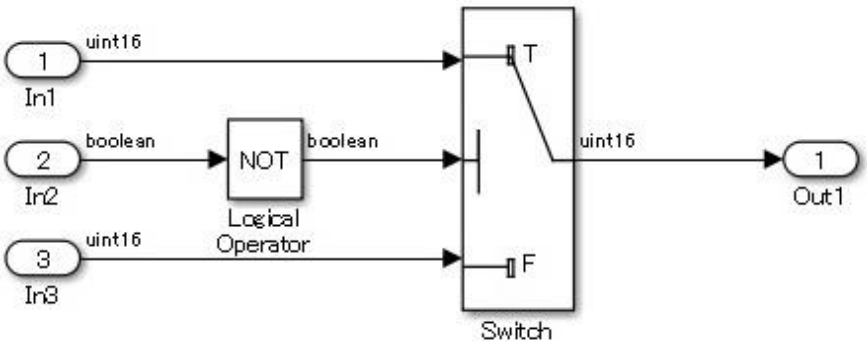


b	論理型信号の入力を期待するブロックには、数値型信号を入力しません。	論理型信号の入力を期待するブロック

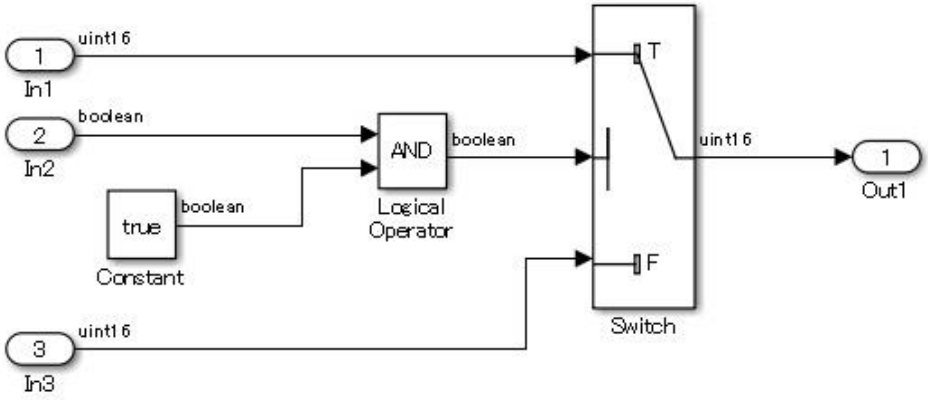
【正】



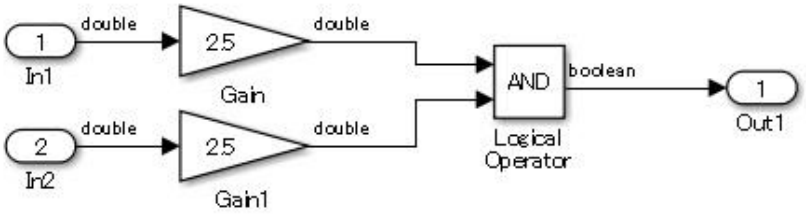
論理型の信号を論理演算によって反転しています。



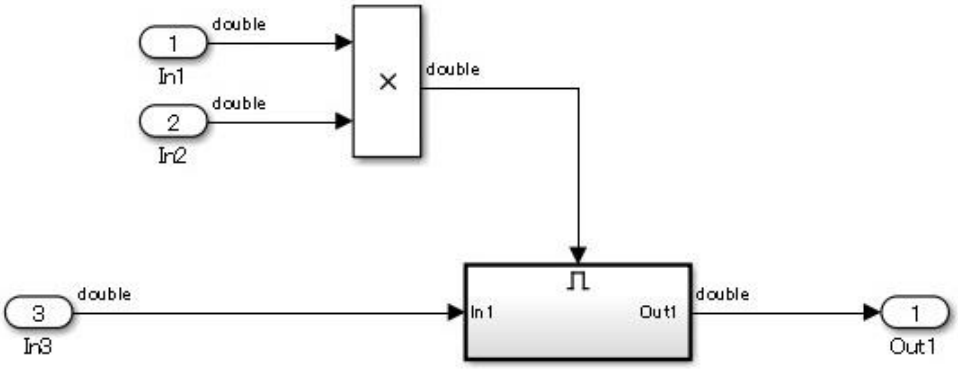
論理型の信号を、論理演算を用いて判定しています。



【誤】
・論理演算を実行するためのブロックを、数値演算の実行に使用しています。
・数値出力を、論理入力を処理するブロックの入力に接続しています。



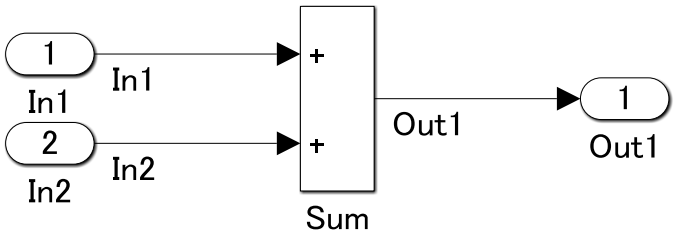
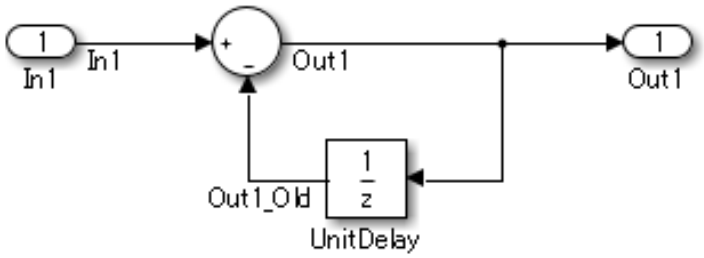
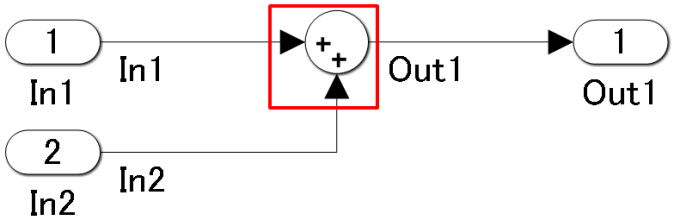
・数値演算を実行するためのブロックを、論理演算の実行に使用しています。
　Enable Port は論理値以外を入力することができますが、On か Off しか存在しない論理信号を期待するブロックです。
　[Product]は double と double の演算を行っていますが、数値演算結果を Enable Port という論理値を期待するブロックに接続しているので、[Product]は論理演算を行っていることになります。

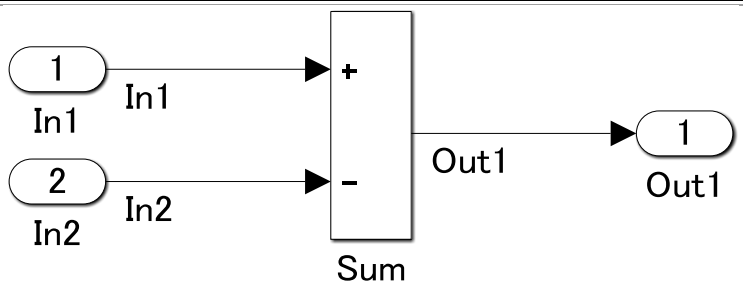


根拠

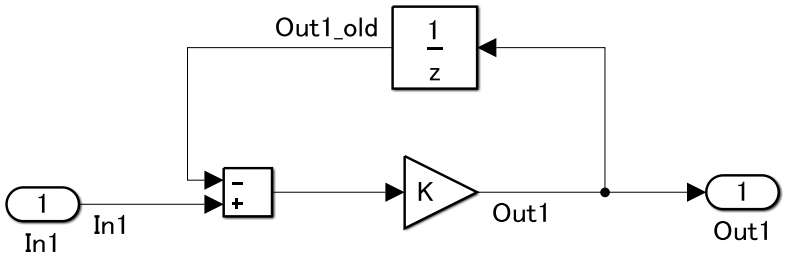
サブ ID	記述内容
ab	・論理と数値を同じ扱いにして計算すると、元の意図が不明になり、次の変更時に意図を間違え、ミスの要因になります。

3.6.2. jc_0121 ： 加減算ブロックの使用方法

ルール ID : タイトル	jc_0121 ： 加減算ブロックの使用方法	
ルール		
サブ ID	記述内容	カスタムパラメーター
a	加減算ブロックの{アイコン形状}は”四角形”に設定します。 ただし、フィードバックループの場合は、{アイコン形状}を”円形”に設定できます。	-
	【正】 加減算ブロックの{アイコン形状}を”四角形”に設定しています。  加減算ブロックの第二入力がフィードバックループなので、{アイコン形状}を”円形”に設定できます。  【誤】 フィードバックループではないのに、加減算ブロックの{アイコン形状}を”円形”に設定しています。  	
b	加減算ブロックの第一入力の符号は”+”に設定します。 ただし、フィードバックループの場合は、第一入力の符号を”-”に設定できます。	-
	【正】 加減算ブロックの第一入力の符号を”+”に設定しています。	

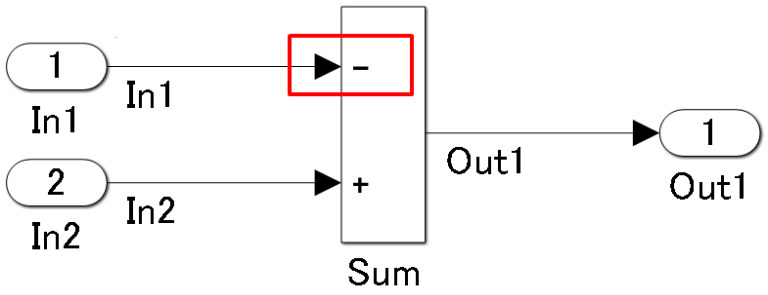


加減算ブロックの第一入力がフィードバックループなので、符号を”-”に設定できます。



【誤】

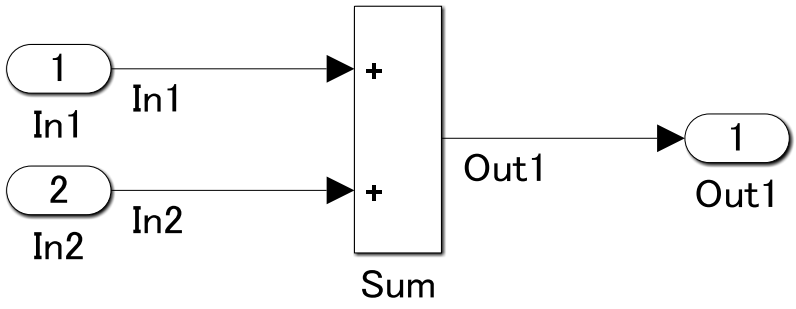
加減算ブロックの第一入力の符号が”-”に設定しています。



c	加減算ブロックの入力数は 2 つ以下に設定します。	-

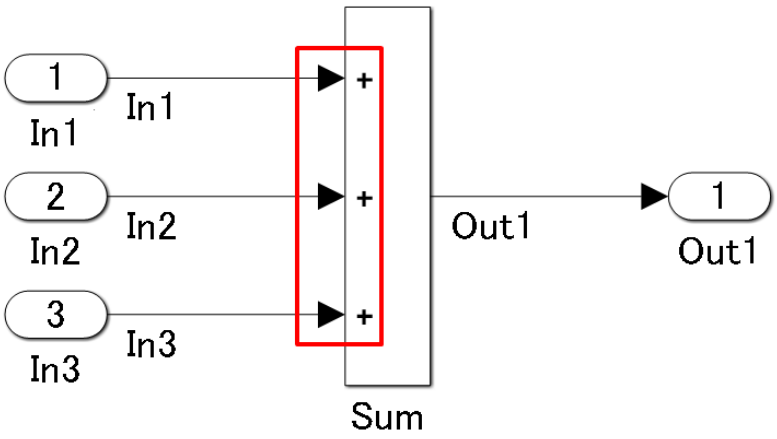
【正】

加減算ブロックの入力数を 2 つ以下に設定しています。



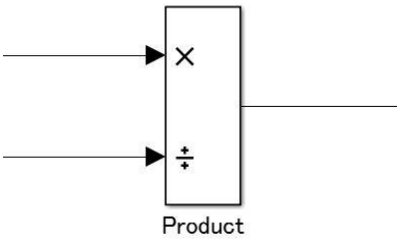
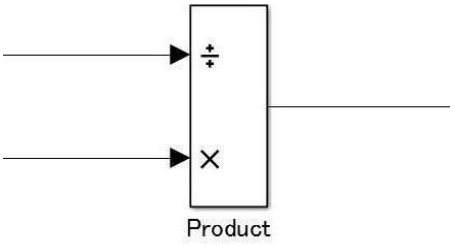
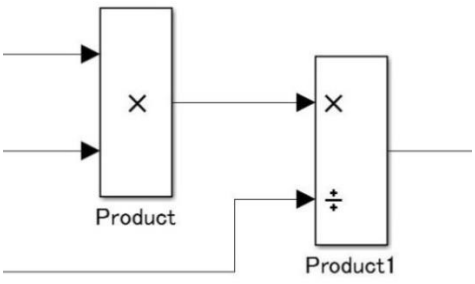
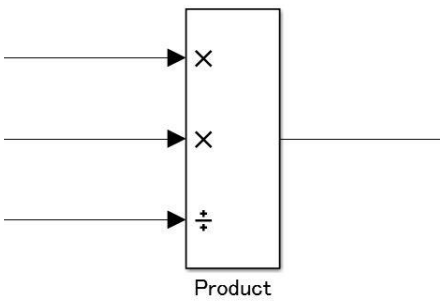
【誤】

加減算ブロックの入力数を 3 つ以上に設定しています。



根拠	
サブ ID	**記述内容**
a	・使用方法を揃えることで可読性を向上します。
b	・第一入力の符号が統一される事で制御仕様の可読性が向上します。
c	・演算順序を明確に規定できます。

3.6.3. jc_0610：乗除算ブロックの演算子順序

ルール ID：タイトル	**jc_0610：乗除算ブロックの演算子順序**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	乗除算ブロックの第一入力の符号は"*"に設定します。	-
	【正】 第一入力の符号を"*"に設定しています。  	【誤】 第一入力の符号を"/"に設定しています。 
b	乗除算ブロックの入力数は 2 つ以下に設定します。	-
	【正】 入力数を 2 つ以下に設定しています。  	【誤】 入力数を 3 つ以上に設定しています。 
根拠		
サブ ID	**記述内容**	
a	・ブロックを確認した場合、数式とブロックの入力順が逆になり、可読性が悪くなります。	

	・浮動小数点の場合、ブロック通りの演算順序((1÷第一入力) × 第二入力)のコードが生成されますが、除算を後にすれば演算回数を削減できます。
b	・演算順序が明確に規定できます。

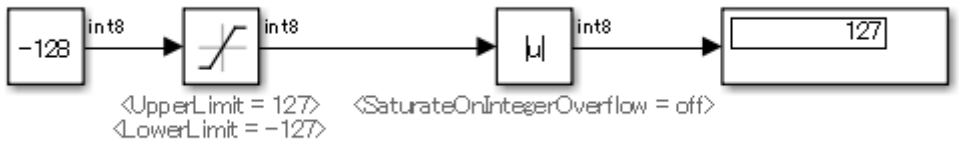
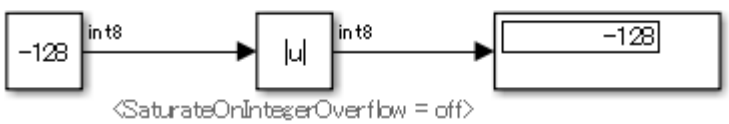
3.6.4. jc_0611：乗除算ブロックの入力符号

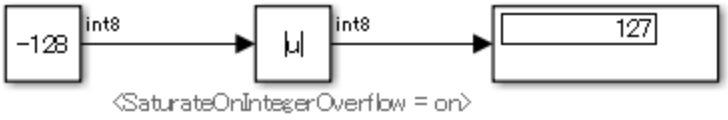
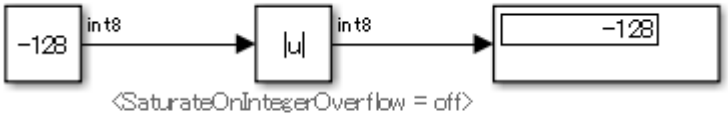
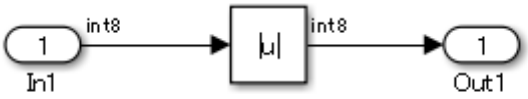
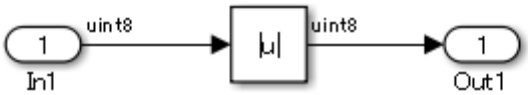
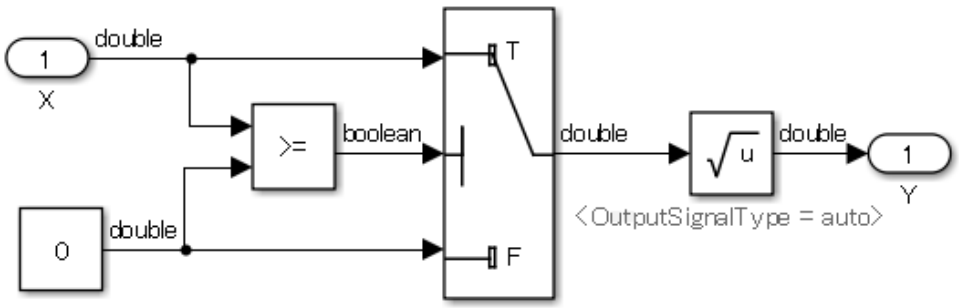
ルール ID：タイトル	jc_0611：乗除算ブロックの入力符号	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	乗除算ブロックの入力に固定小数点型を利用する場合は、入力信号のデータ型の符号の有無を同一にします。	-
根拠		
サブ ID	**記述内容**	
a	・固定小数点コードの生成時に LSB 毎のユーティリティ関数が作成されます。使用する型を統一する事でユーティリティ関数の数を抑制できます。	

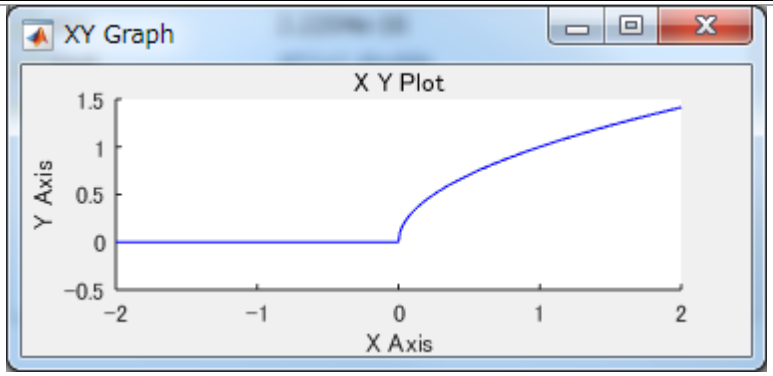
3.6.5. jc_0794：Simulink における除算

ルール ID：タイトル	jc_0794：Simulink における除算	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	除算を行う場合、ゼロ割回避の処理を入れます。	-
根拠		
サブ ID	**記述内容**	
a	・意図した動作、コード生成結果にならない可能性があります。	

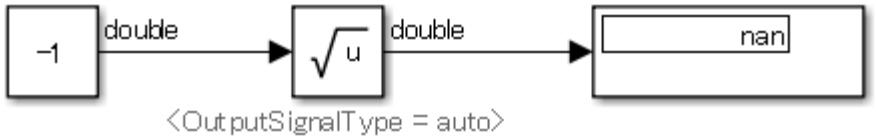
3.6.6. jc_0805：数値演算ブロックの入力

ルール ID：タイトル	jc_0805：数値演算ブロックの入力	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a1	[abs]を使用する場合は、符号付き整数型の最小値を入力しません。	-
	【正】  【誤】  	

a2	[abs]を使用する場合は、{整数オーバーフローで飽和}にチェックを入れます。	-
	【正】  【誤】  	
b	[abs]を使用する場合は、符号なしの整数型や固定小数点型を入力しません。	-
	【正】  【誤】  	
c1	[Sqrt]を使用する場合は、負の数を入力しません。	-
	【正】 負の数を 0 で飽和  実行結果	

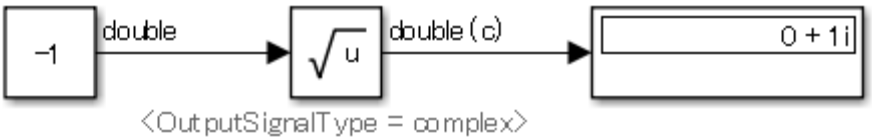


【誤】

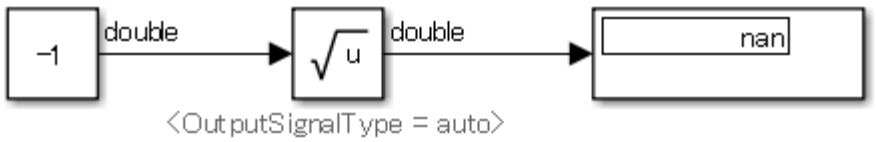


c2	[Sqrt]を使用する場合は、{出力信号タイプ}を”複素数”に設定します。	-

【正】



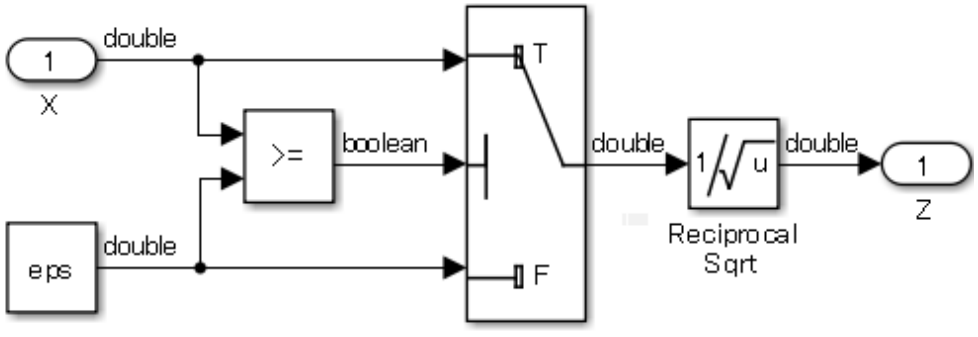
【誤】



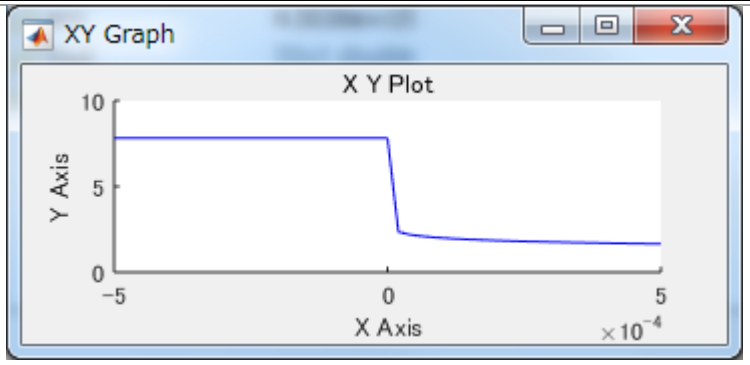
d	[Reciprocal Sqrt]を使用する場合、ゼロ以下の数を入力しません。	-

【正】

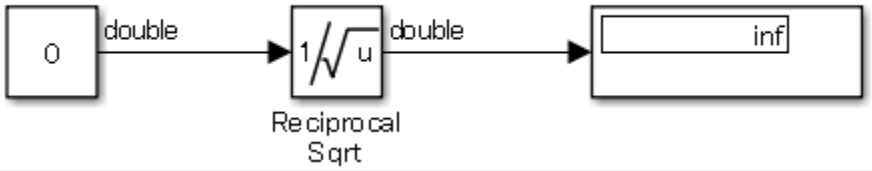
eps 未満を eps で飽和



実行結果：Y=log10(Z)としてプロット



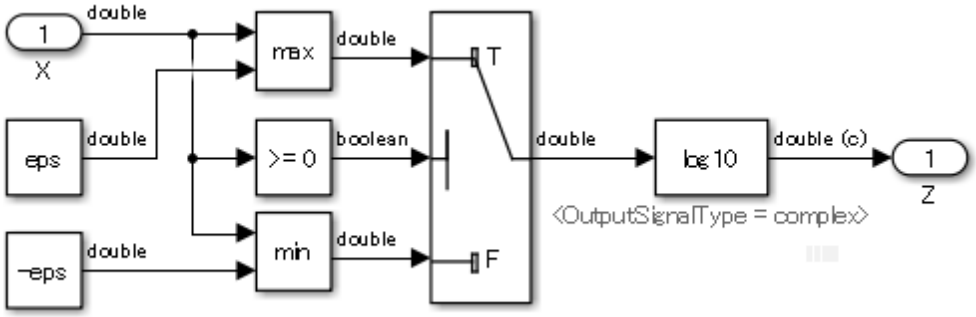
【誤】



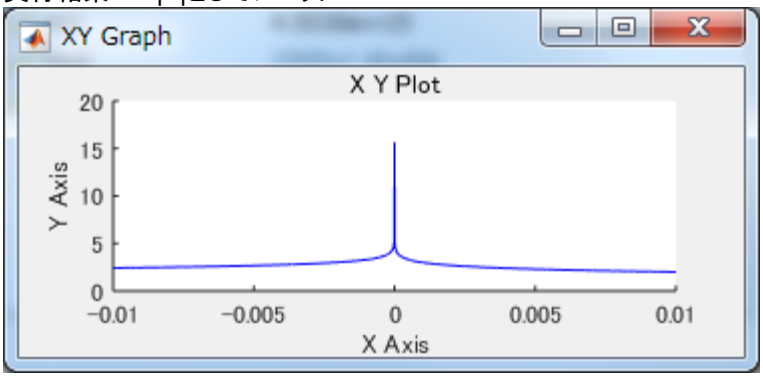
e	[Math Function]の{関数}を”log”または”log10”に設定して使用する場合、ゼロを入力しません。	-

【正】

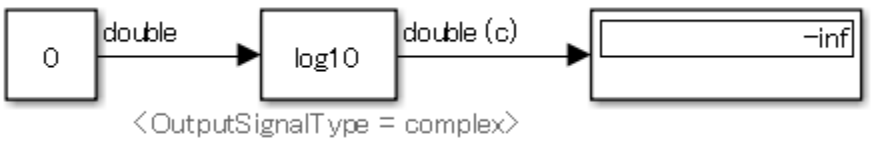
±eps 以内を±eps に置き換え



実行結果：Y=|Z|としてプロット

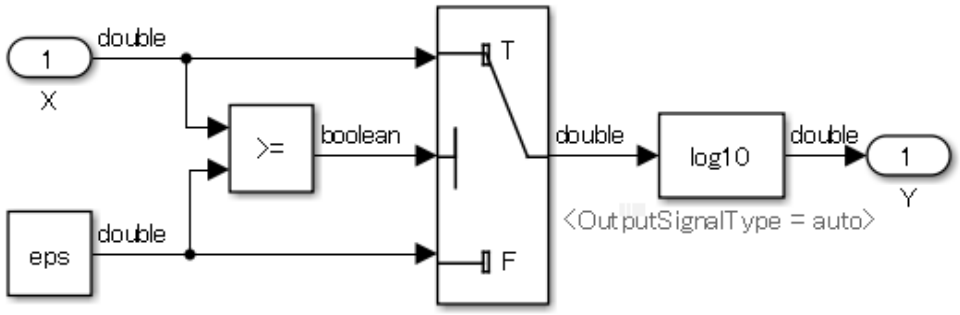
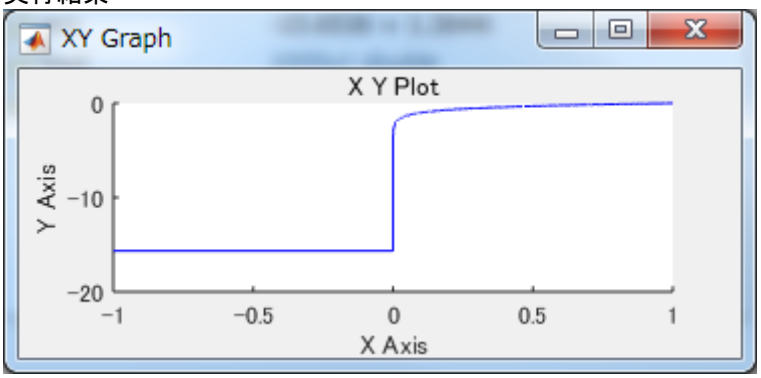
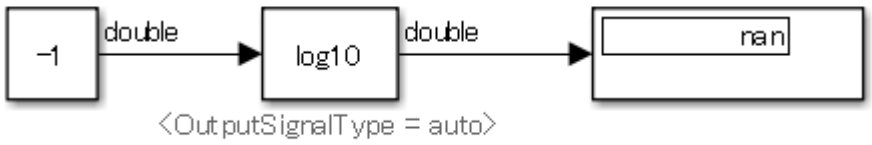
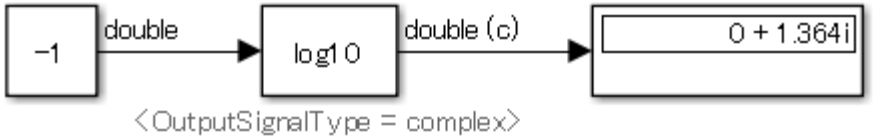
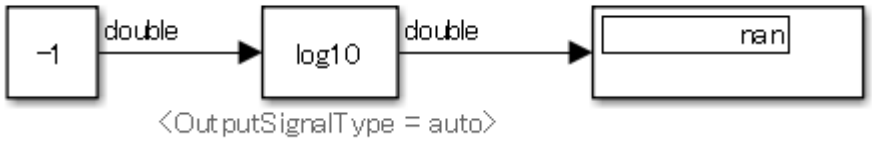


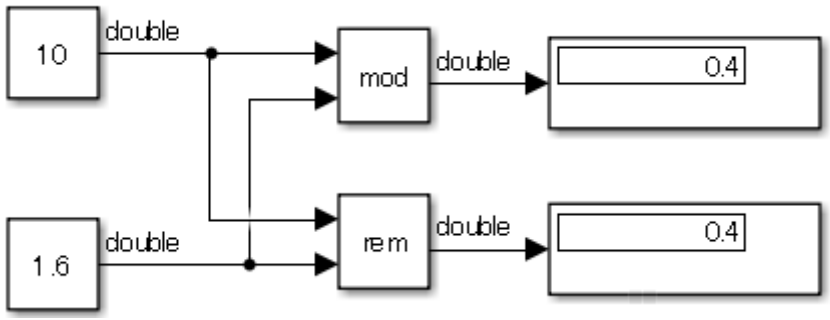
【誤】



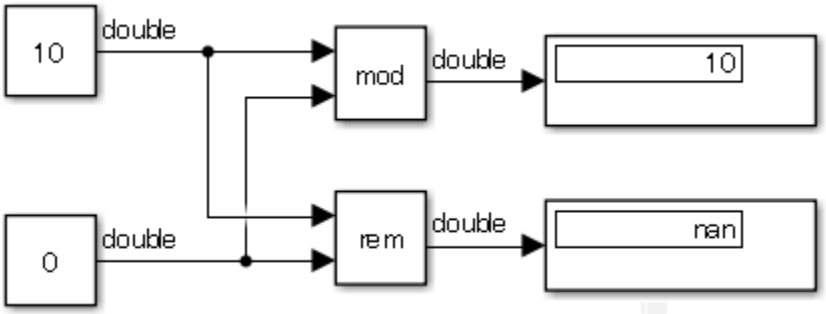
f1	[Math Function]の{関数}を”log”または”log10”に設定して使用する場合、負の数を入力しません。	-

【正】

	eps 未満を eps で飽和  実行結果  【誤】  	
f2	[Math Function]の{関数}を”log”または”log10”に設定して使用する場合、{出力信号タイプ}を”複素数”に設定します。	-
	【正】  【誤】  	
g	[Math Function]の{関数}を”mod”または”rem”に設定して使用する場合、第二引数にゼロを入力しません。	-
	【正】	



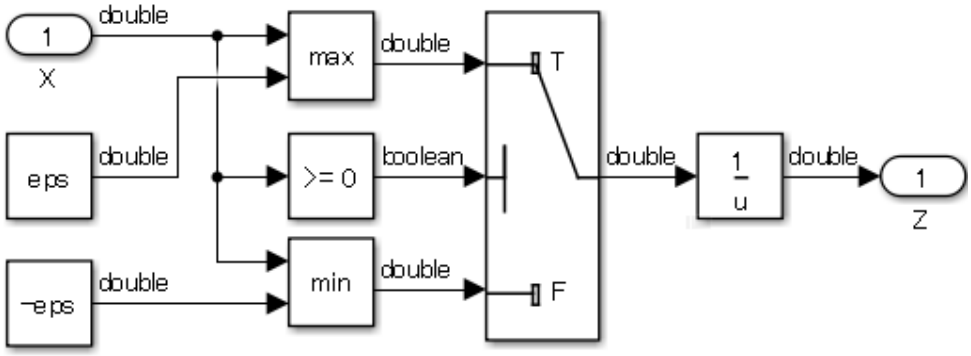
【誤】



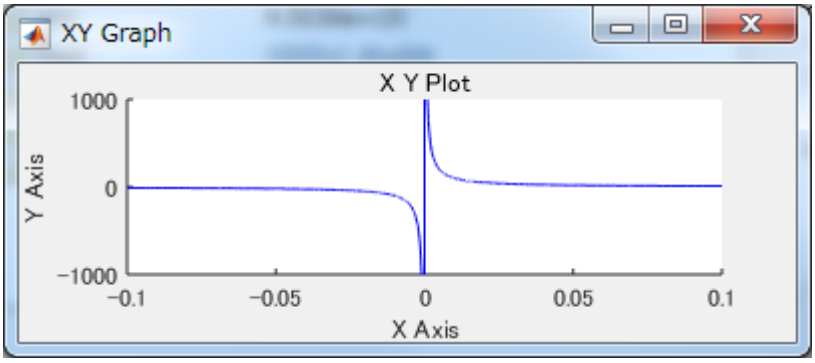
h	[Math Function]の{関数}を”reciprocal”に設定して使用する場合、ゼロを入力しません。	-

【正】

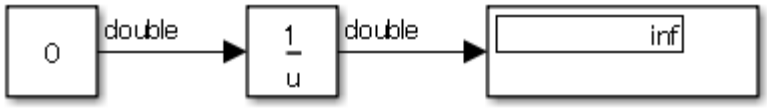
±eps 以内を±eps に置き換え



実行結果：演算結果は inf にはならないが、ゼロ近傍で出力が大きく変化

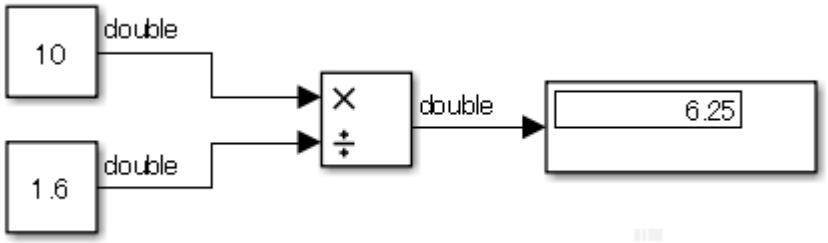


【誤】

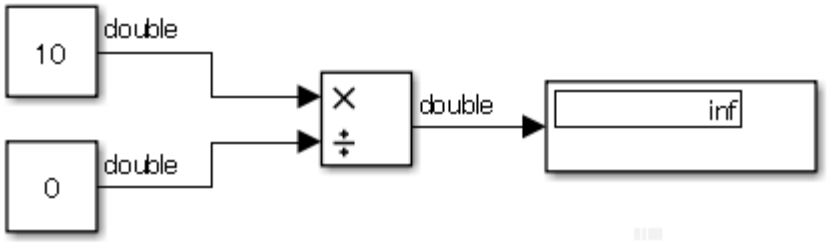


i	[Product]の{乗算}を”要素単位 (.*)”に設定して使用する場合、約数入力({入力数}で”/”を設定した入力ポート)にゼロを入力しません。	-

【正】

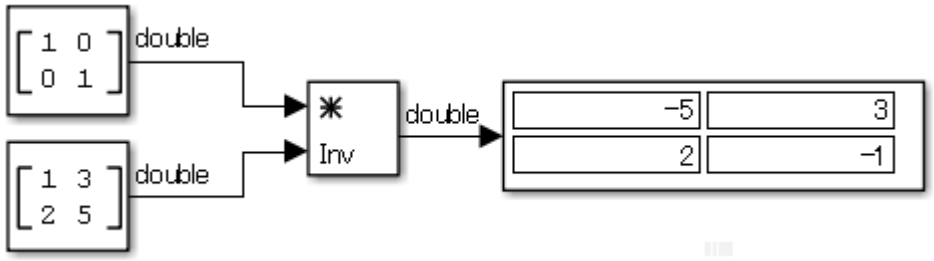


【誤】

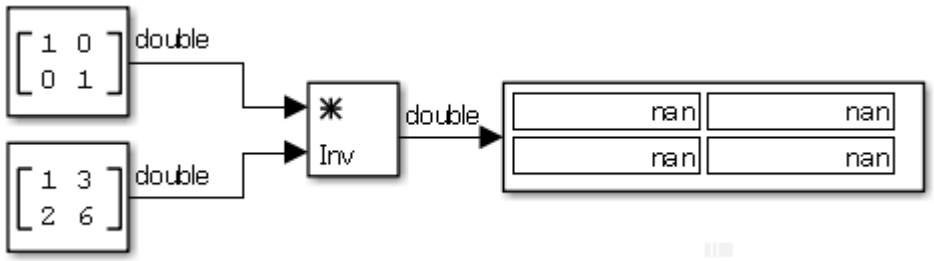


j	[Product]の{乗算}を”行列(*)”に設定して使用する場合、約数入力({入力数}で”/”を設定した入力ポート)に特異値行列を入力しません。	-

【正】



【誤】



根拠

サブ ID	記述内容
a1c1def1ghij	・不正な値を入力すると結果が処理系依存になるため、意図しない動作となる可能性があります。
a2	・設定により、不正な値が入力された時の意図しない動作を防止します。
b	・コード生成時に最適化され、コードまでトレースできないブロックとなる可能性があります。

c2f2	・設定により、負の数が不正な入力値がなくなるため、意図しない動作を防止できます。

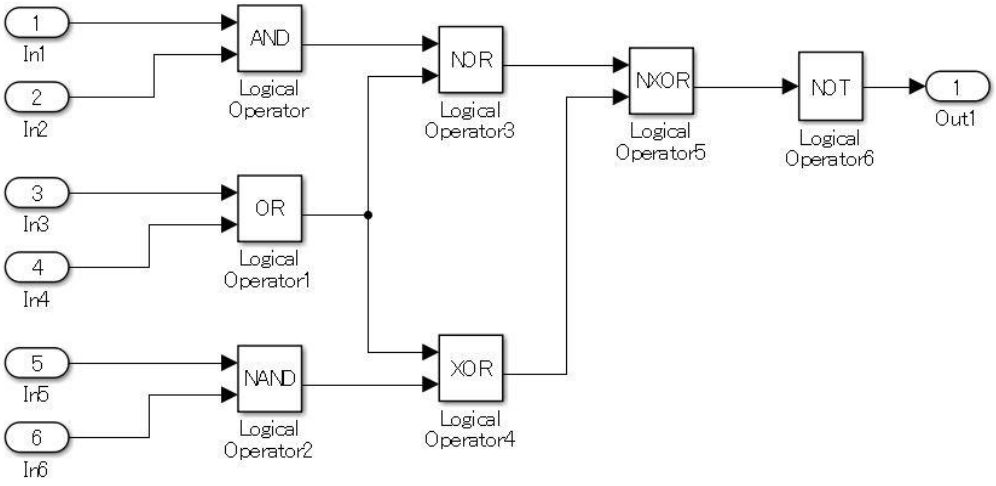
3.6.7. jc_0622：Fcn ブロックの使用方法

ルール ID：タイトル	jc_0622：Fcn ブロックの使用方法	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	[Fcn]の演算にて優先順位が異なる演算子が混在する場合は、括弧を付け優先順位を明示します。	-
根拠		
サブ ID	**記述内容**	
a	・優先順位が異なる演算子を使用する場合、括弧を用いて計算順序を明確化しないと可読性が落ちる、もしくは間違って解釈して演算を間違える可能性があります。	

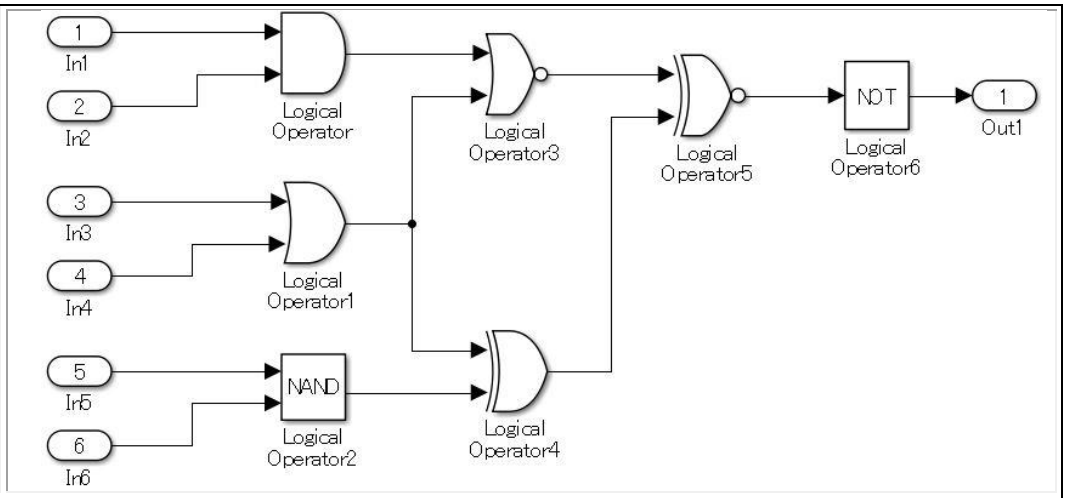
3.6.8. jc_0621：論理演算ブロックのアイコン形状

ルール ID：タイトル	jc_0621：論理演算ブロックのアイコン形状	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	[Logical Operator]の{アイコン形状}は、”四角形”にします。	-

【正】
アイコン形状が全て”四角形”で統一されています。



【誤】
アイコン形状が”四角形”で統一されていません

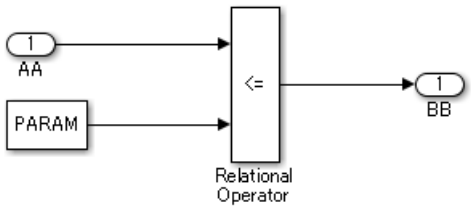


根拠	
サブ ID	**記述内容**
a	・同一の機能を複数の表現で記述できる場合には、表現を統一することで可読性を向上させることができます。 また、”特徴”の場合、それぞれの形状が似ているため、見間違うリスクが”四角形”に比べて増加します。

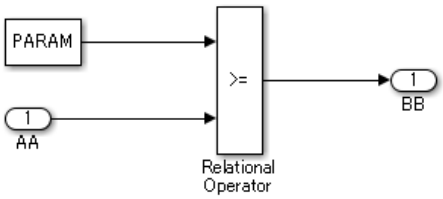
3.6.9. jc_0131：Relational Operator の使用方法

ルール ID：タイトル	**jc_0131：Relational Operator の使用方法**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	[Relational Operator]を信号と定数値の比較に使用する場合は、定数値を第二入力とします。	-

【正】



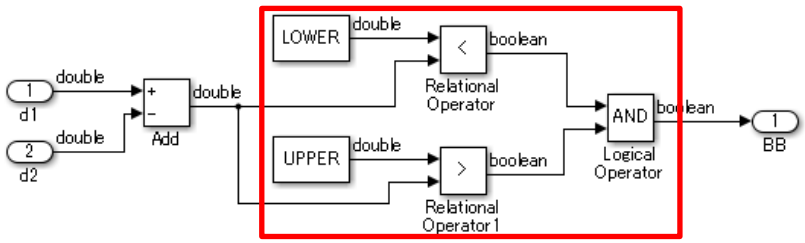
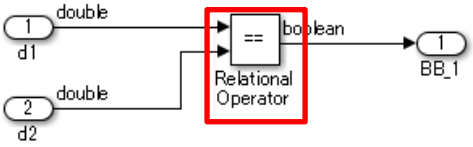
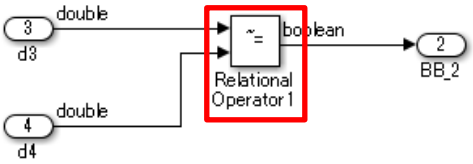
【誤】



根拠	
サブ ID	**記述内容**
a	・定数値との比較方法を統一することで、モデルの解釈ミスが低減します。

3.6.10. jc_0800：Simulink における浮動小数点型の比較

ルール ID：タイトル	**jc_0800：Simulink における浮動小数点型の比較**
ルール	

サブ ID	記述内容	カスタムパラメーター
a	浮動小数点型の等価比較演算==、~=は使用しません。	-
	【正】  【誤】 浮動小数点型の等価比較演算==、~=を使用しています。   	

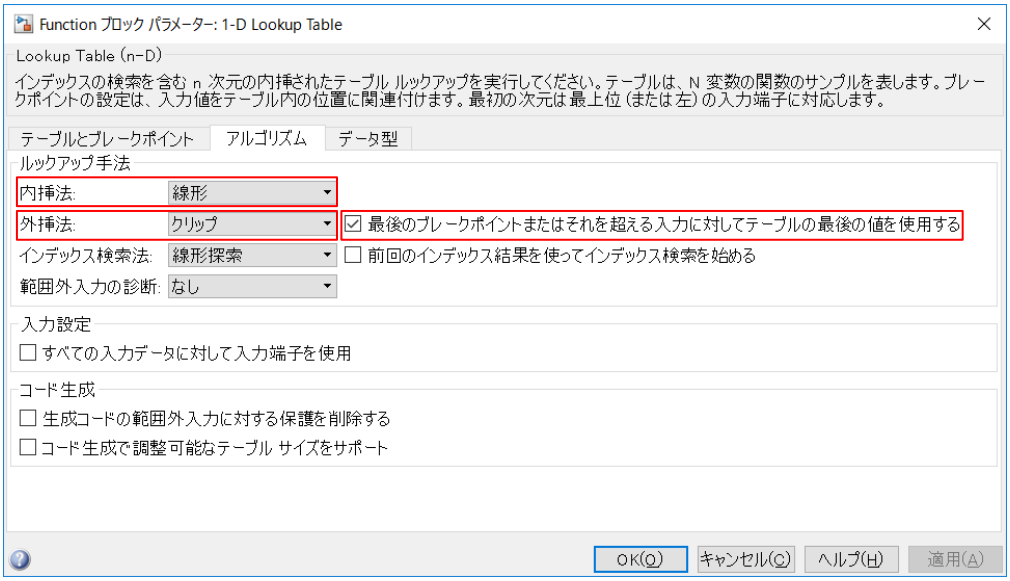
根拠	
サブ ID	**記述内容**
a	・浮動小数点の性質上、値に誤差を含むため、真を期待した等価比較結果が偽となる可能性があります。

3.6.11. jc_0626：Lookup Table 系ブロックの使用方法

ルール ID：タイトル	jc_0626：Lookup Table 系ブロックの使用方法	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	{ルックアップ手法}は"内挿 - 最後の値を使用"に設定します。	-
	【正】	

b	{内挿法}は"線形"、{外挿法}は"クリップ"に設定し、{最後のブレークポイントまたはそれを超える入力に対してテーブルの最後の値を使用する}にチェックを入れます。	-

【正】

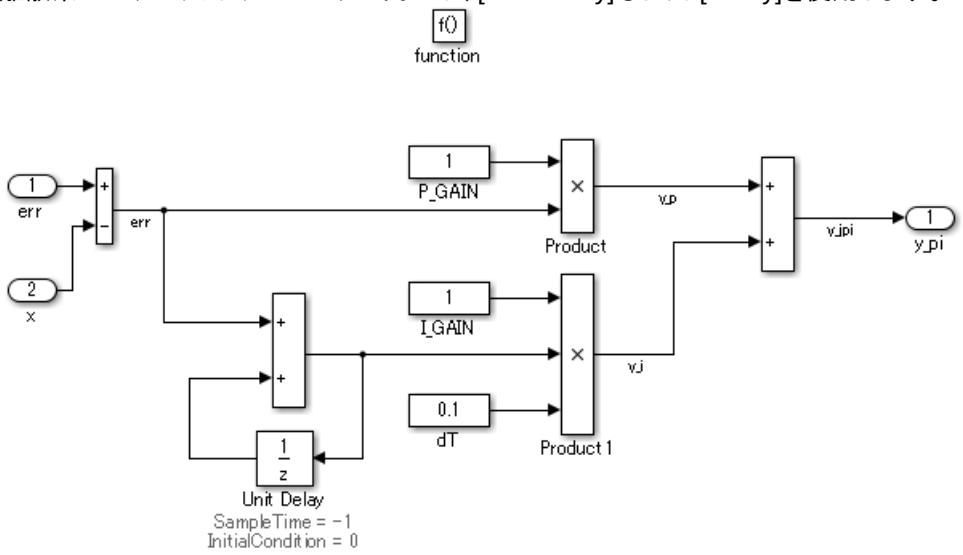
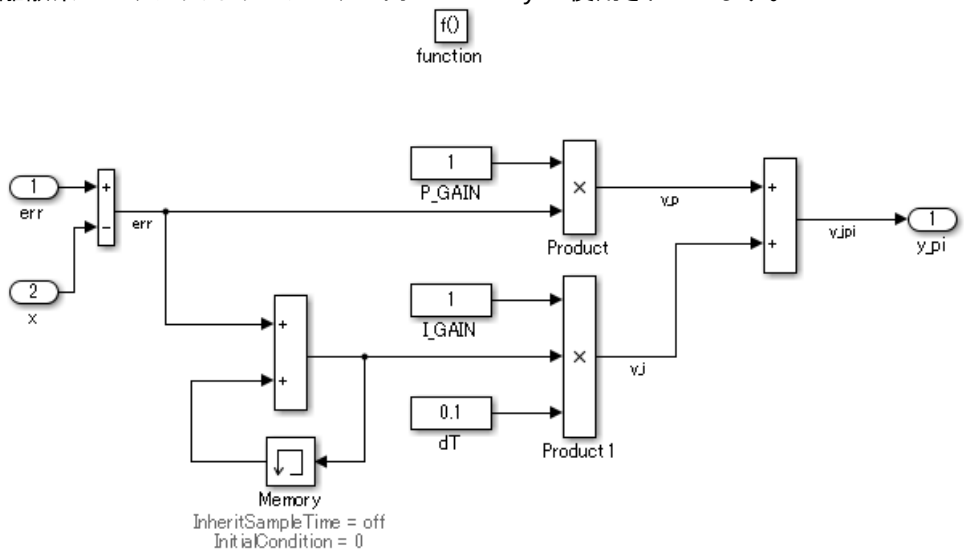


根拠

サブ ID	記述内容
ab	・Lookup Table 系ブロックに想定外の値が入力された場合、外挿法によって出力値を推測すると、実際には取り得ない値になったり、Lookup Table 系ブロックの出力がオーバーフローしたりする可能性があります。

3.6.12. jc_0623：連続系遅延ブロックと離散系遅延ブロックの使い分け

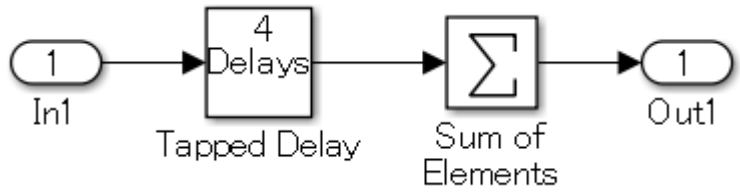
ルール ID：タイトル	jc_0623：連続系遅延ブロックと離散系遅延ブロックの使い分け	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	離散系のモデル又はサブシステム内では、[Unit Delay]もしくは[Delay]を使用します。	-

	連続系のモデル又はサブシステム内では、[Memory]を使用します。	
	【正】 離散系のモデル又はサブシステム内では、[Unit Delay]もしくは[Delay]を使用します。  【誤】 離散系のモデル又はサブシステム内で Memory が使用されています。  	

根拠	
サブ ID	**記述内容**
a	・使用方法を揃えることで可読性を向上します。

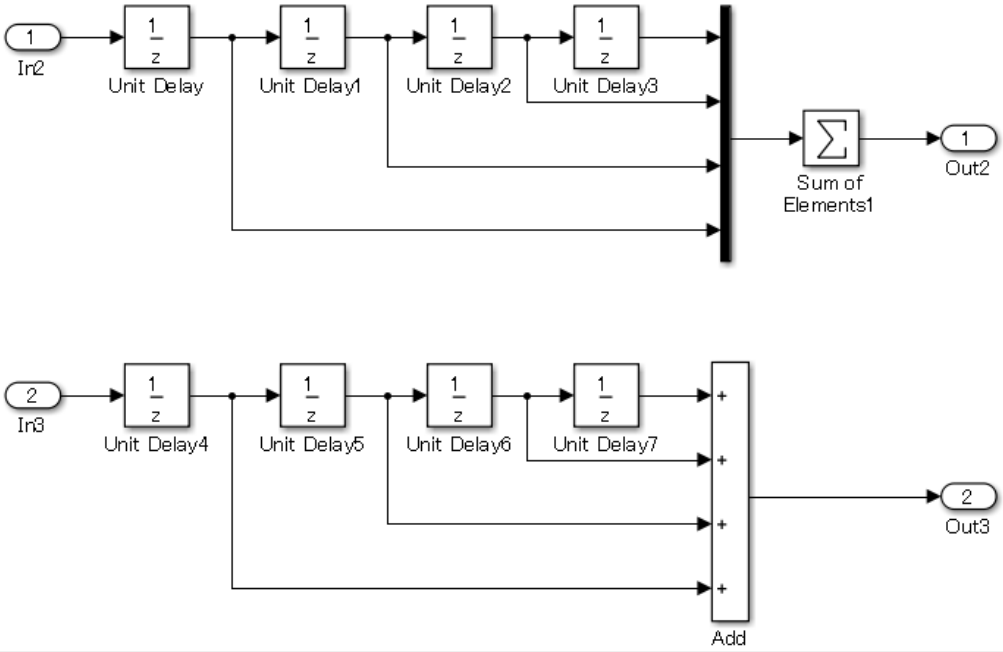
3.6.13. jc_0624：Tapped Delay ブロック / Delay ブロックの使用方法

ルール ID：タイトル	jc_0624：Tapped Delay ブロック / Delay ブロックの使用方法	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	複数回前の過去値を保持する場合、保持している全ての値をベクトル信号として生成するには[Tapped Delay]を使用します。	-
	【正】 [Tapped Delay]を使用しています。	



【誤】

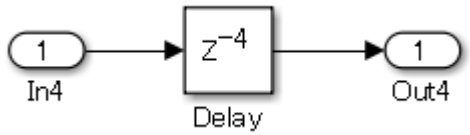
[Tapped Delay]を使用していません。



b	複数回前の過去値を保持する場合、保持している最も古い信号の値のみを取得するには[Delay]を使用します。	-

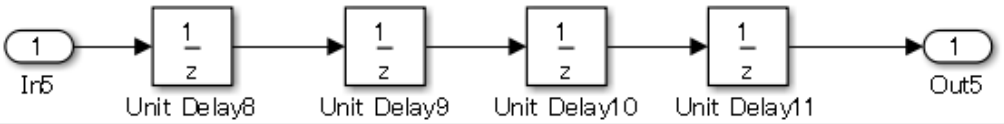
【正】

[Delay]を使用しています。



【誤】

[Delay]を使用していません。

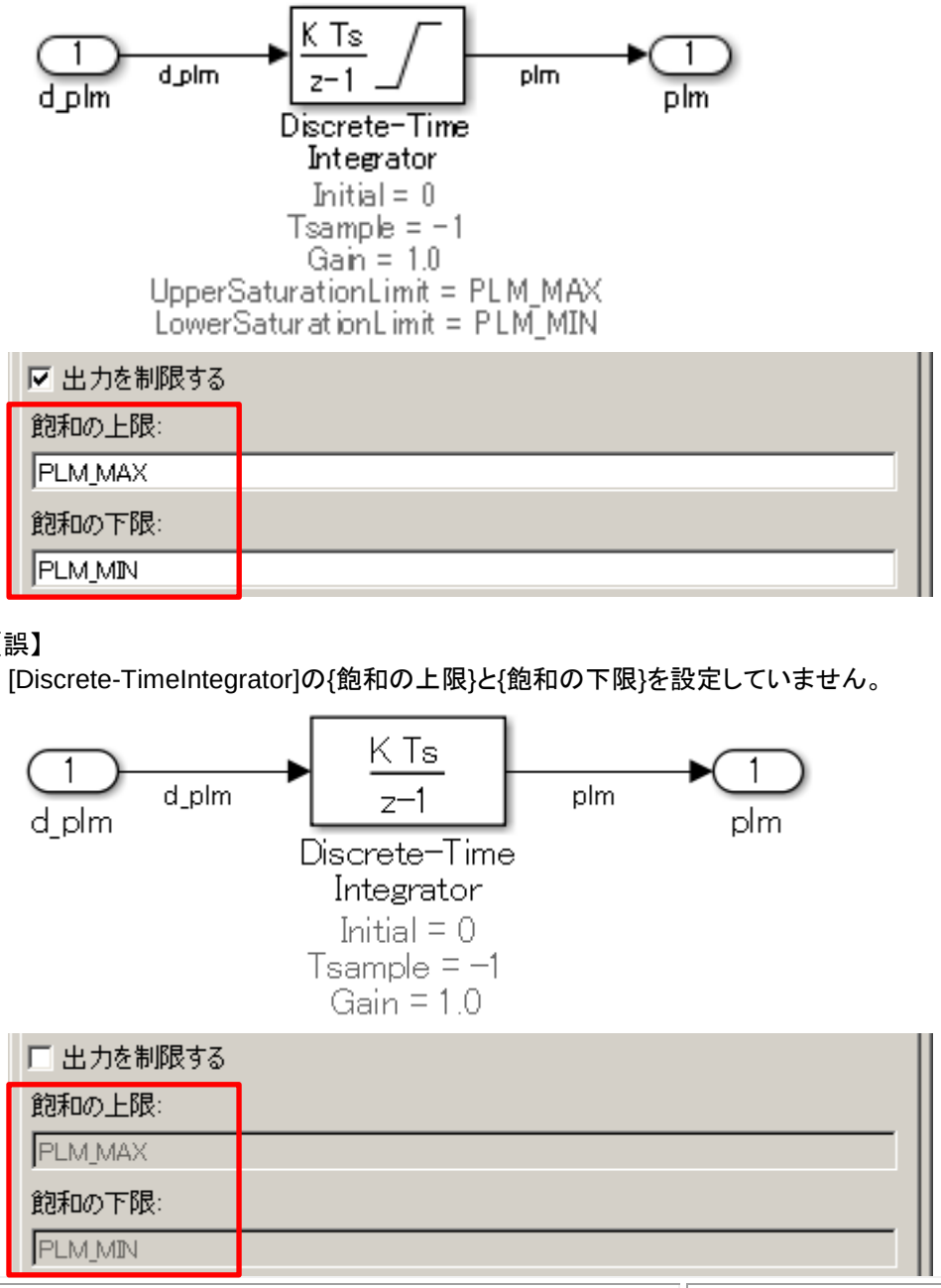


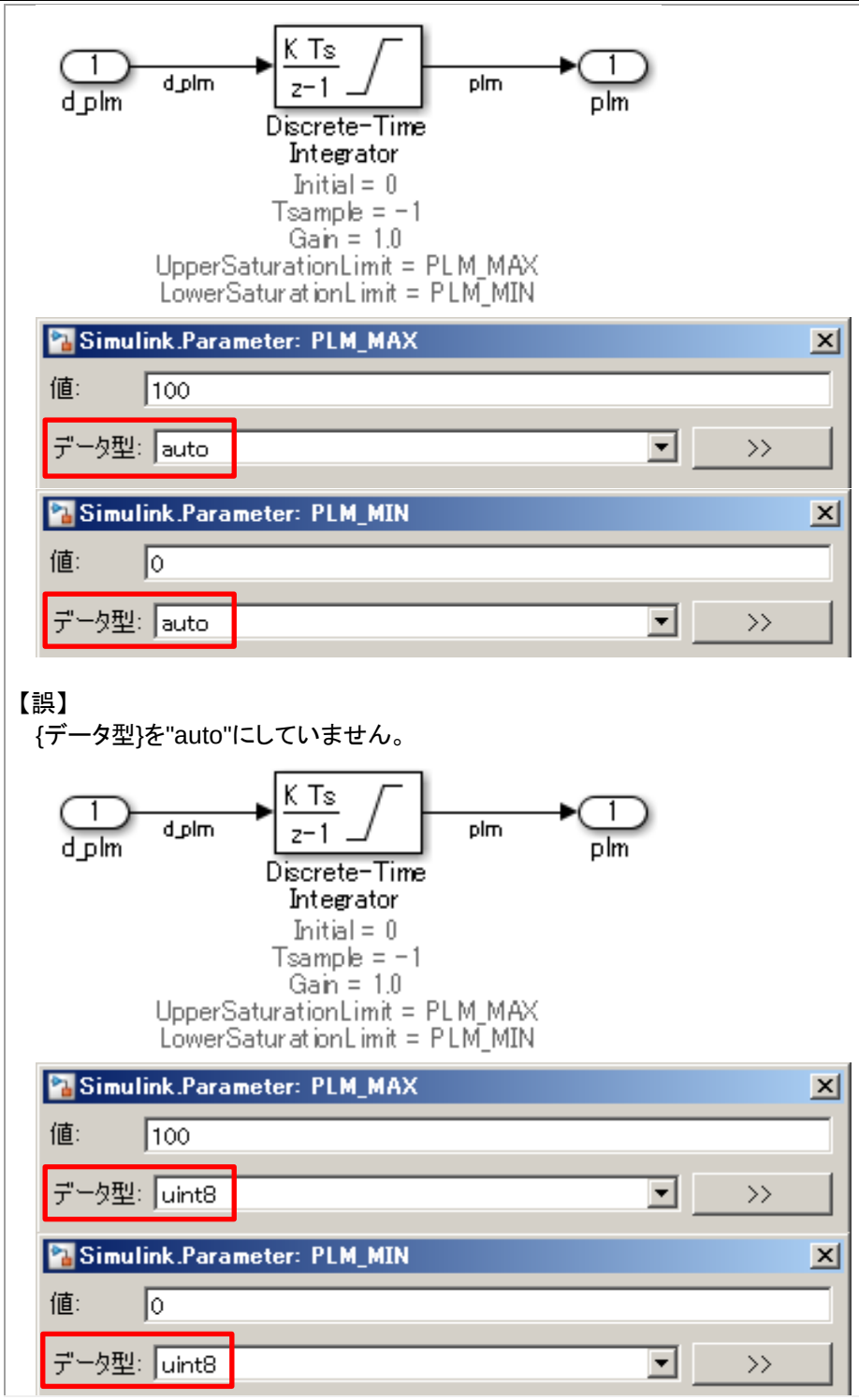
根拠

サブ ID	記述内容
a	・Tapped Delay ブロックは、過去の値を持つ配列が設定され、コードの可読性が向上し、コード効率面でも有利になります。
b	・モデルの可読性が向上し、コード効率面でも有利になります。

3.6.14. jc_0627 : Discrete-TimeIntegrator ブロックの使用方法

ルール ID : タイトル	jc_0627 : Discrete-TimeIntegrator ブロックの使用方法	
ルール		
サブ ID	記述内容	カスタムパラメーター

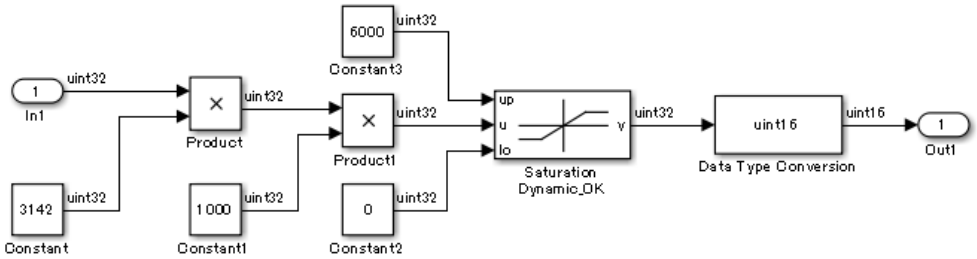
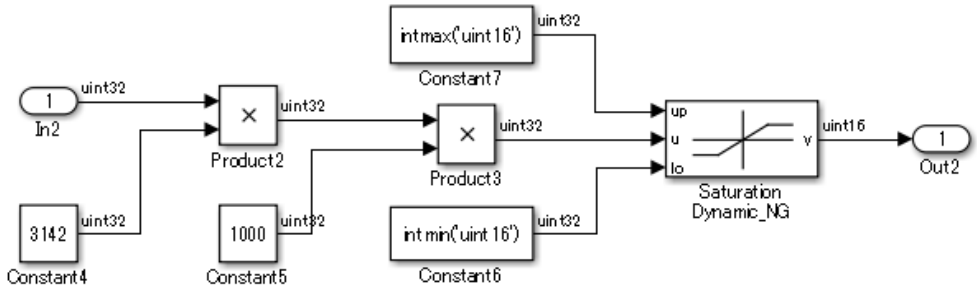
a	[Discrete-TimeIntegrator]の{飽和の上限}と{飽和の下限}を設定します。	-
	【正】 [Discrete-TimeIntegrator]の{飽和の上限}と{飽和の下限}を設定しています。  【誤】 [Discrete-TimeIntegrator]の{飽和の上限}と{飽和の下限}を設定していません。 	
b	[Discrete-Time Integrator]の{飽和の上限}と{飽和の下限}を mpt.Parameter 等で設定する場合、そのパラメーターのコード生成用の設定は、{データ型}を"auto"にします。	-
	【正】 {データ型}を"auto"にしています。	



根拠	
サブ ID	**記述内容**
a	・ブロックの出力がオーバーフローするのを避け、このブロックの出力を使う他の演算ブロックが想定外の結果になることを防止します。
b	・auto、single、double 以外のデータ型を設定するとモデルコンパイル(シミュレーション)時にエラーが発生します。

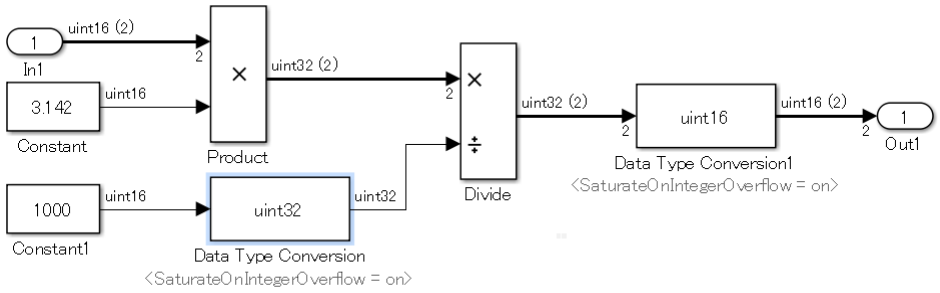
3.6.15. jc_0628 : Saturation ブロックの使用方法

ルール ID : タイトル	**jc_0628 : Saturation ブロックの使用方法**
ルール	

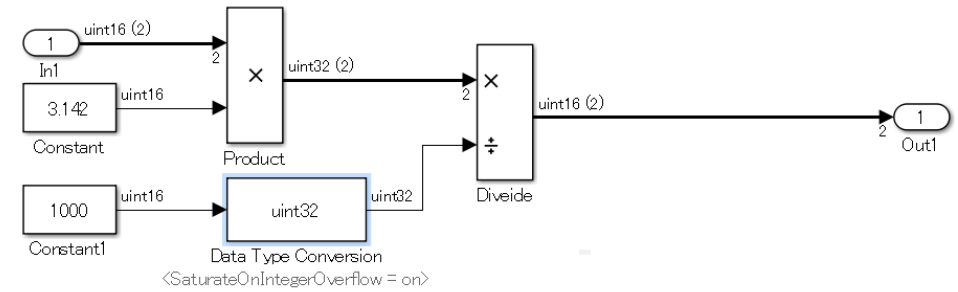
サブ ID	記述内容	カスタムパラメーター
a	[Saturation]または[Saturation Dynamic]は、物理量を制限する目的で使用します。型変換には使用しません。{上限}、{下限} にデータ型の最大・最小値を設定しません。	-
	【正】 [Saturation Dynamic]は、物理量を制限する目的で使用されています。型変換には使用されていません。  【誤】 [Saturation Dynamic]は、物理量を制限する目的で使用されていません。型変換に使用されています。{上限}、{下限} にデータ型の最大・最小値が設定されています。  	
根拠		
サブ ID	**記述内容**	
a	・[Saturation]の使用方法を統一することでモデルの保守性が良くなります。	

3.6.16. jc_0651 :型変換を実施する場合の使用方法

ルール ID : タイトル	**jc_0651 :型変換を実施する場合の使用方法**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	ブロックの出力信号のデータ型を、初期設定で決まる型以外に変える場合は[Data Type Conversion]を使用します。	-
	【正】 数値演算ブロック[Divide]の出力信号のデータ型変換に[Data Type Conversion]が使用されています。	



【誤】
　数値演算ブロック[Divide]の出力信号のデータ型変換に[Data Type Conversion]が使用されていません。



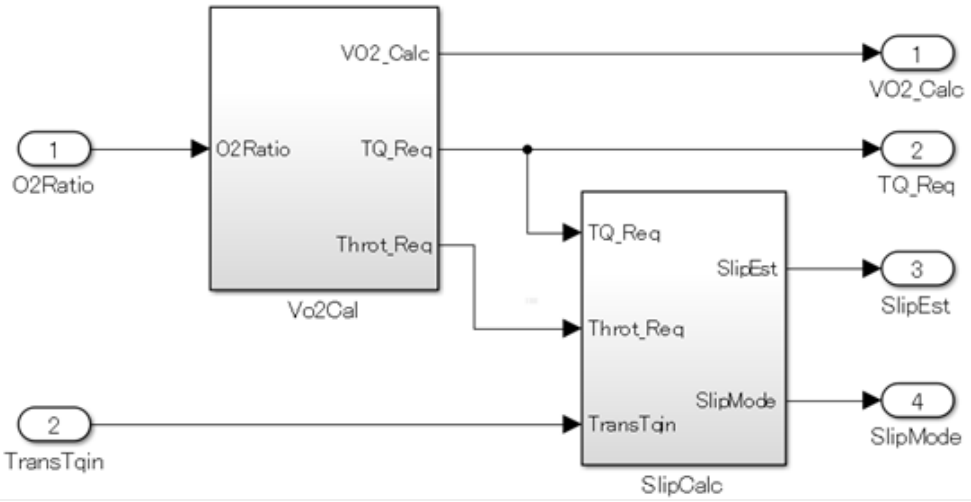
根拠	
サブ ID	**記述内容**
a	・演算と型変換を分ける事によって、実行順序とどの演算までがどのデータタイプを使用するかがブロックの構成上で明確化できます。

3.7. その他のブロック

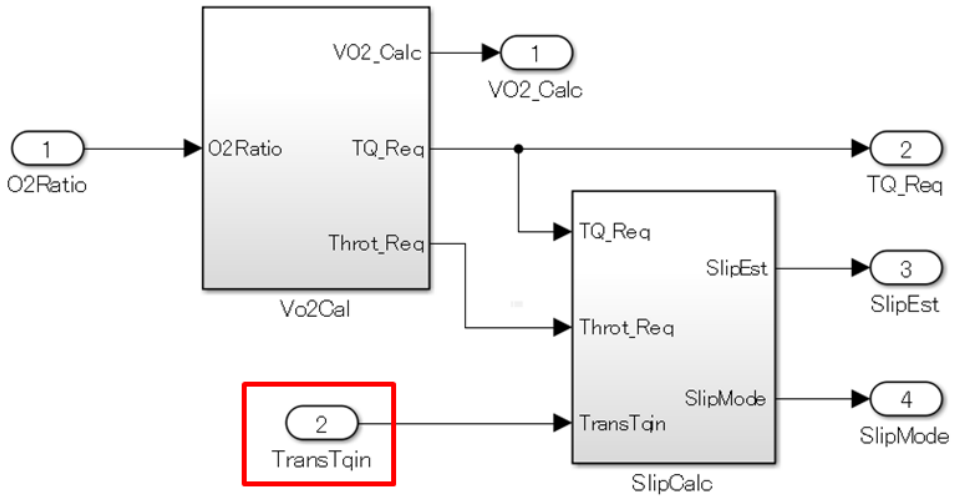
3.7.1. db_0042：Inport ブロック / Outport ブロックの使用方法

ルール ID：タイトル	**db_0042：Inport ブロック / Outport ブロックの使用方法**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	[Inport]はダイアグラムの左側に配置します(交差する場合は移動可能)	-

【正】
　[Inport]はダイアグラムの左側に配置されています。

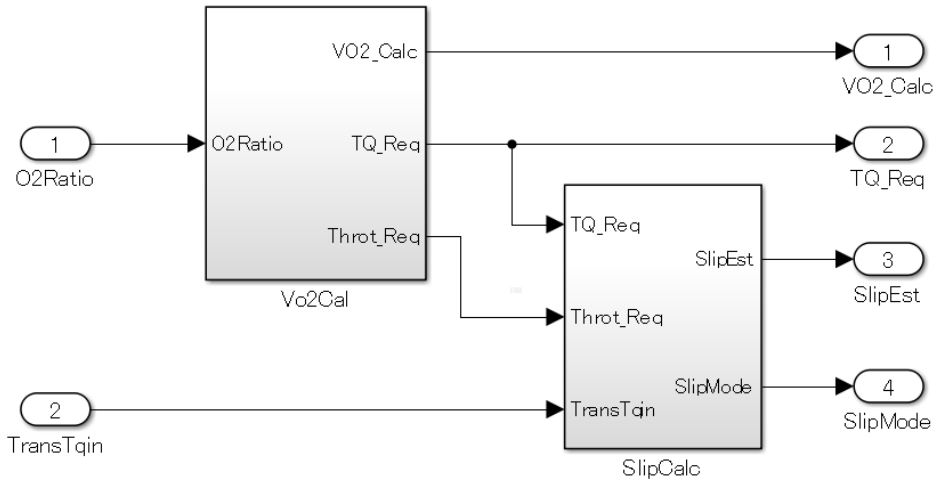


【誤】
[Inport] はダイアグラムの左側に配置されていません。

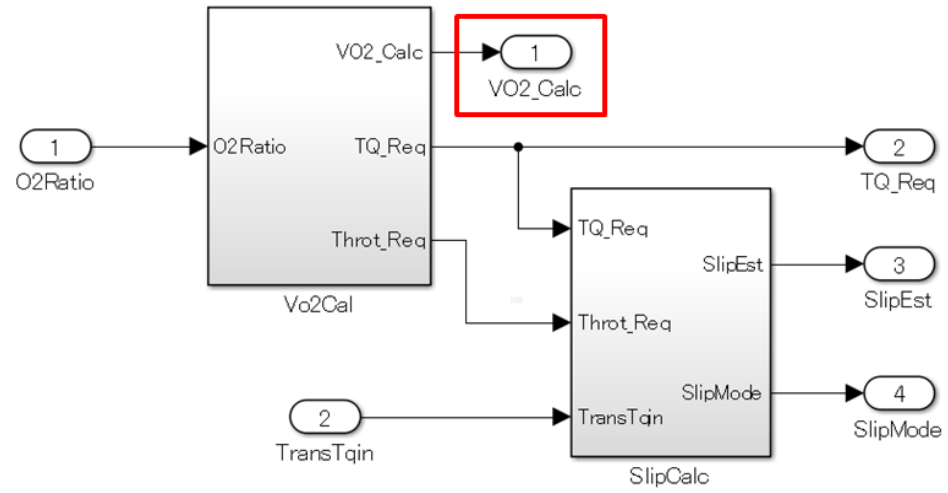


b	[Outport]はダイアグラムの右側に配置します(交差する場合は移動可能)	-

【正】
[Outport]はダイアグラムの右側に配置されています。

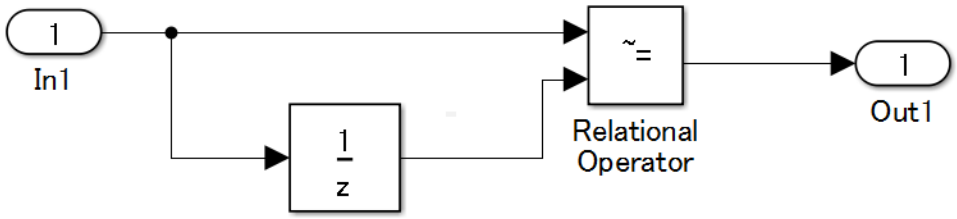


【誤】
[Outport] はダイアグラムの右側に配置されていません。

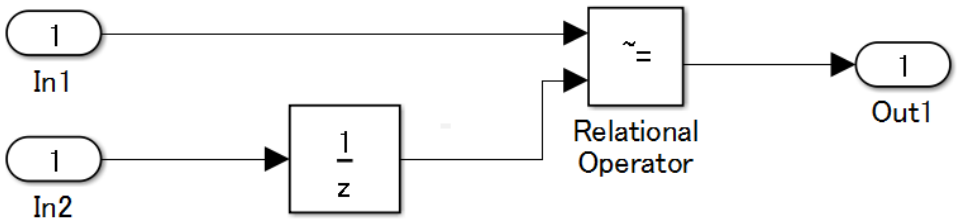


c	重複するインポートの作成は禁止します。	-

【正】
重複するインポートは使われていません。



【誤】
重複するインポートが使われています。

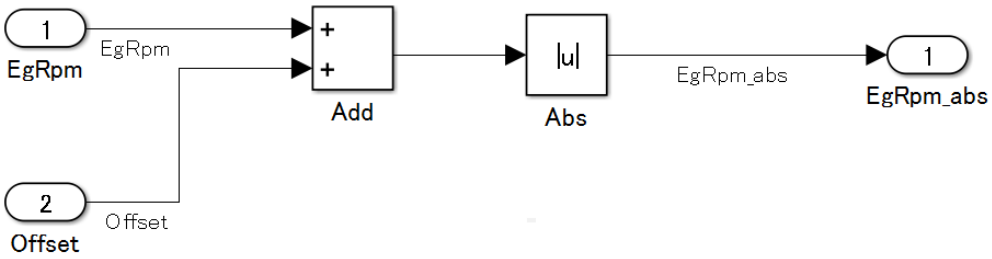


根拠	
サブ ID	**記述内容**
abc	・運用ルールを決めることにより可読性が向上します。

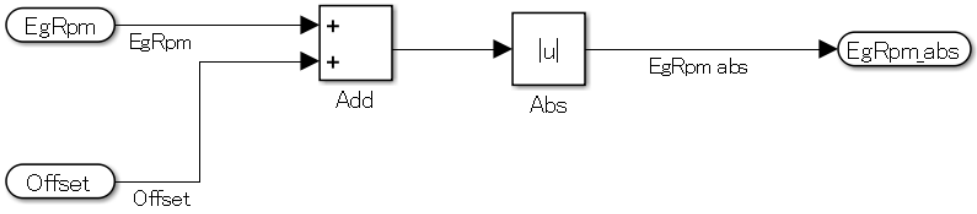
3.7.2. jc_0081 ： Inport ブロック / Outport ブロックのアイコン表示

ルール ID : タイトル	**jc_0081 ： Inport ブロック / Outport ブロックのアイコン表示**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	[Inport]、[Outport]の{アイコン表示}は”端子番号”にします。	-
	【正】	

[Inport]、[Outport]の{アイコン表示}が"端子番号"です。



【誤】
[Inport]、[Outport]の{アイコン表示}が"端子番号"ではありません。



根拠	
サブ ID	**記述内容**
a	・サブシステム外部で[Inport]または[Outport]の順番を、端子番号で確認するため、また、[Inport]または[Outport]にブロック名を表示するために、[Inport]、[Outport]の{アイコン表示}は”端子番号”にします。 ・ブロック名を表示し、[Inport]または[Outport]に接続される信号線の名前と、ブロック名を同一にする事で、階層化されたサブシステムにおいて、接続ミスを防止します。

3.7.3. na_0011：Goto / From の範囲

ルール ID：タイトル	**na_0011：Goto / From の範囲**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	[Goto]の{タグの可視性}は”ローカル”にします。	-
根拠		
サブ ID	**記述内容**	
a	・[Goto]と対応する[Form]が異なる階層にあると、接続関係が解りにくくなります。 ・[Goto]と対応する[Form]が異なる階層にあると、バーチャルサブシステムを Atomic サブシステムに変更したときに、シミュレーションエラーが発生する可能性があります。	

3.7.4. jc_0161：Data Store Memory ブロックの定義方法

ルール ID：タイトル	**jc_0161：Data Store Memory ブロックの定義方法**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	[Data Store Memory]は、使用する最小スコープレベルで定義します。	-

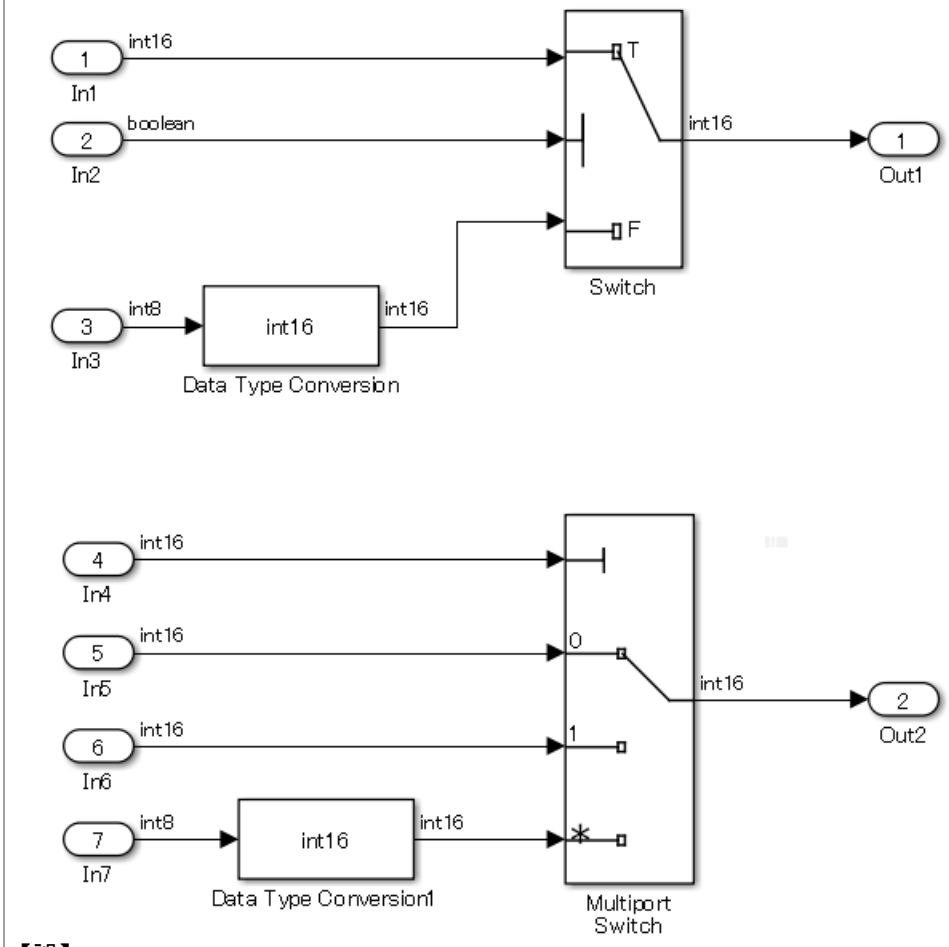
b	[Data Store Memory]には、実行、コード生成に必要となるデータのみを定義します。	-
根拠		
サブ ID	**記述内容**	
a	・使用箇所を限定することで、信頼性が向上します。	
b	・未使用の[Data Store Memory](データ)が存在すると保守性、運用性に影響がある可能性があります。	

3.7.5. jc_0141 : Switch ブロックの使用方法

ルール ID : タイトル	**jc_0141 : Switch ブロックの使用方法**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	[Switch]の 2 番目の入力(条件) は論理型を入力します。 [Switch]の{1 番目の入力が通過する条件} は “u2~=0 “にします。	-
根拠		
サブ ID	**記述内容**	
a	・ブロックの中に演算式を書くのではなく、Simulink ブロックで構成した方が、見た目で仕様を理解し易くなります。	

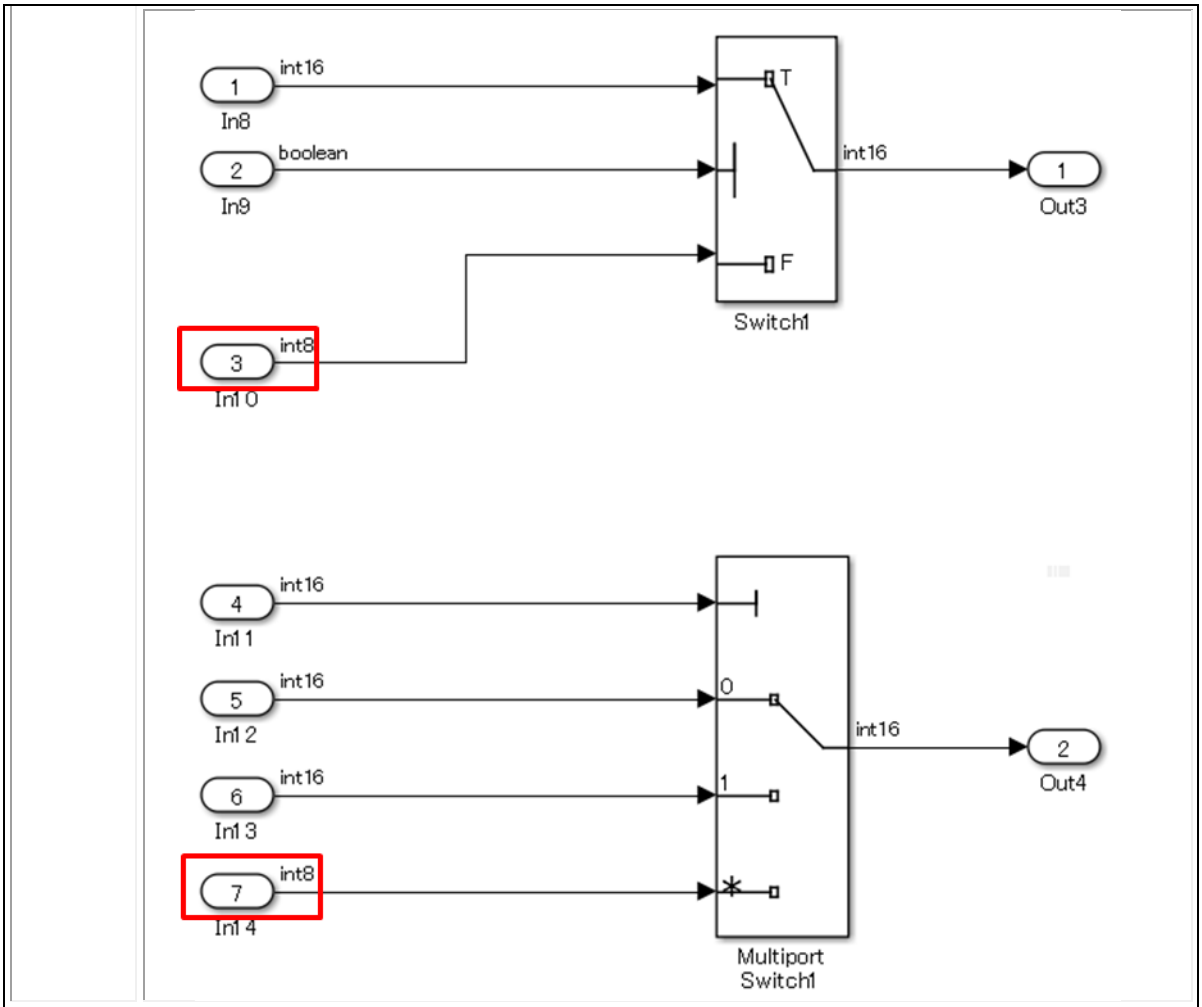
3.7.6. jc_0650 : 切替機能を持つブロックの入出力データ型

ルール ID : タイトル	**jc_0650 : 切替機能を持つブロックの入出力データ型**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	切替機能を持つブロック([Switch]、[Multiport Switch]、[Index Vector])は、データ端子と出力ポートのデータ型を同じにします。	-
	【正】 データ端子と出力端子のデータ型は同じです。	



【誤】

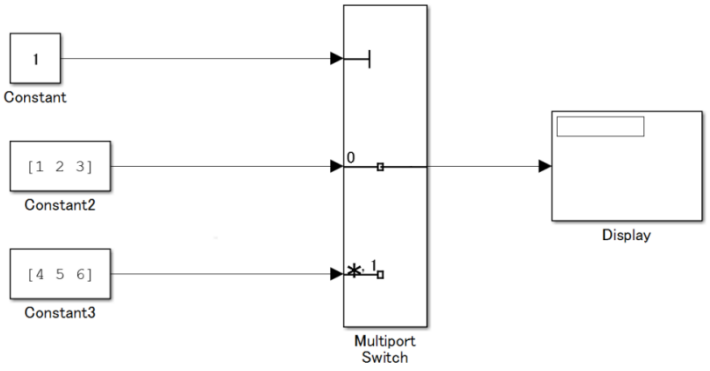
データ端子と出力端子のデータ型が異なる箇所があります。



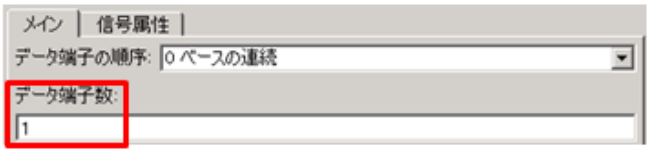
根拠	
サブ ID	**記述内容**
a	・暗黙の型変換を防止します。

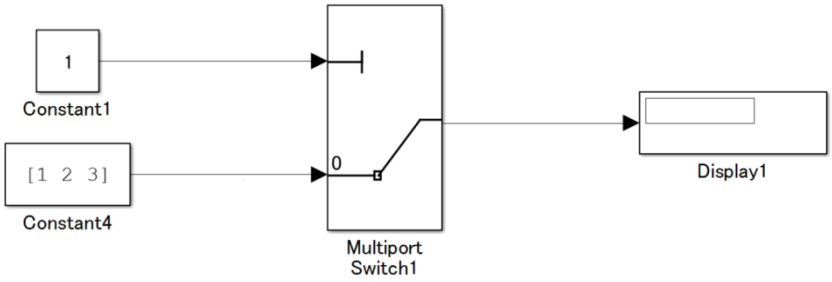
3.7.7. jc_0630：Multiport Switch ブロックの使用方法

ルール ID：タイトル	**jc_0630：Multiport Switch ブロックの使用方法**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	[Multiport Switch]の{データ端子数}は”2”以上にします。	-
	【正】 	



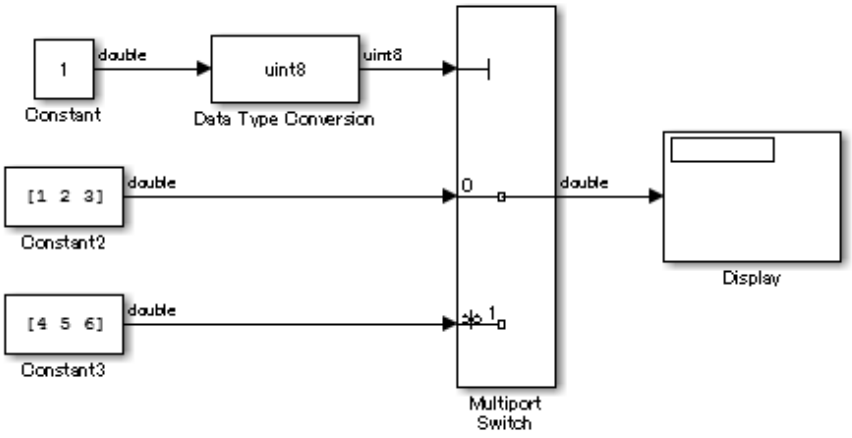
【誤】



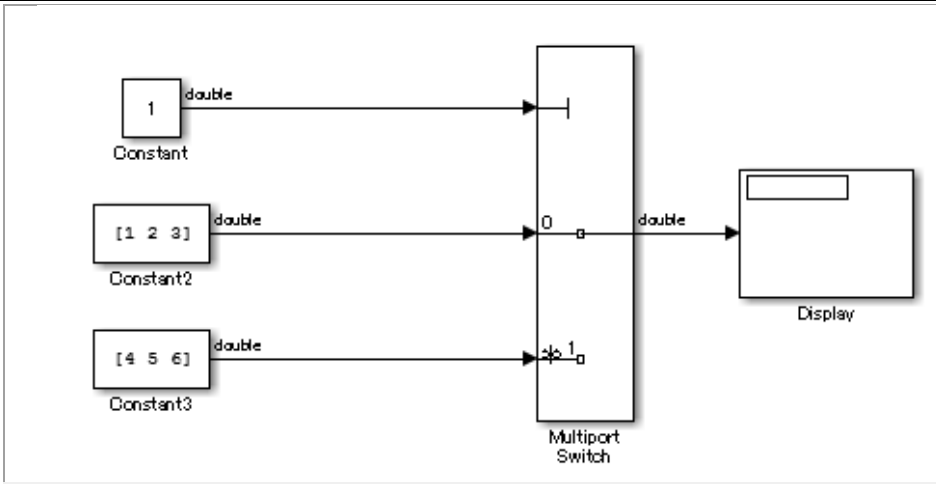


b	[Multiport Switch]の制御端子への入力は、符号なし整数にします。	-

【正】

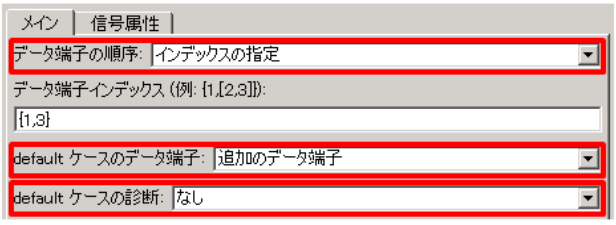


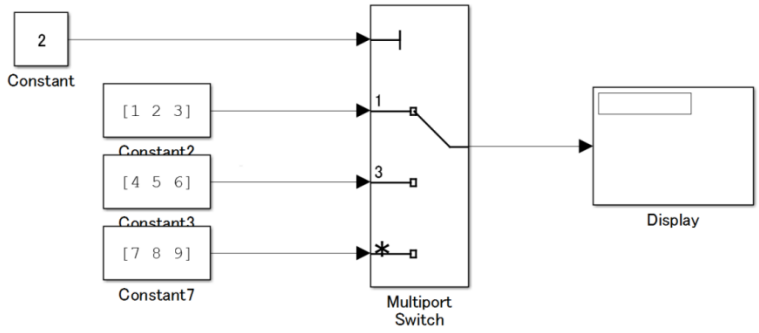
【誤】



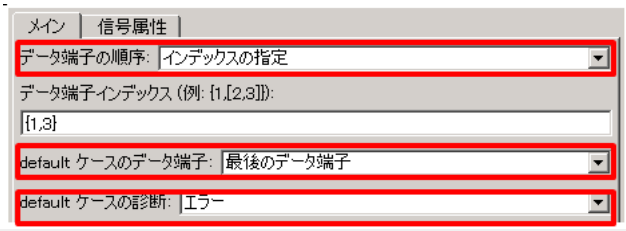
c	[Multiport Switch]の{データ端子の順序}を”インデックスの指定”とした場合は、下記の設定を行います。 ・{default ケースのデータ端子}を”追加のデータ端子”にします。 ・{default ケースの診断}を”なし”にします。	-

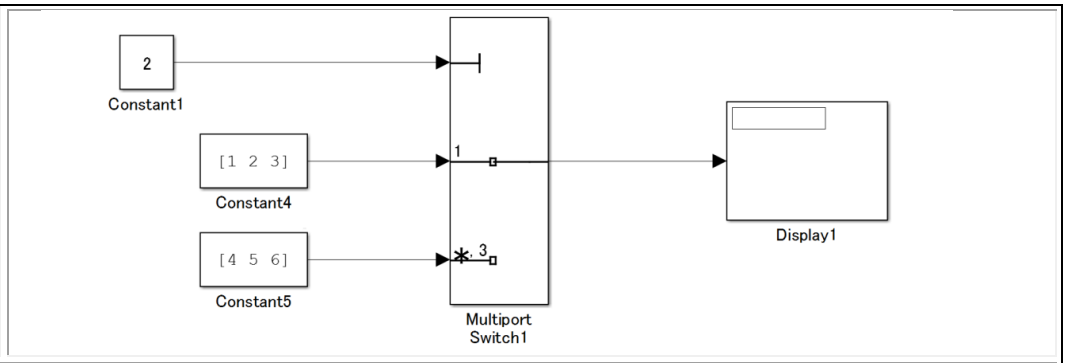
【正】





【誤】





根拠

サブ ID	記述内容
a	・データ端子数が1つの場合は、ベクトルの中からスカラーを取り出すブロックに変わるため、意図しない出力になる可能性があります。
b	・制御端子は 0 以上の整数値が期待する入力範囲のため、符号付き、もしくは、非整数の信号が制御端子に接続されると、接続ミスのように見えます。 ・実際に負の値、非整数値が入力されたときに、意図しないデータ端子が選択される可能性があります。
c	・“インデックスの指定”の場合、制御端子に指定したインデックス以外の値が入力されると、指定したインデックスの最後の値と同じ扱いとなります。そのため、意図しないデータ端子が選択される可能性があります。

4. Stateflow

4.1. Stateflow ブロック / データ / イベント

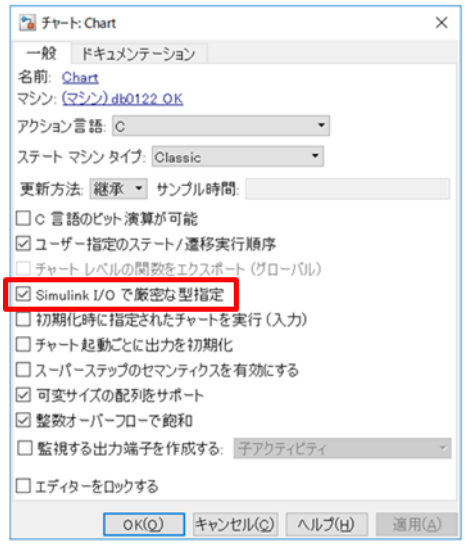
4.1.1. db_0122：Stateflow と Simulink の接続信号とパラメーター

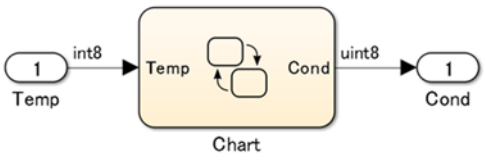
ルール ID：タイトル	db_0122：Stateflow と Simulink の接続信号とパラメーター

ルール

サブ ID	記述内容	カスタムパラメーター
a	[Chart]に入出力する Simulink の信号のデータ型を指定可能な設定にします。 そのために、{ファイル}-{モデル プロパティ}-{チャート プロパティ}の{SimulinkI/O で厳密な型指定}にチェックを入れます。	-

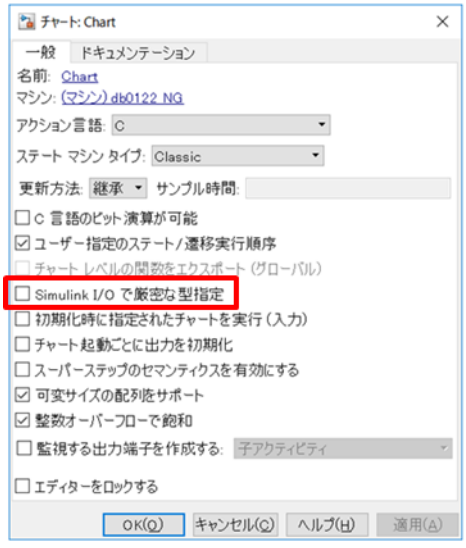
【正】

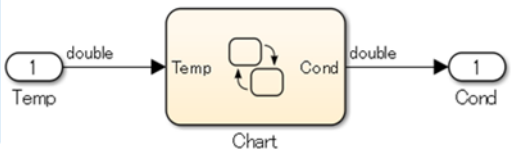




【誤】

SimulinkI/O で厳密な型指定をチェックしない場合、入出力は double 型に固定されます。





根拠

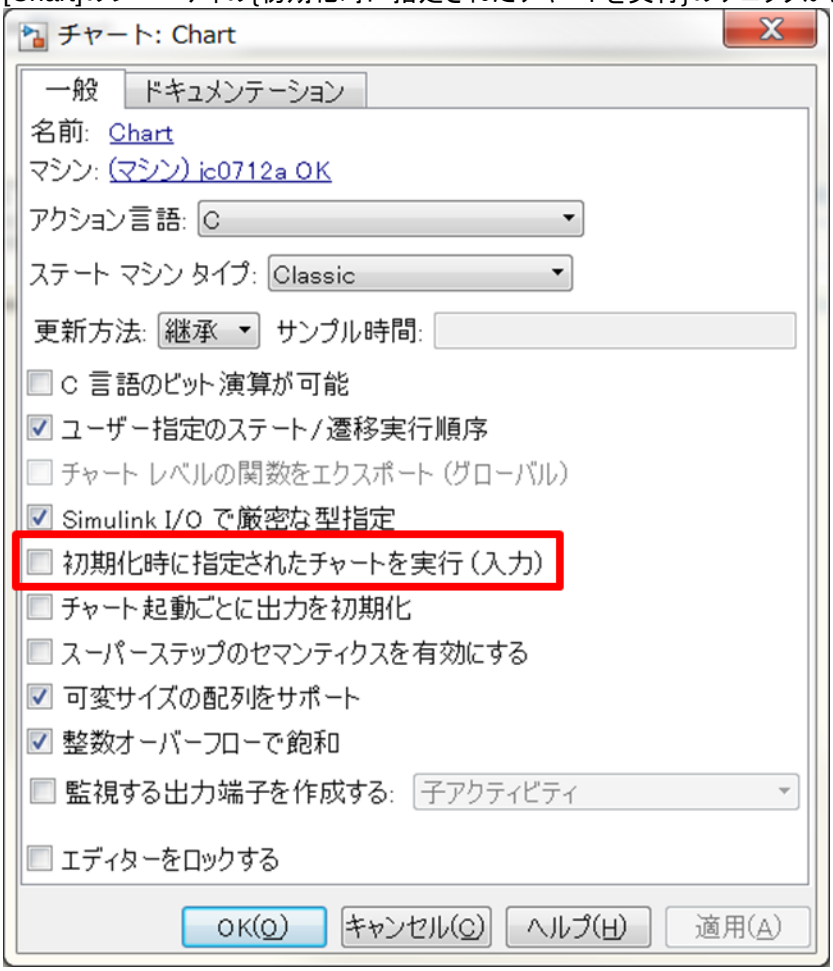
サブ ID	記述内容
a	・{SimulinkI/O で厳密な型指定}のチェックを入れていない場合、[Chart]に入出力できる Simulink の信号のデータ型は double 型のみです。

	それにより、[Chart]へ入出力する前後に型変換が必要となり、ブロック量が増加して可読性が低下します。 ・{SimulinkI/O で厳密な型指定}のチェックを入れていない場合、[Chart]に入力できる Simulink の信号のデータ型は double 型のみですが、その信号と直接接続する[Chart]内の入力データはどのデータ型でも問題ありません。 この 2 つの信号が異なるデータ型を持つ場合、暗黙的なデータ型変換が行われます。 {SimulinkI/O で厳密な型指定}のチェックを入れることで、暗黙的なデータ変換が行われずにデータ型不整合のエラーが出力されます。 それにより、データ型の違いによる誤解を防ぎ、可読性が向上します。

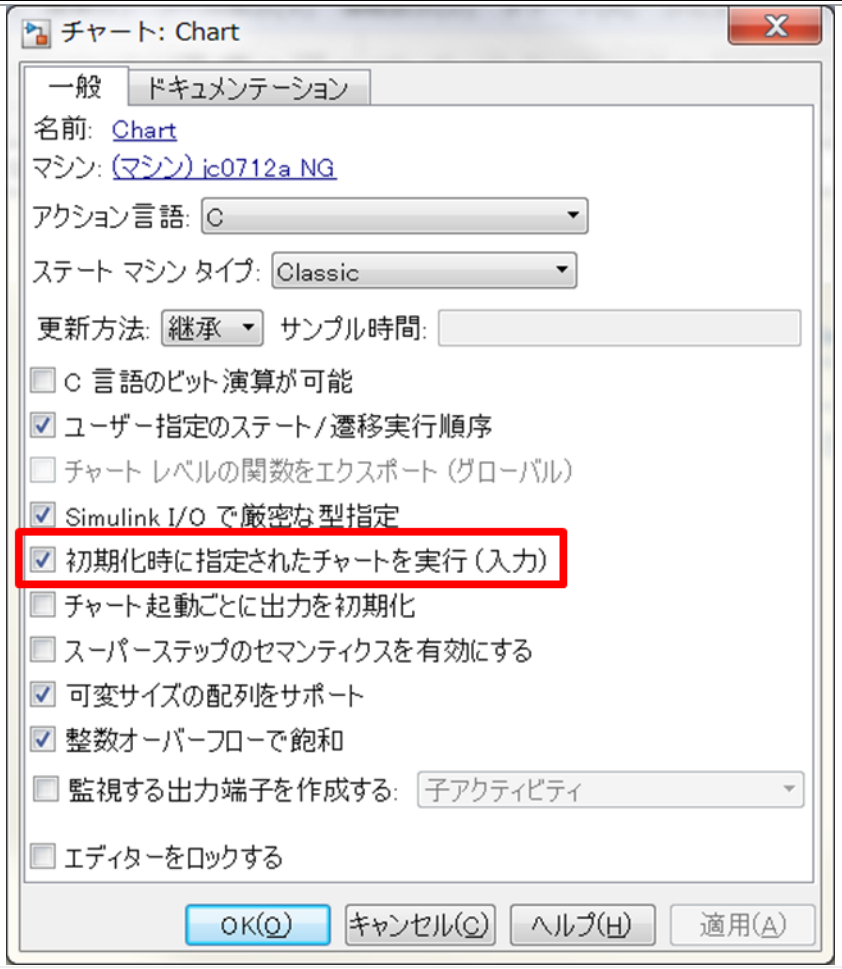
4.1.2. jc_0712 : デフォルト遷移パスの実行タイミング

ルール ID : タイトル	jc_0712 : デフォルト遷移パスの実行タイミング	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	[Chart]のプロパティの{初期化時に指定されたチャートを実行}のチェックを外します。	-

【正】
[Chart]のプロパティの{初期化時に指定されたチャートを実行}のチェックが外れています。

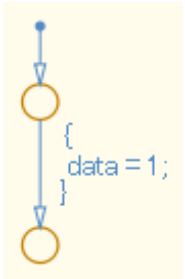
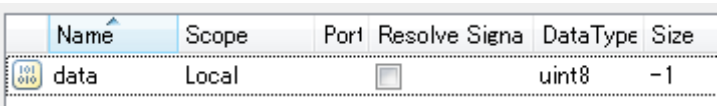


【誤】
[Chart]のプロパティの{初期化時に指定されたチャートを実行}のチェックが付いています。



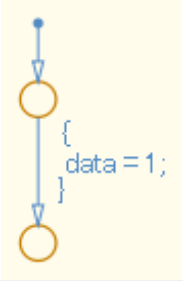
根拠	
サブ ID	**記述内容**
a	・各[Chart]の設定を統一することでモデルの誤解釈を防ぎます。 チェックを入れた場合に、デフォルト遷移線で入力信号を参照する場合は注意が必要となります。(詳細はマニュアルを参照)

4.1.3. jc_0700 : Stateflow ブロックにおける未使用のデータ

ルール ID : タイトル	**jc_0700 : Stateflow ブロックにおける未使用のデータ**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	Stateflow ブロックに未使用の Stateflow のデータ、イベントおよびメッセージは存在させません。 そのために、コンフィギュレーションパラメーターの{診断} - {Stateflow} - {使用されていないデータ、イベントおよびメッセージ} を”なし”以外にします。	-
【正】  		

Name	Scope	Port	Resolve Signa	DataType	Size
data	Local			uint8	-1

【誤】
未使用データが定義されています。



Name	Scope	Port	Resolve Signa	DataType	Size
data	Local			uint8	-1
data1	Local			uint8	-1

根拠

サブ ID	記述内容
a	・Stateflow ブロックに未使用のデータおよびイベントが存在すると保守性、運用性に影響がある可能性があります。 また、生成コードに使用されていないデータの宣言文が挿入されます。

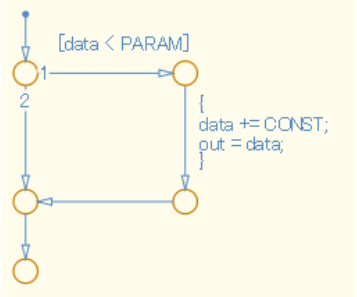
4.1.4. db_0125：Stateflow のローカルデータ

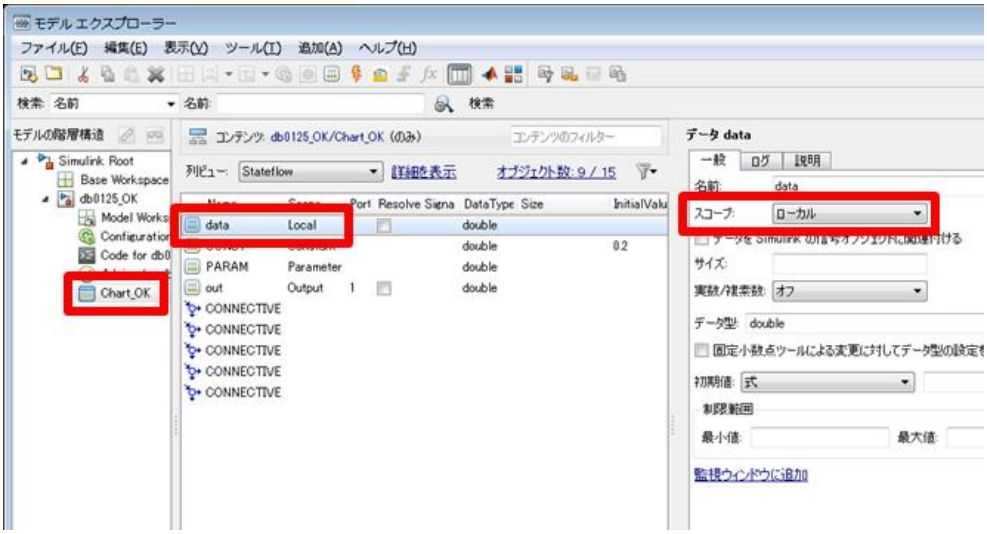
ルール ID：タイトル	db_0125：Stateflow のローカルデータ

ルール

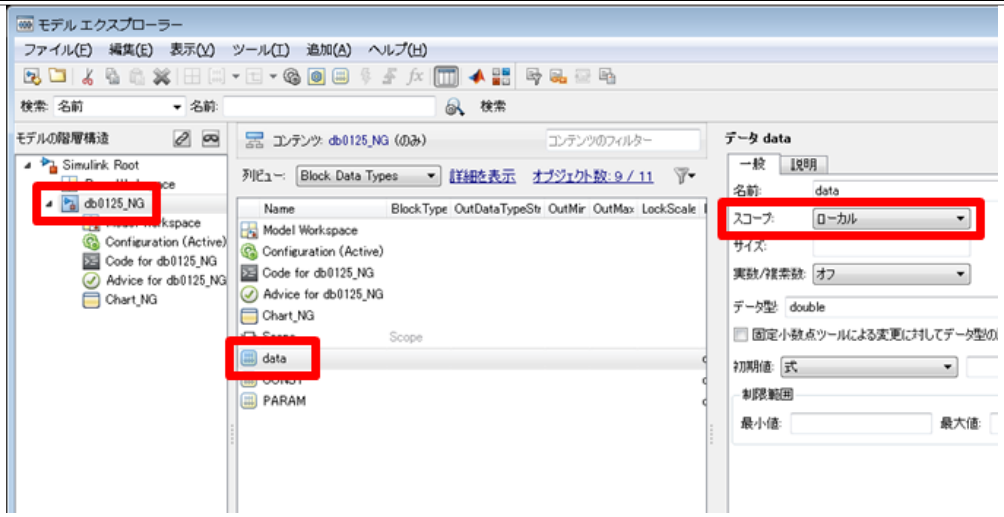
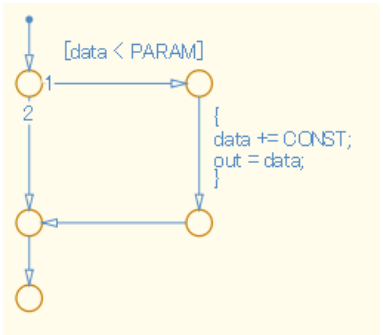
サブ ID	記述内容	カスタムパラメーター
a	{スコープ}が"ローカル"のローカルデータをマシンレベルで定義しません。	-

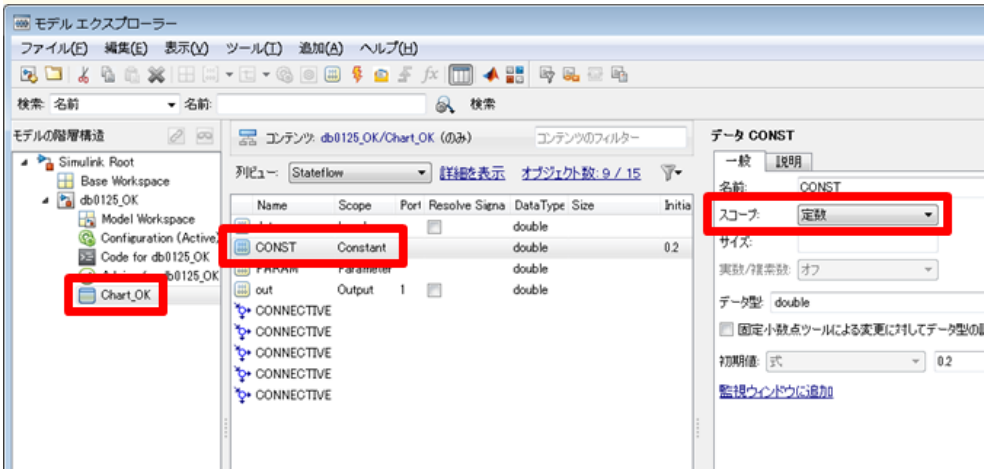
【正】



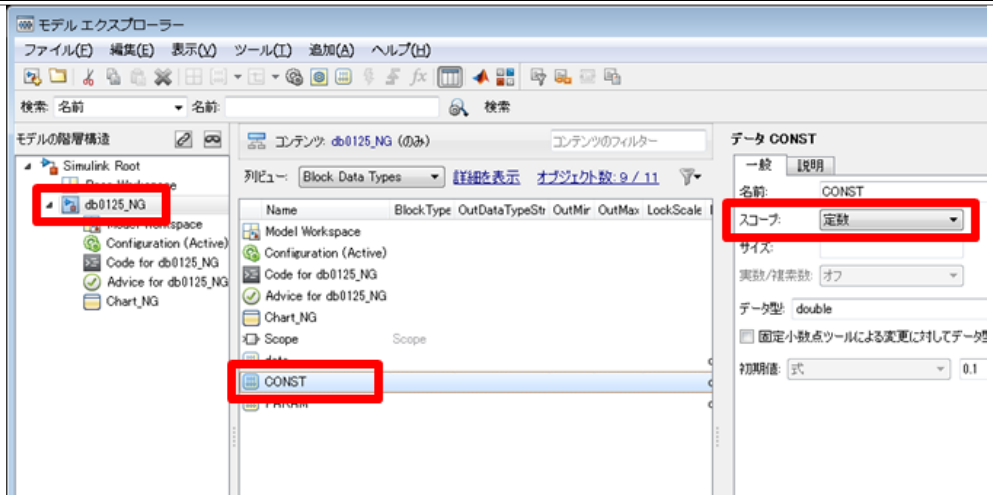
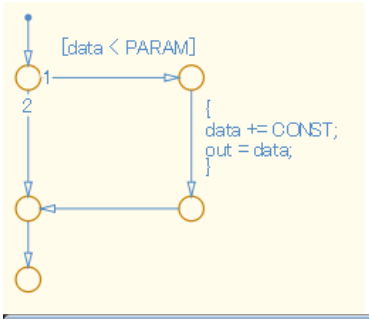


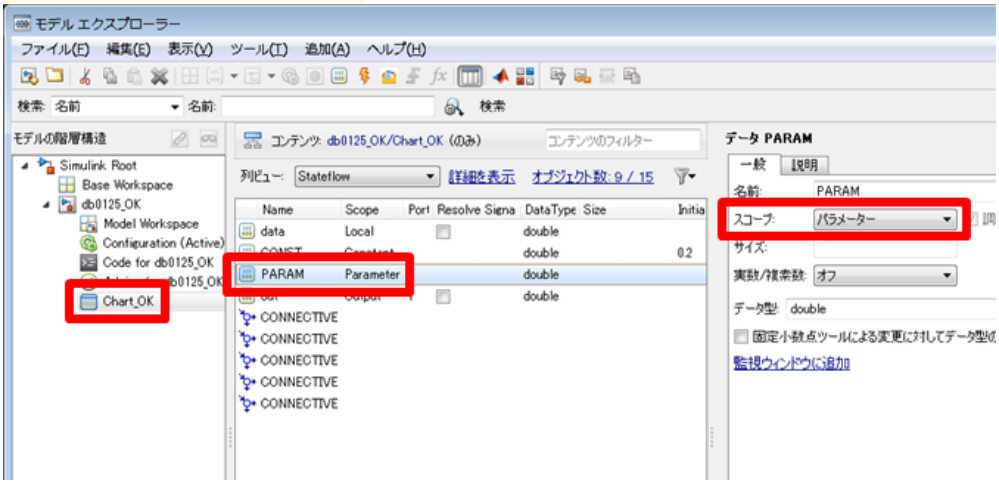
【誤】
スコープがローカルのローカルデータをマシンレベルで設定しています。

<table>
<tr><td></td><td> 
</td></tr>
<tr><td>b</td><td>{スコープ}が”定数”のローカルデータをマシンレベルで定義しません。</td><td>-</td></tr>
<tr><td></td><td>【正】
 

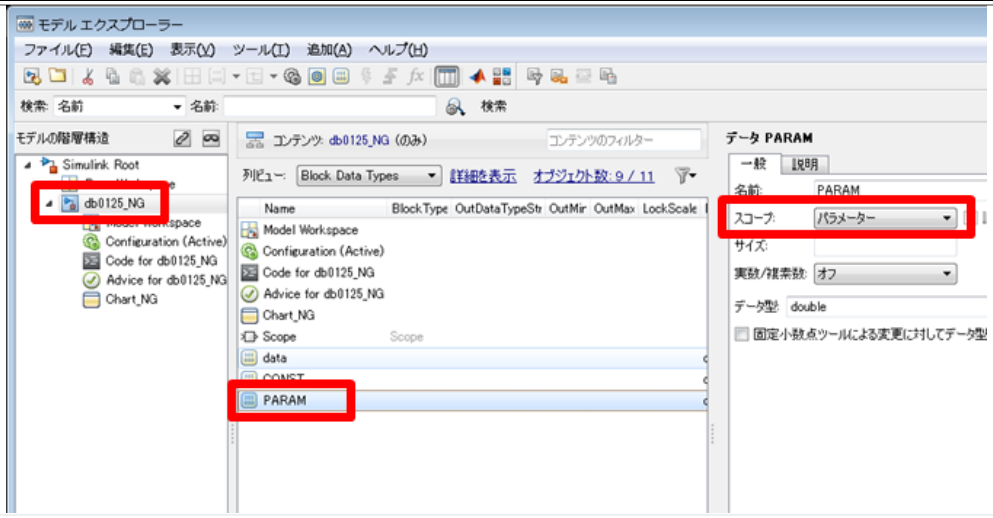


【誤】
スコープが定数のローカルデータをマシンレベルで設定しています。</td><td></td></tr>
</table>

<table>
<tr><td></td><td> 
</td></tr>
<tr><td>c</td><td>{スコープ}が”パラメーター”のローカルデータをマシンレベルで定義しません。</td><td>-</td></tr>
<tr><td></td><td>【正】
 

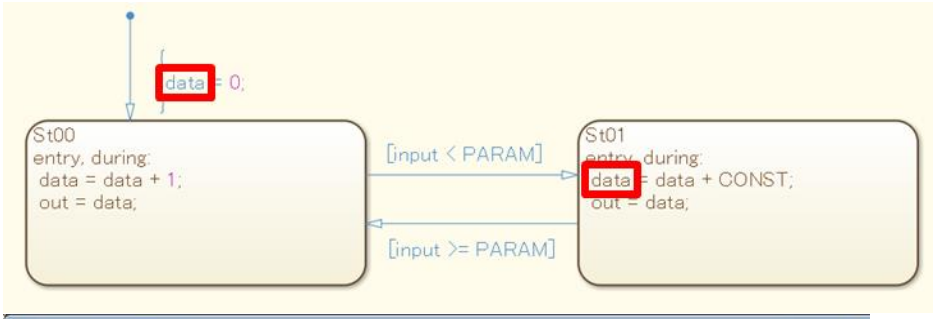


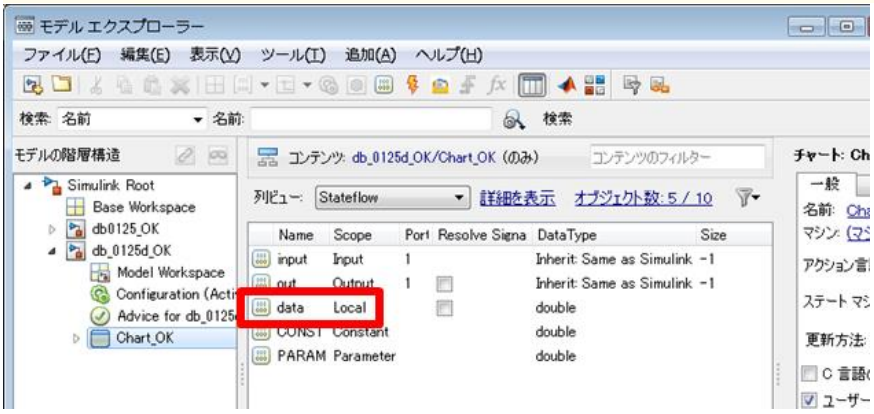
【誤】
スコープがパラメーターのローカルデータをマシンレベルで設定しています。</td><td></td></tr>
</table>



d	親子関係になる Stateflow ブロック内で、同じ名前のローカルデータを含んではなりません。	-

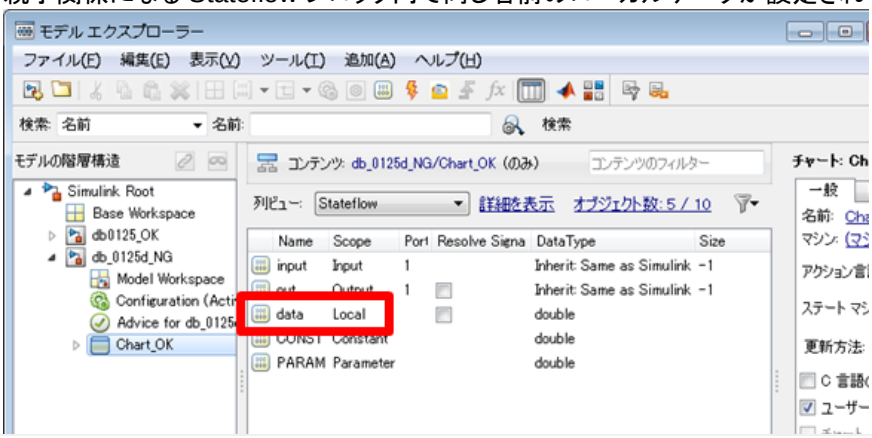
【正】

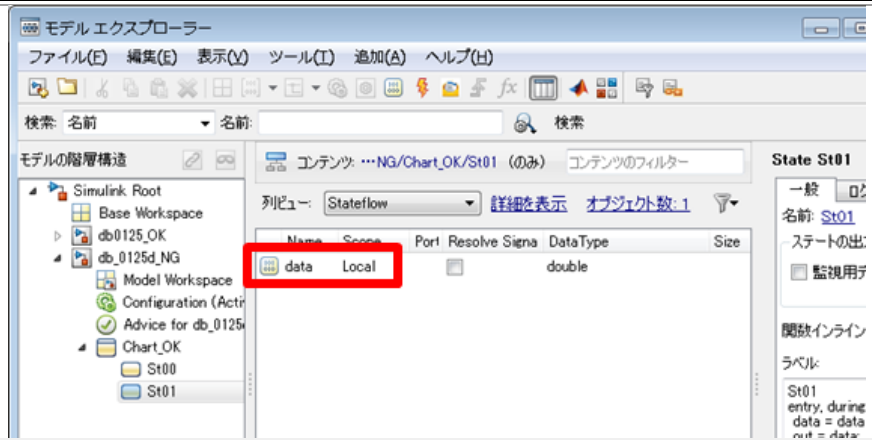




【誤】

親子関係になる Stateflow ブロック内で同じ名前のローカルデータが設定されています。



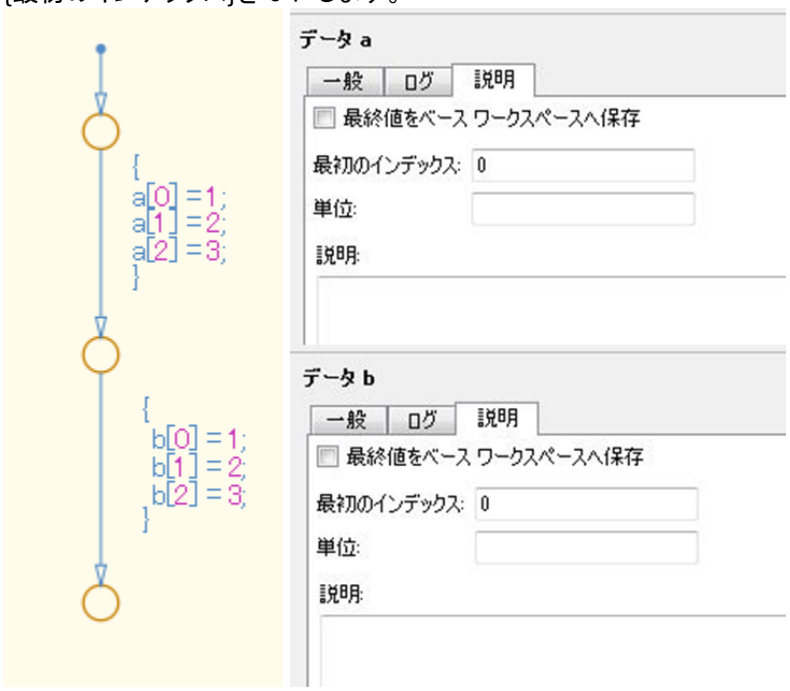


根拠	
サブ ID	**記述内容**
a	・マシンレベルで定義すると他の Stateflow ブロックと共通の変数の取り扱いとなり、複数の Stateflow ブロックで同じ変数名を使っていた場合、それぞれの Stateflow ブロック動作の影響を受けローカル変数として動作しません。 ・Stateflow ブロックを別モデルにコピーした際に、定義がなくなってしまうことを防ぎます。
bc	・Stateflow ブロックを別モデルにコピーした際に、定義がなくなってしまうことを防ぎます。
d	・親子関係になる Stateflow ブロック内で、同じ名前のローカルデータが在ると、ローカルデータの影響範囲が解らないので可読性が低下します。

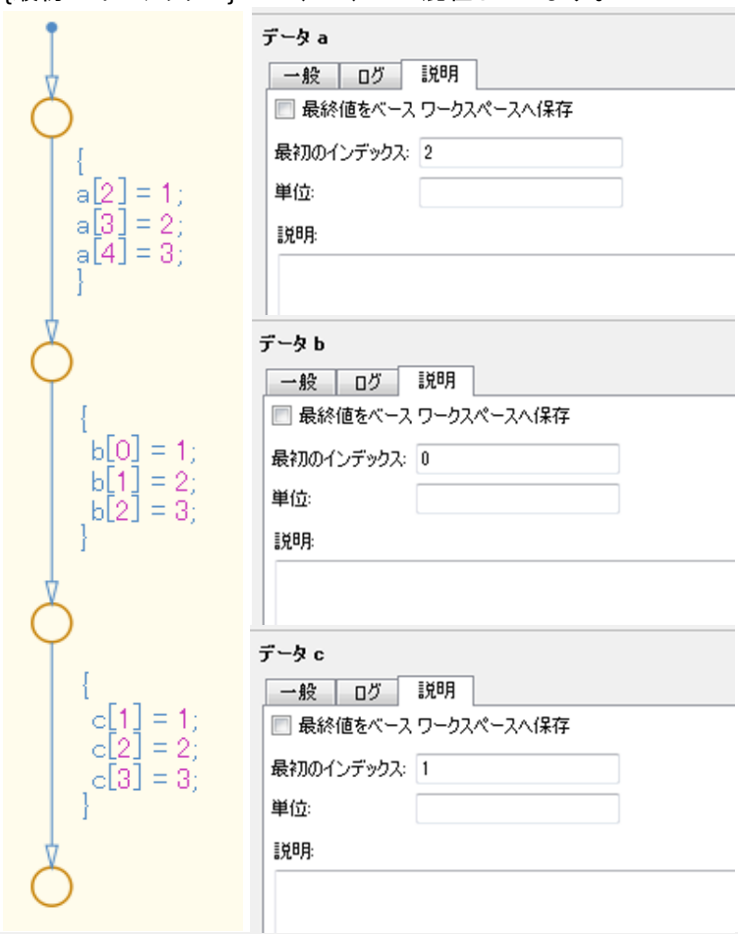
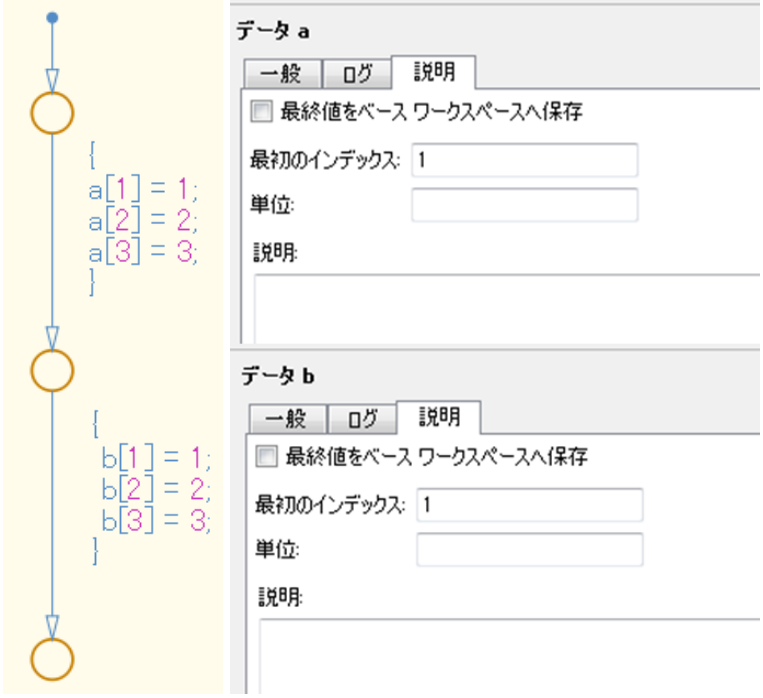
4.1.5. jc_0701 ：最初のインデックスで使用可能な数値

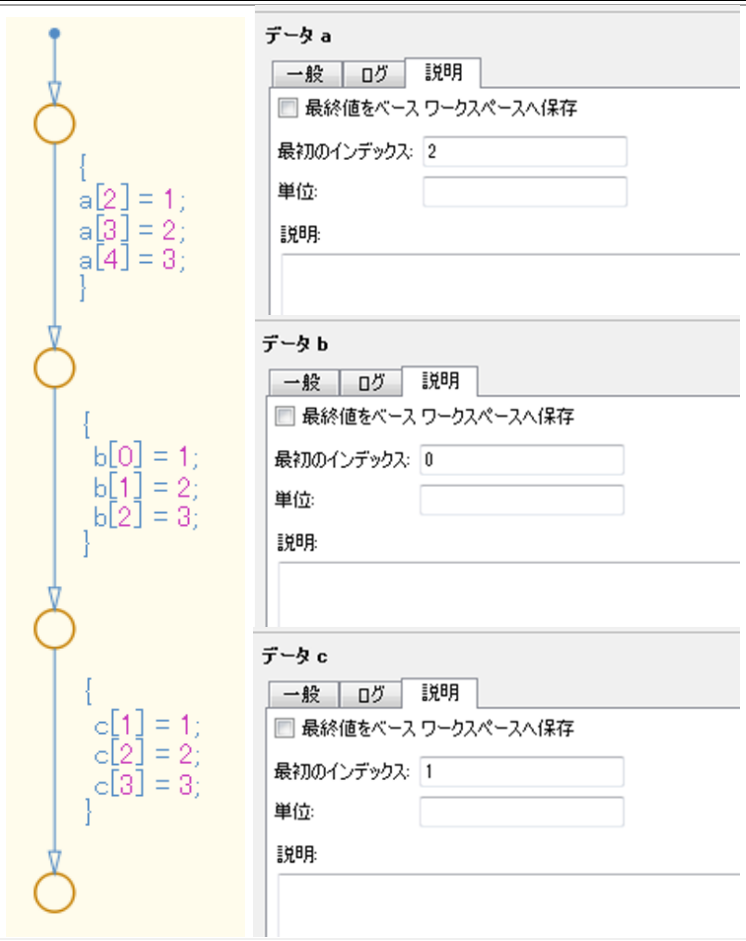
ルール ID：タイトル	**jc_0701 ：最初のインデックスで使用可能な数値**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a1	アクション言語が C 言語の場合、Stateflow のデータの{最初のインデックス}を”0”にします。	-

【正】
{最初のインデックス}を”0”にします。



【誤】

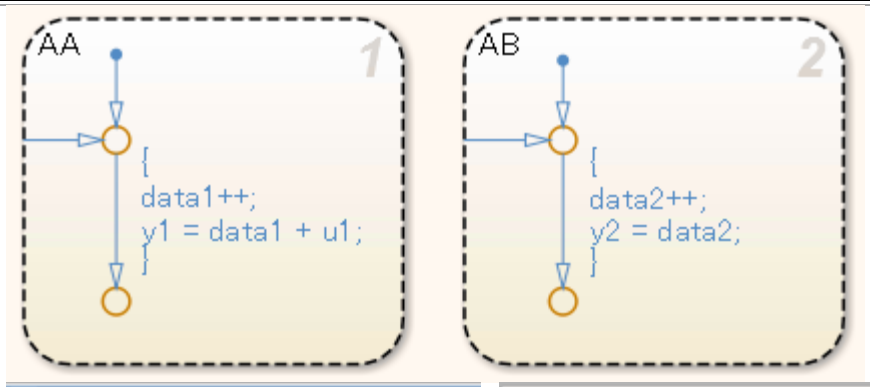
	{最初のインデックス}が”0”、”1”、”2”で混在しています。 	
a2	アクション言語が C 言語の場合、Stateflow のデータの{最初のインデックス}を”1”にします。	-
	【正】 {最初のインデックス}を”1”にします。  【誤】 {最初のインデックス}が”0”、”1”、”2”で混在しています。	

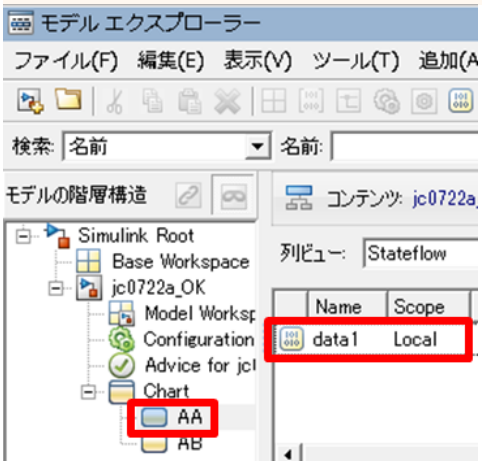


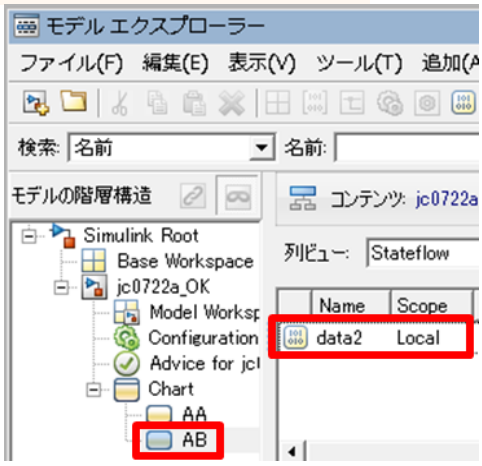
根拠	
サブ ID	**記述内容**
a1	・最初のインデックスを統一することでロジックが容易に理解できます。
a2	・最初のインデックスを統一することでロジックが容易に理解できます。ただし、C 言語は 0 ベースであり、1 ベースへ合わせるインデックス計算処理がコードへ反映されるため、コード可読性が低下します。

4.1.6. jc_0722：パラレルステートにおけるローカルデータの設定方法

ルール ID：タイトル	**jc_0722：パラレルステートにおけるローカルデータの設定方法**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	1 つのステートで完結するローカル変数は、使用するステートで定義します。	-
	【正】 使用するステートでローカル変数が定義されています。	

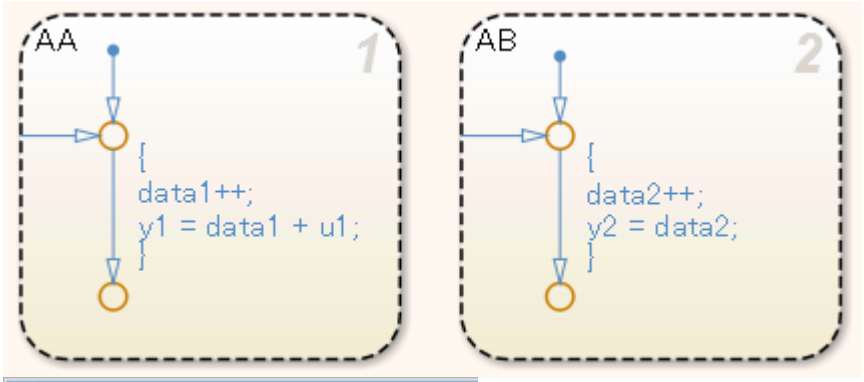


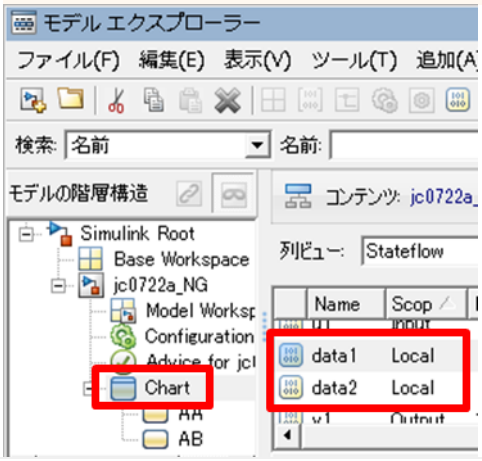




【誤】

使用するステートでローカル変数が定義されていません。

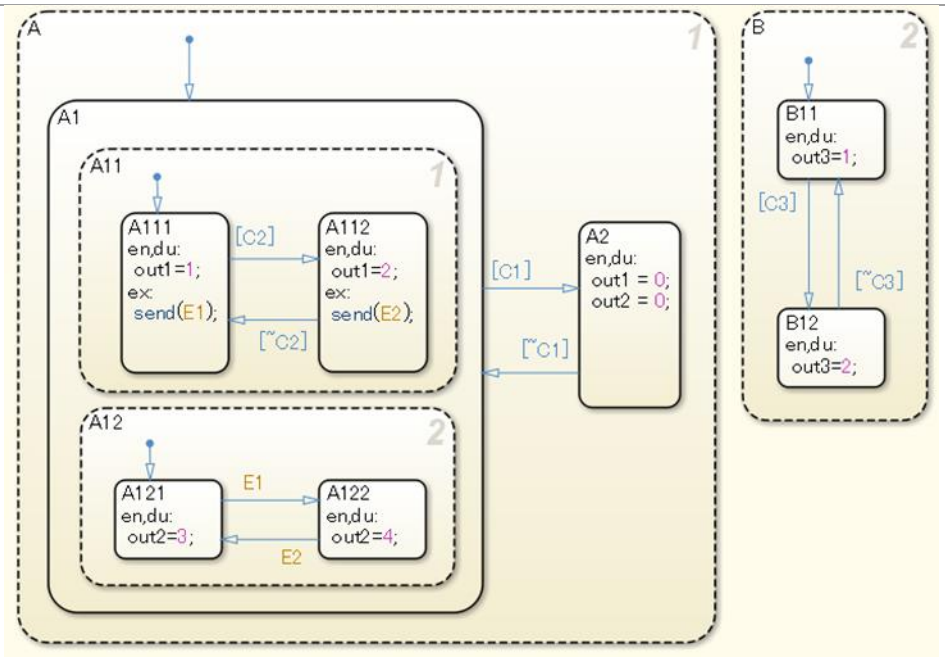




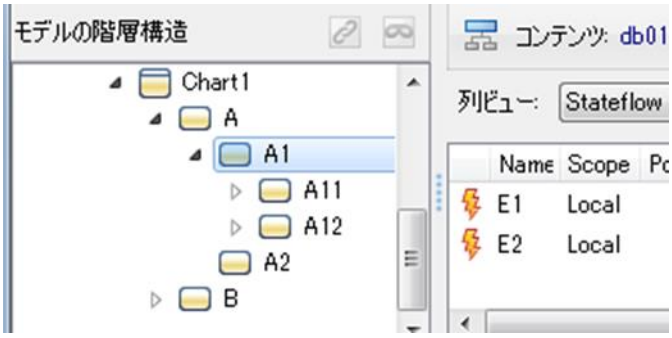
根拠	
サブ ID	**記述内容**
a	・変数の有効範囲を明示的に限定することで、可読性と保守性を向上し、意図しない参照や変更を回避します。

4.1.7. db_0126：Stateflow のイベントの定義方法

ルール ID：タイトル	db_0126：Stateflow のイベントの定義方法	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	Stateflow のイベントは、使用する Stateflow ブロック内の最小スコープレベルで定義します。	-



【正】

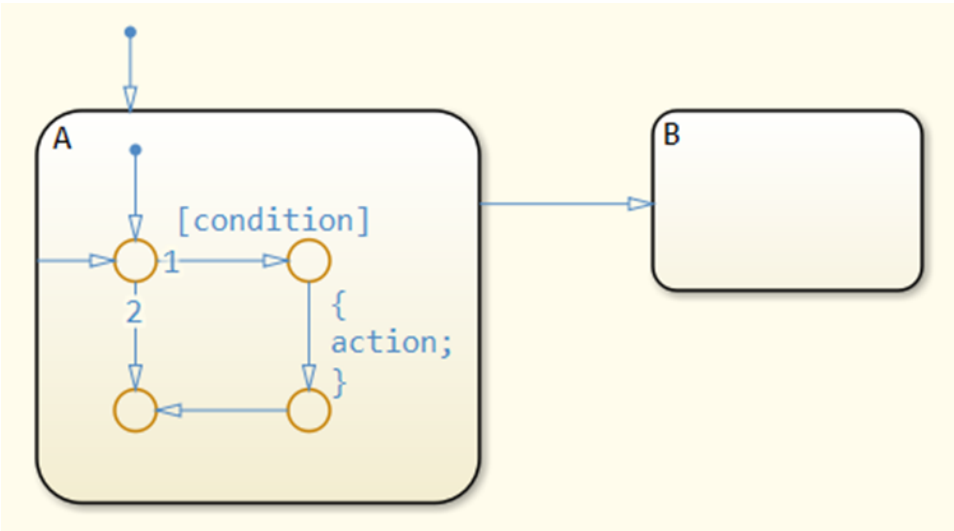
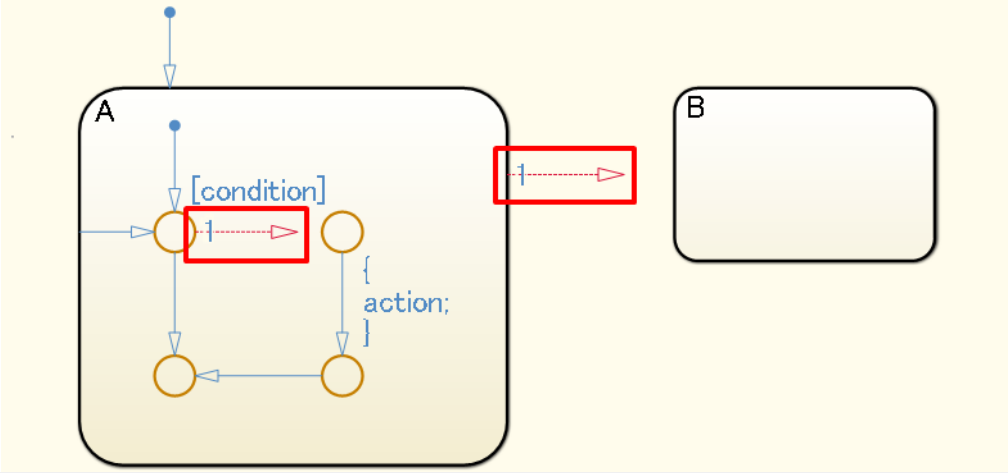


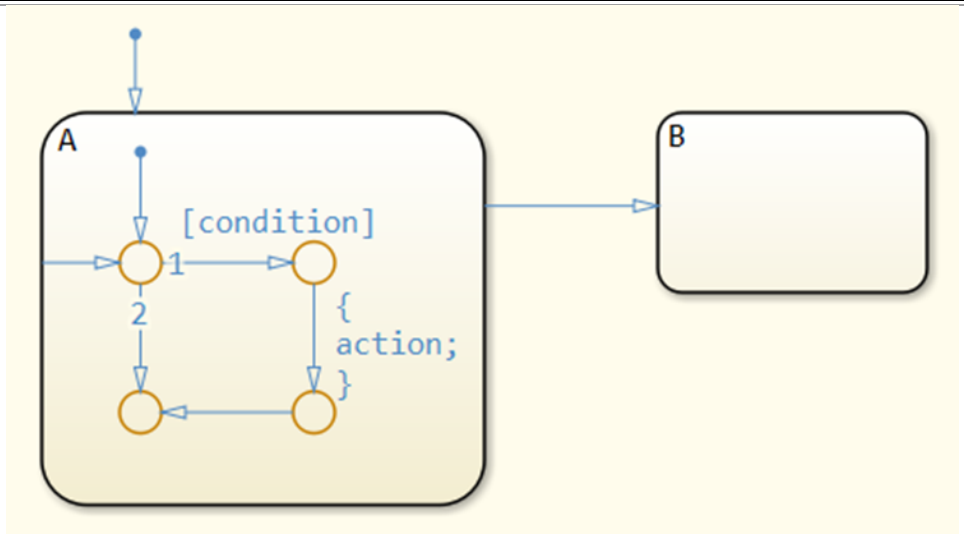
【誤】

根拠	
サブ ID	**記述内容**
a	・使用箇所を限定することで、信頼性が向上します。

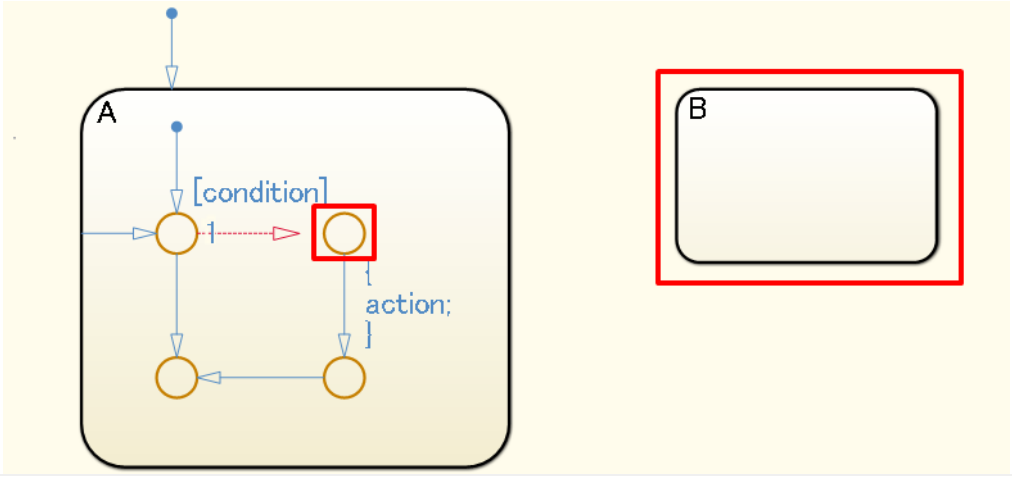
4.2. Stateflow ダイアグラム

4.2.1. jc_0797 ： 未接続の遷移線 / ステート / コネクティブジャンクション

ルール ID : タイトル	jc_0797 ： 未接続の遷移線 / ステート / コネクティブジャンクション	
ルール		
サブ ID	記述内容	カスタムパラメーター
a	未接続の遷移線を含みません。	-
	【正】 未接続の遷移線がありません。  	
	【誤】 未接続の遷移線があります。  	
b	未接続の排他的(OR)ステート、遷移元がないコネクティブジャンクションを含みません。	-
	【正】 未接続の排他的(OR)ステートや遷移元がないコネクティブジャンクションがありません。	



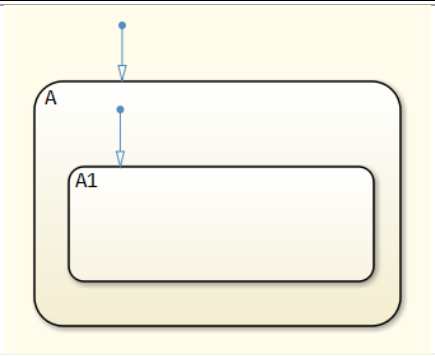
【誤】
未接続の排他的(OR)ステートや遷移元がないコネクティブジャンクションがあります。



根拠	
サブ ID	**記述内容**
ab	・結線忘れによるシミュレーション結果の誤判定やコードが生成されないなどの弊害が懸念されます。

4.2.2. db_0137 : ステートチャートのステート

ルール ID : タイトル	**db_0137 : ステートチャートのステート**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	[Chart]、もしくは、ステートの{構造}に”OR(排他)”を設定し、その直下の階層にステートを配置する場合、その階層には少なくとも 2 つ以上のステートが存在しなければなりません。	-
	【誤】 [Chart]、および、ステート A の{構造}に"OR(排他)"が設定されていますが、それぞれの直下の階層にステートが1つしかありません。	

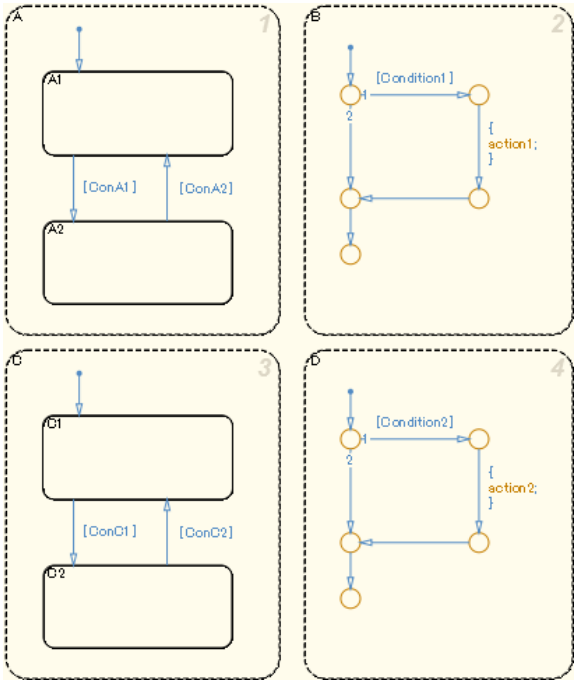


根拠	
サブ ID	**記述内容**
a	・冗長な記述により可読性が低下します。 ・コード生成時に無駄な状態変数が存在してしまいます。

4.2.3. jc_0721 : パラレルステートの使用方法

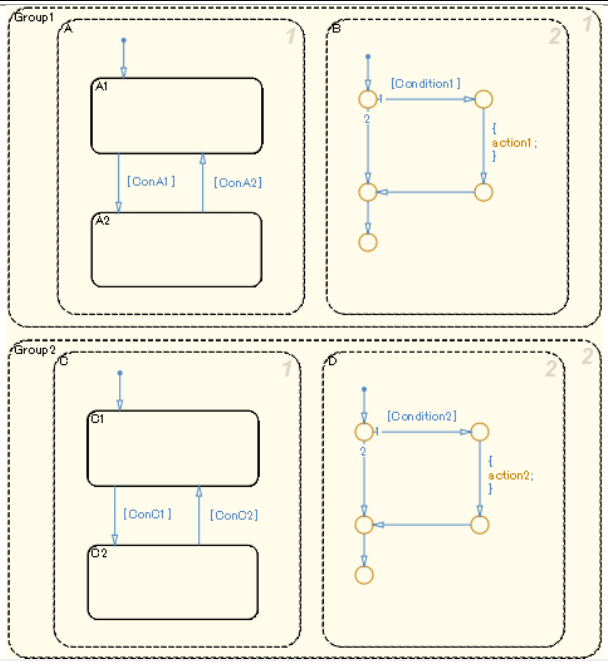
ルール ID : タイトル	**jc_0721 : パラレルステートの使用方法**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	パラレルステートのサブステートに、パラレルステートは配置しません。	-

【正】



【誤】

並列ステートのサブステートが並列ステートになっています。

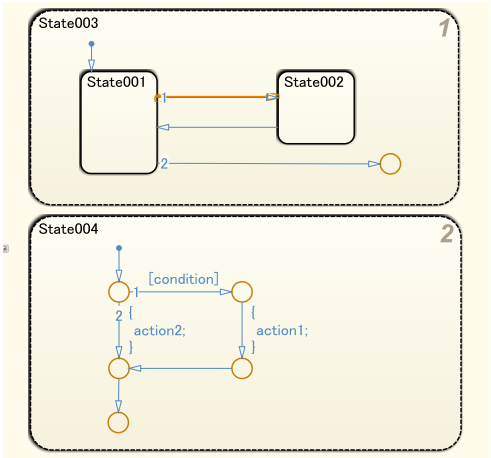


根拠	
サブ ID	**記述内容**
a	・パラレルステートのサブステートにパラレルステートを置いた場合と、直接、パラレルステートを置いた場合は同じ機能になります。 ・パラレルステートの階層化で複雑化し可読性が低下します。

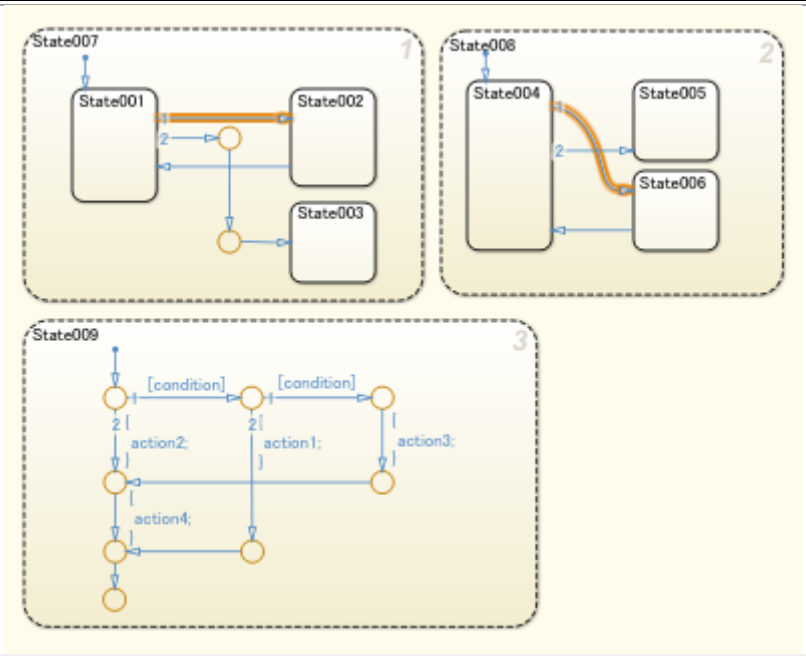
4.2.4. db_0129：遷移線の結線

ルール ID：タイトル	**db_0129：遷移線の結線**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	遷移線は、交差させません。	-

【正】
遷移線が交差していません。

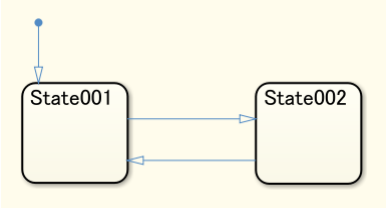


【誤】
遷移線が交差しています。

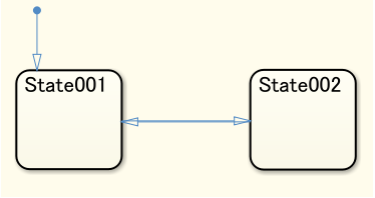


b	遷移線は、他の遷移線と重ねません。	-

【正】
遷移線を他の遷移線と重ねません。

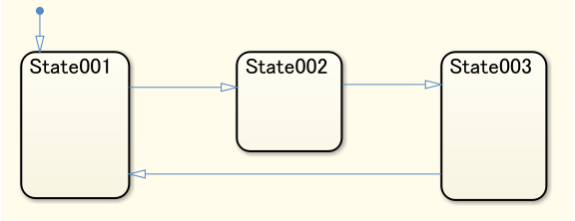


【誤】
遷移線が他の遷移線と重なっています。

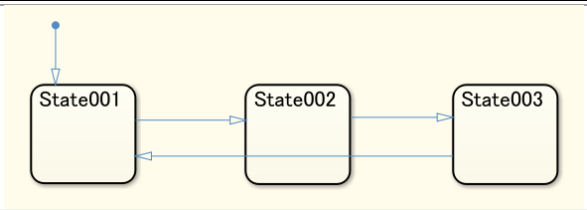


c	遷移線は、ステートを横切りません。	-

【正】
遷移線はステートを横切りません。

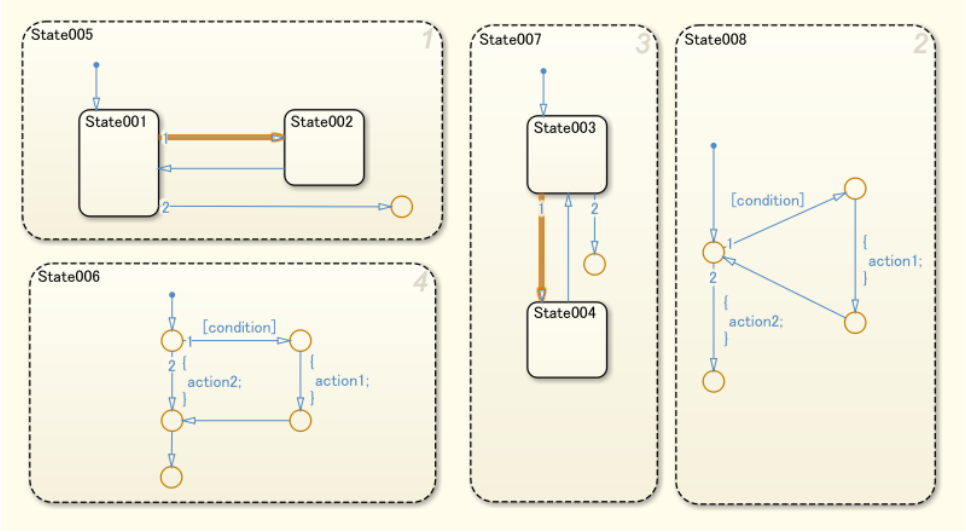


【誤】
遷移線がステートを横切っています。

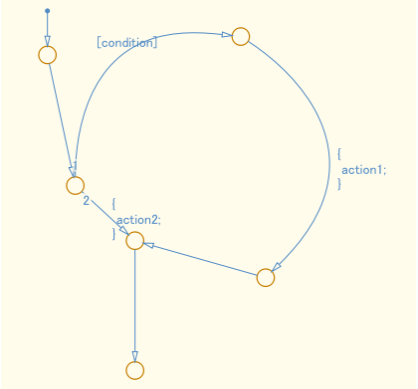


d	遷移線は、垂直、水平方向に引きます。 フローチャートのループを除き、ななめ線を使用しません。	-

【正】
遷移線が垂直、水平方向に引かれています。また、フローチャートのループを除き、ななめ線は使用しません。

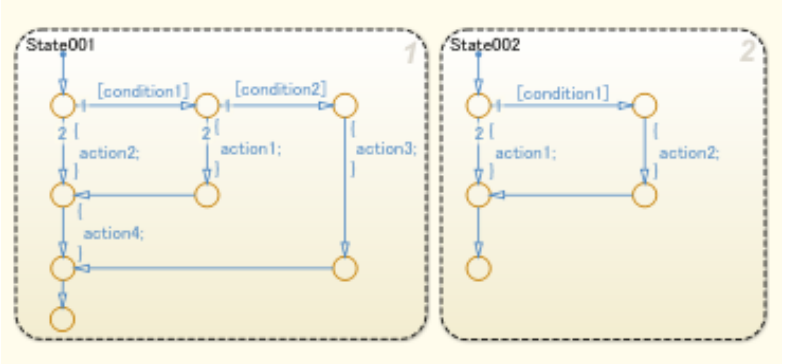


【誤】
遷移線が垂直、水平方向に引かれていません。

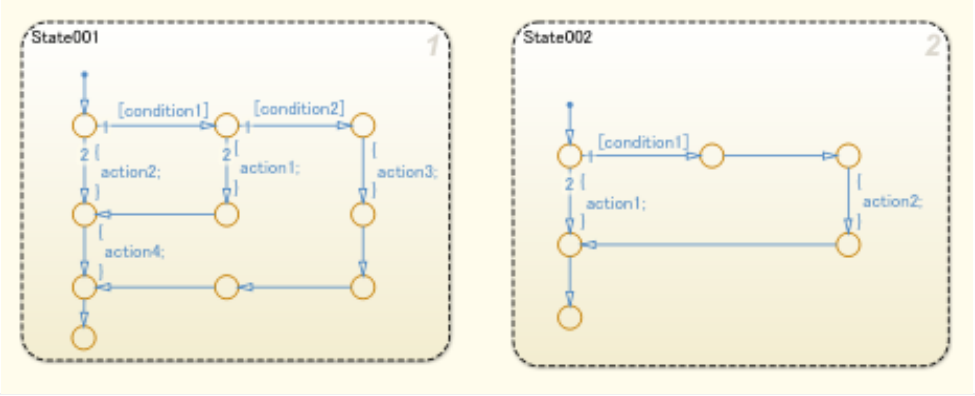


e	不要なコネクティブジャンクションは設けません。	-

【正】
不要なコネクティブジャンクションは設けません。



【誤】
不要なコネクティブジャンクションが設けられています。

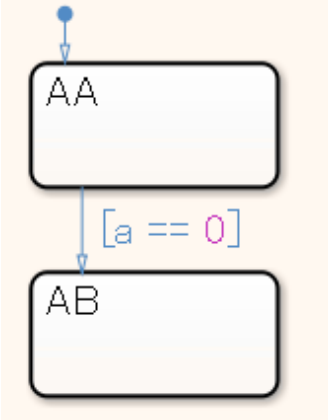


根拠	
サブ ID	**記述内容**
a	・遷移線が交差すると、ステート間の接続関係が解りにくくなります。
b	・遷移線が重なると、ステート間の接続関係が解りにくくなります。
c	・遷移線とステートが交差すると、ステート間の接続関係が解りにくくなります。
d	・遷移線の引き方を統一することで可読性が向上します。
e	・不要なコネクティブジャンクションを設けると遷移が解りにくくなります。

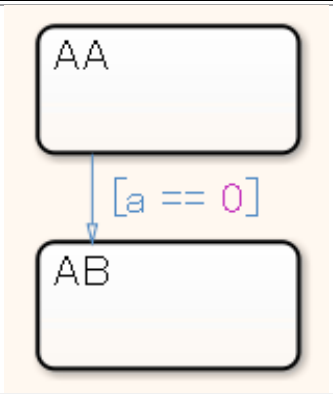
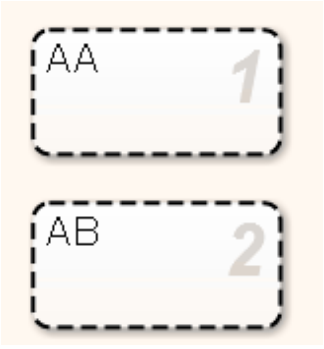
4.2.5. jc_0531 ： デフォルト遷移

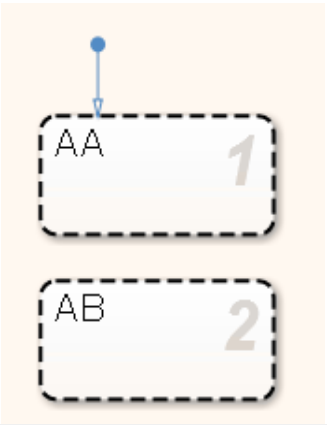
ルール ID：タイトル	**jc_0531 ： デフォルト遷移**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	・[Chart]の{構造}が"OR(排他)"の場合は、[Chart]のトップにデフォルト遷移を記述します。 ・ステートの{構造}が"OR(排他)"の場合は、そのステートの直下にデフォルト遷移を記述します。	-

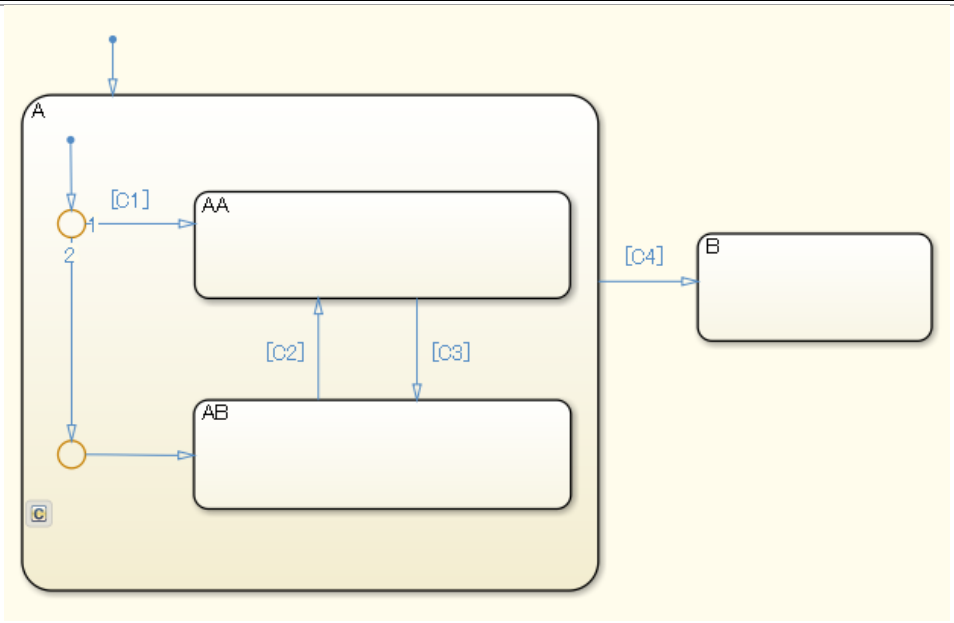
【正】
デフォルト遷移線が記述されています。

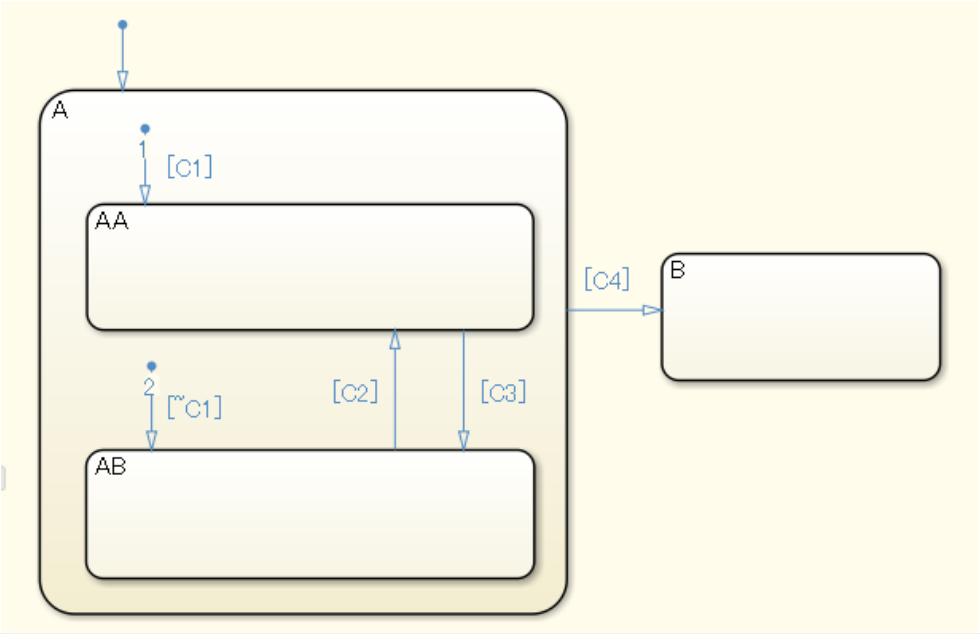


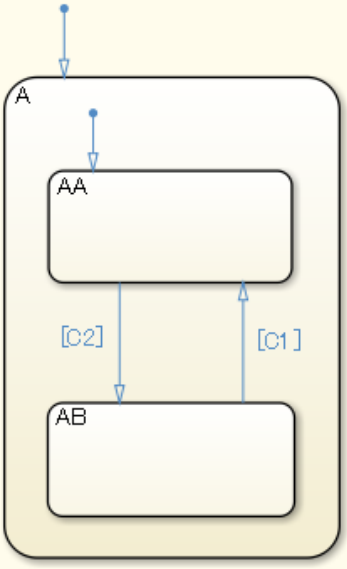
【誤】
デフォルト遷移線が記述されていません。

<table>
<tr><td></td><td> 
</td></tr>
<tr><td>b</td><td>{構造}が"AND(パラレル)"の場合は、デフォルト遷移線を記述しません。</td><td>-</td></tr>
<tr><td></td><td colspan="2">【正】
ステート AA、AB の親オブジェクトの{構造}が"AND(パラレル)"の場合、ステート AA、AB はパラレルステートです。このパラレルステートに対し、デフォルト遷移線を記述していません。
 

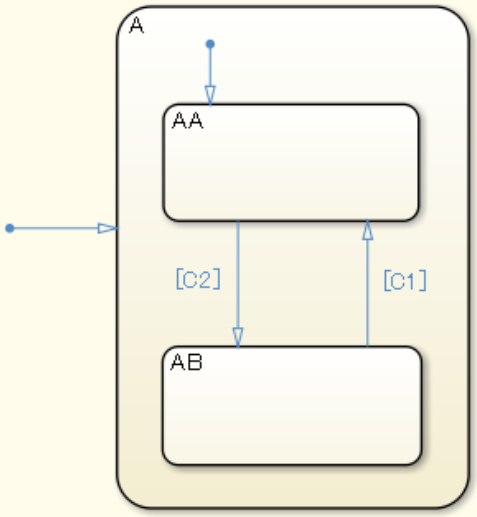
【誤】
パラレルステートであるステート AA に対し、デフォルト遷移線が記述されています。
 
</td></tr>
<tr><td>c</td><td>同一の階層に複数のデフォルト遷移を含みません。</td><td>-</td></tr>
<tr><td></td><td colspan="2">【正】
同一の階層に複数のデフォルト遷移を含んでいません。</td></tr>
</table>

<table>
<tr><td></td><td> 

【誤】
ステート A は、同一の階層に複数のデフォルト遷移を含んでいます。
 
</td></tr>
<tr><td>d</td><td>デフォルト遷移は、ステートまたはコネクティブジャンクションの上部へ垂直に接続します。</td><td>-</td></tr>
<tr><td></td><td colspan="2">【正】
デフォルト遷移は、ステートの上部へ垂直に接続されています。</td></tr>
</table>

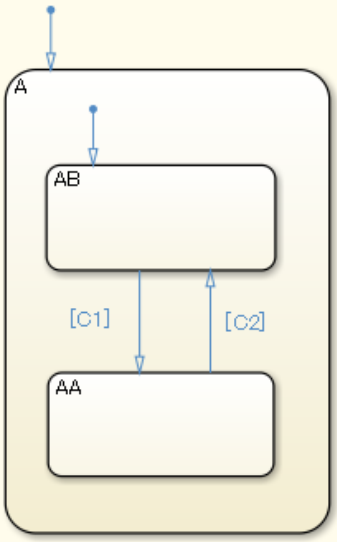


【誤】
ステート A へのデフォルト遷移は、ステートの上部へ垂直に接続されていません。

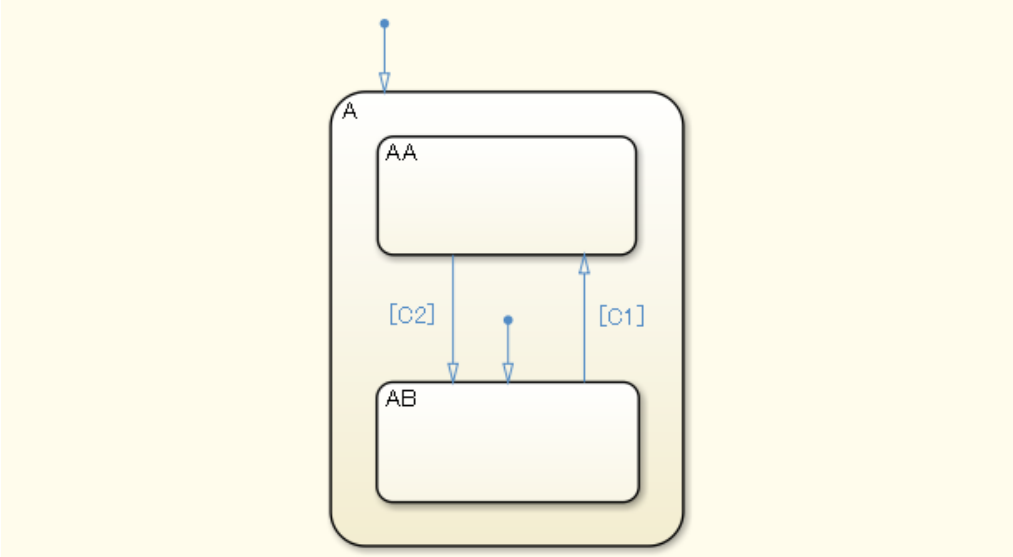


e	デフォルト遷移の遷移先ステートまたは遷移先コネクティブジャンクションは、同一階層内で最左上に配置します。	-

【正】
デフォルト遷移は同一階層内で最左上に配置されています。

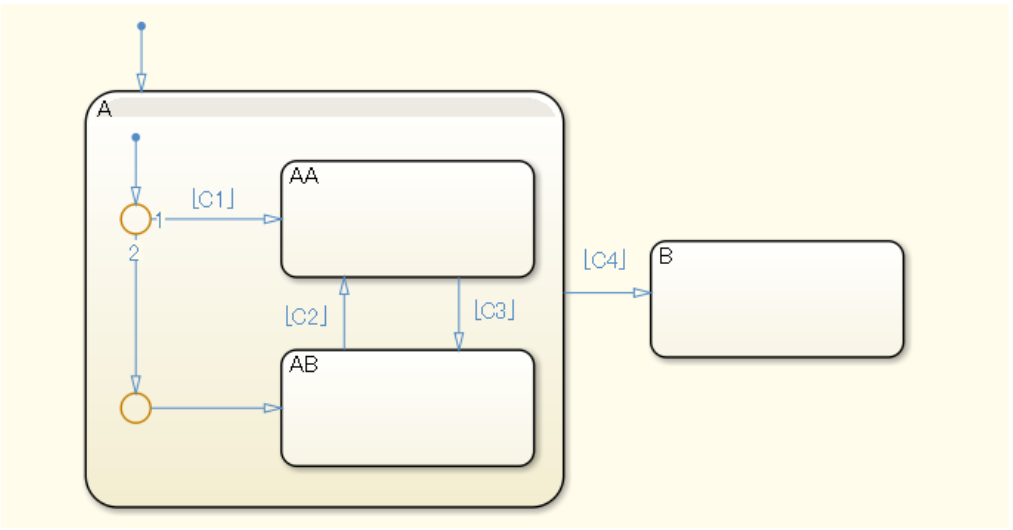


【誤】
ステート AB へのデフォルト遷移は、同一階層内で最左上に配置されていません。

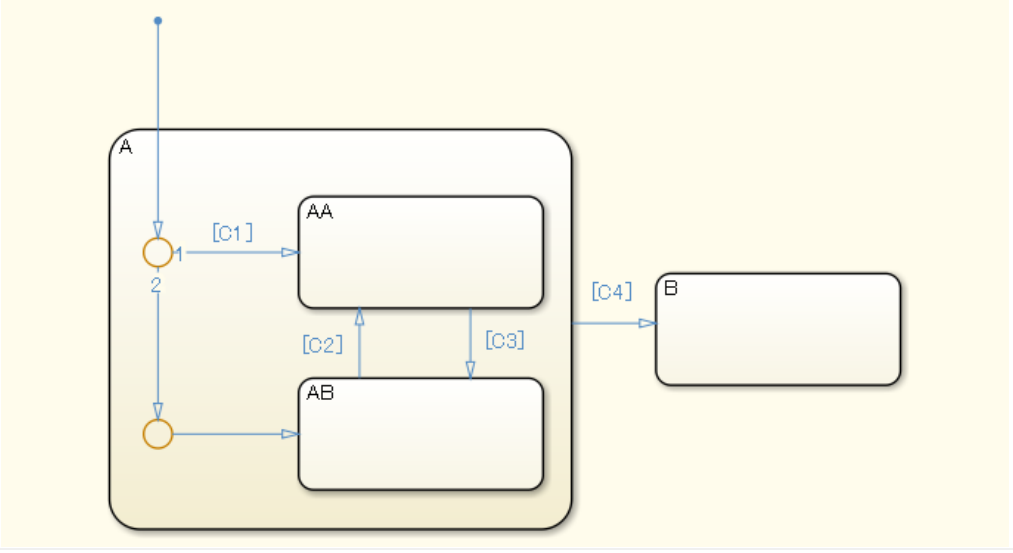


f	デフォルト遷移はステートの境界を越えません。	-

【正】
デフォルト遷移はステートの境界を越えていません。

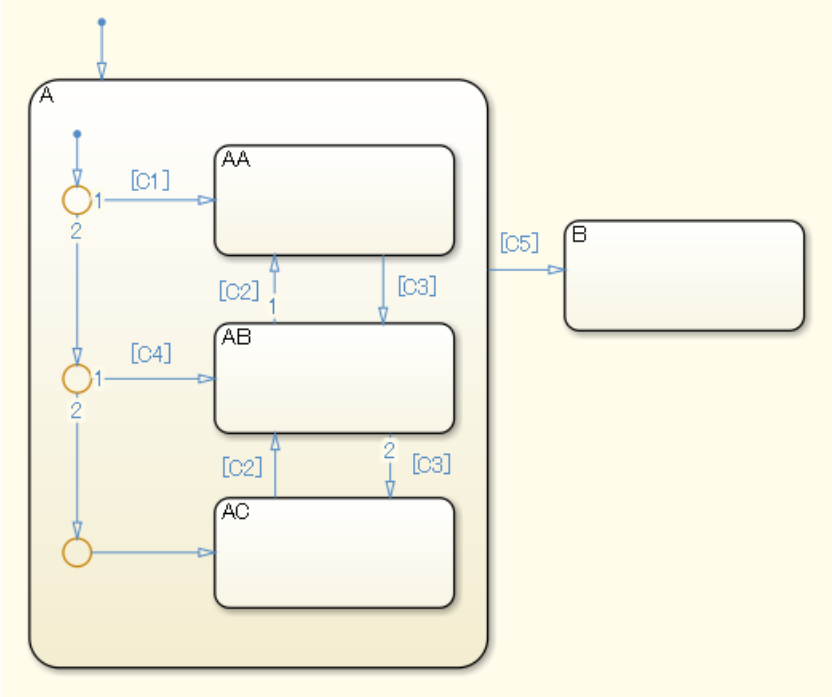


【誤】
デフォルト遷移はステートの境界を越えています。

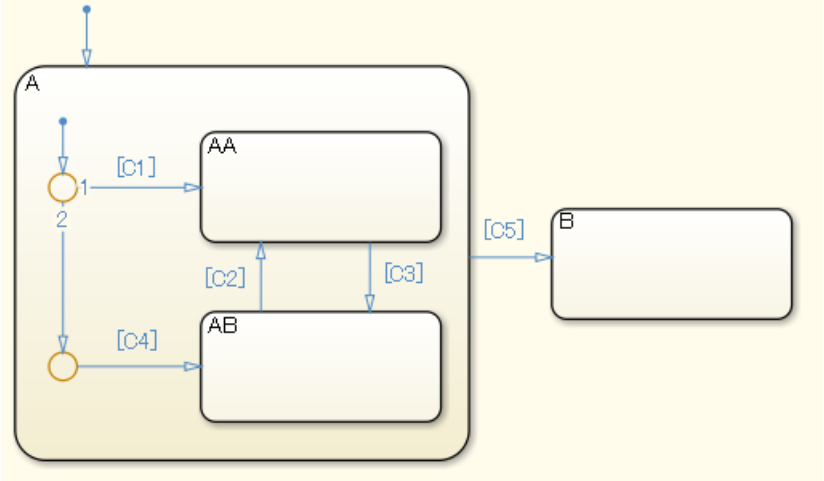


g	デフォルト遷移の遷移パスの中で、最も優先度が低いパスは無条件遷移にします。 本ルールの逸脱を検出するために、コンフィギュレーションパラメーターの{診断} - {Stateflow} - {無条件のデフォルト遷移がない}を"エラー"にします。	-

【正】
デフォルト遷移の遷移パスの中で、最も優先度が低いパスは無条件遷移です。



【誤】
デフォルト遷移の遷移パスの中で、最も優先度が低いパスは無条件遷移ではありません。



根拠

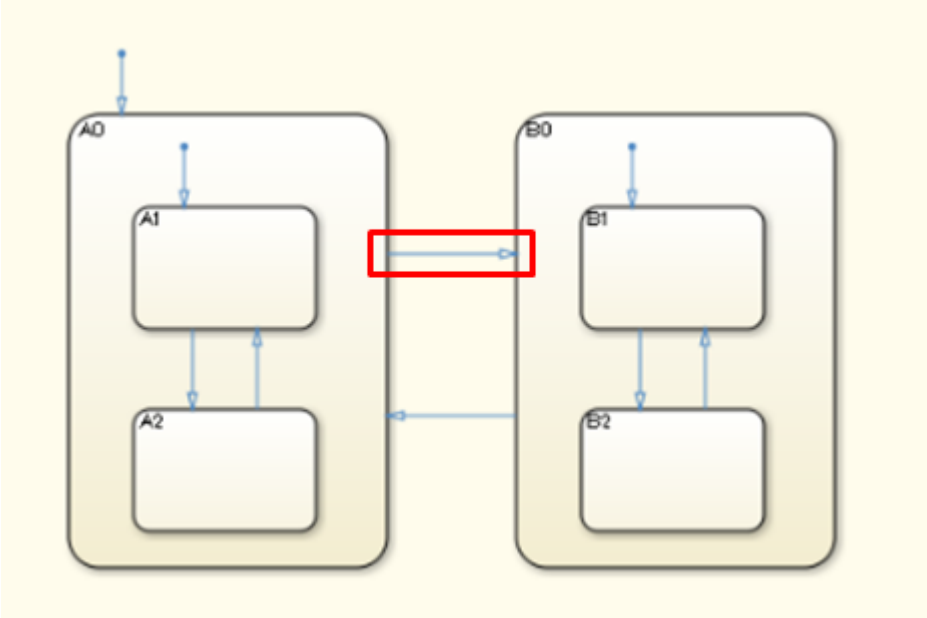
サブ ID	記述内容
a	・ステートチャートにデフォルト遷移線がないとシミュレーション時エラーとなります。 ・フローチャートにデフォルト遷移線がないと、意図的なのか入れ忘れなのか区別がつかなくなります。
b	・不要なデフォルト遷移線を記述しないことで可読性が向上します。
c	・同一階層内に複数のデフォルト遷移があると、意図したステート実行がされない可能性があります。 尚、同一階層内に複数のデフォルト遷移がある場合には、ワーニングが発生します。
d	・デフォルト遷移の角度や位置にばらつきがあったり、曲線だったりすると、可読性が低下します。

e	・デフォルト遷移に遷移先ステートや遷移先コネクティブジャンクションの位置にばらつきがあると、可読性が低下します。
f	・デフォルト遷移がステートの境界を超えると、ステートの境界線やステート内の式に交差し、可読性が低下します。
g	・デフォルト遷移の遷移パスの中に無条件遷移が無いと、遷移パスの全ての遷移条件が不成立の場合に遷移先がなくなり、意図しない動作をするおそれがあります。

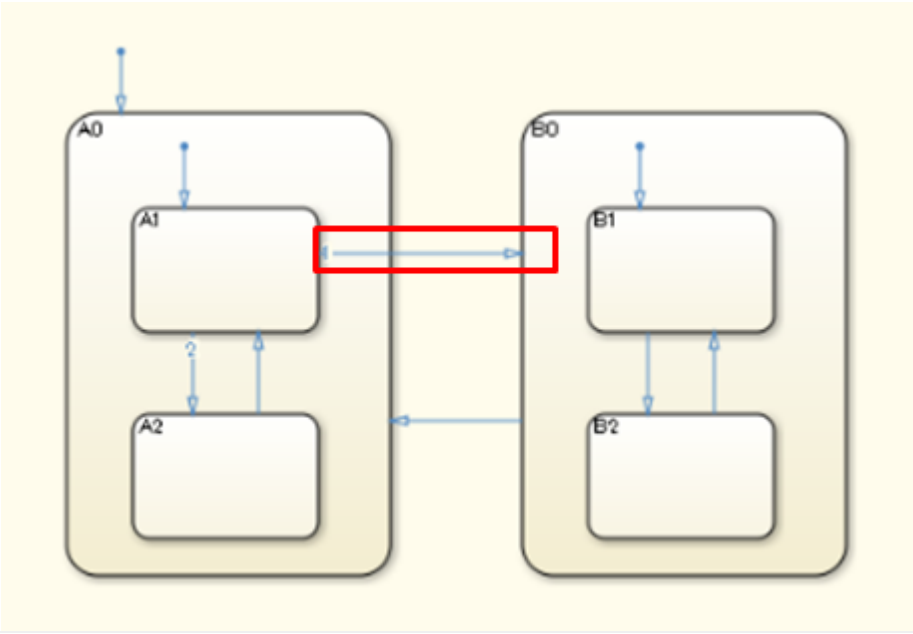
4.2.6. jc_0723：外部ステートから子ステートへの直接遷移の禁止

ルール ID：タイトル	jc_0723：外部ステートから子ステートへの直接遷移の禁止	
ルール		
サブ ID	記述内容	カスタムパラメーター
a	あるステートから外部ステートの子ステートへは、直接遷移しません。	-

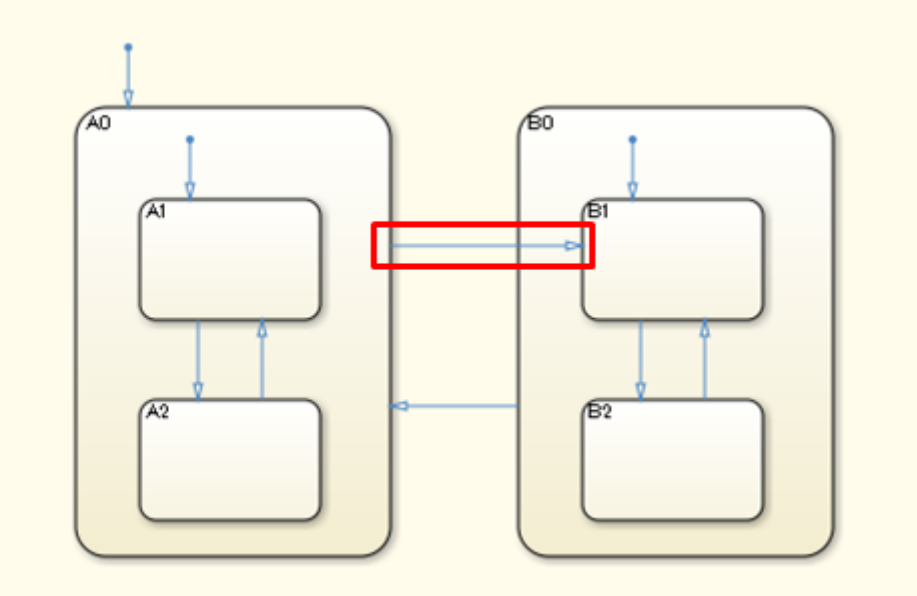
【正】
親ステートから親ステートへの遷移



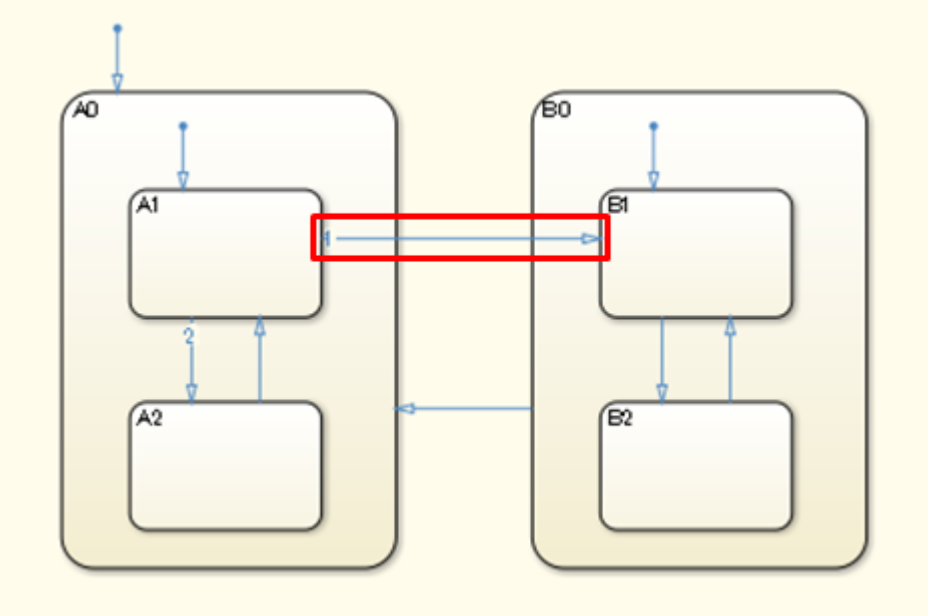
子ステートから他の親ステートへの遷移



【誤】
外部ステートからステート内の子ステートへ直接遷移しています。



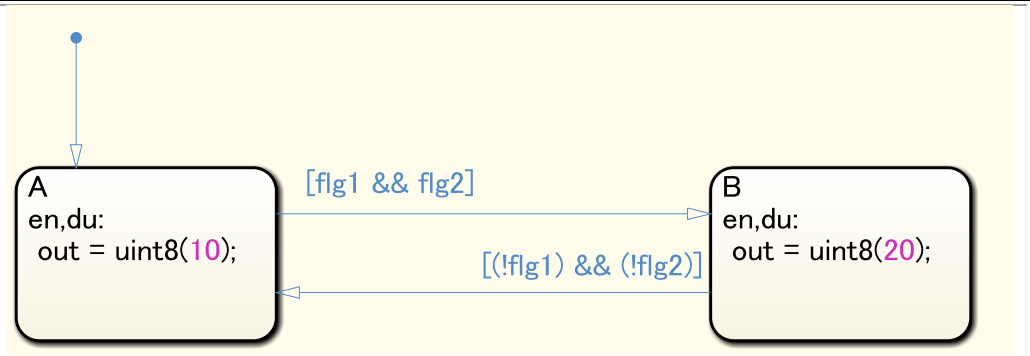
外部ステートの子ステートからステート内の子ステートへ直接遷移しています。



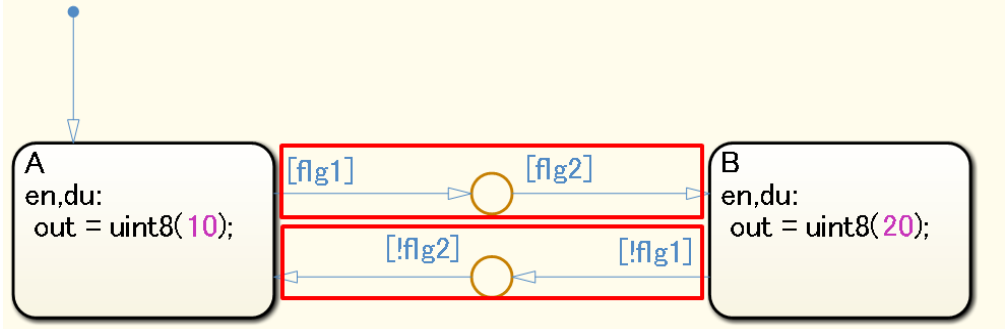
根拠	
サブ ID	**記述内容**
a	・子ステートへの直接遷移は、ステートが複雑化し、可読性が損なわれます。

4.2.7. jc_0751 ： 状態遷移におけるバックトラックの予防

ルール ID : タイトル	**jc_0751 ： 状態遷移におけるバックトラックの予防**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	一つなぎの条件式は、コネクティブジャンクションを用いた条件の分割を行いません。	-
	【正】 コネクティブジャンクションを使用して条件を分割していません。	



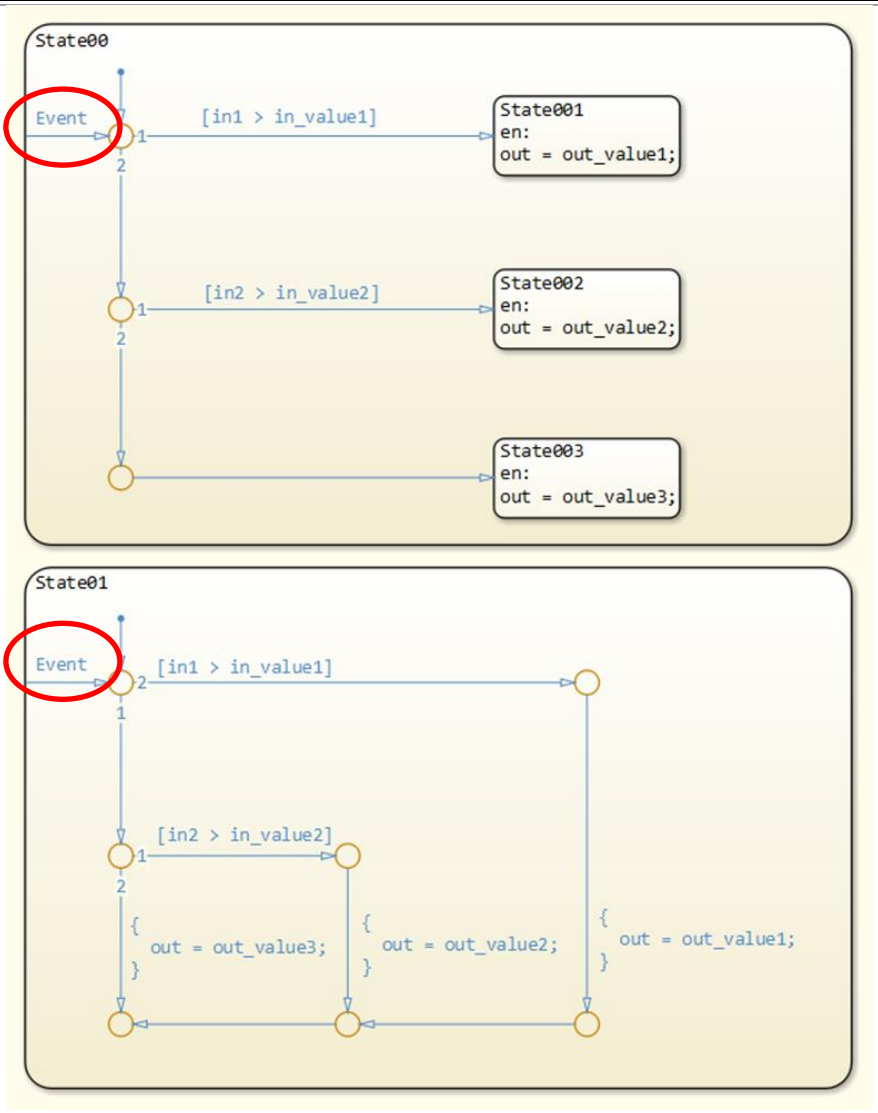
【誤】
コネクティブジャンクションを使用して条件を分割しています。



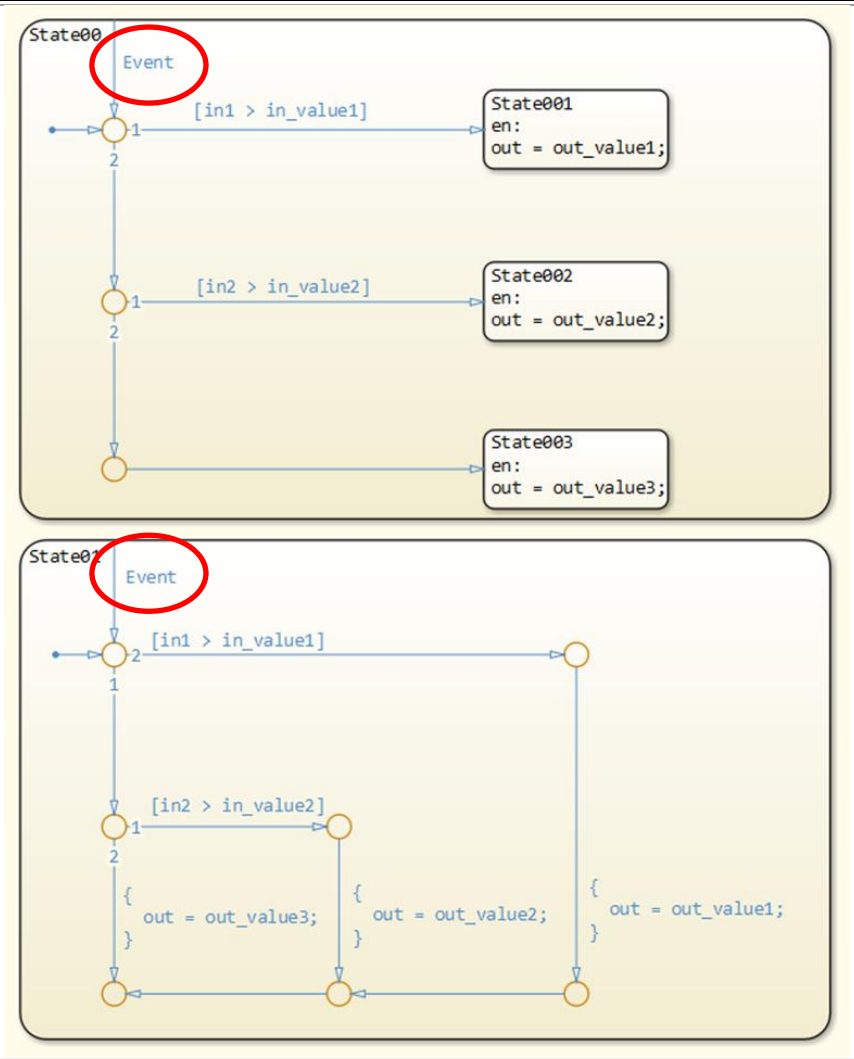
根拠	
サブ ID	**記述内容**
a	・違反した場合、バックトラッキングが発生する可能性があり、これにより設計者の意図とは異なる動作をする場合があります。

4.2.8. jc_0760 : 内部遷移線の始点

ルール ID : タイトル	**jc_0760 : 内部遷移線の始点**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	内部遷移線は、ステートの左端を始点とします。	-
	【正】 始点がステートの左端です。	



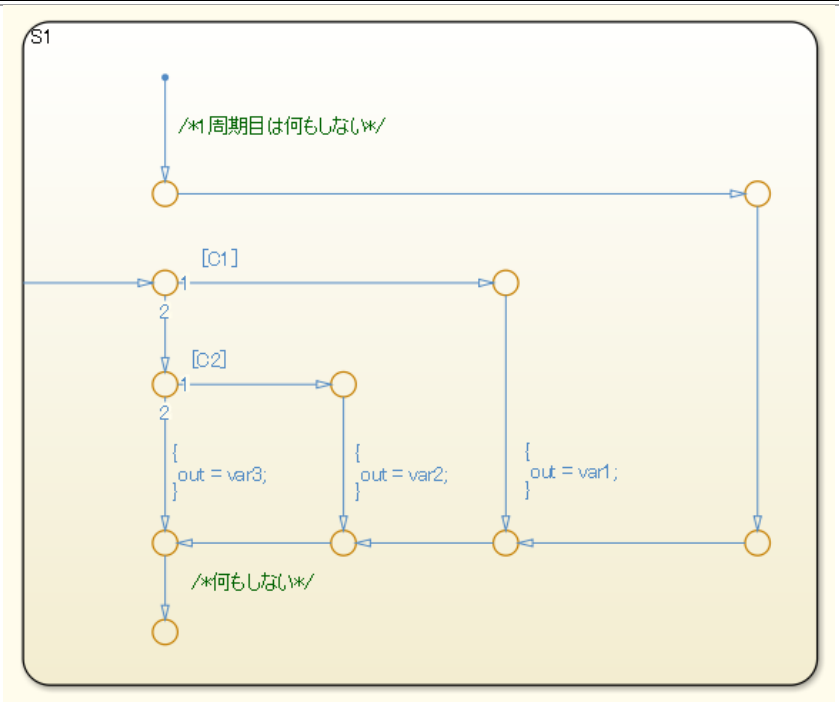
【誤】
始点がステートの左端ではなく上端です。

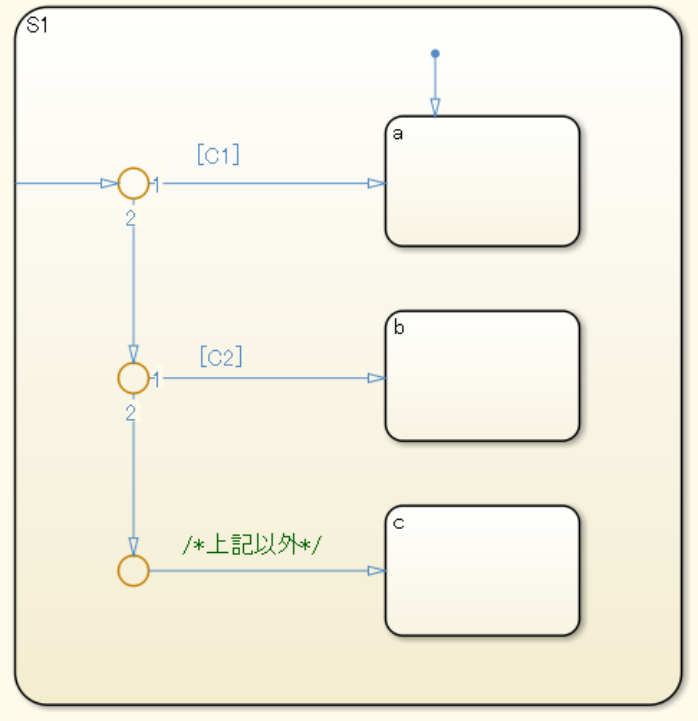


根拠	
サブ ID	**記述内容**
a	・定められた信号線フロー記述規則に従うことで、可読性が向上します。

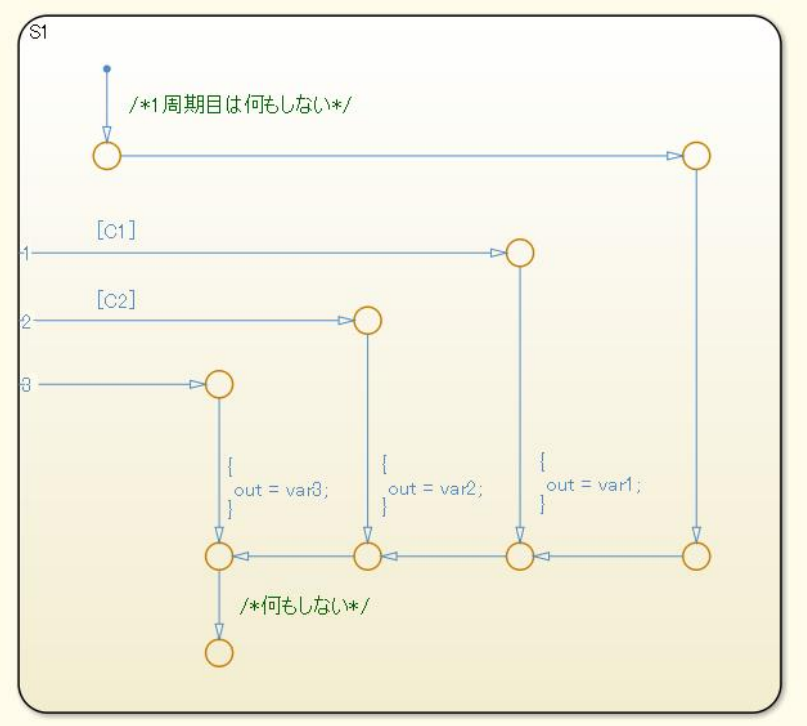
4.2.9. jc_0763：複数の内部遷移の記述方法

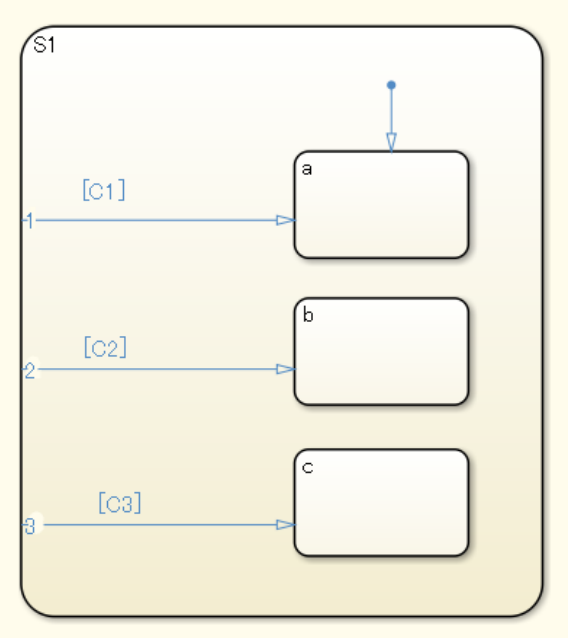
ルール ID：タイトル	**jc_0763：複数の内部遷移の記述方法**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a1	1 つのステート内で、複数の内部遷移を使用しません。	-
	【正】	



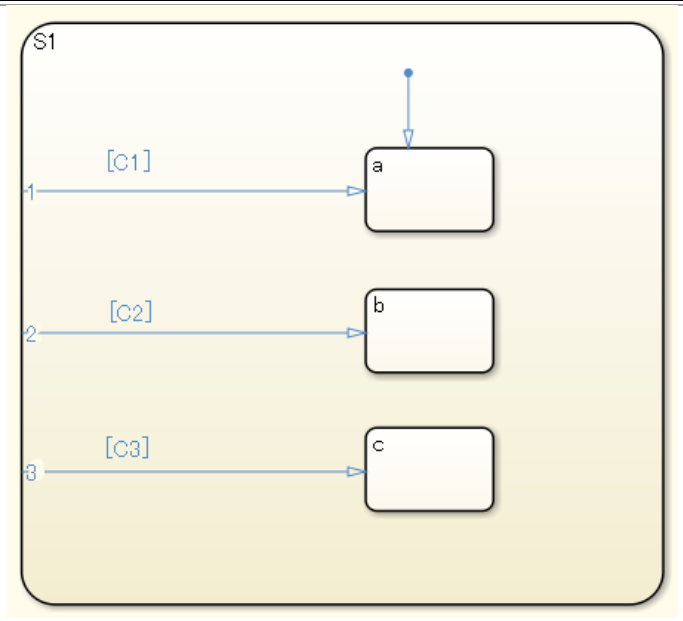


【誤】

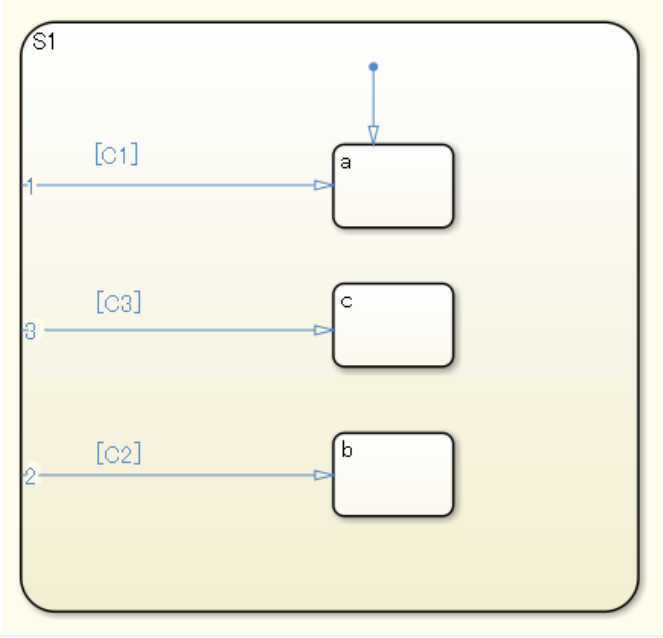




a2	1つのステート内で、複数の内部遷移を使用する場合、内部遷移の実行順序に応じて上から下に並べます。	-
	【正】	



【誤】

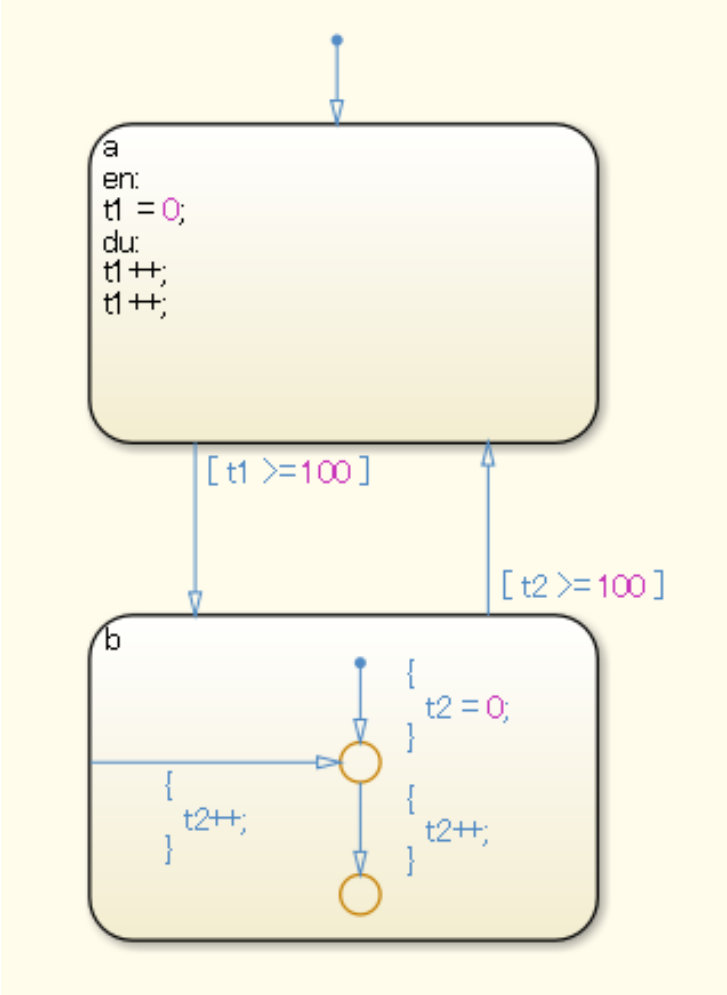


根拠	
サブ ID	**記述内容**
a1	・複数の内部遷移を用いると、遷移条件の個数が不明確となります。内部遷移の使用を 1 つに制限することで遷移が明確になるため、可読性が向上します。
a2	・複数の内部遷移により、遷移線の交差を防止し状態遷移をシンプルに表現できます。 また、内部遷移を実行順序に応じて並べることで、可読性が向上します。

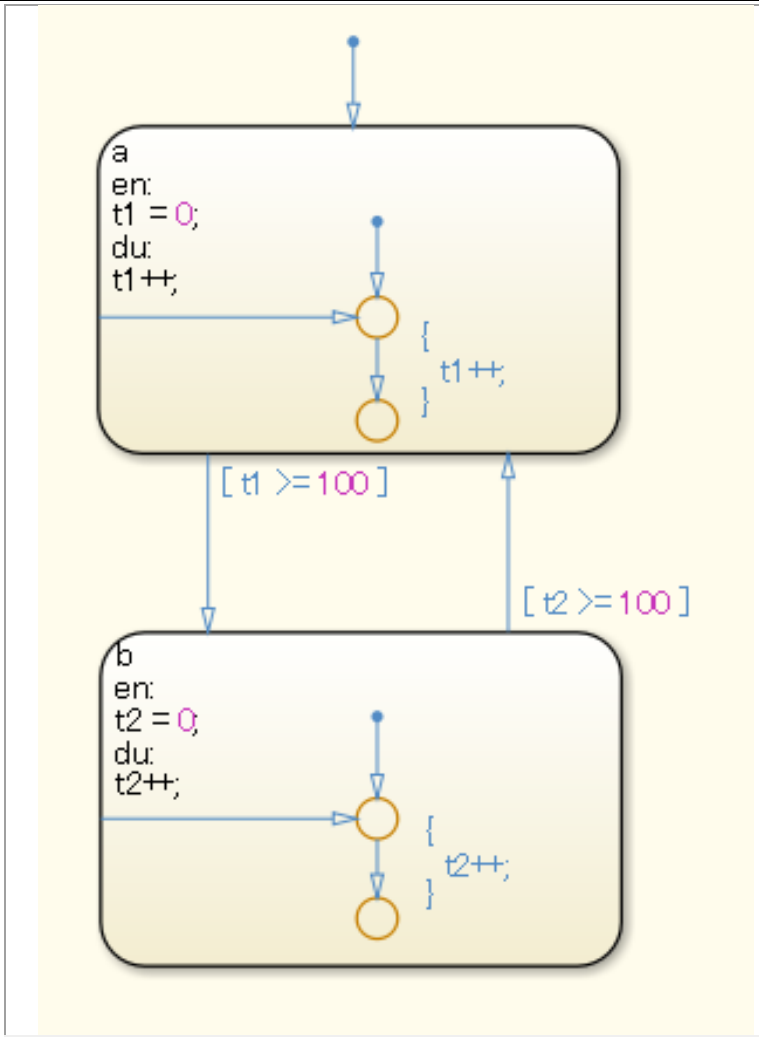
4.2.10. jc_0762 : ステートアクションタイプとフローチャート記述の併用禁止

ルール ID : タイトル	**jc_0762 : ステートアクションタイプとフローチャート記述の併用禁止**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	ステート内にステートアクションタイプとフローチャート記述を併用しません。	-
	【正】	

ステート内はステートアクションタイプ、またはフローチャート記述のいずれかで統一されています。



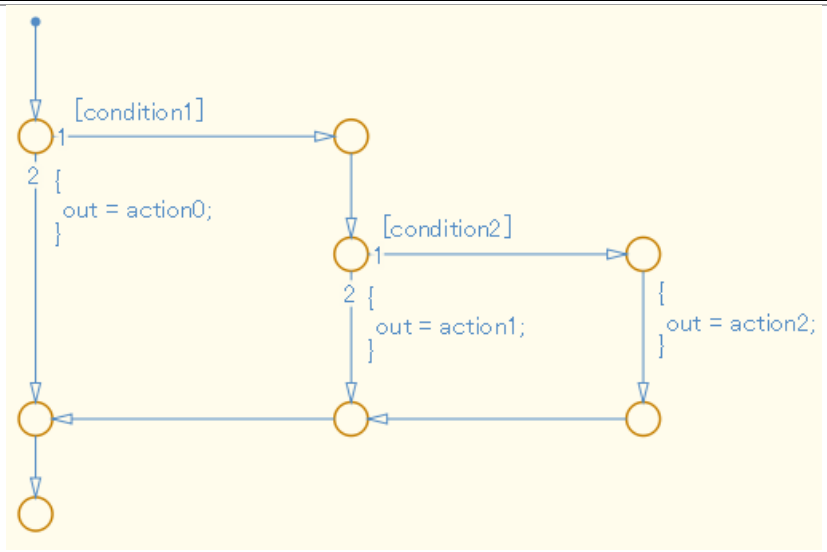
【誤】
ステート内でステートアクションタイプとフローチャート記述が併用されています。

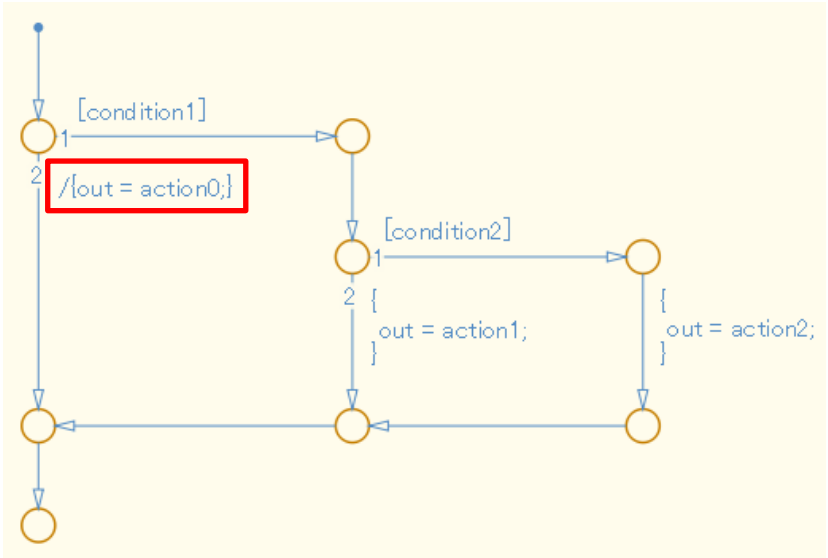


根拠	
サブ ID	**記述内容**
a	・ステートアクションタイプとフローチャート記述を併用すると、実行順序が解りにくく、可読性が低下します。

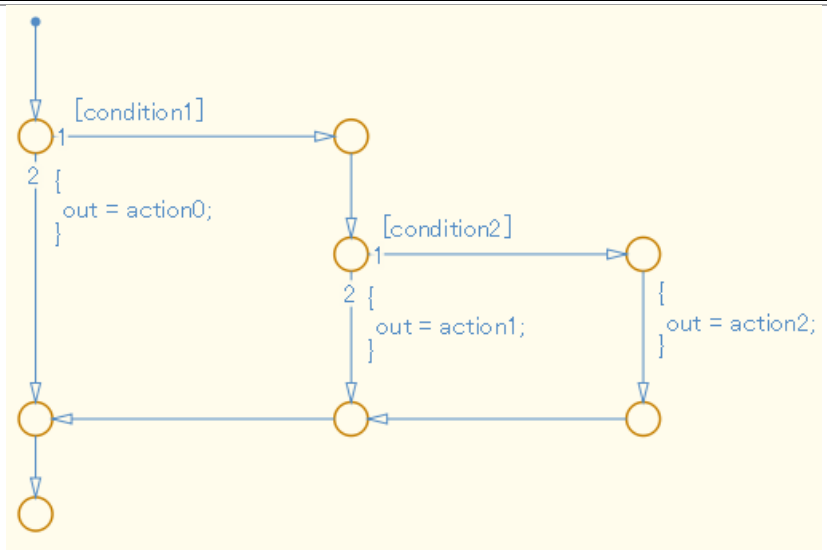
4.2.11. db_0132：フローチャートの遷移

ルール ID：タイトル	**db_0132：フローチャートの遷移**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	フローチャートには遷移アクションは使用しません。	-
	【正】 遷移アクションを使用していません。	

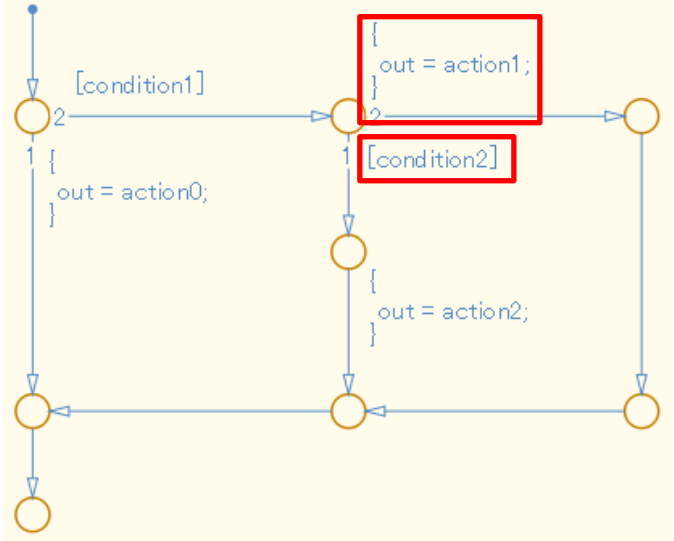
<table>
<tr><td></td><td> 

【誤】
遷移アクションを使用しています。
 
</td><td></td></tr>
<tr><td>b</td><td>フローチャートでは、条件は水平な遷移線に、条件アクションは垂直な遷移線に記述します。

＜例外＞
・ループ構文におけるななめに引かれた遷移線</td><td>-</td></tr>
<tr><td></td><td colspan="2">【正】
条件を水平な遷移線に、条件アクションを垂直な遷移線に記述しています。</td></tr>
</table>



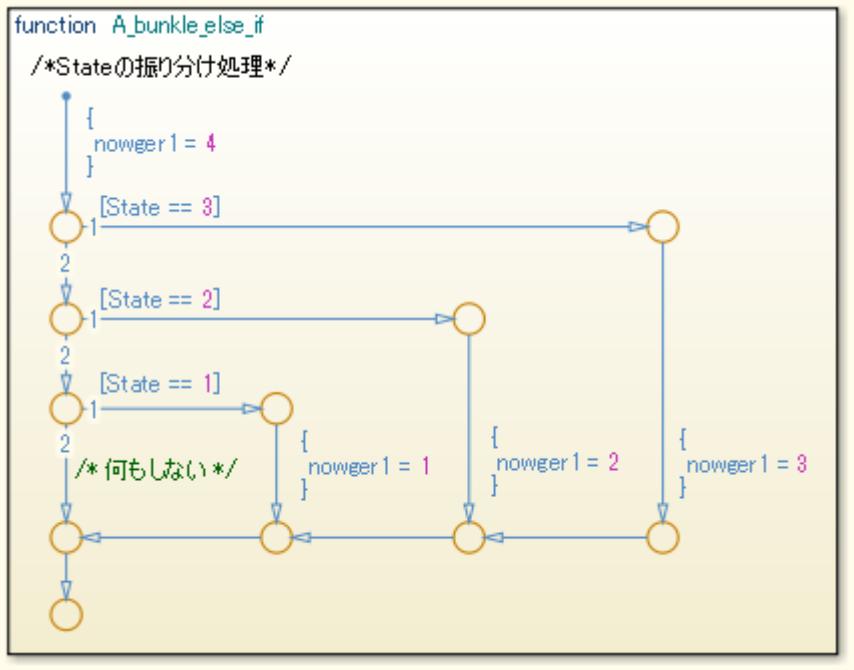
【誤】
条件を垂直な遷移線に、条件アクションを水平な遷移線に記述しています。



根拠	
サブ ID	**記述内容**
a	・フローチャートに遷移アクションを記述してもそのアクションは実行されません。
b	・条件と条件アクションの記述位置を統一することで、可読性が向上します。

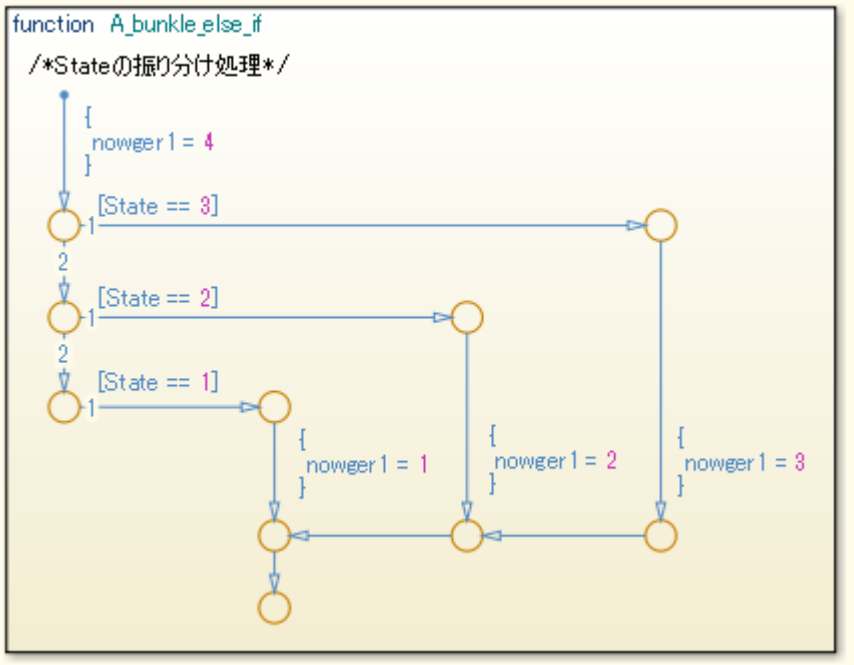
4.2.12. jc_0773：フローチャートの無条件遷移

ルール ID：タイトル	**jc_0773：フローチャートの無条件遷移**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	コネクティブジャンクションを起点にした遷移条件付きの遷移線がある場合には、必ず、そのコネクティブジャンクションを起点にした無条件遷移線を設けます。	-
	【正】	

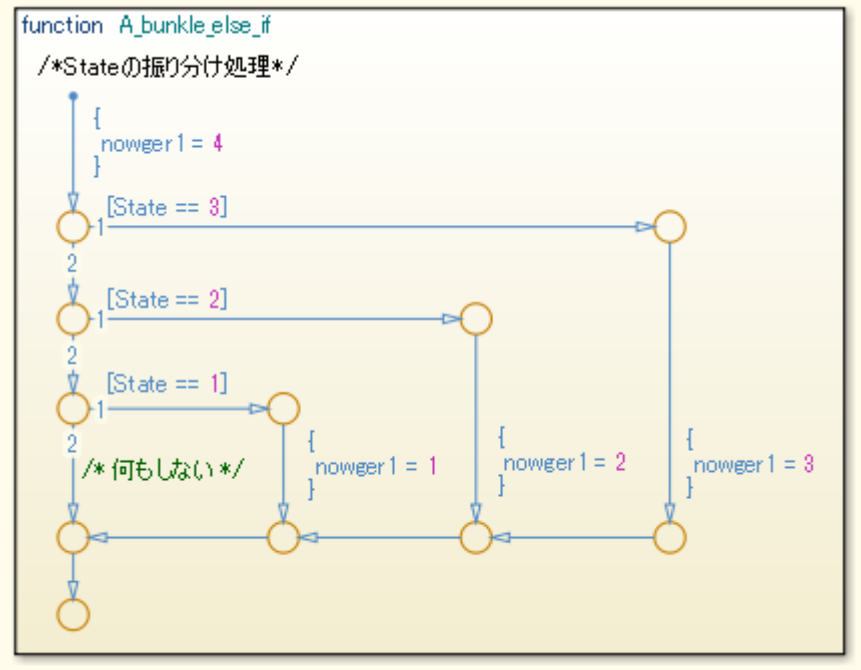


【誤】

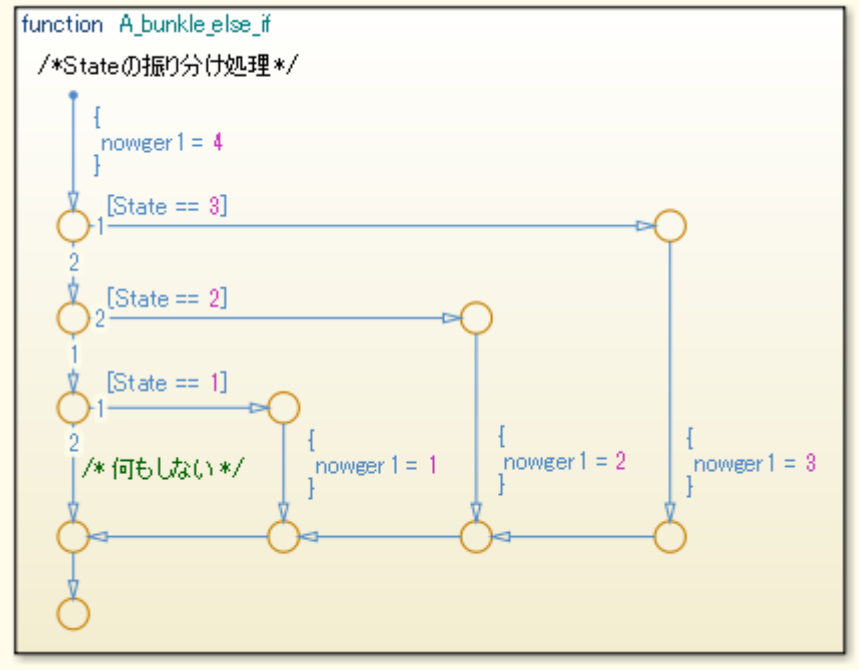
無条件遷移線がありません。



b	無条件遷移の{実行順序}に一番最後の値を設定します。	-
	【正】	



【誤】
無条件遷移線の実行順序が最後になっていません。

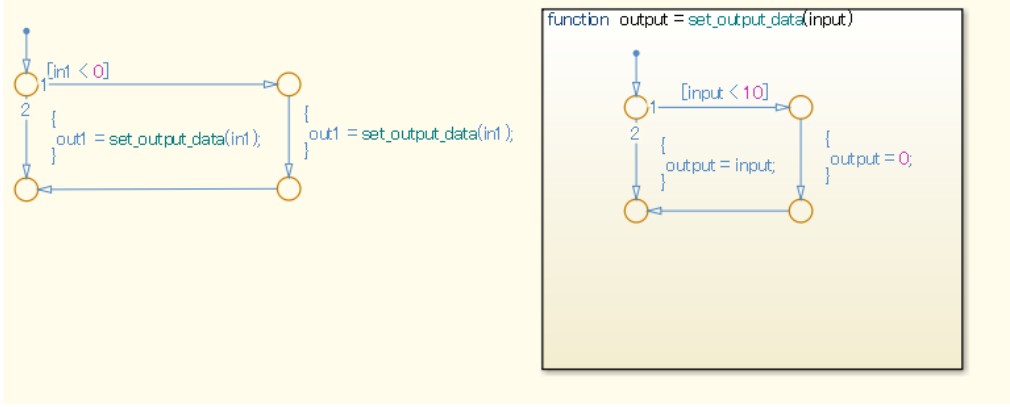


根拠	
サブ ID	**記述内容**
a	・バックトラックによる意図しない動作を防止します。 また、無条件遷移を設けることで条件を満たさないときの動作を明示的にします。
b	・無条件遷移が優先されることによる意図しない動作を防止することができます。

4.2.13. jc_0775 : フローチャートの終端コネクティブジャンクション

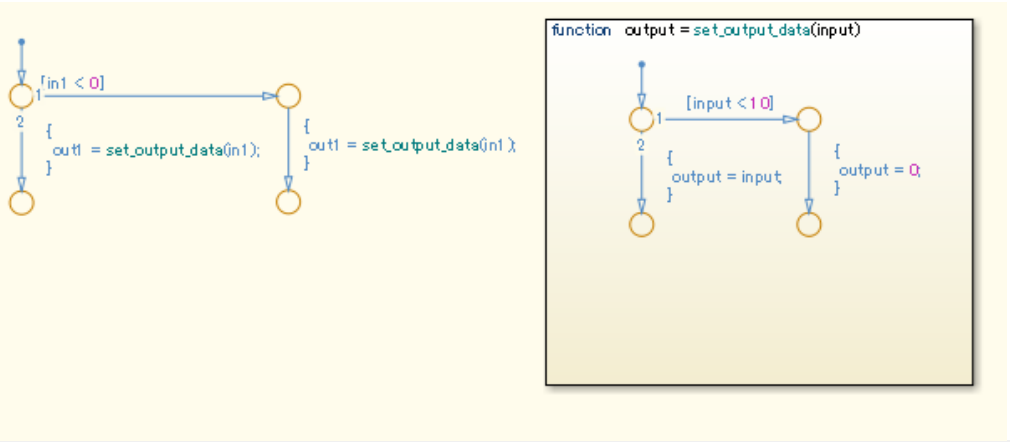
ルール ID : タイトル	**jc_0775 : フローチャートの終端コネクティブジャンクション**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a1	終端コネクティブジャンクションは 1 つとします。	-

【正】



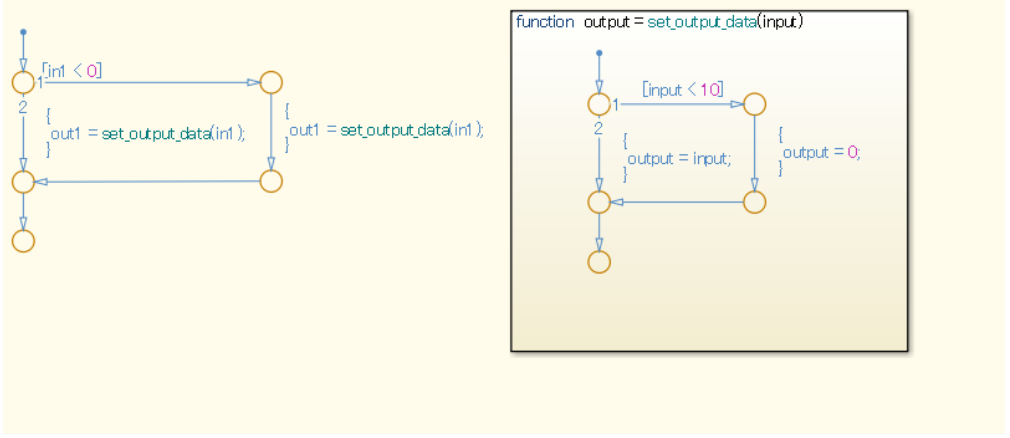
【誤】

終端コネクティブジャンクションが複数あります。



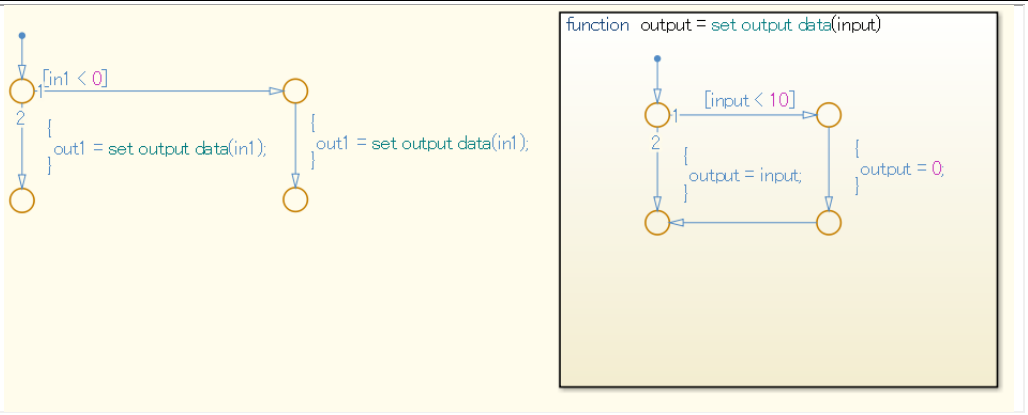
a2	終端コネクティブジャンクションは 1 つとし、終端コネクティブジャンクションへの入力は 1 つの無条件遷移のみとします。	-

【正】



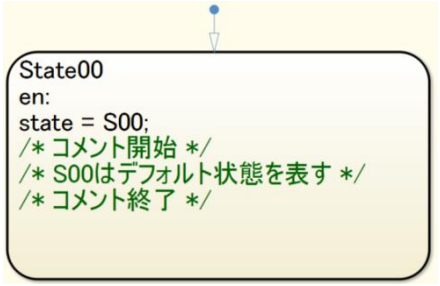
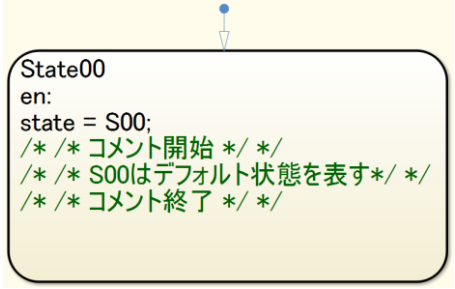
【誤】

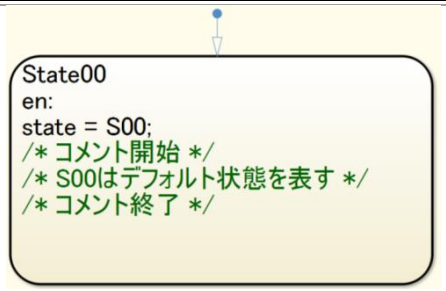
終端コネクティブジャンクションおよび終端コネクティブジャンクションへの入力が複数あります。



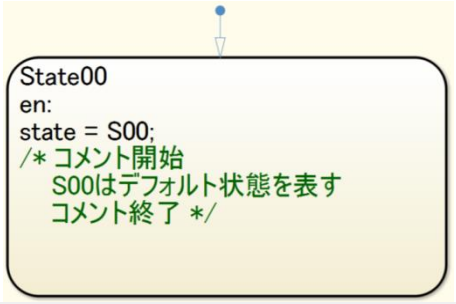
根拠	
サブ ID	**記述内容**
a1a2	・フローチャートの終端をまとめることによって処理の終端部分が把握しやすくなります。 ・終端コネクティブジャンクションの描き方を統一することにより、可読性が向上します。

4.2.14. jc_0738：Stateflow でのコメントの書き方

ルール ID：タイトル	**jc_0738：Stateflow でのコメントの書き方**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	{アクション言語}が“C”の場合、/*…*/コメントはネストしません。	-
	【正】  【誤】 	
b	{アクション言語}が“C”の場合、コメントに/* */を使用する場合は、途中で改行しません。	-
	【正】	



【誤】



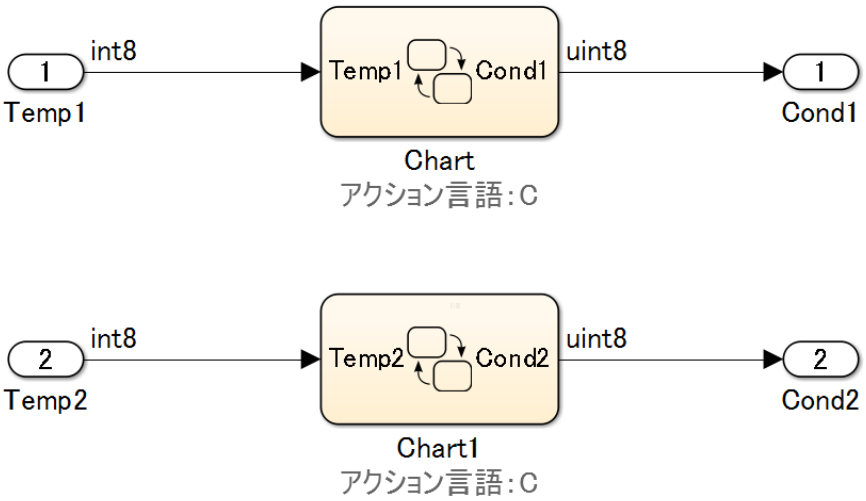
根拠	
サブ ID	**記述内容**
a	本来コメントである箇所をコンパイラがプログラムと誤解釈する可能性があります。
b	・途中で改行を入れた場合、編集中の箇所がコメント中であることに気付きにくくなり、コメントをネストしてしまう可能性があります。 ・アクション言語が“MALTAB”の場合、コメントは%コメントしか使えません。

4.3. 遷移条件 / アクション

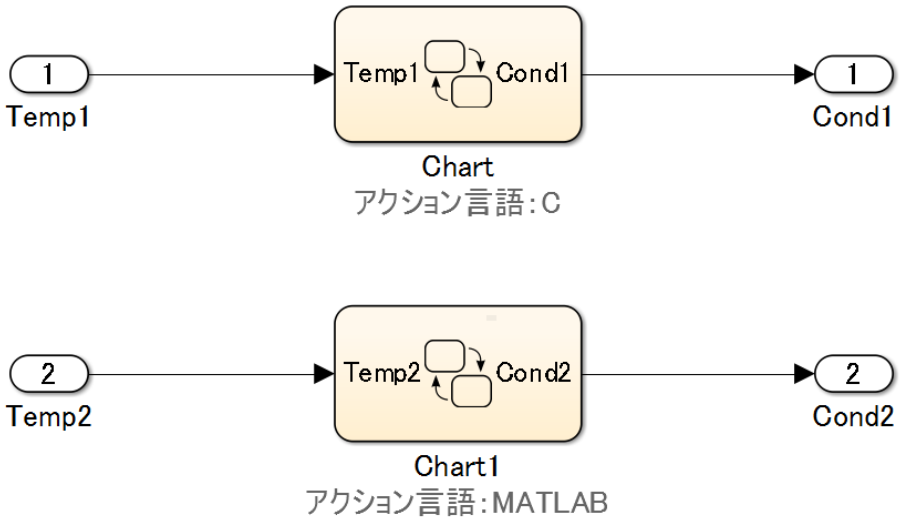
4.3.1. jc_0790 : Chart ブロックのアクション言語

ルール ID : タイトル	jc_0790 : Chart ブロックのアクション言語	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	[Chart]の{アクション言語}は"C"にします。	-

【正】
[Chart]の{アクション言語}は"C"に設定されています。



【誤】
[Chart1]の{アクション言語}が"C"に設定されていません。

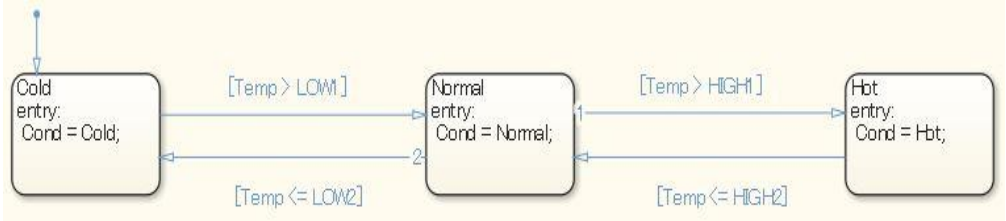


根拠	
サブ ID	**記述内容**
a	・アクション言語を統一することで、アクション言語構文として MATLAB と C の相違点に注意を払う必要がなく、可読性が向上します。 ・アクション言語が MATLAB の場合と比べて、モデルと生成コードとの整合が取りやすくなります。 ・C プログラマーがモデルを理解しやすくなります。

4.3.2. jc_0702：Stateflow のパラメーター / 定数名の設定

ルール ID：タイトル	**jc_0702：Stateflow のパラメーター / 定数名の設定**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	Stateflow ブロック内で直値を使用しません。 ＜例外＞ ・初期値の 0 ・インクリメント・デクリメントの 1	-

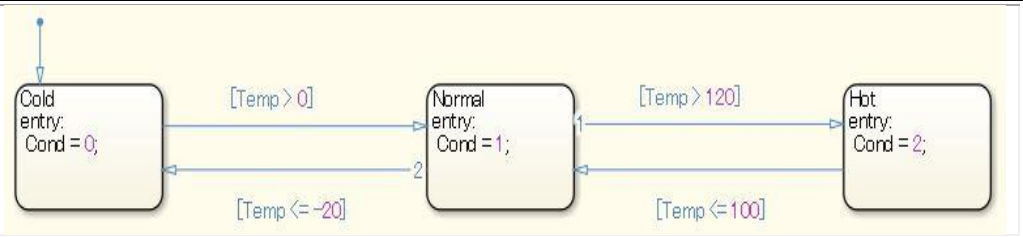
【正】
Stateflow ブロック内で直値を使用していません。



jc0702a_OK_parameter.m

```
%
HIGH1=100;
HIGH2=120;
LOW1=0;
LOW2=10;
```

【誤】
Stateflow ブロック内で直値を使用しています。

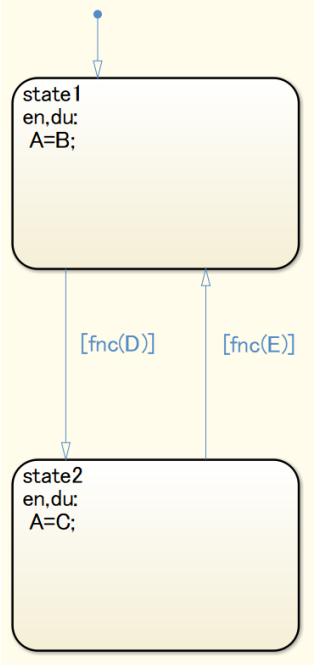


根拠	
サブ ID	**記述内容**
a	・定数が直値で書かれていると設計者以外には値の意図が解らず、可読性が損なわれます。 ・キャリブレーションを意図したものが直値でコード生成されます。

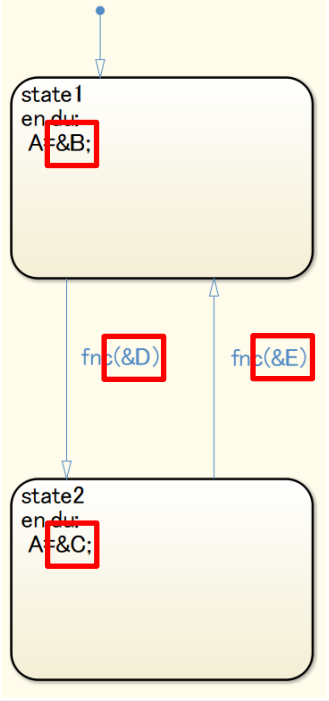
4.3.3. jm_0011：Stateflow のポインタ

ルール ID：タイトル	**jm_0011：Stateflow のポインタ**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	Stateflow ブロック内でポインタ変数を用いません。	-

【正】
Stateflow ブロック内でポインタ変数を使用していません。



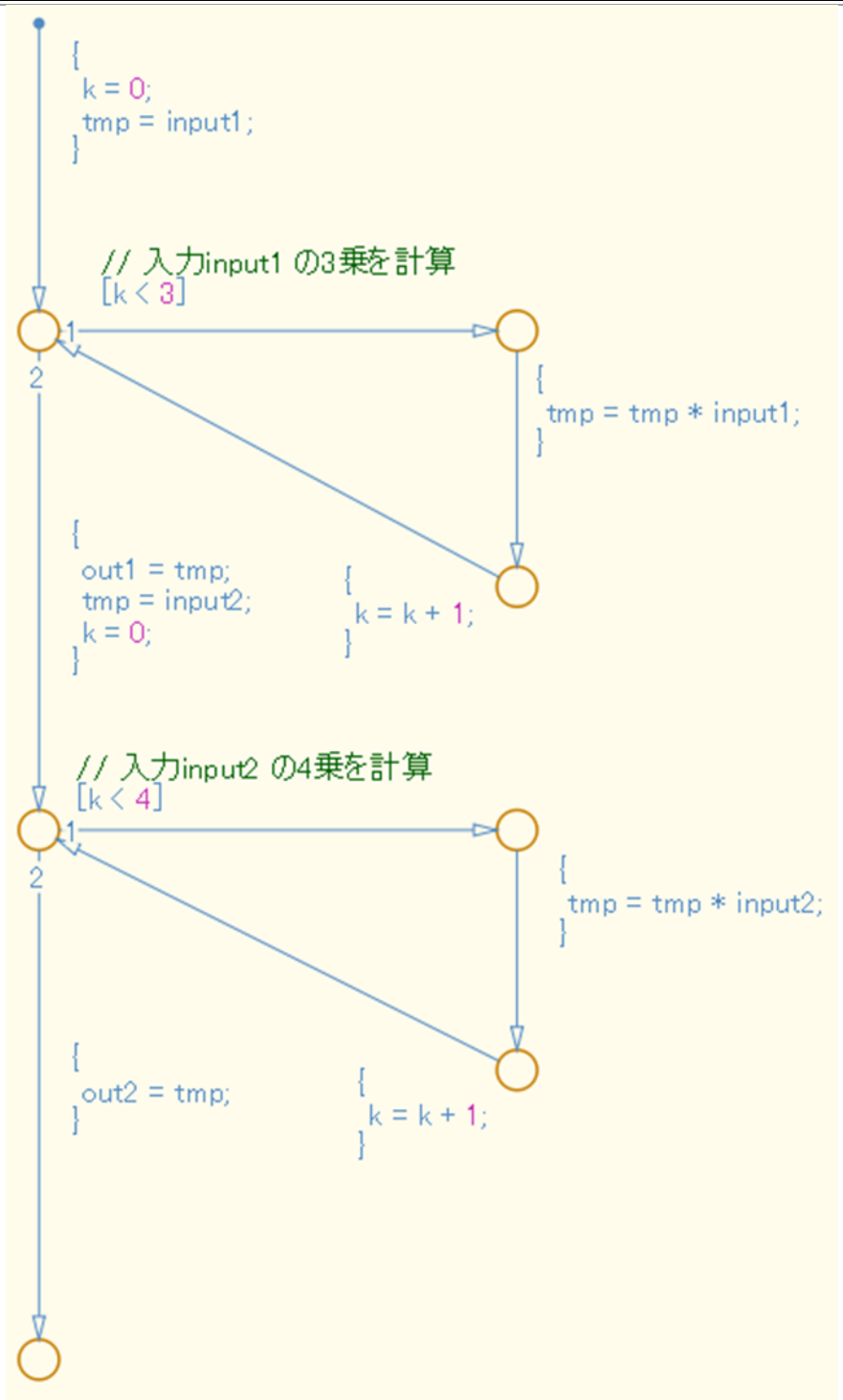
【誤】
Stateflow ブロック内でポインタ変数を使用しています。



根拠	
サブ ID	**記述内容**
a	・ポインタ変数を用いると可読性が損なわれます。 ・コード生成できない可能性があります。

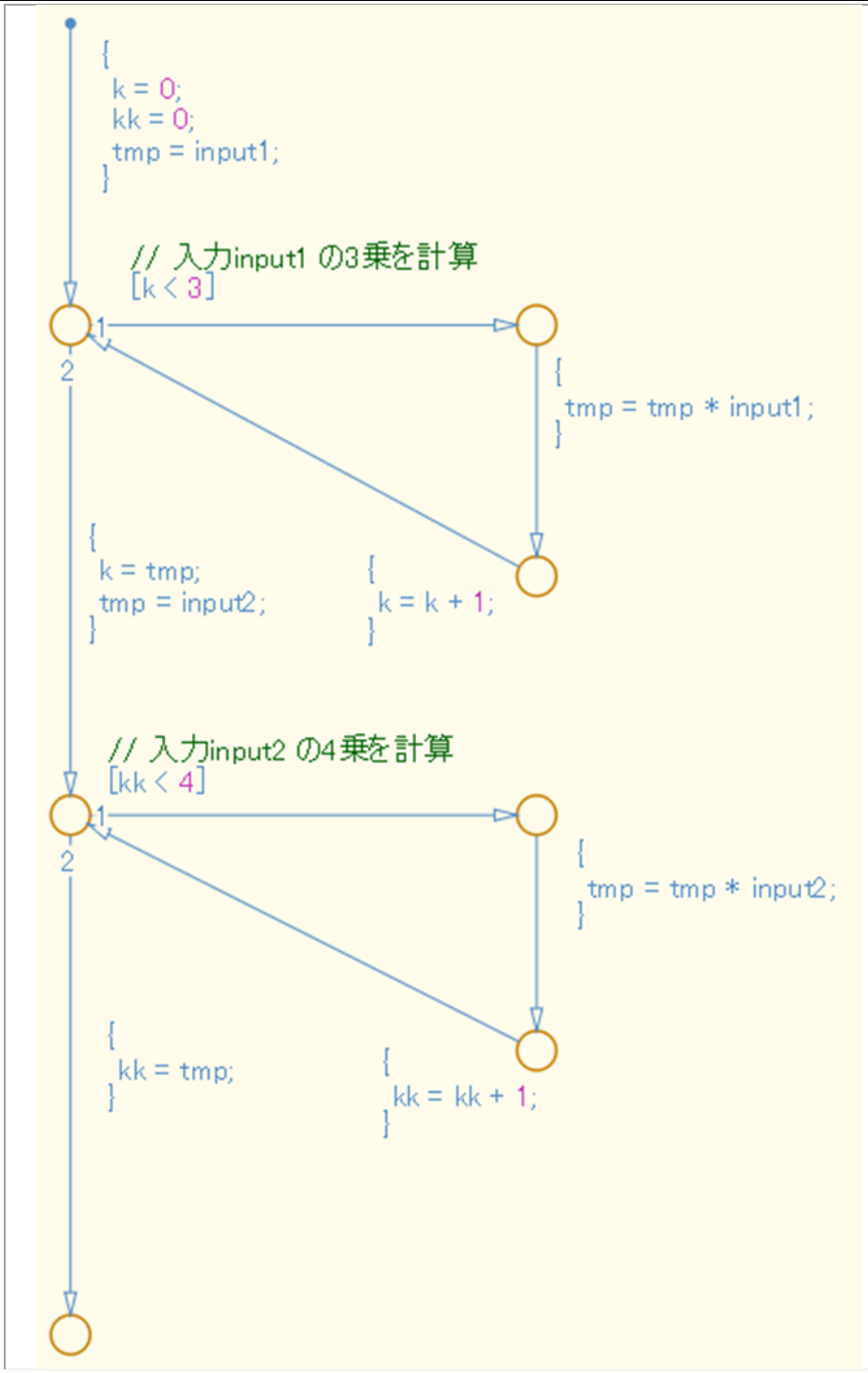
4.3.4. jc_0491：Stateflow におけるデータの再利用

ルール ID：タイトル	**jc_0491：Stateflow におけるデータの再利用**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	1 つの[Chart]内で同じ変数を複数の意味(用途)で使用しません。	-
	【正】 1 つの[Chart]内で同じ変数を複数の意味(用途)で使用していません。	



【誤】

1 つの[Chart]内で k、kk が複数の意味 (用途) で使用されています。

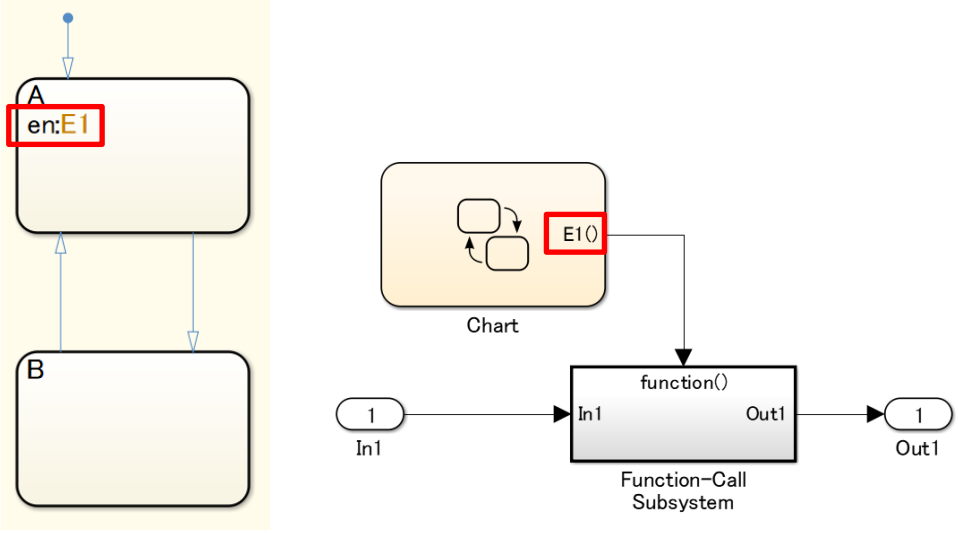


根拠	
サブ ID	**記述内容**
a	・変数につけられている名前とその変数に代入される数値の意味が異なると誤解釈をする可能性があります。

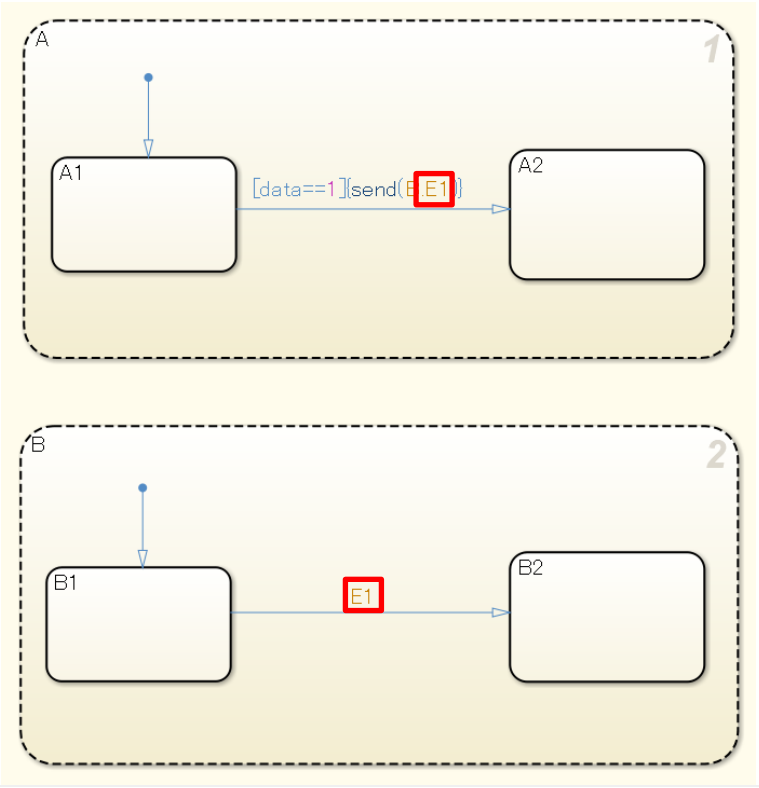
4.3.5. jm_0012 : イベントブロードキャストとイベントの使用制限

ルール ID : タイトル	**jm_0012 : イベント ブロードキャストとイベントの使用制限**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a1	Stateflow のイベントは Stateflow ブロックの出力でのみ使用します。	-

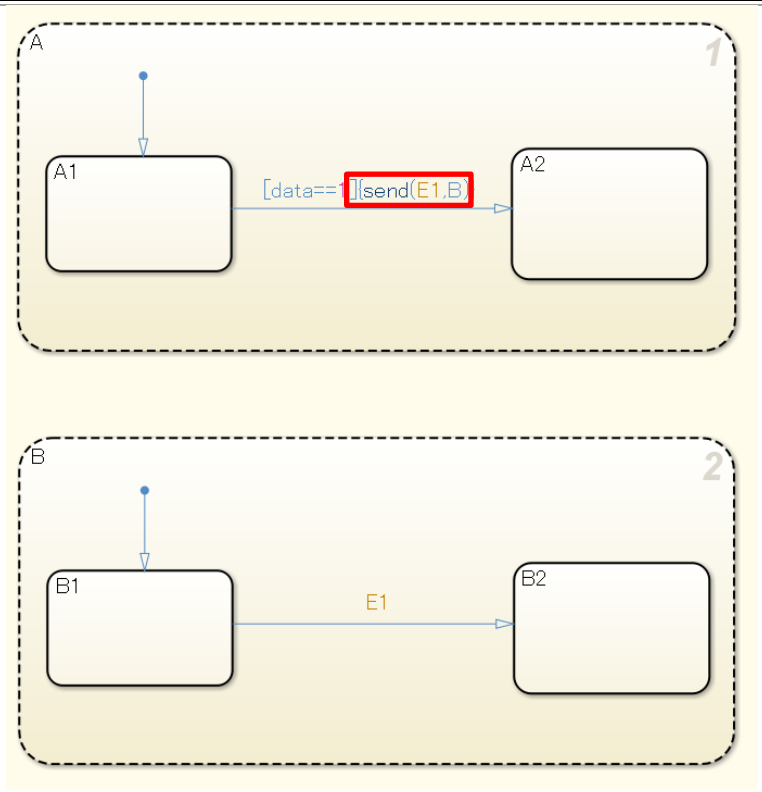
【正】
Stateflow ブロックの出力のみでイベントを使用します。



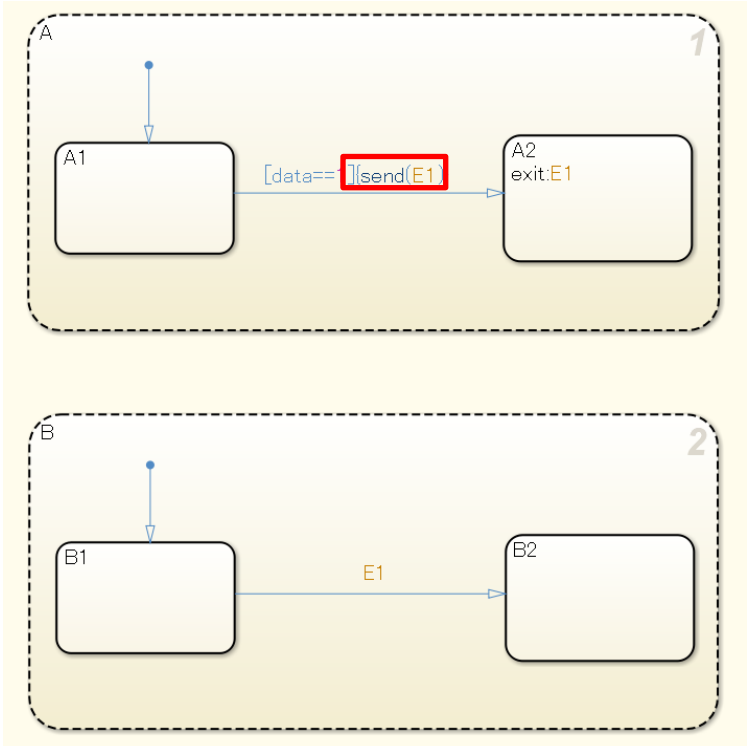
【誤】
Stateflow ブロックの出力以外でイベントを使用しています。



a2	Stateflow のイベントをブロードキャストするときは send 構文 send(event_name、state_name) を使用します。	-
	【正】 send 構文を使用して、イベントをブロードキャストします。	

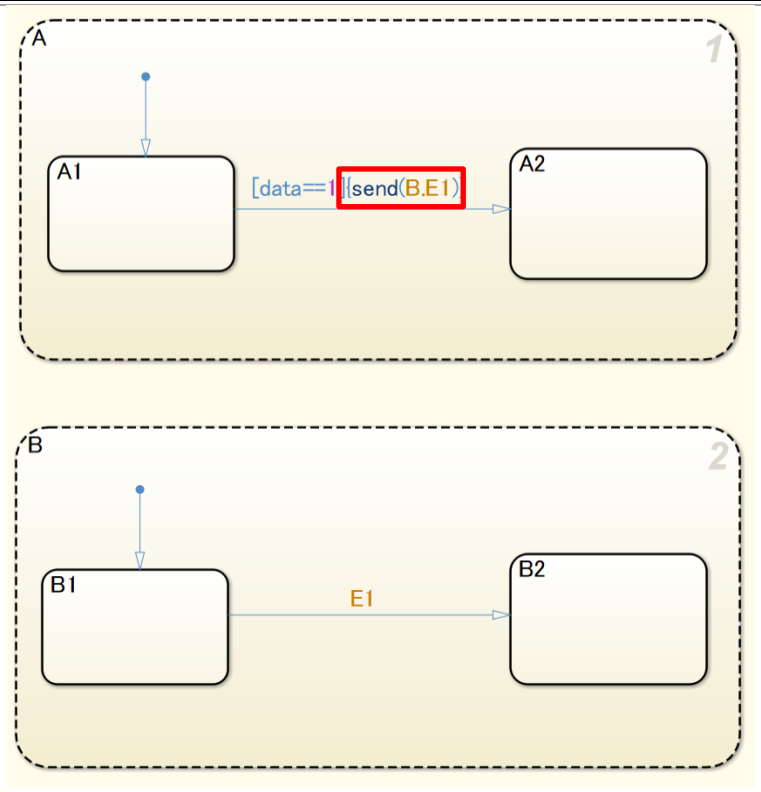


【誤】
send 構文として、受信側のステートが記述されていません。



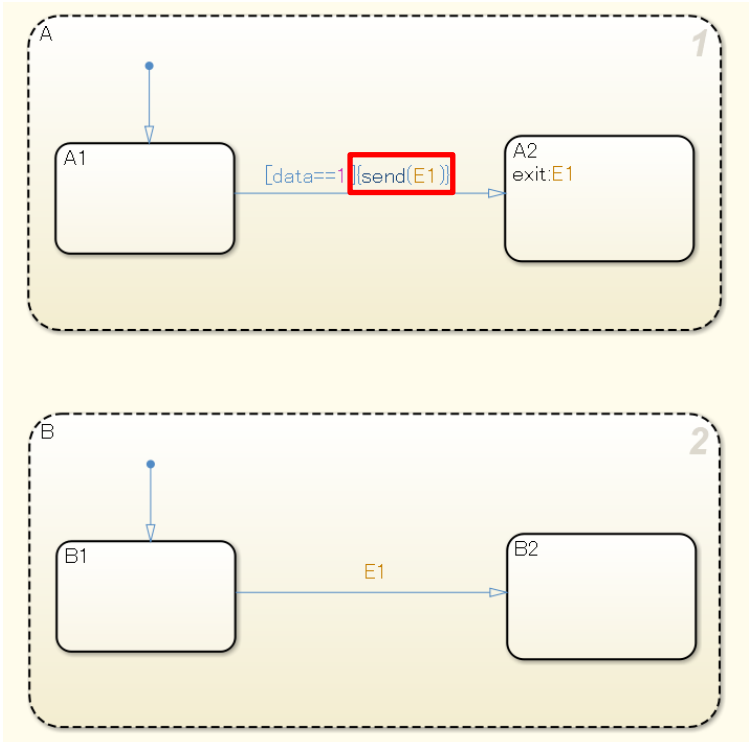
a3	Stateflow のイベントをブロードキャストするときは修飾イベント名を使用した send 構文 send(state_name.event_name) を使用します。	-

【正】
修飾イベント名を使用して、イベントをブロードキャストします。



【誤】

send 構文として、受信側のステートが記述されていません。



根拠	
サブ ID	**記述内容**
a1	・出力以外でイベントを使用しないことにより、[Chart]内における、再帰処理を排除することができます。
a2a3	・イベントにより行われる遷移が明確になり、可読性が向上します。

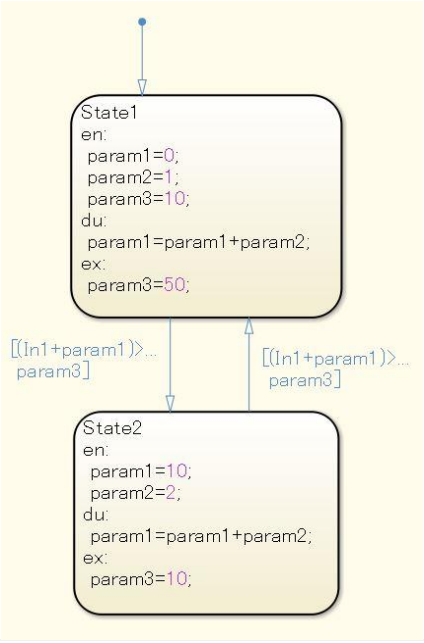
4.3.6. jc_0733 : ステートアクションタイプの記述順序

ルール ID : タイトル	jc_0733 : ステートアクションタイプの記述順序

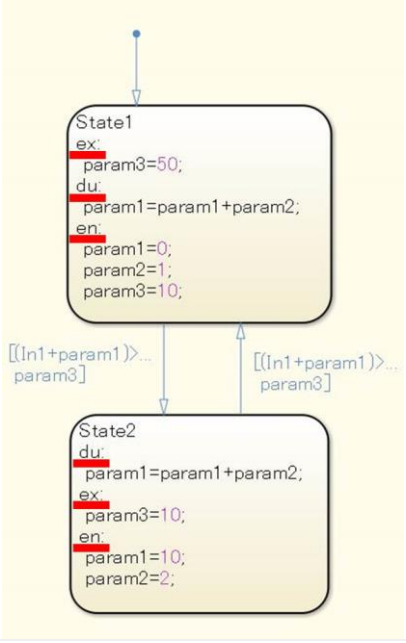
ルール

サブ ID	記述内容	カスタムパラメーター
a	基本ステートアクションタイプは、entry(en)、during(du)、exit(ex)の順序で記述します。	-

【正】
en、du、ex の順序で記述しています。

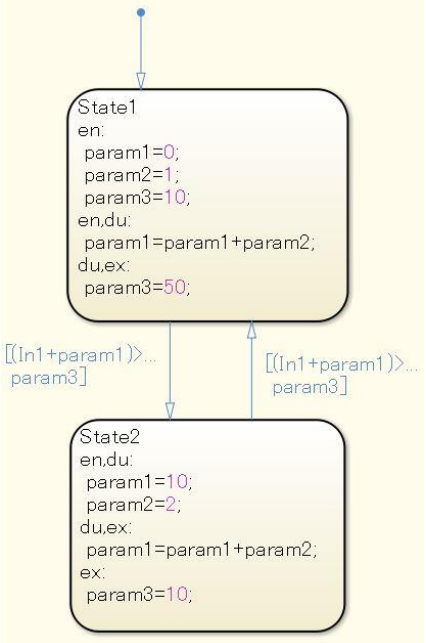


【誤】
en、du、ex の順序で記述していません。

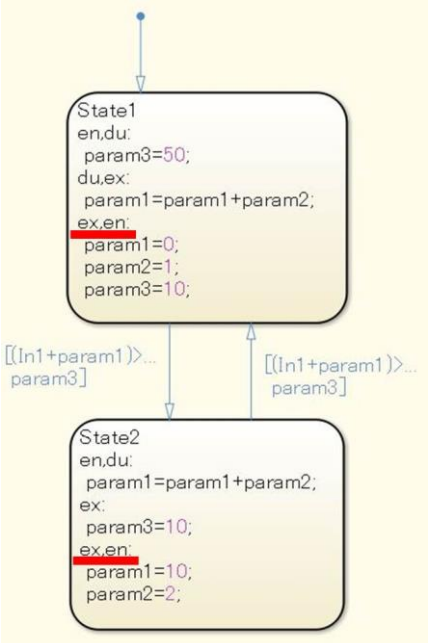


b	組み合わせステートアクションタイプを使用する場合は、entry(en)、during(du)、exit(ex)の順序で記述します。	-

【正】
en、du、ex の順序で記述しています。



【誤】
en、du、ex の順序で記述していません。



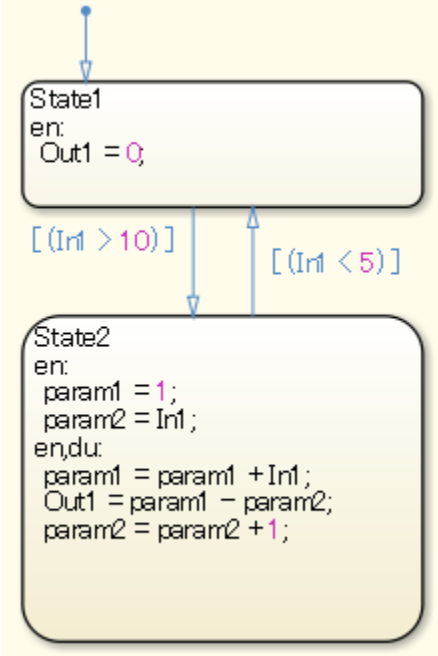
根拠

サブ ID	記述内容
ab	・一貫したモデリングで可読性、メンテナンス性を向上します。

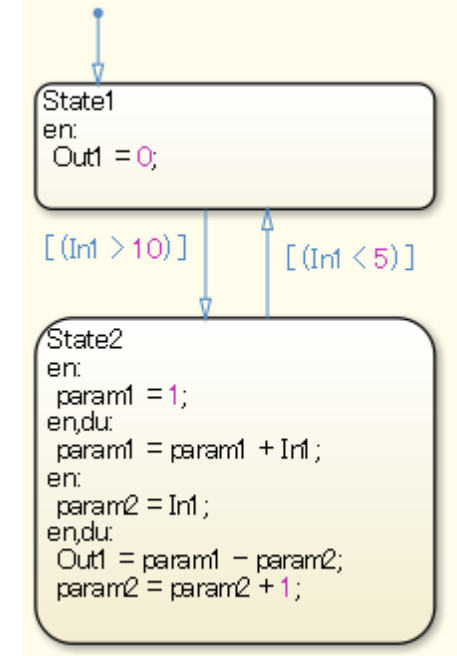
4.3.7. jc_0734 : ステートアクションタイプの記述回数

ルール ID : タイトル	jc_0734 : ステートアクションタイプの記述回数	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	ステートアクションタイプは、同じものを 2 回以上記述しません。	-

【正】



【誤】

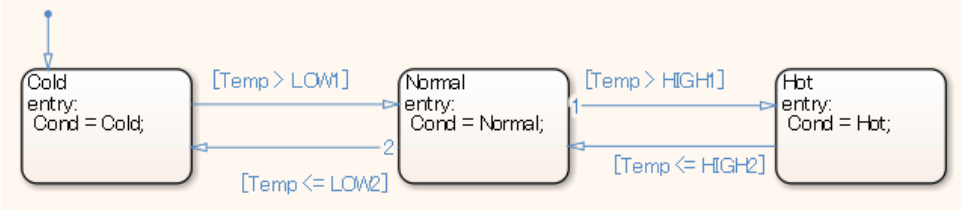


根拠	
サブ ID	**記述内容**
a	・記述した順序によって、実行順序が異なります。また、アクションタイプを複数回記述すると、実際の実行順序が解り辛くなります。

4.3.8. jc_0740 : ステートアクションタイプ exit の使用制限

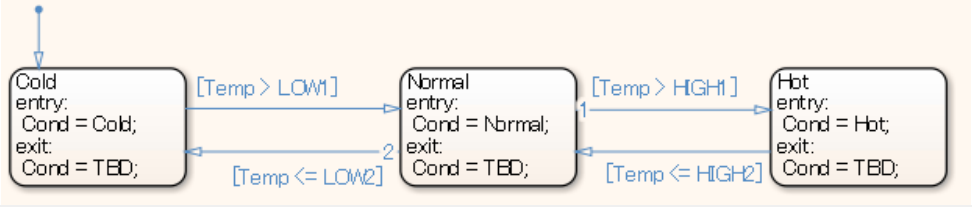
ルール ID : タイトル	jc_0740 : ステートアクションタイプ exit の使用制限	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	ステートアクションタイプ exit(ex)は使用しません。	-

【正】



【誤】

ステートアクションタイプ exit により TBD が出力されるように見えますが、実際には遷移先ステートのステートアクションタイプ entry で上書きされるため、[Chart]からは出力されません。

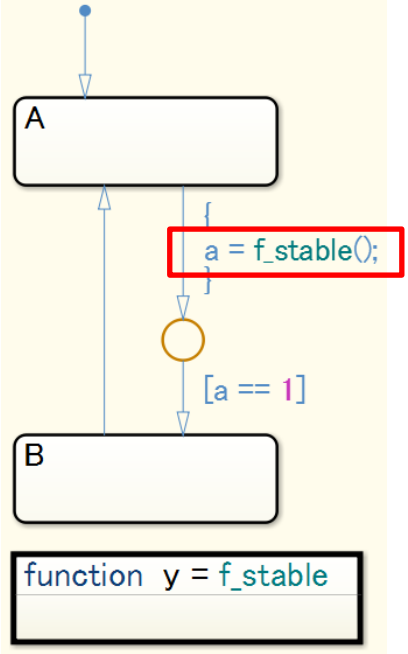


根拠	
サブ ID	**記述内容**
a	・ステートアクションタイプ exit(ex)を条件アクション、遷移アクション、ステートアクションタイプ entry(en)と併用した場合、実行タイミングが解りづらく、モデルの振る舞いの誤解釈を招く可能性があります。

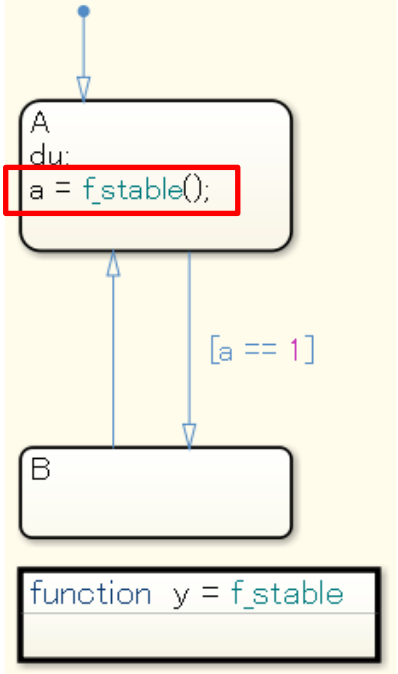
4.3.9. jc_0741 ： ステートチャートの遷移条件に使用するデータの更新タイミング

ルール ID : タイトル	**jc_0741 ： ステートチャートの遷移条件に使用するデータの更新タイミング**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	遷移条件に使用するデータは、during で更新を記述しません。	-

【正】
during で更新を行っていません。



【誤】
during で更新を行っています。

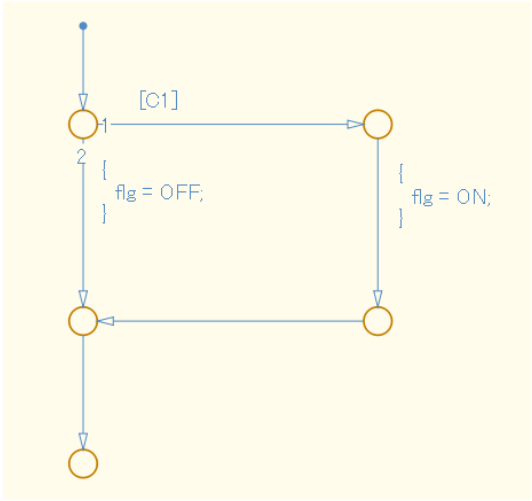


根拠	
サブ ID	**記述内容**
a	・遷移条件と during の実行順序が解りにくく、誤り混入のリスクがあります。

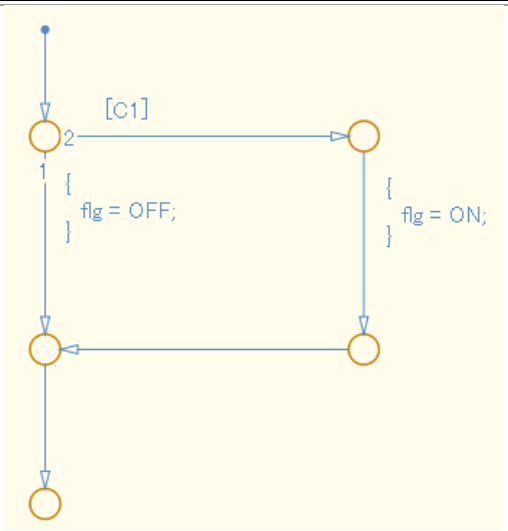
4.3.10. jc_0772：遷移線の実行順序と遷移条件

ルール ID：タイトル	**jc_0772：遷移線の実行順序と遷移条件**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	実行順序が最後以外の遷移は必ず条件を設定しなければなりません。 (2011b 以降は、コンフィギュレーションパラメーターの{診断} - {Stateflow} - {遷移の優先順位}を”エラー”にします。)	-

【正】

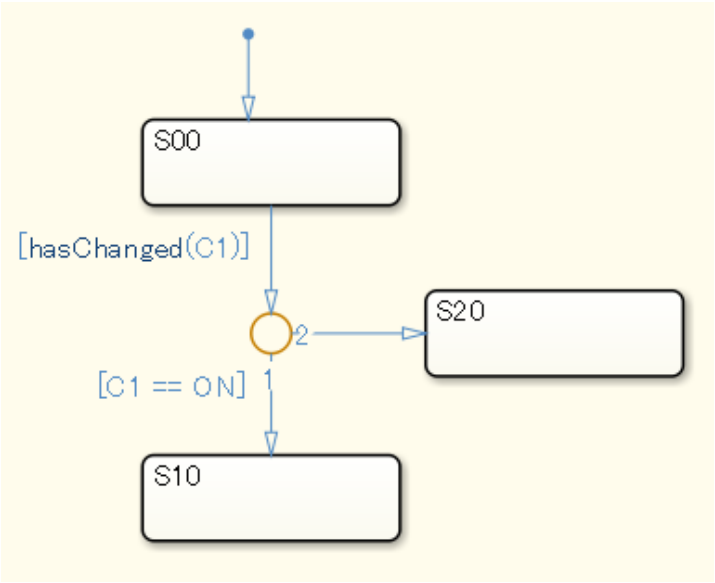


【誤】
実行順序 1 が無条件遷移で、実行条件 2 に条件式[C1]が記載されています。



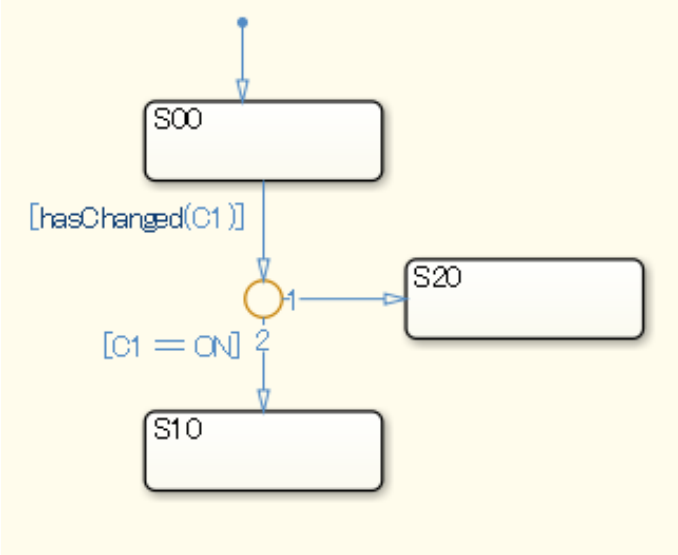
状態遷移の例

【正】



【誤】

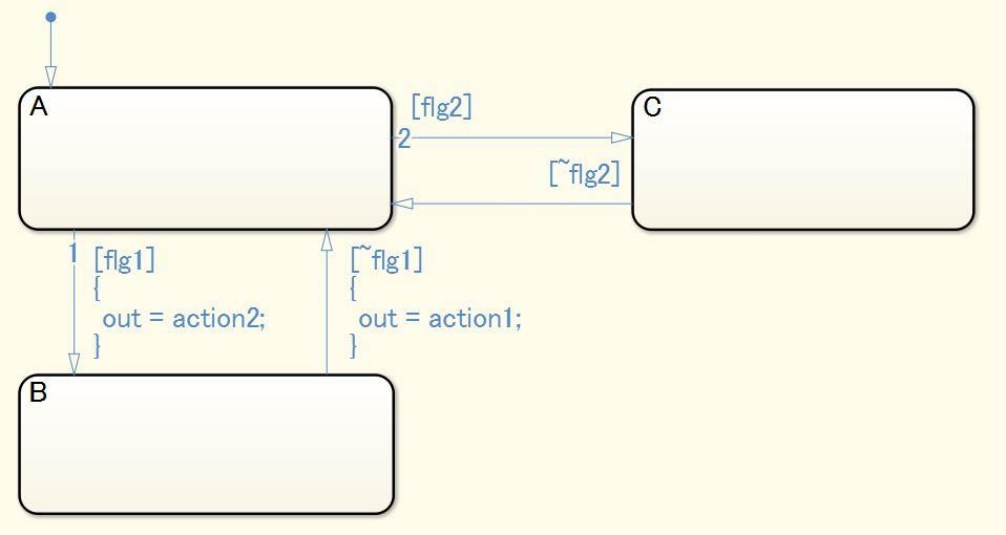
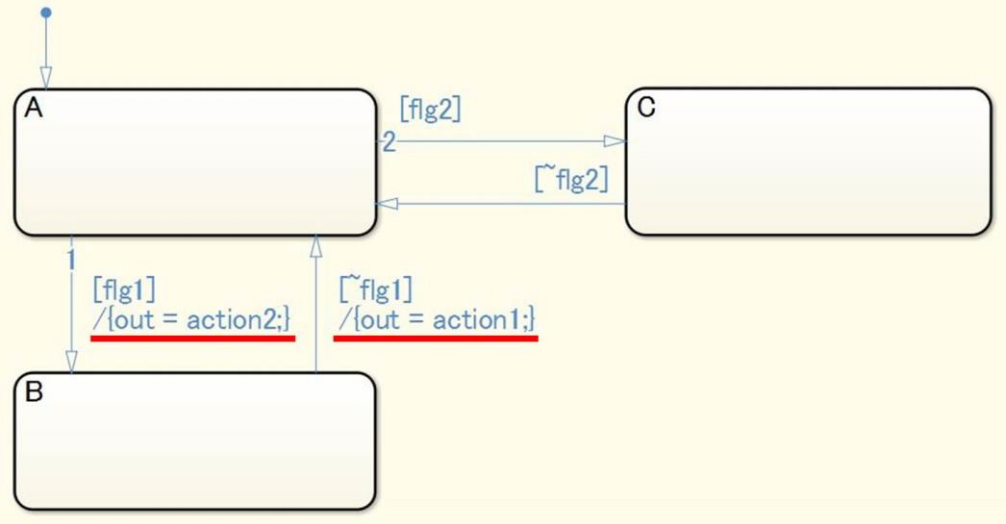
条件のある遷移線よりも、無条件遷移線の方が実行順序が高くなっています。

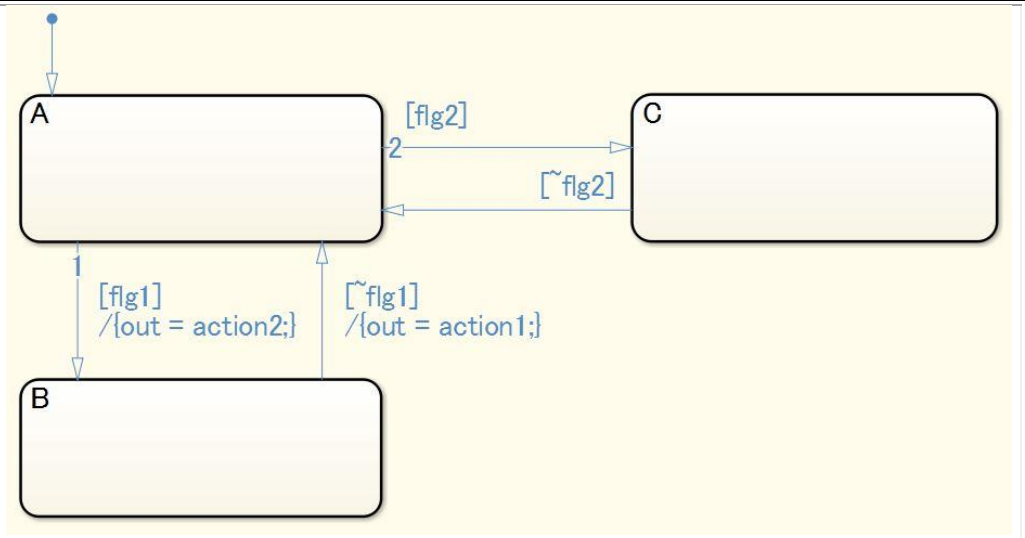


根拠

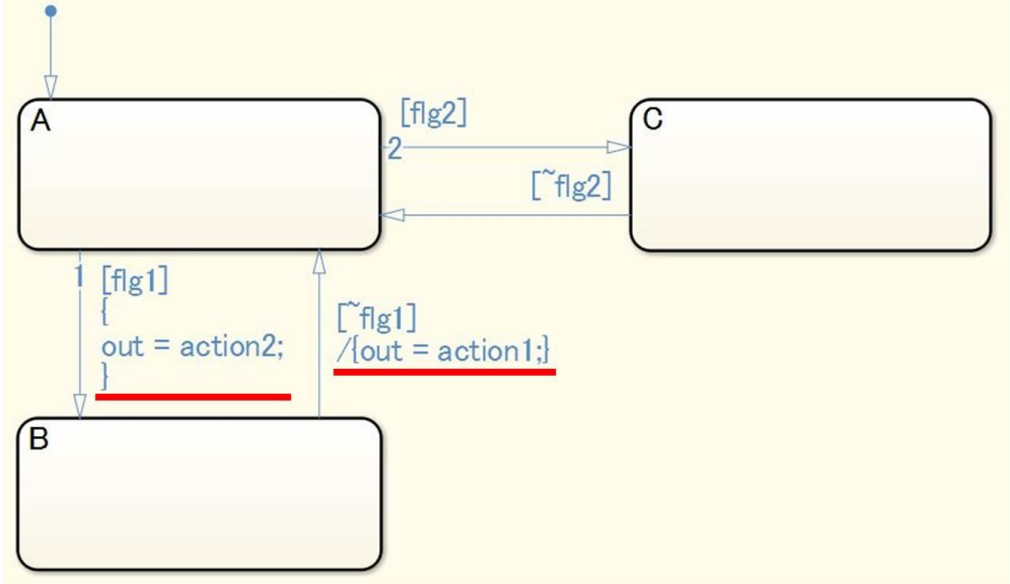
サブ ID	記述内容
a	・最後の実行順序以外で無条件遷移がある場合、それ以降の遷移が必ずデッドパスになるため、意図しないシミュレーション結果やコードが生成される可能性があります。

4.3.11. jc_0753 : Stateflow における条件アクションと遷移アクション

ルール ID : タイトル	jc_0753 : Stateflow における条件アクションと遷移アクション	
ルール		
サブ ID	記述内容	カスタムパラメーター
a1	ステートチャートには遷移アクションは使用しません。	-
	【正】 ステートチャートで条件アクションのみを使用しています。  【誤】 ステートチャートで遷移アクションを使用しています。  	
a2	条件アクションと遷移アクションは、同じ[Chart]内で混在させません。	-
	【正】 [Chart]内で条件アクションまたは遷移アクションのどちらか一方のみを使用しています。 (下図では遷移アクションのみを使用しています。)	



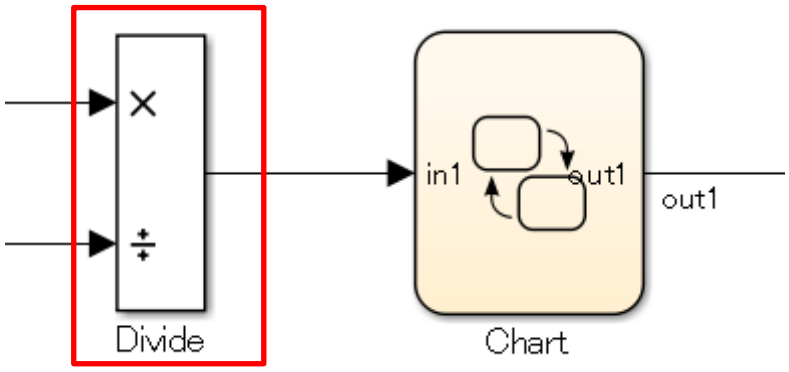
【誤】
[Chart]内で条件アクションと遷移アクションの両方を使用しています。

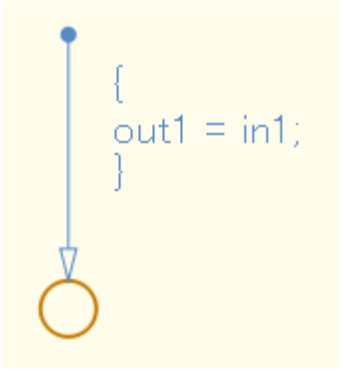


根拠	
サブ ID	**記述内容**
a1	・条件アクションとの勘違いと可読性が低下することを防止できます。
a2	・条件アクションと遷移アクションとの勘違いを防止するためです。条件アクションは遷移に入ったら実行され、遷移アクションは次の状態へ遷移することが決まってから実行されます。

4.3.12. jc_0711：Stateflow における除算

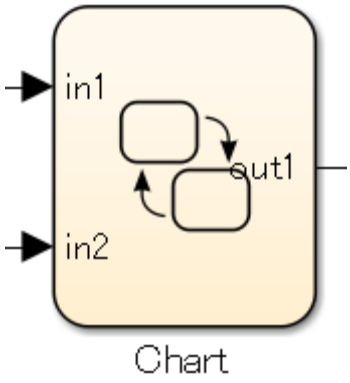
ルール ID：タイトル	jc_0711：Stateflow における除算	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a1	Stateflow ブロック内では、変数、定数、パラメーターで除算をしません。	-
	【正】 Stateflow ブロックの外で除算をしています。	

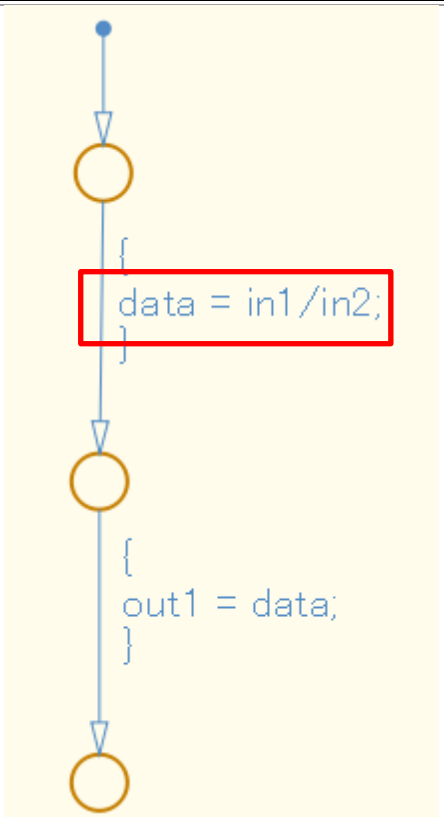
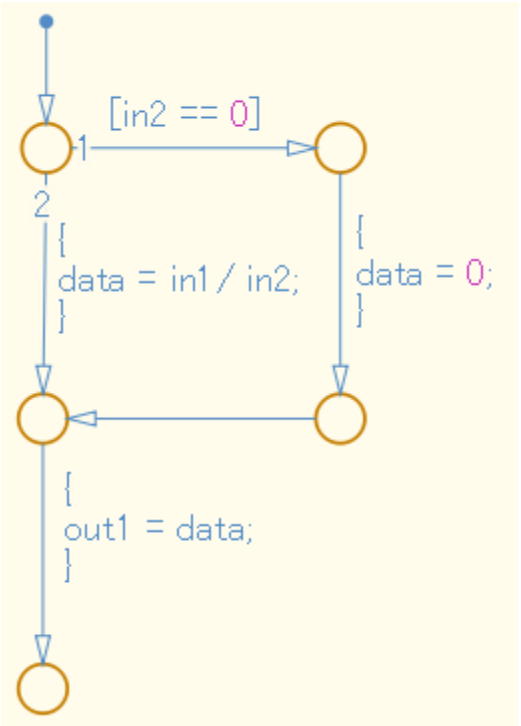




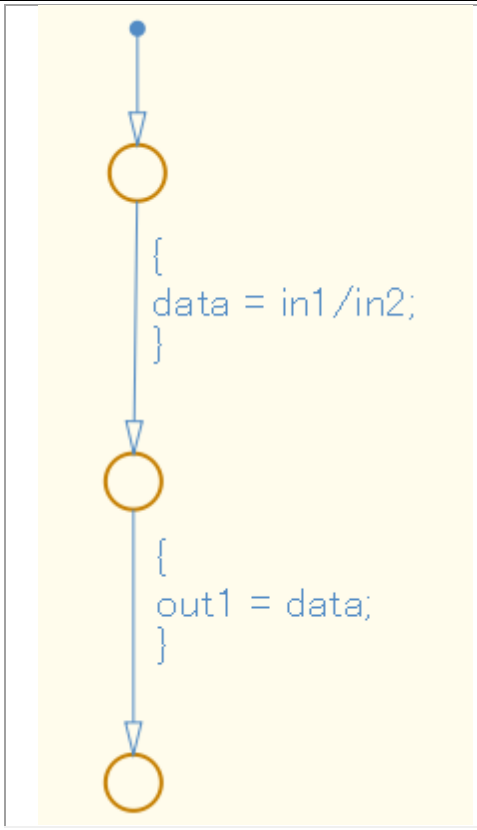
【誤】

Stateflow ブロック内で除算をしています。



<table>
<tr><td></td><td> 
</td><td></td></tr>
<tr><td>a2</td><td>Stateflow ブロック内で除算を行う場合、ゼロ割回避の処理を記述します。</td><td>-</td></tr>
<tr><td></td><td>【正】
ゼロ割回避の処理を入れています。
 

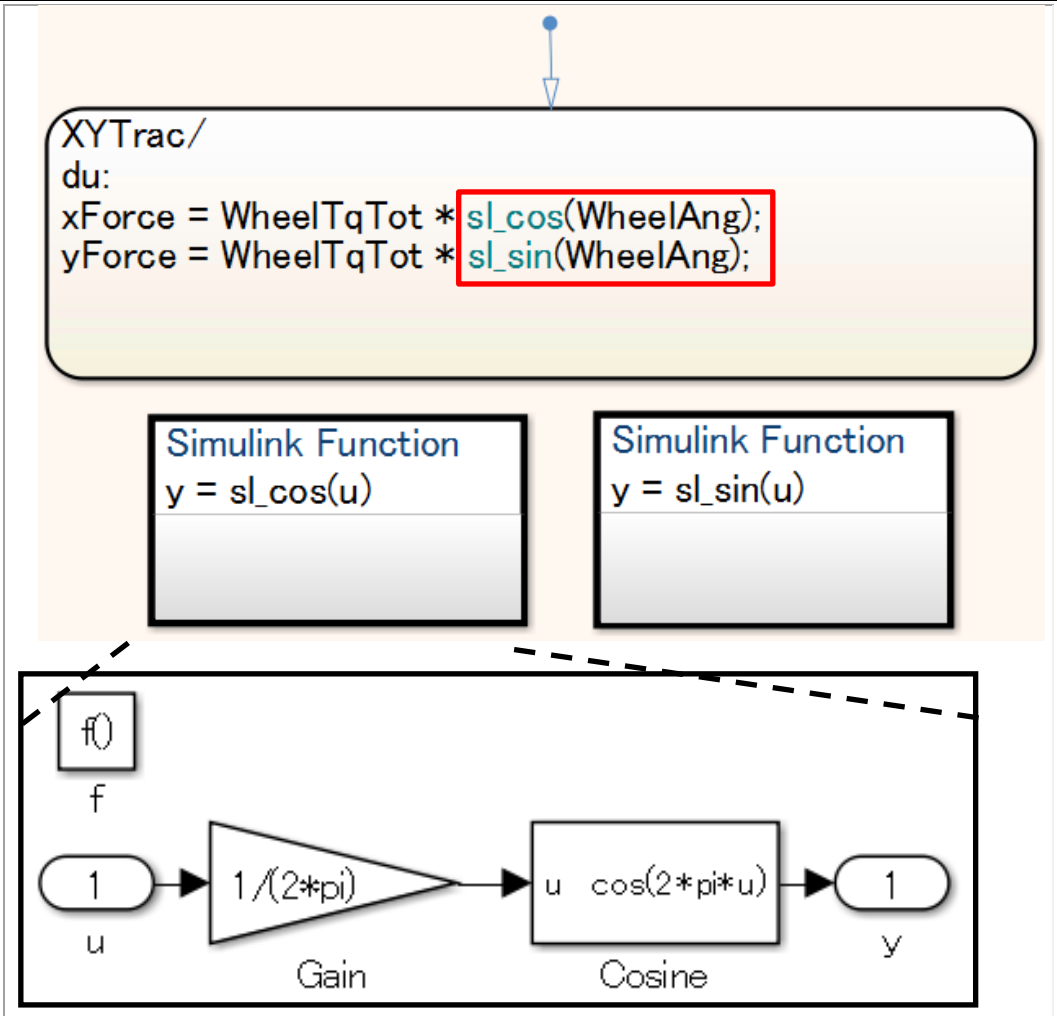
【誤】
ゼロ割回避の処理を入れていません。</td><td></td></tr>
</table>



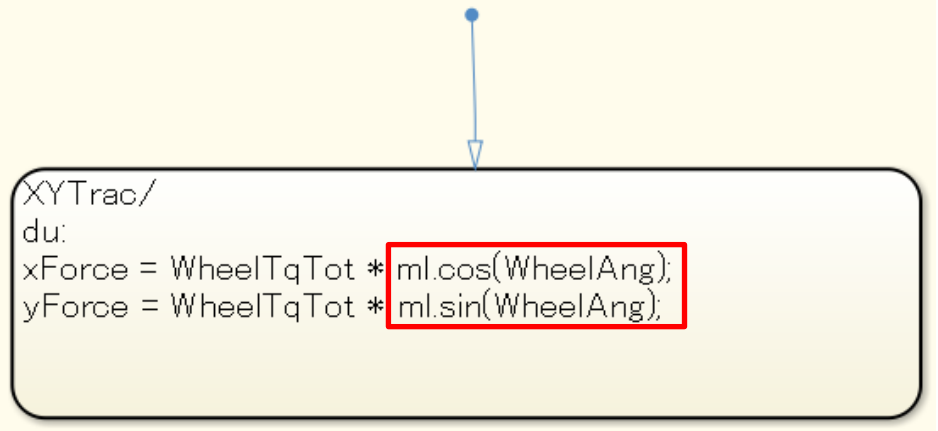
根拠	
サブ ID	**記述内容**
a1a2	・意図した動作・コード生成結果にならない可能性があります。

4.3.13. db_0127：Stateflow ブロック内の MATLAB コマンド使用制限

ルール ID：タイトル	**db_0127：Stateflow ブロック内の MATLAB のコマンド使用制限**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a1	Stateflow ブロック内では、MATLAB コマンドを使用しません。	-
	【正】 Stateflow ブロック内で、MATLAB コマンドを使用していません。	

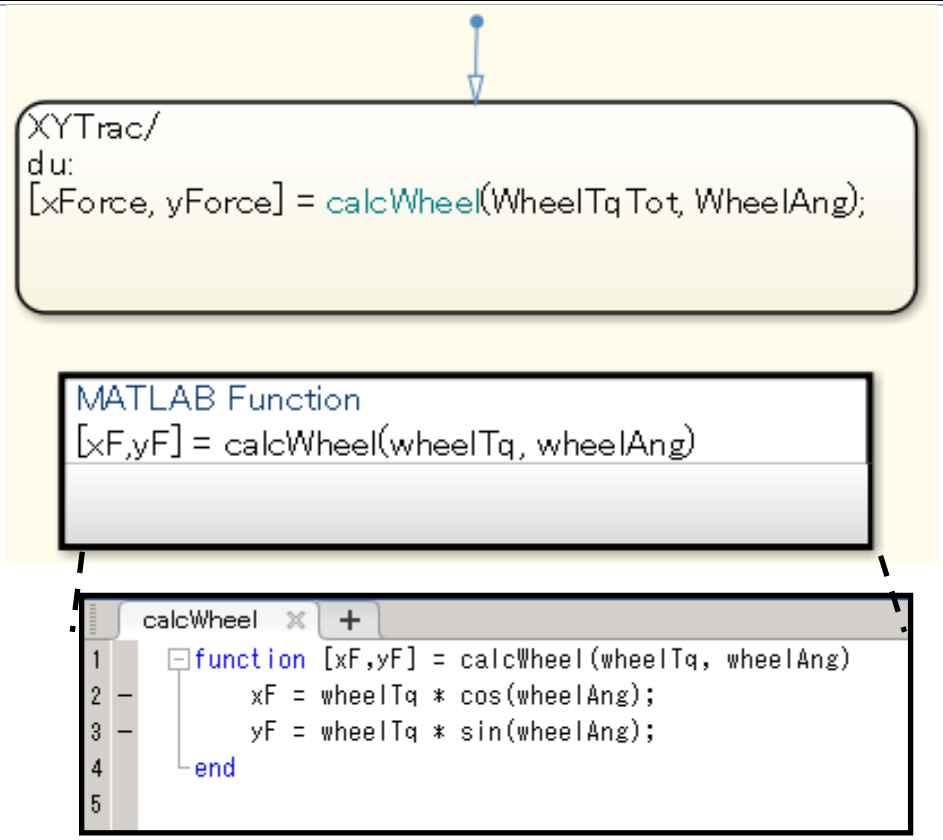


【誤】
Stateflow ブロック内で、MATLAB コマンドを使用しています。

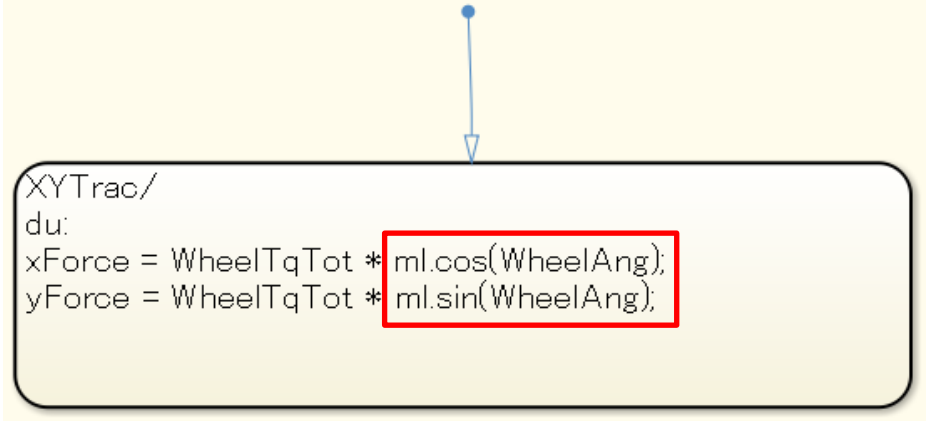


a2	MATLAB コマンドを使用する場合、[MATLAB Function]内に記述します。	-

【正】
MATLAB コマンドを[MATLAB Function]内に記述しています。



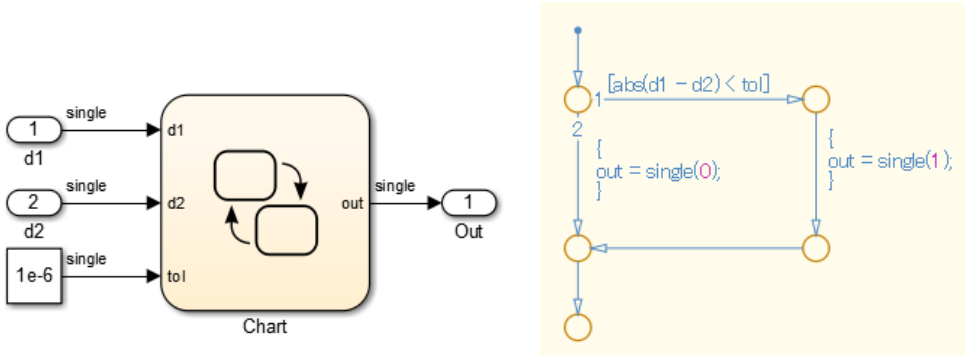
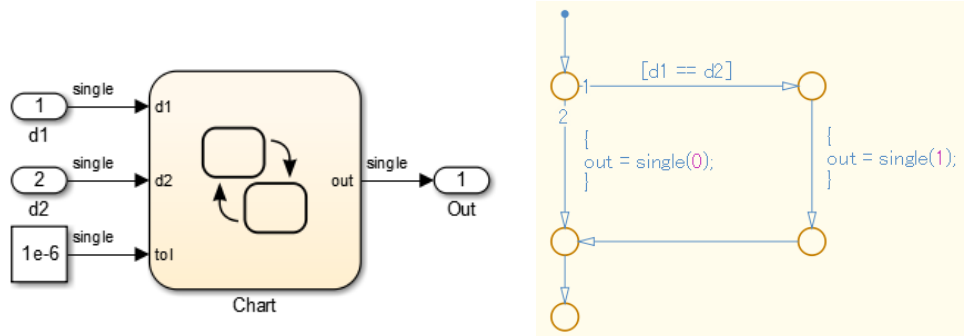
【誤】
MATLAB コマンドを[MATLAB Function]内に記述していません。



根拠	
サブ ID	**記述内容**
a1	・[Chart]のアクション言語としては、コード生成において一部の MATLAB コマンドしかサポートされていないため、MATLAB コマンドを使用するとコード生成できないおそれがあります。
a2	・[Chart]のアクション言語としては、コード生成において一部の MATLAB コマンドしかサポートされていないため、MATLAB コマンドを使用するとコード生成できないおそれがあります。 ・アクション言語の C と MATLAB コマンドの記述場所を分けて、可読性を向上します。

4.3.14. jc_0481 : Stateflow における浮動小数点型の比較

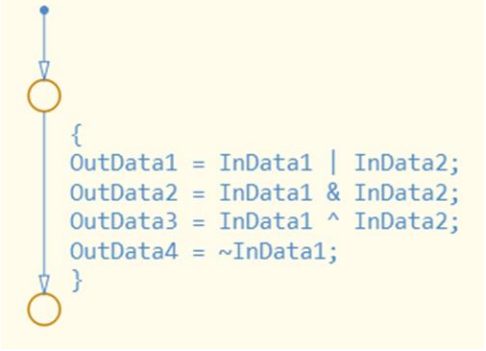
ルール ID : タイトル	**jc_0481 : Stateflow における浮動小数点型の比較**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**

a	浮動小数点型の等価比較演算==、!=、~=は使用しません。	-
	【正】 浮動小数点型の等価比較演算==、!=、~=を使用していません。  【誤】 浮動小数点型の等価比較演算==、!=、~=を使用しています。  	
根拠		
サブ ID	**記述内容**	
a	・浮動小数点の性質上、値に誤差を含むため、真を期待した等価比較結果が偽となる可能性があります。	

4.3.15. na_0001：Stateflow における演算子の統一

ルール ID：タイトル	**na_0001：Stateflow における演算子の統一**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	Stateflow のアクション言語が C の場合、演算子(”&”、”|”、”^”、”~”)は、ビット演算にのみ使用します。	-
	【正】 アクション言語が C の場合、演算子(“&”、“|”、“~”)はビット演算にのみ使用しています。	

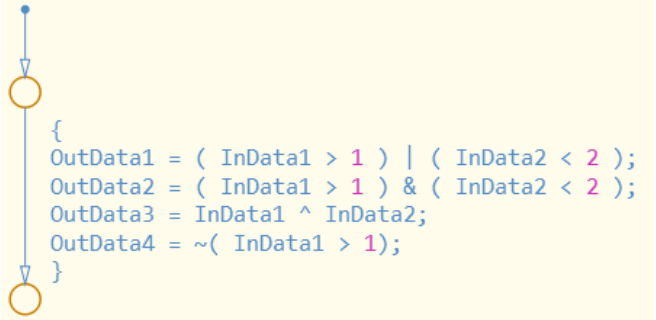
<チャートプロパティ>
アクション言語:C
[C 言語のビット演算が可能]:選択
InData1、InData2のデータ型:uint8



【誤】

” & ”, ” | ”, ” ^ ”, ” ~ ”をビット演算子として使用していません。

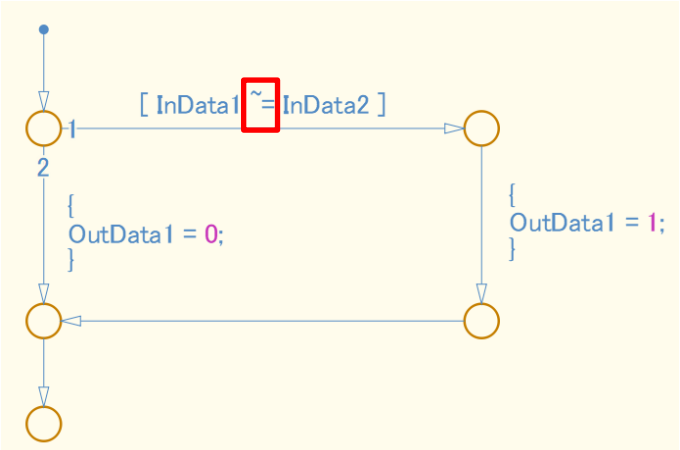
<チャートプロパティ>
アクション言語:C
[C 言語のビット演算が可能]:未選択
Indata1、Indata2のデータ型:uint8



b1	{チャートプロパティ}-{アクション言語}が”C”の時、等号否定をする場合は”~=”を使用します。	-

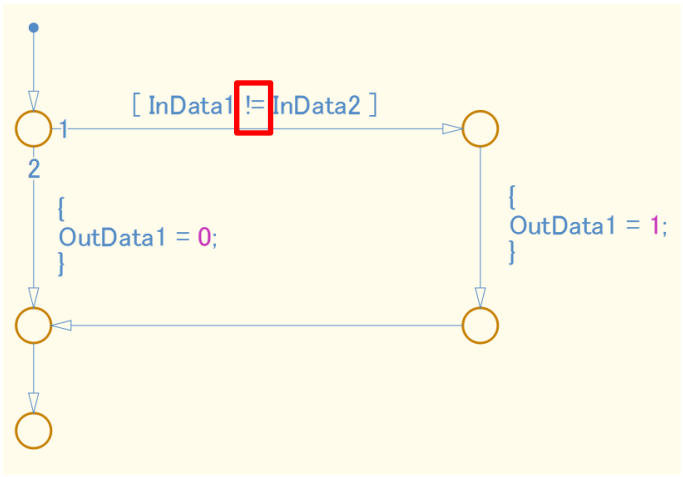
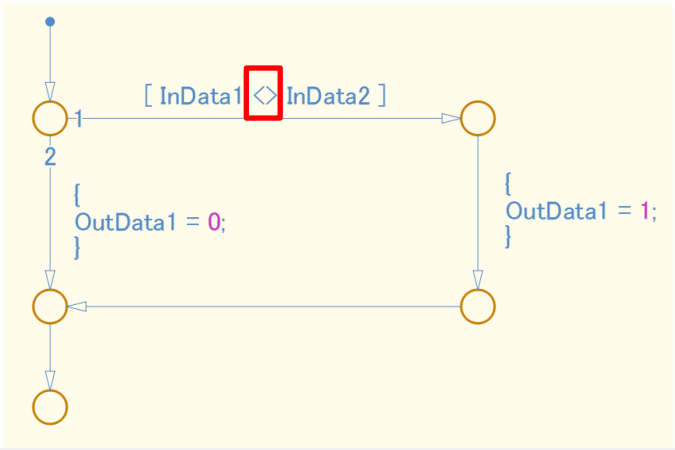
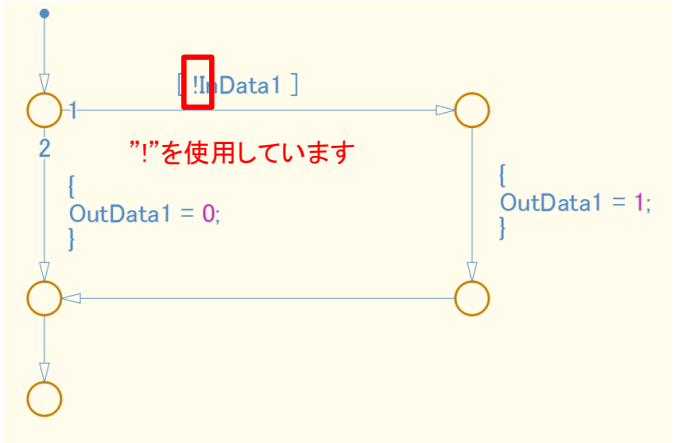
【正】

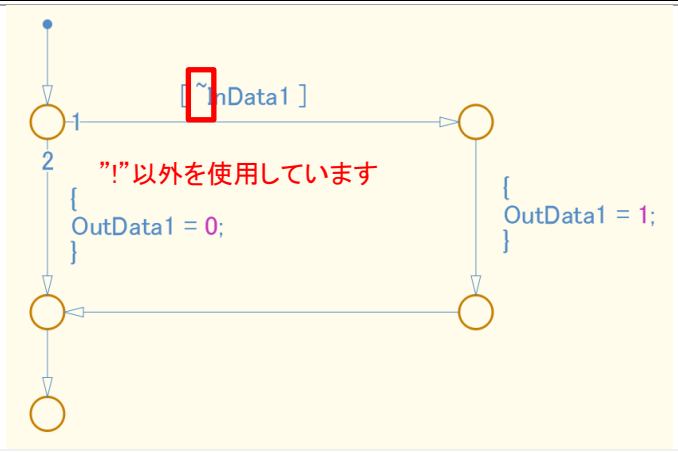
{チャートプロパティ}-{アクション言語}が"C"の時、等号否定をする場合は”~=”を使用しています。



b2	{チャートプロパティ}-{アクション言語}が”C”の時、等号否定をする場合は”!=”を使用します。	-

【正】

	{チャートプロパティ}-{アクション言語}が"C"の時、等号否定をする場合は"!="を使用しています。  	
b3	{チャートプロパティ}-{アクション言語}が"C"の時、等号否定をする場合は"<>"を使用します。	-
	【正】 {チャートプロパティ}-{アクション言語}が"C"の時、等号否定をする場合は"<>"を使用しています。  	
c	{チャートプロパティ}-{アクション言語}が"C"の時、論理否定をする場合は"!"を使用します。	-
	【正】 {チャートプロパティ}-{アクション言語}が"C"の時、論理否定をする場合は"!"を使用しています。  【誤】 {チャートプロパティ}-{アクション言語}が"C"の時、論理否定をする場合に"!"以外を使用しています。	

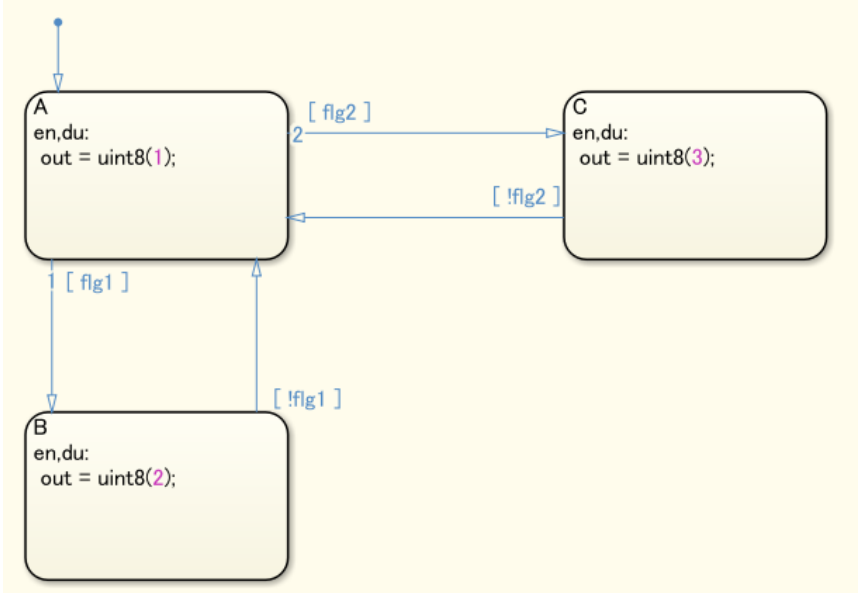


根拠	
サブ ID	**記述内容**
a	・「Stateflow のアクション言語」が"MATLAB"もしくは、"C"で{C 言語のビット演算が可能}のチェックを入れていない時、"&&"と"&"、"||"と"|"は同じ演算機能を有しています。しかし、同一[Chart]内で演算子"&&"と"&"もしくは"||"と"|"を混在させて使用すると、モデル閲覧者に対してそれらが異なる演算機能であるのか同一の演算機能であるのかの判断の手間と誤認識を与えてしまう可能性があります。
b1b2b3	・等号否定の記述方法を統一することで可読性が向上します。
c	・論理否定の記述方法を統一することで可読性が向上します。 ・{C 言語のビット演算が可能}のチェックを切り替えても演算子"!"の機能はかわらないので設定の変更によるロジックへの影響は受けません。

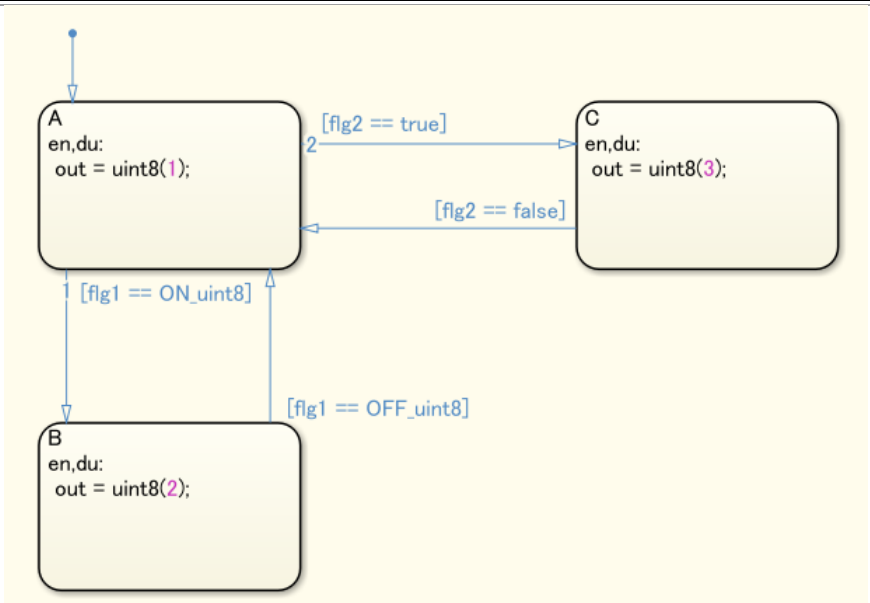
4.3.16. jc_0655 : Stateflow における論理型の比較演算禁止

ルール ID : タイトル	jc_0655 : Steteflow における論理型の比較演算禁止	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	論理型定数との比較演算はしません。	-

【正】
論理型定数との比較演算を行っていません。



【誤】
論理型定数との比較演算が行われています。

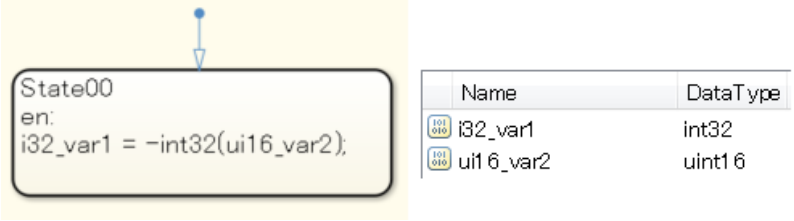


根拠	
サブ ID	**記述内容**
a	・論理信号による条件式を、「boolean 値信号==true(boolean 型定数)」とするか、「(boolean 値信号)」とするか、方法を統一することで可読性が向上します。 ・冗長なモデルになることを防ぎます。 ・一般的な作法から外れることにより、想定外の問題が起こるリスクがあります。

4.3.17. jc_0451 ： 符号なし整数に対する単項マイナス

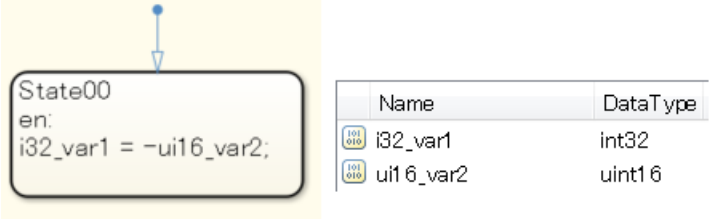
ルール ID：タイトル	**jc_0451 ： 符号なし整数に対する単項マイナス**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	符号なし整数には単項マイナスを使用しません。	-

【正】



【誤】

16bit 環境では負の値は出力されません。
(32bit 環境では負の値が出力されます)



根拠	
サブ ID	**記述内容**
a	・実行結果が実行環境に依存するため、意図しない実行結果となる可能性があります。

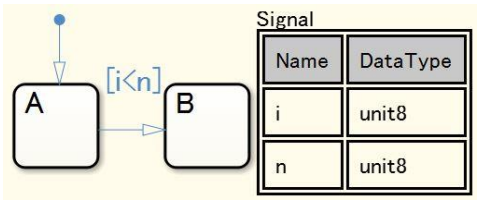
4.3.18. jc_0802 ： Stateflow における暗黙の型変換の禁止

ルール ID：タイトル	jc_0802 ： Stateflow における暗黙の型変換の禁止	
ルール		
サブ ID	記述内容	カスタムパラメーター
a	演算(代入・比較・算術等の全て演算)は、同じデータ型の変数間で行います。また、関数呼び出しにおける実引数と仮引数のデータ型は同じデータ型にします。	-

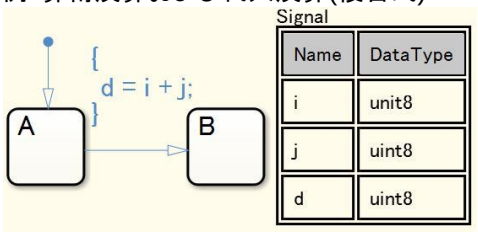
【正】

・同じデータ型同士を演算しています。

例：比較演算

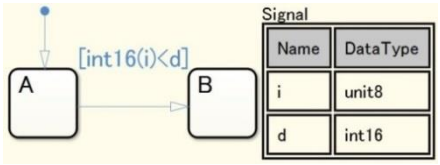


例：算術演算および代入演算(複合式)

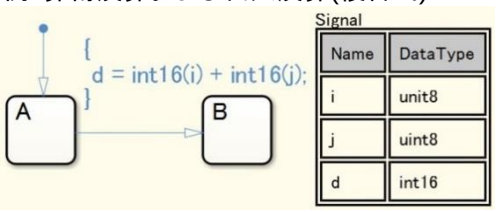


・異なるデータ型同士ですが、明示的に型変換してから演算しています。

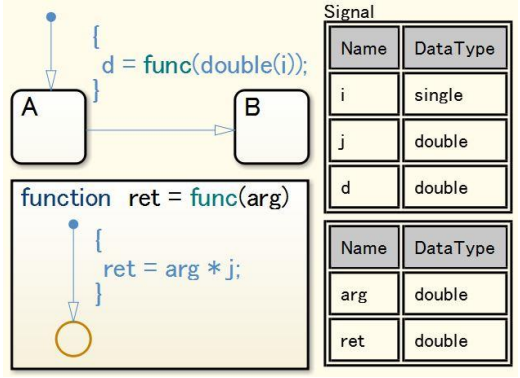
例：比較演算



例：算術演算および代入演算(複合式)



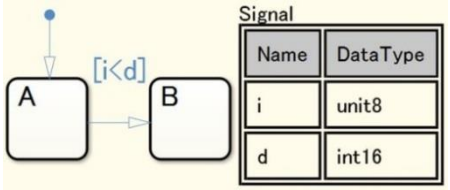
・関数呼び出しにおける実引数と仮引数のデータ型が同じです。



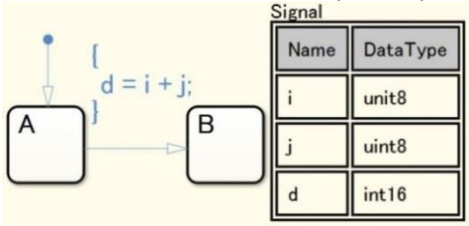
【誤】

・異なるデータ型同士で演算しています。

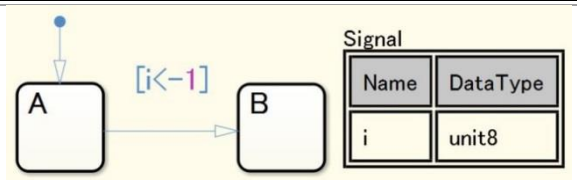
例：比較演算



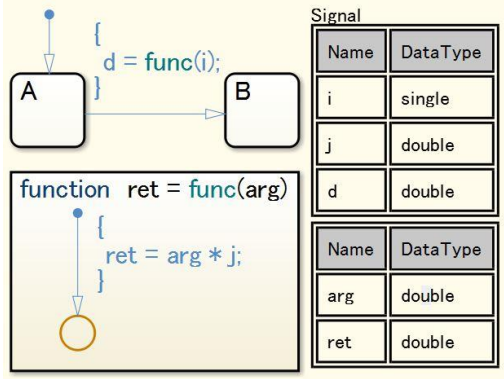
例：算術演算および代入演算(複合式)



・符号なし整数型変数と符号付き整数の間で演算をしています。

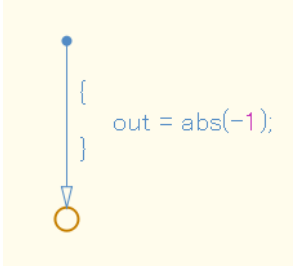
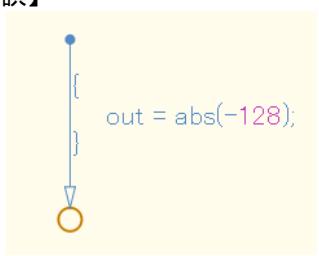


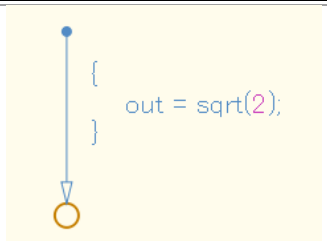
・関数呼び出しにおける実引数と仮引数のデータ型が異なります。

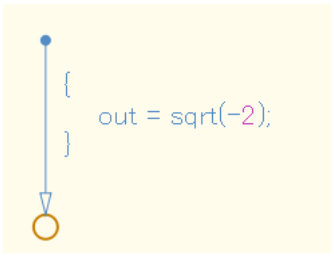
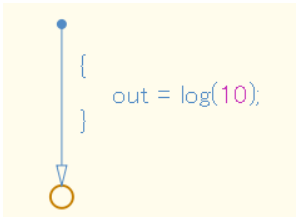


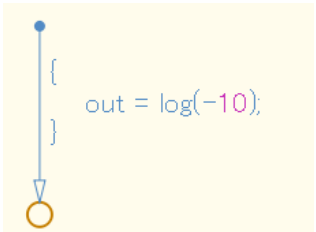
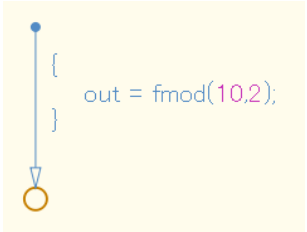
根拠	
サブ ID	**記述内容**
a	・暗黙の型変換により、想定外の動作結果となるリスクがあります。

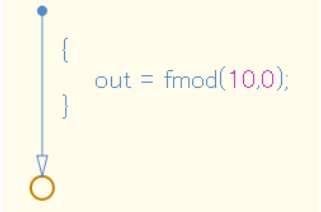
4.3.19. jc_0803：ライブラリ関数に引き渡される値

ルール ID：タイトル	**jc_0803：ライブラリ関数に引き渡される値**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a1	abs ライブラリ関数を使用する場合は、符号付整数値の最小値を入力しません。	-
	【正】  【誤】 	
a2	abs ライブラリ関数は使用しません。	-
b1	sqrt ライブラリ関数を使用する場合は、負の数を入力しません。	-
	【正】	

<table>
<tr><td></td><td> 

【誤】

</td><td></td></tr>
<tr><td>b2</td><td>sqrt ライブラリ関数は使用しません。</td><td>-</td></tr>
<tr><td>c1</td><td>log、log10 ライブラリ関数を使用する場合は、負の数は入力しません。</td><td>-</td></tr>
<tr><td></td><td>【正】


【誤】

</td><td></td></tr>
<tr><td>c2</td><td>log、log10 ライブラリ関数は使用しません。</td><td>-</td></tr>
<tr><td>d1</td><td>f mod ライブラリ関数を使用する場合は、第二引数にゼロを入力しません。</td><td>-</td></tr>
<tr><td></td><td>【正】


【誤】

</td><td></td></tr>
</table>

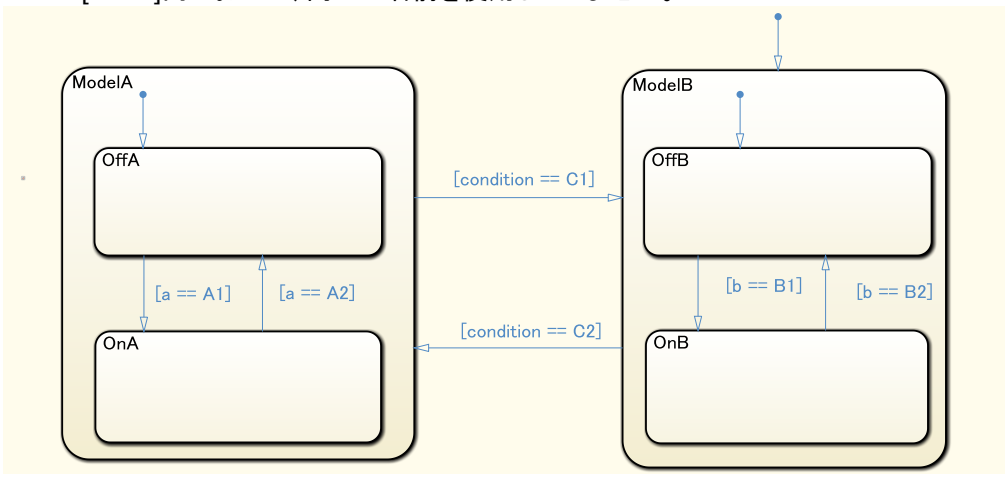
d2	fmod ライブラリ関数は使用しません。	-
根拠		
サブ ID	**記述内容**	
a1	・不正な値が引き渡される場合のライブラリ関数の動作は処理系依存のため、意図しない動作となる可能性があります。	
a2b2 c2d2	・同じガード処理を Simulink と Stateflow で重複してモデル化することを避けるため、数値演算処理は Simulink で行うよう統一します。	
b1c1d1	・不正な値が引き渡される場合のライブラリ関数の動作は処理系依存のため、意図しない動作となる可能性があり、回避処理を設計する必要があります。	

4.4. ラベルの記述

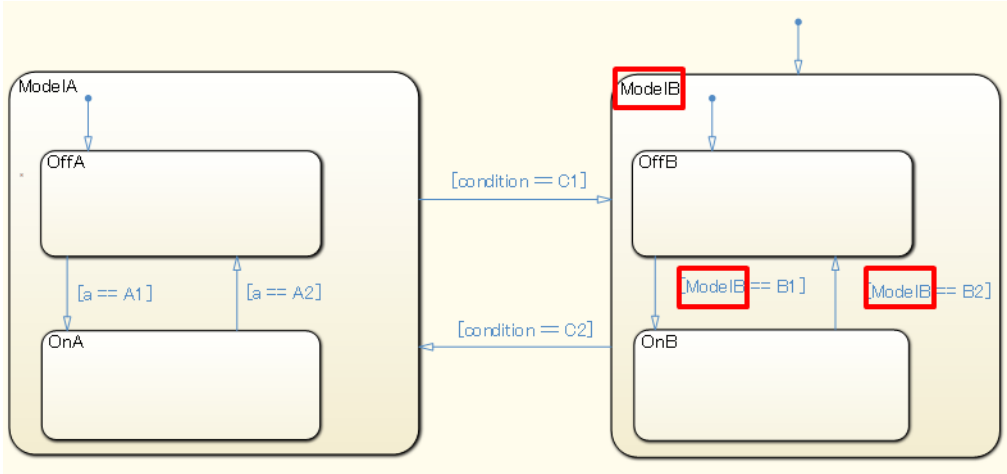
4.4.1. jc_0732：ステート名 / データ名 / イベント名の区別

ルール ID：タイトル	**jc_0732：ステート名 / データ名 / イベント名の区別**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	一つの[Chart]内において、ステートとデータ(入出力、ローカルデータ、定数、パラメーター、データストアメモリ)とイベントに同一の名前を使用しません。	-

【正】
一つの[Chart]内において、同一の名前を使用していません。



【誤】
一つの[Chart]内において、同一の名前を使用しています。

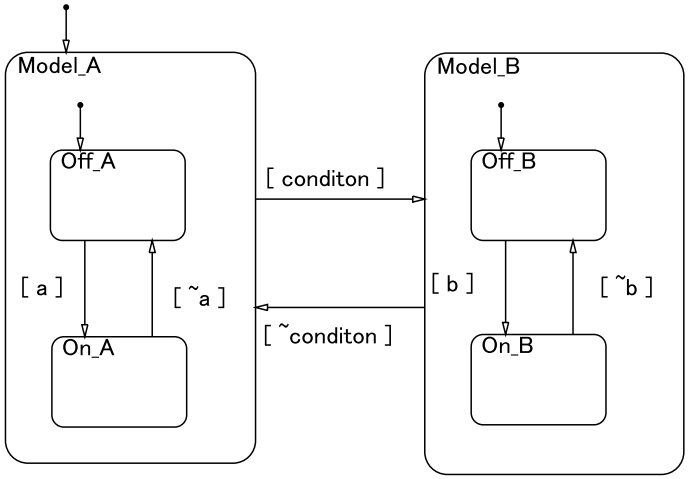


根拠	
サブ ID	**記述内容**
a	・ステート、データ、イベントに同一の名前を使用しないことで開発者の誤解を防ぎます。

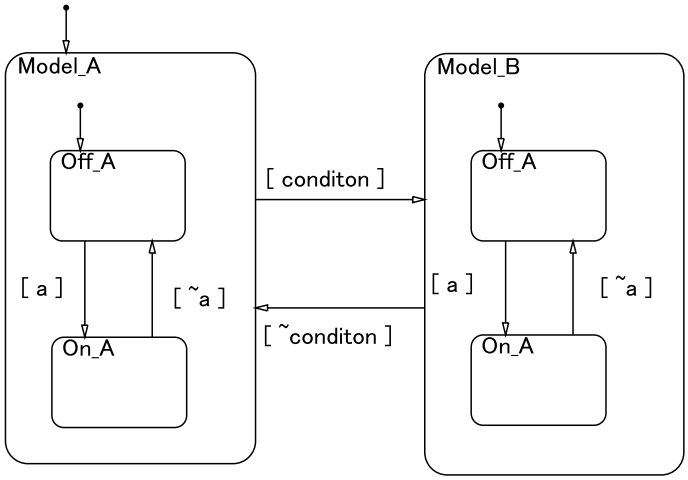
4.4.2. jc_0730：Stateflow ブロック内でのステート名の独立性

ルール ID：タイトル	**jc_0730：Stateflow ブロック内でのステート名の独立性**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	ステート名は、[Chart]内で唯一にします。 リンクされた Atomic サブチャートの内部は別の[Chart]として扱います。	-

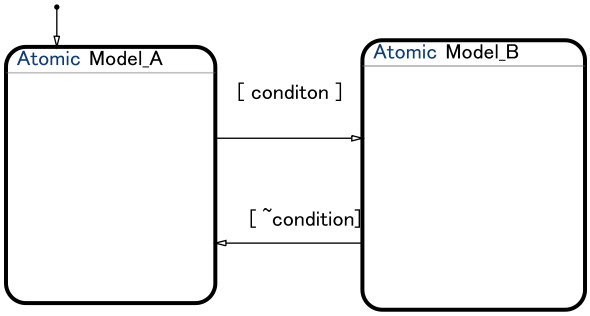
【正】



【誤】

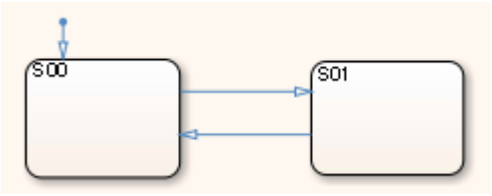
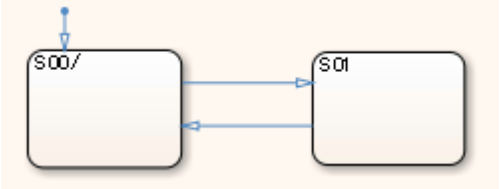


参考：Atomic サブチャート化の方法
右クリックメニュから Atomic サブチャート化できます。



根拠	
サブ ID	**記述内容**
a	・逸脱した場合、可読性が損なわれます。 ・逸脱した場合、意図したコードとならない可能性があります。

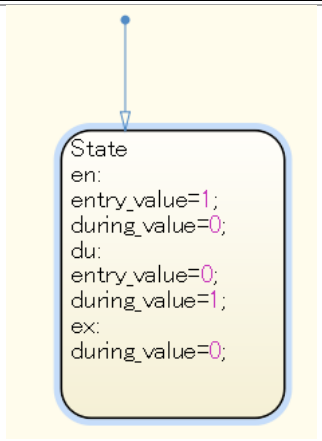
4.4.3. jc_0731 ： ステート名の記述

ルール ID : タイトル	**jc_0731 ： ステート名の記述**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	ステート名の後ろはスラッシュ(/)をつけずに改行します。	-
	【正】  【誤】  	

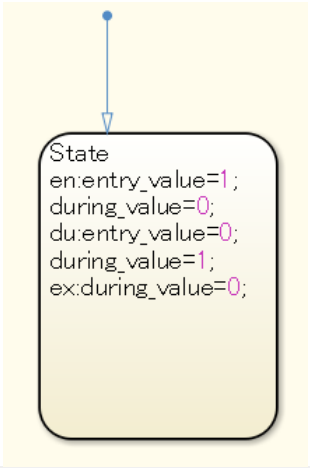
根拠	
サブ ID	**記述内容**
a	・ステート名の表記方法を統一することで可読性が向上します。

4.4.4. jc_0501 ： ステートラベルの改行

ルール ID : タイトル	**jc_0501 ： ステートラベルの改行**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	ステートアクションタイプを記述した行には、ステートアクションを記述しません。	-
	【正】	



【誤】



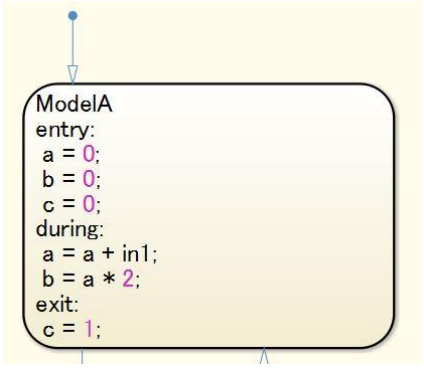
根拠	
サブ ID	**記述内容**
a	・可読性が損なわれます。

4.4.5. jc_0736 ： Stateflow ブロック内のインデント統一

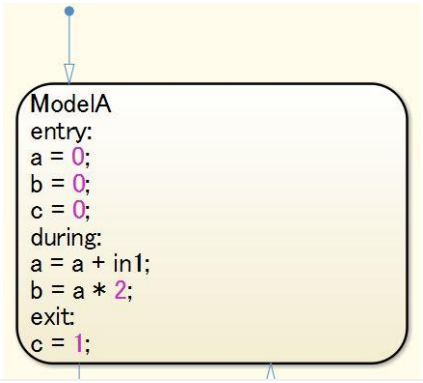
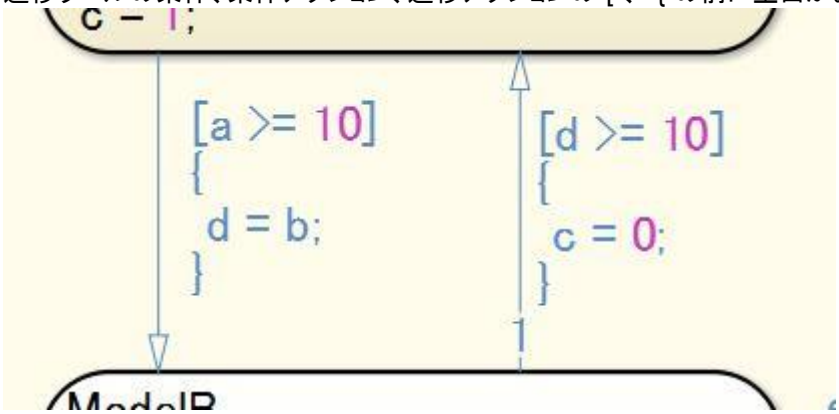
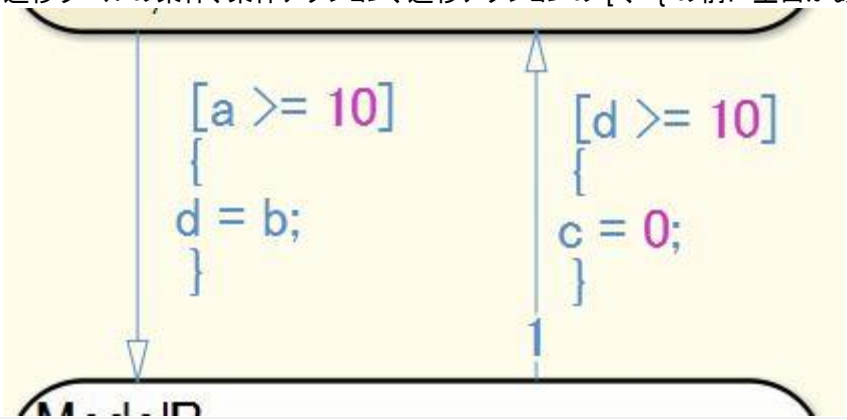
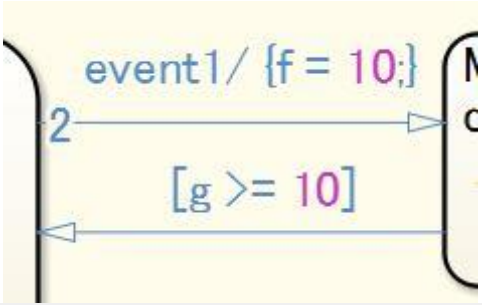
ルール ID : タイトル	jc_0736 ： Stateflow ブロック内のインデント統一	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	ステートアクションタイプの先頭に空白を使用しません。 実行文は、先頭に半角スペースを 1 文字入れます。	半角スペースの数

【正】

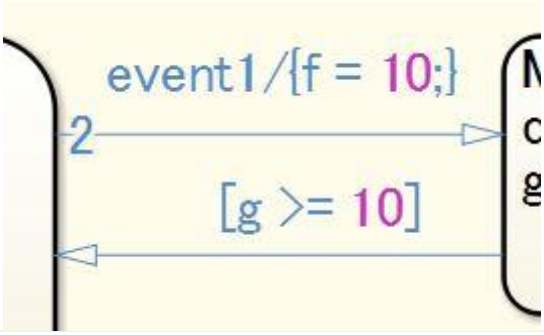
実行文の前に半角スペースが 1 文字あります。



【誤】

	実行文の前に半角スペースがありません。  	
b	遷移条件の”[“、条件アクションの”{“、遷移アクションの”/”の前に空白を入れません。	-
	【正】 遷移ラベルの条件、条件アクション、遷移アクションの”[“、”{“の前に空白がありません。  【誤】 遷移ラベルの条件、条件アクション、遷移アクションの”[“、”{“の前に空白があります。  	
c	遷移アクションの”/”の後ろに半角スペースを 1 文字以上入れます。	半角スペースの数
	【正】 遷移アクションの“/” の後ろに半角スペースがあります。  	

【誤】
遷移アクションの “/” の後ろに半角スペースがありません。



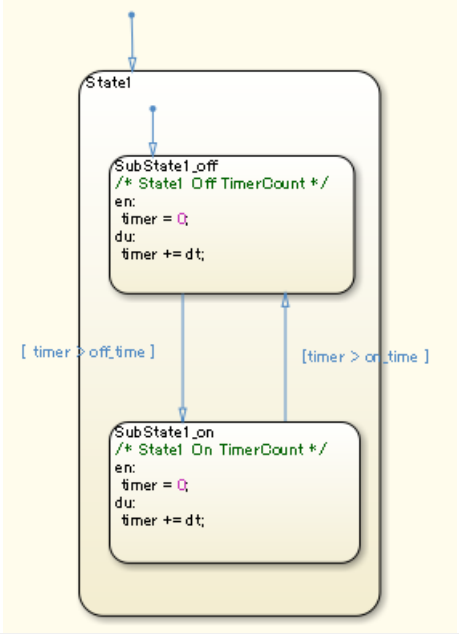
根拠

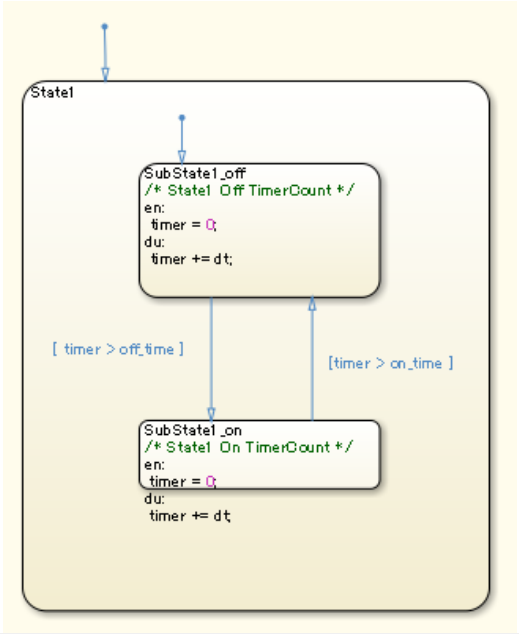
サブ ID	記述内容
a	・ステートラベルのステートアクションタイプと実行文の紐付けが解るようにインデントを統一することで可読性が向上します。
b	・遷移条件、条件アクション、遷移アクションのインデントを統一することで可読性が向上します。
c	・空白の入れ方を統一することで可読性が向上します。

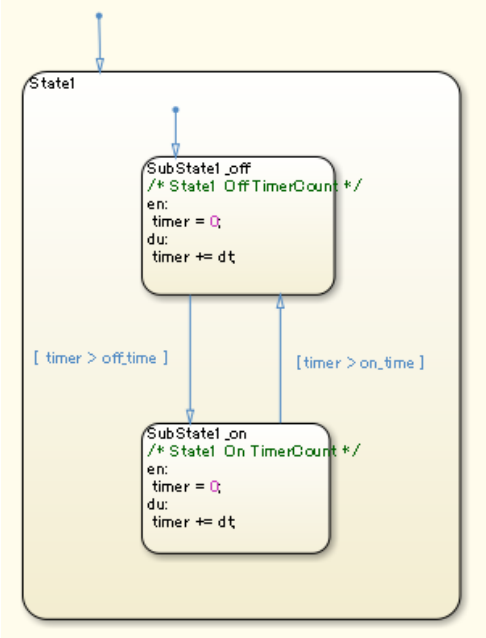
4.4.6. jc_0739 ： ステート内テキストの記述方法

ルール ID : タイトル	jc_0739 ： ステート内テキストの記述方法	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	ステート内のテキストは、そのステートの境界線を越えて記述しません。	-

【誤】





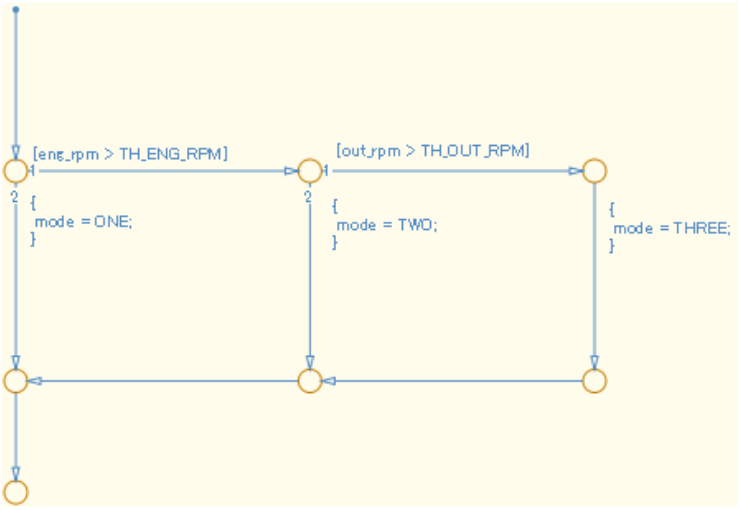


根拠	
サブ ID	**記述内容**
a	・ステート内のテキストが境界線を越えると、そのテキストがどの箇所に対するテキストなのかが解りにくくなります。

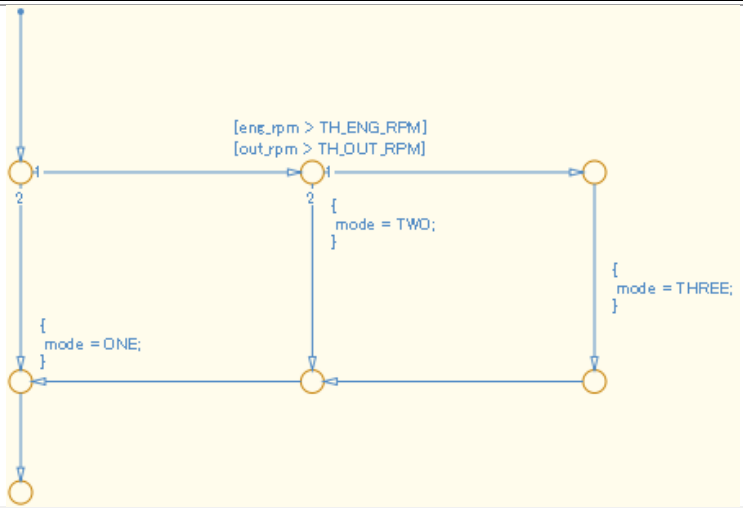
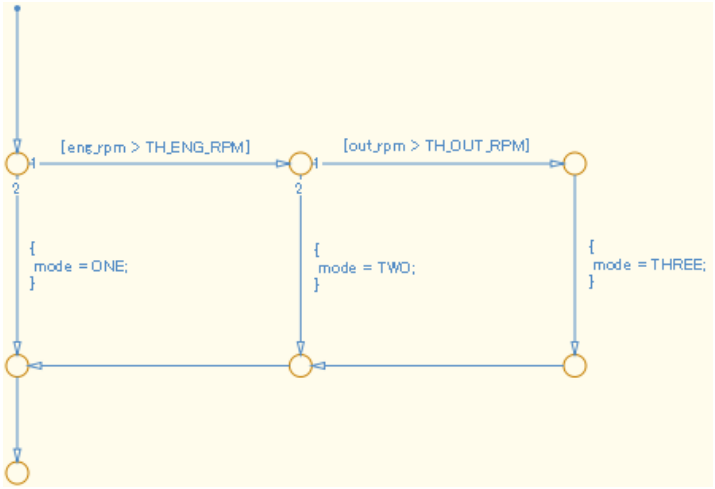
4.4.7. jc_0770：遷移ラベルの配置

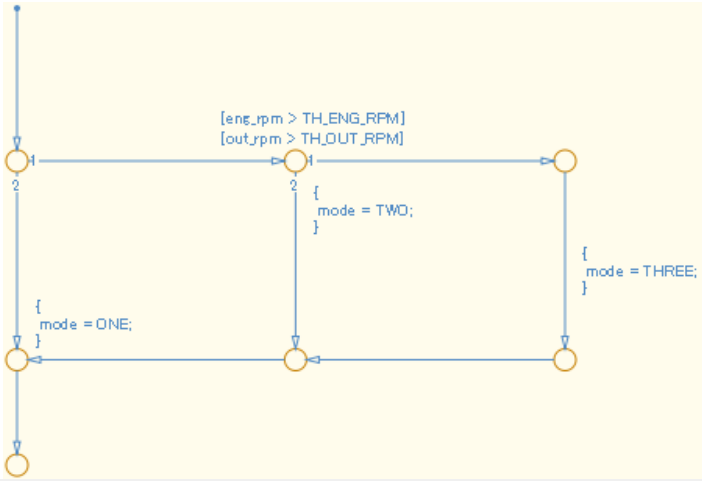
ルール ID：タイトル	**jc_0770：遷移ラベルの配置**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a1	遷移ラベルは遷移線の遷移元に記述します。	-

【正】
遷移ラベルが遷移線の遷移元に記述されている。



【誤】
遷移ラベルの配置が統一されておらず、遷移線と対応した記述になっていない。

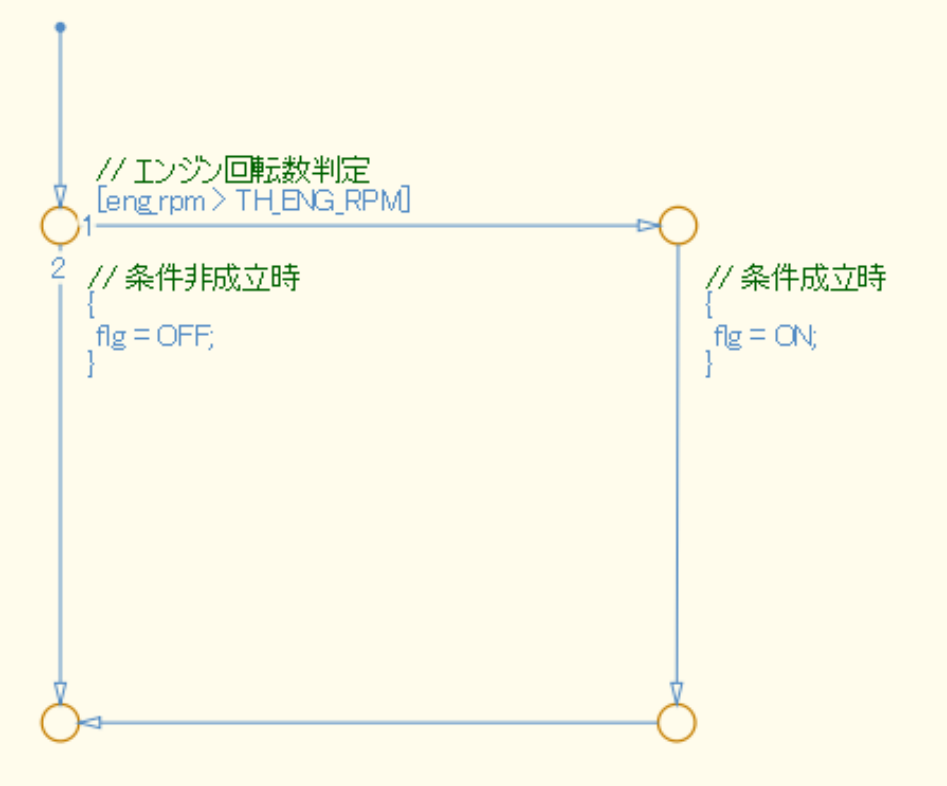
<table>
<tr><td></td><td> 
</td><td></td></tr>
<tr><td>a2</td><td>遷移ラベルは遷移線の中央付近に記述します。</td><td>-</td></tr>
<tr><td></td><td colspan="2">【正】
遷移ラベルが遷移線の中央付近に記述されている。
 

【誤】
遷移ラベルの配置が統一されておらず、遷移線と対応した記述になっていない。
 
</td></tr>
<tr><td colspan="3">根拠</td></tr>
<tr><td>サブ ID</td><td colspan="2">記述内容</td></tr>
<tr><td>a1a2</td><td colspan="2">・遷移ラベルの配置を統一することで、遷移線と対応する遷移ラベルが明確になります。</td></tr>
</table>

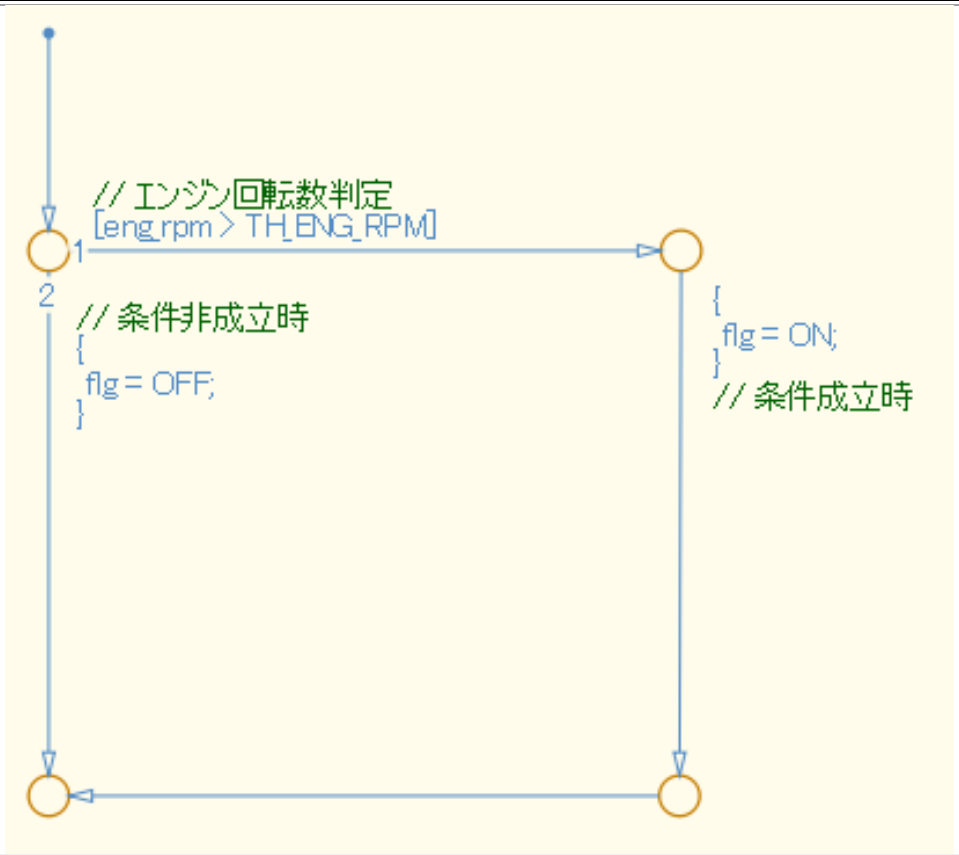
4.4.8. jc_0771 : 遷移ラベル内のコメントの配置

ルール ID : タイトル	jc_0771 : 遷移ラベル内のコメントの配置	
ルール		
サブ ID	記述内容	カスタムパラメーター
a1	遷移ラベル内のコメントは、遷移条件・条件アクション・遷移アクション・Stateflow のイベントの上に記述します。	-

【正】
遷移ラベル内のコメント配置が統一されています。

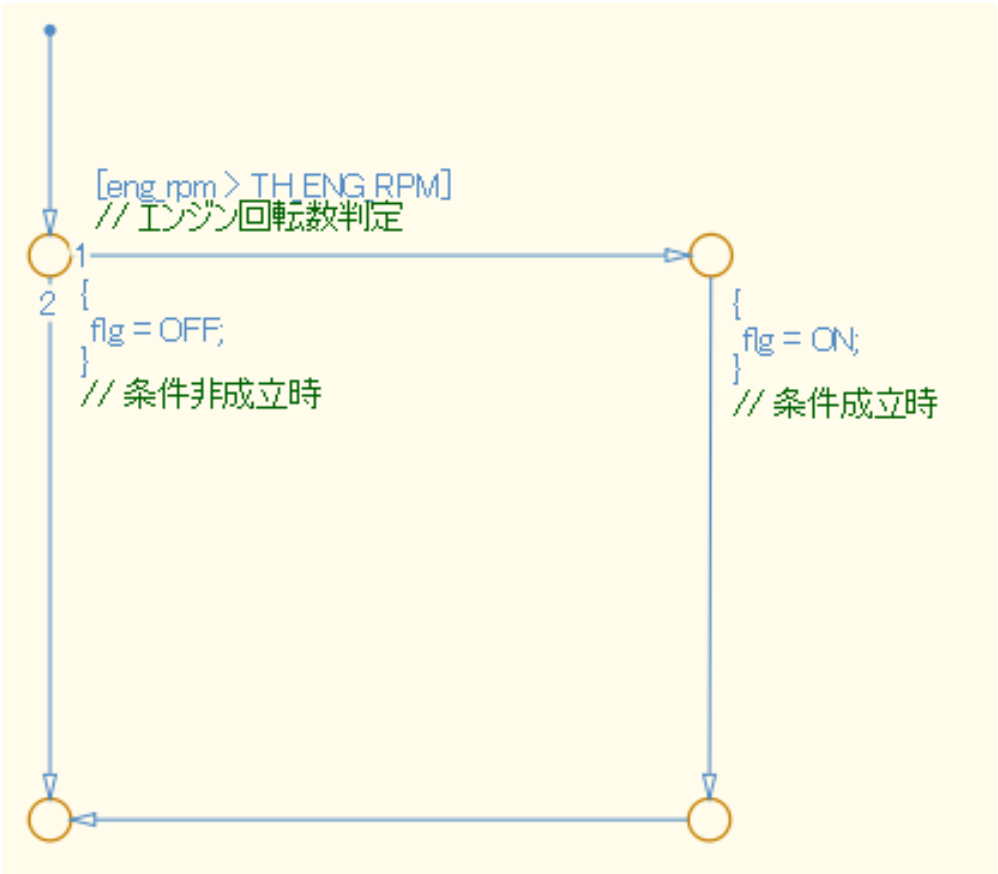


【誤】
遷移ラベル内のコメント配置が統一されていません。



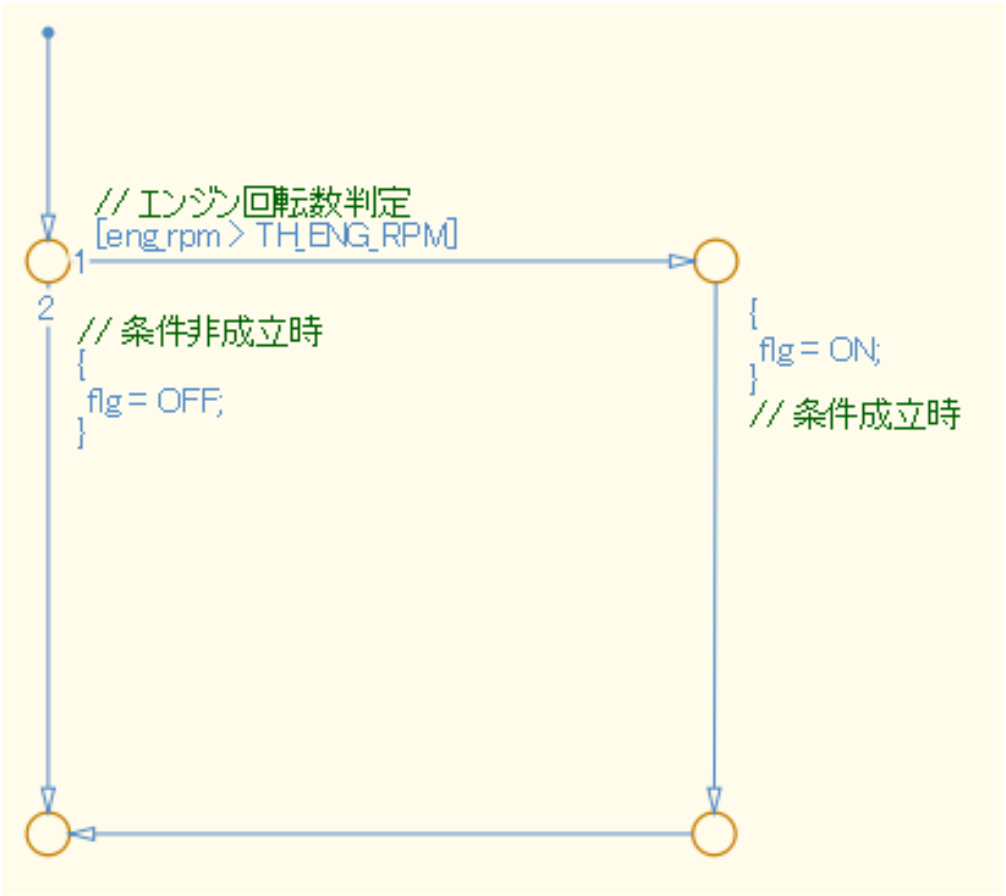
a2	遷移ラベル内のコメントは、遷移条件・条件アクション・遷移アクション・Stateflow のイベントの下に記述します。	-

【正】
遷移ラベル内のコメント配置が統一されています。



【誤】

遷移ラベル内のコメント配置が統一されていません。



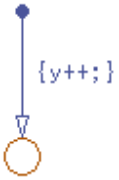
根拠	
サブ ID	**記述内容**
a1a2	・遷移ラベル内のコメントの配置を統一することで、コメントと対応する遷移条件・条件アクション・遷移アクション・イベントが明確になります。

4.4.9. jc_0752 : 遷移ラベルにおける条件アクションの記述方法

ルール ID : タイトル	**jc_0752 : 遷移ラベルにおける条件アクションの記述方法**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	条件アクションの括弧は、波括弧のみで 1 行とします。 (波括弧の前後で改行します。)	-

【正】

【誤】



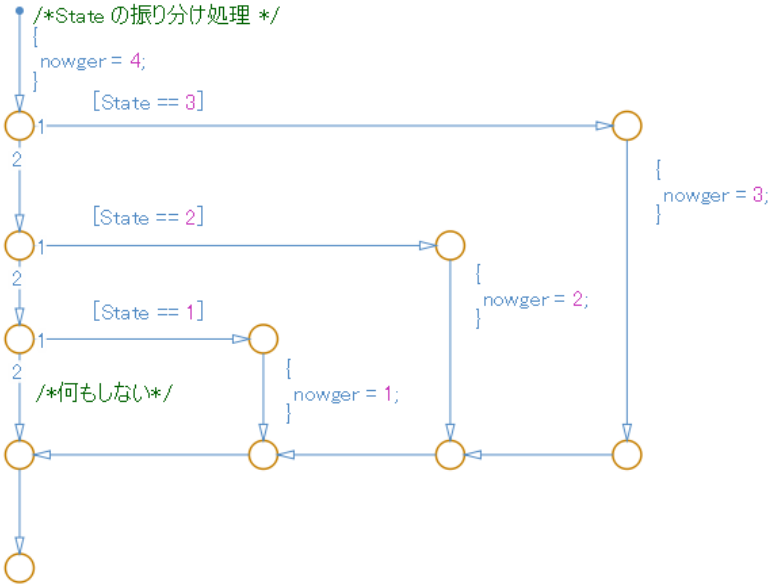
事例は、フローチャートで記載しましたが、状態遷移でも同様です。

根拠	
サブ ID	**記述内容**
a	・条件アクションであることを明示し、可読性を向上します。

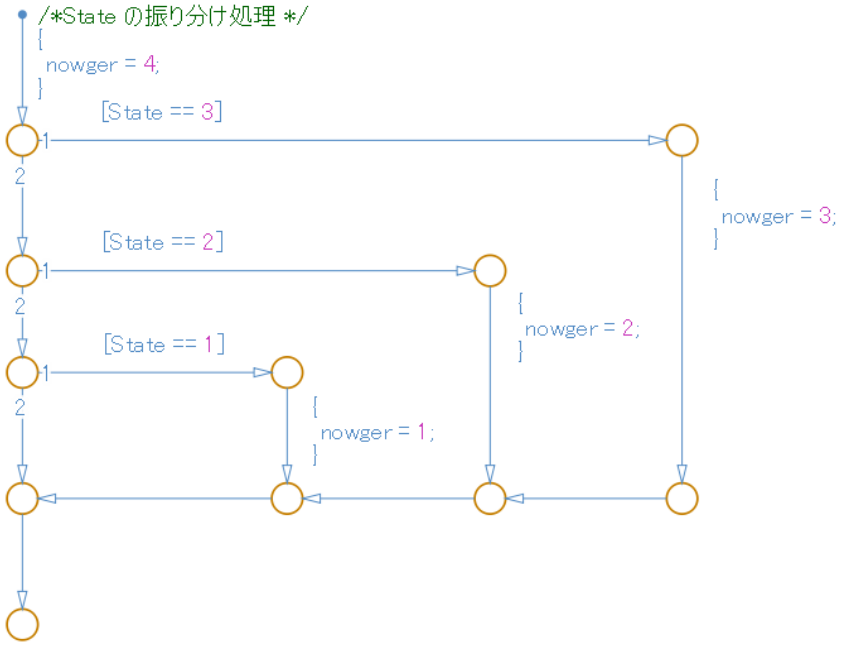
4.4.10. jc_0774 : 処理なし無条件遷移へのコメント

ルール ID : タイトル	**jc_0774 : 処理無し無条件遷移へのコメント**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	無条件遷移で処理が無い場合は、意図が解るように遷移ラベルにコメントを記載します。	-

【正】
処理がない条件パスにコメントを記載しています。



【誤】
処理がない条件パスにコメントが記載されていません。
アクションが無いことを意図しているのか、書き忘れなのか解りません。



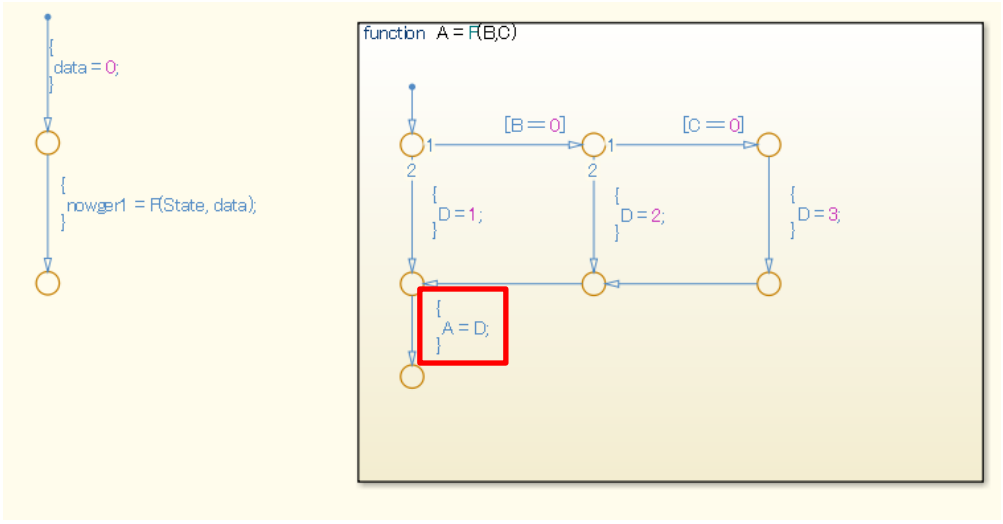
根拠	
サブ ID	**記述内容**
a	・意図的に処理を記載していないことを明示的にするためです。 遷移ラベルにコメントを記載すると、コード生成時にコードに埋め込まれます。

4.5. その他

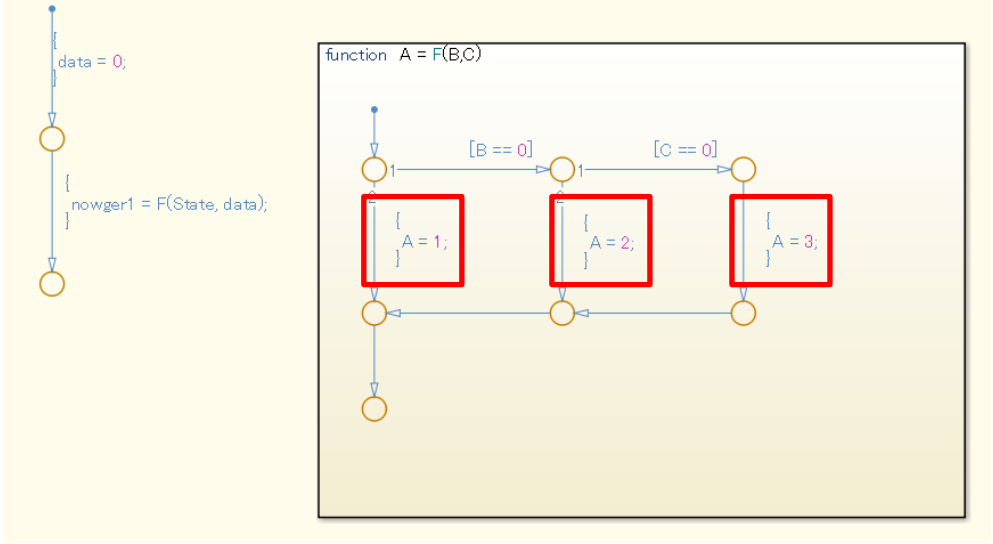
4.5.1. jc_0511 ：グラフィカル関数からの戻り値の設定

ルール ID：タイトル	**jc_0511 ：グラフィカル関数からの戻り値の設定**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	グラフィカル関数の戻り値は、1 か所のみで設定します。	-

【正】



【誤】



根拠	
サブ ID	**記述内容**
a	・出力名を変更した際に、変更箇所が限定され変更漏れを防ぐことができます。

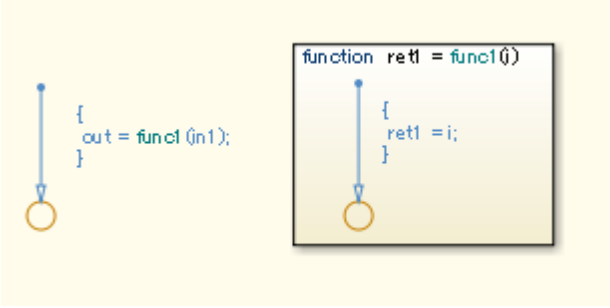
4.5.2. jc_0804 ：グラフィカル関数による再帰的呼び出しの禁止

ルール ID：タイトル	jc_0804 ：グラフィカル関数による再帰的呼び出しの禁止

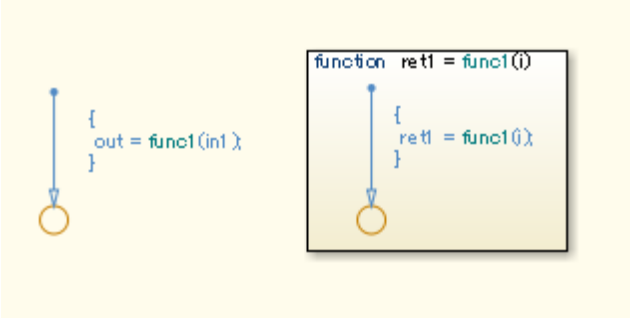
ルール

サブ ID	記述内容	カスタムパラメーター
a	グラフィカル関数内での自身の呼び出しと、グラフィカル関数同士での呼び出しを禁止します。	-

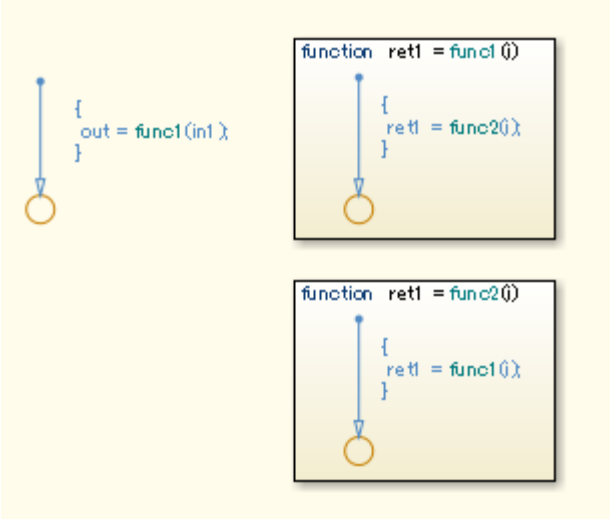
【正】
グラフィカル関数内で処理を実行しています。



【誤】
グラフィカル関数内で自身をコールしています。



グラフィカル関数同士でコールしています。



根拠

サブ ID	記述内容
a	・可読性が低下します。また、オーバーフローや無限ループなど意図しない動作となる可能性があります。

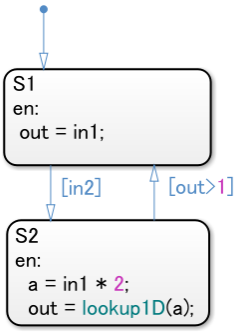
4.5.3. na_0042：Simulink 関数を使用する場面

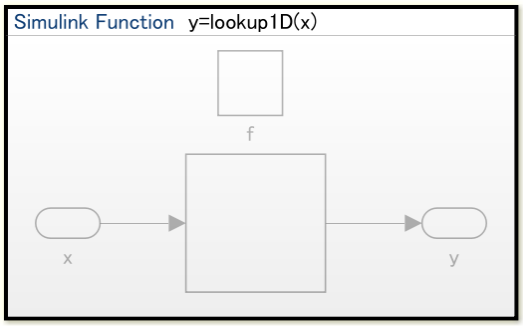
ルール ID：タイトル	na_0042：Simulink 関数を使用する場面

ルール

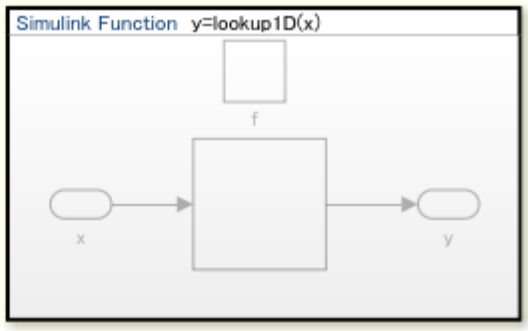
サブ ID	記述内容	カスタムパラメーター
a	[Chart]内で Stateflow の Simulink 関数を使用するには、以下の条件のうち、一つ以上を満たす必要があります。 ・Stateflow の Simulink 関数において、入出力変数が Stateflow のローカルデータのみを使用する。 ・Stateflow の Simulink 関数において、入出力変数が Stateflow のローカルデータと入力データのみを使用する。 ・Stateflow の Simulink 関数が、[Chart]内で複数箇所から呼び出される。 ・Stateflow の Simulink 関数が、全ての時間ステップでは呼び出されない。	-

【正】
Simulink 関数 lookup1D は全ての時間ステップでは呼び出されないため利用できます。





【誤】
Simulink 関数 lookup1D は全ての時間ステップで呼び出されるため利用できません。
(out は Stateflow の出力データ)



根拠

サブ ID	記述内容
a	・モデルの可読性を向上させるため、[Chart]内で無軌道に Simulink 関数を使わないようにします。

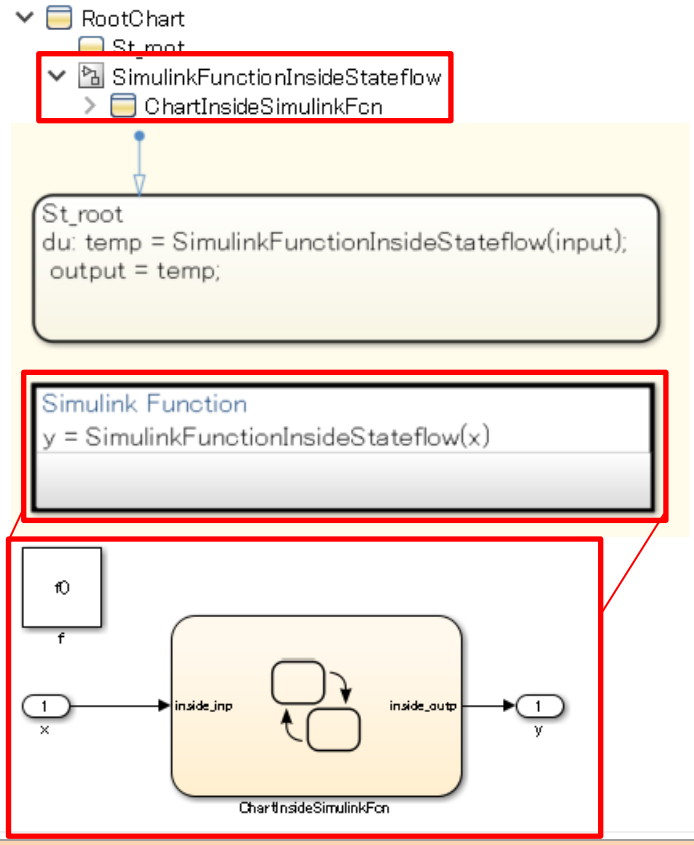
4.5.4. na_0039：Chart ブロック内の Simulink 関数の制約

ルール ID：タイトル	na_0039：Chart ブロック内の Simulink 関数の制約

ルール

サブ ID	記述内容	カスタムパラメーター
a	[Chart]内の[Simulink Function]の中では、Stateflow ブロックを使用しません。	-

【誤】



根拠	
サブ ID	**記述内容**
a	・可読性が低下し、設計ミスに繋がるおそれがあります。

5. その他

5.1. その他のルール

5.1.1. na_0037 : 単一変数のバリアント条件式の使用

ルール ID : タイトル	na_0037 : 単一変数のバリアント条件式の使用

ルール

サブ ID	記述内容	カスタムパラメーター
a	バリアント条件式は、複数の変数からなる複合条件を禁止します。 ＜例外＞ 既定のバリアントを使用する場合は、複数の変数からなる条件式になる場合があります。	-

【正】
バリアント条件式は複数の変数からなる単一条件で設定されています。

名前	サブモデ...	バリアント制御	条件
na0037a_OK			
Variant Subsystem			
Default_FofA		DefaultVar	(INLINE==0)&&(FUNC==0)
Function_FofA		FunctionVar	FUNC==1
InLineVar_FofA		InLineVar	INLINE==1

条件式では、列挙型変数の使用を推奨します。この例では、例としての可読性を高めるために、数値を使用しています。

【誤】
バリアント条件式が複数の変数からなる複合条件で設定されています。

名前	.. バリアント制御	条件
na0037a_NG		
Variant Subsystem		
AutoTrans	autoTrans	(INLINE==0)&&(transType==5)
Default_4speed	defaultTrans	(((INLINE==0)&&(transType==3))==0)&&(FUNC==0)&&(transType~=2)
ManualTrans	manualTrans	(FUNC==1)||(transType==2)

根拠

サブ ID	記述内容
a	・条件式が複雑化することで、どのサブシステムがアクティブになるか解りにくくなり、条件の抜け漏れに繋がります。 ・条件の抜け漏れがあると、アクティブなサブシステムが存在しない状態になるおそれがあります。

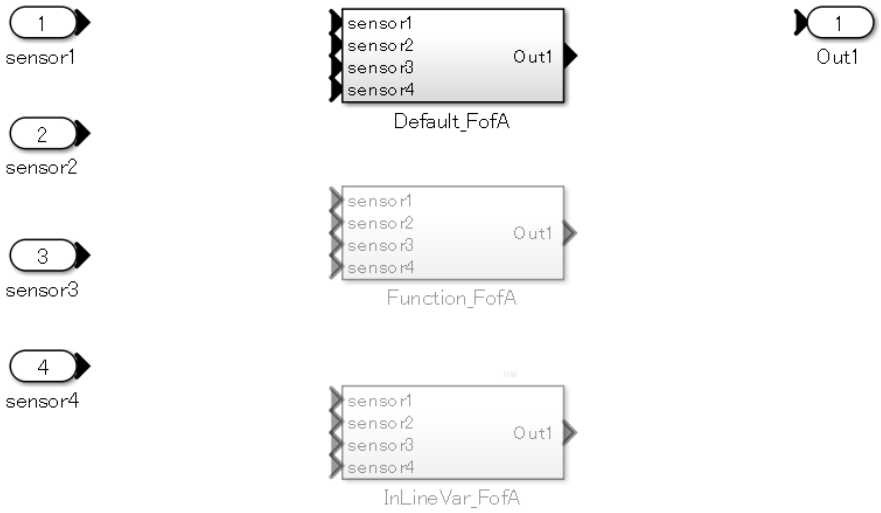
5.1.2. na_0020 : バリアントシステムへの入力数

ルール ID : タイトル	na_0020 : バリアントシステムへの入力数

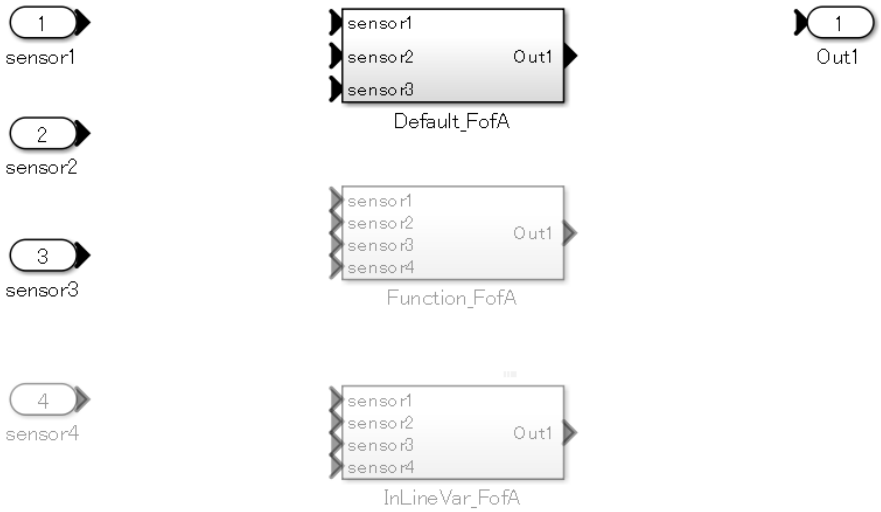
ルール

サブ ID	記述内容	カスタムパラメーター
a	[Variant Subsystem]とその配下のサブシステム、[ModelReference]の入出力数を統一します。	-

【正】
配下のサブシステムの入力数が同じです。

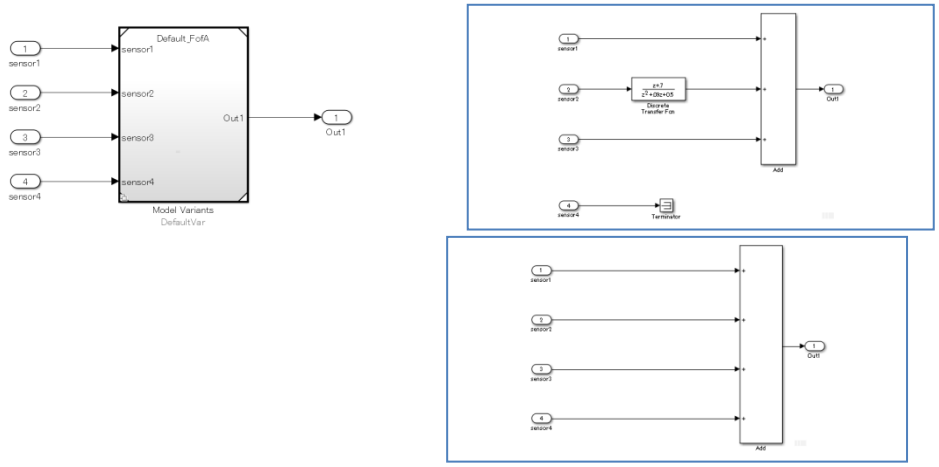


【誤】
配下のサブシステムの入力数が異なります。

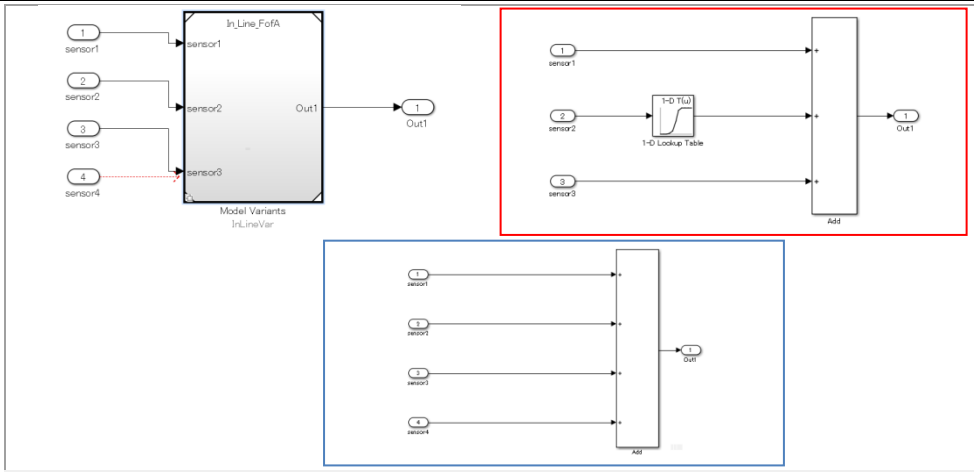


b	[Model Variant]と配下のモデルの入出力数を統一します。	-

【正】
配下のモデルの入力数が同じです。



【誤】
配下のモデルの入力数が異なります。



根拠	
サブ ID	**記述内容**
ab	・入出力数が異なることで、意図せず信号が未接続になるおそれがあります。

5.1.3. na_0036：既定のバリアント

ルール ID：タイトル	na_0036：既定のバリアント	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	[Variant Subsystem]は全て、必ず 1 つのサブシステムが選択されるように構成します。これは、以下のいずれかの方法で実現することができます。 ・[Variant Subsystem]の既定のバリアント(default)を使用します。 ・条件信号が取りうる全ての値を完全にカバーするように条件を定義します。たとえば、論理型の信号が真の値の場合と偽の値の場合について条件を定義します。	-

【正】
既定のバリアントを使用します。

バリアントの選択（子サブシステムまたは Model ブロックのリスト）

名前（読み取り専用）	バリアント制御	条件（読み取り専用）
Default_FofA	(default)	(N/A)
Function_FofA	functionVar	FUNC==1
In_Line_FofA	inLineVar	FUNC==2

【正】
FUNC が論理型である場合

バリアントの選択（子サブシステムまたは Model ブロックのリスト）

名前（読み取り専用）	バリアント制御	条件（読み取り専用）
Function_FofA	functionVar	FUNC==1
In_Line_FofA	inLineVar2	FUNC==0

【誤】
FUNC が1でも2でもない場合、アクティブなサブシステムは存在しません。

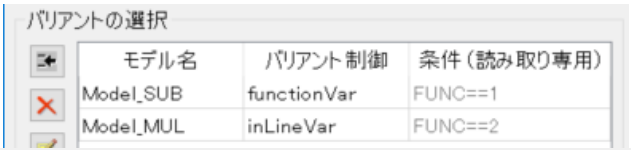
バリアントの選択（子サブシステムまたは Model ブロックのリスト）

名前（読み取り専用）	バリアント制御	条件（読み取り専用）
Function_FofA	functionVar	FUNC==1
In_Line_FofA	inLineVar	FUNC==2

b	[Model Variants]は、条件変数信号が取りうる全ての値を完全にカバーするように条件を定義し、必ず 1 つのサブシステムが選択されるように構成します。たとえば、論理型の変数信号が真の値の場合と偽の値の場合について条件を定義します。	-

【正】
条件変数が取りうる全ての値を完全にカバーするように条件を定義します。

【誤】
FUNC が 1 でも 2 でもない場合、アクティブなサブシステムは存在しません。



根拠	
サブ ID	**記述内容**
ab	・条件の抜け漏れを防ぎます。 ・条件の抜け漏れがあると、アクティブなサブシステムが存在しない状態になるおそれがあります。

5.1.4. na_0031：列挙型の既定値の定義

ルール ID：タイトル	**na_0031：列挙型の既定値の定義**	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	列挙型の既定値は、getDefaultValue を用いて明示的に定義します。	-

【正】

```
classdef(Enumeration) BasicColors < Simulink.IntEnumType
    enumeration
        Red(0)
        Yellow(1)
        Blue(2)
    end
    methods(Static = true)
        function retVal = getDefaultValue()
            retVal = BasicColors.Red;
        end
    end
end
```

【誤】

```
classdef(Enumeration) BasicColors < Simulink.IntEnumType
    enumeration
        Red(0)
        Yellow(1)
        Blue(2)
    end
end
```

根拠

サブ ID	記述内容
a	・列挙型に明示的な既定値を設定しない場合、enumeration の最初に記載した文字列が既定値として定義されるため、意図した値にならないことがあります。

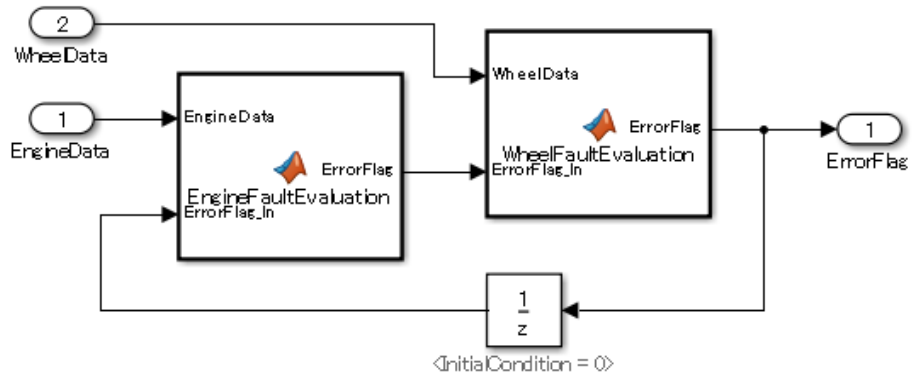
5.1.5. na_0034：MATLAB Function ブロックの入出力設定

ルール ID：タイトル	na_0034：MATLAB Function ブロックの入出力設定	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	[MATLAB Function]への全ての入出力に対し、モデルエクスプローラにてデータ型を定義します。	-
根拠		
サブ ID	**記述内容**	
a	・[MATLAB Function]への全ての入出力のデータ型を定義することで、シミュレーションエラーや予期せぬ動作の防止に繋がります。	

5.1.6. na_0024：MATLAB Function 間における共通データ

ルール ID：タイトル	na_0024：MATLAB Function 間における共通データ	
ルール		
サブ ID	**記述内容**	**カスタムパラメーター**
a	[MATLAB Function]間の共通データは信号線で結線します。	-

【正】

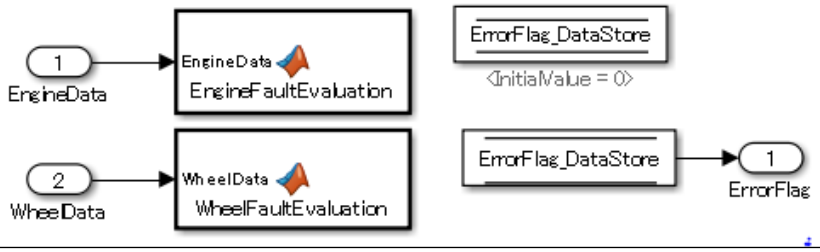


```
function ErrorFlag =
EngineFaultEvaluation(EngineData,ErrorFlag_In)
%#codegen
    RPM_HIGH = 10000;
    RPM_LOW = 10;
    HIGHRPMFAULT = 2^1;
    LOWRPMFAULT = 2^2;
    ErrorFlag = ErrorFlag_In;
    if EngineData > RPM_HIGH
        ErrorFlag = bitor(ErrorFlag,HIGHRPMFAULT);
    end
    if EngineData < RPM_LOW
        ErrorFlag = bitor(ErrorFlag,LOWRPMFAULT);
    end
```

```
function ErrorFlag = WheelFaultEvaluation(WheelData,ErrorFlag_In)
%#codegen
    SLIP_HIGH = 1000;
    WHEELSLIP = 2^3;
    ErrorFlag = ErrorFlag_In;
    if WheelData > SLIP_HIGH
        ErrorFlag = bitor(ErrorFlag,WHEELSLIP);
    end
end
```

【誤】

この例は、それほど可読性は悪くありませんが、本ルールを採用すると、このようなパターンを使用できなくなります。




```
function EngineFaultEvaluation(EngineData)
%#codegen
    global ErrorFlag_DataStore
    RPM_HIGH = 10000;
    RPM_LOW = 10;
    HIGHRPMFAULT = 2^1;
    LOWRPMFAULT = 2^2;
    if EngineData > RPM_HIGH
        ErrorFlag_DataStore =
bitor(ErrorFlag_DataStore,HIGHRPMFAULT);
    end
    if EngineData < RPM_LOW
        ErrorFlag_DataStore =
bitor(ErrorFlag_DataStore,LOWRPMFAULT);
```

```
function WheelFaultEvaluation(WheelData)
%#codegen
    global ErrorFlag_DataStore
    SLIP_HIGH = 1000;
    WHEELSLIP = 2^3;
    if WheelData > SLIP_HIGH
        ErrorFlag_DataStore =
bitor(ErrorFlag_DataStore,WHEELSLIP);
    end
```

根拠

サブ ID	記述内容
a	・データストアを使用するとデータフローの可読性低下、更新参照タイミングの設計ミスに繋がります。

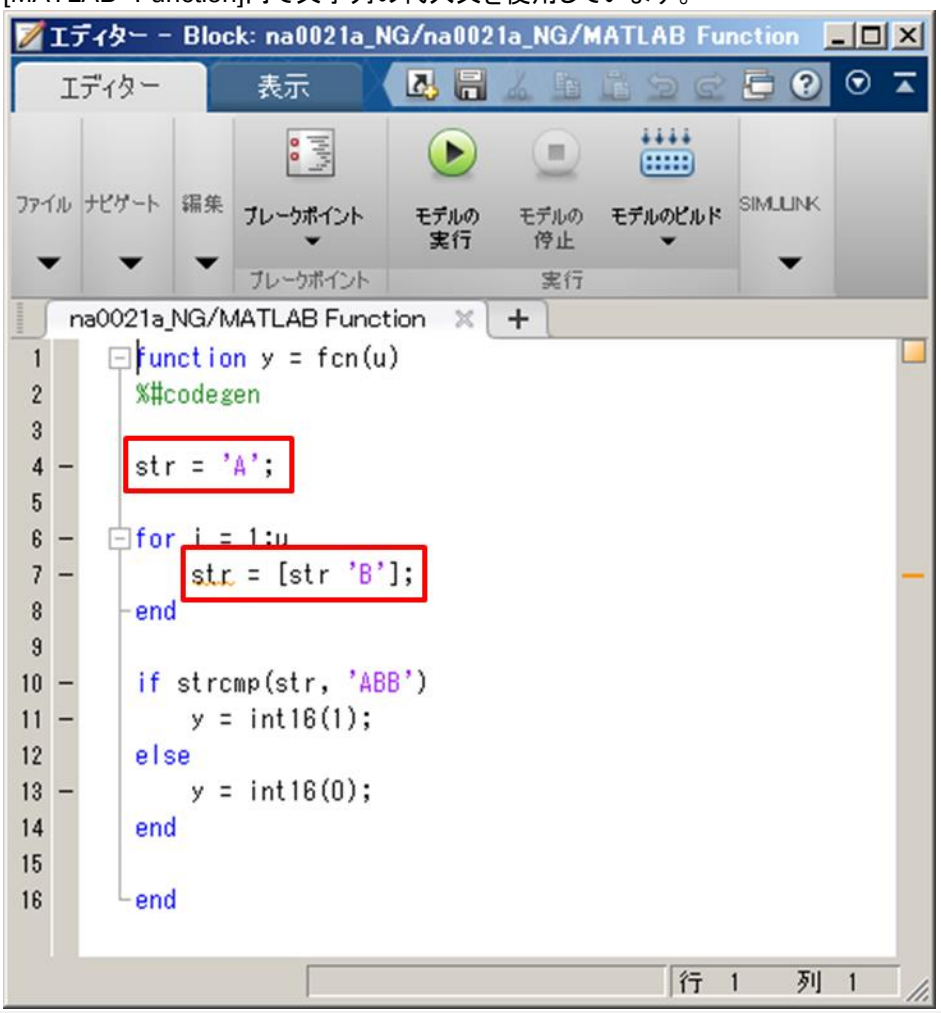
5.1.7. na_0021：MATLAB Function における文字列

ルール ID：タイトル	na_0021：MATLAB Function における文字列

ルール

サブ ID	記述内容	カスタムパラメーター
a	[MATLAB Function]内には文字列の代入文を使用しません。	-

【誤】
[MATLAB Function]内で文字列の代入文を使用しています。



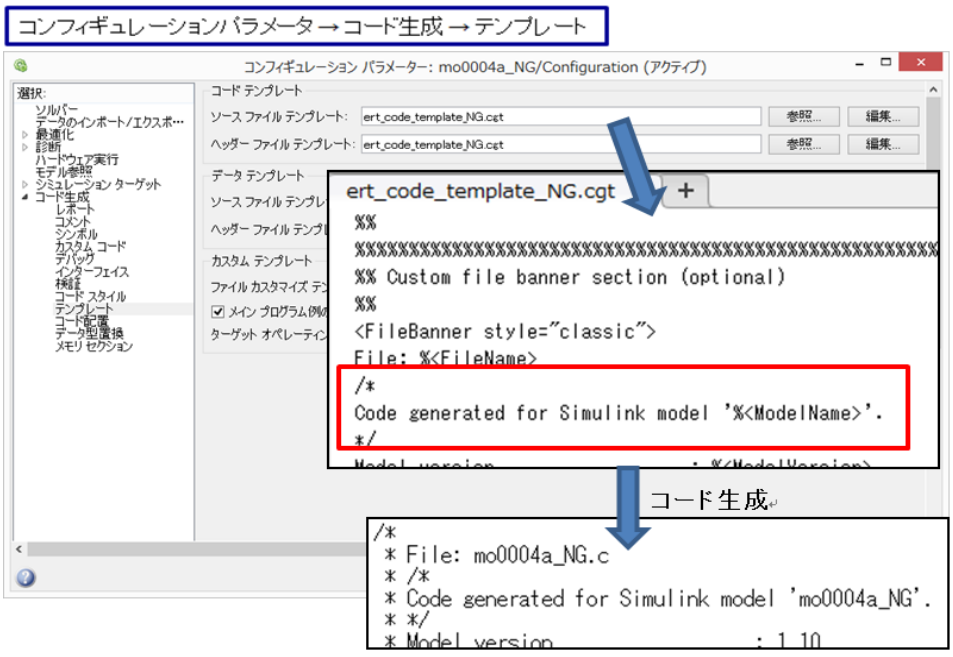
根拠

サブ ID	記述内容
a	・[MATLAB Function]は文字列を文字配列で格納します。 そのため、同一の変数に異なる長さの文字列を格納すると、動的メモリ割り当てに対応していないため、その文字列を格納することができず、エラーが生じます。 (文字列を Switch Case 文に使用したい場合は、列挙型の使用を検討してください)

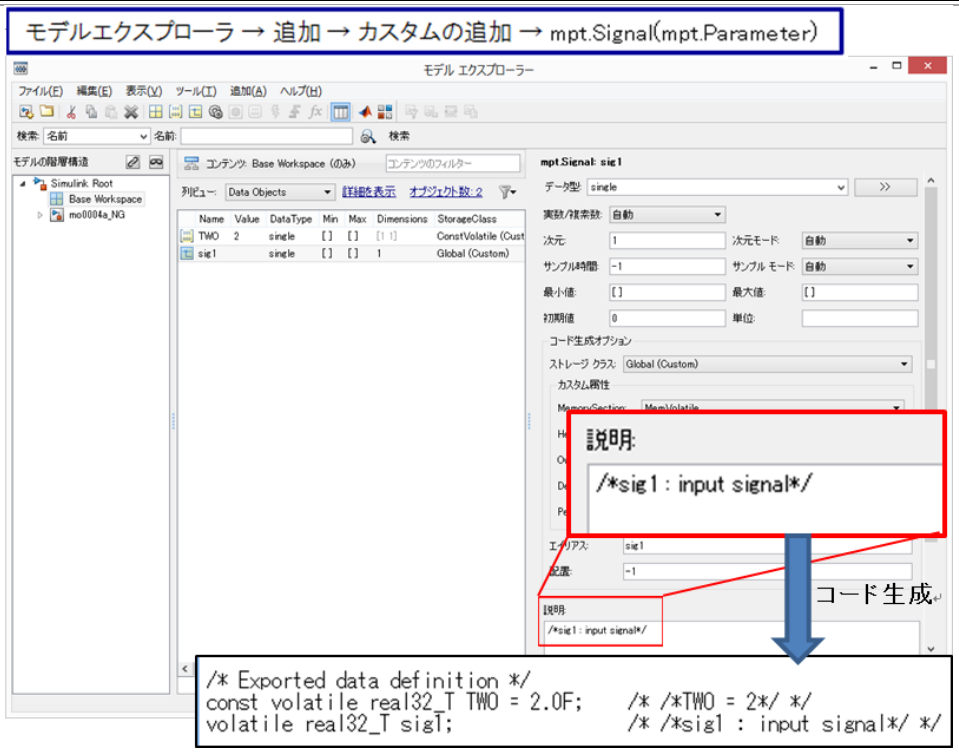
5.1.8. jc_0801 : コメント記号 /* 、*/ の使用禁止

ルール ID : タイトル	jc_0801 :コメント記号/*、*/の使用禁止	
ルール		
サブ ID	記述内容	カスタムパラメーター
a	コード生成時にコメント記号が付与される以下の箇所には、コメント記号/*、*/を使用しません。 ・cgt ファイル内 ・mpt Signal の[説明] ・mpt Parameter の[説明]	-

【誤】
cgt ファイル内



【誤】
mpt Signal の[説明](mpt.Parameter も同様)



根拠

サブ ID	記述内容
a	・コード生成時に自動的にコメント記号/*、*/が付与されるため、意図せずコメントが入れ子になり、意図と異なる動作になる可能性があります。

6. 根拠分類・関連一覧

6.1. 根拠分類

根拠には、ガイドラインを推奨する以下の理由の内、1 つ以上が選択されています。

1. 可読性
   - ・モデルのグラフィカルな理解度向上
   - ・モデルの機能を解析する可読性向上
   - ・接続ミスの防止
   - ・ コメントなど
2. シミュレーション、検証
   - ・シミュレーションが実行できる仕組み
   - ・テストの容易化
3. コード生成
   - ・コード効率の向上(ROM、RAM 効率)
   - ・ 生成コードのロバスト性確保

6.2. 関連

参考となる他のガイドライン ID を記載します。
MAAB ガイドライン以外に下記のガイドラインを参考にしています。

・ Modeling Guidelines for Code Generation(cgsl_)
・ Modeling Guidelines for High-Integrity Systems(hisl_)
・ Orion GN&C MATLAB/Simulink Standards(Orion_[bn_、ek_、im_、jr_、jh])
  Ver3.0 から、NASA オリオン・スタイル・ガイドラインの番号が関連として加わりました。
  http://www.mathworks.co.jp/aerospace-defense/standards/nasa.html
・ MISRA SLSF ガイドライン(MISRA AC SLSF_)
・ Ver4.0 から、MISRA 発行の MISRA AC SLSF のガイドラインが関連として加わりました。

上記のガイドラインの記載内容は、本ガイドラインの本文に掲載されていません。
本ガイドラインの内容と上記ガイドラインの内容が異なる場合もあります。
あくまで正は MAAB ルールとしてコントローラモデリングに必要なルールが記載されています。
上記ガイドラインの全ての番号とは対応しません。

6.3. 一覧表

ルール ID	サブ ID	根拠分類			関連
		可読性	シミュレーション	コード生成	
ar_0001	a	○	○	○	
	b	○	○	○	
	c	○	○	○	
	d	○			
	e	○			
	f	○	○	○	
	g	○	○	○	
ar_0002	a	○	○	○	
	b	○			
	c	○			
	d	○			
	e	○			
	f	○	○		
jc_0241	a		○		
jc_0242	a		○	○	
jc_0201	a	○		○	

	b	○		○	
	c	○		○	
	d	○		○	
	e	○		○	
	f	○		○	
jc_0231	a			○	
	b			○	
	c	○		○	
	d	○		○	
	e	○		○	
	f	○		○	
jc_0211	a	○		○	
	b	○		○	
	c	○		○	
	d	○		○	
	e	○		○	
jc_0243	a	○		○	
jc_0247	a	○		○	
jc_0244	a	○		○	
jc_0222	a	○		○	
	b	○		○	
	c	○		○	
	d	○			
	e	○			
	f	○		○	
jc_0232	a	○		○	
	b	○	○	○	
	c	○	○	○	
	d	○			
	e	○			
	f	○		○	
jc_0245	a	○		○	
jc_0246	a	○		○	
jc_0795	a	○		○	
	b	○		○	
	c	○		○	
	d	○		○	
jc_0796	a	○		○	
jc_0791	a	○	○		
	b	○	○		
	c	○	○		
jc_0792	a	○	○		
	b	○			
db_0043	a	○			
	b	○			
	c	○			MISRA AC SLSF 050B
	d	○			MISRA AC SLSF 050B
jc_0644	a	○	○		
jc_0011	a			○	

jc_0642	a			○	MISRA AC SLSF 008B
jc_0806	a		○		hisl_0005C
	b		○		
	c		○		
na_0004	a	○			MISRA AC SLSF 023A
jm_0002	a	○			
db_0142	a	○			
jc_0061	a	○			MISRA AC SLSF 026A
db_0140	a	○			MISRA AC SLSF 026E
jc_0603	a	○			MISRA AC SLSF 022
	b	○			MISRA AC SLSF 022
jc_0604	a	○			MISRA AC SLSF 024A
db_0081	a	○		○	
	b	○		○	
db_0032	a1	○			
	a2	○			
	b	○			
	c	○			
	d	○			
	e	○			
db_0141	a	○			
	b	○			
	c	○			
	d	○			
jc_0110	a	○			
jc_0171	a	○			
	b	○		○	
jc_0602	a	○			MISRA AC SLSF 036C
jc_0281	a1	○			MISRA AC SLSF 026C
	a2	○			MISRA AC SLSF 026C
	a3	○			MISRA AC SLSF 026C
	a4	○			MISRA AC SLSF 026C
	b1	○			
	b2	○			
	b3	○			
	b4	○			
db_0143	a	○	○		
db_0144	a	○			
	b		○	○	
jc_0653	a	○	○	○	
na_0010	a		○		MISRA AC SLSF 015A,B
	b		○		MISRA AC SLSF 015A,B
	c		○		MISRA AC SLSF 015A,B
	d		○		MISRA AC SLSF 015A,B
jc_0008	a	○			MISRA AC SLSF 027C,D,F,G,I,J
jc_0009	a	○	○		
	b	○	○		
db_0097	a	○			MISRA AC SLSF 027A
	b	○			MISRA AC SLSF 027A

	c	○			MISRA AC SLSF 027A
db_0112	a1	○			cgsl_0101 hisl_0021
	a2	○			cgsl_0101 hisl_0021
db_0110	a	○		○	MISRA AC SLSF 006A
jc_0645	a			○	MISRA AC SLSF 006B
jc_0641	a		○		MISRA AC SLSF 009D
jc_0643	a		○	○	
db_0146	a	○			
	b	○			
jc_0640	a		○		MISRA AC SLSF 007
jc_0659	a	○	○		
na_0003	a	○	○		
jc_0656	a	○			hisl_0010 hisl_0011 MISRA AC SLSF 011B
jc_0657	a1	○		○	hisl_0010 hisl_0011 hisl_0015 MISRA AC SLSF 011B
	a2	○		○	hisl_0010 hisl_0011 hisl_0015 MISRA AC SLSF 011B
na_0002	a		○		
	b		○		
jc_0121	a	○			MISRA AC SLSF 010A
	b	○		○	MISRA AC SLSF 010A
	c			○	MISRA AC SLSF 010A
jc_0610	a	○		○	
	b			○	
jc_0611	a			○	
jc_0794	a		○	○	
jc_0805	a1		○	○	
	a2		○	○	hisl_001A
	b			○	hisl_001A
	c1		○	○	hisl_0003B
	c2		○	○	hisl_0003A
	d		○	○	jc_0794 hisl_0028
	e		○	○	hisl_0004A,B
	f1		○	○	hisl_0004C
	f2		○	○	hisl_0004C
	g		○	○	jc_0794 hisl_0002B
	h		○	○	jc_0794 hisl_0002A
	i		○	○	jc_0794 hisl_0005A
	j		○	○	hisl_0005B
jc_0622	a	○	○		

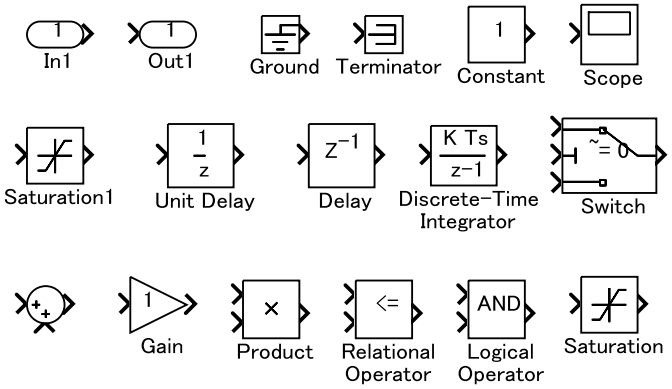
jc_0621	a	○			
jc_0131	a	○			
jc_0800	a		○	○	
jc_0626	a	○			
	b			○	
jc_0623	a	○	○		
jc_0624	a	○		○	
	b	○		○	
jc_0627	a		○		
	b		○		
jc_0628	a		○		MISRA AC SLSF 002A
jc_0651	a	○			MISRA AC SLSF 002A
db_0042	a	○			
	b	○			
	c	○			
jc_0081	a	○			MISRA AC SLSF 036C
na_0011	a	○	○		
jc_0161	a			○	
	b	○	○		
jc_0141	a	○	○		
jc_0650	a			○	
jc_0630	a		○		MISRA AC SLSF 013A
	b		○		MISRA AC SLSF 013B hisl_0022
	c		○		hisl_0022
db_0122	a		○	○	
jc_0712	a	○	○	○	MISRA AC SLSF 034D
jc_0700	a		○	○	MISRA AC SLSF 037G
db_0125	a		○		
	b		○		
	c		○		
	d	○	○		
jc_0701	a1	○	○	○	
	a2	○	○	○	
jc_0722	a	○	○		MISRA AC SLSF 037D
db_0126	a			○	
jc_0797	a	○		○	
	b	○		○	
db_0137	a	○	○	○	
jc_0721	a	○	○		MISRA AC SLSF 040B
db_0129	a	○			
	b	○			
	c	○			
	d	○			
	e	○			
jc_0531	a		○		MISRA AC SLSF 042A
	b	○			MISRA AC SLSF 042A
	c		○		MISRA AC SLSF 042B,C
	d	○			MISRA AC SLSF 051A
	e	○			MISRA AC SLSF 051A

	f	○			MISRA AC SLSF 042E
	g		○		MISRA AC SLSF 042D
jc_0723	a	○			
jc_0751	a		○	○	MISRA AC SLSF 043C
jc_0760	a	○			MISRA AC SLSF 053F
jc_0763	a1	○			MISRA AC SLSF 043F
	a2	○			
jc_0762	a	○		○	
db_0132	a	○	○		
	b	○	○		
jc_0773	a		○		MISRA AC SLSF 043C
	b		○		
jc_0775	a1	○			MISRA AC SLSF 053J
	a2	○			MISRA AC SLSF 053K
jc_0738	a			○	
	b	○			
jc_0790	a		○	○	
jc_0702	a	○	○	○	MISRA AC SLSF 048G,H
jm_0011	a	○	○	○	
jc_0491	a	○	○	○	
jm_0012	a1		○		MISRA AC SLSF 047A
	a2	○	○		MISRA AC SLSF 047A
	a3	○	○		MISRA AC SLSF 047A
jc_0733	a	○	○		MISRA AC SLSF 055A
	b	○	○		MISRA AC SLSF 055A
jc_0734	a	○	○		MISRA AC SLSF 055D
jc_0740	a	○	○	○	
jc_0741	a		○	○	
jc_0772	a		○	○	
jc_0753	a1	○	○		MISRA AC SLSF 043A
	a2	○	○		MISRA AC SLSF 043A
jc_0711	a1		○	○	MISRA AC SLSF 038B
	a2		○	○	MISRA AC SLSF 038B
db_0127	a1			○	
	a2	○		○	
jc_0481	a		○	○	
na_0001	a	○	○	○	
	b1	○	○	○	
	b2	○	○	○	
	b3	○	○	○	
	c	○	○	○	
jc_0655	a	○		○	
jc_0451	a		○	○	
jc_0802	a	○	○	○	
jc_0803	a1		○	○	
	a2		○	○	
	b1		○	○	
	b2		○	○	
	c1		○	○	

	c2		○	○	
	d1		○	○	
	d2		○	○	
jc_0732	a	○			MISRA AC SLSF 052B
jc_0730	a	○	○	○	MISRA AC SLSF 052A
jc_0731	a	○			
jc_0501	a	○			
jc_0736	a	○			
	b	○			
	c	○			
jc_0739	a	○			MISRA AC SLSF 050F
jc_0770	a1	○			
	a2	○			
jc_0771	a1	○			
	a2	○			
jc_0752	a	○			
jc_0774	a	○			
jc_0511	a		○		
jc_0804	a	○	○	○	
na_0042	a	○			MISRA AC SLSF 048A
na_0039	a	○			MISRA AC SLSF 048A
na_0037	a		○		
na_0020	a		○		
	b		○		
na_0036	a		○		
	b		○		
na_0031	a		○		
na_0034	a		○		Orion_jh_0063
na_0024	a	○	○		Orion_ek_0003
na_0021	a		○		Orion_jh_0024
jc_0801	a			○	

7. ガイドライン用語説明

ガイドラインのルール本文のなかで、解釈の幅が広いと思われる用語について、ガイドラインとしての意味を定義します。

用語	ガイドライン上の定義
パラメーター	特に修飾なく「パラメーター」とある場合、ベースワークスペース / モデルワークスペースで定義される定数を指します。
MATLAB 組み込み関数	MATLAB の関数、スクリプトを指します。
MATLAB 予約語	MATLAB キーワード、及び、MATLAB 組み込み関数を指します。
ブロック	全てのブロックを指します(Type=Block)。 特に断り書きがない限り、サブシステム、[Model]、[Chart]等も全て含まれます。 標準 Simulink ライブラリのブロックには、基本ブロックと構造サブシステムの2種類があります。
基本ブロック	標準 Simulink ライブラリのビルトインブロックを“基本ブロック”とします。 ([Subsystem]のように内部の処理が未定義のブロックは、基本ブロックではありません。) 基本ブロックの例を以下に示します。 
構造サブシステム	サブシステム、[Model]、[Chart]、[MATLAB Function]等、構造を定義するための枠であり、ユーザーが内部の処理を記述するブロックを”構造サブシステム”とします。
サブシステム	内部を Simulink の基本図法でモデリングできるサブシステムを指します。 {BlockType}が"SubSystem"であっても、[Chart]、[MATLAB Function]といった内部を Simulink 基本図法以外で記述するブロックは含みません。[Model]は含みません。
条件付きサブシステム	条件入力ポートを持った構造サブシステムを指します。
Atomic サブシステム	{BlockType}が"SubSystem"で、構造サブシステムを単一ユニットとして実行するように設定したものを指します。 条件付きサブシステムは自動的に Atomic サブシステムになります。 [Chart]、[MATLAB Function]等もデフォルトでは Atomic サブシステムです。
ポートラベル名	構造サブシステムの入出力ポートのラベルを指します。 デフォルトではサブシステム内に配置された[Inport][Outport]のブロック名や、Stateflow の入力データ名／出力データ名、が表示されます。サブシステムをマスク化することで表示を変更することもできます。
条件入力ブロック	[Trigger]、[Enable]、[FunctionCall]、[Reset]を指します。

遅延ブロック	2つの意味で用います。 ① 代数ループ(循環参照)での実行順序を規定するために、ループ経路に配置する前回値参照ブロックを指します。 基本的に[UnitDelay]、[Memory]を使います。R2011b 以降では[Delay]ブロックも使用可能です。 ② 過去の値を保持する用途のブロックを指します。 上記に加え、[Tapped Delay]も使用可能です。
加減算ブロック	加算または減算を行う、{BlockType}が"Sum"のブロックを指します。 [Sum]、[Add]、[Subtract]、[Sum of Elements]などが該当します。
乗除算ブロック	乗算または除算を行う、{BlockType}が"Product"のブロックを指します。 [Product]、 [Divide]、[Product of Elements]などが該当します。
Stateflow ブロック	[Chart]、[State Transition Table]、[Truth Table]を指します。
マシンレベル	Stateflow ブロックのあるモデルのルートシステムを指します。 (直訳の場合：Stateflow ブロックのある Simulink サブシステムを指します。)
フローチャート	遷移条件と条件アクションを使い、遷移条件に対するアクションを記述したモデル部分を指します。始点は、デフォルト遷移線、もしくは、内部遷移線で、終端はコネクティブジャンクションです。始点から終端の間にステートは含みません。 グラフィカル関数内部はフローチャートとしてモデリングします。 また、ステート内部にフローチャートをモデリングすることも可能です。
ステートアクションタイプ	基本ステートアクションタイプ、組み合わせステートアクションタイプを指します。
基本ステートアクションタイプ	entry(en)、during(du)、exit(ex)を指します。
組み合わせステートアクションタイプ	基本ステートアクションを組み合わせたものを指します。 ・ entry(en)、during(du) ・ during(du)、exit(ex) ・ entry(en)、exit(ex) ・ entry(en)、during(du)、exit(ex)
ステート	Atomic サブチャートもステートに含まれます。

8. ガイドライン運用ルールの決定

ガイドラインを運用する為の運用ルール、プロセスについての説明を行います。

8.1. プロセス定義の必要性

自動車は、安全でなければなりません。安全な製品を開発するには、様々な取り組みが必要になります。シミュレーションを活用するモデルベース開発は、より安全なシステムを開発するのに適しています。しかし、ただシミュレーションを活用したから安全なシステムができるわけではありません。良き制御・良き機能の開発は重要ですが、それと同じぐらいプロセス定義や、使われる開発環境も重要になります。開発開始時に様々な取り決めを実施してから、安全なシステム設計を行います。

8.2. MATLAB / Simulink のバージョン

プロジェクトの開始時に、使用する MATLAB/Simulink のバージョンを決めます。
これは、プロセス毎に複数の MATLAB バージョンを混在させることも含まれます。
例えば、自動コード生成を行うバージョンを R2011b とし、形式手法を使った検証ツールボックス Simulink Design Verifier (SLDV)は R2013a を用いてテストケースを作り出し、テストケースをダウングレードして R2011b でコード生成および検証の実行を行うことも可能です。
プロジェクト毎にどの段階でどのソフトウェアバージョンを使うのか決め、その工程では必ず決められたバージョンを全員が使用する必要があります。
さらに、定期的に最新のバグレポートを確認する必要があります。バグの内容によっては、最新バージョンへの変更が必要な場合があります。一度決めたら、変更できないわけではありません。バグによる不具合発生のリスクと、バージョンアップによるリスクを適切に評価する必要があります。常に最新バージョンに変更できる体制を整え、何が最も安全なのか適切な評価と判断が必要になります。

8.3. MATLAB / Simulink 設定

MATLAB/Simulink の設定は、プロジェクトで統一した運用を行う必要があります。特に見た目に影響を与える Simulink 設定は、統一が必要です。以下に統一を推奨するオプション名を記載します。

- Simulink 環境設定
  - 新しいモデルのフォント規定値(ブロック・ライン・注釈)
- マスク(マスクの編集)
  - アイコンと端子
- 情報表示
  - ライブラリリンク
  - サンプル時間
  - (ブロック)ソートされた実行順序
  - (信号と端子)非スカラーラインを太く表示
  - (信号と端子)端子のデータ型

関連 ID：na_0004、db_0043

8.4. 使用可能なブロック

Simulink には沢山のブロックがあります。ブロックによっては、効率の良いコードが生成できたり、複数の基本ブロックを組み合わせた機能が一つのブロックで提供されているものもありますが、使用するブロックを限定し、決められた範囲で設計する必要があります。

運用ルールを決定する際は、以下の点に注意をしてください。

- 使用可能なブロックの定義は db_0143 を参考
- 余りにブロック種類数を少なくすると、ユーザーライブラリが増えたり、同一の機能に対しての記述がばらつき可読性が悪化したり、コード効率の悪化を招くなど、弊害が懸念されます
- 例外的に特殊なブロックを使う場合は管理者がユーザーライブラリとして登録する必要があります

8.5. 使用するコンフィギュレーションの設定

8.5.1. 最適化パラメータ設定

最適化のオプションは、自動コード生成で生成されるコードへの影響度が非常に大きい項目です。安全に配慮しなければならない製品では、安易に最適化を使用してはいけません。自社の製品の特性を良く理解し、製品に対して適切な安全レベルと一致した設定を行う必要があります。

例を挙げると、一般的な自動車用組込み製品では、計算速度を重視し、さらに RAM が少なく、ROM も小さい方が好ましいとされます。そのため、たとえば最適化の「条件付入力分岐実行」のペインは ON に設定します。これによって、[Switch]を用いた条件分岐の実行時に、条件成立側の片側しか計算されなくなり、計算速度が向上します。

一方で、航空業界ではこのペインを OFF に設定します。これは自動車とは対照的に、実行速度の安定化を重視し、条件成立側しか計算する必要が無くても、両方計算することで常に安定した計算時間で計算されることを好むためです。

8.5.2. その他のコンフィギュレーションの解説

・ハードウェア実行パラメータ設定

モデル システムのハードウェア特性 (シミュレーションおよびコード生成の両方の製品およびテスト ハードウェア設定のセットアップを含む) を記述します。

プロジェクトが使用するマイコンに合わせた適切な設定を行ってください。特に、符号付整数の除算の丸め定義を行わない場合、意図しないユーティリティ関数が挿入される事があります。

・モデル参照パラメータ設定

モデルリファレンスを使用する場合に指定します。

このモデルにその他のモデルを含めるオプション、その他のモデルをこのモデルに含めるオプション、およびシミュレーションとコード生成ターゲットのビルド オプションを指定します。

・シミュレーションターゲットパラメーター設定

[MATLAB Function]、[Stateflow]、または [Truth Table]を含むモデルのシミュレーション ターゲットを設定します。

8.5.3. コンフィギュレーションに関するルールの紹介

コンフィギュレーションの設定については、MathWorks 社が作った hisl、cgsl のガイドラインを参照してください。こちらのガイドラインには、バージョン毎に推奨パターンが記載されています。各プロジェクトのニーズによって採択してください。

・ソルバー

- hisl_0040: [コンフィギュレーション パラメーター]、[ソルバー]、[シミュレーション時間]
- hisl_0041: [コンフィギュレーション パラメーター]、[ソルバー]、[ソルバーオプション]
- hisl_0042: [コンフィギュレーション パラメーター]、[ソルバー]、[タスクとサンプル時間オプション]

・診断

- hisl_0043: [コンフィギュレーション パラメーター]、[診断]、[ソルバー]
- hisl_0044: [コンフィギュレーション パラメーター]、[診断]、[サンプル時間]
- hisl_0301: [コンフィギュレーション パラメーター]、[診断]、[互換性]
- hisl_0302: [コンフィギュレーション パラメーター]、[診断]、[データ有効性]、[パラメーター]
- hisl_0303: [コンフィギュレーション パラメーター]、[診断]、[データ有効性]、[Merge ブロック]
- hisl_0304: [コンフィギュレーション パラメーター]、[診断]、[データ有効性]、[モデル初期化]
- hisl_0305: [コンフィギュレーション パラメーター]、[診断]、[データ有効性]、[デバッグ]
- hisl_0306: [コンフィギュレーション パラメーター]、[診断]、[接続性]、[信号]
- hisl_0307: [コンフィギュレーション パラメーター]、[診断]、[接続性]、[バス]
- hisl_0308: [コンフィギュレーション パラメーター]、[診断]、[接続性]、[関数の呼び出し]
- hisl_0309: [コンフィギュレーション パラメーター]、[診断]、[タイプ変換]
- hisl_0310: [コンフィギュレーション パラメーター]、[診断]、[モデル参照]
- hisl_0311: [コンフィギュレーション パラメーター]、[診断]、[Stateflow]

・最適化

- hisl_0045: [コンフィギュレーション パラメーター]、[最適化]、[Boolean データ (対 double) として論理信号を処理]
- hisl_0046: [コンフィギュレーション パラメーター]、[最適化]、[ブロック削減]
- hisl_0048: [コンフィギュレーション パラメーター]、[最適化]、[アプリケーションのライフスパン (日)]
- hisl_0051: [コンフィギュレーション パラメーター]、[最適化]、[信号とパラメーター]、[ループの展開のしきい値]
- hisl_0052: [コンフィギュレーション パラメーター]、[最適化]、[データの初期化]
- hisl_0053: [コンフィギュレーション パラメーター]、[最適化]、[浮動小数点から範囲外の値を含む整数変換へのコードを削除]
- hisl_0054: [コンフィギュレーション パラメーター]、[最適化]、[除算演算の例外処理に対して保護されたコードの削除]
- hisl_0055: 整合性の高いシステムのコード生成目的の優先順位付け

・モデリング ガイドライン
- cgsl_0301: コードの効率性のためのコード生成目的の優先順位付け
- cgsl_0302: マルチレート モデルとマルチタスク モデルの診断設定

8.6. 適用するガイドラインルール

8.6.1. 適用するガイドラインルールの採択とプロセスの設定

開発プロセス毎に、どのルールを適用するかプロジェクトの最初に決める必要があります。ガイドラインルールは、自動コード生成を行う最終段階だけで適用するのか、開発初期から段階に応じて適用するルールを切り替えるのか、開発プロセスと一致した適切な運用が必要です。

8.6.2. ガイドラインルール適用領域の設定と除外条件の明確化

ルールを適用する領域を決める必要があります。例えば、多くのルールは、AUTOSAR のアプリケーション領域を表現したモデルに適用を限定してください。ベーシックソフトウェア領域で使われる割り込みを実現したモデル、あるいは計算実行中に割り込みを禁止させるような処理を追加するモデルでは、専用のカスタム[S-function]や[Data Store Memory]を多用しなければ実現できない処理がたくさんあります。また、多くのユーザーが使用する専用のライブラリブロックの設計など、その領域を専門とするプロフェッショナルしか記載しない領域は、本ガイドラインが狙っている制限エリアではありません。

本ガイドラインの多くのルールは、一般的なエンジニアがモデルを編集する領域をターゲットとし、その領域で理解度の高いモデルを設計することを意図して作られています。一部の限られたプロフェッショナルが、テクニックを使わなければ実現できない領域は、その領域を限定し、プロフェッショナルしか触らない独自の仕組みを設けることで、ガイドラインの制約対象から除外することが出来ます。

また、RCP で動作させるモデル全体を、制御モデル用ガイドラインの適用対象とするのかについても、安易に全体を対象とするのではなく、領域を限定する必要があります。なぜならコード生成を行い、組込みマイコンに実装する部分と、実装しない部分を含むためです。実装されない RCP のためだけに作られたスケジューラモデルや、実機を動作させるための PWM 信号、CAN 信号などのドライバーに相当するブロックを含めたインターフェース部分は、本ガイドラインが対象とする制御モデルではありません。

上述のように一つのモデル内で、ガイドラインの適用領域を変更する場合、コード生成対象とそうでない領域を分けたモデル構造を用いる必要があり、それらの独自のルール追加も必要となります。

8.6.3. ガイドラインで規定されるパラメーターの決定

ルールの中にある数値やリストは、推奨標準値です。必ずしも守らなければならない数値ではありません。

例:

・「jc_0061：ブロック名の表示」

ブロック名を表示すべきブロックタイプ、非表示すべきブロックタイプ、どちらでも良いブロックタイプは、組織の教育状況、使用者のグループ・プロセスの違いで、それぞれ異なった設定値を決めることが可能です。

このように、ガイドライン内には数値だけでなく、対象となるブロックの種類などルール内で変更可能なカスタムパラメーターがあります。

単純にそのまま適用するのではなく、プロダクトの特徴や、使用する開発環境ツールに合わせて、様々なカスタムパラメーターを見直す必要があります。

8.6.4. ガイドラインチェッカー採択とプロセスの設定

チェックに自動チェッカーを適用するのか、レビューにて目視確認するのか最初に決める必要があります。チェッカーを、自社で作成し、それを使用することも可能です。自動チェックの項目が多いほどレビューの時間が削減されます。しかし、仮に全てが自動でチェックできるとしても、必ず上位者によるレビューを実施する必要があります。チェックは、自動だけでなく、レビューとの併用が有効です。

そして、どの時期にどのようなチェックを行うのかプロジェクト初期に決めます。さらに運用中にチェックルールの基準を見直す仕組みを設けてください。

8.6.5. モデル分析工程の追加

設計したモデルは、プロセスの途中でブロックの使用傾向や、記述パターンを分析し、適用するルールのリストを見直す工程を設定することが好ましいです。可能であれば、プロジェクトの初期の段階で、あらかじめルールの見直し時期を設定します。例えば、単純なモデルの分析は、[sldiagnostics]を使うことで、ブロックの使用頻度を調査できます。よく使うブロックと、あまり使われないブロックを認識し、適用ルールのリストを調整してください。また、フィードバックの[UnitDelay]など状態変数を持つブロックが画面のどこに配置するか、[UnitDelay]をサブシステムの外に出すか、中に入れるか、[abs]をサブシステムの出力側に設定するか、信号を受けてから、入力側で処理するか、などの傾向を測定し、記述パターンの統一を行うルールの追加と、それに合わせたモデルの修正工数をあらかじめ見込んでおくことが後の再利用性の向上に繋がります。

8.6.6. ルール変更の手続き

一度決まったルールが永遠に絶対厳守であることはありえません。

ルールを変更する場合は、正しい手順と手続きが必要です。設計者のニーズを聞き取り、何を変更すべきなのか検討します。ルール変更の要望がルールやブロックの使い方の誤解によるものであれば、ルールを改めるのではなく、教育の実施・追加をするべきです。しかし、自社の制御仕様や目的、あるいは実装されるマイコン等のハードウェアに起因するような制約があった場合は、必要に応じてルールを緩和する手続きを設ける必要があります。

9. モデルアーキテクチャの解説

ここではモデルアーキテクチャのルールを決めず、モデルベース開発に適したモデルアーキテクチャの考え方だけを共有する事を目的とします。

9.1. Simulink と Stateflow の使い分け

全てのシステムを Simulink、または Stateflow のどちらかに統一して記述することは可能です。

Stateflow だけを用いる場合、入出力と構造化のみ Simulink を必要としますが、Stateflow 内部で様々な数式処理が可能です。Simulink を用いる場合、複雑な状態変数も、[Switch Case]を駆使するなどの方法で実現することは可能です。

したがって、制御アルゴリズムの特定部分のモデリングに Simulink を使用するか Stateflow を使用するかは、どちらがより解りやすいかといった主観となります。組織内でどちらを選択するか決定します。

多くのケースで Stateflow は、Simulink よりも RAM 効率が悪化します。したがって、単純な数式を用いた計算では Simulink が有利です。その他にも、単純なフリップフロップ回路、[Relay]で演算できる状態変数なども Simulink の方が有利です。Simulink で記述できることを Stateflow で記述する場合、以下のリスクを考えて最適な手法を検討する必要があります。

- RAM の増加：Stateflow の入力・出力・内部変数は必ず可視化可能な RAM 領域を確保されます
- 計算式のエラー処理：内部で一般計算式を用いた場合、オーバーフロー防止をユーザーが設計します
- 機能の分割、分離：Stateflow 外で Simulink を使った計算を実施すると、全体が分割され、可読性が下がる場合があります。逆に上がる場合もあります。判断は非常に難しいです。

Stateflow を用いて、コードに近い最適表現を行うと、Simulink よりも効率の良いコードを得ることができるケースがありますが、その大半は理解しにくいモデルが出来上がってしまいます。コードがすでに存在する場合、あえて Stateflow でモデリングするのではなく、S-function を作成した方が有利です。Stateflow は特定の配列を指定した計算や、for ループを用いた計算は、Simulink よりも効率よく記載できますが、近年の MATLAB では MATLAB 言語を用いた記述も非常に便利になりました。場合によっては、MATLAB 言語を用いてモデリングする事も検討してください。

Stateflow モデルで下記のように定義される状態を扱う場合は、状態遷移として記述したほうが、可読性が向上します。

1. 同一の入力に対して異なる出力値が出力される
2. 状態が複数個存在する(3 個以上が目安)
3. 状態に数字ではなく、意味のある名称を付けられる
4. 状態内部で、初期化(初回)、実行中(2 回目以降)の区別が必要

例えば、フリップフロップ回路では、入力に対して異なる出力値が出力されます。しかし、状態変数は、0、1 の 2 つに限定され、boolean 型の数値を記憶しているだけで、それぞれの状態に意味のある名称を付ける事はできません。さらに、状態内で、初期化・実行中の区別もありません。つまりフリップフロップ回路は上記 4 つのうち、1 番しか該当しないので、Simulink 側が有利といえます。

Simulink/Stateflow のどちらで設計する必要があるかは、実装する必要がある課題に応じて複数人で協議して決定する必要があります。Stateflow にて状態遷移で実装するか、フローチャートで実装するかも協議によって決定します。

状態として扱うべき物は状態遷移、状態ではない条件分岐はフローチャートです。真理値表も条件分岐の実装方法に分類されます。

また、上記の状態を状態遷移として Stateflow を用いて設計する場合、制御系の「組込みマイコン」にソフトウェアとして実装する為には、{ステートマシンタイプ}を"Classic"にします。

Stateflow は、HDL Coder に対応しています。HDL Coder を使用する場合は、"Mealy"または、"Moore"にする必要があります。さらに、電気的な内部遅れの保証が必要な場合、"Moore"の方が適しています。

また、本ガイドラインでは、HDL Coder を使用するケースは記載されていません。制御系のソフトウェアを実装するための Simulink/Stateflow のガイドラインであることに注意してください。

9.2. コントローラモデルの階層構造

コントローラモデルの階層構造について分離概念、あるいはレイアウトの概念を参考例として示します。これは、ルールとしての明確な基準ではなく、モデリングの基本的な考え方です。

9.2.1. 階層構造の種類

・階層構造の構築方法
  - レイヤ内のスペース調整を主目的に、サブシステムに分割することは避ける必要があります
  - 各レイヤに以下の階層概念を割り当てて、それに基づいてサブシステムに分割する必要があります
  - 不要な階層概念をレイヤに割り当てる必要はありません
  - 複数の階層概念を一つのレイヤに割り当ててもかまいません

・階層概念

	階層概念	階層の目的
上位階層	機能レイヤ	大きな機能の分割
	スケジュールレイヤ	実行タイミング(サンプリング・順序)の表現
下位階層	サブ機能レイヤ	小さな機能の分割
	制御フローレイヤ	処理手順(入力→判断→出力、など)に沿った分割
	選択レイヤ	アクティブとなるサブシステムを切換えて実行する形式に分割([Merge]で出力を選択)
	データフローレイヤ	分割不可能な一つの計算を行うレイヤ

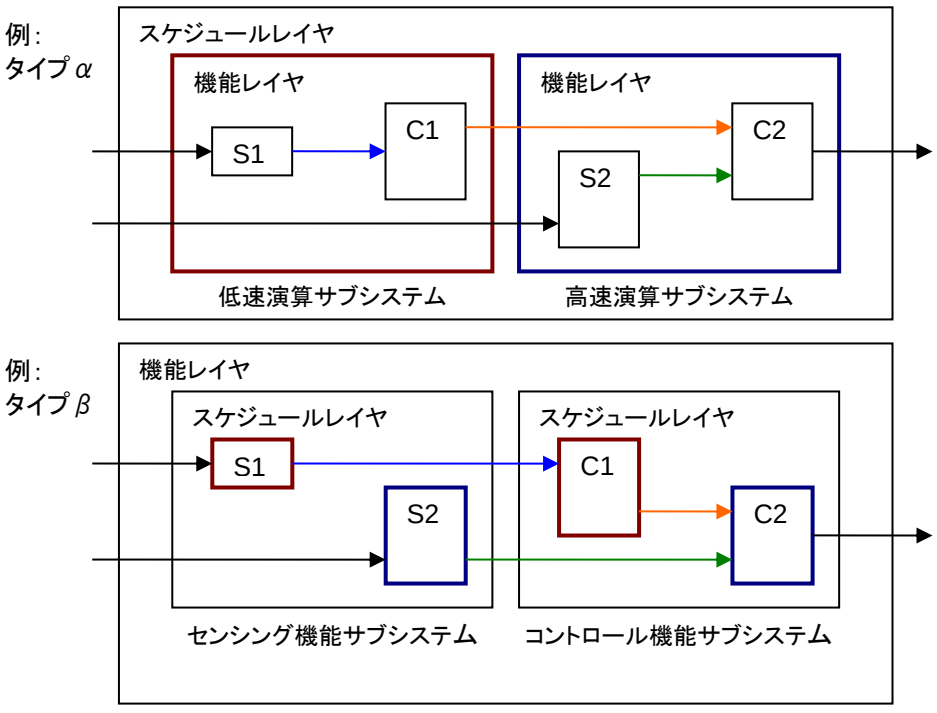
9.2.2. 上位階層のレイアウト方法

上位階層のレイアウト方法は主に3種類あります。

・簡易な制御モデル
  機能レイヤとスケジュールレイヤを同一レイヤで表現します。機能＝実行単位となります。
  例:制御モデルが一つのサンプリングのみであり、各機能が実行順序順に配置されている場合

・複雑な制御モデル　タイプ α
  スケジュールレイヤを上位に配置します。
  コードとの統合は楽になりますが、機能が分断されモデルとしての可読性が下がります。

・複雑な制御モデル　タイプ β
  機能レイヤを上位に配置し、個々の機能レイヤの下位にスケジュールレイヤを配置します。

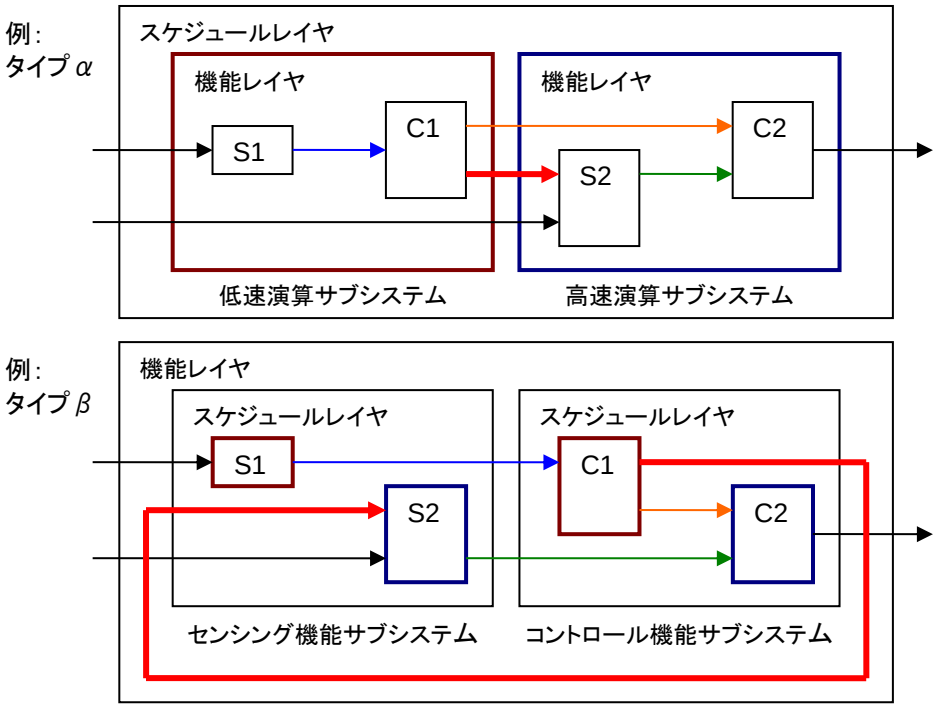


サブシステムの太線枠は、Atomic 化された設定

9.2.3. 機能レイヤ、サブ機能レイヤのモデリング方法

・機能でサブシステムに分割します。それぞれのサブシステムが「1つの機能」を表します。

・「1つの機能」が必ずしも実行単位になるとは限りません。そのため、それぞれのサブシステムを Atomic サブシステムにできるとは限りません。
(下記のタイプ βの例では、機能レイヤのサブシステムは、バーチャルサブシステムとするのが適切です。Atomic サブシステムに変更すると代数ループができます。)



・説明を付記する機能単位を定め、機能単位ごとに説明を記載します。
・複数の大きな機能が存在する場合、機能毎にモデル参照を用いてモデルを分割することも検討する必要があります。

9.2.4. スケジュールレイヤのモデリング方法

システムのサンプリング周期や実行の優先順位の設定を行います。
・複数のサンプリング周期を設定する場合の注意点
　異なるサンプリング周期のシステム同士を結線した場合、早いサンプリング周期で更新される信号は、遅いサンプリング周期で更新される信号が計算されていない時も信号が必要となるため、前回値を保持した RAM を別途生成します。この RAM の生成を最小限にするため、異なるサンプリング周期毎にシステムを分離してください。

・優先順位の設定について
　異なる複数の独立した機能を設計する場合に重要になります。できる限り全てのサブシステムの計算順序が結線によって任意に決定されることが望ましいです。
　優先順位については、異なるサンプリング周期間の優先順位と、同一サンプリング周期内の優先順位の、二つの順位を設定する必要があります。

・サンプリング周期・優先順位の設定方法
　設定方法は、大きく 2 種類あります。
1. サブシステムやブロックに対し、ブロックパラメーターの{サンプル時間}、ブロックプロパティの{優先順位}の設定を行います。
2. 条件付きサブシステムを用いて、ユーザーがスケジューラとあわせて、独自に優先順位を設定します。

　これには、コンフィギュレーションパラメーターの{タスクとサンプル時間オプション}、Atomic サブシステム設定、モデル参照の使用有無など、様々な条件でパターンが存在します。どの様なパターンを採用するかは、コードの実装方法と密接に結びつき、プロジェクトの状況によって大きく異なります。
　代表的な、大きく影響を受ける要因を下に記載します。
・モデル側

- モデルにおける複数のサンプリング周期の有無
- モデルにおける複数の独立した機能の有無
- モデル参照の有無
- モデルの個数(Simulink でコード生成したコードが複数個あるかどうか)

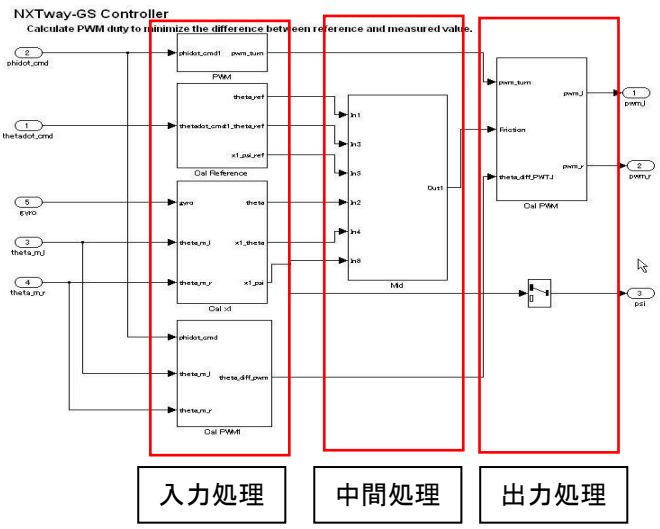
・コード側

- リアルタイム OS の適用有無
- 使用可能なサンプリング周期と実装する計算周期の一致性
- 適用領域(アプリケーション領域か、ベーシックソフトウェアか)
- コードのタイプ：AUTOSAR 準拠～ほぼ準拠～非準拠
- RAM、ROM の余裕度(特に RAM)

9.2.5. 制御フローレイヤのモデリング方法

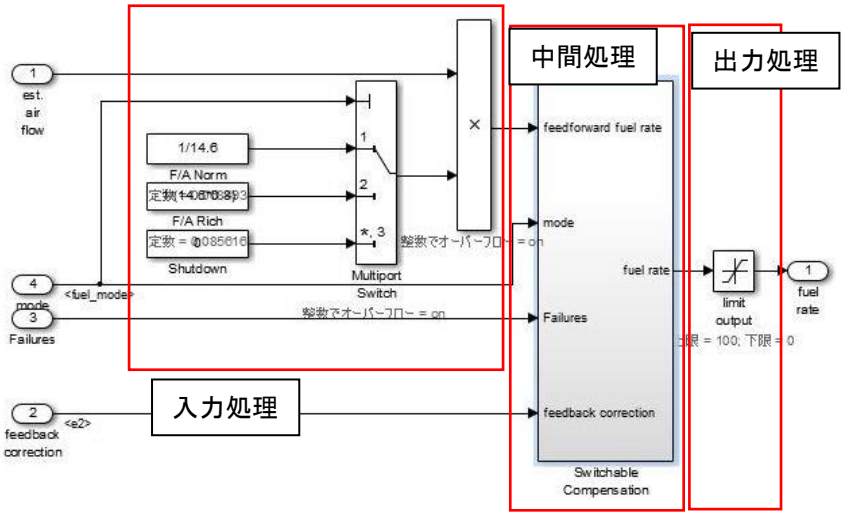
制御フローレイヤの配置は、入力処理、中間処理、出力処理の全体で一つの機能を表現するときに使われるレイヤです。ブロックや、サブシステムの配置に意味を持たせます。概念的には制御の基本である、入力処理、中間処理、出力処理の最大で 3 段階に分割することで複数個の混在する小機能をまとめます。全体の構成イメージは、データフローレイヤに近く、横に並べて表現します。データフローレイヤとの違いは、複数個のサブシステムとブロックで構成されます。

制御フローレイヤは、横方向が異なる処理の意味を持たせ、同じ意味を持つブロックは縦に並べます。



ブロック群のまとまりを横に並べて、それに仮称の意味を持たせ全体を整理します。

赤枠線が見えない処理の区切りを意味し、赤枠線がバーチャルオブジェクトというオブジェクトに対応します。アノテーションなどで区切りをマークすると解りやすくなります。



制御フローレイヤは、機能を持つブロックとの共存ができます。

サブ機能レイヤと、データフローレイヤの中間領域に位置します。

制御フローレイヤは、全てをデータフローレイヤで記述するとブロック数が多くなりすぎる場合に、一部の最小限の意味を与えられる単位でサブシステム化する場合に使用します。配置に意味を持たせることで、レイヤ内の構成を整理し、理解度を向上させます。また不必要な階層化を防ぐことで、保守性を高める効果もあります。

サブシステムとブロックの混在がなく、ブロックのみで構成されても、横のレイアウトで入力・中間・出力と分けることができれば制御フローレイヤとなります。

9.2.6. 選択レイヤのモデリング方法

選択レイヤは縦、または横に並べて書きます。(並びには意味はありません)

選択レイヤは、制御フローレイヤと混在します。

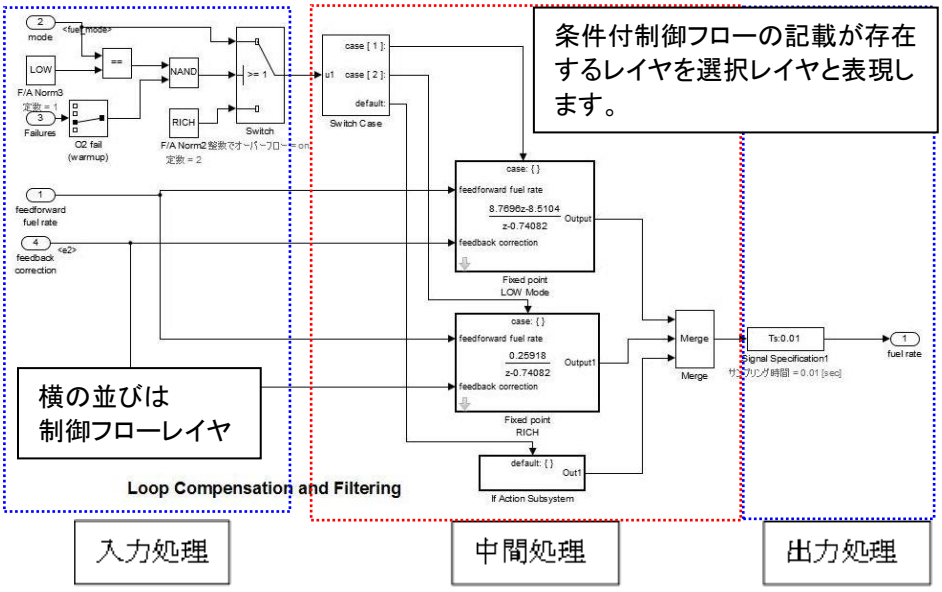
ここでは、赤枠の中の条件付き制御フローによって、いずれかしか動作しないサブシステムの切り替え機能が存在するので、選択レイヤと呼びます。全体は入力処理・中間処理(条件付き制御フロー)、出力処理が並んでいるので、制御フローレイヤとしても記載されます。制御フローレイヤの並びは、横方向に異なる処理の意味を持たせ、同じ意味を持つ並列処理は縦に並べます。選択レイヤは、並べ方に意味を持たせるのではなく、いずれかしか動作しないサブシステム群が記載されているレイヤを示します。

例:

・時系列で変化するアップ・ダウンの動作切り替えに連動した機能の切り替え

・初回(リセット直後)、2 回目以降の計算を切り替える設定の切り替え

・仕向け A、仕向け B の切り替え



9.2.7. データフローレイヤのモデリング方法

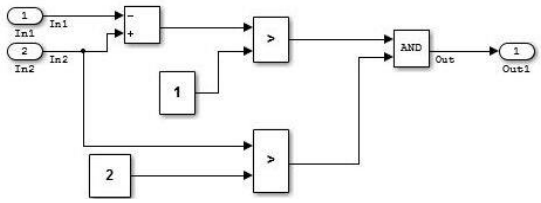
データフローレイヤは、制御フローレイヤや選択レイヤの下の階層になります。

全体で一つの機能を表現し、入力処理、中間処理、出力処理といった役割の分割ができない場合がデータフローレイヤです。例えば、一つの分割できない連続的な計算を行うシステムです。データフローレイヤでは、除外条件以外のサブシステムと共存しません。

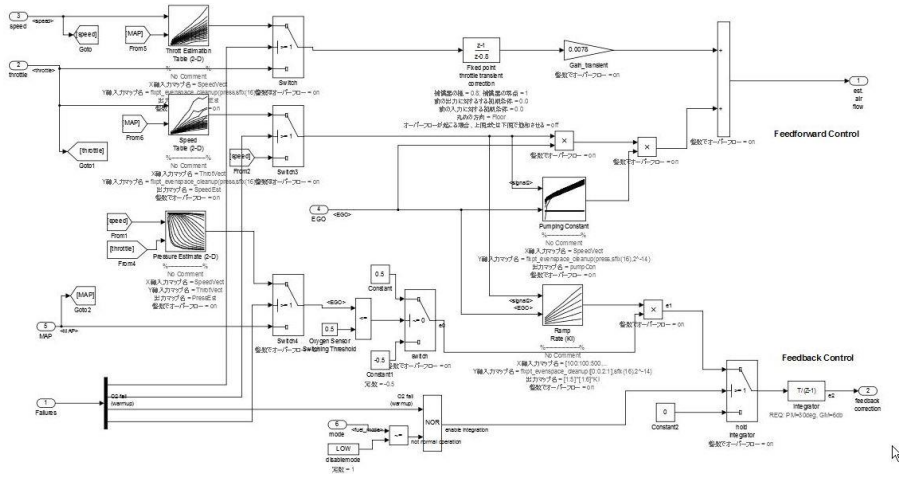
除外条件：下記のサブシステムとは共存できます。

・再利用可能な関数が設定されたサブシステム
・Simulink 標準で登録されたマスクサブシステム
・ユーザーがライブラリ登録したマスクサブシステム

単純なデータフローレイヤの例



複雑な構造のデータフローレイヤの例



上記の様にレイアウトでも明確に入力信号処理、中間処理の区別ができない場合をデータフローレイヤと表現します。

同じ入力信号からフィードフォワードとフィードバックの二つの答えを同時に計算するような場合に、データフローレイヤは複雑になります。このような場合、ブロックが多少多くても、明確な機能分割ができない場合は、途中をつまみ食いしたようなサブシステム化を設計すべきではありません。分割によって意味を持たせられる場合は、制御フローレイヤとして設計します。

9.2.8. 組込みの実装と Simulink モデルの関係

実際の組込みマイコンで動作させる為には、Simulink モデルから生成したコードをマイコンに組み込む必要があります。この際に、Simulink モデルが対象機能をどの程度モデル化したか、またどうやって組み込むのか、さらに組込み側のスケジュールがどの様に設定されているかで、Simulink モデルの構造に大きな影響を及ぼします。

実装する組込みマイコンのタスクと、Simulink モデルで使用しているタスクが異なる時に重大な影響を及ぼします。

9.3. AUTOSAR の概念

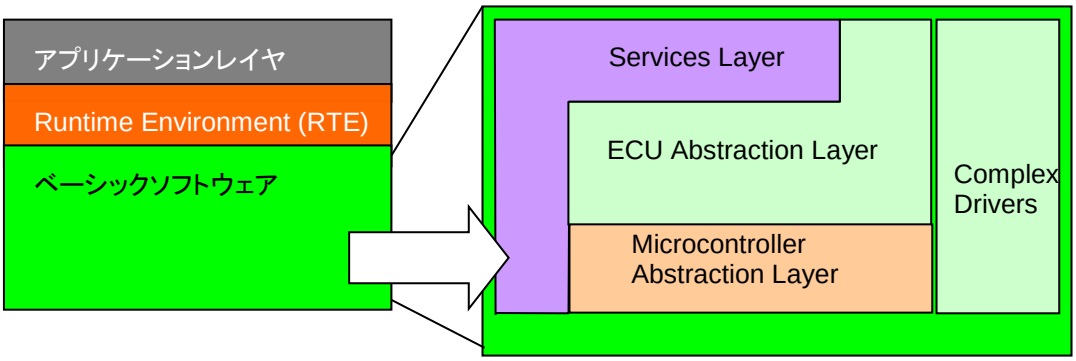
AUTOSAR(AUTomotive Open System ARchitecture)の概念を説明します。ユーザーは AUTOSAR プラットフォームに完全準拠する必要はありませんが、概念を理解し、モデリングの参考にしてください。

9.3.1. AUTOSAR ソフトウェアプラットフォームの概念

制御モデルを設計する場合、AUTOSAR のソフトウェアプラットフォームの概念を利用し、自分の設計しているモデルがアプリケーションとベーシックソフトウェアのどちらに分類されるか検討の必要があります。
そして、アプリケーションとベーシックソフトウェアが同一のモデル内に混在しないよう設計段階で分離する必要があります。

AUTOSAR ソフトウェアプラットフォームの概念
・高負荷・低速・定期的な処理は、アプリケーションレイヤ
・高速あるいは不定期なドライバー類は、ベーシックソフトウェア

AUTOSAR ソフトウェアプラットフォームは、下のような構造として表現します。



システム構成の例(Architecture – Overview of Software Layers Top view and Coarse view
AUTOSAR Release 4.0 Document Title： Layered Software Architecture を参照。)

例えば、エンジンの制御モデルを設計する場合、回転中のある動作を起点とする割り込みイベント時に、全ての計算を実行するモデルを作るのではありません。
アプリケーション領域で、定期計算によって全気筒で共通した計算を行います。例えば現在の排ガスの状況や、目標トルクの計算などです。そしてベーシックソフトウェア領域から不定期の割り込みが発生し、その時に RTE を通してアプリケーションで計算された計算結果を受け取り、実際にアクチュエータを動作させます。センサーによる割り込みイベントで値を計算し、アクチュエータ側へ渡す為の RAM に書き込み、アクチュエータはアクチュエータの割り込み周期で指定された RAM から値を取得し、シンプルな計算を行い、アクチュエータに指示を出力します。
ベーシックソフトウェア領域は、なるべく簡単な計算とし、各アプリケーション機能に対して共通の計算機能を配置する事が AUTOSAR のコンセプトです。
Simulink を用いて全体のモデリングを行う場合、割り込み領域での計算はシンプルであればあるほど望ましくなります。できる限りシンプルな制御をベーシックソフトウェアに配置し、その瞬間に計算する量を減らし、一定周期で計算された結果を取得してくるように設計します。ただし、必要な計算を除外すべきではありません。例えば、故障診断のような瞬間に判断すべき計算は、複雑であっても、計算させなければなりません。
そして、センサー系の割り込み処理、アクチュエータ、アプリケーションは全く別々の Simulink の関数から、それぞれ独立したソースファイルを出力します。
モデル上で同一にあったとしても、センサーのプログラムは Simulink 上では擬似的な動きしかできません。シミュレーションできない領域を、ソースコードを出力する目的のみで Simulink 上に記述する必要はありません。必要な場合は、擬似的なソースコードを作成し、S-function として実現します。

9.3.2. RCP と AUTOSAR ソフトウェアプラットフォーム

実は、RCP(Rapid Control Prototyping)等の機器を用いたモデリングは、AUTOSAR のソフトウェアプラットフォームの考え方に非常に近い考え方になっています。もちろん生成されたソースコードが AUTOSAR に準じている訳ではありません。例えば、RCP の I/O 系のソフトウェアは、ベンダーが提供する S-function をリンクさせます。そしてユーザーはアプリケーション領域の設計を行います。アプリケーション領域のユーザーが作った機能と、S-function は、Simulink ブロック結線で行っていますが、RAM のやり取りなど意識する必要はありません。

出力された C コードは、リアルタイム OS の上で動作し、I/O 系のソフトウェアと、Simulink で作ったアプリケーションは異なるソースファイルに出力され、リアルタイムで動作する部分と、割り込み処理で動作する部分が自然と分離されます。ユーザーがそれらのプラットフォームを意識する必要はなく、ベンダーが作った I/O 系 S-function は必要なタイミングで実行され、アプリケーションのモデルは、I/O 処理の中身やタイミングを意識することなくモデリングができ、動作もします。

実際の制御モデル・制御ソフトウェアもこのようなソフトウェア構造に近い方が、メリットが多くなります。RCP がソフトウェア構造をあまり意識することなくアプリケーション領域の開発に集中できるのは、AUTOSAR を意識せずとも AUTOSAR のソフトウェアプラットフォームの概念を使えているからです。このように AUTOSAR とモデルベース開発の親和性は高く、自社製品のベーシックソフトウェアが AUTOSAR に準じていれば、RCP で AUTOSAR に準拠したアプリケーション開発を行うことで、出力されたソフトウェアをそのまま使用することができます。逆に言うと、自社製品が AUTOSAR に準じていなければ、出力されたソフトウェアをカスタマイズしなければならなくなり、モデルベース開発の魅力がどんどん失われる事を意識して下さい。

9.4. シングルタスクとマルチタスク

組込みソフトウェアのスケジュール実現方法に、シングルタスクとマルチタスクの設定が存在します。本機能は、Simulink 機能の補足的な説明ですが、モデルアーキテクチャの設計に必要な機能になりますので、本章にて解説する事にします。

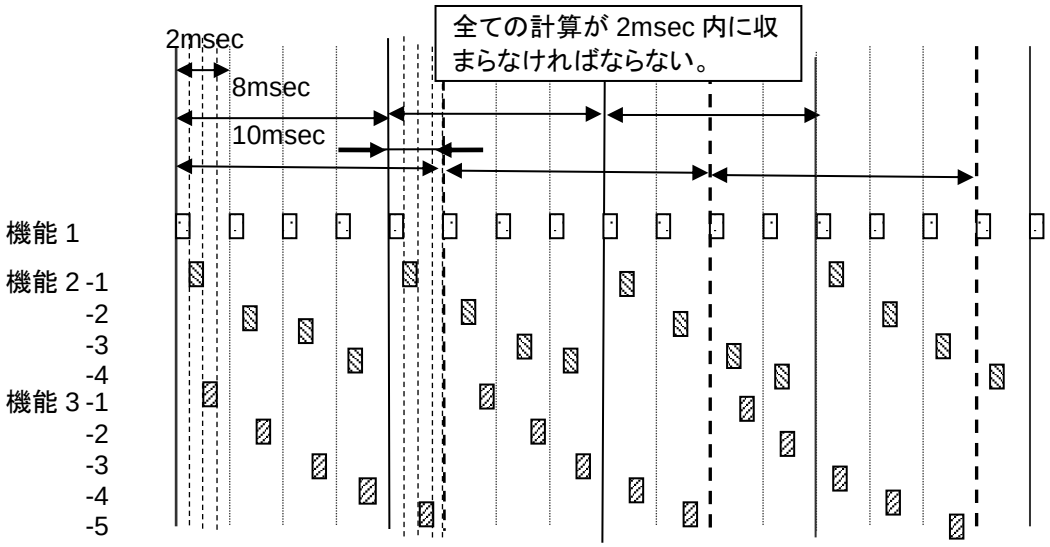
9.4.1. シングルタスク

シングルタスクでは、基本サンプリングですべての処理を実行します。したがって、基本サンプリングよりも長いサンプリングで処理したいタスクがある場合には、その機能をなるべく CPU 負荷が均一となるように分割し、基本サンプリングで処理します。ただし、機能を等分割する事ができない場合もあるので、全てのサイクルに関数が割り振られるわけではありません。

例えば、基本サンプリングが 2msec で、モデル内部に 2msec、8msec、10msec の 3 つのサンプリングが存在する場合を考えます。8msec の機能は 2msec で 4 回に 1 回、10mse の機能は 2msec で 5 回に 1 回実行するように、2msec 毎に実行回数をカウントし、カウントの回数によって指定されたサンプリングの機能が実行されます。注意すべき点は、2msec、8msec、10msec が全て同一の 2msec で計算される事がある点です。全ての計算を 2msec の中で終了させる必要があるので、このようなケースでは、8msec、10msec の機能を幾つかに分割し、全ての 2msec でほぼ均一の計算量になるように調整し、CPU 負荷を平坦化します。以下の図では、10msec の機能は 5 つに、8msec の機能は、4 つに分割しています。

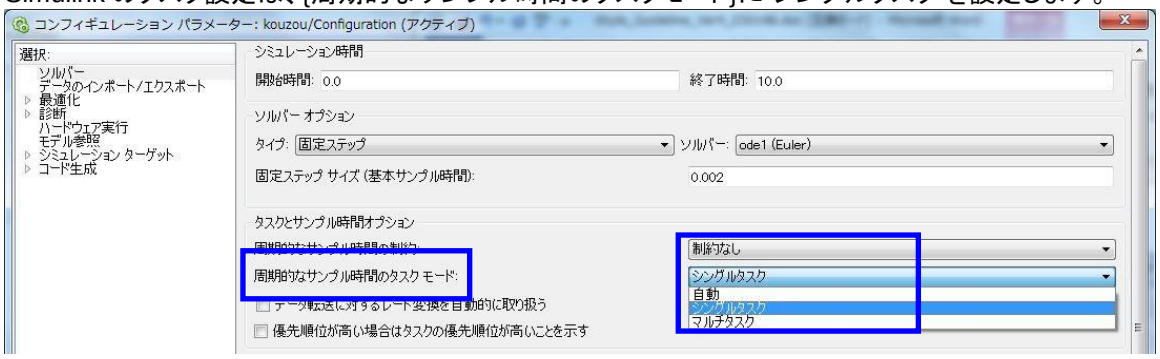
機能	基本周波数	オフセット
2-1	8msec	0msec
2-2	8msec	2msec
2-3	8msec	4msec
2-4	8msec	6msec

機能	基本周波数	オフセット
3-1	10msec	0msec
3-2	10msec	2msec
3-3	10msec	4msec
3-4	10msec	6msec
3-5	10msec	8msec

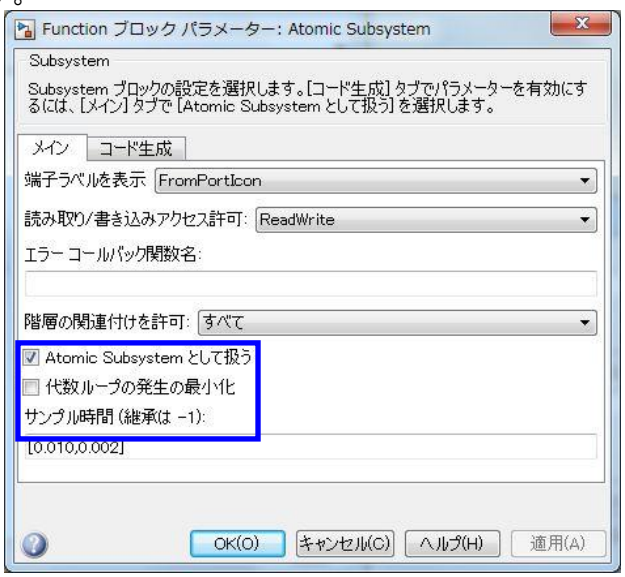


・タスクの分周設定方法

Simulink のタスク設定は、{周期的なサンプル時間のタスクモード}に”シングルタスク”を設定します。



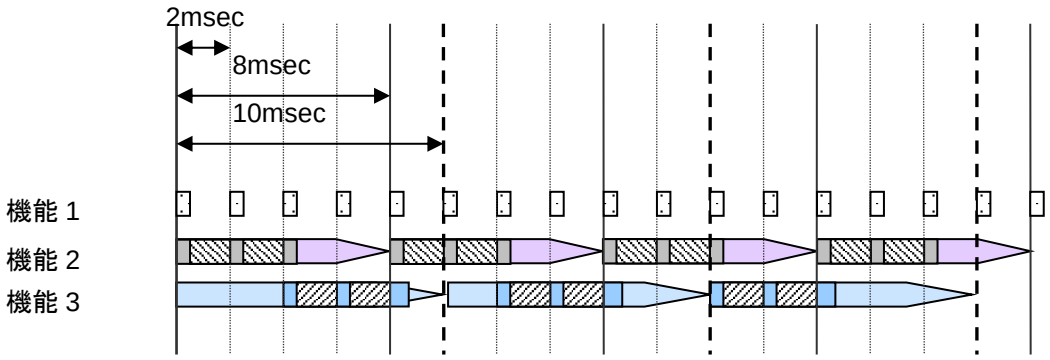
次に、サブシステムのサンプリング周期を設定する欄に、[サンプリング周期、オフセット]の形で入力します。サンプリング周期の設定が可能なサブシステムは、Atomic サブシステム化されたサブシステムです。



9.4.2. マルチタスク

マルチタスクは、マルチタスクに対応したリアルタイム OS を用いて実行します。先ほどのシングルタスクは、CPU 負荷均一化の作業は自動で行われるのではなく、人が機能の分割を行い、どのタスクに割り振るか、人が指定します。マルチタスクでは、現在の状況に合わせて、自動的に CPU が計算を行っていくので、

人が細かい設定を行う必要はありません。優先度の高いタスクから順に計算を行い、結果を出力しますが、タスクの優先度は人が指定します。早いタスクの優先順位を高く設定するケースが多いです。同タスクに関しては、実行順序を人が決定します。



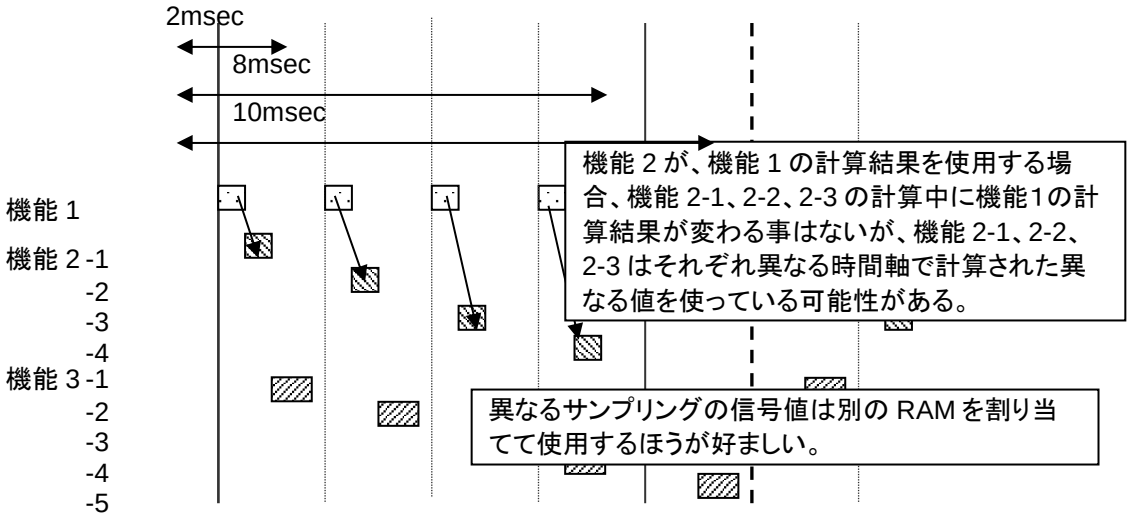
遅いタスクを含めて周期内で計算が終了する事が重視され、重要度の高い処理が終わり、CPU が空けば、次の重要度の高いシステムの計算を行います。計算中に重要度の高い計算処理がはいれば、重要度の低い計算を途中でやめて、重要度の高い計算処理を行います。

9.4.3. サンプリング違いのサブシステムを結線する影響

サンプリング周期 10msec のサブシステム A の出力を、サンプリング周期 20msec のサブシステム B が使用する場合、サブシステム B が計算している途中でサブシステム A の出力結果が変わってしまう可能性があります。値が途中で変化すると、サブシステム B の計算結果が想定外の結果になるケースがありえます。例えばサブシステム B の最初の計算でサブシステム A の出力値と比較し、その結果によって条件判定を行い計算します。この時点では比較結果が true だとします。更にサブシステム B の最後で再び比較を実行すると、A の結果が変わっており、false となる可能性があります。一般的にはこのような機能開発では、両方共が true、true を前提に設定されていますが、true、false となり想定外の計算結果となり得るケースが想定されます。その様な不具合を防止する為、一般的にはタスクの変化がある場合には、サブシステム A の出力結果をサブシステム B が使用する場合は、サブシステム A の出力信号とは別の RAM を用意し、直前に値を固定します。つまり、サブシステム A の値が途中で変化しても、サブシステム B が見ている値は別の RAM なので、影響が現れません。

Simulink 上でモデルを作り、Simulink 上で異なるサンプリング周期のサブシステムを結合する場合、Simulink が自動的に必要な RAM を確保します。

しかし、ハンドコードとの統合で、異なるサンプリングの入力値を取得する場合は、組込み作業を実施するエンジニアがそれらの設定を設計します。AUTOSAR を用いた RTW の概念では、全て受け側、書き出し側で異なる RAM を定義します。



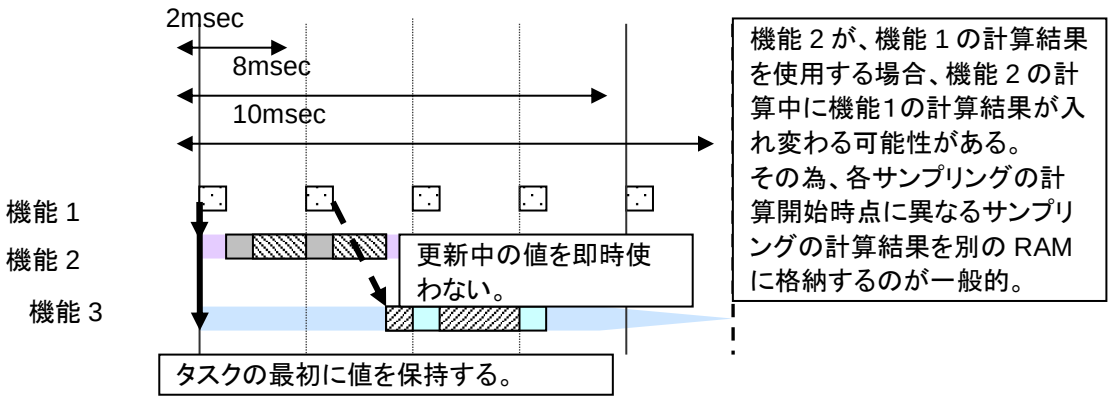
シングルタスクの場合

同じ 2msec 内では、同じ信号値ですが、異なる 2msec では、一つ前と異なる計算値になっている事に注意が必要です。機能 2-1 と 2-2 が機能 1 の信号 A を使っていた場合、2-1 と 2-2 は、異なる時間の結果を使用する事になるので、注意が必要です。

マルチタスクの場合

マルチタスクは、いつの時点の計算結果を使っているか、特定することができません。マルチタスクでは必ずタスク違いの信号を新たな RAM に格納して使用します。

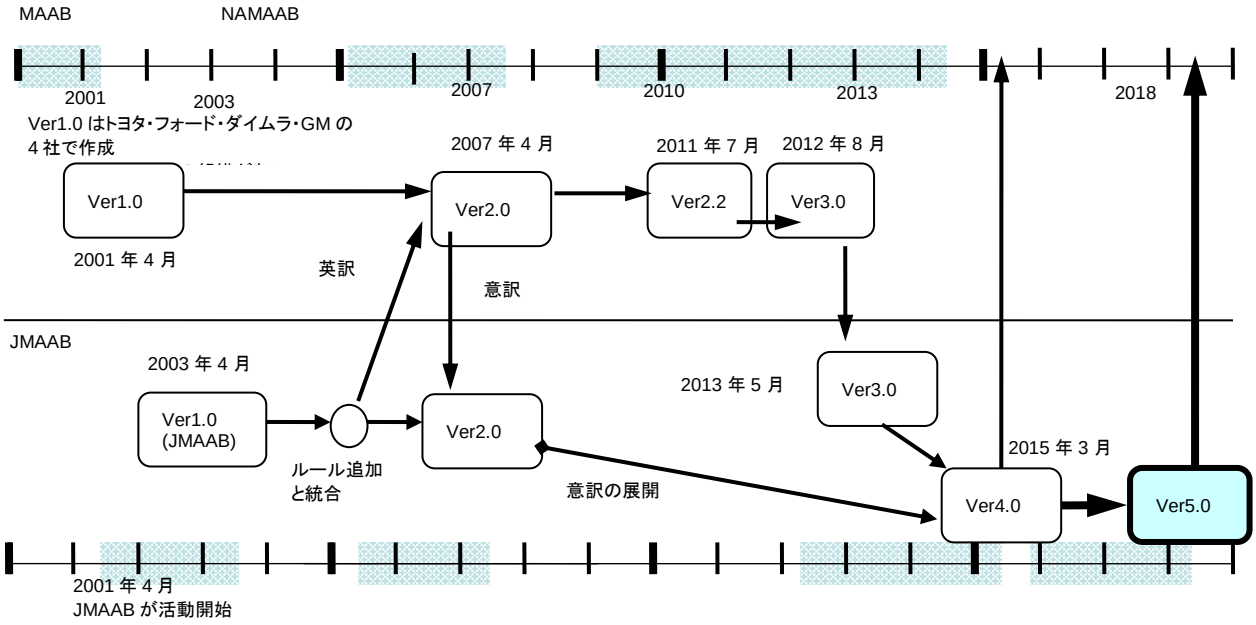
タスク内の新たな計算を行う前に一斉に値をコピーします。



10. 更新履歴

更新日	更新内容
2001.04.02	NAMAAB Initial document Release、Version 1.0(Eng)
2003.04.xx	JMAAB Initial document Release、Version 1.0(Jp)
2007.04.27	Version 2.0 Update release(Jp&Eng) この文書は、JMAAB と NAMAAB の合作です。
2011.07.30	Version 2.2 Update release(Eng)
2012.08.31	Version 3.0 Update release(Eng)
2013.05.30	Version 3.0 日本語化(Jp)
2015.03.31	Version 4.0 Update release(Jp&Eng)
2015.06.19	Version 4.01 誤字訂正(Jp&Eng)
2018.03.30	Version 5.0 Update release(Jp)
2018.08.31	Version 5.1 誤字訂正(Jp)

10.1. スタイルガイドライン改定の流れ



10.2. スタイルガイドラインとしての変更点(Ver4.0⇒5.0)

Ver4.0 から 5.0 で変更したガイドラインコンセプトを記載します。

1. ルール準拠の背景を明確化
   - サブ ID 毎に根拠を明確化
   - 関連ルールを明確化
2. ルール選択の柔軟化
   - ルール選択を柔軟化するため、重要度、範囲を廃止
   - 1 つの ID に存在する複数ルールを明確にするためのサブ ID 化
   - 複数のモデリングパターンは選択式のサブ ID(a1、a2)の付与

10.3. ルールとしての変更点(Ver4.0⇒5.0)

Ver4.0 からの変更点を、以下の内容で分類しています。

- ○：新規追加
  (ID として新規追加)

・ △：変更
(サブ ID の追加やルールの変更)
・ ×：削除
(付録等にも残らず、ガイドラインから一切削除)
・ ▽：付録移動
(ID は消えたが内容はガイドラインに残る)
・ ▼：他ガイドラインに移動
(ルール内容が MATLAB® PROGRAMMING GUIDELINES に移動されるため ID 削除)
・ →xxxx：統合
(ルール内容が別 ID(サブ ID)に統合されたため、Ver5.0 では ID 削除)
・ ←xxxx：分割
(別 ID のサブ ID に分割)
・ -：変更なし
(Ver4.0 からはサブ ID にしたのみで、ルール自体の変更はない)

Ver4.0 のルール ID	Ver5.0 のルール ID	Ver5.0 での変更内容	最終更新	備考
ar_0001	ar_0001	−	Ver4.0	
ar_0002	ar_0002	△	Ver5.0	
jc_0241	jc_0241	△	Ver5.0	
jc_0242	jc_0242	△	Ver5.0	
jc_0201	jc_0201	△	Ver5.0	
jc_0211	jc_0211	△	Ver5.0	
jc_0222	jc_0222	△	Ver5.0	
jc_0232	jc_0232	△	Ver5.0	
jc_0231	jc_0231	△	Ver5.0	
jc_0243	jc_0243	△	Ver5.0	
jc_0244	jc_0244	△	Ver5.0	
jc_0245	jc_0245	△	Ver5.0	
jc_0246	jc_0246	△	Ver5.0	
jc_0247	jc_0247	−	Ver4.0	
na_0035	−	→jc_0201 →jc_0211 →jc_0222 →jc_0232 →jc_0231	Ver5.0	
jc_0251	−	→jc_0222 →jc_0232	Ver5.0	
na_0014	−	→jc_0201 →jc_0211 →jc_0222 →jc_0232 →jc_0231	Ver5.0	
−	jc_0795	〇	Ver5.0	
−	jc_0796	〇	Ver5.0	
−	jc_0791	〇	Ver5.0	
−	jc_0792	〇	Ver5.0	
na_0006	−	×	Ver5.0	ノウハウのため削除
na_0007	−	×	Ver5.0	ノウハウのため削除
db_0143	db_0143	−	Ver4.0	
db_0144	db_0144	−	Ver2.2	

na_0004	na_0004	−	Ver2.0	
db_0043	db_0043	△	Ver5.0	
db_0042	db_0042	−	Ver2.0	
jm_0002	jm_0002	−	Ver2.0	
db_0142	db_0142	−	Ver2.0	
jc_0061	jc_0061	−	Ver4.0	
db_0140	db_0140	−	Ver4.0	
db_0032	db_0032	△	Ver5.0	
db_0141	db_0141	△	Ver5.0	
jc_0110	jc_0110	△	Ver5.0	
jc_0111	−	→jc_0110	Ver5.0	
jc_0653	jc_0653	−	Ver4.0	
jc_0171	jc_0171	△	Ver5.0	
jc_0602	jc_0602	−	Ver4.0	
db_0146	db_0146	−	Ver4.0	
jc_0281	jc_0281	△	Ver5.0	
jc_0603	jc_0603	−	Ver4.0	
jc_0604	jc_0604	△	Ver5.0	
−	jc_0806	○	Ver5.0	
na_0010	na_0010	−	Ver4.0	
na_0008	-	×	Ver5.0	JMAAB では jc_0008 採用のため削除
jc_0008	jc_0008	−	Ver4.0	
na_0009	−	×	Ver5.0	JMAAB では jc_0009 採用のため削除
jc_0009	jc_0009	−	Ver4.0	
na_0005	−	→jc_0061	Ver5.0	
jc_0082	−	→jc_0602 →jc_0061 →jc_0081	Ver5.0	
−	jc_0081	←jc_0082	Ver5.0	
jc_0083	−	×	Ver5.0	JMAAB では jc_0082 採用のため削除
db_0097	db_0097	−	Ver2.0	
db_0081	db_0081	−	Ver2.0	
na_0003	na_0003	△	Ver5.0	
na_0002	na_0002	−	Ver4.0	
jm_0001	−	×	Ver5.0	db_0143 で使用可能ブロックを定義するため削除
hd_0001	−	×	Ver5.0	db_0143 で使用可能ブロックを定義するため削除
na_0011	na_0011	−	Ver4.0	
jc_0141	jc_0141	−	Ver2.2	
jc_0121	jc_0121	△	Ver5.0	
jc_0610	jc_0610	△	Ver5.0	
jc_0611	jc_0611	△	Ver5.0	
jc_0131	jc_0131	−	Ver2.0	
jc_0161	jc_0161	△	Ver5.0	
jc_0621	jc_0621	−	Ver4.0	
jc_0011	jc_0011	−	Ver2.2	
jc_0629	−	×	Ver5.0	db_0143 で使用可能ブロックを定義するため削除

jc_0622	jc_0622	−	Ver4.0	
jc_0626	jc_0626	−	Ver4.0	
jc_0627	jc_0627	−	Ver4.0	
jc_0628	jc_0628	−	Ver4.0	
jc_0650	jc_0650	−	Ver4.0	
jc_0630	jc_0630	△	Ver5.0	
jc_0631	−	→jc_0630	Ver5.0	
jc_0632	−	→jc_0630	Ver5.0	
−	jc_0794	○	Ver5.0	
−	jc_0805	○	Ver5.0	
−	jc_0800	○	Ver5.0	
jc_0625	−	×	Ver5.0	db_0143 で使用可能ブロックを定義するため削除
jc_0640	jc_0640	△	Ver5.0	
db_0112	db_0112	−	Ver2.2	
db_0110	db_0110	△	Ver5.0	
jc_0645	jc_0645	△	Ver5.0	
jc_0641	jc_0641	△	Ver5.0	
jc_0642	jc_0642	−	Ver4.0	
jc_0643	jc_0643	−	Ver4.0	
jc_0644	jc_0644	−	Ver4.0	
db_0114	付録 1	▽	Ver5.0	パターン説明のため付録移動
db_0115	付録 2	▽	Ver5.0	パターン説明のため付録移動
db_0116	付録 3	▽	Ver5.0	パターン説明のため付録移動
db_0117	付録 4	▽	Ver5.0	パターン説明のため付録移動
na_0012	付録 12	▽	Ver5.0	
na_0028	付録 13	▽	Ver5.0	
jc_0658	付録 14	▽	Ver5.0	
jc_0623	jc_0623	−	Ver4.0	
jc_0624	jc_0624	−	Ver4.0	
jc_0651	jc_0651	△	Ver5.0	
jc_0652	−	×	Ver5.0	ルール内容形式化困難なため削除
jc_0659	jc_0659	−	Ver4.0	
jc_0656	jc_0656	−	Ver4.0	
jc_0657	jc_0657	△	Ver5.0	
db_0123	−	→jc_0602	Ver5.0	
jc_0700	jc_0700	−	Ver4.0	
db_0122	db_0122	−	Ver2.0	
db_0125	db_0125	△	Ver5.0	
jc_0701	jc_0701	−	Ver4.0	
jc_0702	jc_0702	−	Ver4.0	
jm_0011	jm_0011	−	Ver1.0	
db_0129	db_0129	△	Ver5.0	
db_0137	db_0137	−	Ver4.0	
jc_0711	jc_0711	△	Ver5.0	
jc_0531	jc_0531	△	Ver5.0	
jc_0712	jc_0712	−	Ver4.0	
na_0038	付録 8	▽	Ver5.0	ノウハウのため付録移動
na_0040	付録 9	▽	Ver5.0	ノウハウのため付録移動

jc_0720	−	×	Ver5.0	ルール内容形式化困難なため削除
jc_0721	jc_0721	−	Ver4.0	
jc_0722	jc_0722	−	Ver4.0	
jc_0723	jc_0723	−	Ver4.0	
−	jc_0797	○	Ver5.0	
jc_0730	jc_0730	△	Ver5.0	
jc_0731	jc_0731	△	Ver5.0	
jc_0732	jc_0732	△	Ver5.0	
jc_0733	jc_0733	△	Ver5.0	
jc_0734	jc_0734	−	Ver4.0	
jc_0740	jc_0740	−	Ver4.0	
jc_0501	jc_0501	△	Ver5.0	
jc_0735	−	▼	Ver5.0	
jc_0736	jc_0736	△	Ver5.0	
jc_0737	−	▼	Ver5.0	
jc_0738	−	△	Ver5.0	
jc_0739	jc_0739	−	Ver4.0	
jc_0741	jc_0741	−	Ver4.0	
jc_0742	−	▼	Ver5.0	
jc_0770	jc_0770	−	Ver4.0	
jc_0771	jc_0771	△	Ver5.0	
jc_0772	jc_0772	−	Ver4.0	
jc_0752	jc_0752	−	Ver4.0	
jc_0743	−	▼	Ver5.0	
−	jc_0790	○	Ver5.0	
−	jc_0802	○	Ver5.0	
−	jc_0803	○	Ver5.0	
jc_0750	−	→db_0129	Ver5.0	
jc_0751	jc_0751	−	Ver4.0	
jc_0754	−	→jc_0753	Ver5.0	
jc_0753	jc_0753	△	Ver5.0	
db_0151	−	▼	Ver5.0	
na_0013	−	→jc_0802	Ver5.0	
jc_0481	jc_0481	−	Ver2.0	
na_0001	na_0001	△	Ver5.0	
jc_0655	jc_0655	△	Ver5.0	
jc_0451	jc_0451	−	Ver2.0	
jc_0755	−	×	Ver5.0	ルール違反でエラーとなるため削除
jc_0756	−	▼	Ver5.0	
jc_0757	−	▼	Ver5.0	
jc_0491	jc_0491	−	Ver2.2	
jc_0521	−	▼	Ver5.0	
jc_0760	jc_0760	−	Ver4.0	
jc_0762	jc_0762	−	Ver4.0	
jc_0763	jc_0763	△	Ver5.0	
jc_0761	−	→jc_0763	Ver5.0	
db_0132	db_0132	△	Ver5.0	
db_0134	付録 5	▽	Ver5.0	パターン説明のため付録移動
db_0159	付録 6	▽	Ver5.0	パターン説明のため付録移動

db_0135	付録 7	▽	Ver5.0	パターン説明のため付録移動
jc_0773	jc_0773	−	Ver4.0	
jc_0774	jc_0774	−	Ver4.0	
jc_0511	jc_0511	−	Ver4.0	
jc_0775	jc_0775	△	Ver5.0	
jc_0776	−	→jc_0775	Ver5.0	
−	jc_0804	○	Ver5.0	
db_0126	db_0126	△	Ver5.0	
jc_0780	−	→jm_0012	Ver5.0	
jc_0781	付録 10	▽	Ver5.0	ノウハウのため付録移動
jm_0012	jm_0012	△	Ver5.0	
na_0041	付録 11	▽	Ver5.0	ルールではないため付録移動
na_0042	na_0042	−	Ver3.0	
na_0039	na_0039	−	Ver3.0	
db_0127	db_0127	−	Ver2.2	
na_0037	na_0037	−	Ver3.0	
na_0020	na_0020	−	Ver4.0	
na_0036	na_0036	−	Ver3.0	
na_0033	−	×	Ver5.0	ルール内容が形式化困難なため削除
na_0031	na_0031	−	Ver3.0	
na_0018	−	▼	Ver5.0	
na_0025	−	▼	Ver5.0	
na_0034	na_0034	△	Ver5.0	
na_0024	na_0024	△	Ver5.0	
na_0022	−	▼	Ver5.0	
na_0016	−	▼	Ver5.0	
na_0017	−	▼	Ver5.0	
na_0021	na_0021	−	Ver3.0	
−	jc_0801	○	Ver5.0	

10.4. JMAAB スタイルガイドライン編集委員

JMAAB スタイルガイドライン編集委員　会社名

Ver1.0	Ver2.0
アイシン精機	アイシン・エィ・ダブリュ
ジヤトコ	アイシン精機
デンソー	いすゞ自動車
トヨタ自動車	ジヤトコ
日産自動車	トヨタ自動車
日立製作所	日立製作所
本田技術研究所	マツダ
マツダ	三菱電機
サポート：サイバネットシステム	サポート：サイバネットシステム

Ver4.0	Ver5.0
アイシン・エィ・ダブリュ	アイシン・エィ・ダブリュ
アイシン精機	アイシン・コムクルーズ

いすゞ自動車	アイシン精機
オムロンオートモーティブエレクトロニクス	いすゞ自動車
カルソニックカンセイ	オムロンオートモーティブエレクトロニクス
スズキ	ケーヒン
ダイハツ工業	小松製作所
デンソー	スズキ
トヨタ自動車	ダイハツ工業
日産自動車	デンソー
ミツバ	デンソーテン
三菱電機	トヨタ自動車
サポート：マスワークスジャパン	日産自動車
	日立オートモティブシステムズ
	マツダ
	三菱自動車工業
	三菱電機
	両毛システムズ
	サポート：マスワークスジャパン

株式会社省略、参加メンバー五十音順で掲載

11. 付録

11.1. Simulink / Stateflow 機能の解説

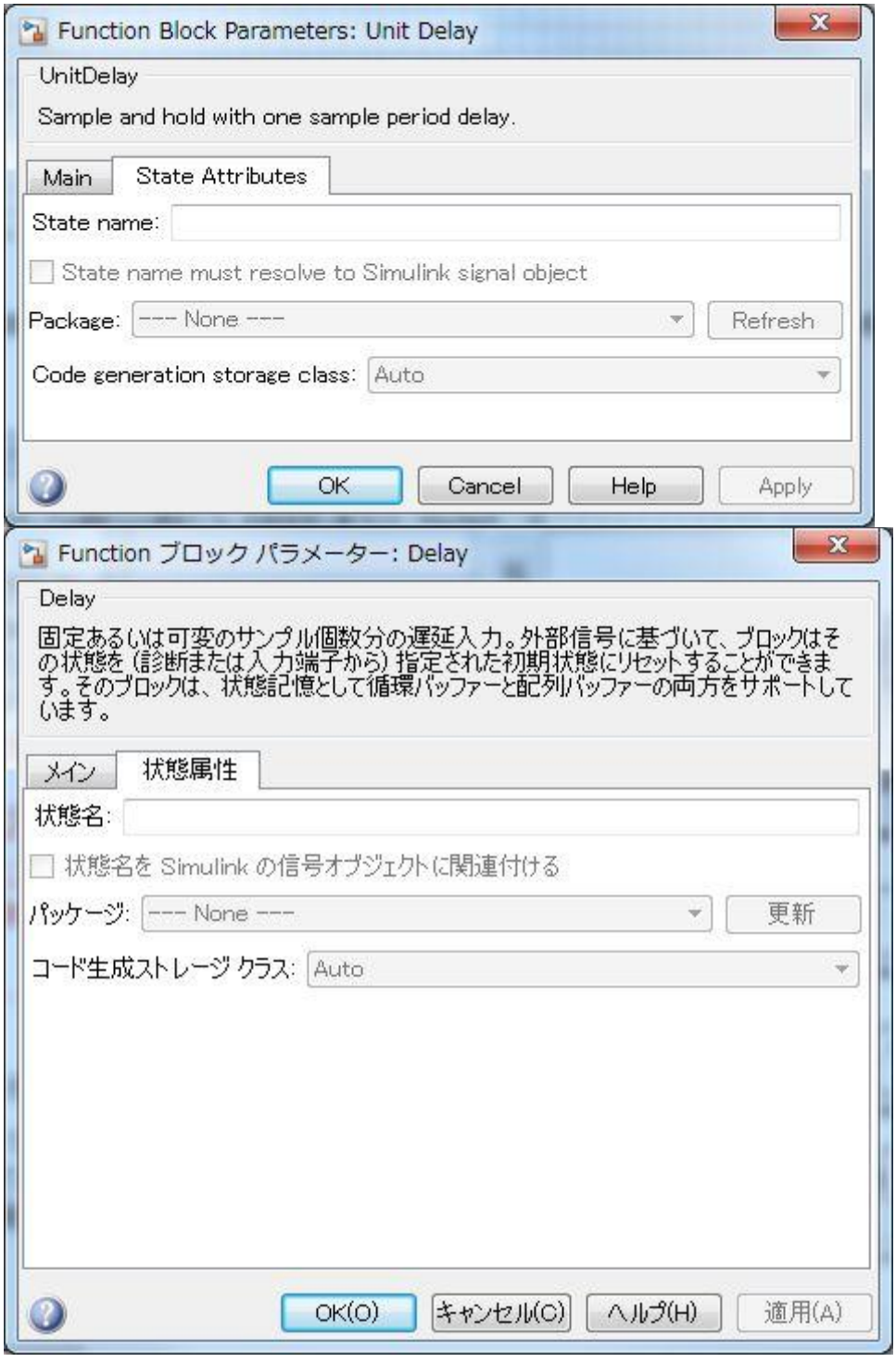
11.1.1. Simulink 機能の解説

11.1.1.1. 状態変数を持つブロック

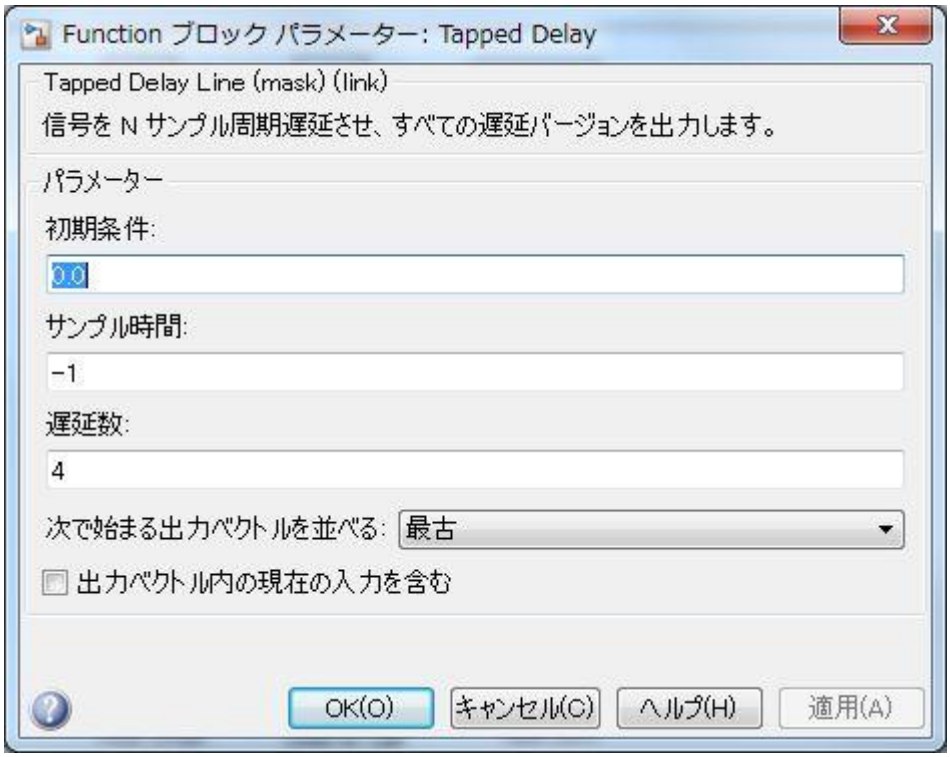
状態変数を持つブロックは、主に<simulink><Discrete>にまとめられています。

これらのブロックのほとんどは、ブロックのパラメーターにより、状態属性や初期値の設定を行う事ができます。

状態属性のプロパティを持つブロックの例



例外として、状態属性を持たないブロックとして TappedDelay 等があります。



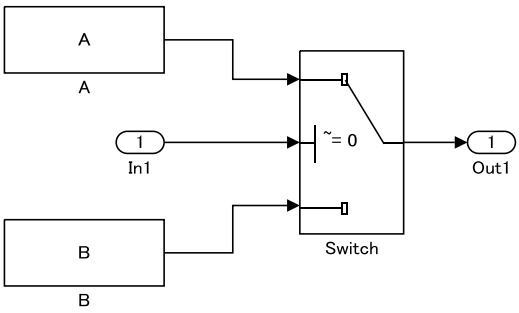
条件付きサブシステムを用いた場合に、構造のパターンによって状態変数を持つ場合があります。

関連 ID：jc_0640

11.1.1.2. 状態変数が関係する分岐構文

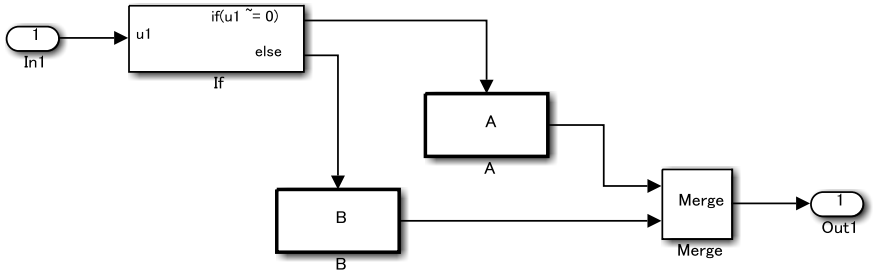
状態変数が関係する場合の Switch と条件付きサブシステムの違いについて説明します。

まず、コンフィギュレーションの設定にもよりますが、Switch ブロックは、状態変数が無い場合、制御端子の条件成立でサブシステム A が実行され、条件不成立でサブシステム A は計算せず、サブシステム B だけが計算されます。



しかし、サブシステム A の中に状態変数が存在する場合サブシステム A の状態変数を計算する為に必要な計算は制御端子が条件不成立時も計算されます。

条件付きサブシステムを使用した場合、サブシステム A に状態変数が存在するしないに関係なく、条件成立で A が計算、条件不成立でサブシステム A は計算せず、サブシステム B が計算されます。



また、Action Port の設定で、再計算時にリセットするのか、しないのかが選択できます。

以下に一覧表にその動作を記載します。

サブシステム A の挙動

制御端子の条件	(サブシステムA内)状態変数	Switch	条件付きサブシステム
成立	なし	実行される	実行される
	あり		
不成立	なし	実行されない	実行されない
	あり	最小限実行 ※状態変数に関係する計算のみ実行	

サブシステム A の初期化タイミング

	ActionPort	初期化
Switch	ー	初回のみ
条件付きサブシステム	保持	初回のみ
	リセット	条件復帰時

Switch ブロックを使うのか、条件付きサブシステムを使うのかは、これらの動作の特性を熟知し、用途に応じて、適した構造を採用して下さい。

関連 ID:jc_0656、jc_0657

11.1.1.3. サブシステムの使い方

サブシステムとは、複数のブロックやサブシステムをまとめる為に使用します。
しかし、それ以外にも使用することがあります。下記にブロック等をまとめる以外の使い方を示します。

- サブシステムのマスク表示を利用します。
  - 概要を記載したり、「機密」 などの定型文書を表示したりします。
- サブシステムのオープンファンクション(ブロックプロパティ内コールバック関数)を利用します。
  - 様々なツールを起動させたり、モデルとは別の説明文書を表示したりするなどの目的で使用します。
- 組織が認定した上位レベルのユーザーが設計・検討後にマスク化し、一般ユーザーが中を見られない設定に変更したマスクサブシステム(単に NoReadOrWrite に設定されたサブシステム)

上記のような一般的でないサブシステムはガイドラインの検査対象から除外してもよく、除外対象としてリストを作成し、プロジェクトで管理すべきです。

関連 ID:jc_0201、jc_0243、db_0143、db_0144、db_0141、jc_0653、jc_0171、jc_0602、jc_0081、db_0081、他

11.1.1.4. 信号名

信号には名前をつけることができ、それを信号名と呼びます。信号に名前をつけた場合はその信号名がラベルとして表示され、ラベルを編集した場合は信号名に反映されるので、結果的に同じ意味を示します。

入力した信号名は分岐した信号線やポートブロックなどを経由した信号線に伝播し、信号名として表示できます。

関連 ID:jc_0222、jc_0245

また、信号名に信号オブジェクト(Simulink オブジェクトあるいは mpt オブジェクト)を関連付けすることで、コード生成が可能になります。型設定はデータディクショナリで行い、ストレージクラスの設定は任意です。尚、以下のブロックの型設定は推奨となります。

- Inport ブロック・・・データ型”auto”
- Outport ブロック・・・データ型”auto”
- Sum ブロック・・・出力データ型” Inherit via back propagation”

関連 ID:jc_0644

11.1.1.5. ベクトル信号 / バス信号の解説

ベクトル

ベクトルを構成する個々のスカラー信号は、共通する機能、データ型および単位をもっていなければなりません。

バス

前述のような、ベクトルとしての条件を満たさない信号は、バス信号にしかグループ化できません。

[Bus Selector]は、バス信号入力でのみ使用し、ベクトル信号からスカラー信号を抽出するためには使用しないでください。

例

次に、ベクトル信号の例を示します。

ベクトルのタイプ	サイズ
行ベクトル	[1 n]
列ベクトル	[n 1]
ホイール速度サブシステム	[1 ホイール数]
シリンダー ベクトル	[1 シリンダー数]
2 次元座標に基づく位置ベクトル	[1 2]
3 次元座標に基づく位置ベクトル	[1 3]

次に、バス信号の例を示します。

バスのタイプ	要素
センサー バス	カベクトル [Fx, Fy, Fz]
	位置
	ホイール速度ベクトル [Θ_{fl}, Θ_{rl}, Θ_{fr}, Θ_{rr}]
	加速
	圧力
コントローラ バス	センサー バス
	アクチュエータ バス
シリアル データ バス	冷却水温度
	エンジン速度、助手席ドア開

関連 ID：na_0010、jc_0222、jc_0245、db_0097、jc_0630、jc_0659

11.1.1.6. 列挙型について

"列挙型データ" は決まった数の値に制約されるデータです。
Simulink で列挙型が使用出来るブロックの種類は限定されています。
ヘルプ Simulink－モデル－モデルの構成－データ型「列挙型をサポートする Simulink の構成」に指定可能なブロック種別の記載があります。
列挙型を使用するためには、下記例のように、MATLAB 上に m ファイルを用いて列挙型を定義しておく必要があります。

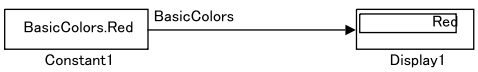
例：BasicColors.m

```
classdef(Enumeration) BasicColors < Simulink.IntEnumType
 enumeration
   Red(0)
   Yellow(1)
   Blue(2)
   Green(2)
 end

 methods (Static = true)
   function retVal = getDefaultValue()
     retVal = BasicColors. Blue;
   end

   function retVal = getDescription()
     retVal = 'This defines an enumerated type for colors';
   end

   % function retVal = getHeaderFile()
   %  retVal = 'imported_enum_type.h';
   % end

   function retVal = addClassNameToEnumNames()
     retVal = true;
       % true とするとコード上で BasicColors_Red で記載されます。
       % 指定しないあるいは false を選択すると、コード上で Red と記載されます。
     end
 end
end
```

上の例では、Red、Yellow と Blue(Green)の文字を使用可能です。また、下表に示すデータ型をカスタマイズする方法を提供します。

getDefaultValue	許容値リストの最初の値以外の既定の列挙値を指定します。
getDescription	Simulink® Coder™で 生成したコードのデータ型の説明を提供します。
getHeaderFile	Simulink Coder で生成したコードの列挙型定義を含むカスタム ヘッダーファイルのインポートを可能にします。
addClassNameToEnumNames	Simulink Coder で 生成したコードの識別子との名前の競合を回避し、読みやすくします。

Display ブロックを用いた場合に、通常の定数は 0、1、2 の表示を行いますが、列挙型を用いれば文字を表示します。



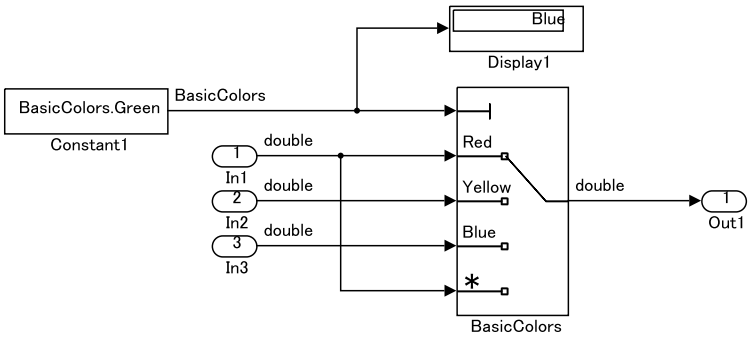
Simulink Coder は列挙型を用いてもコード生成可能です。
既定の設定では、コード内の列挙型データはモデル用に生成されたヘッダーファイル model_types.h 内で定義されます。
例えば、BasicColors 用の既定のコードは以下のようになります。

```
#ifndef _DEFINED_TYPEDEF_FOR_BasicColors_
#define _DEFINED_TYPEDEF_FOR_BasicColors_

typedef enum {
 BasicColors_Red = 0,
 BasicColors_Yellow = 1,
 BasicColors_Blue = 2,     /* Default value */
 BasicColors_Green = 2
} BasicColors;             /* This defines an enumerated type for colors */
#endif
```

信号機の色は、青と緑の 2 色のどちらも使われます。列挙型は、例題のように、同じ 2 という数値に Blue、Green の 2 つの文字を設定可能です。

例題のケースを Simulink で使用した場合、Green を設定すると Blue と解釈します。

このように 1 つの定数に 2 つの文字を設定した場合、初期値の設定同様、m ファイル上で上に書かれた文字が優先されます。



11.1.2. Stateflow 機能の解説

11.1.2.1. Stateflow で使える演算子

Stateflow で使用できる演算子

演算子	説明
a * b	乗算
a / b	除算(条件付使用可能)
a %% b	残余
a + b	加算
a - b	減算
a >> b	オペランド a を右に b ビット移動します。
a << b	オペランド a を左に b ビット移動します。
a > b	1 番目のオペランドが 2 番目のオペランドよりも大きいかを比較
a < b	1 番目のオペランドが 2 番目のオペランドよりも小さいかを比較
a >= b	1 番目のオペランドが 2 番目のオペランド以上であるかを比較
a <= b	1 番目のオペランドが 2 番目のオペランド以下であるかを比較
a == b	2 つのオペランドが等しいかを比較
a ~= b	2 つのオペランドが等しくないかを比較
a != b	2 つのオペランドが等しくないかを比較
a <> b	2 つのオペランドが等しくないかを比較

Stateflow のアクション言語	MATLAB	C	
C 言語のビット演算が可能	-	OFF	ON
論理 OR	|、||	|、||	||

論理 AND	&、&&	&、&&	&&
ビット単位 OR	bitor()	不可能	|
ビット単位 AND	bitand()	不可能	&
ビット単位 XOR	bitxor()	不可能	^
べき乗	^	^	不可能
論理否定	~	!、~	!
1 の補数	bitcmp()	不可能	~

C チャートでは、以下の単項アクションがサポートされています。

演算子	説明
a++	インクリメント
a--	デクリメント

(MATLAB チャートでは、a++と記載すると自動的に a=a+1 に変換されます。)

代入演算ベクトルと行列のオペランドに対し、要素単位の代入演算を実行することができます。

代入演算	同等記述
a = expression	
a += expression	a= a + expression
a -= expression	a= a - expression
a *= expression	a= a * expression
a /= expression	a= a / expression

関連 ID: na_0001、jc_0655

11.1.2.2. フローチャートと状態遷移の違い

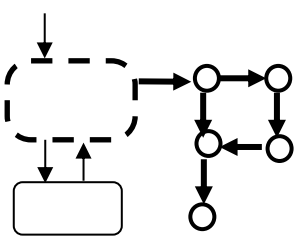
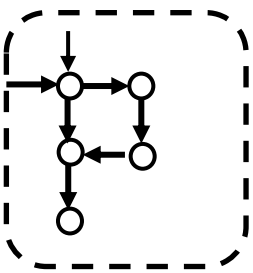
Stateflow は、状態遷移と、フローチャート 2 つの機能を表現できます。
Stateflow では状態遷移の中にもフローチャートを設計することが可能です。
例えば、図例のようにステート内部にデフォルト遷移線からコネクティブジャンクションと遷移線を用いて描けば、entry アクションをフローチャートで表現できます。内部遷移線から始めれば、during アクションをフローチャートで表現できます。

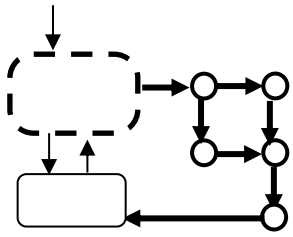
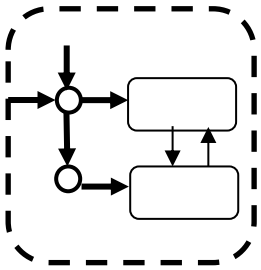
補足説明:
フローチャートは更新と更新の間のアクティブステートを維持しません。そのため、フローチャートの実行は常に“終端ジャンクション”(有効な出力遷移をもたないコネクティブジャンクション)で終わります。
一方、状態遷移はメモリに現在のステートを保存して、更新と更新の間でローカルデータとアクティブステートを維持します。つまり、前のタイムステップで終了したところから実行を開始できます。このため、状態遷移は前回結果に依存して処理を行うリアクティブシステムや管理システムのモデリングに適しています。

フローチャートと状態遷移の分類表

	始点	終点
フローチャート	デフォルト遷移 または、ステートから	全ての終端がコネクティブジャンクションに接続される。
状態遷移	デフォルト遷移 または、ステートから	いずれかの終端がステートに接続される

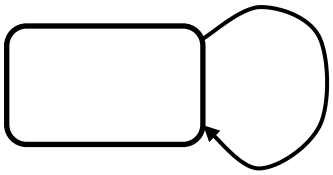
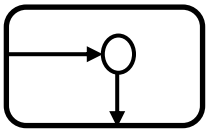
一般的なフローチャートと状態遷移の区別事例

	ステート外フローチャート	ステート内フローチャート
フローチャート		

	ステート外状態遷移	ステート内状態遷移
状態遷移		

自己遷移を持つ、フローチャート、状態遷移の混在は、両方の厳しい側の制約を受ける。

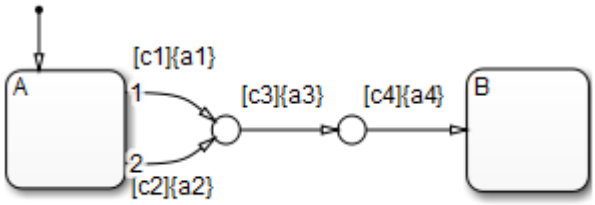
フローチャート＋自己遷移の例

	ステート外自己遷移 ・ステートの外側で自己遷移を形成、実施後にリセットされる	ステート内自己遷移 ・ステートの中で自己遷移を形成、during アクション＋リセット
状態遷移		

関連 ID:db_0132、jc_0752

11.1.2.3. バックトラック

以下の例では、フローチャートでのバックトラッキング動作を強制するコネクティブジャンクションを経由した遷移の動作を説明します。チャートでは、出力遷移の暗黙的な順序付けを使用します (ヘルプの「出力遷移の評価順序」-「出力遷移の暗黙的な順序付け」を参照)。

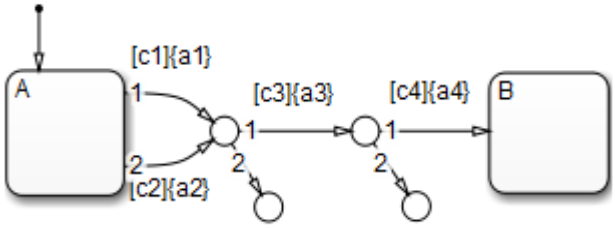


最初はステート A がアクティブであり、遷移条件 c1、c2、および c3 は真、c4 は偽です。

1. チャートのルートは、ステート A から有効な遷移が存在しないかどうかをチェックします。
   ステート A からコネクティブジャンクションに、遷移条件 c1 としてマークされた有効な遷移セグメントが存在します。
2. 遷移条件 c1 は真であるため、アクション a1 が実行されます。
3. 遷移条件 c3 は真であるため、アクション a3 が実行されます。
4. 遷移条件 c4 は真でないため、制御フローはステート A に戻ります。
5. チャートのルートは、ステート A から他に有効な遷移が存在しないかどうかをチェックします。
   ステート A からコネクティブジャンクションに、遷移条件 c2 としてマークされた有効な遷移セグメントが存在します。
6. 遷移条件 c2 は真であるため、アクション a2 が実行されます。
7. 遷移条件 c3 は真であるため、アクション a3 が実行されます。
8. 遷移条件 c4 は真でないため、制御フローはステート A に戻ります。
9. チャートはスリープします。

この問題を解決するには、無条件遷移線を終端ジャンクションに追加することを検討してください。

終端ジャンクションを使用すると、c3 または c4 のどちらかが真でない場合にフローを終了できます。これにより、ステート A は不要なアクションを伴わずにアクティブなままです。



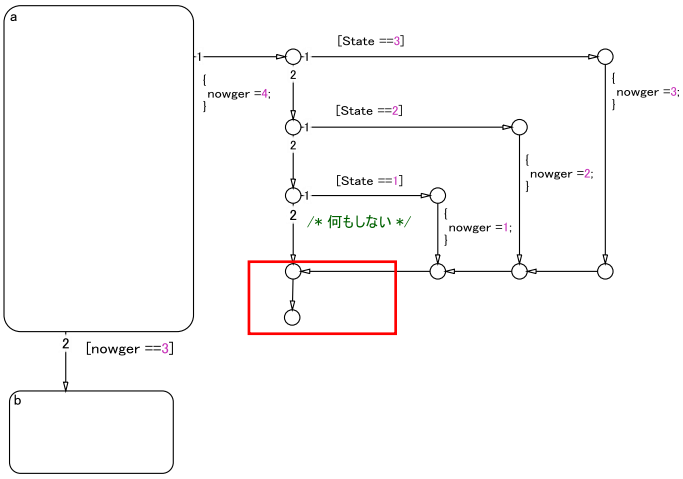
関連 ID:jc_0751、jc_0773

11.1.2.4. 状態に付随したフローチャート作成時の注意点

ステートに付随したフローチャートは、ステートの中にも外にも記載ができますが、その実行順序やバックトラックに注意した記述が必要です。

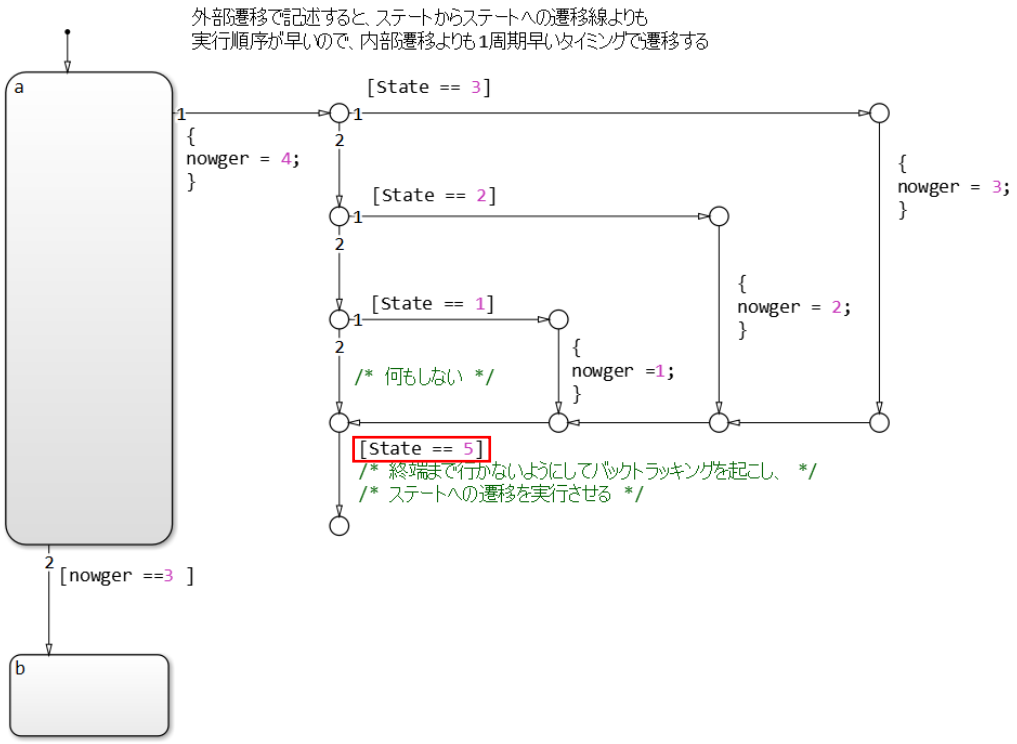
下の図では、ステート外フローチャートを実施後に、a から b の遷移を評価するので、nowger の計算と同一周期で遷移が実行されるように見えます。

しかし、実はステート外遷移を計算し、終端まで到達すると b への遷移線は評価されません。これは、常に a にとどまる状態遷移図です。



正しくは、下の線のように、ステート外フローチャートの終端にわざと成立しない遷移条件を埋め込み、ステート外フローチャート実行後に a から b への遷移線を評価するように記述します。

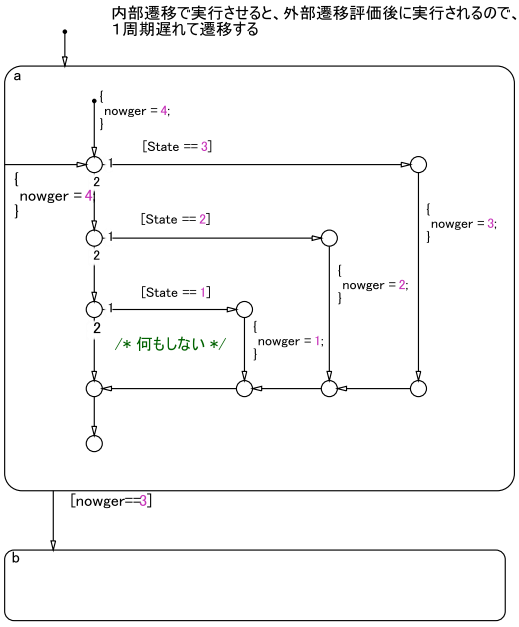
これで、ステート外フローチャートが遷移の前に実行でき、遷移の瞬間に最新の値を用いて遷移を評価する事ができるようになります。この場合、絶対に遷移条件が成立しないデッドパスができるので、後の仕様変更時にバグの要因となる可能性があります。十分な注意が必要です。



それに対して、下記のようなステート内フローチャートを用いると、ステート a を実行する場合、必ず内部フローチャートが計算され、デッドパスの無い理解しやすい構造で記述できます。

ただし、動作の特性として、ステート a の実行時に、内部フローチャートを計算した次の周期で、a から b への遷移が評価されます。

外部フローチャートに対して、計算の実行と、遷移のタイミングがずれる場合があります。十分な注意が必要です。



関連 ID:jc_0751、jc_0773

11.1.2.5. ポインタ変数

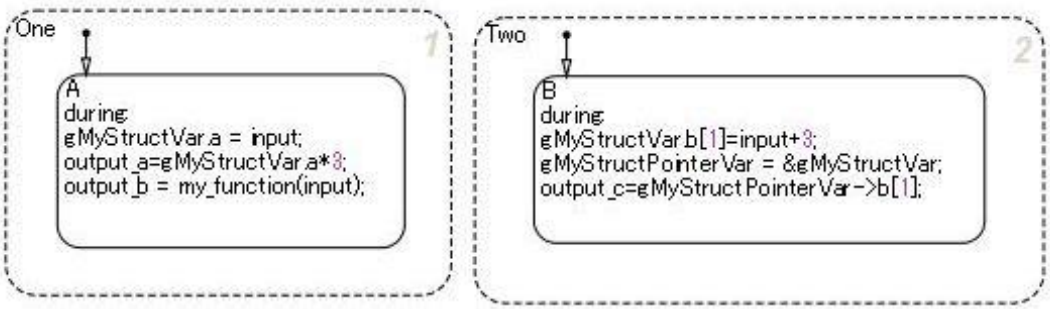
デモモデル sf_custom を参考に説明します。

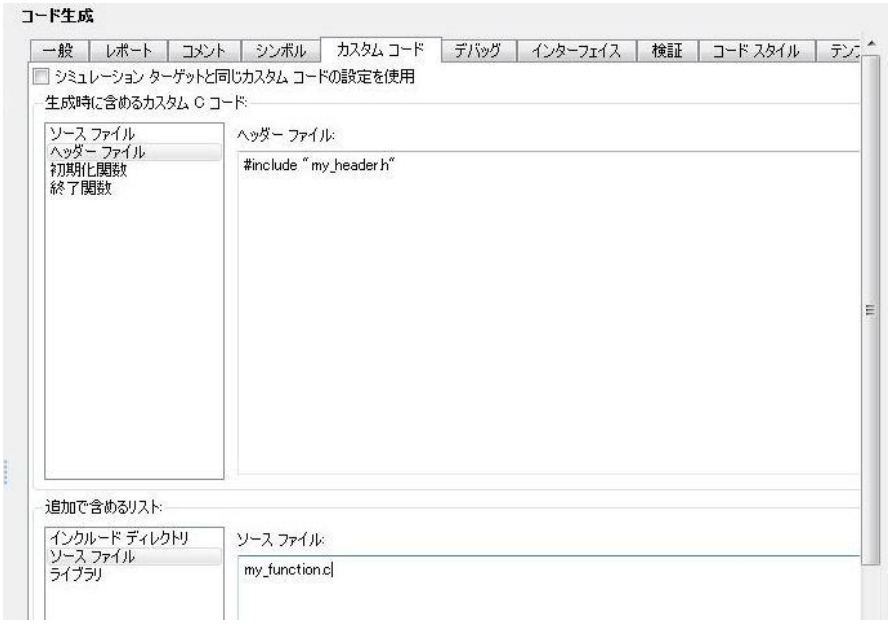
解説

Stateflow では、gMyStructVar は定義されていません。
コンフィギュレーションのコード生成ペイン内に、C ソースファイルの読み込み設定を行っています。
通常は、C ソースからは、my_function の関数を呼び出して、Stateflow 内部で使用します。
しかし、参照した C ソースが公開しているグローバル変数は Stateflow からも直接参照可能です。

```
---------my_header.h--------------
#include "tmwtypes.h"

extern real_T my_function(real_T x);

/* Definition of custom type */
typedef struct {
 real_T a;
 int8_T b[10];
}MyStruct;

/* External declaration of a global struct variable */
extern MyStruct gMyStructVar;
extern MyStruct *gMyStructPointerVar;

---------------my_function.c--------------
#include "my_header.h"
#include <stdio.h>

/* Definition of global struct var */
MyStruct gMyStructVar;
MyStruct *gMyStructPointerVar=NULL;

real_T my_function(real_T x)
{
 real_T y;

 y=2*x;

 return(y);
}
```

----------Stateflow 内部----------





11.1.3. 初期化

11.1.3.1. 初期化における初期値設定

信号の初期化が必要な場合、初期値を設定します。
ブロック内部に、初期値を設定した場合、入力した初期値が一目で確認できるよう、ブロック注釈によって初期値を表示させると効果的です。
以下に初期値が必要とされるケースをまとめます。

1. 状態変数が定義される場合
   ① 状態変数を持つブロックを使用した場合
      A) ブロック内部の設定を利用する。
      B) 外部入力値を利用する。
   ② 特定の構成を行った場合にブロックの初期値が有効となる場合
      A) Merge ブロックに初期値を設定する。
      B) データディクショナリに登録した信号を用いる。
2. 外部から参照できる(RAM を持つ)信号設定を定義した場合
      A) データディクショナリに登録した信号を用いる。

これ以降で、上記のケースについて、モデリングの参考情報を記載します。

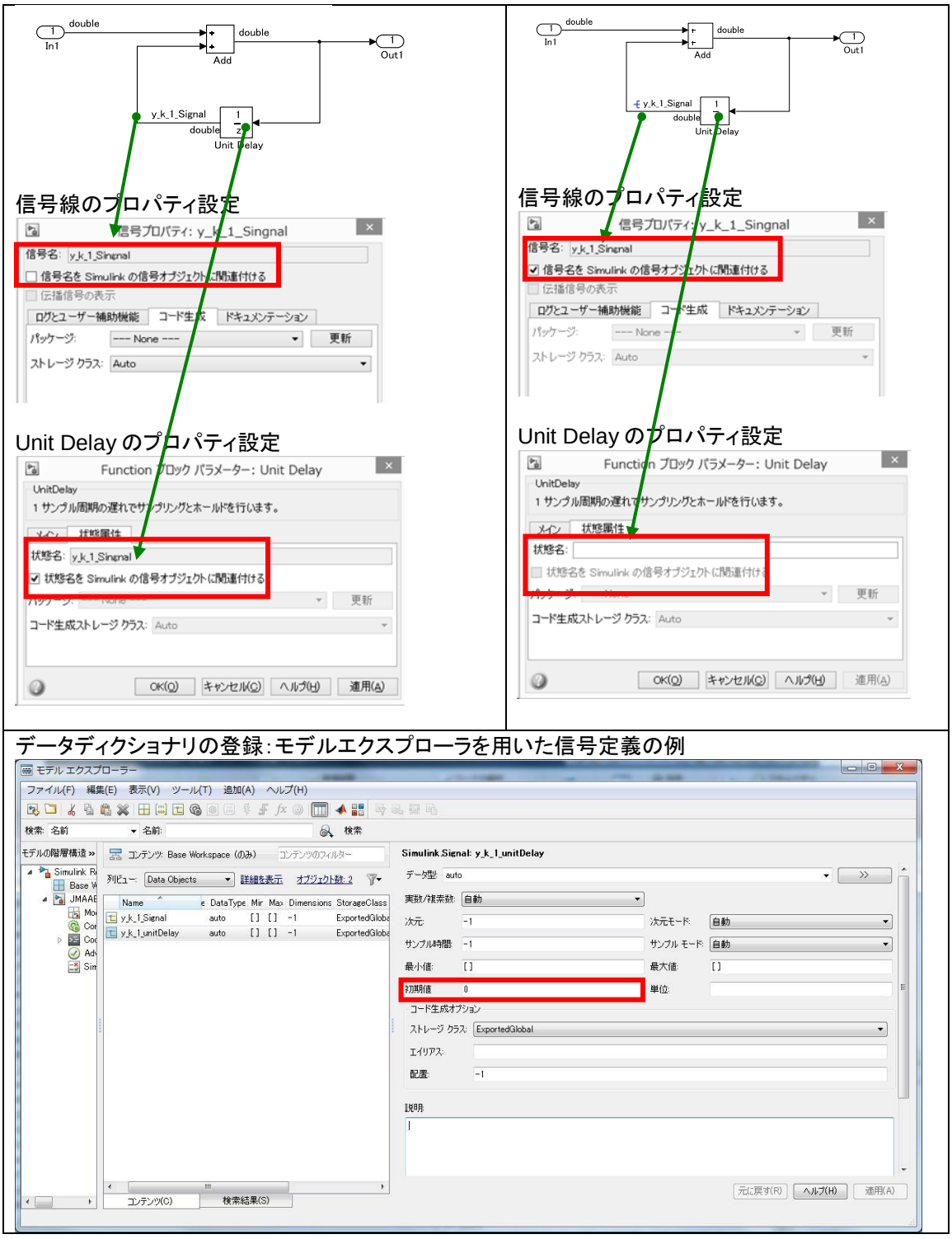
11.1.3.2. データディクショナリに登録した信号の初期値

データディクショナリに登録した信号は、初期値を設定します。

・UnitDelay 等の Discrete ブロック群と Data Store Memory は、状態変数を持ちます。
自動コード生成を行う場合、状態変数は、データディクショナリ上の信号に対応させる(Simulink の信号オブジェクトに関連付ける)ことで信号名や型、初期値を設定することができます。状態変数にデータディクショナリで定義された信号を用いる場合、それぞれの初期値を同じ値に一致させる、または、ブロックの初期値を空に指定します。

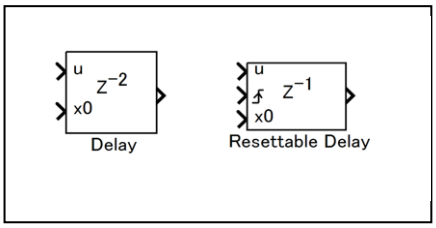
・状態変数にデータディクショナリで定義された信号を使用する場合
UnitDelay 等の Discrete ブロック群と Data Store Memory は、ブロックの出力線ではなく、ブロック内部の状態変数にデータディクショナリで定義した信号を使用する設定を行います。信号線にデータディクショナリの信号名を割り付けても 2 重に RAM が確保されることになり、RAM の無駄となります。

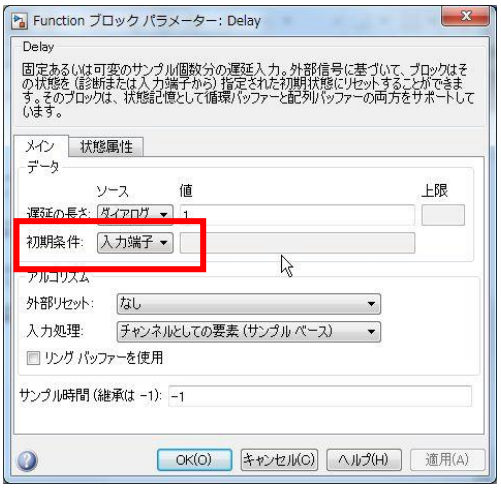
【正】ブロック内部の状態変数に信号定義をした場合	【誤】状態変数を持つブロックの出力信号線に信号定義を行った場合



ワークスペースに定義された信号オブジェクトは、disableimplicitsignalresolution(モデル名)のコマンドを用いる事で、自動的にモデル内の同名の信号情報と関連付けが可能です。よって、ブロック内部の状態変数と、ブロックからの出力信号線にどちらも同じ名称を設定している場合、両方に対して信号オブジェクトへの関連付けがされてしまいます。モデルの複数個所にグローバルな設定がされた信号を関連付けするとモデルが実行不可能となりますので、ブロック内部の状態変数と信号線の信号ラベル名は、異なる名称の設定が推奨されます。

11.1.3.3. 外部入力値を初期値とするブロックの例



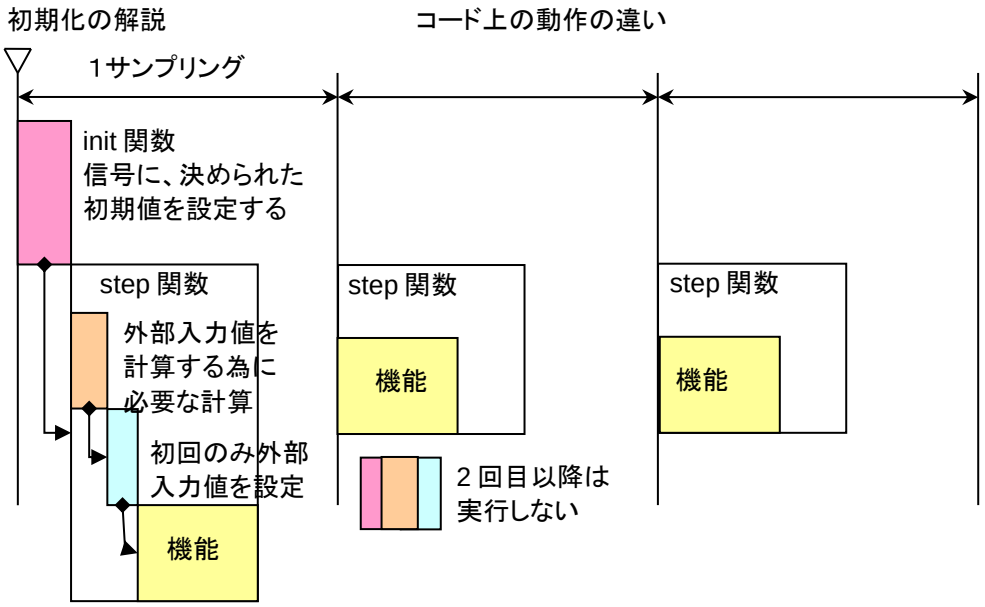


・初期化の動作

始めに、ブロック内部に設定した値あるいは、データディクショナリで定義された初期値を信号に設定する為、init 関数が呼び出されます。

次にデータフローの実行関数である step 関数が実行されます。ここで、外部入力値が初期値として設定されます。

初期化に関する実行関数・実行タイミングに留意してモデリングをしてください。



11.1.3.4. 初期化パラメータが有効となるシステム構成の初期値設定

条件付サブシステムと Merge の組み合わせは、設定によって、初期化パラメーターが有効となるシステム構成が存在します。この条件付サブシステムと Merge の組み合わせで、初期値を必要とする場合は、下記のようなモデリングを実施します。

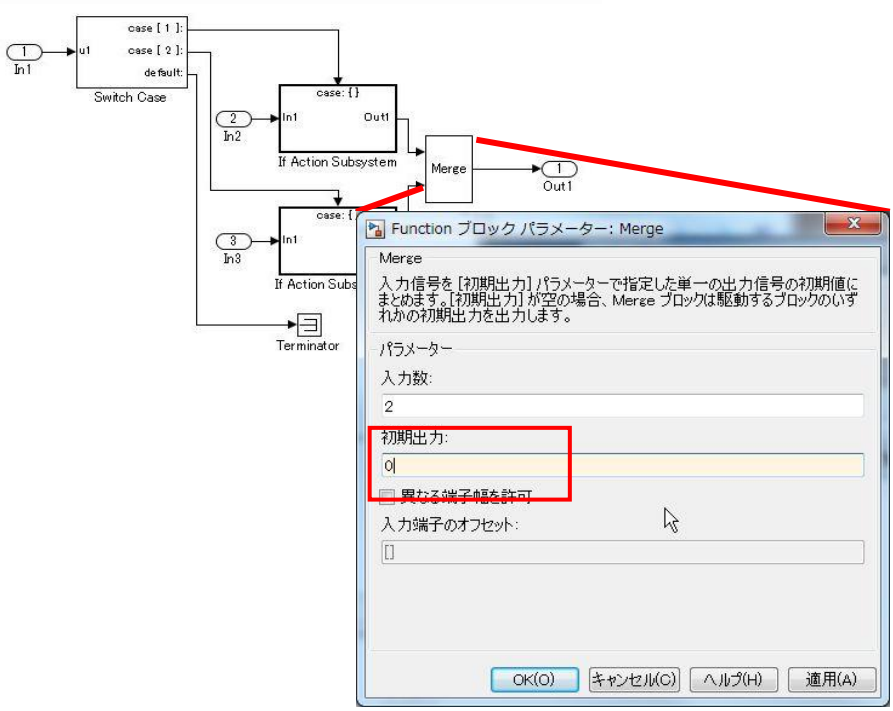
例えば、条件付サブシステムの Outport ＋ Merge の場合は以下のいずれかの手法が使用可能です。

・Outport で設定
・Merge で設定
・Merge の後ろに mpt シグナルが定義されていれば mpt シグナルで設定

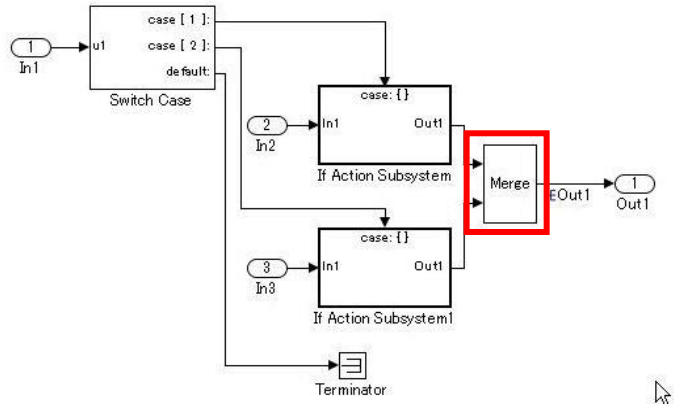
例外：

初期値を持つブロックが連続しており、信号の初期値を明示するためにブロック毎に設定することが不要である場合

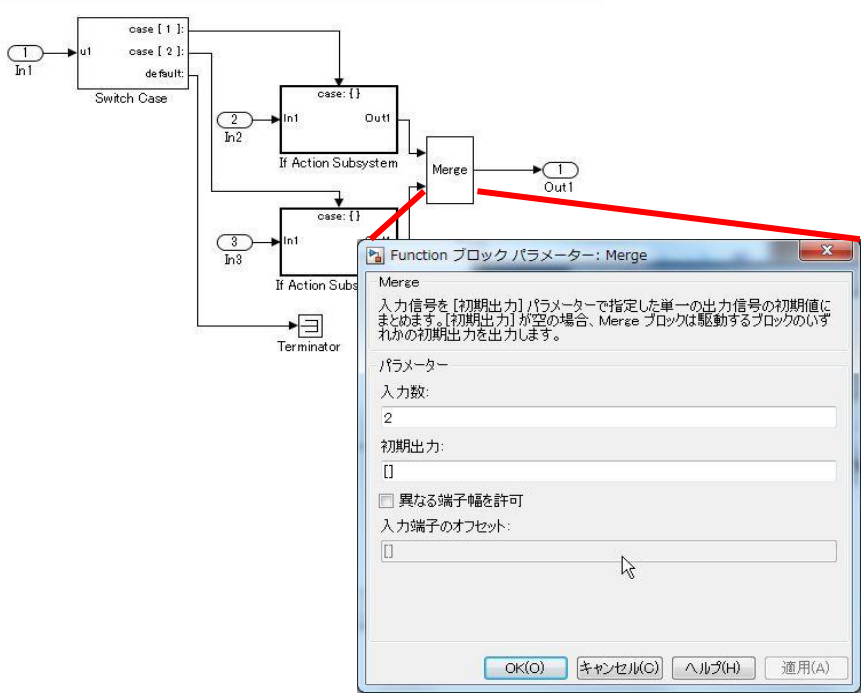
【正】Merge で初期値を設定した場合



【正】mpt オブジェクトで初期値を設定した場合



【誤】初期値の設定が必要にも関わらず、どこにも明示されていない。



11.1.4. その他

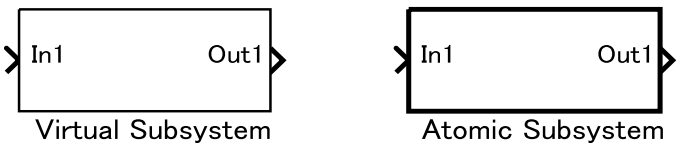
11.1.4.1. Atomic サブシステムとバーチャルサブシステムの違い

サブシステムは、Atomic サブシステムと呼ばれる設定と、バーチャルと呼ばれる 2 種類の設定があります。バーチャル サブシステム(デフォルト設定の サブシステム ブロック)と Atomic サブシステムの違いは、一つのサブシステムを一つの実行単位として扱うか扱わないかの違いです。

Atomic サブシステムは、外部とのシステムから切り離され、境界をまたいだ最適化の対象外となります。

一方、単に視覚的な表現のみを与えるブロックを バーチャル ブロックと呼びます。例えば、複数の信号線を 1 本にまとめる Mux ブロックや、信号の受渡しを行う From ブロック、Goto ブロックが バーチャル ブロックに相当します。デフォルト設定の サブシステム ブロックも、単に視覚的な階層構造を構成しているだけですので、バーチャル ブロックに属します。このサブシステムの事を バーチャル サブシステムと呼びます。

バーチャルサブシステムは外枠の線が細く、Atomic サブシステムは太く表示されます。

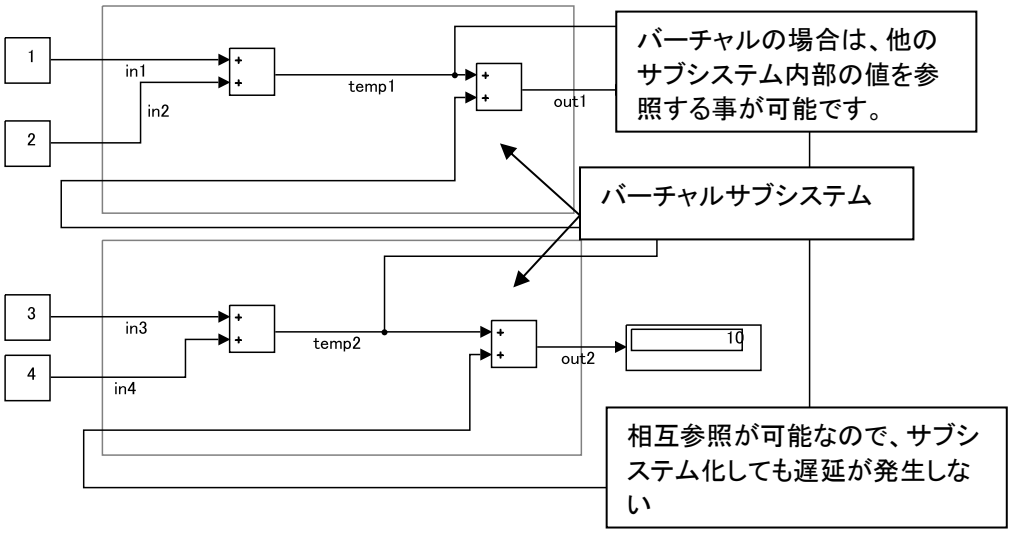


バーチャルサブシステムと Atomic サブシステムの違いを例で説明します。

バーチャルサブシステムの例

下の例題のように、サブシステム内に外部の計算結果を参照するシステムを想定します。このシステムは、以下の4つの数式から結果が計算されます。

```
temp1= in1 + in2
temp2= in3 + in4
out1= in1 + in2 + temp2
out2= temp1 + in3 + in4
```

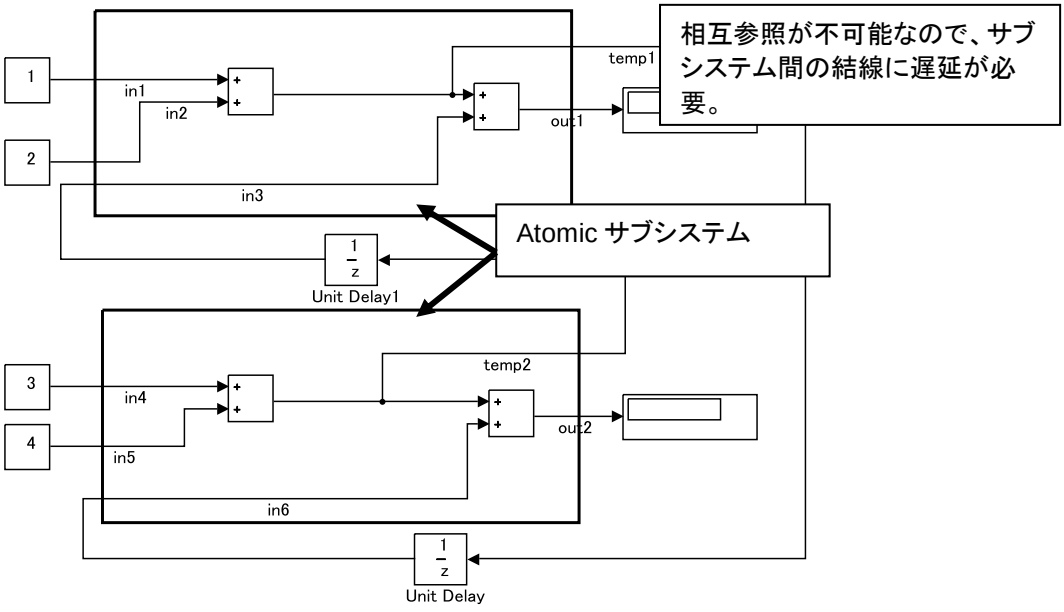


Atomic サブシステムの例

Atomic サブシステムでは、それぞれのサブシステムの内部計算結果を使用しません。その為、中間出力値は、1回遅れた計算結果を使用することになります。

```
temp1= in1 + in2
temp2= in4 + in5
out1= in1+ in2 + in3
out2= in4+ in5 + in6
in3= temp2
in6= temp1
```

Atomic サブシステムは、他のサブシステムへ中間の計算結果を直接参照することを禁止します。



Atomic サブシステムの注意点

Atomic サブシステム は、C ソースの関数設定を選択できます。

Atomic サブシステムは、上記で説明したように、サブシステムの内部がカプセル化(オブジェクト化)されます。前後関係によっては入力信号、出力信号に静的な RAM 領域が確保されることも認識すべきです。

単にテストをやりやすくする為などの理由で不用意に Atomic サブシステム(関数設定の追加を含む)を使用すべきではありません。関数設定を行う影響は、単に C コード上に関数名が挿入されることではありません。数学的に独立したシステムとして記述されることを認識し、どのようなケースで Atomic サブシステムを使用するか検討が必要です。

後述する構造レイヤの関係を含め、プロジェクト毎で運用ルールを決定し、ガイドラインルールとの関係を決める必要があります。

参考

Atomic サブシステムとバーチャルサブシステムの違い(日本語)
http://www.mathworks.co.jp/support/solutions/ja/data/1-CYPFSL/index.html?product=SL&solution=1-CYPFSL

Atomic サブシステム(日本語)
http://www.mathworks.co.jp/jp/help/simulink/slref/Subsystem.html

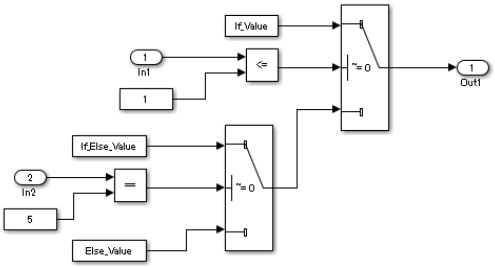
代数ループの説明(日本語)
http://www.mathworks.co.jp/jp/help/simulink/ug/simulating-dynamic-systems.html#f7-19688

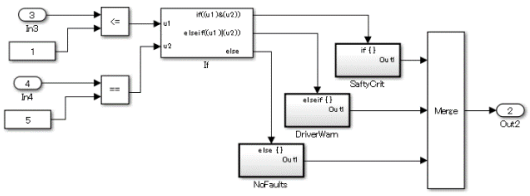
11.2. モデリングノウハウ

11.2.1. 付録 1：if-then-else-if 構文の Simulink パターン

if-then-else-if 構文は、以下のパターンを使用できます。

機能	Simulink パターン

Switch ブロックを使用した
if-then-else-if 構文
 	
`if (If_Condition)`
`{`
`  output_signal = If_Value;`
`}`
`else if (Else_If_Condition)`
`{`
`  output_signal = Else_If_Value;`
`}`
`else`
`{`
`  output_signal = Else_Value;`
`}`
	

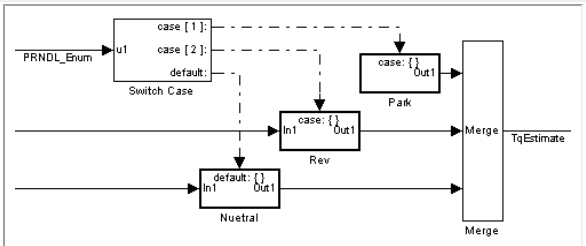
If Action Subsystem を
使用した if-then-else-if 構文
 	
`if (Fault_1_Active & Fault_2_Active)`
`{`
`  ErrMsg = SaftyCrit;`
`}`
`else if (Fault_1_Active | Fault_2_Active)`
`{`
`  ErrMsg = DriverWarn;`
`}`
`else`
`{`
`  ErrMsg = NoFaults;`
`}`
	

11.2.2. 付録 2：case 構文の Simulink パターン

case 構文は、以下のパターンを使用できます。

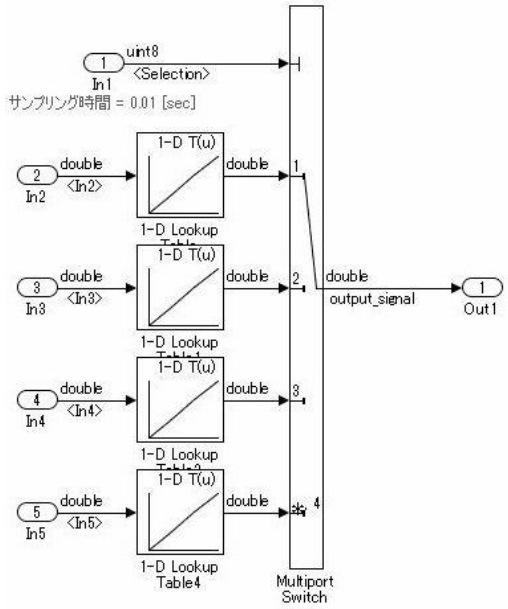
機能	Simulink パターン
[If Action Subsystem]を使用した case 構文 `switch (PRNDL_Enum)`	

```
{
  case 1
    TqEstimate = ParkV;
    break;
  case 2
    TqEstimate = RevV;
    break;
  default
    TqEstimate = NeutralV;
    break;
}
```



[Multiport Switch]を使用した
case 構文

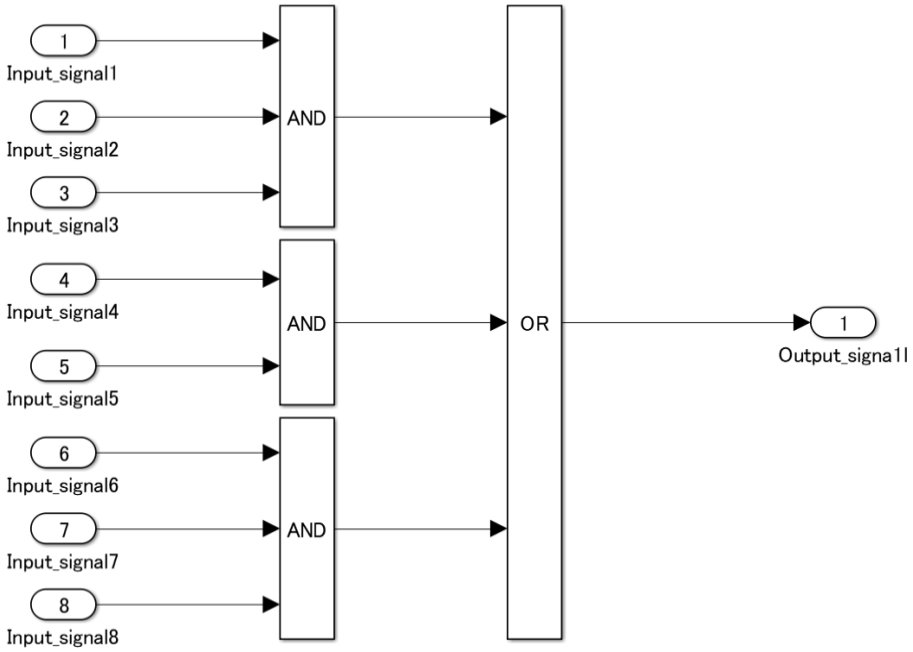
```
switch (Selection)
{
  case 1:
    output_signal =
      look1_binlxpw(In2,y1,x1,3U);
    break;
  case 2:
    output_signal =
      look1_binlxpw(In3,y2,x2,3U);
    break;
  case 3:
    output_signal =
      look1_binlxpw(In4,y3,x3,3U);
    break;
  default:
    output_signal =
      look1_binlxpw(In5,y4,x4,3U);
    break;
}
```



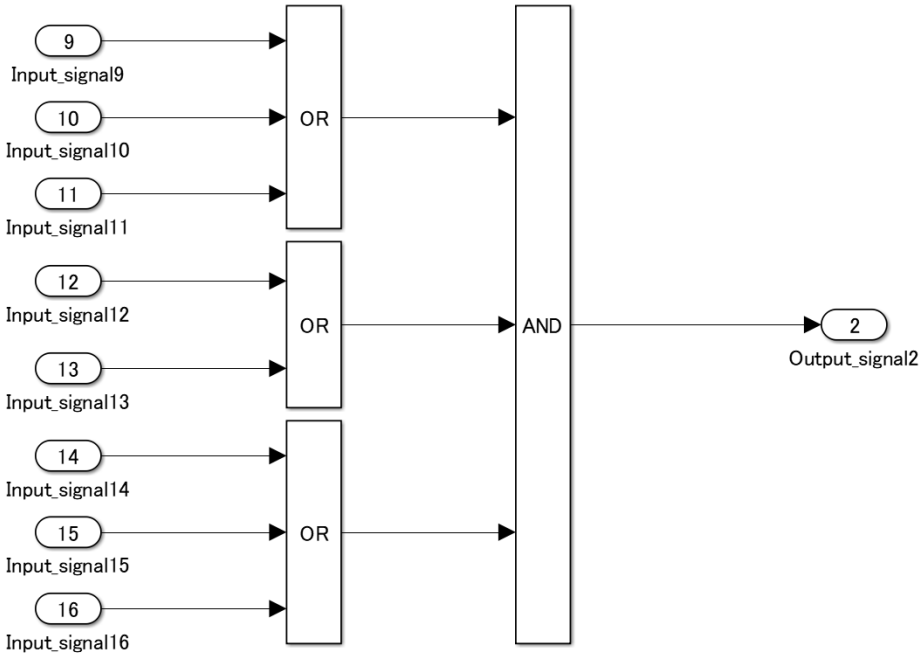
11.2.3. 付録 3：論理構文の Simulink パターン

Simulink の論理構文は、以下のパターンを使用できます。

・乗法標準形(Conjunctive normal form)



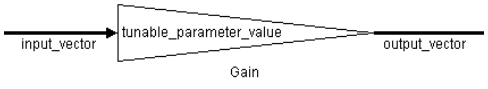
・加法標準形(Disjunctive normal form)



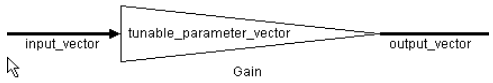
11.2.4. 付録 4：ベクトル信号の Simulink パターン

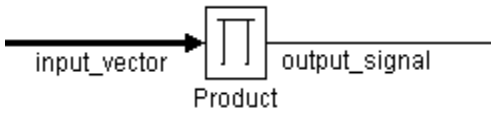
ベクトル信号は以下のパターンを使用できます。

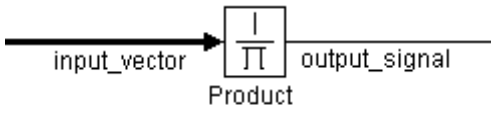
機能	Simulink パターン

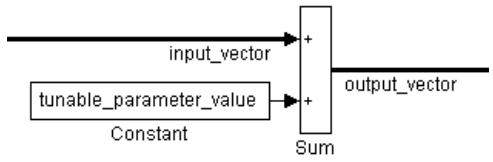
ベクトル信号とパラメータ(スカラー)の乗算
`for (i=0; i<input_vector_size; i++) {`
`  output_vector[i] = input_vector[i] *`
`  tunable_parameter_value;`
`}`
 	
(参考 R2013b のコード生成結果)
`for (i = 0; i < input_vectorDim; i++) {`
`  output_vector[i] =`
`  tunable_parameter_value *`
`  input_vector[i];`
`}`
	

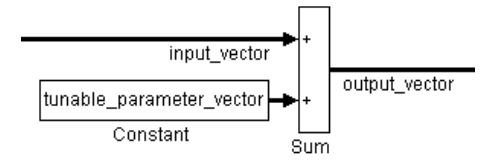
(次元数を可変でコード生成、 通常ループの上限は直値です。)	

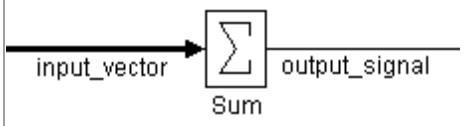
ベクトル信号とパラメーター(ベクトル)の乗算
 	
`for (i=0; i<input_vector_size; i++) {`
`  output_vector[i] = input_vector[i] *`
`  tunable_parameter_vector[i];`
`}`
	

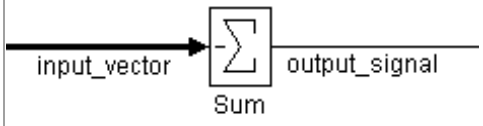
ベクトル信号の要素の乗算
 	
`output_signal = 1;`
`for (i=0; i<input_vector_size; i++) {`
`  output_signal = output_signal *`
`  input_vector[i];`
`}`
	

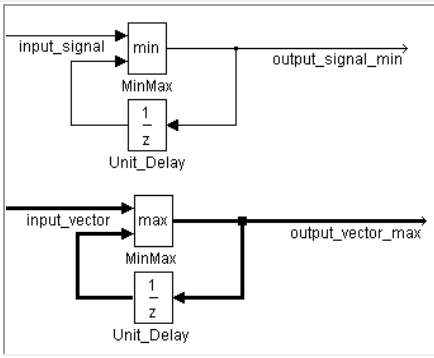
ベクトル信号の要素の除算
 	
`output_signal = 1;`
`for (i=0; i<input_vector_size; i++) {`
`  output_signal = output_signal /`
`  input_vector[i];`
`}`
	

ベクトル信号とパラメーター(スカラー)の加算
 	
`for (i=0; i<input_vector_size; i++) {`
`  output_vector[i] = input_vector[i] +`
`  tunable_parameter_value;`
`}`
	

ベクトル信号とパラメーター(ベクトル)の加算
 	
`for (i=0; i<input_vector_size; i++) {`
`  output_vector[i] = input_vector[i] +`
`  tunable_parameter_vector[i];`
`}`
	

ベクトル信号の要素の加算
 	
`output_signal = 0;`
`for (i=0; i<input_vector_size; i++) {`
`  output_signal = output_signal +`
`  input_vector[i];`
`}`
	

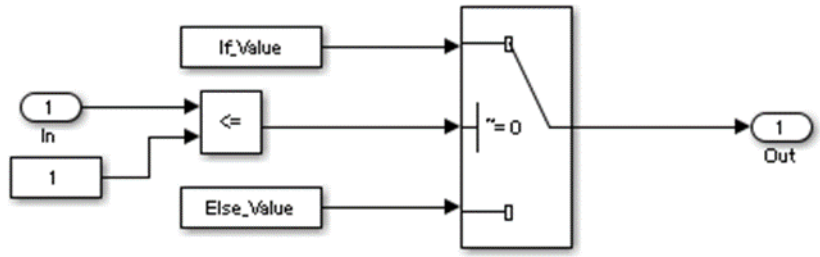
ベクトル信号の要素の減算
 	
`output_signal = 0;`
`for (i=0; i<input_vector_size; i++) {`
`  output_signal = output_signal –`
`  input_vector[i];`
`}`
	

最小値 / 最大値の保持	

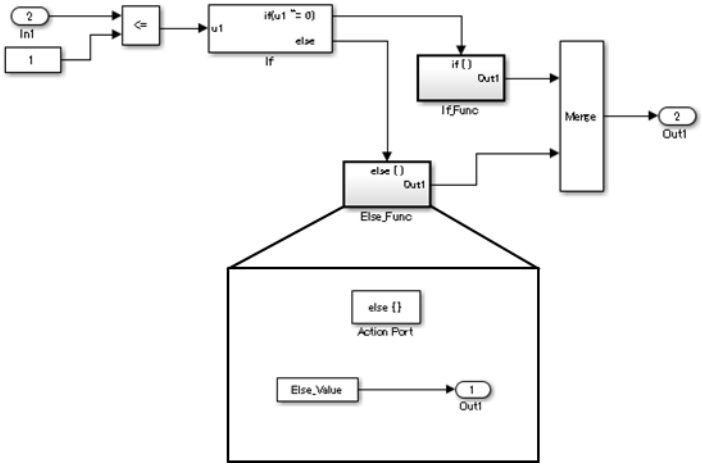
11.2.5. 付録 5： Switch と if-then-else Action Subsystem の使い分け

関連する then アクションと else アクションに定数値の代入のみが含まれるような、単純な if-then-else 構造のモデル化には [Switch]を使用してください。

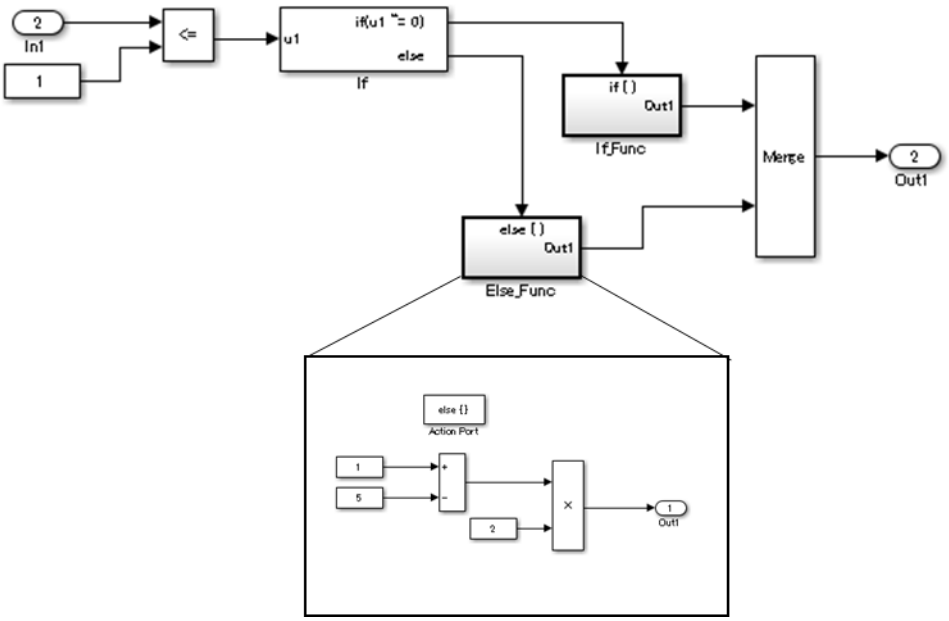
【推奨】単純な if-then-else 構造のため、[Switch]を使用しています。



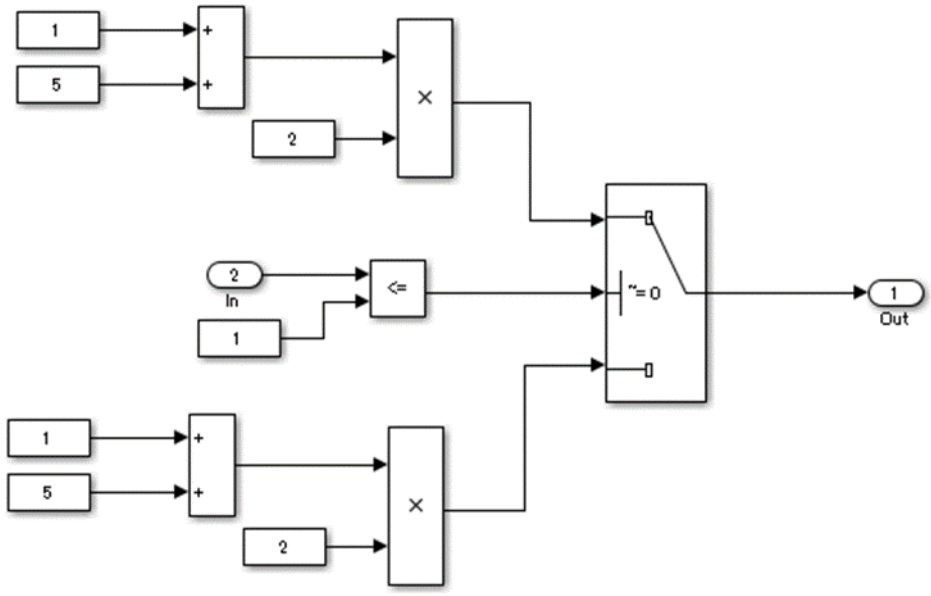
【非推奨】単純な if-then-else 構造ですが、[If]、[If Action Subsystem]を使用しています。



【推奨】複雑な if-then-else 構造のため、[If]、[If Action Subsystem]を使用しています。



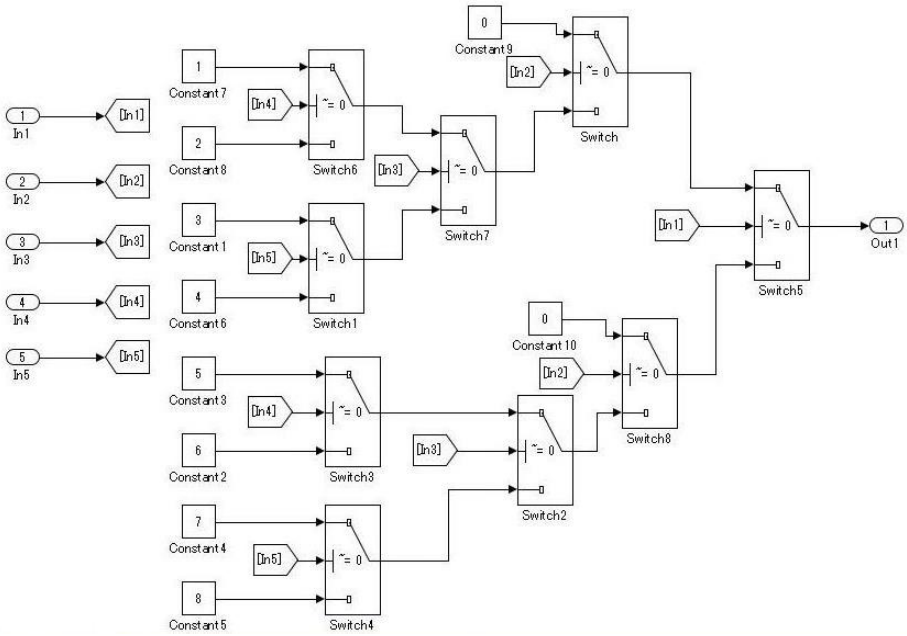
【非推奨】複雑な if-then-else 構造ですが、[Switch]を使用しています。



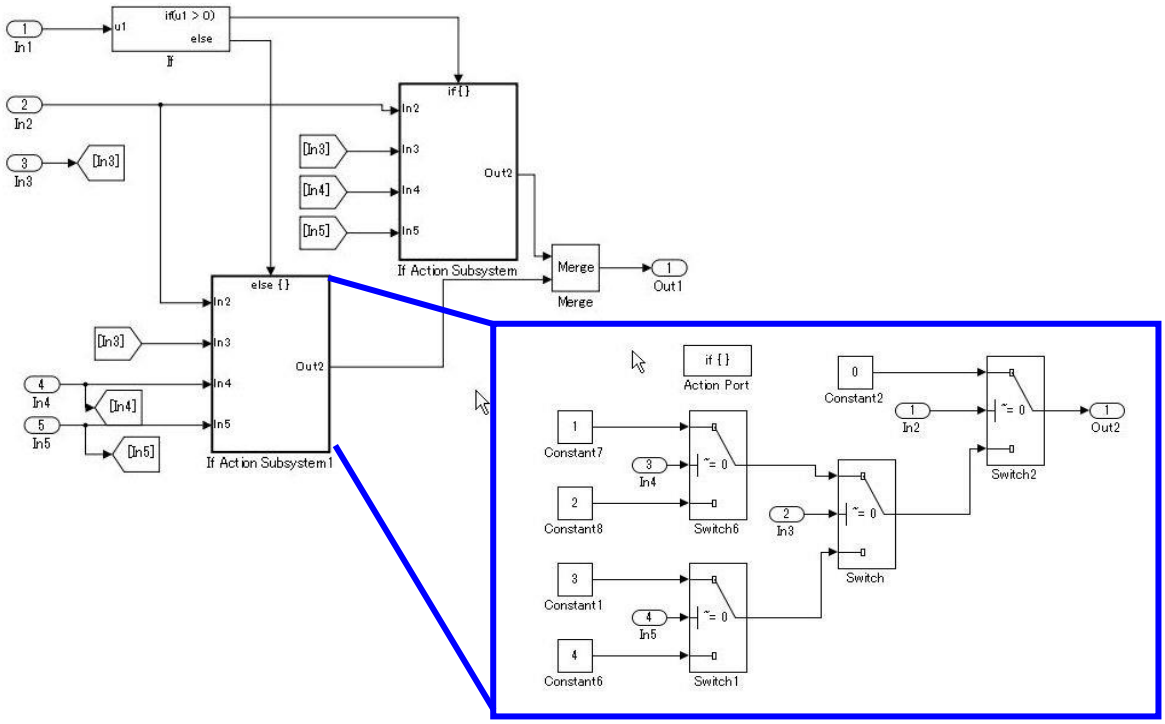
11.2.6. 付録 6：複数の Switch の if-then-else Action Subsystem への置き換え

[Switch]による条件分岐の多用は避けるべきです。上限目標値を設定し、運用します。(例えば 3 段まで)目標値を超えた場合、代わりに If-Then-else Action Subsystem を使用した条件付制御フローを用いて記載することができます。

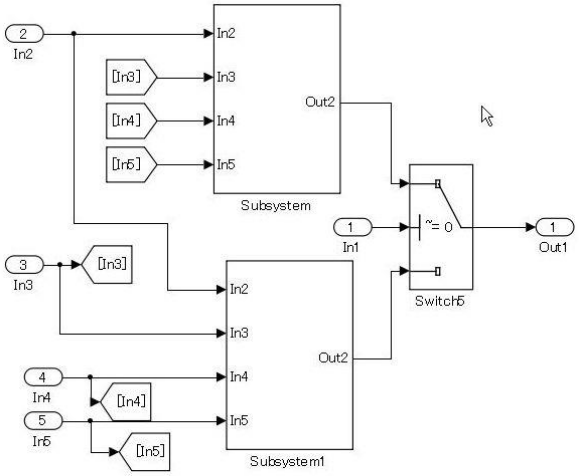
【非推奨】ネストが 4 段になっています。



【推奨】4 段目を if-action Subsystem にすることで、一つの階層内のネストを制限しています。

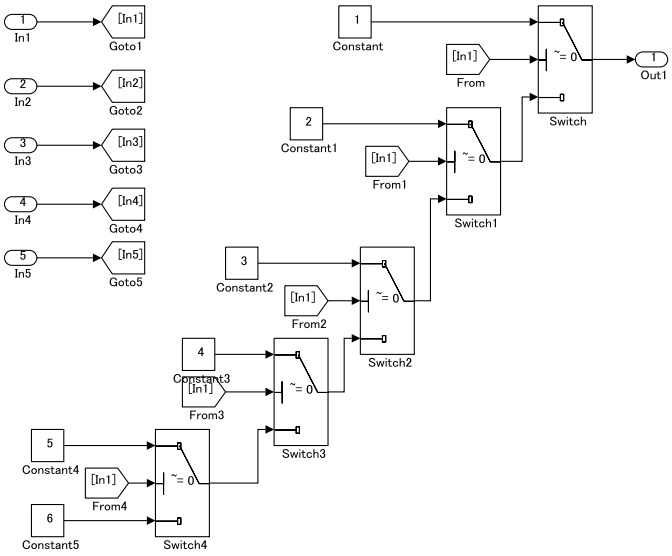


【非推奨】if –action の形式で分割していません。

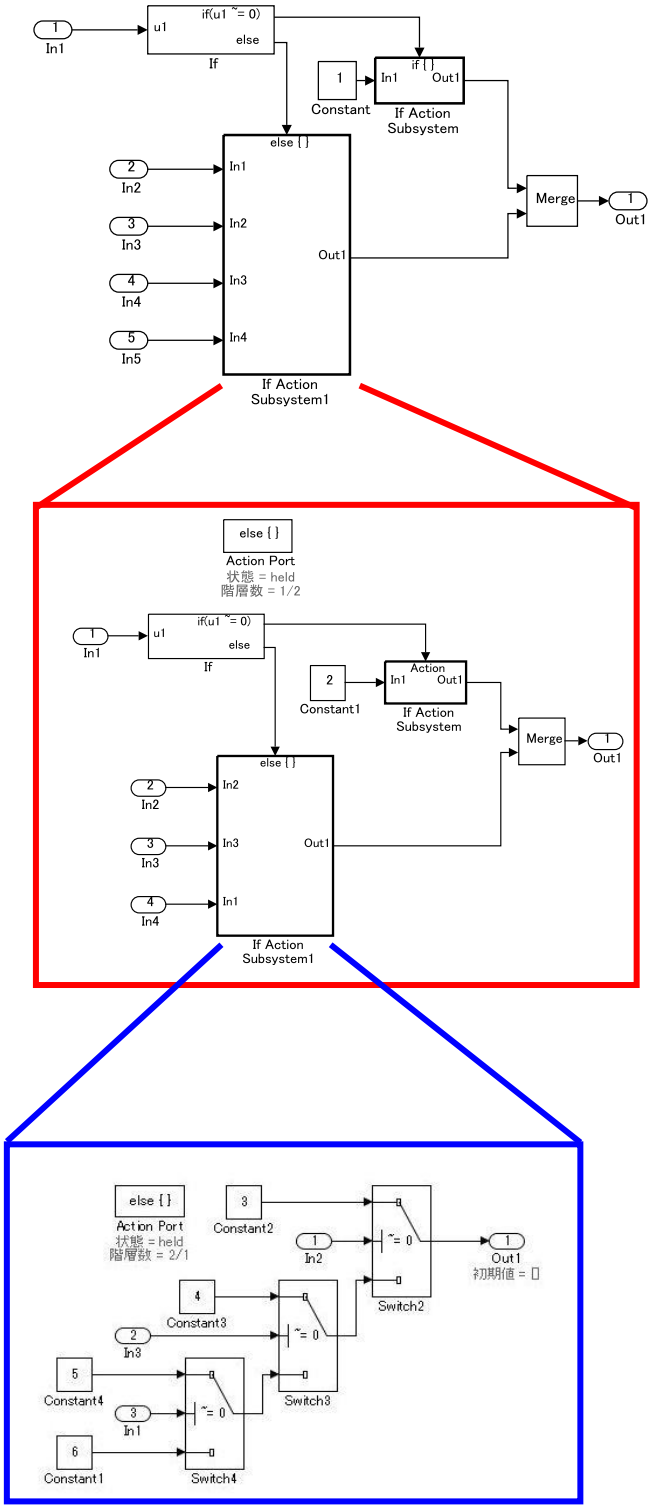


C コード上の制限が反映されるようにする場合は、Atomic Subsystem＋関数設定で分離することができます。その場合は if-then-else Action Subsystem を使う必要はなく、Switch ブロックの構成を途中で分離し、サブシステムでカプセル化するだけです。

5 段のネストを持つモデル例
【非推奨】



【推奨】Switch のネストを階層化によって回避した記述方法



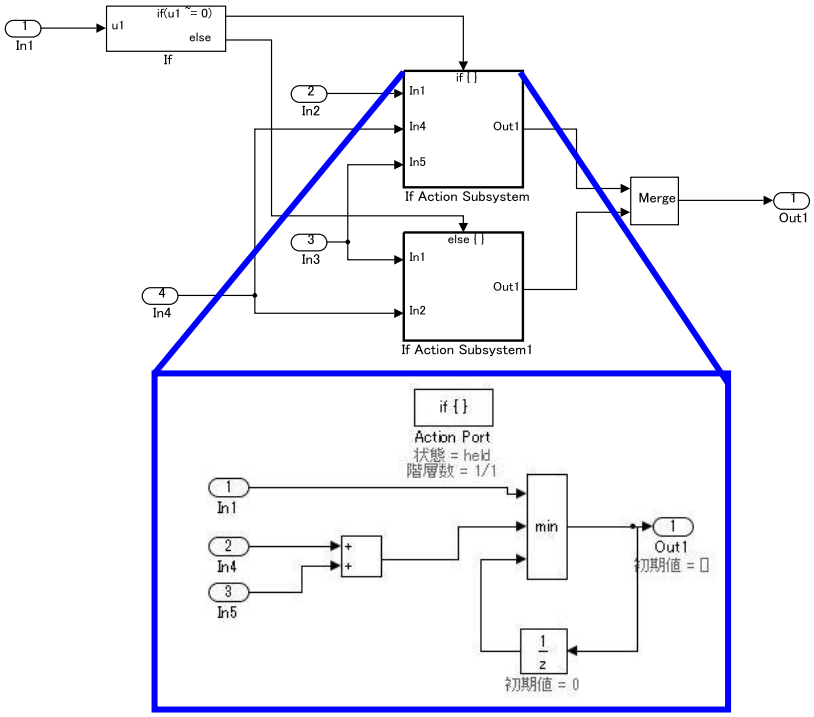
事例の説明として記載しているが、本来固定値の切り替えに If Action Subsystem は使用しません

上記の【推奨】【非推奨】共に、ユーザーが関数化設定を付加しなければ、生成される C コードは同一です。(R2010b~R2013a で確認済み)C コード上の制約ではありません。

11.2.7. 付録 7 : 条件付き制御フローを用いた Action Subsystem の使用ルール

関連する全てのアクションに状態変数を持たない場合は、[If Action Subsystem]を用いてはなりません。

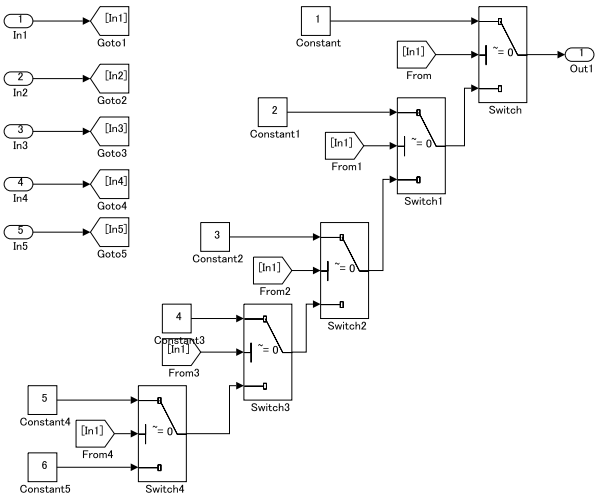
【推奨】



5 段のネストを持つモデルの例

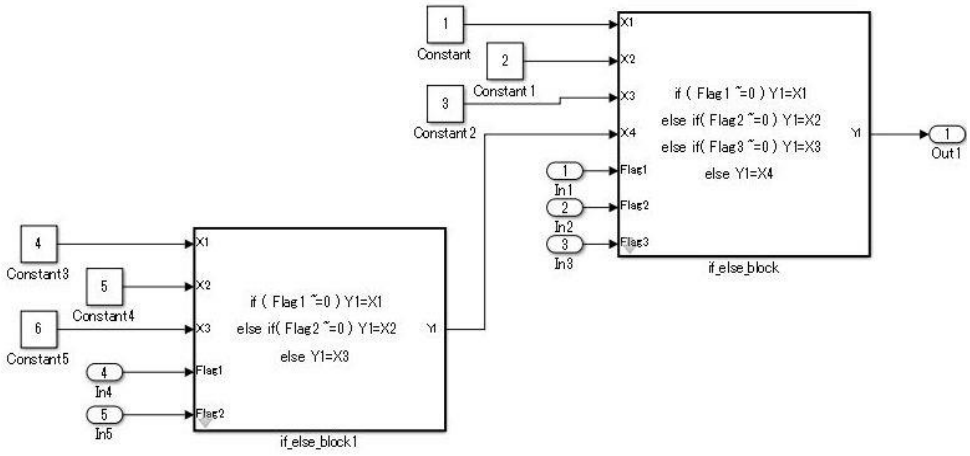
【推奨】

内部状態がないので、サブシステムを用いた階層化を行わない。

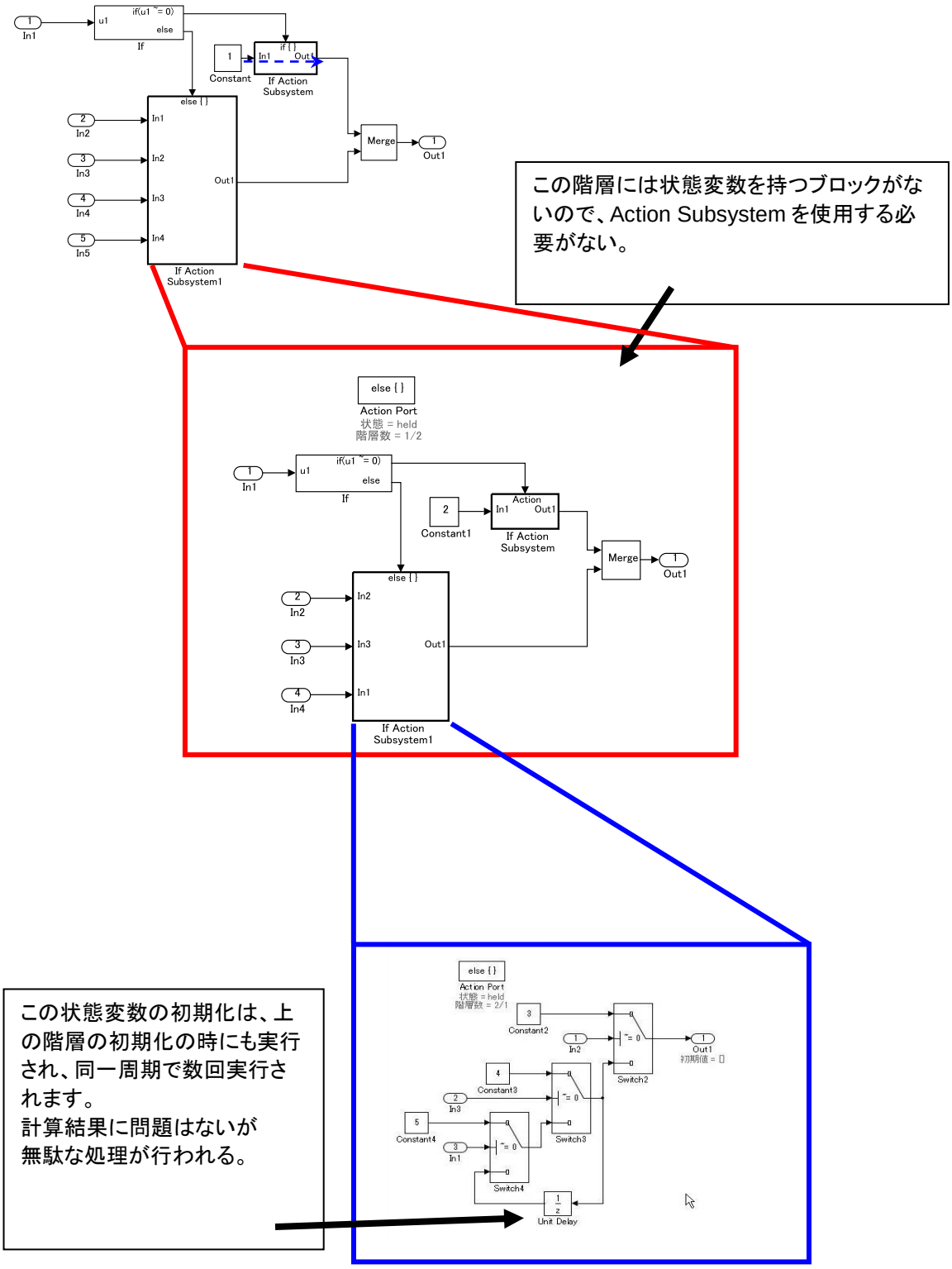


【推奨】

Action Subsystem は用いず、Switch の前後で Atomic Subsystem で分離します。



【非推奨】
不必要な Action Subsystem を用いた階層化を行っている。



Action Subsystem を使用しなくても機能が実現できる場合は、Action Subsystem を用いた階層化を行わないようにします。

誤りの例題は、3 階層目に存在する最下層の UnitDelay を初期化する時、まず上位の 1 階層目の条件付サブシステムの初期化で 1 回、2 階層目の条件付サブシステムの初期化で合計 2 回の初期値設定が実行されます。不要なコードを生成させない為にも、状態変数が存在しない階層は、条件付サブシステムで記載しないことを推奨します。

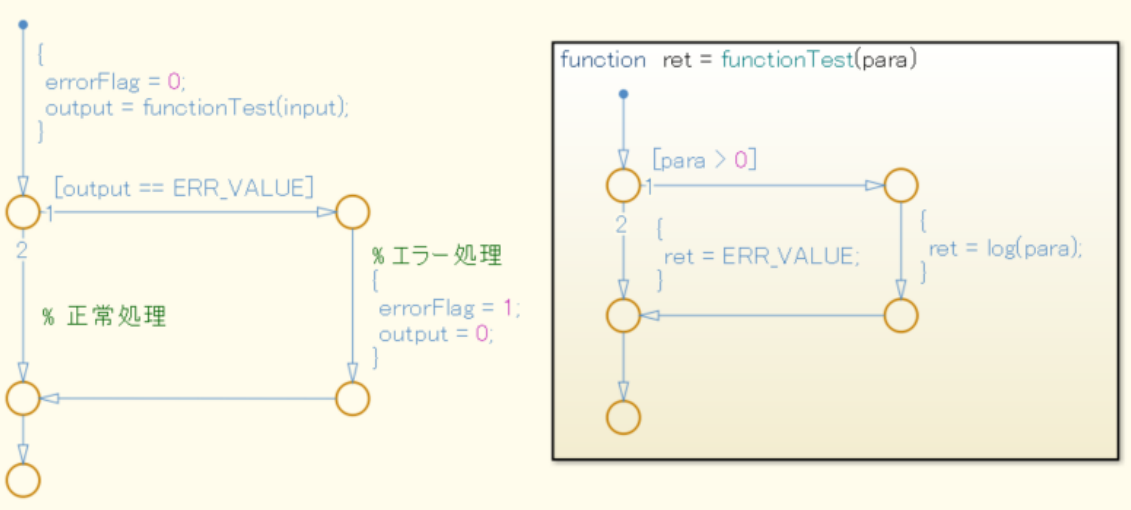
階層に落とすことでモデルの(コードではない)複雑度が下がると考える思想に基づきます。本ルールは、不要な初期化を実行させないことが目的のルールです。

本ルールの規制とは無関係ですが、分岐条件のネストが深い構造になるシステムに対して、コードの分岐ネストを下げる為には、関数化によって分岐を分離します。その為には、Switch ブロック前後の機能をそれぞれのサブシステムに分割し、Atomic Subsystem＋function の関数設定を行います。ただし、不用意に実施すると不要な RAM が追加される可能性があるので、トレードオフの確認が必要です。どちらも一長一短があるので、モデルにあった記述方法を選択してください。

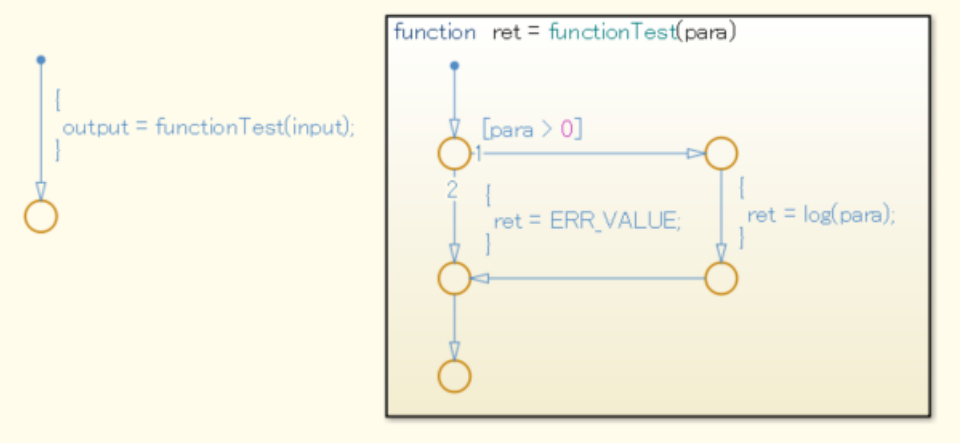
11.2.8. 付録 8：エラー情報に対するテスト

Stateflow で使用する関数(グラフィカル関数、MATLAB 関数等)がエラー情報を返す場合は、エラー情報がテストされるようなモデル構成とします。
関数が返すエラー情報を確認しない場合、意図しない動作となる可能性があります。

【推奨】
エラー情報を確認し、エラー処理を行います。



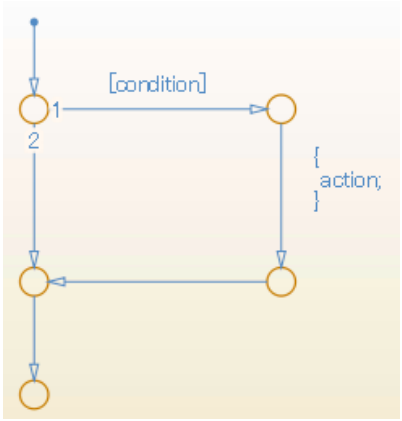
【非推奨】
エラー情報を確認しません。

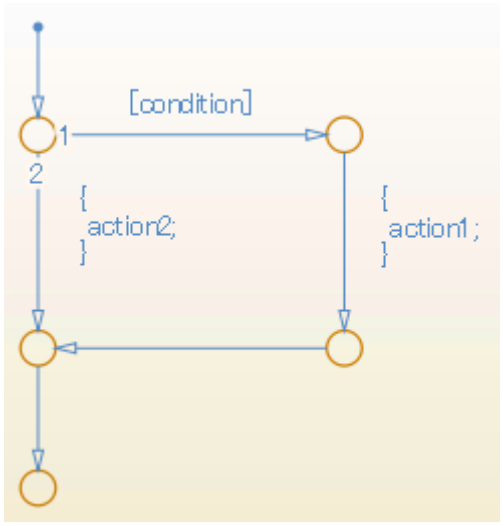


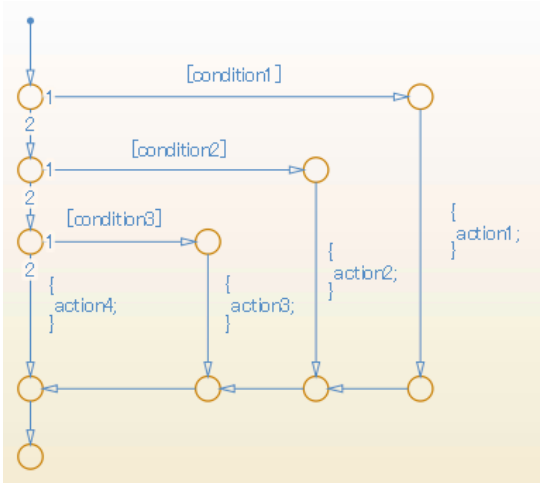
11.2.9. 付録 9：if 構文のフローチャートパターン

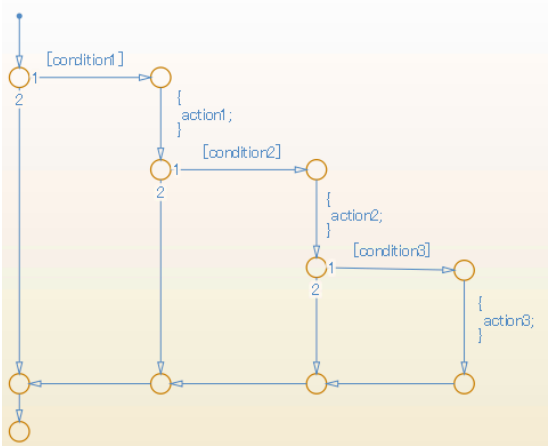
フローチャートの if 構文には、以下のパターンを使用します。

機能	フローチャートのパターン

if-then 構文
 	
`if (condition){`
`  action;`
`}`
	

if-then-else 構文
 	
`if (condition) {`
`  action1;`
`}`
`else {`
`  action2;`
`}`
	

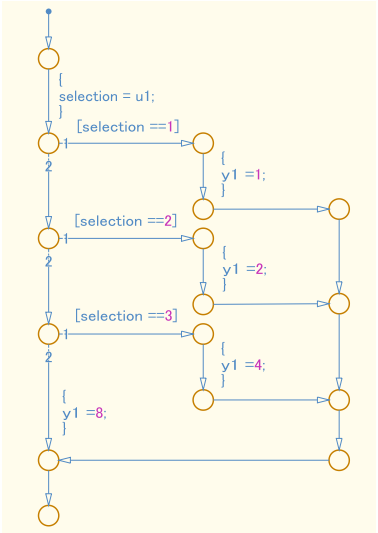
if-then-else-if 構文
 	
`if (condition1) {`
`  action1;`
`}`
`else if (condition2) {`
`  action2;`
`}`
`else if (condition3) {`
`  action3;`
`}`
`else {`
`  action4;`
`}`
	

if-then 構文のネスト
 	
`if (condition1) {`
`  action1;`
`  if (condition2) {`
`  action2;`
`  if (condition3) {`
`  action3;`
`  }`
`  }`
`}`
	

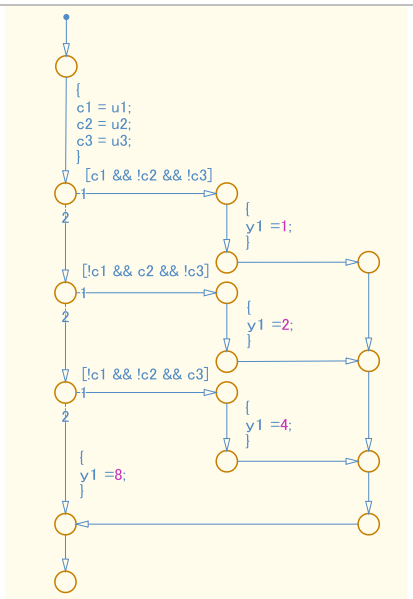
11.2.10. 付録 10：case 構文のフローチャートパターン

フローチャートの case 構文には、以下のパターンを使用できます。

機能	Simulink パターン

排他的選択による case 構文
 	
`selection = u1;`
`switch (selection) {`
`  case 1:`
`  y1 = 1;`
`  break;`
`  case 2:`
`  y1 = 2;`
`  break;`
`  case 3:`
`  y1 = 4;`
`  break;`
`  default:`
`  y1 = 8;`
`}`
	

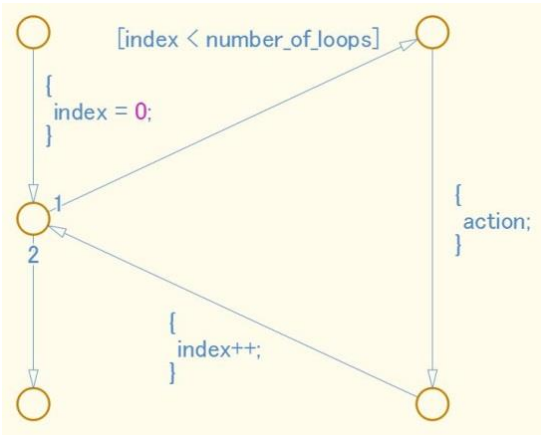
排他的条件による case 構文
 	
`c1 = u1;`
`c2 = u2;`
`c3 = u3;`
`if (c1 && !c2 && !c3) {`
`  y1 = 1;`
`}`
`elseif (!c1 && c2 && !c3) {`
`  y1 = 2;`
`}`
`elseif (!c1 && !c2 && c3) {`
`  y1 = 4;`
`}`
`else {`
`  y1 = 8;`
`}`	

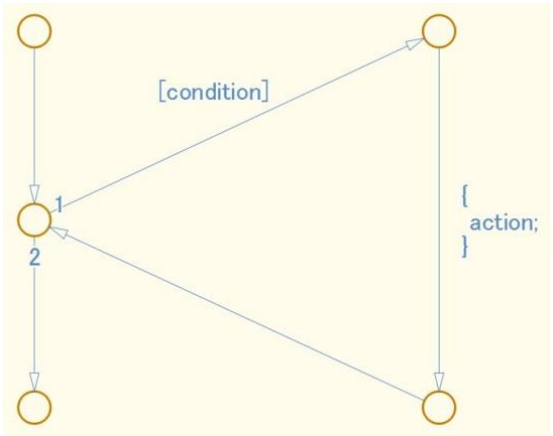
	

11.2.11. 付録 11 : ループ構文のフローチャートパターン

フローチャートのループ構文には、以下のパターンを使用できます。

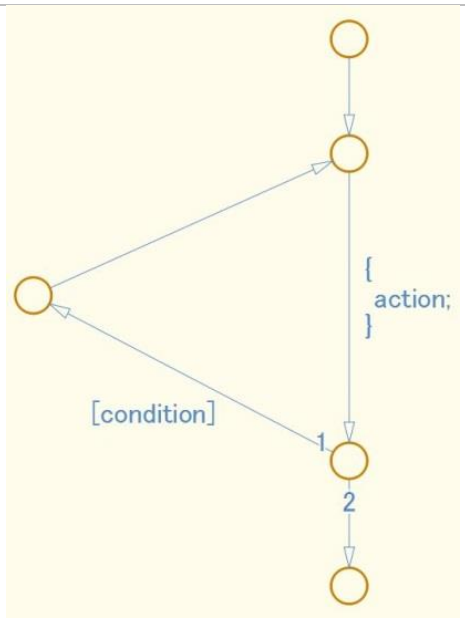
機能	フローチャートのパターン

for ループ構文
 	
`for ( index = 0;`
`index < number_of_loops;`
`index++ )`
`{`
`action;`
`}`
	

while ループ構文
 	
`while ( condition )`
`{`
`action;`
`}`
	

do while ループ構文 do	

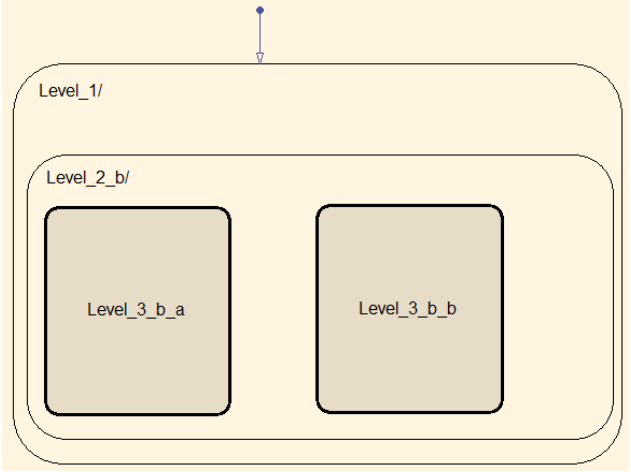
```
{
  action;
}
while ( condition )
```



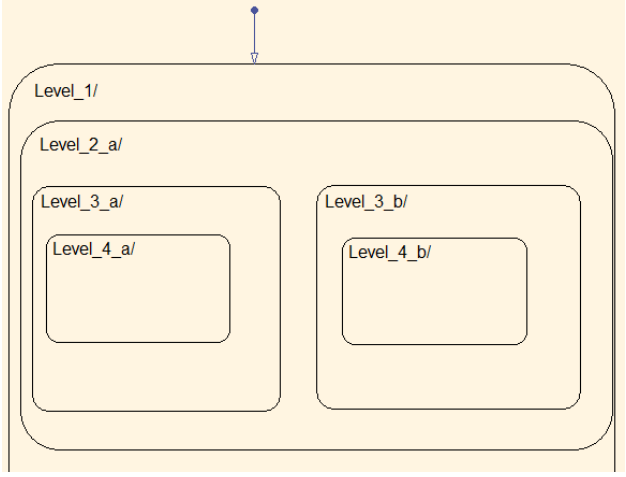
11.2.12. 付録 12：状態の階層化の制限

一つのビューアー(サブビューアー)内で、多層の階層化は限定されるべきです。1 ビューアー(サブビューアー)内での制限目標を設け、制限目標を超える場合は、サブチャート化によって画面を切り替えます。

【推奨】
4 階層目がサブチャートにカプセル化されています。

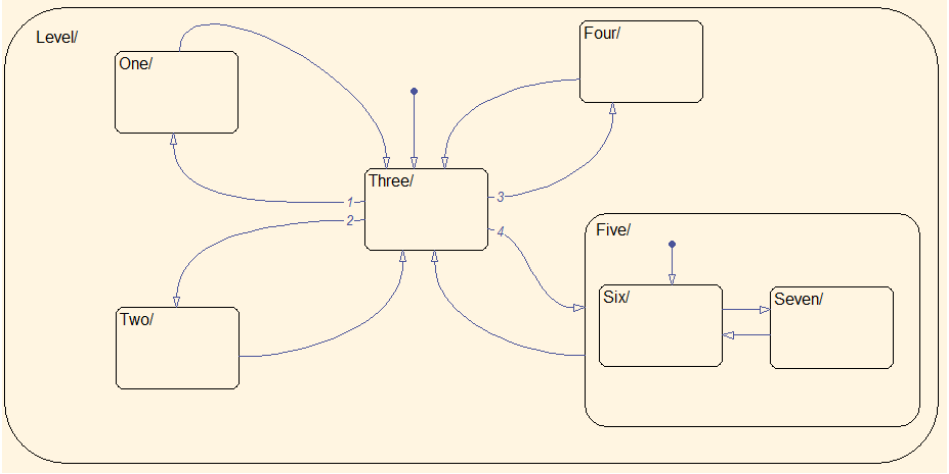


【非推奨】
制限目標を 3 階層とした場合、Level_4_a と Level_4_b が 3 階層より多くネストされています。



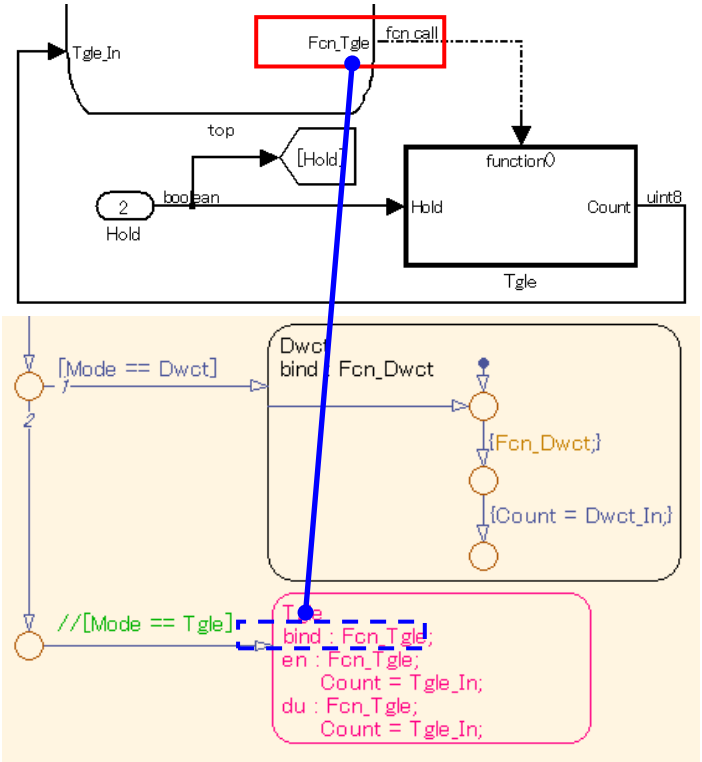
11.2.13. 付録 13：Stateflow の 1 画面あたりの状態数

Stateflow の 1 画面あたりの状態数は、可視性を考慮すべきです。状態数は、ダイアグラム内で表示可能な状態によります。



11.2.14. 付録 14：Stateflow からの Function Call

呼び出し先の[Function-Call Subsystem]内に状態が存在し、かつ、呼び元の状態が非アクティブになったら状態の”reset”が必要な場合、呼び出し元で bind アクションを記述します。



11.2.15. 付録 15：Stateflow 内で使用可能な関数の種類

どの関数を使用すべきかは、必要とされる処理により選択します。

・グラフィック関数

- If Then Else　ロジック
- Simulink 関数
- 伝達関数
- 積分器
- ルックアップテーブル
- MATLAB Function
- 複雑な方程式
- If Then Else　ロジック

11.3. ガイドラインに対するサンプルプログラム

ガイドラインの一部は、モデルアドバイザーによるチェックだけでなく、条件を決めれば、自動で設定する事ができるルールがあります。ここでは自動的に設定にする方法について、いくつかのサンプルプログラムを掲示します。

モデルベース開発は、このように自動的な修正によって工数を削減し品質を向上させる事が可能です。ルールを作ってユーザーにルールを守らせるだけではなく、自動的に修正する事によって、利便性を向上させる事も必要です。

11.3.1. ガイドライン準拠のための自動設定方法

11.3.1.1. Simulink モデルの表示設定の例

・「na_0004：Simulink モデルの表示設定」

```
SettingItems= {...
  ...% 表示オプション
  'ModelBrowserVisibility',      'off',   'ブラウザ表示';...
  'ScreenColor',                 'white', 'スクリーンカラー';...
  'StatusBar',                   'on',    'ステータスバー';...
  'ToolBar',                     'on',    'ツールバー';...
  'ZoomFactor',                  '100',   'ズームバー&';....
  ... % 端子の表示オプション
  'ShowPortDataTypes',           'off',   '端子のデータタイプ';...
  'ShowLineDimensions',          'off',   '信号の次元';...
  'ShowStorageClass',            'off',   'ストレージクラス';...
  'ShowTestPointIcons',          'on',    'テストポイントインジケータ';...
  'ShowSignalResolutionIcons',   'on',    'テストポイントインジケータ';...
  'ShowViewerIcons',             'on',    'ビューワインジケータ';....
  'WideLines',                   'on',    '非スカラー信号線を太く表示';...
};

for k=1 :  size(SettingItems,1)
  set_param(0,SettingItems{k,1},SettingItems{k,2})
end
```

set_param(0,SettingItems{k,1},SettingItems{k,2})は、Simulink への設定です。設定を行えば、新規に作るモデルファイルに上記の設定が引き継がれます。この設定は、Simulink の再起動時に再び実行しなければ有効になりません。パス上に存在する startupsl.m ファイルに記載すると Simulink 起動時に設定が実施されます。

既存ファイルの設定を変える場合は、set_param(bdroot,SettingItems{k,1},SettingItems{k,2})を用いて変更します。

11.3.1.2. Simulink モデルで仕様するフォントとフォントサイズ設定の例

・「db_0043：モデルで使用するフォントとフォントサイズ」

```
SettingItems= {...
  ...%% フォント設定
  ...% ブロックのデフォルトフォント設定
  'DefaultBlockFontName',         'MS UI Gothic', 'ブロックのデフォルトフォント名'; ...
  'DefaultBlockFontSize',         12,             'ブロックのデフォルトフォントサイズ'; ...
  'DefaultBlockFontWeight',       'normal'        'ブロックのデフォルトフォントの太さ'; ...
  'DefaultBlockFontAngle',        'normal',       'ブロックのデフォルトフォントの傾き'; ...
  ...% 信号線のデフォルトフォント設定
  'DefaultLineFontName',          'MS UI Gothic', '信号線のデフォルトフォント名'; ...
  'DefaultLineFontSize',          12,             '信号線のデフォルトフォントサイズ'; ...
  'DefaultLineFontWeight',        'normal',       '信号線のデフォルトフォントの太さ'; ...
  'DefaultLineFontAngle',         'normal',       '信号線のデフォルトフォントの傾き'; ...
  ...% 注釈のデフォルトフォント設定
  'DefaultAnnotationFontName',    'MS UI Gothic', '注釈のデフォルトフォント名'; ...
  'DefaultAnnotationFontSize',    14,             '注釈のデフォルトフォントサイズ'; ...
  'DefaultAnnotationFontWeight',  'normal',       '注釈のデフォルトフォントの太さ'; ...
  'DefaultAnnotationFontAngle',   'normal',       '注釈のデフォルトフォントの傾き'; ...
};

for k=1 : size(SettingItems,1)
  set_param(0,SettingItems{k,1},SettingItems{k,2})
end
```

既存ファイルに設定する場合、set_param(bdroot, SettingItems{k,1},SettingItems{k,2})だけを実行しても、全て変更されるわけではありません。既に手動で変更された内容まで含めて変更する場合、find_system を用いてモデルファイル内すべての情報を検索し変更する必要がありますが、意図して変更した記述まで統一される可能性がありますので、なるべく新規の段階での設定をお勧めします。

11.3.1.3. Stateflow チャートのビット演算可否設定の例

・「na_0001_a:Stateflow における演算子の統一」
下記の例は、既に存在するモデルに対して、内部に存在する[Chart]の設定を変更する例です。

```
rt = sfroot;
modelH = get_param(bdroot, 'Handle');
rt = rt.find('-isa', 'Simulink.BlockDiagram', '-and', 'handle', modelH);
result = rt.find('-isa', 'Stateflow.Chart');
if ~isempty(result)
  for n1=1:length(result)
    result(n1).EnableBitOps=true;
  end
end
```