## ブロック実行順序を決定するためのタスクベースのソート

## Task-Based Sorting to Determine Block Execution Order

**概観**

タスクベースの並べ替えは、派生したサンプル時間情報に基づいてブロックとポートの実行順序を設定します。ブロックの実行順序を決定するためのより効率的で単純なプロセスを提供することにより、ブロックのソートを強化します。

タスクベースの並べ替え：

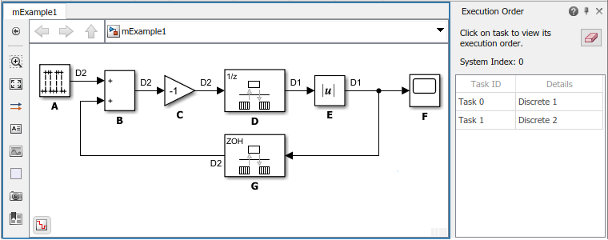
* タスクは、サンプル時間に基づいて個別に並べ替えられます。
* すべてのタスクにわたってブロックの1つのフラット化されたソートされたリストの代わりに、複数のソートされたリストが生成されます。
* レート遷移処理が簡素化されています。
* さまざまなタスクでのブロックの関与により、誤ったデータ依存違反が回避されます。
* コード生成の結果は、より効率的なレートグループに分類されます。
* 1つのサブシステムは、複数のソートされたリストに属することができます。

APIは、タスクベースのソートされたリストの詳細情報を取得する機能を提供します。更新アドバイザーチェックは、タスクベースの並べ替えとブロックの並べ替えの間の動作の変更に対処します。

**実行順序ビューア**

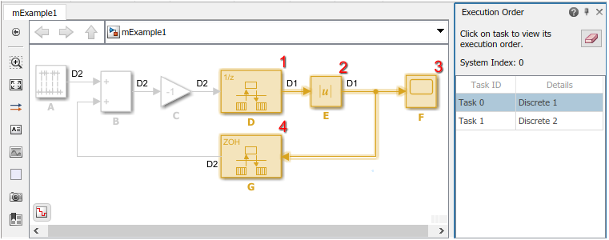
実行順序ビューアにタスクのリストが表示されます。タスクを選択すると、そのタスクに属するブロックの実行順序が表示されます。

1. **デバッグ** ]タブとで**診断** セクション。[ **情報オーバーレイ]**ドロップダウン矢印を選択します。で **ブロック**ダイアログのセクション、選択**実行オーデルを**。実行オーダービューアーがSimulinkエディターの右側のペインで開きます。



1. タスクテーブルでタスクをクリックします。

選択した行が青色で強調表示されます。タスクのすべてのブロックがブロック図で強調表示されます。タスク内のブロックの実行順序は、赤い数字で表示されます。他のタスクのブロックはグレー表示されます。



**実行順序API**

提供されているAPIを使用して、ブロックのタスクと実行順序を表示します。

1. モデル内のタスクを取得します。

<説明>

>> sLists = get\_param（ 'mExample1'、 'SortedLists'）

sLists =

フィールドを持つ2×1構造体配列：

TaskIndex

SampleTimes

SortedBlocks

1. 最初のタスクのタスクインデックス（Task 0実行順序ビューアー内）

<説明>

>> sLists（1）

ans =

フィールドを持つ構造体：

TaskIndex：0

SampleTimes：[1×1構造体]

SortedBlocks：[4×1構造体]

1. のサンプル時間の詳細Task 0。

<説明>

>> sLists（1）.SampleTimes

ans =

フィールドを持つ構造体：

RateSpec：[1×1構造体]

注釈：「D1」

説明：「個別1」

>> sLists（1）.SampleTimes.RateSpec

ans =

フィールドを持つ構造体：

期間：1

オフセット：0

rateType： 'ClassicPeriodicDiscrete'

1. のブロックTask 0。
2. >> sLists（1）.SortedBlocks.BlockType
3. ans =
4. 「RateTransition」
5. ans =
6. 「腹筋」
7. ans =
8. '範囲'
9. ans =
10. 「RateTransition」
11. の最初のブロックの詳細Task 0。
12. >> sLists（1）.SortedBlocks（1）
13. ans =
14. フィールドを持つ構造体：
15. BlockHandle：109.0001
16. BlockPath： 'mExample1 / D'
17. BlockType： 'RateTransition'
18. InputPorts：[]

出力ポート：1

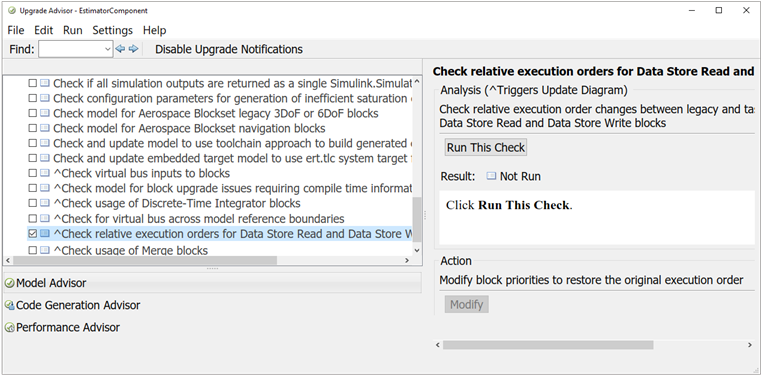
1. ブロックの実行（ソート）順序を取得します。
2. >> blockOrder = get\_param（ 'mExample1 / D'、 'SortedOrder'）
3. blockOrder =
4. フィールドを持つ構造体：
5. TaskIndex：0
6. SystemIndex：0

BlockIndex：1

**データストアのメモリ変更の更新アドバイザーチェック**

ブロックの並べ替えの代わりにタスクベースの並べ替えを使用すると、Data Store Memoryブロックに関連する相対的な実行順序を変更できます。モデルアドバイザーチェックは変更を検出し、モデルを元の実行順序で更新するオプションを提供します。

1. アップグレードアドバイザーを開きます。上の**モデル**タブとで **評価し、管理**セクション、クリック**モデルアドバイザーを**。ドロップダウンダイアログから、[ **Update Advisor** ]を選択し**ます**。
2. のチェックボックスを選択しますCheck relative execution orders for Data Store Read and Data Store Write blocks。



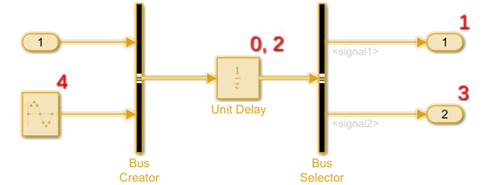
1. [ **このチェック**を**実行** ]ボタンをクリックします。
2. **結果**テーブルの変更を確認します。元の実行順序を保持する場合は、[ **変更** ]ボタンをクリックします。

**タスクベースの並べ替えを使用したモデルパターン**

このセクションでは、一般的なモデルパターンのタスクベースの並べ替えを使用したブロックの実行順序を示します。

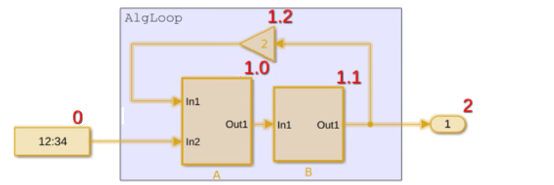
**バス拡張**

バスをDelayブロックの入力に接続すると、それぞれが独自のブロック実行順序を持つ複数の非表示ブロックが作成されます。



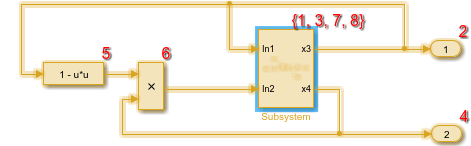
**代数ループ**

代数ループ内のブロックは、非仮想の非表示サブシステムに移動されます。隠しサブシステムの実行順序は、他のモデルブロックのコンテキスト内で決定され、次に、隠しサブシステム内のブロックの実行順序が決定されます。



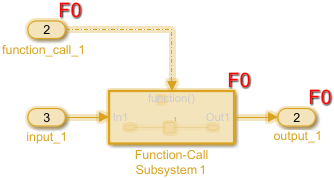
**仮想サブシステム**

仮想サブシステムの場合、サブシステム内のブロック実行順序は、中括弧{}内にリストされています。



**関数呼び出しサブシステムブロック**

以下のために関数呼び出しサブシステムのブロック、関数呼び出し のInportブロック、サブシステムブロック、およびのOutport ブロックが一緒に実行されます。モデル内の他のブロックに対する実行順序はシミュレーション前に決定できないため、ブロックはF0やF1などのオーバーレイを使用して関数呼び出し順序を表示します。



次のコマンドを使用して、非表示のブロックを含む実行順序を表示できます。Inportブロックが隠さに関連付けられている関数呼び出しジェネレータブロック、およびファンクションコールサブシステムのブロックはの実行順序でリストされています2。

>> set\_param（ 'export\_function\_model'、 'DisplaySortedLists'、 'on'）

>> get\_param（ 'export\_function\_model'、 'SortedLists'）

---- 'export\_function\_model'の並べ替えられたリスト[6つの非仮想ブロック、directFeed = 1]

タスクの総数= 2

-タスクインデックスのソートされたリスト[1]、3つの非仮想ブロック

（1）0：1 'export\_function\_model / RootFcnCall\_InsertedFor\_function\_call\_1\_at\_outport\_1'（RootInportFunctionCallGenerator）

入力ポート：[0]

出力ポート：[0]

（1）0：2 'export\_function\_model / Function-Call Subsystem 1'（サブシステム）

入力ポート：[0]

出力ポート：[0]

（1）0：3 'export\_function\_model / output\_1'（アウトポート）

入力ポート：[0]

出力ポート：[]