Subsystems

Subsystems allow you to build hierarchical models by grouping related blocks into a single Subsystem block. Use subsystems to simplify large and complex models, keep functionally related logic together, and reduce the number of blocks displayed in the model window while preserving execution behaviour.

There are various types of subsystems. They are:

- Basic subsystem
- Enabled Subsystem
- Triggered Subsystem
- Triggered and Enabled Subsystem
- Control Flow Subsystem

Basic Subsystem

Overview

A basic subsystem is a set of blocks that you replace with a single block known as a subsystem block. The Subsystem block appears on one layer of the model, while the blocks that implement its behaviour appear on another layer inside the subsystem.

Using a basic subsystem, you can:

- Establish a hierarchical block diagram.
- Keep functionally related blocks together.
- Reduce the number of blocks displayed in the model window.

A subsystem can be created in different ways. You can add a Subsystem block to the model and then add blocks inside the subsystem window. You can also group existing blocks into a subsystem. Another option is to create a subsystem using on-canvas context menu options.

When you copy a subsystem, the copy is independent of the original subsystem. To reuse the contents of a subsystem within a model or across models, use either model referencing or a library.

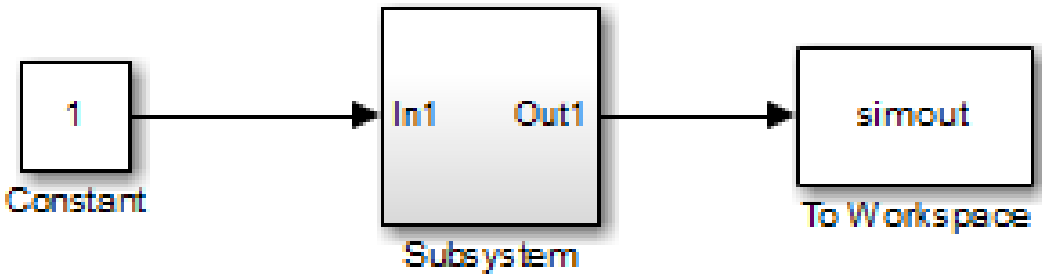

Figure 1. A Subsystem block representing grouped blocks in a model.

Procedure: Create a Subsystem Before Adding Blocks

You can define the subsystem structure before adding logic. To do so:

1. **User action:** Copy the Subsystem block from the **Ports** and **Subsystems library** to the model.
   System response: An empty Subsystem block is added to the model.
2. **User action:** Double-click the Subsystem block to open it.
   System response: The subsystem window opens.
3. **User action:** Add blocks to the subsystem window.
   System response: The blocks become part of the subsystem.
4. **User action:** Add Inport and Outport blocks.
   System response: The subsystem accepts input signals and provides output signals to external blocks.

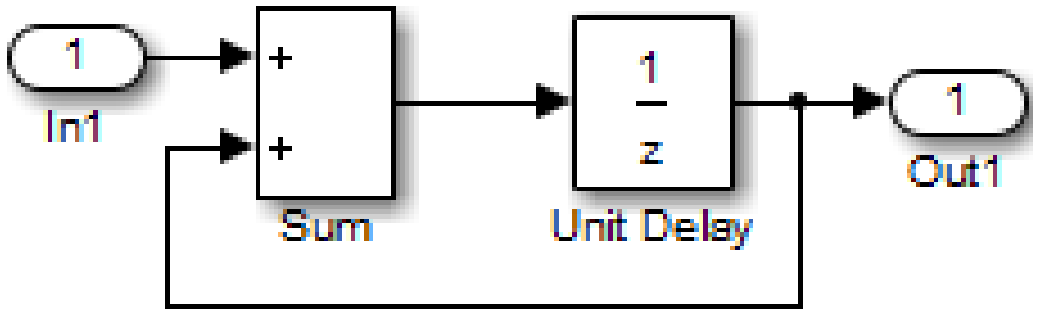

Figure 2. Blocks inside a subsystem with Inport and Outport blocks.

Procedure: Create a Subsystem from Existing Blocks

If a model already contains blocks, you can group them and create a Subsystem:

1. **User action:** Select the blocks you want to include.
   System response: The selected blocks are highlighted.
2. **User action:** Drag a bounding box to include the blocks and connecting lines.
   System response: All enclosed blocks and lines are selected.
3. **User action:** To create a subsystem from the selection, choose the **Diagram**, select **Subsystems and Model Reference**, and click **Create Subsystem from Selection**.
   System response: You get a single Subsystem block in place of the selected blocks.

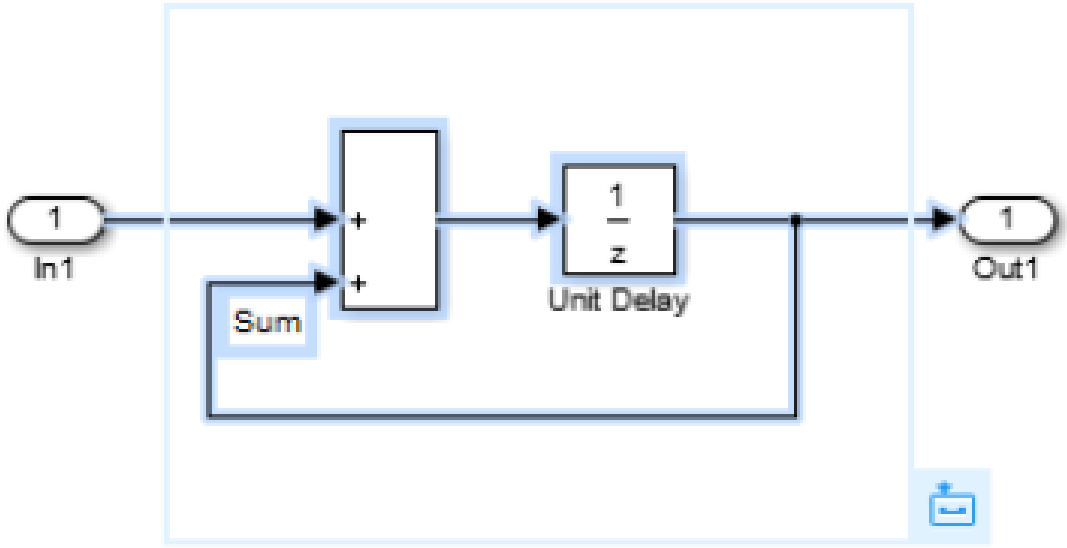

Figure 3. Selecting multiple blocks using a bounding box.

Procedure: Create a Subsystem Using Context Options

You can create a subsystem from the selected blocks by using the on-canvas context menu options available in the model:

1. **User action:** Drag a box around the blocks you want to include.
   System response: A blue context option appears.
2. **User action:** Hover over the context option.
   System response: The Create Subsystem toolstrip appears.
3. **User action:** Select the type of subsystem to create.
   System response: A Subsystem block encloses the selected blocks.

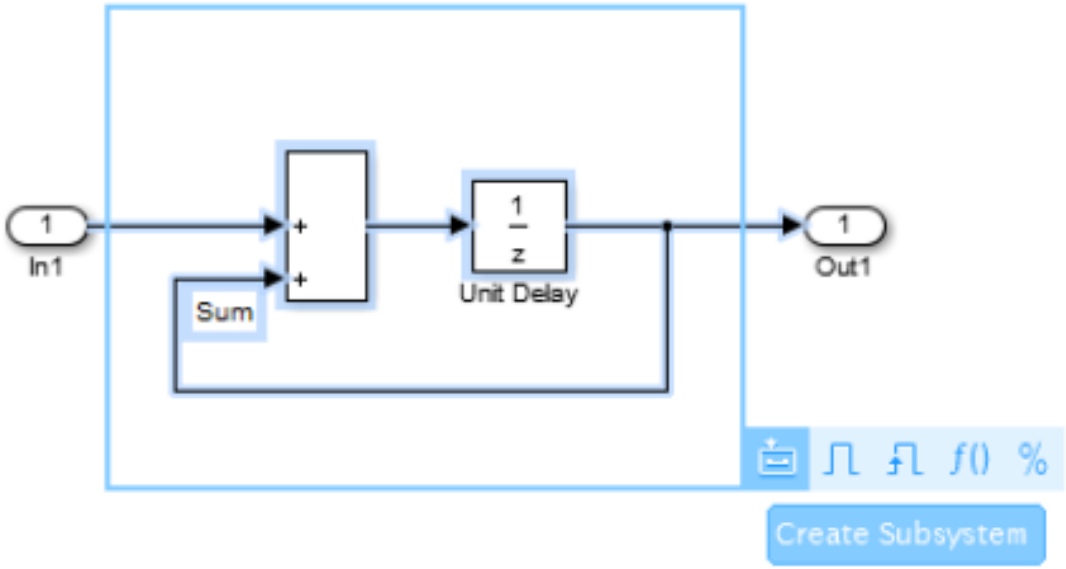

Figure 4. Creating a subsystem using the on-canvas context option.

Enabled Subsystem

Overview

An enabled subsystem executes while its enable control signal is positive. Execution starts when the enable signal crosses zero from negative to positive and continues as long as the signal remains positive.

Procedure: Create an Enabled Subsystem

Create an enabled subsystem when you want execution to occur only while an enable signal is positive.

1. **User action:** Create or select a subsystem.
   System response: The subsystem is selected.
2. **User action:** Add an **Enable block** to the subsystem.
   System response: The subsystem gains an enable control input.
3. **User action:** Connect a control signal to the enable input.
   System response: The enable input is now connected to the control signal.

Triggered Subsystem

Concept

A triggered subsystem executes once for each trigger event that occurs. A trigger event can occur on a rising edge, a falling edge, or either edge of the trigger signal. The trigger signal can be continuous or discrete.

A trigger event can occur on the following control signals:

- **A rising trigger** executes the subsystem when the control signal rises from a negative or zero value to a positive value, or to zero if the initial value is negative.
- **A falling trigger** executes the subsystem when a falling trigger event occurs.
- **Either trigger** executes the subsystem when the control signal either rises or falls.

Procedure: Create a Triggered Subsystem

You can add logic at signal transitions by using a triggered subsystem. To do so:

1. **User action:** Create or select a subsystem.
   System response: The subsystem is available for configuration.
2. **User action:** Add a Trigger block to the subsystem.
   System response: The subsystem gains a trigger control input.
3. **User action:** Configure the trigger type as rising, falling, or either.
   System response: The subsystem executes when the trigger event occurs.

Triggered and Enabled Subsystem

Concept

A triggered and enabled subsystem executes when a trigger event occurs, but only if the enable control signal is positive at the same time step.

Control Flow Subsystem

Concept

A control flow subsystem executes when enabled by a control flow block. Control flow blocks implement logical conditions similar to programming language statements and execute one or more times.

> **Note:** To improve simulation performance, Simulink avoids unnecessary execution of blocks connected to Switch blocks, Multiport Switch blocks, and conditionally executed subsystems by default. This optimization is known as **conditional execution (CE) behaviour**. You can disable this behaviour for all Switch and Multiport Switch blocks in a model or for specific conditional subsystems.

Procedure: Create a control flow in your subsystems

You can configure a control flow block with various conditions and manage the control flow of your subsystem:

1. **User action:** Provide data inputs to the If block.
   System response: The *If* block receives the input signals used to evaluate conditions.
2. **User action:** Open the ***If* block parameters** dialog box and define the input variables.
   System response: The inputs appear internally as variables named *u1, u2, ... un*.
3. **User action:** Open the ***If* block parameters** dialog box, and define the output conditions.
   System response: The *If* block creates output ports corresponding to the defined conditions.
4. **User action:** Specify expressions for the *if*, *elseif*, and optional *else* condition fields by using the input variables *u1, u2, ... un*.
   System response: The *If* block evaluates the expressions based on the input values.
5. **User action:** Enter one or more *elseif* conditions and select the **check box** to enable the *else* condition.
   System response: The *If* block adds an output port for the defined condition.
6. **User action:** Create a subsystem for each condition and add an **Action Port block** to each subsystem.
   System response: Each subsystem becomes an atomic **Action subsystem** with an **Action port**.
7. **User action:** Connect each condition output port on the *If* block to the corresponding **Action subsystem**.
   System response: Each **Action subsystem** is associated with its connected condition.

8. **User action:** Review the connected subsystems.
   System response: Each subsystem takes on the identity of its connected condition and behaves like an enabled subsystem.

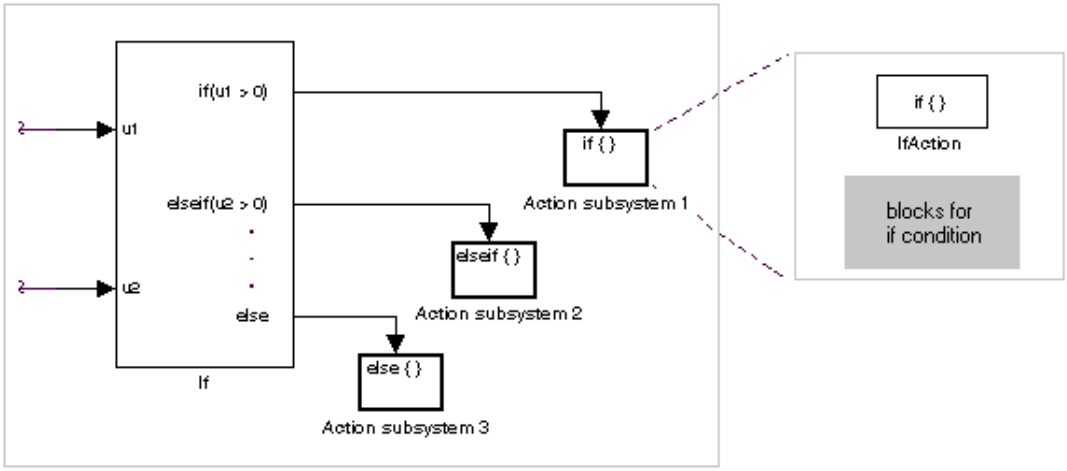

Figure 5. If-else control flow implemented using Action subsystems.

Questions for the Developer

The following questions can help me identify gaps:

- Which subsystem types should be emphasized for new users?
- Should trigger configuration be documented as a separate procedure?
- When should conditional execution behaviour be disabled in practice?
- How does execution precedence work when multiple conditional subsystems execute within the same model step?
- How does a falling trigger determine execution, and are there any differences in behaviour compared to a rising trigger?
- How do users create a Triggered and Enabled Subsystem in the Simulink UI, and what steps or configurations are required beyond adding trigger and enable signals?
- Are there recommended scenarios or design guidelines for users to choose the most appropriate subsystem for their use case?