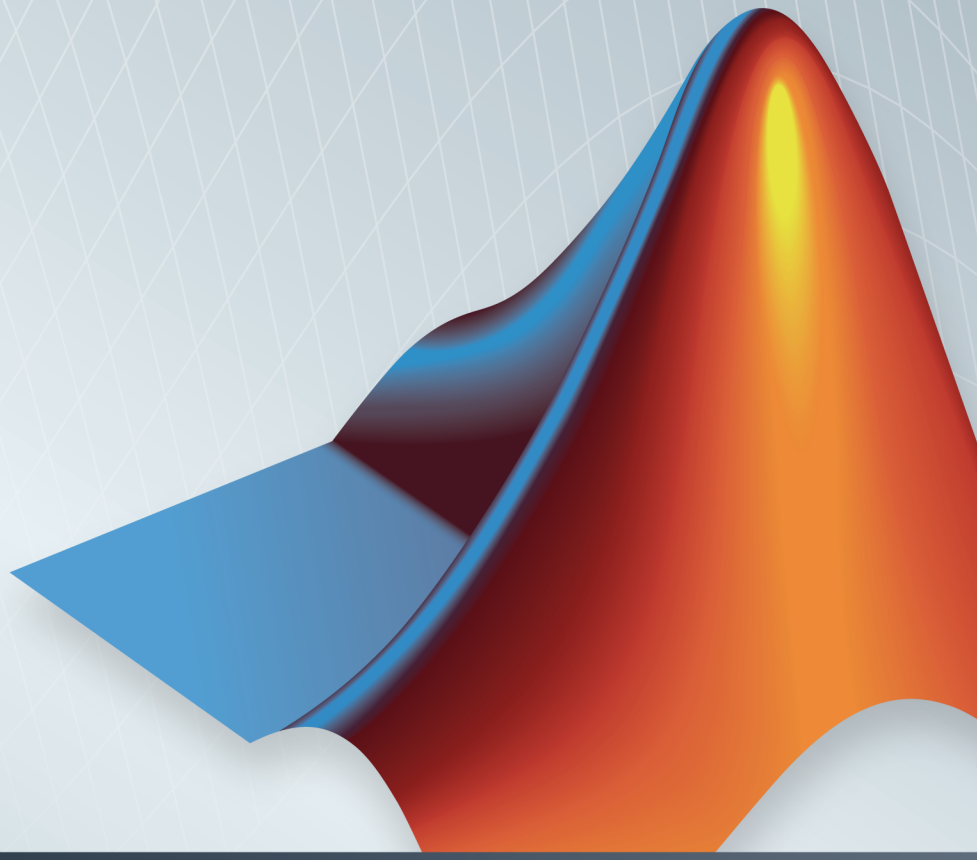
Gauges Blockset™

User's Guide

R2014**b**



MATLAB® & SIMULINK®

MathWorks®

How to Contact MathWorks

Latest news: `www.mathworks.com`

Sales and services: `www.mathworks.com/sales_and_services`

User community: `www.mathworks.com/matlabcentral`

Technical support: `www.mathworks.com/support/contact_us`

Phone: 508-647-7000

The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098

Gauges Blockset™ User's Guide

Revision History

September 1999	Online only	New for Version 1.0
September 2000	First printing	Revised for Version 1.1 (Release 12)
May 2001	Online only	Revised for Version 1.1.1 (Release 12.1)
July 2002	Second printing	Revised for Version 1.1.2 (Release 13)
June 2004	Online only	Revised for Version 1.2 (Release 14)
October 2004	Third printing	Revised for Version 2.0 (Release 14SP1) (New title)
March 2005	Online only	Revised for Version 2.0.1 (Release 14SP2)
September 2005	Online only	Revised for Version 2.0.2 (Release 14SP3)
March 2006	Online only	Revised for Version 2.0.3 (Release 2006a)
September 2006	Online only	Revised for Version 2.0.4 (Release 2006b)
March 2007	Online only	Revised for Version 2.0.5 (Release 2007a)
September 2007	Online only	Revised for Version 2.0.5 (Release 2007b)
March 2008	Online only	Revised for Version 2.0.5 (Release 2008a)
October 2008	Online only	Revised for Version 2.0.5 (Release 2008b)
March 2009	Online only	Revised for Version 2.0.5 (Release 2009a)
September 2009	Online only	Revised for Version 2.0.5 (Release 2009b)
March 2010	Online only	Revised for Version 2.0.5 (Release 2010a)
September 2010	Online only	Revised for Version 2.0.5 (Release 2010b)
April 2011	Online only	Revised for Version 2.0.5 (Release 2011a)
September 2011	Online only	Revised for Version 2.0.6 (Release 2011b)
March 2012	Online only	Revised for Version 2.0.6 (Release 2012a)
September 2012	Online only	Revised for Version 2.0.6 (Release 2012b)
March 2013	Online only	Rereleased for Version 2.0.6 (Release 2013a)
September 2013	Online only	Revised for Version 2.0.8 (Release 2013b)
March 2014	Online only	Revised for Version 2.0.9 (Release 2014a)
October 2014	Online only	Revised for Version 2.0.9 (Release 2014b)

Contents

1 Getting Started

2 Using Gauges in a Model

3 Categories of Controls

5 Block Reference

1

Getting Started

Gauges Blockset Product Description

Monitor signals with graphical instruments

Gauges Blockset lets you add graphical instrumentation to your Simulink® models. You can create realistic-looking displays that are customized for your model and visually representative of the environment that you are modeling.

Key Features

- Provides a library of customizable graphical instrumentation blocks to view signals
- Interfaces your Simulink model with any ActiveX control indicator
- Enables signal visualization in Simulink for real-time applications
- Includes an automotive instrumentation library

Related Products

In this section...

For information about related products, see `http://www.mathworks.com/products/gauges/related.html`.

External Mode Support

Support for external mode enables you to incorporate gauges into any target that you can connect to through external mode (such as the Simulink Real-Time™ and Real-Time Windows Target™ products; see the documentation for those products for details).

For more information about external mode, see “Host/Target Communication” in the Simulink Coder™ documentation.

Simulink Coder Product Support

You can use the Simulink Coder software to generate code from models that include blocks from the Gauges Blockset library.

Gauges are ignored during code generation, except through the use of external mode (see above). If you want to view the gauges, you can do so through the Simulink Coder external mode.

Installation

In this section...

Installation Requirements

The Gauges Blockset software requires the MATLAB® and Simulink products. It uses Component Object Model (COM) technology and runs only on Microsoft Windows® platforms.

Installing and Confirming Installation

To build and run the models in this manual, you must first install the Simulink and Gauges Blockset products. You can find instructions for installing these products in the installation documentation.

To determine what products are installed on your system, enter `ver` in the MATLAB Command Window.

Troubleshooting the Installation

Normally, the installation process automatically registers the Microsoft ActiveX controls associated with Gauges Blockset software. However, in exceptional cases you might see an error message referring to an `.ocx` component, similar to the following message:

```
Copying Gauges Blockset files
ads.ocx self registering file did not register
```

If you see such a message, or if the graphical icons do not appear on the blocks in this blockset, then do the following:

1 In the Windows **Start** menu, right-click **Command Prompt** and select `run as administrator`.

   You must have administrative privileges to use this menu item.

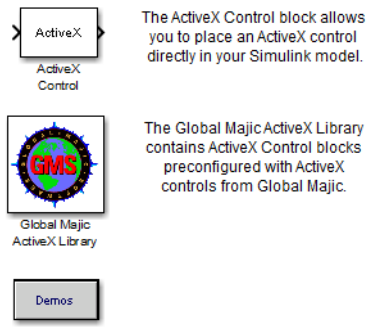
2. In the Windows command window, type `matlab -win32`.
3. In the MATLAB Command Window, type `gauges_register_ocx`.

Accessing the Preconfigured Blocks

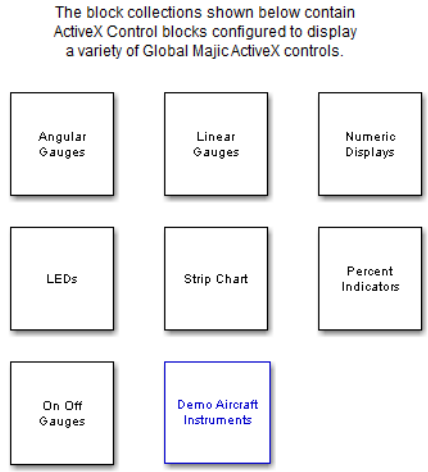
In this section...

Using the gaugeslib Command

1 Enter the `gaugeslib` command in the MATLAB Command Window. The following window opens.



2 Double-click the Global Majic ActiveX Library icon to access the libraries it contains.



3 Double-click an icon to access the blocks in the library that the icon represents. If all the blocks say `ActiveX` and do not look like graphical displays, then follow the instructions in "Troubleshooting the Installation" on page 1-4. Each library also includes a question-mark block that provides access to online help for the controls in that library.

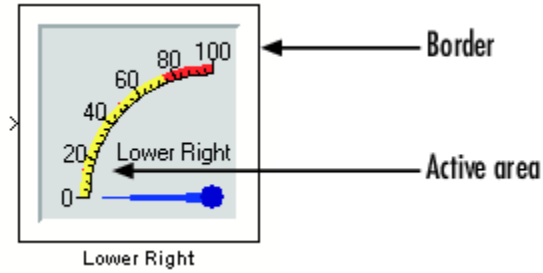
Gauges Blockset software is extensible. A general-purpose block is provided in which you can place your custom-built controls.

Using the Simulink Library Browser

As an alternative to the `gaugeslib` command, you can use the Simulink Library Browser to access the preconfigured blocks. For details, see "Simulink Library Browser" in the Simulink documentation.

Moving and Selecting ActiveX Control Blocks

The way you move and select blocks in the Gauges Blockset library differs from how you move and select a block in the Simulink library. Blocks in the Gauges Blockset library consist of an “active” area containing the actual Microsoft ActiveX control, and a border surrounding the active area. The border gives you a way to manipulate the block without affecting the control.



The table below describes how to manipulate a Gauges Blockset block.

Task	Mouse Action
Add block to model	From the Simulink Library Browser, drag the block by its icon in the right pane.
	From the library window (displaying blocks as icons), drag the block by its border.
Move block	Drag the block's border. You can do this only if the border is visible.
Select block	Click the block's border. Or “rubber-band select” the block.
Resize block	Select the block, and then drag one of the selection handles (as you would resize a Simulink block).

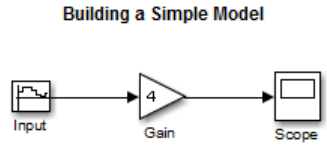
Building an Example Model

This section illustrates how to build and use a simple system, first using Simulink blocks alone, and then using a Gauges Blockset block. By building the latter model, you can practice finding and using a Gauges Blockset block. By comparing the two models, you can get a better sense of how graphical icons might enhance the look, feel, and usability of your own models.

In this section...

The Original Simulink Model

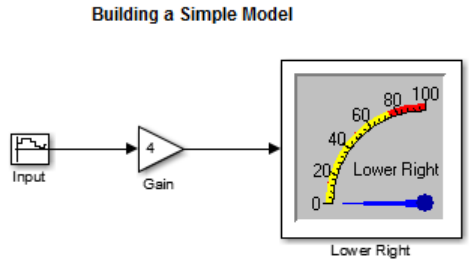
Consider a system in which a Sine Wave block feeds into a Gain block, while a Scope block displays the output from the Gain block. These blocks are in the Simulink library.



If you simulate this system and double-click the Scope block, then the Scope traces the value of its input signal over time.

Replacing Simulink Blocks with Gauges

You can replace the Scope block from the Simulink library with a realistic-looking display. For example, a Lower Right block can display the sine wave value at each instant during the simulation.



Building the Model

To build the model described earlier, follow the steps below. Alternatively, enter `gauges_simple` in the MATLAB Command Window to open a completed copy of the model.

1. Open the Simulink Library Browser and create a new model window. For details on how to do this, see "Start the Simulink Software".
2. From the Angular Gauges library, drag the Lower Right block into the model.
3. From the Simulink Sources library, drag the Sine Wave block into the model window.
4. Double-click the Sine Wave block and set the block's parameters as follows:
   - **Sine type** — `Time based`
   - **Time (t)** — `Use simulation time`
   - **Amplitude** — `10`
   - **Bias** — `12.5`
   - **Frequency (rad/sec)** — `0.1`
   - **Phase (rad)** — `0`
   - **Sample time** — `0.1`
   - **Interpret vector parameters as 1–D** — `on`
5. From the Simulink Math Operations library, drag the Gain block into the model window.
6. Double-click the Gain block and change the **Gain** parameter to `4`.
7. Draw connection lines from the Sine Wave block to the Gain block, and from the Gain block to the Lower Right block.

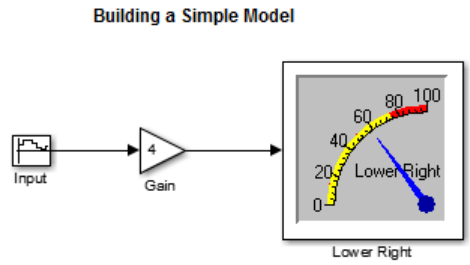
8 From the model window's **Simulation** menu, choose **Model Configuration Parameters**.

9 In the **Simulation time** section of the Solver pane, set the **Stop time** parameter to `Inf`. In the **Solver options** section of the Solver pane, use the **Solver** menu to select **discrete (no continuous states)** since this model does not have continuous states.

Now you can run the model and watch how the sine wave affects the needle on the Lower Right block.

Running the Simulation

Run the simulation by choosing **Run** from the model window's **Simulation** menu. While the simulation is running, you can observe results on the Lower Right block. This figure shows the model after the needle of the Lower Right block is displaced from its default position.



To stop the simulation, choose **Stop** from the model window's **Simulation** menu.

Saving the Model

Save the model by choosing **Save** from the model window's **File** menu. When you save a model that contains blocks from the Gauges Blockset library, the file automatically reflects the current state of the control that is embedded in the block.

Printing the Model

You can print the structure of the model by choosing **Print** from the model window's **File** menu. Or you can capture the same image in a `.bmp` file and print the file by entering either of the following commands in the MATLAB Command Window.

```
print -smodelname -dbitmap filename

print(['-s','modelname'],'-dbitmap','filename')
```

Here, `modelname` and `filename` list the names of the Simulink model and the bitmap file, respectively. For example, if the open model is called `sample`, then this command saves it in a file called `samplepic.bmp`.

```
print -ssample -dbitmap samplepic
```

After the application creates the bitmap file, you can insert it into an application that can print it.

Note: Note that the printing functionality in Simulink software does not print the active areas of Gauges Blockset blocks. Instead, it shows only the outline of those blocks.

Modifying Properties of Blocks

In this section...

Accessing the Properties

You can view properties of the controls by using one of these procedures:

- Double-click the active area of the block that contains the control.
- Right-click the active area of the block and select the **Control Display Properties** option.

After you select the **Control Display Properties** option, the ActiveX Control Properties dialog box appears. This dialog box enables you to modify properties of the control.

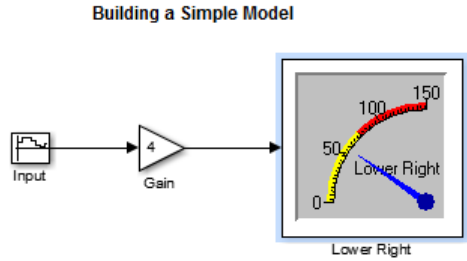
If you modify values in this dialog box, then the block is visually updated immediately. However, the changes are not permanent until you choose **OK** or **Apply**; if you choose **Cancel**, then the changes will be undone.

Example of Modifying Properties

Returning to the model that you built in the section "Building an Example Model" on page 1-9, you can modify the range of output values by modifying the properties of the Lower Right block. For example, the instructions below change the maximum needle value from 100 to 150.

1. Open the ActiveX Control Properties dialog box by double-clicking the active area of the Lower Right block.
2. Display the panel that controls the scaling of values by clicking the **Scales** tab.
3. Set the **Max** parameter to 150.
4. Display the panel that controls tick marks by clicking the **Ticks** tab.
5. Set the **TickCount (Major Ticks)** parameter to 3. This prevents the block from looking too crowded.

The resulting model looks like this:



Learning More About Properties

Gauges Blockset blocks have many properties. Changing the appearance of a block might require changing several properties and can be quite complex. “Modifying Properties of a Control” discusses how to make some common changes, such as changing the range of values displayed on a block.

For information about specific properties, consult the help for the control by double-clicking the question-mark block that appears in each sublibrary within the Gauges Blockset library. Some sublibraries provide more than one question-mark block when the blocks contained in that sublibrary are significantly different from each other. Once in the Help window, use the Properties link to display information about block properties.

Using Gauges in a Model

Connecting Blocks in a Model

Before you connect a Gauges Blockset block with other blocks, you should know whether it is meant to be an output device (with an input connection), or a pass-through device (with an input and output connection). Gauges Blockset blocks cannot serve as input devices (with only an output connection). Gauges Blockset blocks initially appear with both an inport and an outport, but unused ports disappear when you start the simulation or update the block diagram.

To determine whether a Gauges Blockset block is meant to be used as an output or pass-through device, right-click the block and select the **Block Parameters** option.

Note: If you customized the generic ActiveX Control block to accommodate your own control, then another way to display the custom block's Block Parameters dialog box is to double-click the border of the block.

In the Block Parameters dialog box, the **Connections** field determines the type of connection the block currently uses:

- `Input` indicates that the block is a sink; that is, it has an inport and receives a signal. The **Input property** parameter indicates the block's property whose value is changed by the input.
- `Both` indicates that the block has an inport and an outport. The values at both ports are the same. That is, the block becomes a unity function.
- `Neither` indicates that the block has neither an inport nor an outport.

To specify a connection different from the block's default setup, change the block's **Connection** setting and make sure that the **Input property** field is filled in with the required property name. See “Block Parameters for the ActiveX Control Block” for information about the other fields and check boxes. You can also use the **Help** button to find out about other parameters.

Modifying Properties of a Control

In this section...

Control Basics

You can modify many properties of a preconfigured Gauges Blockset block using its ActiveX Control Properties dialog box, introduced in "Modifying Properties of Blocks". This section discusses some of the more complicated tasks and concepts associated with the modification of properties.

For more information about individual properties of the preconfigured blocks, see the online help for the corresponding controls. To access such help, open the library window and double-click the question-mark block. The online help summarizes the functionality and contains links to information about properties, events, and methods.

Using Multiple Styles Within One Block

Some control properties let you use more than one style for a given component or characteristic, in the same block. For example, you might use multiple styles to create

- Different font characteristics for text in different places
- Multiple colors within a graphical element such as an annular region or a divided pie chart
- Multiple sets of ticks, each with its own size or labeling characteristics
- Multiple components, such as LEDs or needles, each with its own characteristics

These sections discuss the use of multiple styles in preconfigured Gauges Blockset blocks:

- "Blocks That Use Multiple Styles by Default" on page 2-4

Blocks That Use Multiple Styles by Default

Many Gauges Blockset blocks include multiple styles by default:

- The Vacuum block in the Angular Gauges library uses three text styles: one for the tick labels, one for the number at the bottom of the gauge, and one for the text near the center of the gauge.
- The Volume block in the Angular Gauges library uses three adjacent annular regions, each with a different color.
- The Thermometer block in the Linear Gauges library uses two styles for ticks: one for numbered ticks every 10 degrees and another for unnumbered ticks every 2 degrees.
- The Circle Meter block in the LEDs library applies one of three LED styles to each of 10 LEDs. The three styles differ in their colors.

Determining When Multiple Styles Are Allowed

If a component supports multiple styles, then its property dialog box has properties that enable you to set the number of styles and refer to the styles by number. For instance, in the **Fonts** tab of the `Lower Right` Angular Gauge, the **Fonts** property indicates the number of font styles, while the **FontID** property refers to a given style by number.

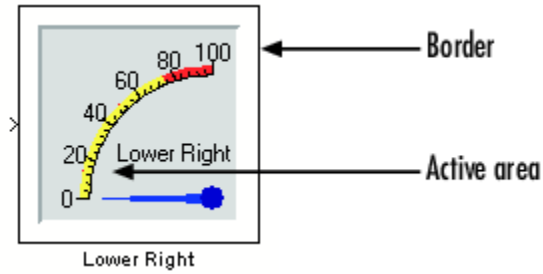
To determine whether a component supports multiple styles, look in the block's property dialog box for a pair of properties whose names are

- A plural noun describing the component, such as **Fonts** or **Scales**
- A word that combines the noun and the letters "ID," such as **FontID** or **ScaleID**

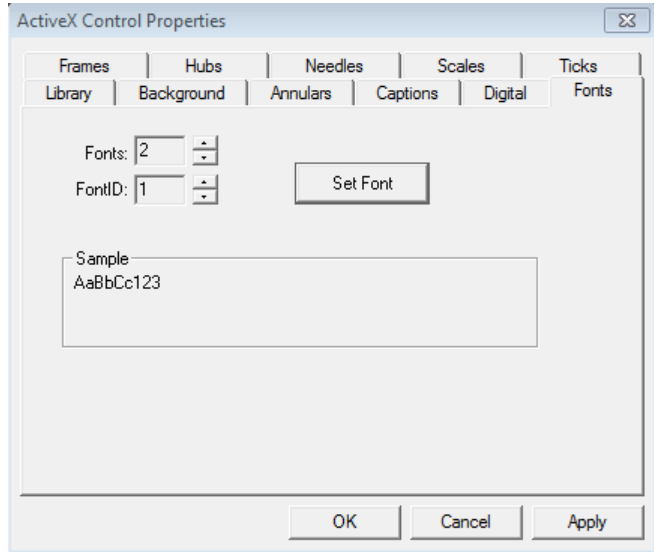
If the dialog box has no such properties for the component, you cannot create multiple styles for that component. For example, in the **Background** panel of a block's dialog box, you can define the color of an outline, but you cannot create multiple concentric outlines of different colors.

Creating Styles

After locating the style-identifying pair of properties for the component you are interested in, follow these steps to create an additional style. This example uses the `Lower Right` Angular Gauge control.



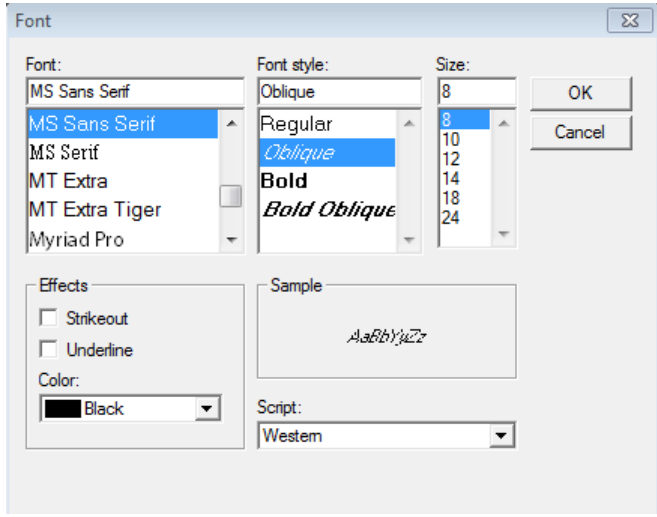
1. Right-click the active area of the `Lower Right` Angular Gauge control, and then click **Control Display Properties**.
2. Click the **Fonts** tab.
3. Click once on the up arrow next to the value of the first property in the pair (**Fonts**). This value is the number of defined styles. If *N* styles are defined, then each is associated with an integer between 0 and *N*–1. The corresponding ID property (**FontID**) can assume values between 0 and *N*–1.
4. Click repeatedly on the up arrow next to **FontID** to set it to its maximum value. This causes the dialog box panel to reflect the attributes of that particular style instead of the other defined styles.



5 Click the **Set Font** button.

6 Select **Font style** `Oblique`.

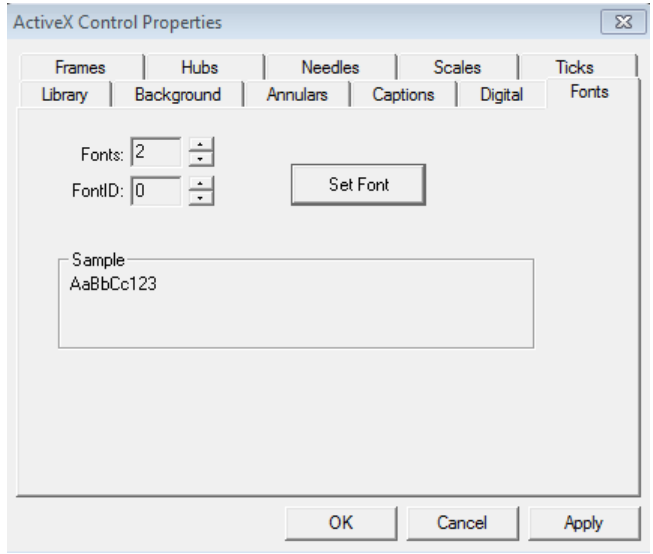
The **Sample** box turns to the Oblique style. The dialog box looks like this:



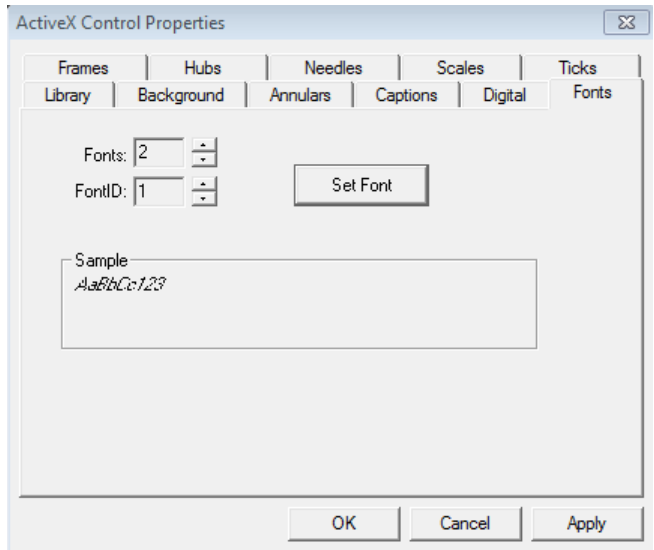
7 Configure the other properties in the dialog box as required. In many cases, all properties in the panel except the original style-identifying pair are attributes of the style. In a few cases, only part of the panel contains attributes of the style and others are global attributes that apply to all styles.

To view attributes of an existing style, set the ID property to the integer associated with that style. Then, properties on the dialog box panel other than the style-identifying pair reflect attributes of that style.

If you set **FontID** to `0`, the **Sample** area of the **Fonts** panel displays a font of medium size and weight.



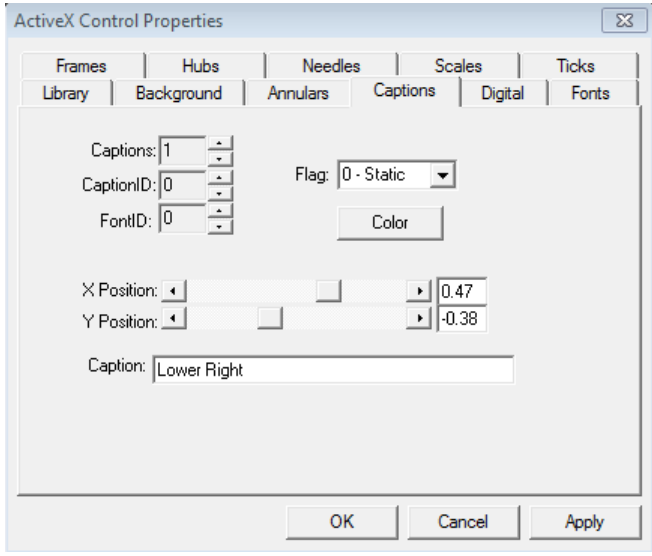
If you set it to 1, the **Sample** area displays an oblique font.



Applying Styles

In some cases, creating a style implicitly causes the block to apply it. For example, creating an additional style for tick marks automatically creates an additional set of tick marks on the block. In other cases, creating a style does not implicitly cause the block to apply it. For example, even after you create an additional font style, you will not see a change until you indicate which text should use that style. This section describes how to apply styles that the block does not apply immediately after you create them.

To determine where you can apply a style you have created, look for the corresponding ID property on a panel of the dialog box *other than* the panel where you defined the style. For example, the **Captions** panel contains the **FontID** property but not a **Fonts** property, indicating that in this panel you can apply font styles but not define them.



In this case, for **CaptionID** 1,**FontID** 0 is in use, specifying a captions font of medium size and weight. If you defined **FontID** 1 as an oblique font, setting **FontID** to 1 changes the caption to oblique font.

Understanding ID Properties

Many blocks have properties whose names end with **ID**, such as **FontID**, **ScaleID**, and **NeedleID**. Such properties enable you to use more than one style in the same

block, as in the situations listed in “Using Multiple Styles Within One Block” on page 2-3. This section describes how to interpret ID property settings. For an example that examines ID property settings among a block's default settings, see “Modifying Multiple Tick Marks” on page 2-15.

The value of an ID property refers to a style by number. To determine the purpose of the ID property, first see whether the property directly above it is a plural noun similar to the ID property's name. (For example, see whether the property directly above **FontID** is **Fonts**.) Then,

- If the property directly above the ID property is a plural noun similar to the ID property's name, then this panel of the dialog box *defines* a set of styles. The ID property associates a number with each style. Other properties in the dialog box panel reflect the definition of the style whose number is the current value of the ID property. By changing the value of the ID property, you can view the definition of a different style.

  For example, in the **Fonts** panel of the Volume block, the **FontID** property occurs directly underneath a **Fonts** property. This panel of the dialog box defines font styles, and the **Sample** box displays text using the font style whose number is the current value of the **FontID** property.

  Note: If you decrease the value of the property named by the plural noun (for example, the **Fonts** property), then the style corresponding to the highest ID value is removed. To replace that style, you have to add a new style and recreate the settings of the deleted style from the default settings.

- If the property directly above the ID property is *not* a plural noun similar to the ID property's name, then the ID property *applies* a style that was previously defined in another panel of the dialog box. Other properties in the dialog box panel indicate the context in which the style is applied. By changing the value of the ID property, you can select a different style to apply.

  For example, in the **Captions** panel of the Volume block, the **FontID** property does not occur directly underneath a **Fonts** property. The purpose of the **FontID** property in this case is to reference previously defined font styles and apply them to captions. (The font styles are defined on the **Fonts** panel of the dialog box.)

Sometimes, multiple styles are combined so seamlessly that it is not obvious why to use different styles or which parts of the block correspond to which style definitions. You can often adjust the definition of the style to make the style usage more apparent.

For example, if you change the colors of different annular regions and then look for the corresponding change in the block, then you should be able to determine how the design is split among multiple annular regions.

Displaying Text on a Block

Many blocks enable you to include text on the block. Such text might describe the quantity being measured, the units of measurement, or other information. The table below lists some types of text that are associated with a specific part of the block, as well as the part of the ActiveX Control Properties dialog box panel that defines the text. Some types of text apply only to certain blocks.

Type of Text	Part of Dialog Box That Defines or Enables Text
Title appearing in block's outline	**Title** property on **Background** panel
Numerical labels near tick marks	**Labels** area on **Ticks** panel. On Strip Chart block, **Labels** properties on **Tracks** and **X Axis** panels.
Numerical labels near pointer or needle	**Digital** panel
Captions appearing anywhere on block	**Captions** panel

Using the Captions Panel to Display Text

When it is present, the **Captions** panel of the ActiveX Control Properties dialog box enables you to place text anywhere on the block. Blocks that use text captions by default include Mixer Scale, Tank, Thermometer, Amp Meter, and Volume. This section describes how to add, remove, and change characteristics of text captions using the **Captions** panel.

Adding Text Captions

To create a new text caption, follow these steps:

1. Increase the value of the **Captions** property by one.
2. Set **CaptionID** to its maximum value. This is the index that corresponds to the newest text caption.
3. Type the desired text in the **Caption** edit field.

Removing Text Captions

To remove the most recently added text caption (that is, the one with the largest **CaptionID** value), decrease the value of the **Captions** property by one. Note that this removes all characteristics of that text caption.

Changing Fonts and Other Characteristics of Text Captions

To change the font of an existing text caption, you must create a numbered font style and then apply that style to the caption. Follow these steps:

1. Open the **Fonts** panel of the dialog box.
2. Allocate space for a new font style by increasing the value of the **Fonts** property by one.
3. Set **FontID** to its maximum value. This is the index that corresponds to the newest font style.
4. Use the **Set Font** button to select font characteristics.
5. Open the **Captions** panel of the dialog box.
6. Set **CaptionID** to the index that corresponds to the text caption whose font you want to change.
7. Apply the font style to the caption by setting **FontID** to the font style's index.

To change other characteristics of an existing text caption, first set the **CaptionID** property on the **Captions** panel to the value that corresponds to the text caption you want to change. Then use other properties on the dialog box panel, *except* the **Captions** counter, to configure the text caption accordingly.

Note: For text captions, the color choice on the **Captions** dialog box panel overrides the color choice on the **Fonts** dialog box panel.

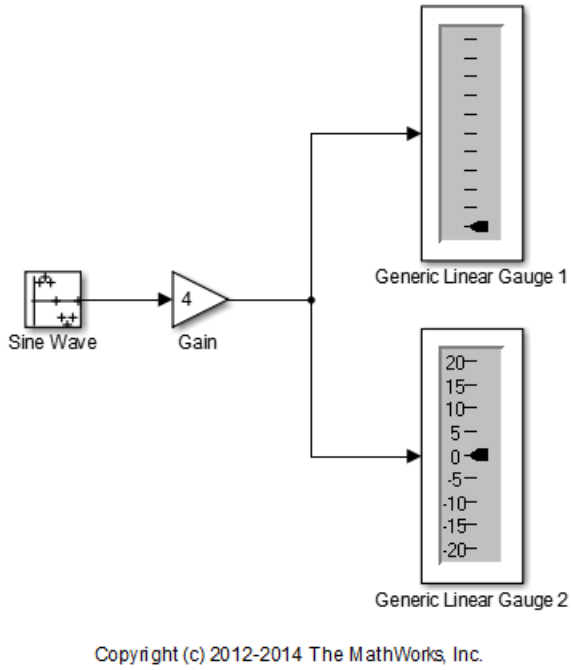
Modifying the Displayed Range

Changing the range of values displayed on a block involves adjusting these properties:

- Scale properties define the extent of the units displayed by the block, the location of the block's center, and the block's start and stop positions.
- Tick-mark properties define tick marks on the block, including start and stop values, the interval between tick marks, and label positions.

- Needle or pointer properties indicate the value.

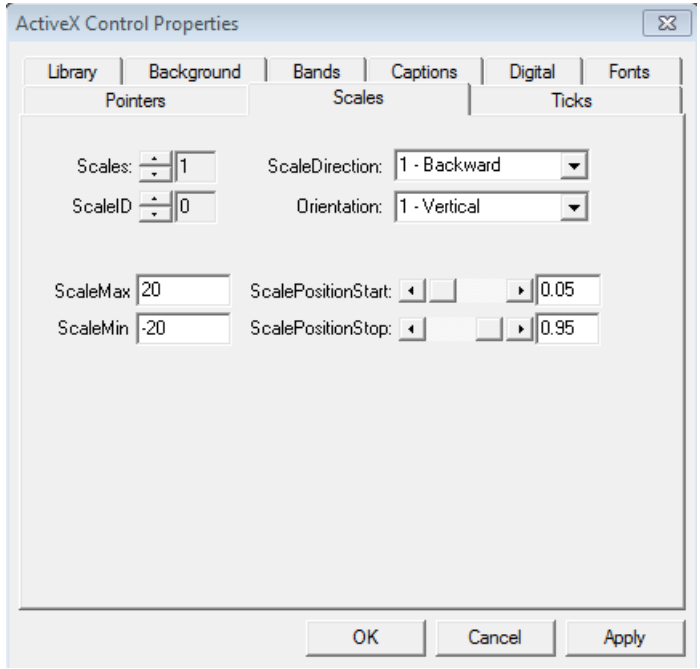
The following example shows how to change the Generic Linear Gauge to display values from `-20` to `20`, sets the interval between tick marks to `5`, and shows the tick-mark labels. `General Linear Gauge 1` shows the Generic Linear Gauge with its default settings; `General Linear Gauge 2` shows it with modified settings.



To modify the `General Linear Gauge 1` settings to duplicate the `General Linear Gauge 2` display:

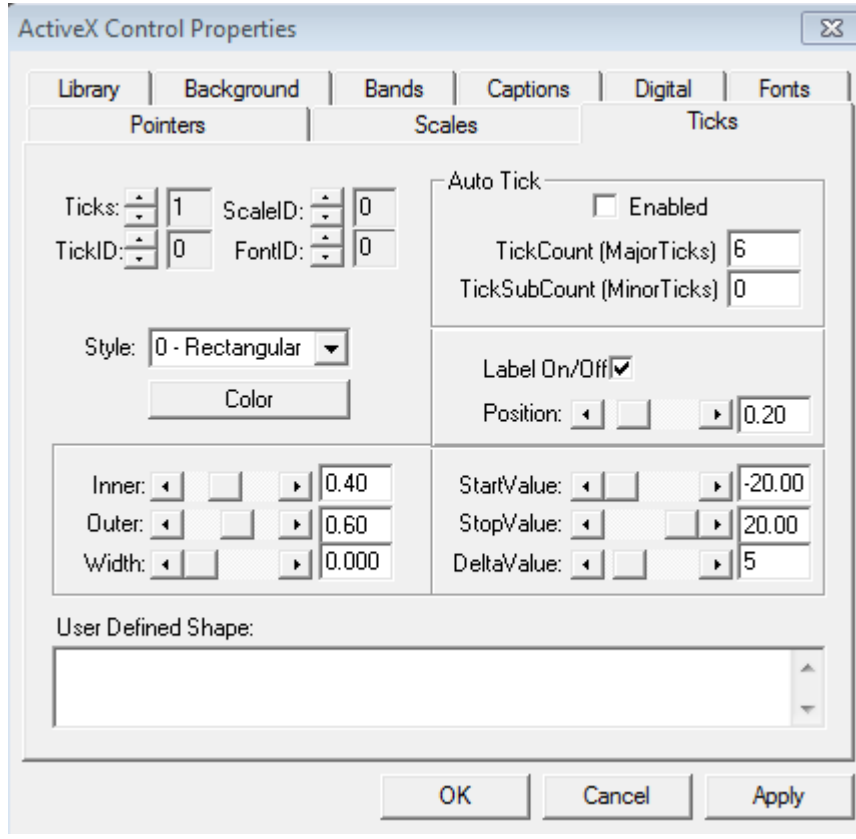
1. Double-click in the active area of `General Linear Gauge 1`.
2. Click the **Scales** tab to display the scales properties page.
3. To modify the scale range, change **ScaleMax** in the lower left corner to `20` and **ScaleMin** to `-20`.
4. Click **Apply**.

The dialog box looks like this:



5. Click the **Ticks** tab to display the tick-mark properties page.
6. To show tick-mark labels, check the **Label On/Off** check box in the lower right corner.
7. To set the starting and ending tick marks so they mark the minimum and maximum scale settings, set **StartValue** to `-20` and **StopValue** to `20`.
8. To set the spacing between tick marks, change the **DeltaValue** property. A value of `5` is reasonable for default block size.
9. Click **Apply**.

The dialog box looks like this:

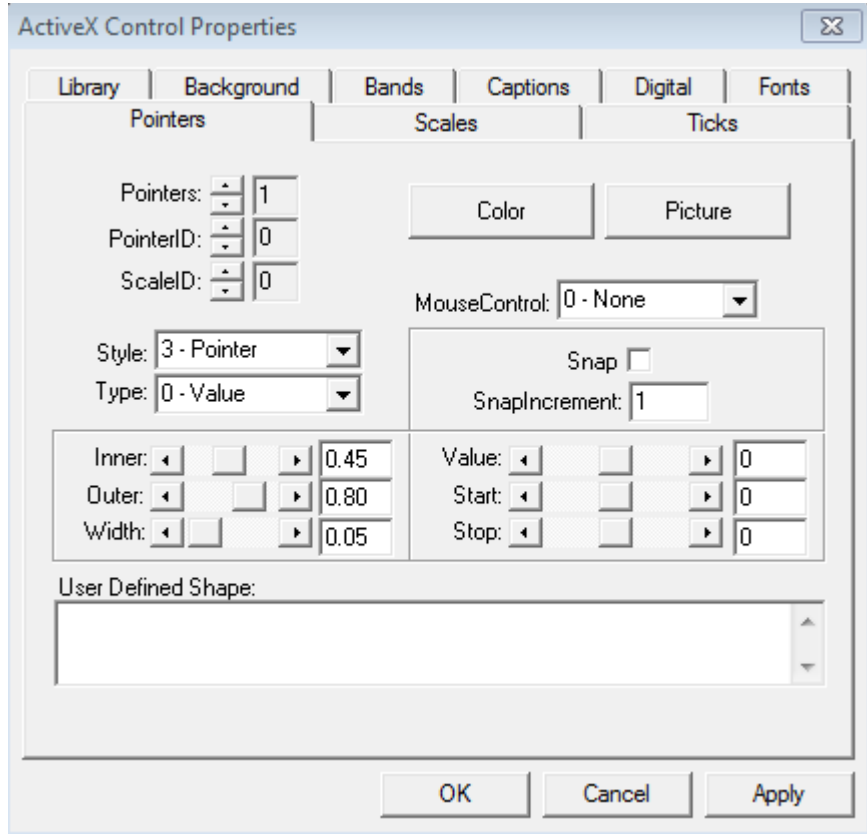


10 Click the **Pointers** tab to display the pointer properties page.

11 Set the **Value** property in the lower right corner to 0, halfway between the maximum and minimum scale values. This is the current pointer value.

12 Click **Apply**.

The dialog box looks like this:



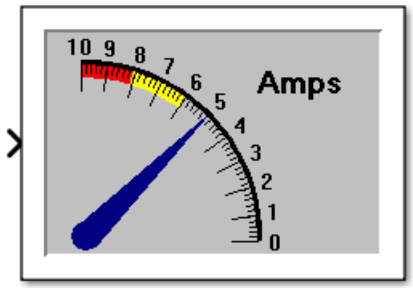
13 Click **OK**.

14 Save your model.

Modifying Multiple Tick Marks

This example illustrates the use of multiple tick marks and the use of the ID property to manage them. Instead of modifying a block, this example examines the default settings for a particular block.

This figure shows the Amp Meter block. Notice that the tick marks have two different lengths. These are created by defining two sets of tick marks.

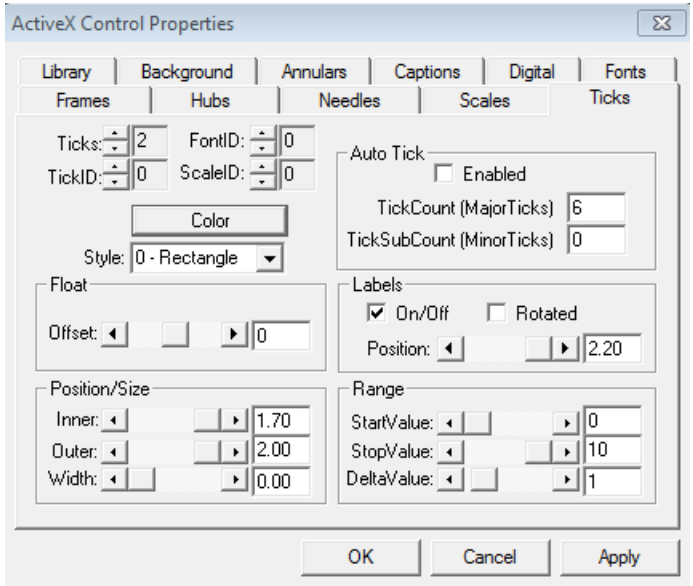


Amp Meter

The first set consists of 11 longer tick marks, each positioned at one of the label values, positioned at increments of 1.0. The second set consists of five shorter tick marks for each integer change in the scale, positioned at increments of 0.2.

To examine how these tick marks were created, double-click the Amp Meter block border to display its properties dialog box. Select the **Ticks** tab.

The **Ticks** pane looks like this:



The **Ticks** and **TickID** properties, in the upper left corner, are defined as follows:

- The **Ticks** property specifies how many sets of tick marks are used by the block. For this block, this property is set to **2**.
- The **TickID** property indicates which set of tick marks is defined by the other properties on this page. When specifying the characteristics of a set of tick marks, you set the **TickID** property, and then define the property values for that set of tick marks. In the dialog box page above, the settings for all the properties on the page apply to the first set, identified as **TickID** 0.

Note: When you define multiple components, the first instance is identified by an ID of 0. In this example, the two sets of tick marks have IDs of 0 and 1.

The **Position/Size** properties, in the lower left corner, are defined as follows:

- The **Inner** property defines the edge of the tick mark closest to the needle center and the **Outer** property defines the edge of the tick mark farthest from the needle center. To see where the tick marks are located relative to the needle length, examine the needle length by selecting the **Needles** page. The needle length is `2.0`. The **Inner** position is `1.70` and the **Outer** position is `2.00`. These tick marks are `0.3` units long.

- The **Width** property of the tick marks is `0.00`, the narrowest width.

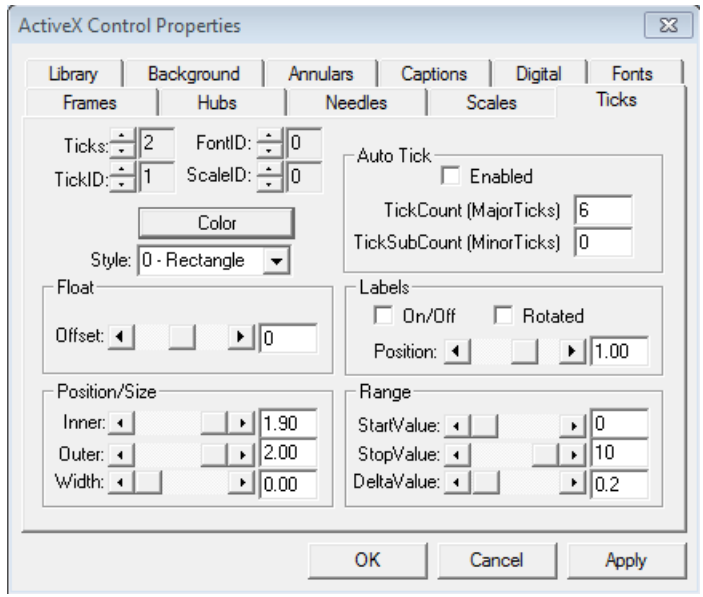
The **Range** properties, in the lower right corner, are defined as follows:

- **StartValue** determines at which scale value the first tick mark is displayed. For these tick marks, the value is `0`.
- **StopValue** determines at which scale value the last tick mark is displayed. For these tick marks, the value is `10`.
- **DeltaValue** determines the interval between tick marks. For these tick marks, the value is `1`.

The **Labels** properties **On/Off** check box, also in the lower right corner, determines whether the labels are displayed. For the first set of tick marks, the labels are displayed.

The **FontID** property, in the upper left corner, determines which of multiple fonts defined for this block is used for the label. In this case, two font sets are defined. The first (**FontID** `0`) is for the tick marks, while the second (**FontID** `1`) is for the caption, "Amps."

To examine the second set of tick marks, change the **TickID** property value to `1` by clicking the up arrow to the left of the value. The **Ticks** pane looks like this.



The **Position/Size** properties, in the lower left corner, are defined as follows:

- The **Inner** position is `1.90` and the **Outer** position is `2.00`. These tick marks are `0.10` units long, one-third the length of the longer tick marks.
- The **Width** property of the tick marks is `0.00`, the same as the longer tick marks.

The **Range** properties, in the lower right corner, are defined as follows.

- **StartValue** for these tick marks is `0`. The first short tick mark and the first long tick mark appear in the same place.
- **StopValue** for these tick marks is `10`. The last short tick mark and the last long tick mark appear in the same place.
- **DeltaValue** determines the interval between tick marks. For these tick marks, the value is `0.2`.

The **Labels** properties in the lower right corner **On/Off** check box determines whether labels appear next to the tick marks. No labels appear next to this set of tick marks.

If you decrease the **Ticks** property, then the tick-mark settings corresponding to the highest **TickID** value is removed. To replace that set of tick marks, you will have to recreate the settings from the defaults.

Controlling Multiple Graphical Elements

This section describes techniques for displaying multiple input values simultaneously and dynamically on a gauge block that includes multiple graphical elements. Examples of multiple-component gauge blocks include these preconfigured blocks:

- Multiple Scales in the Linear Gauges library
- Pie Chart and Dynamic Pie in the Percent Indicators library
- Analog Clock and Stop Watch in the Angular Gauges library

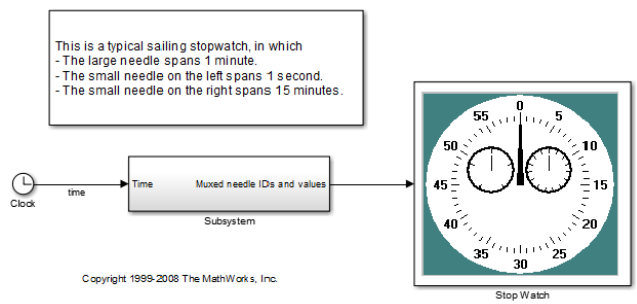
In this section...

Simulating a Multiple-Needle Stopwatch

This example model simulates a typical sailing stopwatch. The model uses the Stop Watch block, which displays three needles on individual angular scales. The needles mark simulation time simultaneously, as follows:

- The large needle spans 1 minute.
- The small needle on the left spans 1 second.
- The small needle on the right spans 15 minutes.

To open this example, enter gauges_stopwatch in the MATLAB Command Window. Run the simulation and watch how the three needles move.



These sections describe how the model works:

How NeedleID Values Correspond to Needles

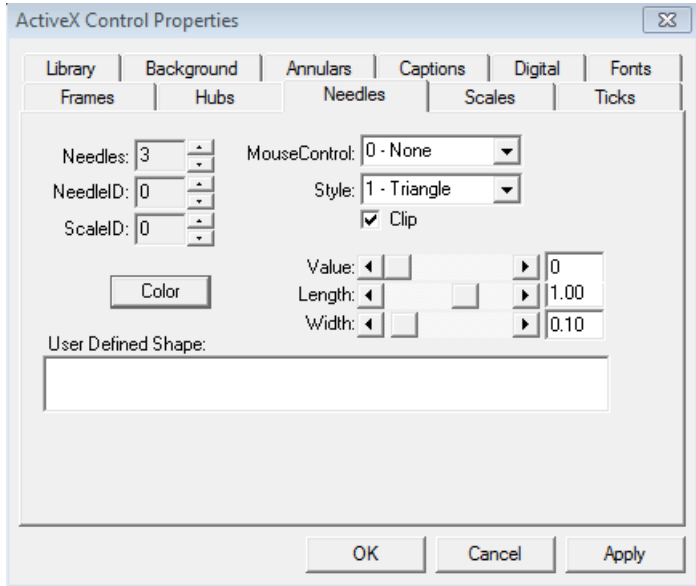
To explain how the model controls three needles, this section first explains how the block makes its needles accessible to you. Each needle on the block has an associated pair of **NeedleID** and **Value** parameters, where

- **NeedleID** is a number that distinguishes that needle from other needles on the block.
- **Value** is the numerical value for that needle along its angular scale.

You can view or control these parameters via the **Needles** panel of the block's ActiveX Control Properties dialog box. To find out which needle and range of values correspond to a given ID number, use this procedure:

1. Right-click the Stop Watch block and choose **Control Display Properties**.
2. Click the **Needles** tab.
3. Set **NeedleID** to the ID number you want to investigate, here 0, 1, or 2.

   The dialog box looks like this:



4 Change the **Value** parameter while watching which needle on the block moves. Drag the **Value** slider to its extremes to find out the minimum and maximum values for the needle:

NeedleID	Needle on Block	Range of Values
0	Large needle	[0, 60] seconds
1	Small needle on right	[0, 15] minutes
2	Small needle on left	[0, 5] fifths of a second

For more information about ID properties, see “Understanding ID Properties” on page 2-8.

Configuration of the Stop Watch Block

After you run the model, the number 6 appears near the connector line that represents the input to the Stop Watch block. This indicates that the signal on that line is a vector of length 6. To understand why, right-click the Stop Watch block and choose **Block Parameters**.

The **Input property** field in the dialog box is set to:

`NeedleID,NeedleValue,NeedleID,NeedleValue,NeedleID,NeedleValue`
By contrast, the default value for this property in the Angular Gauges library is `NeedleValue`.

The comma-separated list in this example model causes the Stop Watch block to use the six values in its vector input signal to assign these parameters to the block, in sequence:

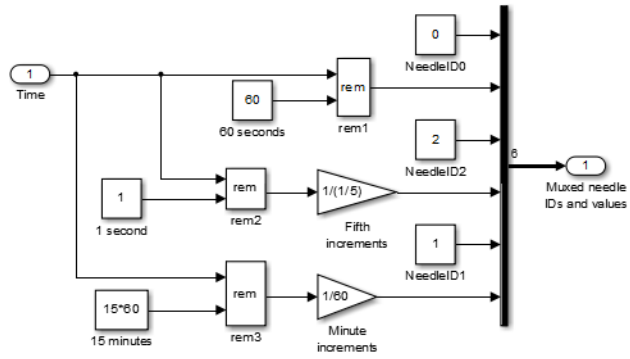
- The ID number, `0`, of the large needle
- The numerical value for the large needle
- The ID number, `2`, of the small needle on the left
- The numerical value for the small needle on the left
- The ID number, `1`, of the small needle on the right
- The numerical value for the small needle on the right

Note: It is important that each needle's ID number precede its numerical value. Referring first to the ID number tells the block which needle to apply changes to.

Preparing the Input Signal for the Stop Watch Block

The model aims to reflect the time on the Stop Watch block. However, the model has to process the Clock block's output to prepare it for the Stop Watch block. Double-click the Subsystem block to open it. The subsystem performs these key tasks:

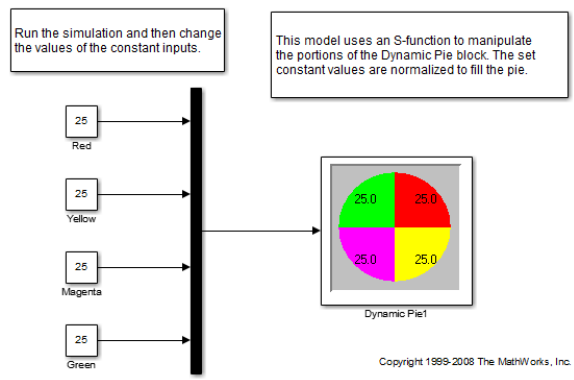
- Processes the Clock block's output value, in seconds, into numerical values suitable for the unit, scale, and range of each needle. To do this, the block performs these computations:
  - Reduces modulo 60, to get the large needle's value.
  - Reduces modulo 1 and then multiplies by 5, to get the left needle's value (ranging between 0 and 5).
  - Reduces modulo 15*60 and then divides by 60, to get the right needle's value (ranging between 0 and 15).
- Forming a six-element vector corresponding to the comma-separated list in "Configuration of the Stop Watch Block" on page 2-22. To form this vector, the subsystem uses three Constant blocks, the three processed Clock block signals, and a Mux block.



Updating Multiple Portions of a Pie Chart

This example model enables you to control four portions of a pie chart independently using four sources. The model uses the Dynamic Pie block. Internally, the model uses a MATLAB S-function to drive the Dynamic Pie block. Compared to the technique in "Simulating a Multiple-Needle Stopwatch" on page 2-20, the S-function technique is more complicated but also more flexible and powerful.

To open this example, enter gauges_pie in the MATLAB Command Window. Change the values of the Constant blocks to alter the composition of the pie chart. You must close the dialog by clicking **OK** to see the result of your changes.



These sections describe how the model works:

How PortionID Values Correspond to Portions of the Pie

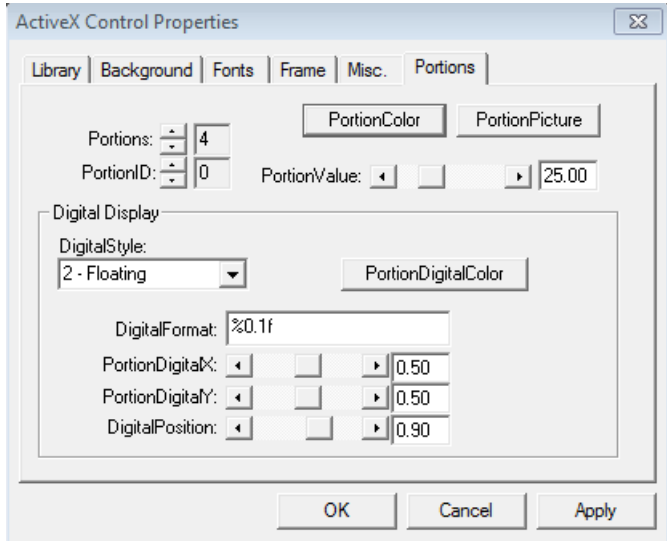
To explain how the model controls four portions, this section first explains how the block makes its portions accessible to you. Each portion on the block has an associated pair of **PortionID** and **PortionValue** parameters, where

- **PortionID** is a number that distinguishes that portion from other portions on the block.
- **PortionValue** is the numerical value for that portion, as a percentage. The **PortionValue** S-function normalizes the sum of the portion percentages to 100.

You can view or control these parameters via the **Portions** panel of the block's ActiveX Control Properties dialog box. To find out which portion corresponds to a given ID number, use this procedure:

1. Right-click the Dynamic Pie block and choose **Control Display Properties**.
2. Click the **Portions** tab.
3. Set **PortionID** in the top left corner to the ID number you want to investigate: `0`, `1`, `2`, or `3` in this case.

   The dialog box looks like this:



4 Change the **PortionValue** parameter while watching which portion on the block changes in size:

PortionID	Portion on Block
0	Red portion
1	Yellow portion
2	Magenta portion
3	Green portion

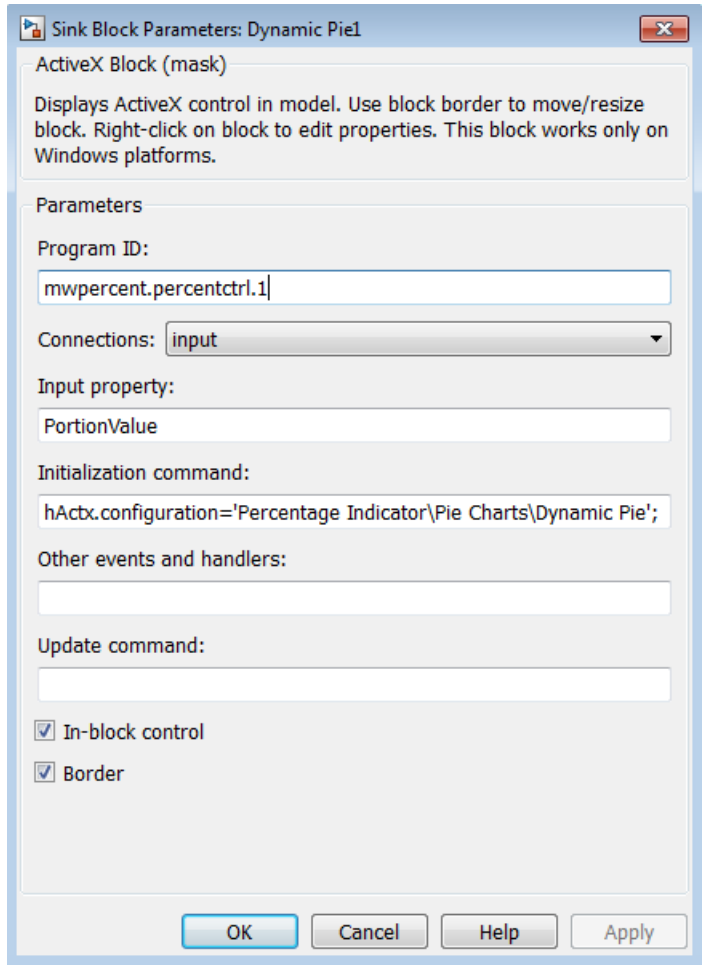
For more information about ID properties, see “Understanding ID Properties” on page 2-8.

Configuration of the Dynamic Pie Block

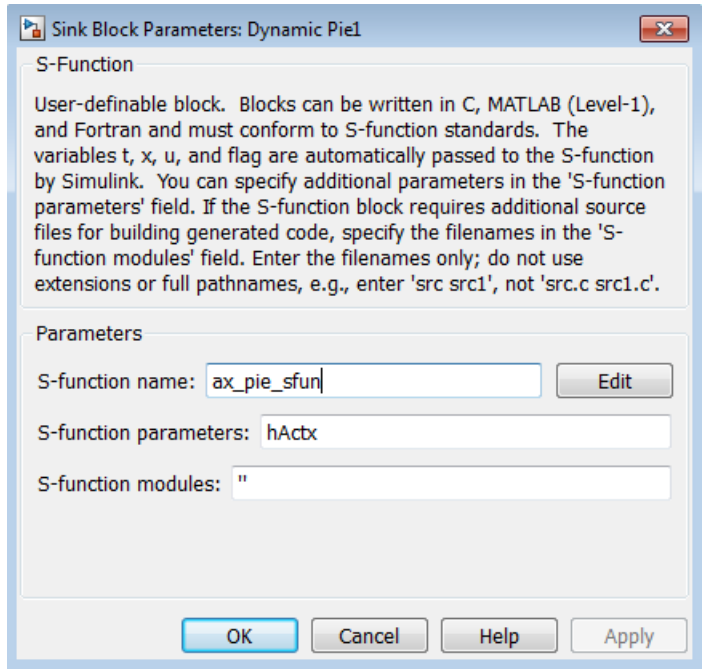
In the model, a vector containing the four constant values is the input to the Dynamic Pie block. While the simulation is running, an S-function called `gauges_pie_sfun.m` uses the vector to make the Dynamic Pie block reflect the constant values. While the behavior of the S-function is discussed below (“How the S-Function Updates the Pie” on page 2-29), this section describes how the Dynamic Pie block is linked with the S-function.

The Dynamic Pie block in this model is a customized copy of the original one in the Percent Indicators library. The customized copy differs from the original in these ways:

- The **Input property** field in the Block Parameters dialog box is blank. This is because the block is being updated by an S-function that ignores the data in the **Input property** field. To see the **Input property** field, right-click the block and choose **Block Parameters**:



- The block is driven by a customized S-function, `ax_pie_sfun.m`, rather than the S-function that drives the original library block. To see this, select the outer border of the block and select **Diagram > Mask > Look Under Mask**:



The S-function `ax_pie_sfun.m` is a customized version of `ax_strip_sfun.m`, which is an S-function designed to drive the Strip Chart block. Many parts of `ax_pie_sfun.m` are also similar to `sfuntmpl.m`, which is a MATLAB S-function template. The S-function variable *hActx* is used in the **Initialization** text box in the **Block Parameters** dialog box.

Many features of that S-function template are not required for controlling Gauges Blockset blocks, which simplifies the task of writing S-functions for use with the blockset.

If you were building this model yourself starting from the original library block, then you would have to break the library link before changing the values in the S-Function dialog box shown previously. To break the library link for a library block, use this procedure:

1. Select the outer border of the block.
2. Select **Diagram > Library Link > Disable link**.

3 Select **Diagram > Library Link > Break link**

How the S-Function Updates the Pie

While the simulation is running, the S-function `ax_pie_sfun.m` drives the Dynamic Pie block. In particular, this S-function

- Receives the vector that is the input signal to the Dynamic Pie block
- Normalizes the vector so that the values add up to 100
- Updates each portion of the pie to reflect the corresponding number from the vector

Receiving the Vector Input Signal

During the simulation, the Simulink engine invokes the S-function and passes it the Dynamic Pie block's input vector. Within the S-function, the vector is called `u`. The engine also passes to the S-function a handle of the Dynamic Pie control. The handle is called `hActx`.

Normalizing the Input Vector

The S-function uses the code below to normalize the vector `u` so that its elements add up to 100:

```
% First make sure there is no division by zero.
if sum(u)==0
   u=u+0.001;
end

% Now perform the normalization.
u = 100/sum(u).*u;
```

Updating the Dynamic Pie Block

After `sum(u)` is 100, the S-function updates the portions of the Dynamic Pie block by setting each one to the corresponding element of `u`. The code uses the handle `hActx` to access the **PortionID** and **PortionValue** parameters of the Dynamic Pie block.

```
% Loop through the portions and update their values.
%
if (length(u) ~= 0)
   for n=1:length(u)
      hActx.PortionID = n-1;
      hActx.PortionValue = u(n);
   end
```

```
end
```

Initial Portion Sizes in the Model

When you first open the `gauges_pie` model, the portions of the pie have equal sizes, unlike the portions in the default instance of the Dynamic Pie block in the Percent Indicators library. The equal-sized portions result from the configuration from which the model was saved.

If you were building this model yourself starting from the original library blocks, then you would first run the model to make the portion sizes reflect the values on the Constant blocks, and then save the model to record the blocks' configurations.

Saving and Reusing a Customized Control

In this section...

Saving Customized Controls Automatically

Saving the model also saves the property settings for the Gauges Blockset blocks in the model file. To share your customized controls with other users, give them the model file. To use your customized block in a new model, copy the block from the old model and paste it into the new model.

Saving Customized Controls Using the Library Panel

Alternatively, you can use the ActiveX Control Properties dialog box to save property settings for later use on your own machine. However, this method does not enable you to share these customized controls with users of other machines. The steps are

1. Select the **Library** tab of the ActiveX Control Properties dialog box.
2. Assign a name to the collection of modified settings by entering a new name in the **Configuration Name** field.

   Note: If you leave this field blank, the new property settings write over the previous settings, which means that you cannot access the original version except by reinstalling the blockset or by registering the controls again. To learn how to register the controls, see "Troubleshooting the Installation".
3. To provide textual information about the block, click **Notes**.
4. Enter a description in the text area and click **OK**.
5. Select the folder in which to store the modified control by expanding the library hierarchy at the left. The new set of property settings is stored in the folder you select. Click **Store**.
6. Click **OK** to accept the changes and close the dialog box.

An alternative to this procedure is to export customized controls to `.gms` files. To do this, select a folder from the left side of the panel and click **Export**. You can later access these controls by using the **Import** button, or share the controls by sharing the `.gms` files.

Categories of Controls

Angular Gauges Library

In this section...

Library Overview

The Angular Gauges library contains controls that show their input value graphically along an arc of a circle. Controls differ in numerical range and display elements. The library provides:

- Display elements — single and multiple needles, labels, annular regions, tick marks
- Preconfigured controls — Amp Meter, Analog Clock, Compass, Lower Left, Lower Right, Stop Watch, Upper Left, Upper Right, Vacuum, Volume
- Custom controls — Generic Angular Gauge

Customizing Angular Gauges

This section describes how to customize angular gauges by making changes that are specific to the Angular Gauges library. For changes that apply to multiple categories of blocks, see these sections:

- "Using Multiple Styles Within One Block"
- "Displaying Text on a Block"
- "Modifying the Displayed Range"
- "Modifying Multiple Tick Marks"

The table below lists some common customizations involving the ActiveX Control Properties dialog box that are specific to blocks in the Angular Gauges library.

Task	Description
Change the shape or size of a needle	On the **Needles** panel, set **NeedleID** to the ID of the needle you want to change (0 if the gauge has exactly one needle). Then use the **Style** property to choose the shape, and the **Length** and **Width** properties to determine the length and thickness.

Task	Description
Label a needle by displaying the corresponding number	On the **Digital** panel, set **NeedleID** to the ID of the needle you want to label and check the **Enabled** check box.
Change the appearance of a needle label	On the **Digital** panel, first set **NeedleID** to the ID of the needle whose label you want to change. Then use **Decimals** to set the number of digits after the decimal point, **Color** to set the color of the number, and **FontID** to refer to a previously defined font (on the **Fonts** panel).
Move a needle label	On the **Digital** panel, first set **NeedleID** to the ID of the needle whose label you want to change. Then use **X Position** and **Y Position** to set the fixed position for the label.
Draw an annular region along the scale	On the **Annulars** panel, increase the value of the **Annulars** property. The ID of the new region is the **Annulars** property value minus one. To specify properties of the new region, see the next task.
Change the appearance of an annular region	On the **Annulars** panel, first set **AnnularID** to the ID of the annular region you want to change. Use the **Radius** properties to control the annular region's thickness and radial position. Use the **Value** properties to control the portion of the scale's range that the annular region includes. Use **Color** to control the annular region's color.
Delete the most recently added annular region	On the **Annulars** panel, decrease the **Annulars** property. This deletes all properties associated with the region, such as its color and thickness.

Combining Multiple Needles in One Display

To display multiple needles on a single block, see the table below. To learn how to control multiple needles simultaneously, see “Controlling Multiple Graphical Elements”.

Task	Description
Add another needle to the display	On the **Needles** panel, increase the **Needles** property. The ID of the new region is the **Needles** property value minus one. To specify properties of the new needle, set **NeedleID** to that ID and then set the remaining properties on the dialog box panel accordingly.

Task	Description
Delete the most recently added needle from the display	On the **Needles** panel, decrease the **Needles** property. This deletes all properties associated with the needle, such as its color and shape.

LEDs Library

In this section...

Library Overview

The LEDs library contains controls that use graphical elements to imitate light-emitting diodes (LEDs). Each block displays its input value by setting one or more graphical elements to an on or off state. By default, the number of LEDs in the on state is the rounded value of the block's input. The library provides:

- Display elements — number of LEDs, shape, color, size, layout, decay timing, binary display
- Preconfigured controls
  - Single LED—Green Rect, Rect Bitmap, Red Rect, Red Rectangle Plain, Red Star, Round Red
  - Multiple-LED — Circle Meter, Horizontal Meter, Vertical Meter
- Custom controls — Generic LED

Customizing LEDs

The table below lists some common ways to customize a block in the LEDs library, using its ActiveX Control Properties dialog box.

Task	Description
Add or remove LEDs	Change the **NumLEDs** property on the **LEDs/General** panel.
Change the shape or color of a particular LED	On the **LEDs/General** panel, set **LEDIndex** to the number corresponding to the LED you want to customize. To apply a previously defined style, set the **LEDStyleID** to the number corresponding to the style. To define a new style for this LED, increase the **StyleID** property on the **Styles** panel and then configure the color, picture, or shape properties accordingly.

Task	Description
Change the size or layout of a set of LEDs	On the **LEDs/General** panel, use **LEDWidth** and **LEDHeight** to control the size of each LED. Use **LEDSeparation** to control the spacing between successive LEDs. Use **Orientation** and/or **Direction** to control how multiple LEDs are arranged along a line.
Display a binary representation of the (rounded) input	Set the **Mode** property on the **LEDs/General** panel to `Bitwise`. The first LED corresponds to the least significant bit.
Display decaying maximum value of the input, in addition to the current input	Check the **MaxDecay** check box on the **LEDs/General** panel. The **DecayRate** value controls how quickly the displayed value decays from the maximum to the current input value. Larger positive values correspond to a slower decay. A value of zero causes the block to reflect its maximum value with no decay.

Linear Gauges Library

In this section...

Library Overview

The Linear Gauges library contains controls that show their input values graphically along a linear scale. Controls differ in numerical range and display elements. The library provides:

- Display elements — pointers, bars, numerical labels, text captions, tick marks
- Preconfigured controls
  - Pointer controls — Min-Max Thermometer, Mixer, Mixer Scale, Multiple Scales
  - Bar controls — Reverse Sliding Scale, Scaled Bar Gauge, Tank, Thermometer
- Custom controls — Generic Linear Gauge, Generic Bar Gauge

Pointer linear gauges use one or more pointers to indicate values. Bar gauges use a level indicator or a two-color bar to reflect a single value or level.

Customizing Linear Gauges

This section describes how to customize linear gauges by making changes that are specific to the Linear Gauges library. For changes that apply to multiple categories of blocks, see these sections:

- "Using Multiple Styles Within One Block"
- "Displaying Text on a Block"
- "Modifying the Displayed Range"
- "Modifying Multiple Tick Marks"

Customizing Pointer Linear Gauges

The table below lists some common customizations involving the ActiveX Control Properties dialog box that are specific to pointer linear gauges in the Linear Gauges library.

Task	Description
Change the shape or size of a pointer	On the **Pointers** panel, set **PointerID** to the ID of the pointer you want to change (0 if the gauge has exactly one pointer). Then use the **Style** property to choose the shape, the **Inner** and **Outer** properties to determine the length, and the **Width** property to determine the thickness.
Label a pointer by displaying the corresponding number	On the **Digital** panel, set **PointerID** to the ID of the pointer you want to label and check the **PointerDigital** check box.
Change the appearance of a pointer label	On the **Digital** panel, first set **PointerID** to the ID of the pointer whose label you want to change. Then use **Decimals** to set the number of digits after the decimal point, **PointerDigitalColor** to set the color of the number, and **FontID** to refer to a previously defined font (on the **Fonts** panel).
Move a pointer label to a fixed position	On the **Digital** panel, first set **PointerID** to the ID of the pointer whose label you want to change. Clear the **PointerDigitalAttach** check box and use **PointerDigitalX** and **PointerDigitalY** to set the fixed position for the label.
Move a pointer label to a position relative to the pointer	On the **Digital** panel, first set **PointerID** to the ID of the pointer whose label you want to change. Check the **PointerDigitalAttach** check box. For a vertical linear scale, use **PointerDigitalX** to set the independent coordinate for the label. For a horizontal linear scale, use **PointerDigitalY** to set the independent coordinate for the label.

Customizing Bar Gauges

The table below lists some common customizations involving the ActiveX Control Properties dialog box that are specific to bar gauges in the Linear Gauges library.

Task	Description
Change the range of values along the bar	On the **General** panel, use the **Min Value** and **Max Value** properties to define the range.
Change the orientation or direction of the bar	On the **General** panel, use **Orientation** to determine whether the bar is horizontal or vertical. Use **Direction**

Task	Description
	to determine which end of the bar corresponds to the minimum value.
Change the size or position of the bar	On the **Bar** panel, use the **BarInner** and **BarOuter** properties to define the width and position of the bar in the direction perpendicular to the linear scale. Use the **BarStart** and **BarStop** properties to define the length and position of the bar in the direction of the linear scale. These properties do not change the numerical values associated with the bar, only the graphical depiction of the bar.
Change the colors of the portions of the bar on either side of the indicated level	On the **Bar** panel, use the **OnColor** and **OffColor** properties to define the colors associated with values below and above, respectively, the indicated level along the bar.
Change the shape or size of the level indicator	On the **Knob** panel, use the **Style** property to choose the shape. Use the **Inner Value** and **Outer Value** properties to determine the thickness and position in the dimension perpendicular to the linear scale. Use the **Width** property to determine the width along the linear scale.
Label the indicated level using a number	On the **Digital** panel, check the **Enabled** check box.
Change the appearance of the label	On the **Digital** panel, check the **Enabled** check box. Then use **Decimals** to set the number of digits after the decimal point, **Color** to set the color of the number, and **FontID** to refer to a previously defined font (on the **Fonts** panel).
Move the label to a fixed position	On the **Digital** panel, clear the **Attach** check box. Then use **X Position** and **Y Position** to set the fixed position for the label.
Move the label to a position relative to the level indicator	On the **Digital** panel, check the **Attach** check box. For a vertical (respectively, horizontal) linear scale, use **X Position** (respectively, **Y Position**) to set the independent coordinate for the label.

Combining Multiple Pointers in One Display

To display multiple pointers on a pointer linear gauge, see the customizations in the table below. To learn how to control multiple pointers simultaneously, see “Controlling Multiple Graphical Elements”.

Task	Description
Add another pointer to the display	On the **Pointers** panel, increase the **Pointers** property. The ID of the new region is the **Pointers** property value minus one. To specify properties of the new pointer, set **PointerID** to that ID and then set the remaining properties on the dialog box panel accordingly.
Delete the most recently added pointer from the display	On the **Pointers** panel, decrease the **Pointers** property. This deletes all properties associated with the pointer, such as its color and shape.

Numeric Displays Library

In this section...

Library Overview

The Numeric Displays library contains controls that display the numerical values of their input signals. Controls differ in numerical range and display elements. The Odometer block uses a specialized dialog box. The library provides:

- Display elements — Digit count, decimals, leading zeros, plus or minus, gradual transition (odometer only)
- Preconfigured controls
  - Numeric LED display— HH:MM, HH:MM:SS, IRIG Format, PlusMinus XX.XXX, VCR Clock
  - Odometer display—Odometer
- Custom controls — Generic Numeric LED

Customizing Numeric LED Displays

The table below lists some common ways to customize any block in the Numeric Displays library, *except* the Odometer block, using the **General** panel of its ActiveX Control Properties dialog box.

Task	Description
Change the number of digits in the display	Set **Digits** to the total number of digits.
Specify the number of digits after the decimal point	Set **Decimals** to the number of digits you want after the decimal point, and check the **FixedDecimal** check box.
Pad the display with leading zeros	Check the **LeadingZeros** check box.

Task	Description
Display a plus or minus sign	Check the **LeadingPlusMinus** check box.
Change the appearance of all digits	Use the **ItalicsOffset** property to control the slanting angle of digits. Use the **Segment Width** and **Segment Separation** properties to control the width of the line segments that compose each digit and the spacing between the line segments, respectively. Use the two **Spacing** properties to control the padding around each digit.
Avoid using nonnumeric characters such as a colon in the display, making the block easier to use with Simulink signals	First set **DisplayMode** to `0 - Numeric`. Then open the block's Block Parameters dialog box by double-clicking the block's border, and set **Input property** to `Value`.

Customizing the Odometer Block

The table below lists some common ways to customize the Odometer block, using the **General** panel of its ActiveX Control Properties dialog box.

Task	Description
Change the number of digits in the display	Set **Digits** to the total number of digits. Set **Decimals** to the number of digits after the decimal point. The block does not display a decimal point character, but digits that represent proper fractions appear with inverted colors.
Make the display change gradually from old to new values, instead of registering the change instantaneously	Check the **Transition Enabled** check box. Set the **Steps** value to the number of steps in the gradual transition. Use the **Rate** value to control the speed of the transition, where larger values indicate a slower transition.

On Off Gauges Library

In this section...

Library Overview

The On Off Gauges library contains gauges that can display two states, on and off. A block input of `0` corresponds to an "off" state and a block input of `1` corresponds to an "on" state. The blocks in this library differ in cosmetic ways, such as the images shown on the block. The library provides:

- Display elements — image, text, sound, beveling
- Preconfigured controls — Dip Switch Readout, Happy Face, Light Bulb, Lock, Mailbox, On Off Readout

Customizing On Off Gauges

The table below lists some common ways to customize a block in the On Off Gauges library, using its ActiveX Control Properties dialog box.

Task	Description
Associate an image with a state	Use the **Picture** button on the **On** or **Off** panel to select a graphics file. You cannot associate both an image and text with a state.
Associate text with a state	Use the **Caption** field on the **On** or **Off** panel. The **X** and **Y** values control the position of the text. The **BackColor** and **ForeColor** buttons control the colors of the background and text, respectively. You cannot associate both an image and text with a state.
Associate a sound with a state	On the **On** or **Off** panel, check the **Sound** check box and type the name of a `.wav` file in the **Wave file** field. You can either type the name of the sound file or browse for it using the **...** button.

Task	Description
Use beveling to make the button appear three-dimensional	Use the **BevelInner** and **BevelOuter** properties on the **Background** panel.

Percent Indicators Library

In this section...

Library Overview

The Percent Indicators library contains controls designed to display percentages and ratios. The library provides:

- Display elements — sector of circle, portion of rectangle, multiple portions
- Preconfigured controls — Dynamic Pie, Pie Chart, Simple Light Blue
- Custom controls — Generic Percent

The Generic Percent and Simple Light Blue blocks are probably the most useful blocks in this library. By default, these blocks reflect scalar input values between 0 and 100 by coloring a corresponding segment of a linear scale. By customizing the blocks, you can also have them display an input value *X* between *m* and *M* as the percentage 100 * ((*X* - *m*) / (*M* - *m*)).

Customizing Percent Indicators

The table below lists some common ways to customize a block in the Percent Indicators library, using its ActiveX Control Properties dialog box.

Task	Description
Use a radial percentage scale that reflects the input as a sector of a circle	On the **Misc**. panel, set **DisplayMode** to `Radial`. Use the **StartAngle** value to indicate where the sector begins; a value of 0 corresponds to a vertical radius above the circle's center, while a value of 90 corresponds to a horizontal radius to the right of the circle's center.
Change the direction in which a radial percentage scale increases	On the **Misc**. panel, use the **Direction** property to reverse the scale's polarity. If **Direction** is set to `Forward`, then the scale increases clockwise.
Use a linear percentage scale that reflects the input as a portion of a rectangle	On the **Misc.** panel, set **DisplayMode** to `Linear`.

Task	Description
Change the direction in which a linear percentage scale increases	On the **Misc.** panel, use the **Orientation** property to indicate whether the linear scale is horizontal or vertical. Use the **Direction** property to reverse the scale's polarity. If **Direction** is set to `Forward`, then a horizontal scale increases to the right and a vertical scale increases downward.
Specify the range to use when converting the input to a percentage	On the **Misc.** panel, use the **Min** and **Max** properties. If the input value is `X`, then the block displays the percentage: `100 * ((X - Min) / (Max - Min))` .
Display a number near or inside the corresponding colored area	On the **Portions** panel, set **DigitalStyle** to `Floating`. You can use the **DigitalPosition** value to vary the position along one dimension (radius in the case of a radial scale, height in the case of a horizontal scale, and horizontal coordinate in the case of a vertical scale).
Display a number in a fixed position	On the **Portions** panel, set **DigitalStyle** to `Fixed`. To specify the position of the number, first set **PortionID** to the ID of the portion you want to configure (`0` if you are displaying only the scalar input signal) and then use the **PortionDigitalX** and **PortionDigitalY** values to indicate the position.

Combining Multiple Regions in One Display

To display multiple regions on a single block, see the customizations in the table below. To learn how to control multiple regions simultaneously, see “Controlling Multiple Graphical Elements”.

Task	Description
Add another region to the display	On the **Portions** panel, increase the **Portions** property. The ID of the new region is the **Portions** property value minus one. To specify properties of the new region, set **PortionID** to that ID and then set the remaining properties on the dialog box panel accordingly. Note that the **DigitalStyle** and **DigitalFormat** properties apply to all regions on the block.

Task	Description
Delete the most recently added region from the display	On the **Portions** panel, decrease the **Portions** property. This deletes all properties associated with the region, such as its color.

Strip Chart Library

Library Overview

The Strip Chart block displays numeric data as if it were being output by pens moving on a strip of paper. The library provides stamps, tracks and track bands, and an `X` axis as display elements.

As with other preconfigured Gauges Blockset blocks, you can configure the Strip Chart block using properties in its dialog box. However, to plot data on the chart, you must invoke methods for the block using the MATLAB `invoke` command.

Plotting on a Strip Chart

A MATLAB S-function provided with Gauges Blockset software plots data on the Strip Chart block by using the `invoke` method. More generally, this S-function illustrates how to communicate with any Microsoft ActiveX control through a MATLAB S-function.

The file is called `ax_strip_sfun.m` and you can use the following MATLAB command to view the contents of the file.

```
edit ax_strip_sfun
```

During initialization, the Simulink block attributes (sample time, input width, etc.) are configured and the Strip Chart configuration is set. The infrastructure of Gauges Blockset software provides the handle of the control (`hActX`) and is available in this S-function.

You can use this handle to set the properties of the Strip Chart through the standard dot notation. For example, the following line sets the `LastX` property of the Strip Chart to zero.

```
hActx.LastX = 0;
```

Any property of the Strip Chart can be set in this fashion.

In the outputs section of the S-function, each track of the Strip Chart is initialized to zero on the time axes and the actual plotting of the data is performed. A loop in this section accounts for input signals that are vectors instead of scalars.

Using a Custom ActiveX Control

In this section...

See also "Block Parameters for the ActiveX Control Block" on page 3-22.

Adding the ActiveX Control Block to a Model

To use a custom Microsoft ActiveX control in a Simulink model, you must associate it with the generic ActiveX Control block. To configure the ActiveX Control block to display your control, you need to know:

- The name under which the control is registered on your system
- The events that cause the control to perform an action
- The control properties that are changed by events, by signals passed to the block, or by initialization commands

To use an ActiveX Control block in a Simulink model:

1. Drag the ActiveX Control block from the top level of the Gauges Blockset library to your model. Place the block where you want the control to appear.
2. Double-click the block to display its Block Parameters dialog box. Specify values as described in subsequent sections.

Note:

- Double-clicking the border of a preconfigured block (supplied with the blockset) displays its ActiveX Control Properties dialog box, which lists properties in multiple tabbed panels.
- Double-clicking a block that you created by customizing the generic ActiveX Control block displays its Block Parameters dialog box.

Notes on Third-Party Controls

This section contains additional notes about third-party controls. One note is about editing controls that ignore mouse events, while another concerns the colors of controls.

Editing Controls That Ignore Mouse Events

Certain controls do not handle typical mouse events (double-click, right-click, etc.). It appears that you cannot edit these controls when you use them with the ActiveX Control block; double-clicking or right-clicking blocks that use these controls produces no response. To edit this type of block, you must first select the block so that it is current in the Simulink model. Then enter the following commands in the MATLAB Command Window:

```
temp = get_param(gcb,'userdata'));
propedit(temp.hActx);
```

This opens the properties dialog box for that control. See the COM sections in the MATLAB documentation for more information on the "`propedit`" command and assigning event callbacks to controls.

Additionally, you can choose an event on your control through which you want to open the property editor. For example, write a function to open the property editor (or whatever you want the event to do). The function must take multiple arguments, of which the first one is the handle of the control. For example, a simple function to open the property editor of a control looks like this:

```
function axeventhandler(varargin)
propedit(varargin{1})
```

Next enter an event with the handler you just wrote in the **Other Events and Handlers** parameter field. Assuming the `keypress` event is valid, the event and handler entry looks like this:

```
{'keypress','axeventhandler'}
```

To use the error-checking code already written for Gauges Blockset software, you can enter `ax_block_dclk` for events that should open the property editor (note that the editor does not open when the simulation is running). For example, to make a keystroke open the property editor (assuming that the `keypress` event is valid), enter the event and handler pair as follows:

```
{'keypress','ax_block_dclk'}
```

Colors of Controls

In Simulink, controls cannot determine their colors by inheriting them from the window in which they reside. More specifically, controls that send the `WM_CTLCOLOR` message to their parent have this problem. `WM_CTLCOLOR` is a Microsoft Windows message sent by a control to enable the parent container to determine the color used by the control.

Caution Placing one of these controls in the ActiveX Control block causes MATLAB software to crash.

Block Parameters for the ActiveX Control Block

In this section...

Summary of Parameters

Name	Description
Program ID	Name of the control
Connections	Whether the ActiveX Control block has ports
Input property	Name of the property that is set when the ActiveX Control block receives a signal
Initialization command	Command that sets the initial conditions for the ActiveX Control block
Other events and handlers	Events that trigger an action by the ActiveX Control block
Update command	Function that the Simulink engine invokes when it updates the block during the simulation
In-block control	Whether the ActiveX Control block displays a control or is connected to an ActiveX Control block somewhere else
Border	Whether a border appears around the control

Program ID

The **Program ID** parameter is the name of the control displayed on the block. To determine the **Program ID** of other controls, consult their documentation.

Connections

The **Connections** parameter determines whether the block has an inport only, both an inport and an outport, or no ports. If the block has both an inport and an outport, then the values at both ports are the same.

Input Property

The **Input property** parameter indicates the name of the block property whose value is set by the input signal. Each preconfigured Gauges Blockset block stores the block's current value in a property, as listed in the table below.

Names of Input Properties

Library	Property Name
Angular Gauges	`NeedleValue`
LEDs	`Value`
Linear Gauges	• `BandStop` (Min-Max Thermometer) • `PointerValue` (other pointer linear gauges) • `Value` (bar gauges)
Numeric Displays	• `Value` (Generic Numeric LED, Odometer, PlusMinus XX.XXX) • `AlphaNumeric` (others)
On Off Gauges	`Value`
Percent Indicators	`PortionValue`

Initialization Command

The **Initialization command** parameter is a string that sets the initial conditions of the ActiveX Control block.

The handle of the control is `hActX`.

Other Events and Handlers

The **Other events and handlers** parameter specifies actions taken by the ActiveX Control block when you perform a defined action on the ActiveX Control block. You must enter an event as an *N* x 2 cell array. The first entry in each row must be the name of the event. The second entry in each row must be the MATLAB callback to be executed.

For a list and description of supported events for a control, consult the control's help.

Update Command

The **Update command** parameter is the name of the function that Simulink software invokes when it updates the block during a simulation. The function has these input arguments:

- The handle of the control
- The current input value

The function is not invoked when you update the diagram.

In-Block Control

The **In-block control** check box determines where the ActiveX Control block value is displayed. The control can be on the block icon, in the same model window, in a different model, in a subsystem, or in a MATLAB figure.

If checked, the control whose name is specified in the **Program ID** field appears on the ActiveX Control block.

If cleared, the block is connected to the control whose handle is specified in the **Handle location** field (this field appears when you clear the box):

- If the window is a MATLAB figure window, specify the name of a function whose return value is the figure handle. You can also specify initialization commands in the function to set the initial conditions of the ActiveX Control block.
- If the window contains a Simulink subsystem, the control must be displayed on an ActiveX Control block contained in that subsystem. Specify the path of the ActiveX Control block on which the control is to appear.

For example, if a model named `my_model` has a subsystem called `sub_disp_signals` that contains an ActiveX Control block named `signal1`, the path is `my_model/sub_disp_signals/signal1`.

Using this feature is useful in a complex model that displays signals in multiple subsystems on ActiveX Control blocks. If you feed the signals into ActiveX Control blocks but display the controls themselves in a separate system or window, you do not have to open the subsystems to see the results.

Border

The **Border** check box determines whether the block displays a border around the control.

Note: Be careful when clearing this box, because the only way to move a block is to drag it with the border. Clearing the **Border** box renders the ActiveX Control block immovable.

4

Placing Controls in a Different Window

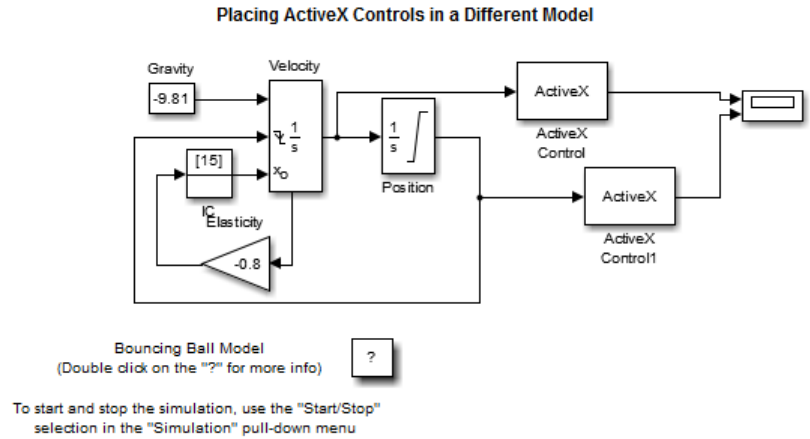
Placing Controls in a Different Model

In this section...

Example Overview

This example modifies the Simulink example, `sldemo_bounce`, by displaying the position and velocity signals on Gauges Blockset blocks contained in another model window.

To open the original example model, enter sldemo_bounce in the MATLAB Command Window. To open the modified version, enter gauges_bounce. The modified version includes two ActiveX Control blocks on the signals that feed into the Scope block:



Creating a Model Window Containing Gauges

Create a new model called `gauges_bounce_gui` and copy the following Gauges Blockset blocks into it:

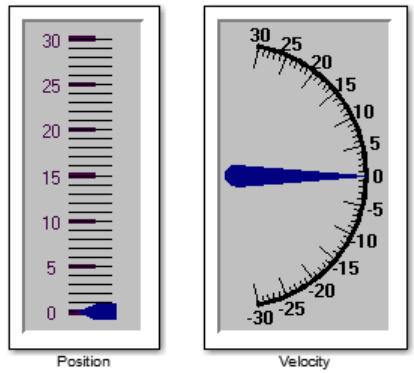
- The Generic Linear Gauge block from the Linear Gauges library. Change the block's name to `Position`.
- The Amp Meter block from the Angular Gauges library. Change the block's name to `Velocity`.

Customizing the Gauges

If you want to customize the gauges, particularly the range of values that they can display, then use this optional procedure:

1. Open the ActiveX Control Properties dialog box for the Position (Generic Linear Gauge) block.
2. From the **Scales** panel, set **ScaleMax** to 30. This enables the gauge to display values between 0 and 30.
3. From the **Ticks** panel, set **StopValue** to 30, set **DeltaValue** to 5, check the **Label On/Off** check box, and set **Width** to `0.012`. This creates labeled major ticks.
4. Still on the **Ticks** panel, set **Ticks** to 2, set **TickID** to 1, set **DeltaValue** to 1, set **Inner** to `0.4`, and set **Outer** to `0.75`. This creates a set of unlabeled minor ticks.
5. From the **Pointers** panel, click **Color**, choose the color that matches the pointer on the Velocity (Amp Meter) block, and click **OK**. Also, set **Value** to 0.
6. Click **OK**.
7. Open the ActiveX Control Properties dialog box for the Velocity (Amp Meter) block.
8. From the **Captions** panel, set **Captions** to 0. This removes the word Amps.
9. From the **Annulars** panel, set **Annulars** to 1. This removes the colored shading of the annular region.
10. From the **Needles** panel, set **Value** to 0. This moves the needle so that it points to zero.
11. From the **Scales** panel, set **Min** to `-30`, set **Max** to 30, select **Backward**, set **Start** to 10, and set **Stop** to 170. This causes the block to display values between `-30` and 30 along the right half of a circle.
12. Still on the **Scales** panel, set **X** to `-1.06` and set **Y** to `0.04`. This helps center the control in the block.
13. From the **Ticks** panel, set **DeltaValue** to 5. This creates labeled major ticks.
14. Still on the **Ticks** panel, set **TickID** to 1 and set **DeltaValue** to 1. This creates unlabeled minor ticks.
15. Click **OK**.

You might also want to enlarge the blocks. They should now look like this.



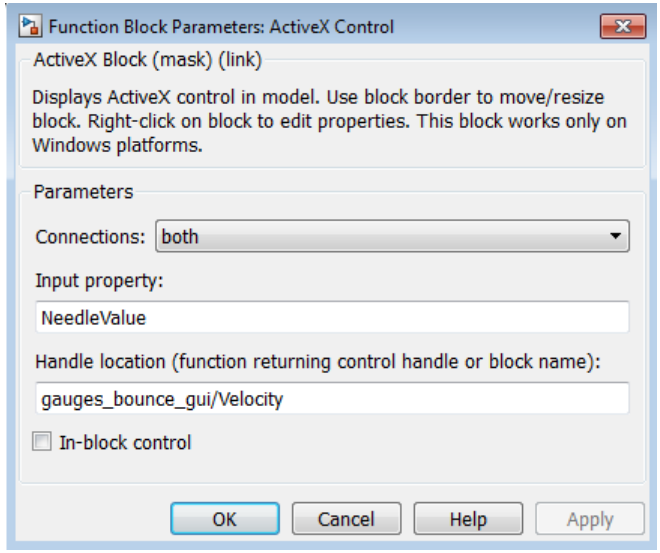
Associating the Main Model with the Gauges

Open the original `sldemo_bounce` model and save it in your working folder as `gauges_bounce`. Insert two ActiveX Control blocks on the signals that feed into the Scope block. To connect the ActiveX Control blocks to the controls, make these changes in the Block Parameters dialog box in each of the ActiveX Control blocks:

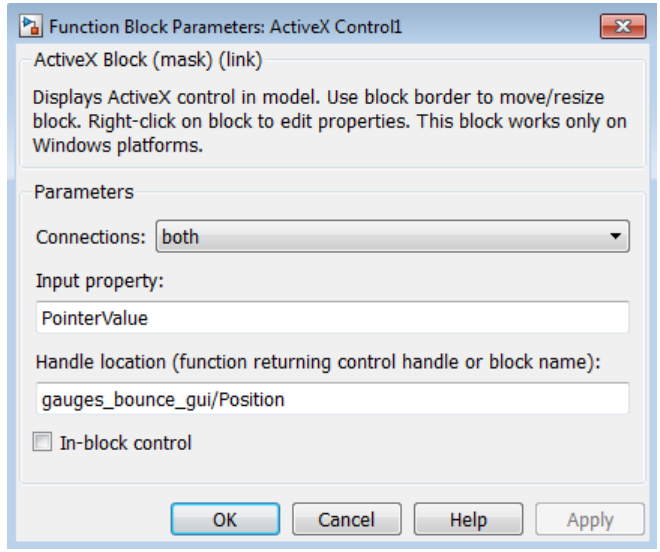
1. Clear the **In-block control** check box, because the signal is being communicated between ActiveX Control blocks in one window and ActiveX Control blocks in another window. When you clear the **In-block control** check box, the number of fields on the dialog box changes.
2. In the **Input property** field, specify `NeedleValue` for the velocity block and `PointerValue` property for the position block. This property controls the current values of these gauges. Doing this passes the value of the input signal to this property.

   Note: If you adapt this example to use the Strip Chart control instead, then set **Input property** to `Y`. For other controls in this blockset, set **Input property** to the value used in the corresponding parameter field in the library block.
3. Specify the path of each gauge in the **Handle location** field. The new model is named `gauges_bounce_gui`, so the path is `gauges_bounce_gui/Velocity` for the velocity block and `gauges_bounce_gui/Position` for the position block.

The dialog boxes look like this:



ActiveX Control Connected to Amp Meter (Velocity)



ActiveX Control Connected to Generic Linear Gauge (Position)

Now, when you simulate the main model window, the gauges in the auxiliary model window reflect the velocity and position of the bouncing ball.

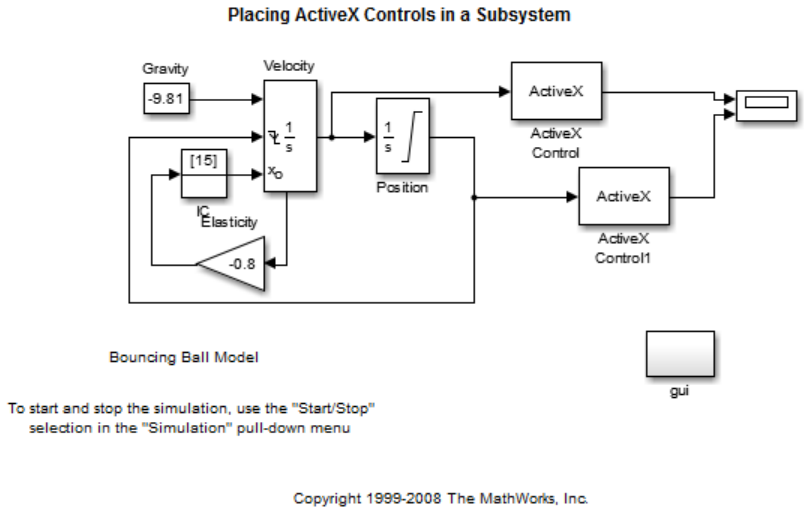
Placing Controls in a Subsystem

In this section...

Example Overview

This example builds on the one described in “Placing Controls in a Different Model” on page 4-2, but places the Gauges Blockset blocks in a subsystem of the main model rather than a different model. This approach simplifies operations such as saving and closing the system because the system involves only a single Simulink model file.

To open a completed version of this example, enter gauges_bounce_subsys in the MATLAB Command Window. Notice that the model includes a subsystem called `gui` in the lower right.



Creating a Subsystem Containing Gauges

To create the subsystem, follow these steps:

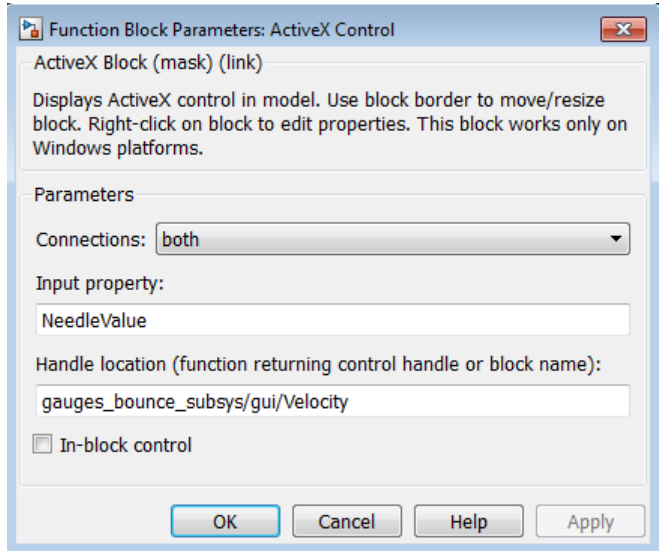
1. Open the `sldemo_bounce` model and save it in your working folder as `gauges_bounce_subsys`.
2. Copy a Subsystem block from the Simulink Signals & Systems library into the model window. Change the block's name to `gui`.
3. Double-click the subsystem to open it.
4. Copy a Generic Linear Gauge block from the Linear Gauges library into the subsystem. Change the block's name to `Position`.
5. Copy an Amp Meter block from the Angular Gauges library into the subsystem. Change the block's name to `Velocity`.
6. In the Block Parameters dialog box for each of the two gauge blocks, set the **Connections** parameter to `neither` and clear the **Input property** edit field.

To customize the gauge blocks, see “Customizing the Gauges” on page 4-3.

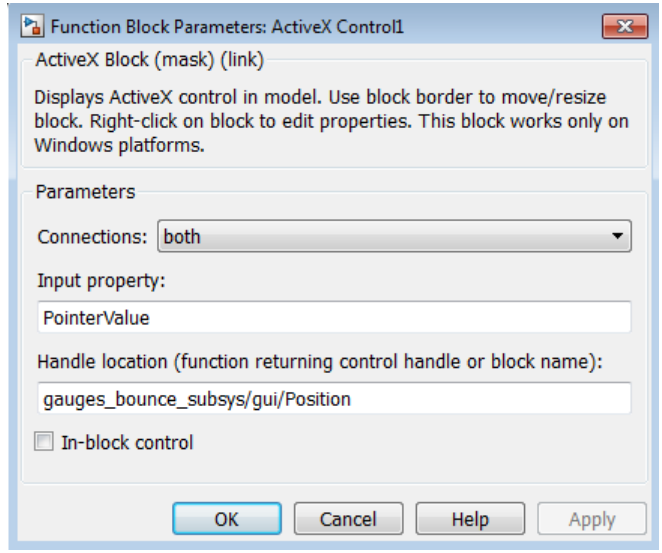
Associating Top-Level Blocks with the Subsystem

The procedure for associating the top-level ActiveX Control blocks with the gauge blocks that are inside the subsystem is similar to the procedure described in “Associating the Main Model with the Gauges” on page 4-4. The only difference is that the **Handle location** parameters have different values for a subsystem than for a separate model. Specifically, the path is `gauges_bounce_subsys/gui/Velocity` for the velocity block and `gauges_bounce_subsys/gui/Position` for the position block.

The dialog boxes look like this:



ActiveX Control Connected to Amp Meter (Velocity)



ActiveX Control Connected to Generic Linear Gauge (Position)

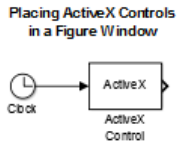
Placing Controls in a Figure Window

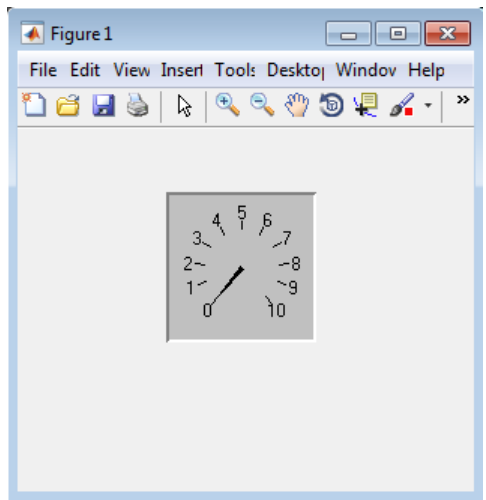
In this section...

Example Overview

In this example, a simple model displays the simulation time on a control located in a MATLAB figure window.

To open a completed copy of the model, enter gauges_offblock in the MATLAB Command Window:





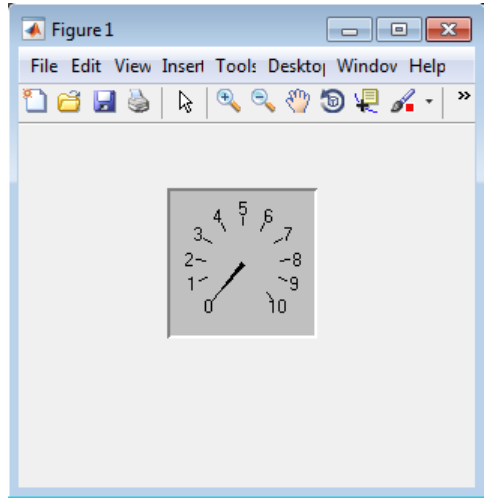
Alternatively, follow the instructions below for building it yourself.

Creating Helper Scripts and Building the Model

1 Create and execute a script called `gauges_gaugewindow` that consists of these statements.

```
f = figure;
h = actxcontrol('mwagauge.agaugectrl.1',[100 100 100 100],f);
```

This script creates a figure window containing a Generic Angular Gauge, with a program ID of `mwagauge.agaugectrl.1` and a position in the figure window of `[100 100 100 100]`. For more information, see `actxcontrol` and "Block Parameters for the ActiveX Control Block".

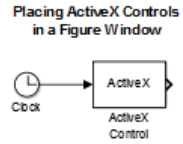


2 Create a script called `gauges_off_block` that consists of these statements.

```
function hactx = gauges_off_block
hactx = evalin('base','h');
```

The `gauges_off_block` function returns the handle of the control that is to be connected to the ActiveX Control block.

3 In a new model window, connect a Clock block to an ActiveX Control block.



4 Open the ActiveX Control block and modify its parameters as follows:

a Clear the **In-block control** check box. The number of fields on the dialog box changes as a result.

b In the **Connections** field, select `input`. When you apply this change later, the outport on the ActiveX Control block will disappear.

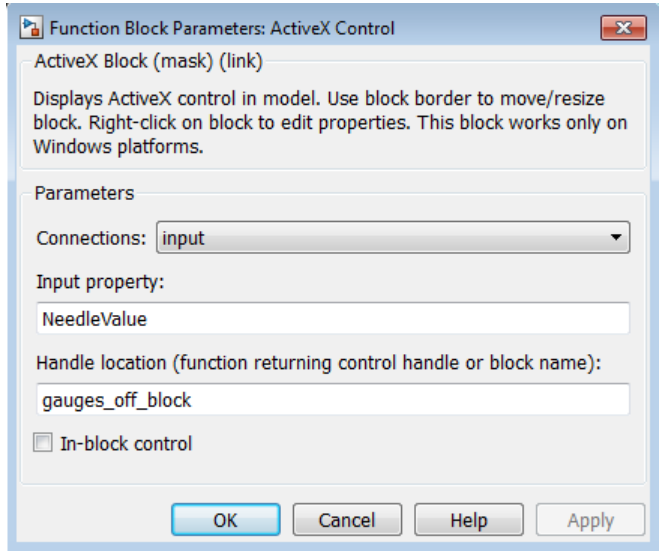
c In the **Input property** field, enter `NeedleValue`. During the simulation, the Generic Angular Gauge (that is, the control referenced by the block) sets the `NeedleValue` property to the value of the signal at the ActiveX Control block's inport.

Note: If you adapt this example to use the Strip Chart control instead, then set **Input property** to `Y`. For other controls in this blockset, set **Input property** to the value used in the corresponding parameter field in the library block.

---

d In the **Handle location** field, enter `gauges_off_block`.

The dialog box looks like this:



5 Click **OK**. The `gauges_off_block` script executes and returns the handle of the control in the figure window.

6 Run the simulation. Notice that the Generic Angular Gauge reflects the clock time.

Note: If you accidentally close the figure window before you are finished exploring the model, you can recreate it by executing `gauges_gaugewindow`.

Saving and Reopening the Model

If you want to use this model in a different MATLAB software session, then you must preserve both the model and the commands that create the figure window and gauge. Here is an easy way to do this:

1 Save the model to give it a name.

2 If the model's name is `mymodel`, then use these commands in the MATLAB Command Window to preserve the commands that create the figure window and gauge.

```
set_param('mymodel','PreLoadFcn','gauges_gaugewindow');
save_system
```

Now, opening `mymodel` automatically creates the figure that contains the gauge.

5

Block Reference

Angular Gauges

Display input value on arc

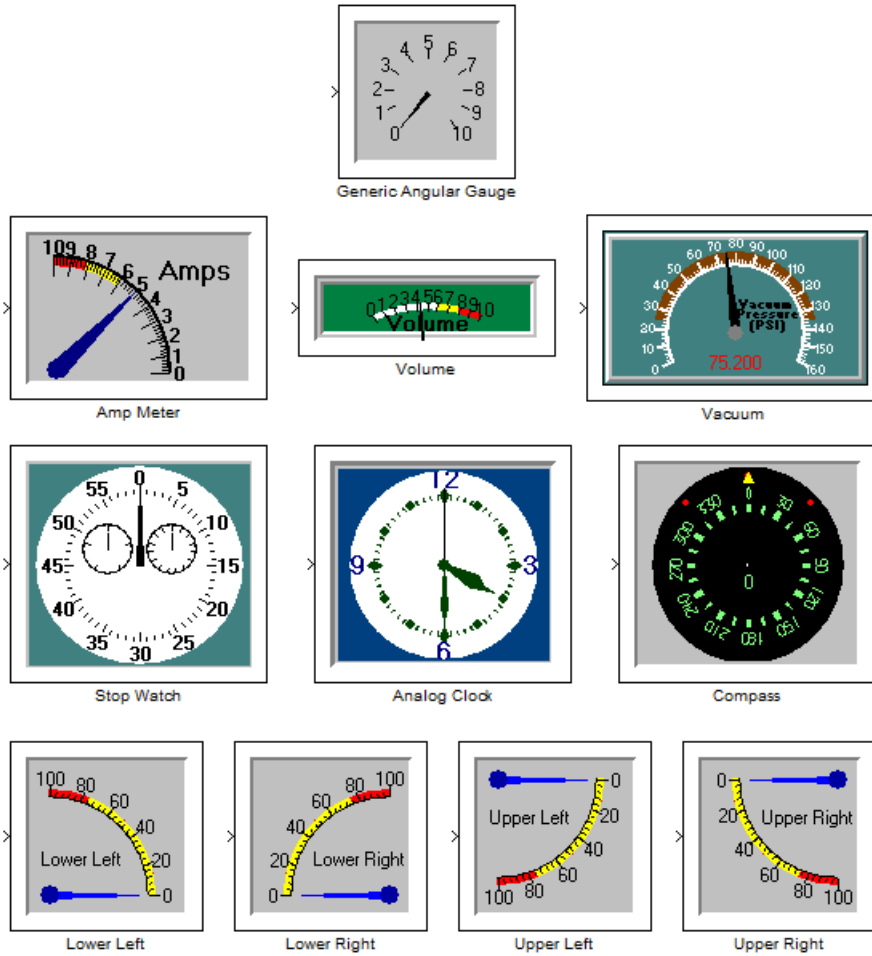
Description

Blocks in the Angular Gauges library show their input values graphically on a scale that lies along an arc of a circle. If the input value is greater than the scale's maximum or less than the scale's minimum, then the block displays the maximum or minimum value, respectively. To learn how to use and customize blocks in this library, see “Angular Gauges Library”.

Note: Blocks in this library can display multiple needles. The Stop Watch and Analog Clock blocks display multiple needles by default. To learn how to control multiple needles simultaneously, see “Controlling Multiple Graphical Elements”.

Blocks in the Library

The Angular Gauges library contains ActiveX Control blocks preconfigured with ActiveX controls from Global Majic, as shown.



Dialog Box

The ActiveX Control Properties dialog box governs the appearance and functionality of the control that is embedded in the block. The table below lists the panels of the ActiveX Control Properties dialog box.

Panel	Purpose
Annulars	Display annular regions along the block's scale
Background	Configure the background and outline of the block
Captions	Display annotations on the block
Digital	Display the numerical value corresponding to the needle
Frames	Display a border on the block
Fonts	Define text styles (The **Captions**, **Digital**, and **Ticks** panels use the **FontID** property to reference the styles defined here.)
Hubs	Embellish a needle's axis of rotation
Library	Refer to property settings as a named collection
Needles	Display one or more needles on the block (The **Digital** panel uses the **NeedleID** property to reference the needles defined here.)
Scales	Define the ranges and locations of values displayed on the block (The **Annulars**, **Hubs**, **Needles**, and **Ticks** panels use the **ScaleID** property to reference the ranges defined here.)
Ticks	Display markers and/or numbers at intervals along the scale

The Block Parameters dialog box governs the relationship between the Simulink block and the control embedded in the block. See “Block Parameters for the ActiveX Control Block” for details.

LEDs

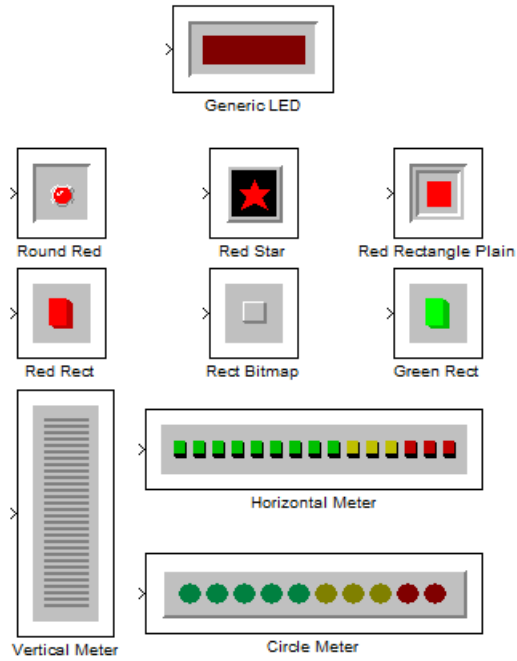
Display input value using two-state graphical elements

Description

Blocks in the LEDs library use graphical elements to imitate light-emitting diodes (LEDs). Each block reflects its input value by setting one or more graphical elements to an on or off state. By default, the number of LEDs in the on state is the rounded value of the block's input. If the rounded value is nonpositive, then all LEDs are in the off state. If the rounded value exceeds the number of LEDs, then all LEDs are in the on state. To learn how to use and customize blocks in this library, see “LEDs Library”.

Blocks in the Library

The LEDs library contains ActiveX Control blocks preconfigured with ActiveX controls from Global Majic, as shown.



Dialog Box

The ActiveX Control Properties dialog box governs the appearance and functionality of the control that is embedded in the block. The table below lists the panels of the ActiveX Control Properties dialog box.

Panel	Purpose
Background	Configure the background and outline of the block
LEDs/General	Define the number, arrangement, and behavior of LEDs on the block
Library	Refer to property settings as a named collection
Style	Define the appearance of LED graphical elements (The **LEDs/General** panel uses the **LEDStyleID** property to reference the styles defined here.)

The Block Parameters dialog box governs the relationship between the Simulink block and the control embedded in the block. See “Block Parameters for the ActiveX Control Block” for details.

Linear Gauges

Display input value on line

Description

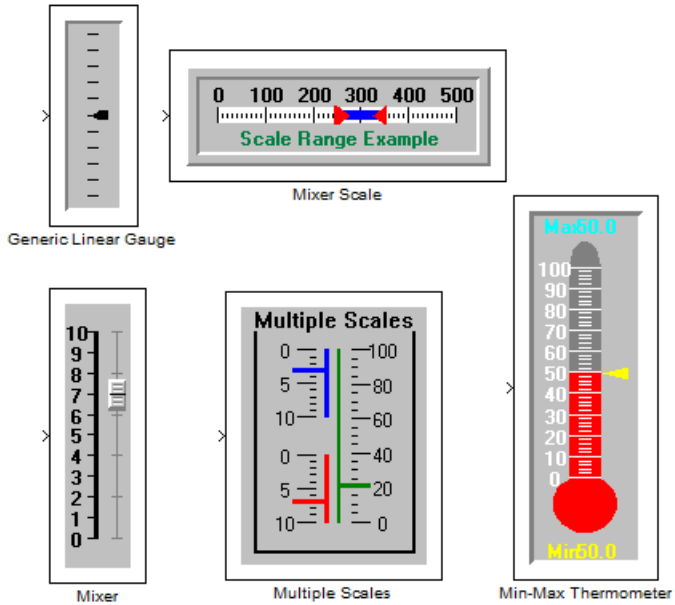
Blocks in the Linear Gauges library show their input value graphically on a scale that lies along a line. If the input value is greater than the scale's maximum or less than the scale's minimum, then the block displays the maximum or minimum value, respectively. To learn how to use and customize blocks in this library, see “Linear Gauges Library”.

Note: Some blocks in this library can display multiple pointers. The Multiple Scales block displays multiple pointers by default. To learn how to control multiple pointers simultaneously, see “Controlling Multiple Graphical Elements”.

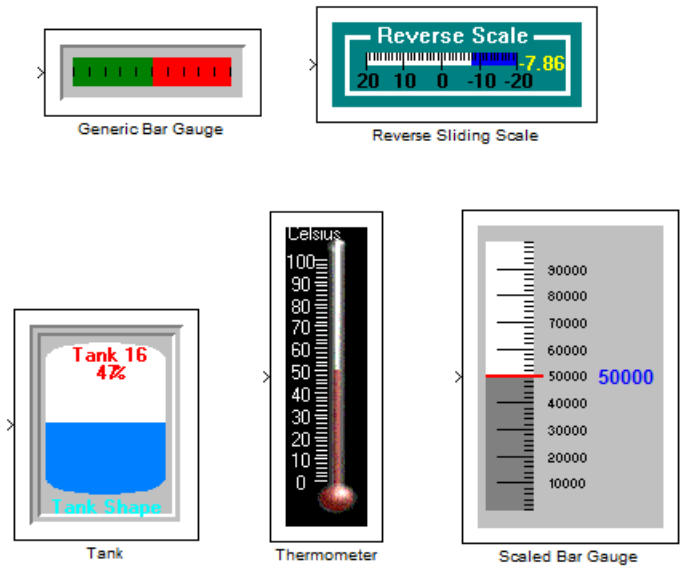
Blocks in the Library

The Linear Gauges library contains ActiveX Control blocks preconfigured with ActiveX controls from Global Majic. The blocks fall into two categories, pointer gauges and bar gauges.

The pointer linear gauges are:



The bar linear gauges are:



Dialog Box

Properties of All Library Blocks

The ActiveX Control Properties dialog box governs the appearance and functionality of the control that is embedded in the block. The table below lists the panels of the ActiveX Control Properties dialog box for all blocks in this library.

Panel	Purpose
Background	Configure the background and outline of the block
Captions	Display annotations on the block
Digital	Display the numerical value corresponding to the pointer
Fonts	Define text styles (The **Captions**, **Digital**, and **Ticks** panels use the **FontID** property to reference the styles defined here.)

Panel	Purpose
Library	Refer to property settings as a named collection
Ticks	Display markers and/or numbers at intervals along the scale

The Block Parameters dialog box governs the relationship between the Simulink block and the control embedded in the block. See “Block Parameters for the ActiveX Control Block” for details.

Properties Specific to Certain Library Blocks

The ActiveX Control Properties dialog box of the pointer linear gauges has additional panels as in the table below.

Panel	Purpose
Bands	Display linear or rectangular regions along the block's scale
Pointers	Display one or more pointers on the block (The **Digital** panel uses the **PointerID** property to reference the pointers defined here.)
Scales	Define the ranges and locations of values displayed on the block (The **Bands**, **Pointers**, and **Ticks** panels use the **ScaleID** property to reference the ranges defined here.)

The ActiveX Control Properties dialog box of the bar gauges has additional panels as in the table below.

Panel	Purpose
Bar	Define the appearance of the linear bar
General	Define the range and orientation of the block's scale
Knob	Define the appearance of the level indicator

Numeric Displays

Display input value using LED digits or numbered wheels

Description

Blocks in the Numeric Displays library show the numerical values of their inputs using graphical elements that imitate either numerals composed of light-emitting diode (LED) segments, or numbered wheels.

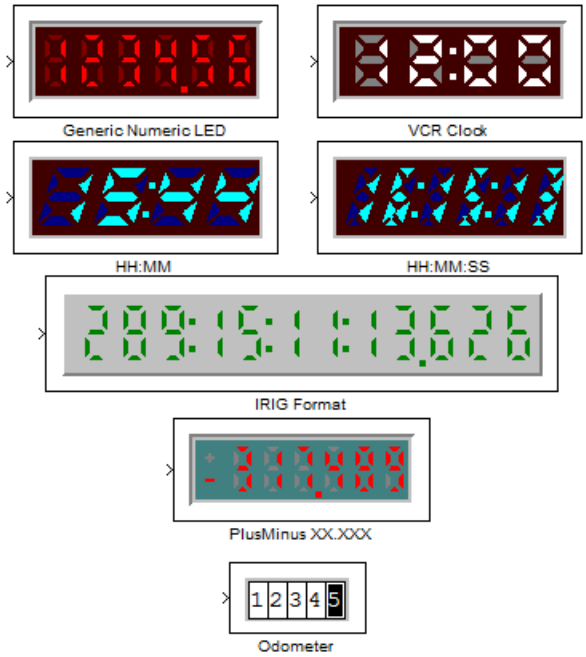
The display of the Odometer block imitates the numbered wheels of a car's odometer.

To learn how to use and customize blocks in this library, see “Numeric Displays Library”.

Note: Blocks in this library can display alphanumeric characters if their **DisplayMode** properties are set to `AlphaNumeric`. Some blocks, such as the IRIG Format block, use this alphanumeric mode by default. However, Simulink signals are always *numeric* values. When using the alphanumeric mode, you can control the display via a MATLAB S-function that uses the “COM support features” in MATLAB software. For an example of controlling a gauge using a MATLAB S-function, see “Updating Multiple Portions of a Pie Chart”.

Blocks in the Library

The Numeric Displays library contains ActiveX Control blocks preconfigured with ActiveX controls from Global Majic, as shown.



Dialog Box

The ActiveX Control Properties dialog box governs the appearance and functionality of the control that is embedded in the block. The table below lists the panels of the ActiveX Control Properties dialog box.

Panel	Purpose
Background	Configure the background and outline of the block
General	Define the number, appearance, and arrangement of digits on the block
Library	Refer to property settings as a named collection

The Block Parameters dialog box governs the relationship between the Simulink block and the control embedded in the block. See “Block Parameters for the ActiveX Control Block” for details.

On Off Gauges

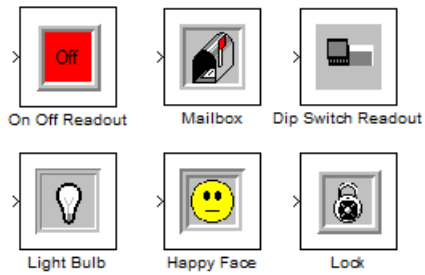
Display two states

Description

Blocks in the On Off Gauges library are two-state gauges. A block input of 0 corresponds to an “off” state and a block input of 1 corresponds to an “on” state. To learn how to use and customize blocks in this library, see “On Off Gauges Library”.

Blocks in the Library

The On Off Gauges library contains ActiveX Control blocks preconfigured with ActiveX controls from Global Majic, as shown.



Dialog Box

The ActiveX Control Properties dialog box governs the appearance and functionality of the control that is embedded in the block. The table below lists the panels of the ActiveX Control Properties dialog box.

Panel	Purpose
Background	Configure the background and outline of the block
General	Determine how the button's beveling (if visible) responds to a state change

Panel	Purpose
Library	Refer to property settings as a named collection
Off	Associate visual (text caption or image) and/or audio cues with the off state of the button
On	Associate visual (text caption or image) and/or audio cues with the on state of the button

If you use the question-mark icon to access the online help for the controls in this library, then note that Simulink blocks use the controls only for readout, with no mouse control.

The Block Parameters dialog box governs the relationship between the Simulink block and the control embedded in the block. See “Block Parameters for the ActiveX Control Block” for details.

Percent Indicators

Display percentage or ratio, using linear or circular scale

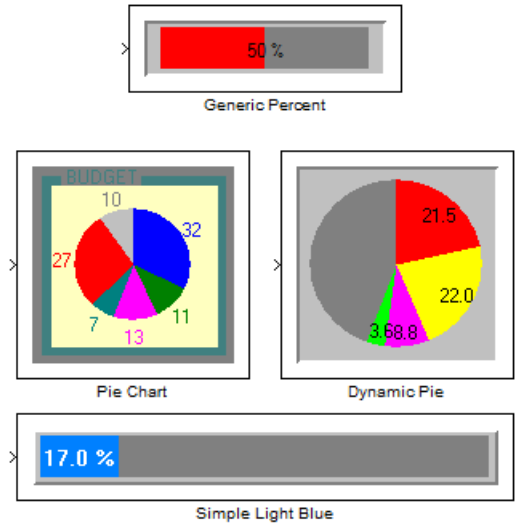
Description

Blocks in the Percent Indicators library convert their input values to percentages or ratios. They display the percentage or ratio graphically as either a segment on a linear scale or a sector of a circle. To learn how to use and customize blocks in this library, see "Percent Indicators Library".

Note: Blocks in this library can display multiple values simultaneously using percentages or ratios. The Pie Chart block displays multiple values by default. To learn how to display multiple values simultaneously, see "Controlling Multiple Graphical Elements".

Blocks in the Library

The Percent Indicators library contains ActiveX Control blocks preconfigured with ActiveX controls from Global Majic, as shown.



Dialog Box

The ActiveX Control Properties dialog box governs the appearance and functionality of the control that is embedded in the block. The table below lists the panels of the ActiveX Control Properties dialog box.

Panel	Purpose
Background	Configure the background and outline of the block
Frames	Display a border on the block
Library	Refer to property settings as a named collection
Misc	Define the shape, orientation, and range of the block's scale
Portions	Define the number, appearance, and labeling style of regions that the block displays

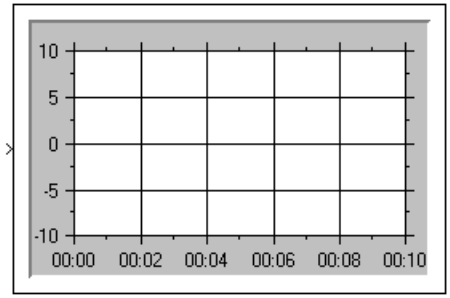
The Block Parameters dialog box governs the relationship between the Simulink block and the control embedded in the block. See “Block Parameters for the ActiveX Control Block” for details.

Strip Chart

Display stream of data in real time

Description

The Strip Chart library contains a single ActiveX Control block configured to display an ActiveX control from Global Majic, the Strip Chart. This block displays one or more signals while the simulation runs. It also enables you to zoom in or out.



Strip Chart

To learn how to use the Strip Chart block, see “Strip Chart Library”.

Dialog Box

The ActiveX Control Properties dialog box governs the appearance and functionality of the control that is embedded in the block. The table below lists the panels of the ActiveX Control Properties dialog box.

Panel	Purpose
Background	Configure the background and outline of the block
Captions	Display annotations on the block
Fonts	Define text styles (The **Captions** and **Stamps** panels use the **FontID** or **Stamp FontID** property to reference the styles defined here.)

Panel	Purpose
General	Define the appearance and behavior of the underlying plotting area
Library	Refer to property settings as a named collection
Stamps	Define the appearance of a symbol that you can place on the control or on an individual plot
Track Bands	Define the number of colored bands displayed on each individual plot, and the appearance of each band
Tracks	Define the number of individual plots, and the appearance of each (The **Track Bands** and **Variables** panels use the **TrackID** property to reference the tracks defined here.)
Variables	Determine which variables appear in each individual plot and how each variable is displayed.
X Axis	Determine what the values along the *x*-axis represent and how they are displayed

The Block Parameters dialog box governs the relationship between the Simulink block and the control embedded in the block. See “Block Parameters for the ActiveX Control Block” for details.