

Using the Embedded MATLAB Function Block

The Embedded MATLAB Function block lets you use MATLAB code in models intended to be deployed as standalone executables generated by Real-Time Workshop. The following sections explain how to use the block to create such models.

Introduction to Embedded MATLAB Function Blocks (p. 16-2)

Overview of the use of Embedded MATLAB Function blocks in Simulink.

Creating an Example Embedded MATLAB Function (p. 16-6)

How to create an example Simulink model with an Embedded MATLAB Function block that you program.

Debugging an Embedded MATLAB Function (p. 16-20)

How to debug the Embedded MATLAB function for the example model you create in the previous section.

The Embedded MATLAB Function Editor (p. 16-32)

Reference of operations available in the Embedded MATLAB Editor.

Typing Function Arguments (p. 16-68)

How to specify argument types for Embedded MATLAB functions.

Sizing Function Arguments (p. 16-81)

How to specify argument sizes for Embedded MATLAB functions.

Parameter Arguments in Embedded MATLAB Functions (p. 16-85)

How to pass Simulink parameters and MATLAB variables as arguments to an Embedded MATLAB Function block.

Introduction to Embedded MATLAB Function Blocks

This section introduces the Embedded MATLAB Function block in Simulink. Use the following topics to get an overview of Embedded MATLAB Function blocks, and how and why they are used in Simulink.

- “What Is an Embedded MATLAB Function Block?” on page 16-2
- “Why Use Embedded MATLAB Function Blocks?” on page 16-4

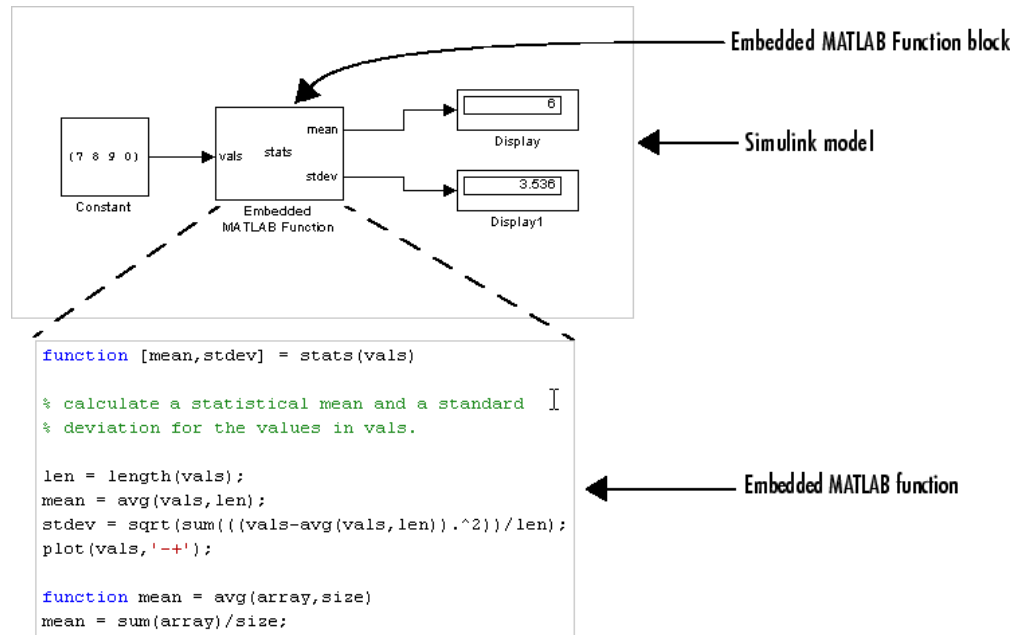
In “Creating an Example Embedded MATLAB Function” on page 16-6, you build a model with an example Embedded MATLAB Function block.

Note For more information on fixed-point support in Embedded MATLAB, refer to “Using the Fixed-Point Toolbox with Embedded MATLAB” in the Fixed-Point Toolbox documentation.

What Is an Embedded MATLAB Function Block?

The Embedded MATLAB Function block contains a MATLAB function in a Simulink model. The function accepts multiple input signals and produces multiple output signals.

You build the following model in “Creating an Example Embedded MATLAB Function” on page 16-6:



As in a MATLAB function, in the Embedded MATLAB Function block, you can declare a local variable implicitly through assignment. The variable takes its type and size from the context in which it is assigned. For example, the following code line declares `x` to be a scalar variable of type double.

```
x = 1.54;
```

Once you define a variable, it cannot be redefined to any other type or size in the function body. For example, you cannot declare `x` and reassign it as follows:

```
x = 2.65; % OK: x is a scalar double
x = [x 2*x]; % Error: x cannot be changed to a vector
```

See “Creating Local Variables Implicitly” for detailed descriptions and examples.

In addition to supporting a rich subset of the MATLAB language, an Embedded MATLAB Function block can call any of the following functions:

- **Subfunctions**

Subfunctions are defined in the body of the Embedded MATLAB block. In the preceding example, `avg` is a subfunction.

- **Embedded MATLAB runtime library functions**

Embedded MATLAB runtime library functions are a subset of the functions that you call in MATLAB. When you build your model with Real-Time Workshop, these functions generate C code that conforms to the memory and variable type requirements of embedded environments. In the preceding example, `length`, `sqrt`, and `sum` are Embedded MATLAB runtime library functions.

- **MATLAB functions**

Function calls that cannot be resolved as subfunctions or Embedded MATLAB runtime library functions are resolved in the MATLAB workspace. These functions do not generate code; they execute only in the MATLAB workspace during simulation of the model.

Why Use Embedded MATLAB Function Blocks?

There are many reasons to use Embedded MATLAB Function blocks in your Simulink models. Here are just a few of them:

- **Embedded MATLAB Function blocks can build standalone simulation applications** -- To support code generation in Real-Time Workshop, the Embedded MATLAB Function block supports a subset of MATLAB commands that generate efficient C code. If you limit the function calls in Embedded MATLAB functions to subfunctions and Embedded MATLAB runtime library functions, you can use Real-Time Workshop to build simulation executables that execute without the MATLAB environment.
- **Embedded MATLAB Function blocks have multiple inputs and outputs** — Unlike MATLAB Fcn blocks, which take a vector input of values and support a single scalar output, the functions in Embedded MATLAB Function blocks accept multiple inputs and return multiple outputs.

- **Embedded MATLAB Function blocks inherit Simulink input and output signals** — By default, both the size and type of input and output signals to an Embedded MATLAB Function block are inherited from Simulink signals. You can also choose to specify the size and type of inputs and outputs explicitly in the **Model Explorer** or **Ports and Data Manager** (see “Ports and Data Manager” on page 16-40).
- **Embedded MATLAB Function blocks have the full power of MATLAB** — Using Embedded MATLAB Function blocks, developers now have access to a wide and growing variety of sophisticated mathematical applications for the embedded environment. You can choose to limit the function calls in Embedded MATLAB Function functions to subfunctions and Embedded MATLAB runtime library functions that generate efficient C code. However, for simulation applications you can call MATLAB functions directly. You can also mix function calls between runtime library functions and MATLAB functions.

Creating an Example Embedded MATLAB Function

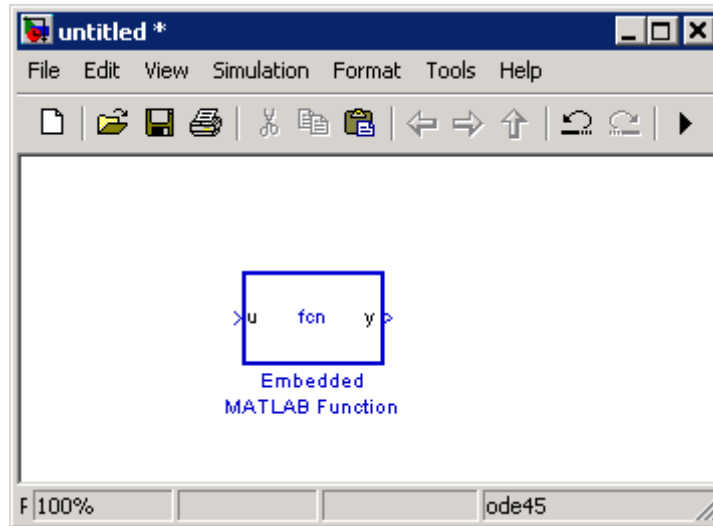
Use the following procedure topics to create a model with an Embedded MATLAB Function block. In the process, learn how to use Embedded MATLAB Function blocks in Simulink.

- 1** “Adding an Embedded MATLAB Function Block to a Model” on page 16-6 -- Start by creating a model with an Embedded MATLAB Function block.
- 2** “Programming the Embedded MATLAB Function” on page 16-8 -- Describes how to program an Embedded MATLAB function. Provides an overview of program syntax, how to use local variables, and how to write callable functions.
- 3** “Checking the Function for Errors” on page 16-13 -- Use the built-in diagnostics for Embedded MATLAB Function blocks to test for syntax errors in the Embedded MATLAB function body.
- 4** “Defining Inputs and Outputs” on page 16-16 -- Define properties for the input and output arguments of the Embedded MATLAB Function block interface with the **Model Explorer** or **Ports and Data Manager** (see “Ports and Data Manager” on page 16-40).

Adding an Embedded MATLAB Function Block to a Model

Start by creating an empty Simulink model and filling it with an Embedded MATLAB Function block, and other blocks necessary to complete the model.

- 1 Create a new Simulink model and add an Embedded MATLAB Function block to it from the User-Defined Function library of the Simulink library.



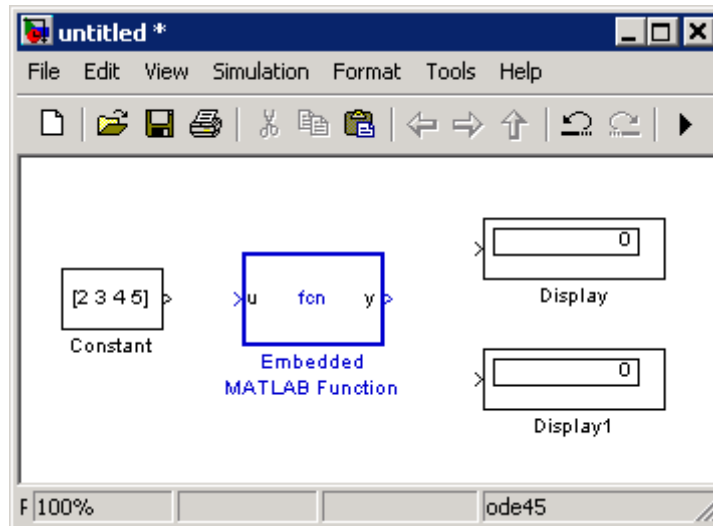
An Embedded MATLAB Function block has two names. The name in the middle of the block is the name of the function you build for the Embedded MATLAB Function block. Its name defaults to `fcn`. The name at the bottom of the block is the name of the block itself. Its name defaults to Embedded MATLAB Function.

The default Embedded MATLAB Function block has an input port and an output port. The input port is associated with the input argument `u`, and the output port is associated with the output argument `y`.

- 2 Add the following Source and Sink blocks to the model:

- From the Simulink Sources library, add a Constant block to the left of the Embedded MATLAB Function block and set its value to the vector `[2 3 4 5]`.
- From the Simulink Sinks library, add two Display blocks to the right of the Embedded MATLAB Function block.

The model should now have the following appearance:



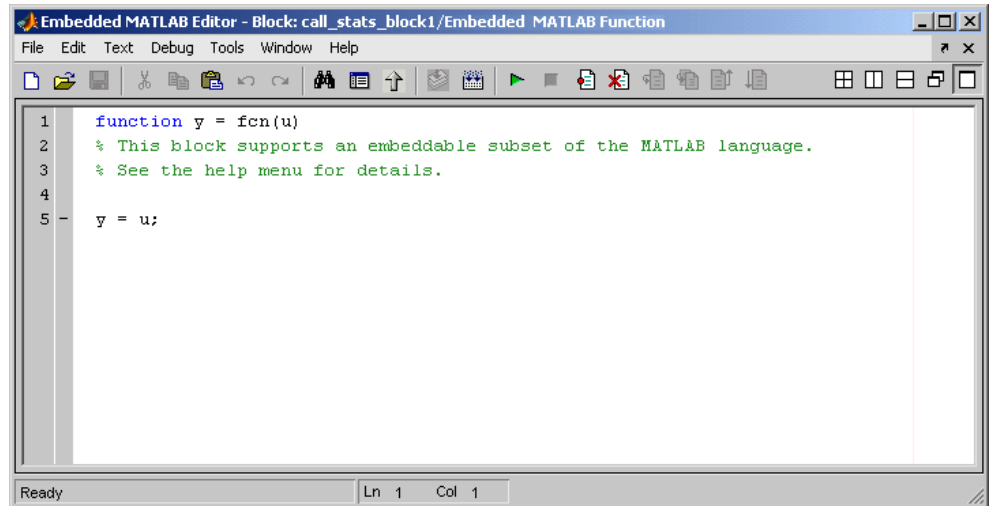
- 3 In the Simulink window, from the **File** menu, select **Save As** and save the model as `call_stats_block1`.

Programming the Embedded MATLAB Function

You create a model with an Embedded MATLAB Function block in “Creating an Example Embedded MATLAB Function” on page 16-6. Now you want to add code to the block to define it as a function that takes a vector set of values and calculates the mean and standard deviation for those values. Use the following steps to program the function stats:

- 1 If not already open, open the `call_stats_block1` model that you save at the end of “Adding an Embedded MATLAB Function Block to a Model” on page 16-6, and double-click its Embedded MATLAB Function block `fcn` to open it for editing.

The **Embedded MATLAB Editor** appears.



The **Embedded MATLAB Editor** window is titled with the syntax *<model name>/<Embedded MATLAB Function block name>* in its header. In this example, the model name is `call_stats_block1`, and the block name is `Embedded MATLAB Function`, the name that appears at the bottom of the `Embedded MATLAB Function` block in Simulink.

Inside the **Embedded MATLAB Editor** is an edit window for editing the function that specifies the `Embedded MATLAB Function` block. A function header with the function name `fcn` is at the top of the edit window. The header specifies an argument to the function, `u`, and a return value, `y`.

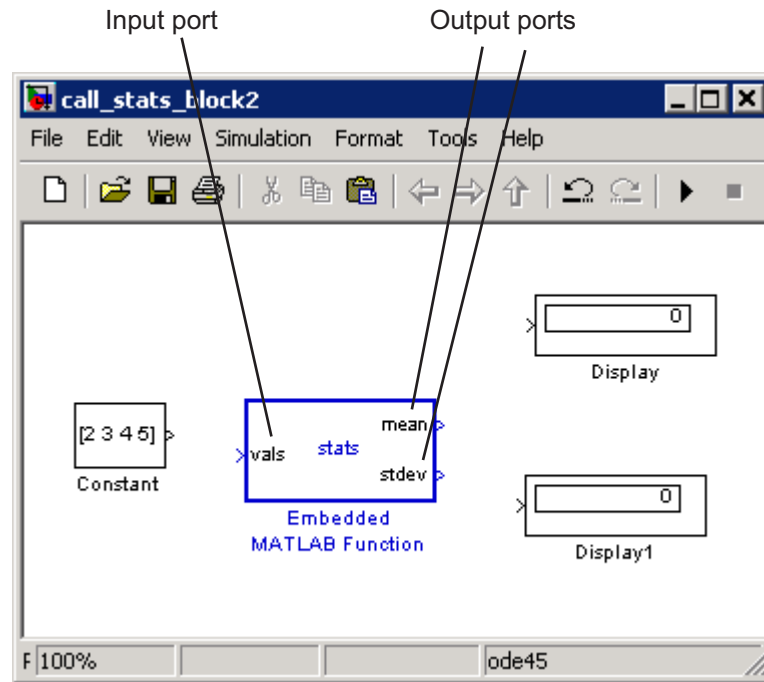
- 2 Edit the function header line with the return values, function name, and argument as follows:

```
function [mean,stdev] = stats(vals)
```

The `Embedded MATLAB` function `stats` calculates a statistical mean and standard deviation for the values in the vector `vals`. The function header declares `vals` to be an argument to the `stats` function and `mean` and `stdev` to be return values from the function.

- 3 In the **Embedded MATLAB Editor**, from the **File** menu, select **Save As Model** and save the model as `call_stats_block2`.

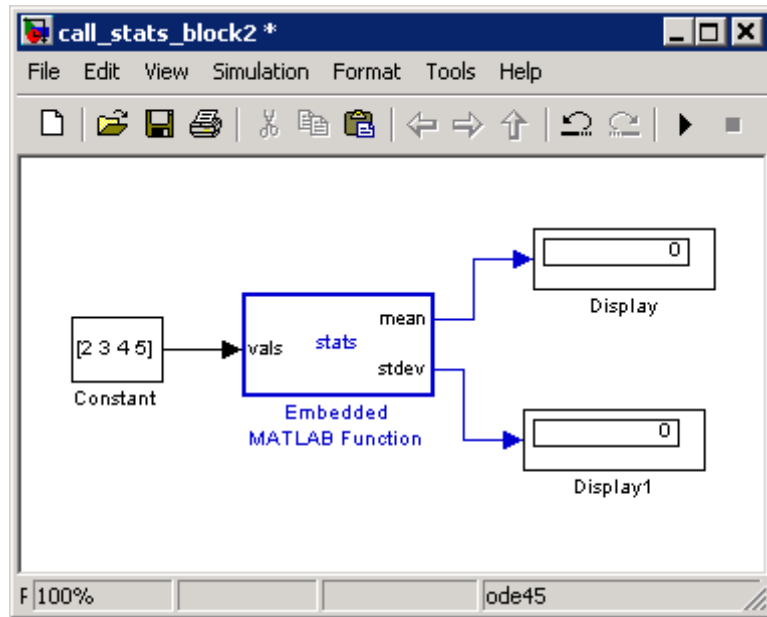
Saving the model updates the Simulink window, which now looks like this:



Changing the function header of the Embedded MATLAB Function block makes the following changes to the Embedded MATLAB Function block in the Simulink model:

- The function name in the middle of the block changes to `stats`.
- The argument `vals` appears as an input port to the block.
- The return values `mean` and `stdev` appear as output ports to the block.

- 4 In the Simulink window, complete connections to the Embedded MATLAB Function block as shown.



- 5 In the **Embedded MATLAB Editor**, enter a line space after the function header and replace the default comment line with the following comment lines:

```
% calculates a statistical mean and a standard
% deviation for the values in vals.
```

You specify comments with a leading percent (%) character, just as you do in MATLAB.

- 6 Enter a line space after the comments and replace the default function line `y = u;` with the following:

```
len = length(vals);
```

The function `length` is an example of a built-in function supported by the runtime function library for Embedded MATLAB Function blocks. This

`length` works just like the MATLAB function `length`. It returns the vector length of its argument `vals`. However, when you simulate this model, Simulink generates C code for this function in the simulation application. Callable functions supported for Embedded MATLAB Function blocks are listed in the topic “Embedded MATLAB Run-Time Function Library”.

The variable `len` is a local variable that is automatically typed as a scalar double because the Embedded MATLAB runtime library function, `length`, returns a scalar of type double. If you want, you can declare `len` to have a different type and size by changing the way you declare it in the function. See “Creating Local Variables Implicitly” for details about implicitly declaring local variables in an Embedded MATLAB Function block.

By default, implicitly declared local variables like `len` are temporary. They come into existence only when the function is called and cease to exist when the function is exited. To persist implicitly declared variables between function calls, see “Declaring Persistent Variables”.

- 7** Enter the following lines to calculate the value of mean and stdev:

```
mean = avg(vals,len);
stdev = sqrt(sum(((vals-avg(vals,len)).^2))/len);
```

`stats` stores the mean and standard deviation values for the values in `vals` in the variable `mean` and `stdev`, which are output by port to the Display blocks in the Simulink model. The line that calculates `mean` calls a subfunction, `avg`, that has not been defined yet. The line that calculates `stdev` calls the Embedded MATLAB runtime library functions `sqrt` and `sum`.

- 8** Enter the following line to plot the input values in `vals`.

```
plot(vals, '-+');
```

This line calls the function `plot` to plot the input values sent to `stats` against their vector indices. Because the Embedded MATLAB runtime library has no `plot` function, the Embedded MATLAB function cannot resolve this call with a subfunction or an Embedded MATLAB runtime function. Instead, it replaces this call with a call to the MATLAB `plot` function in the generated code for the simulation target.

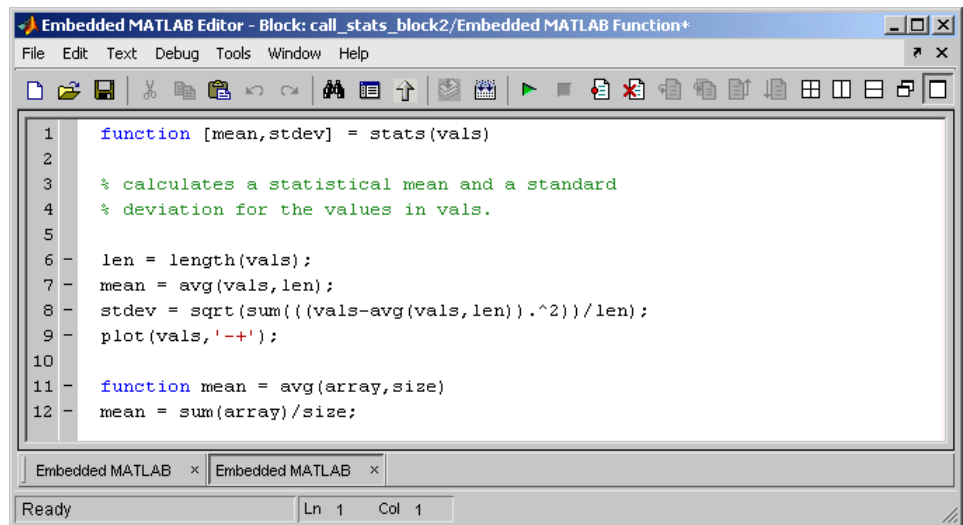
See “Calling MATLAB Functions” for more details on using this mechanism to call MATLAB functions from Embedded MATLAB functions.

- 9 Enter a line space followed by the following lines for the subfunction avg, which is called in an earlier line.

```
function mean = avg(array,size)
mean = sum(array)/size;
```

These two lines define the subfunction avg. You are free to use subfunctions in Embedded MATLAB function code with single or multiple return values, just as you do in regular MATLAB functions.

The **Embedded MATLAB Editor** should now have the following appearance:



- 10 Save the model again as call_stats_block2.

Checking the Function for Errors

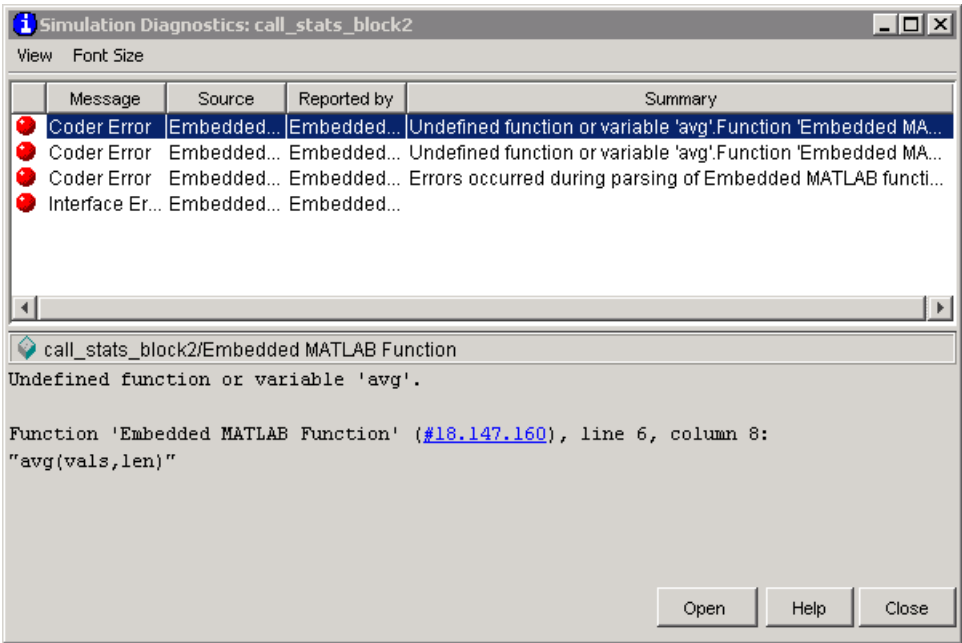
Once you finish specifying an Embedded MATLAB Function block in its Simulink model, use the built-in diagnostics of Embedded MATLAB Function blocks to test for syntax errors with the following procedure:

- 1 If not already open, open the `call_stats_block2` model that you save at the end of “Programming the Embedded MATLAB Function” on page 16-8, and double-click its Embedded MATLAB Function block stats to open it for editing.
- 2 In the **Embedded MATLAB Editor**, click the **Build** icon to compile and build the example Simulink model:



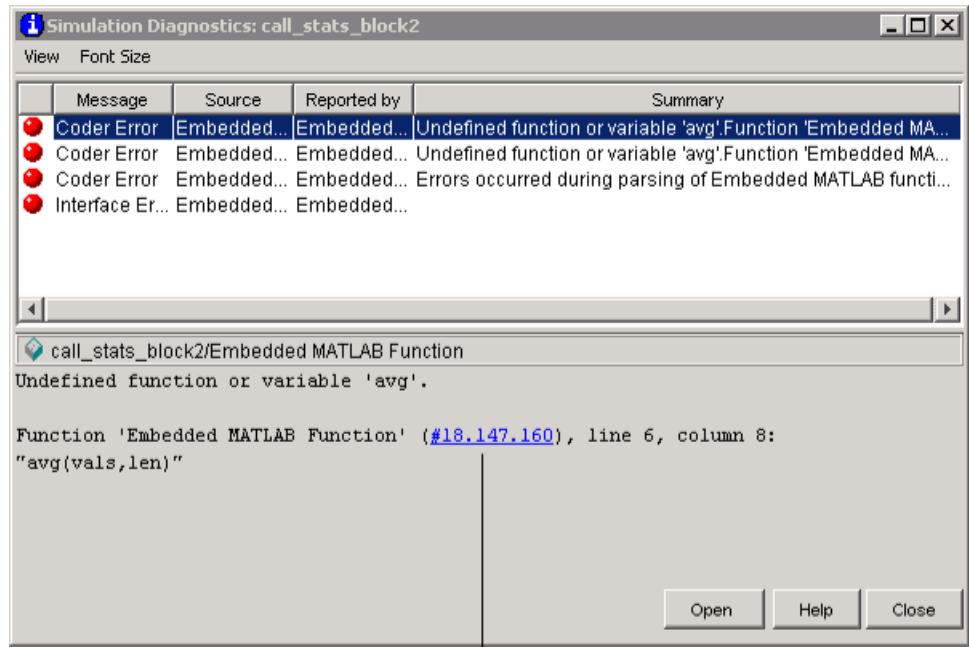
If errors are found, the **Diagnostics Manager** window lists them. Otherwise, nothing happens.

- 3 For example, change the subfunction `avg` to a fictitious subfunction `aug` and then compile to see the following messages in the **Diagnostics Manager** window:



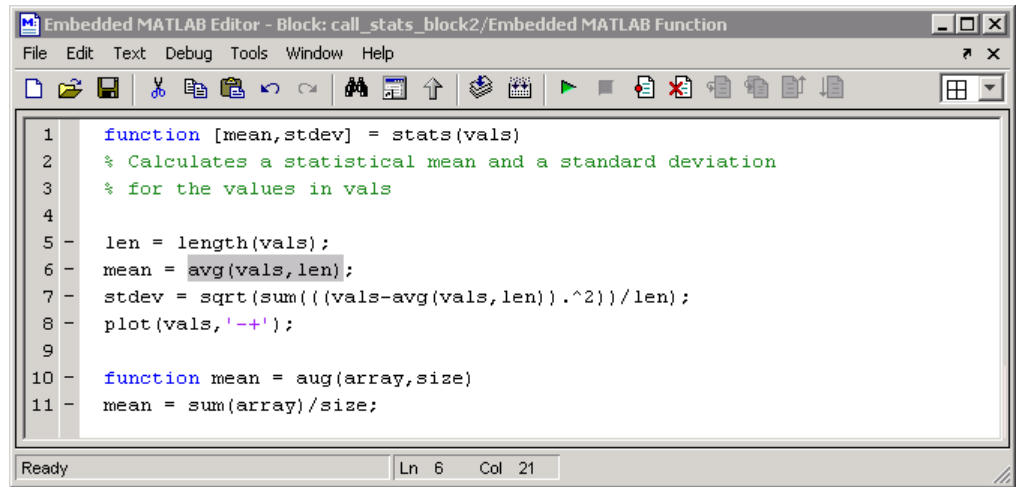
Each detected error appears with a red button.

- 4 Click the first error line to display its message.
- 5 In the diagnostic message for the selected error, click the blue link to find the offending code:



Click to display the offending code
in the Embedded MATLAB Editor

The offending line appears highlighted in the **Embedded MATLAB Editor**:



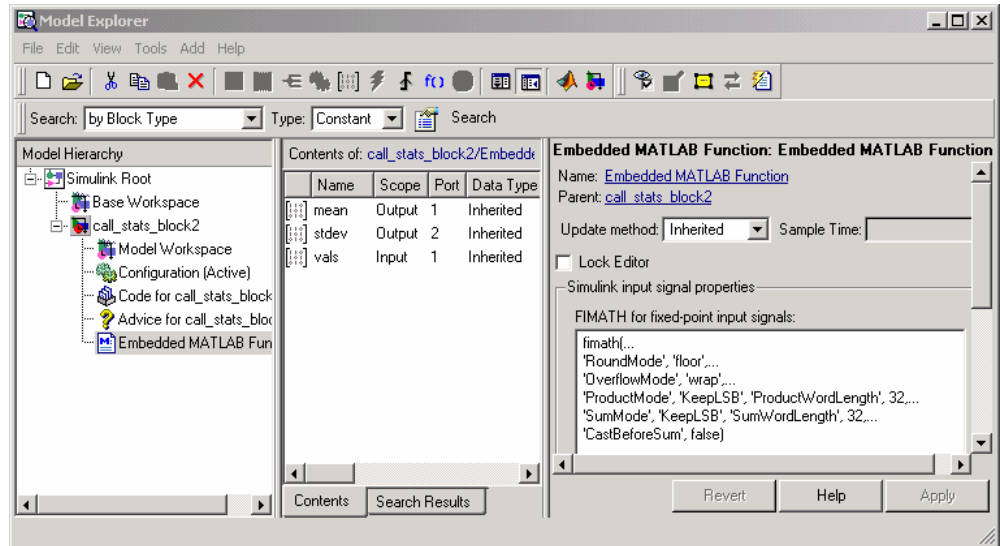
6 Correct the error and recompile.

Defining Inputs and Outputs

In the stats function header for the Embedded MATLAB Function block you define in “Programming the Embedded MATLAB Function” on page 16-8, the function argument vals is an input and mean and stdev are outputs. By default, function inputs and outputs inherit their data type and size from the Simulink signals attached to their ports. In this topic, you examine input and output data for the Embedded MATLAB Function block to verify that it inherits the correct type and size.

- 1** If not already open, open the call_stats_block2 model that you save at the end of “Programming the Embedded MATLAB Function” on page 16-8, and double-click its Embedded MATLAB Function block stats to open it for editing.
- 2** In the **Embedded MATLAB Editor**, select **Tools > Model Explorer** to open the **Model Explorer**.

The **Model Explorer** window opens:

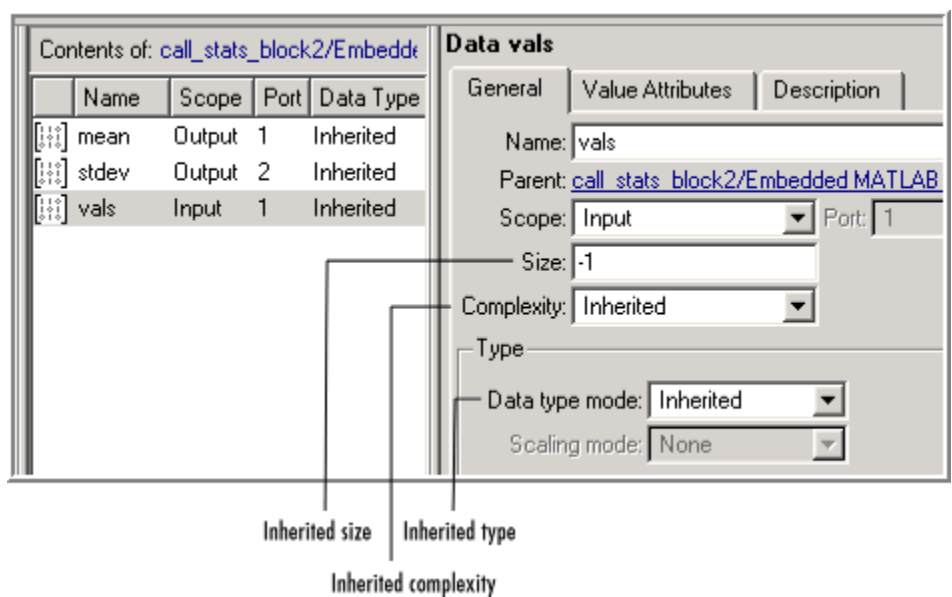


You can use the **Model Explorer** to define arguments for Embedded MATLAB Function blocks. Notice that the Embedded MATLAB Function block Embedded MATLAB is highlighted in the left **Model Hierarchy** pane.

The **Contents** pane displays the argument vals and the return values mean and stdev that you have already created for the Embedded MATLAB Function block. Notice that vals is assigned a **Scope** of Input, which is short for **Input from Simulink**. mean and stdev are assigned the **Scope** of Output, which is short for **Output to Simulink**.

You can also use the **Ports and Data Manager** to define arguments for Embedded MATLAB Function blocks (see “Ports and Data Manager” on page 16-40).

- 3 In the **Contents** pane of the **Model Explorer** window, click anywhere in the row for **vals** to highlight it:



The right pane displays the **Data** properties dialog box for **vals**. By default, the type, size, and complexity of input and output arguments are inherited from the Simulink signals attached to each input or output port. Inheritance is specified by setting **Type** and **Complexity** to **Inherited**, and **Size** to **-1**:

The actual inherited values for size and type are set during compilation of the model, and are reported in the **Compiled Type** and **Compiled Size** columns of the **Contents** pane. You compile and build the model by clicking the **Build** icon:



You can specify the type of an input or output argument directly by selecting a type in the **Type** field of the **Data** properties dialog box, for example, **double**. You can also specify the size of an input or output argument directly by entering an expression in the **Size** field of the **Data** properties

dialog box for the argument. For example, you can enter [2 3] in the **Size** field to size vals as a 2-by-3 matrix. See “Typing Function Arguments” on page 16-68 and “Sizing Function Arguments” on page 16-81 for more information on the expressions that you can enter for type and size.

Note The default first index for any arrays that you add to an Embedded MATLAB Function block function is 1, just as it would be in MATLAB.

Debugging an Embedded MATLAB Function

In “Creating an Example Embedded MATLAB Function” on page 16-6, you create and specify an example Simulink model with an Embedded MATLAB Function block. You use this block to specify an Embedded MATLAB function stats that calculates the mean and standard deviation for a set of input values. In this section, you debug stats in the example model.

Use the following topics to learn how to debug an Embedded MATLAB function in Simulink:

- “Debugging the Function in Simulation” on page 16-20 — Executes the model in simulation and tests the Embedded MATLAB function stats.
- “Watching Function Variables During Simulation” on page 16-28 — Describes tools that you can use to view the values of Embedded MATLAB variables during simulation.
- “How Exiting Debug Mode Affects Simulation” on page 16-31 — Describes how exiting debug mode affects simulation of the Embedded MATLAB function

Debugging the Function in Simulation

You can debug your Embedded MATLAB Function block just like you can debug a function in MATLAB. In simulation, you test your Embedded MATLAB functions for runtime errors with tools similar to the MATLAB debugging tools.

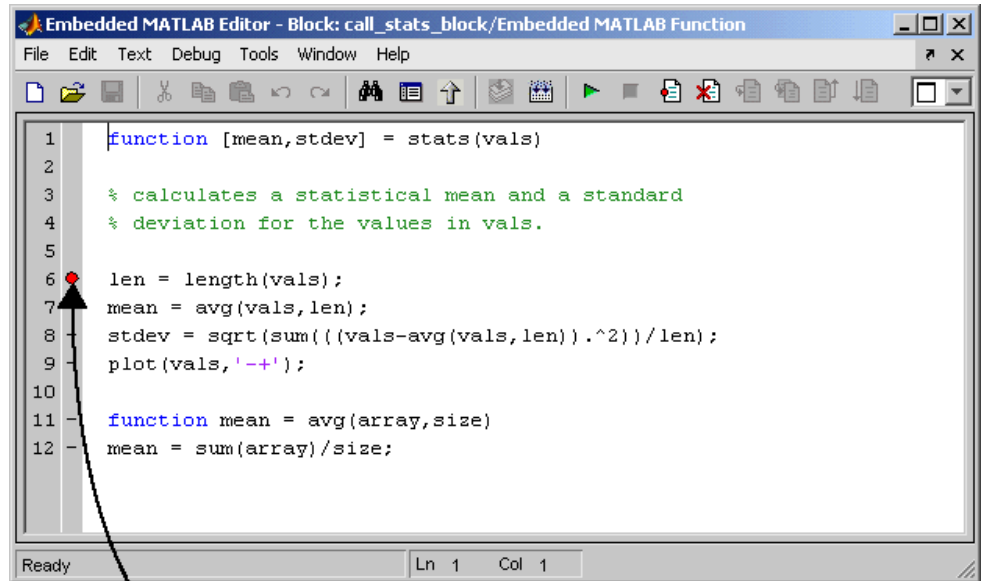
When you start simulation of your model, Simulink checks to see if the Embedded MATLAB Function block has been built since creation, or since a change has been made to the block. If not, it performs the build described in “Checking the Function for Errors” on page 16-13. If no diagnostic errors are found, Simulink begins the simulation of your model.

Use the following procedure to debug the stats Embedded MATLAB function during simulation of the model:

- 1 If not already open, open the `call_stats_block2` model that you save at the end of “Programming the Embedded MATLAB Function” on page 16-8,

and double-click its Embedded MATLAB Function block stats to open it for editing in the **Embedded MATLAB Editor**.

- 2 In the **Embedded MATLAB Editor**, in the left margin of line 6, click the dash (-) character.



Breakpoint indicator

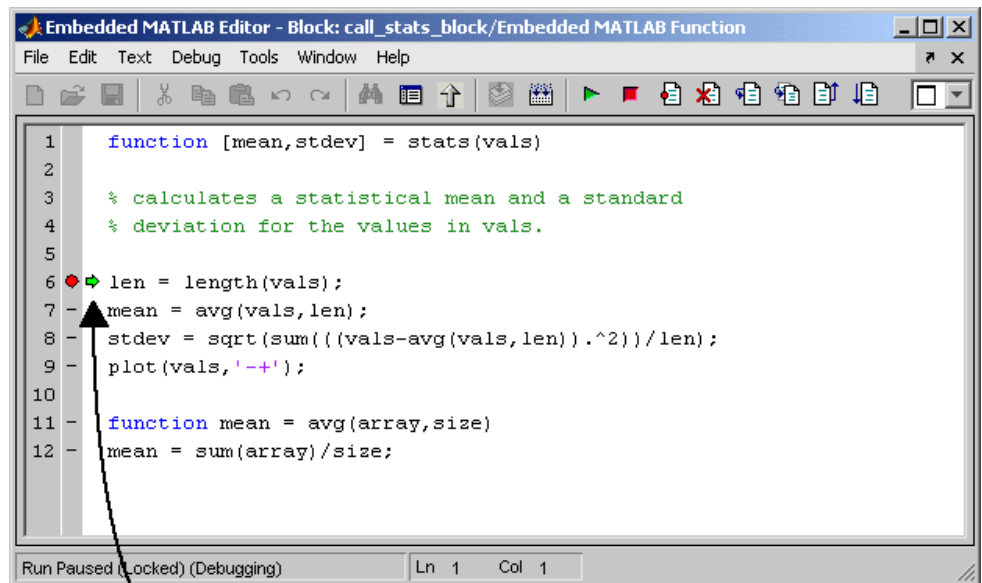
A small red ball appears in the margin of line 6, indicating that you have set a breakpoint. You can also use the **Set/Clear Breakpoint** icon to insert the breakpoint on the line where the cursor is positioned.



- 3 Click the **Start Simulation** icon to begin simulating the model:



If you get any errors or warnings, make corrections before you try to simulate again. Otherwise, simulation pauses when execution reaches the breakpoint you set. This is indicated by a small green arrow in the left margin, as shown.



Execution pauses prior to next step

- 4 In the **Embedded MATLAB Editor** window, click the **Step** icon to advance execution:



The execution arrow advances one line to line 7 of stats.

You can also step execution by entering `dbstep` at the Command Line Debugger. See “Watching with the Command Line Debugger” on page 16-29 for a description of the Command Line Debugger in MATLAB.

Notice that line 7 calls the subfunction avg. If you click **Step** here, execution advances to line 8, past the execution of the subfunction avg. To track execution of the lines in the subfunction avg, you need to click the **Step In** icon.

5 Click the **Step In** icon:



Execution advances to enter the subfunction avg:

```

1  function [mean,stdev] = stats(vals)
2
3  % calculates a statistical mean and a standard
4  % deviation for the values in vals.
5
6  len = length(vals);
7  mean = avg(vals,len);
8  stdev = sqrt(sum((vals-avg(vals,len)).^2)/len);
9  plot(vals,'-+');
10
11 function mean = avg(array,size)
12 mean = sum(array)/size;

```

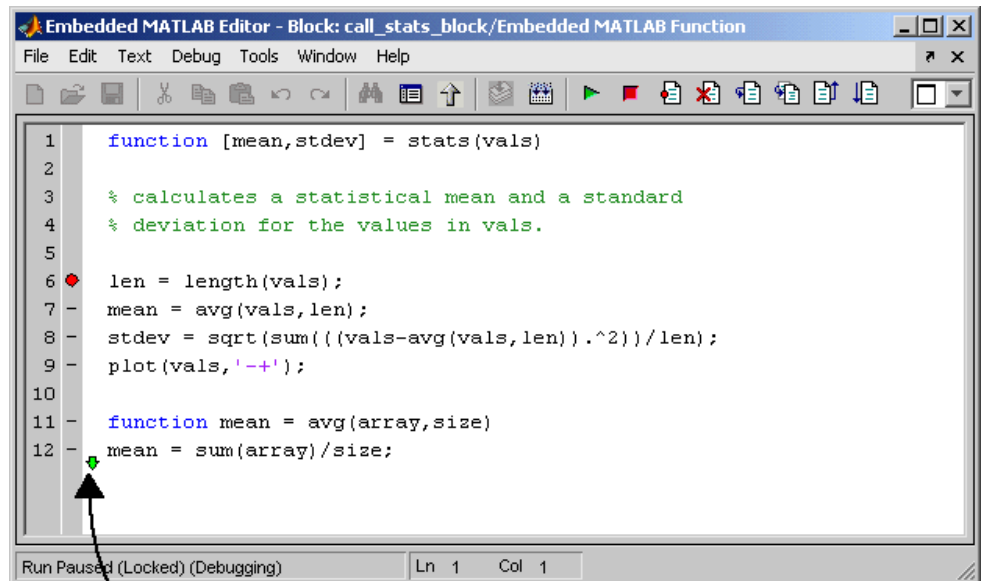
Run Paused (Locked) (Debugging) Ln 1 Col 1

Once you are in a subfunction, you can use the **Step** or **Step In** icon to advance execution. If the subfunction calls another subfunction, use the **Step In** icon to enter it. If you want to execute the remaining lines of the subfunction, click the **Step Out** icon:



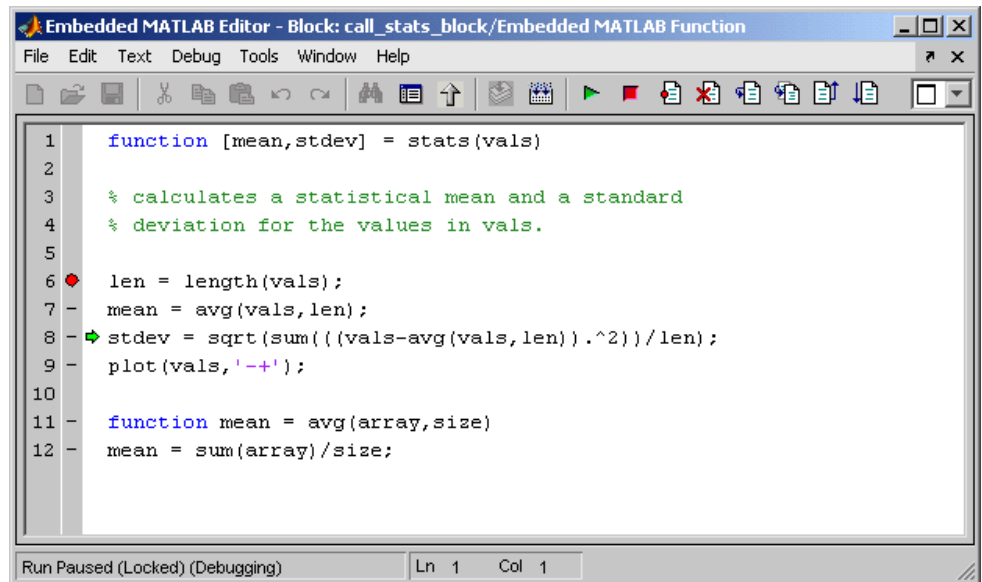
- 6 Click the **Step** icon to execute the only line in the subfunction avg.

The subfunction avg finishes its execution, and you see a green arrow pointing down under its last line as shown.



Subfunction completed

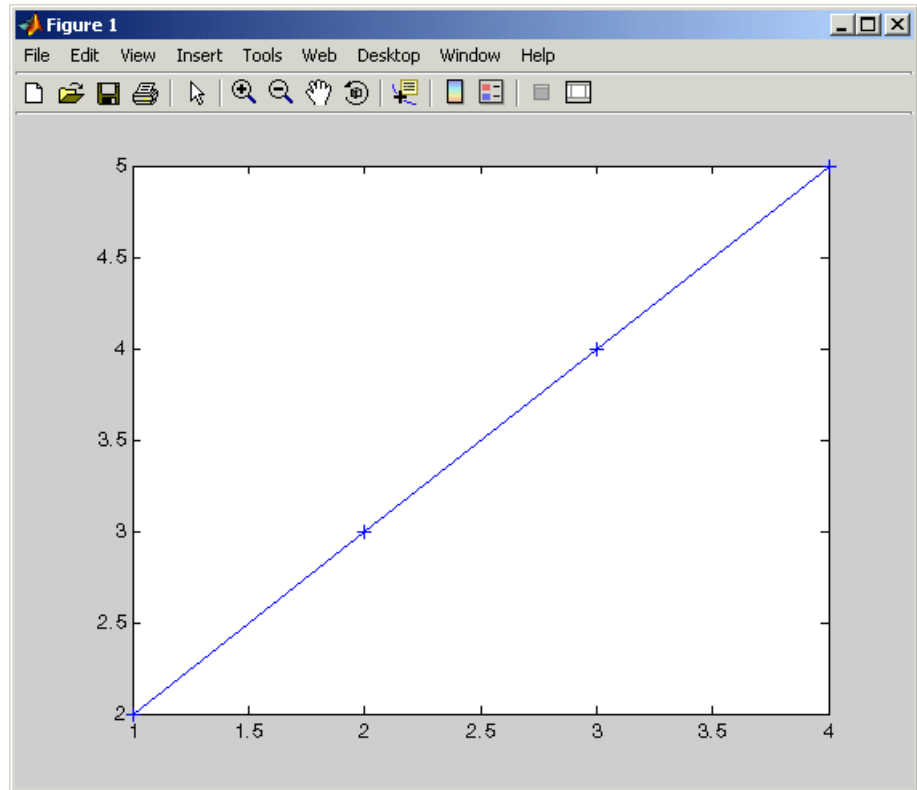
- 7 Click the **Step** icon to return to the function stats.



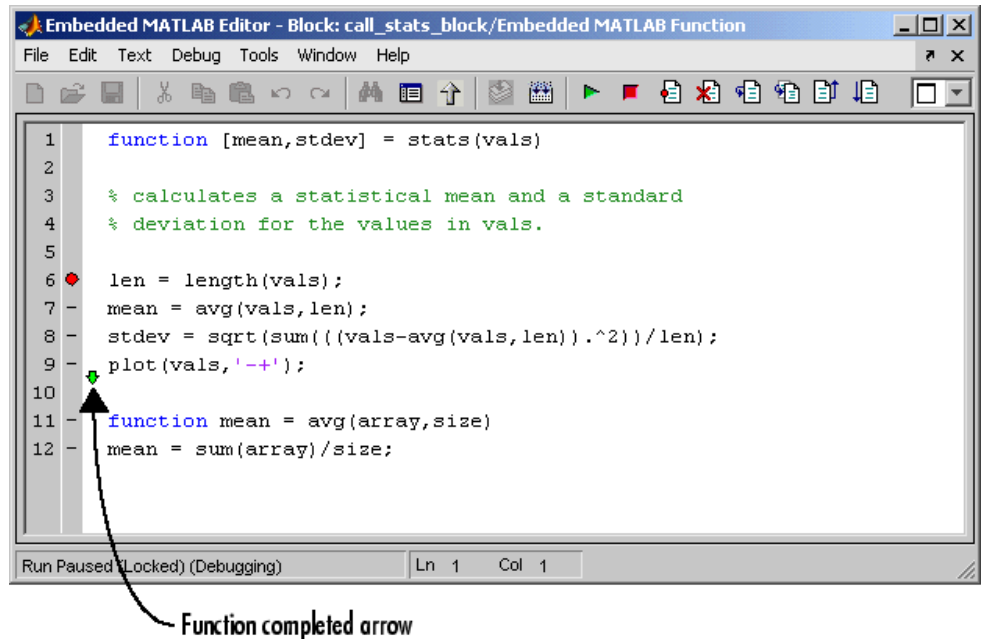
Execution advances to the line after to the call to the subfunction avg, line 8.

- 8** Click **Step** twice to execute line 8 and the plot function in line 9.

The plot function executes in MATLAB, and you see the following plot.



In the **Embedded MATLAB Editor**, a green arrow points down under line 9, indicating the completion of the function stats.



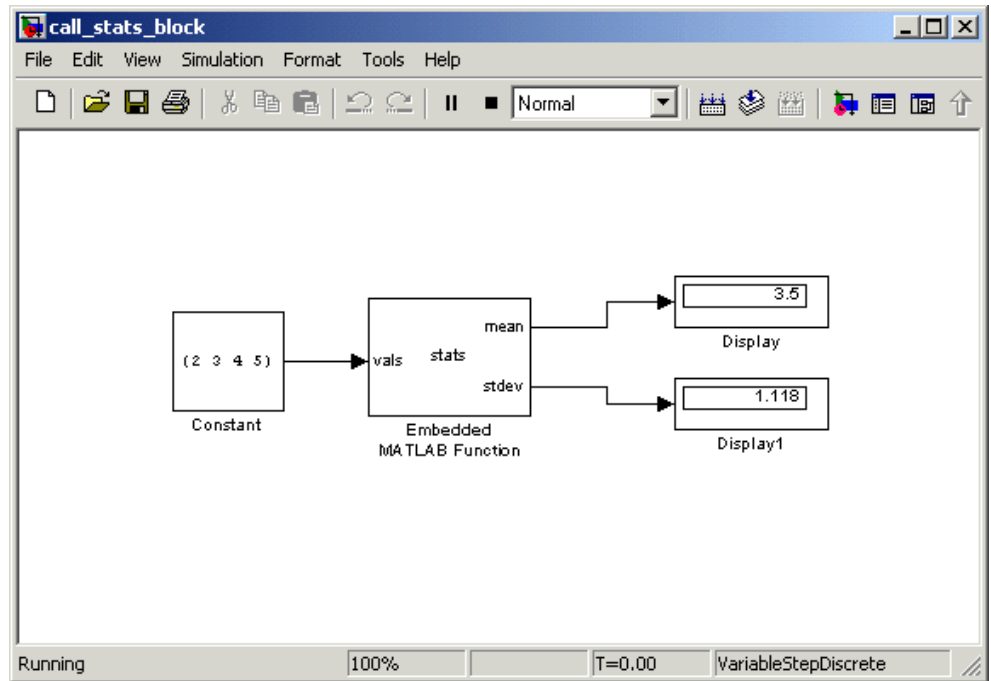
- 9 Click the **Continue Debugging** icon to continue execution of the model.



At any point in a function, you can advance through the execution of the remaining lines of the function with the **Continue Debugging** icon. If you are at the end of the function, clicking the **Step** icon accomplishes the same thing.

You can also continue execution by entering `dbcont` at the Command Line Debugger. See “Watching with the Command Line Debugger” on page 16-29 for a description of the Command Line Debugger in MATLAB.

In the Simulink window, the computed values of mean and stdev now appear in the Display blocks.



- 10 In the **Embedded MATLAB Editor**, click the **Exit Debug Mode** icon to stop simulation:



For more information, see “How Exiting Debug Mode Affects Simulation” on page 16-31.

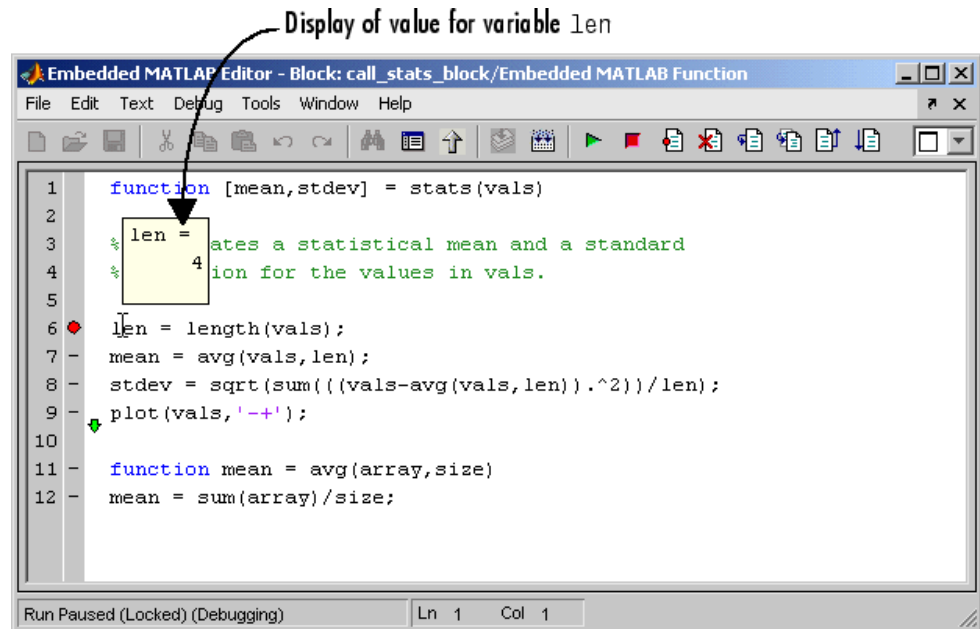
Watching Function Variables During Simulation

While you are simulating the function of an Embedded MATLAB Function block, you can use several tools to keep track of variable values in the function. These tools are described in the topics that follow.

Watching with the Interactive Display

To display the value of a variable in the function of an Embedded MATLAB Function block during simulation, in the **Embedded MATLAB Editor**, place the mouse cursor over the variable text and observe the pop-up display.

For example, to watch the variable `len` during simulation, place the mouse cursor over the text `len` in line 6 for at least a second. The value of `len` appears adjacent to the cursor, as shown:



You can display the value for any variable in the Embedded MATLAB function in this way, no matter where it appears in the function.

Watching with the Command Line Debugger

You can report the values for an Embedded MATLAB function variable with the Command Line Debugger utility in the MATLAB window during simulation. When you reach a breakpoint, press **Enter** in the MATLAB window and the Command Line Debugger prompt, `debug>>`, appears. At

this prompt, you can see the value of a variable defined for the Embedded MATLAB Function block by entering its name:

```
debug>> stdev
```

```
1.1180
```

```
debug>>
```

The Command Line Debugger also provides the following commands during simulation:

Command	Description
dbstep	Advance to next program step after a breakpoint is encountered.
dbcont	Continue execution to next breakpoint.
dbquit	Stop simulation of the model. Press Enter after this command to return the MATLAB prompt.
help	Display help for command line debugging.
print x	Display the value of the variable x. If x is a vector or matrix, you can also index into x. For example, x(1,2).
save	Saves all variables to the specified file. Follows the syntax of the MATLAB save command. To retrieve variables to the MATLAB base workspace, use load command after simulation has been ended.
whos	Display the size and class (type) of all variables in the scope of the halted Embedded MATLAB Function block.

You can issue any other MATLAB command at the debug>> prompt, but the results are executed in the workspace of the Embedded MATLAB Function block. To issue a command in the MATLAB base workspace at the debug>> prompt, use the evalin command with the first argument 'base' followed by the second argument command string, for example, evalin('base','whos'). To return to the MATLAB base workspace, use the dbquit command.

Watching with MATLAB

You can display the execution result of an Embedded MATLAB function line by omitting the terminating semicolon. If you do, execution results for the line are echoed to the MATLAB window during simulation.

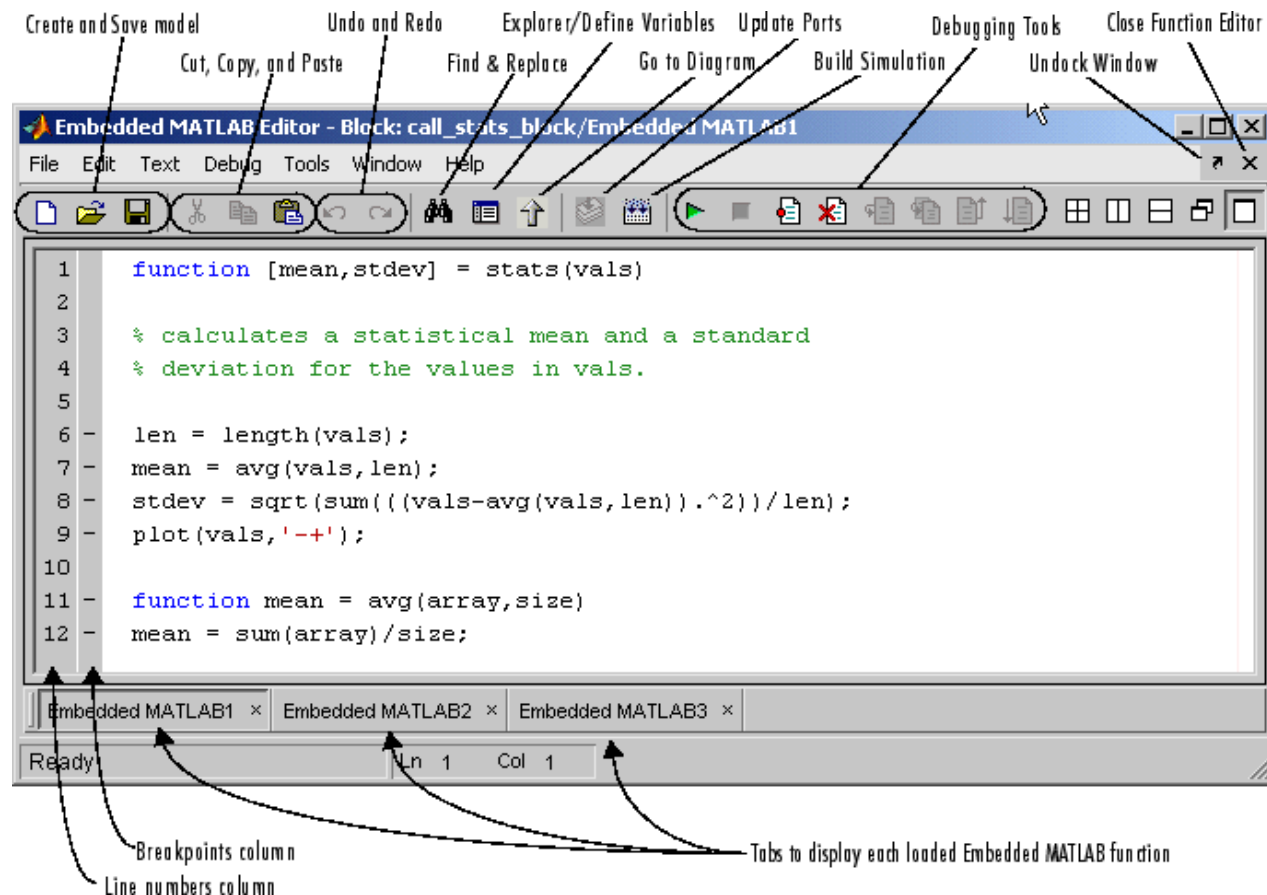
How Exiting Debug Mode Affects Simulation

The behavior of the **Exit Debug Mode** option depends on the run-time context, as follows:

If You Exit Debug Mode	What Happens
When the Embedded MATLAB function is stopped at a breakpoint	Simulation stops immediately
From the Embedded MATLAB Editor	Simulation stops when the Embedded MATLAB Function block finishes executing
From the Simulink Editor	Simulation stops immediately
By typing Ctrl-C when the Simulink Editor window has focus	Simulation stops immediately

The Embedded MATLAB Function Editor

You edit an Embedded MATLAB function to specify its function header and body. When you open an unspecified Embedded MATLAB function for editing, it has the following default appearance in the **Embedded MATLAB Editor**:



This section provides the following topics to describe tools for editing Embedded MATLAB functions in the **Embedded MATLAB Editor**:






Changing the Embedded MATLAB Editor (p. 16-33)	Describes how to customize the appearance of the editor
Editing the Embedded MATLAB Function (p. 16-37)	Shows how to edit the contents of the function in the Embedded MATLAB Editor
Defining Embedded MATLAB Function Arguments (p. 16-39)	Describes how to use the Ports and Data Manager , Model Explorer , or the Simulink model to set the size, type, or source of an input or output argument
Ports and Data Manager (p. 16-40)	Explains how to use the Ports and Data Manager to define data arguments, input triggers, and function-call outputs for an Embedded MATLAB Function block
Debugging Embedded MATLAB Functions (p. 16-66)	Describes how to debug the Embedded MATLAB function during simulation of the model

Changing the Embedded MATLAB Editor

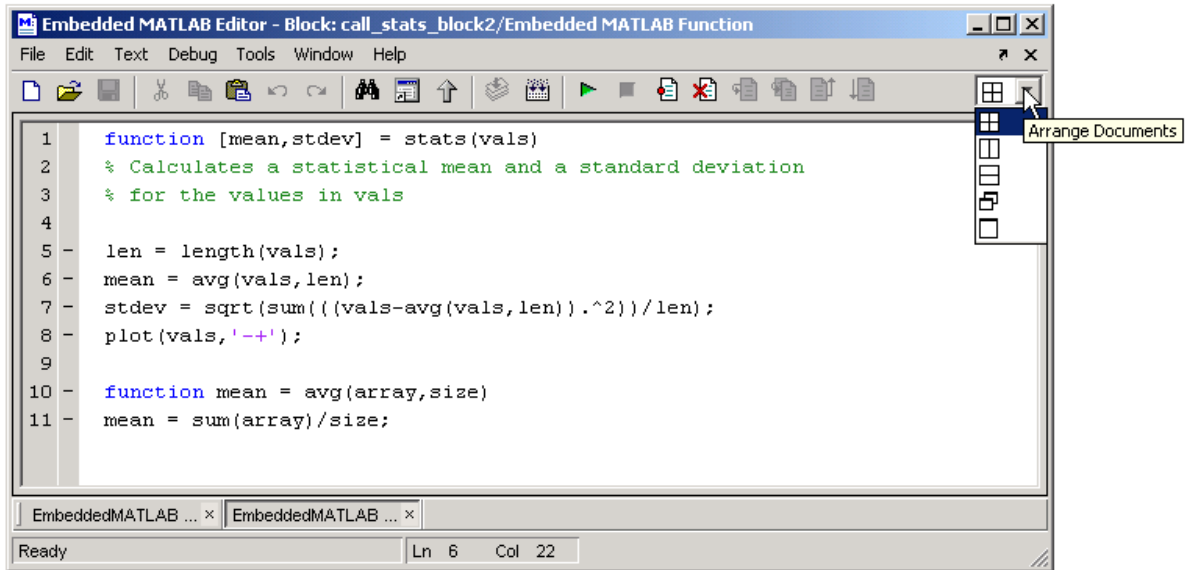
Use the toolbar icons described in the following topics to customize the appearance of the **Embedded MATLAB Editor**.

Displaying Embedded MATLAB Function Windows

By default, if you have more than one Embedded MATLAB function loaded in the **Embedded MATLAB Editor**, only the most recently loaded is displayed. Editing windows for previously loaded functions are accessed individually with tabs in the document bar. You can display the editing windows for all loaded functions simultaneously by selecting one of the following options from the **Window** menu:

Menu Option	Description
 Tile	Tiles all loaded Embedded MATLAB windows into an adjustable matrix of windows. When you select the option, an array of squares representing the tiled windows is available as a submenu. Select an appropriate array of windows.
 Left/Right Tile	Displays the selected window and the next most recently loaded Embedded MATLAB function at full height, side by side.
 Top/Bottom Tile	Displays the selected window and the next most recently loaded Embedded MATLAB function at full width, top to bottom.
 Float	Displays the loaded Embedded MATLAB functions in separate cascading and overlapping windows of the same size.
 Maximize	Displays the Embedded MATLAB function in current focus at the full width and height of the editor. This is the default setting.

These options are also available from the **Arrange Documents** drop-down menu on the right side of the toolbar:



Creating Editors for Embedded MATLAB Functions

When you open more than one Embedded MATLAB function into the **Embedded MATLAB Editor** you can create separate editors for each function by undocking the function from the main editor.

To create a new editor window for the Embedded MATLAB function that is in focus in the **Embedded MATLAB Editor**, select the **Undock Embedded MATLAB Function** icon on the right side of the main editor's menu bar:



To reverse the process, select the **Dock Embedded MATLAB Function** icon on the right side of the editor's menu bar:



Moving the Document Bar

When you edit an Embedded MATLAB function, it is displayed as a window in the **Embedded MATLAB Editor**. You can open more than one Embedded

MATLAB function in the editor. For each function that you open, a tab is added to a document bar at the bottom of the **Embedded MATLAB Editor**. If you want to edit an Embedded MATLAB function not in focus, click its tab.

To change the location of the document bar or hide it altogether, follow these steps:

- 1 Right-click in the document bar and select **Bar Position** from the drop-down menu.
- 2 From the submenu, select **Top**, **Bottom**, **Right**, **Left**, or **Hide**.

Sorting Functions in the Editor

You can sort functions in the **Embedded MATLAB Editor** by right-clicking in the document bar and toggling the **Alphabetize** option on or off.

Toggling the Toolbar Display

By default, the **Embedded MATLAB Editor** has a toolbar with shortcuts to tools that you can access from the menus in the menu bar. You can toggle the toolbar display by following these steps:

- 1 Right-click in the toolbar and select **Embedded MATLAB Editor Toolbar**.

The toolbar disappears.

- 2 Right-click in the menu bar and select **Embedded MATLAB Editor Toolbar**.

The **Toolbar** reappears.

Setting Preferences

You can choose preferences for the **Embedded MATLAB Editor**, such as font size, tab size, and so on, as follows:

- 1 From the **File** menu, select **Preferences**.

The MATLAB **Preferences** dialog box appears.

2 Change preferences in pages accessed only through the following nodes:



- **Fonts**
- **Colors**
- **Display** (under **Editor/Debugger**)
- **Keyboard & Indenting** (under **Editor/Debugger**)

Note The **Embedded MATLAB Editor** is a derivation of the MATLAB editor you use to edit M-files in MATLAB. The preference changes that you specify are made to the MATLAB editor.

Editing the Embedded MATLAB Function

Use the tools in the following topics to edit an Embedded MATLAB function in the **Embedded MATLAB Editor**:

Undoing and Redoing Operations

Tool Button	Description
 Undo	Undo the effects of the preceding operation. Alternatively, from the Edit menu, select Undo .
 Redo	Redo the effects of the most recently undone operation. Alternatively, from the Edit menu, select Redo .

Comment and Uncomment Embedded MATLAB Function Lines

You can comment text or uncomment commented text as follows:

- To turn selected function text lines into commented text lines, from the **Text** menu, select **Comment**.
- To turn selected comment text lines into function text lines, from the **Text** menu, select **Uncomment**.

Any text selected on a line, or the presence of the text cursor, selects the line.

Going to a Specified Line of the Embedded MATLAB Function

To place the text cursor at the beginning of a specified line, from the **Edit** menu, select **Go To Line**. In the resulting dialog box, enter the line number and click **OK**.

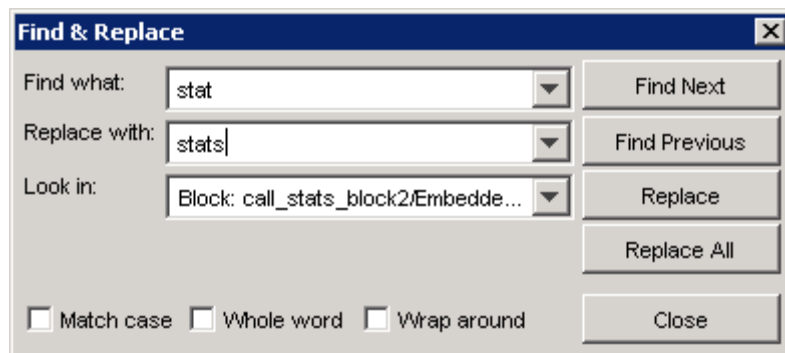
Searching for and Replacing Text in Embedded MATLAB Functions

You can use the Find & Replace icon in the toolbar to search and replace text in the **Embedded MATLAB Editor** as follows.

- 1 Click the **Find & Replace** icon:



The **Find & Replace** dialog box appears.



By default, the **Look in** field is set to search the current Embedded MATLAB function, but you can select from any Embedded MATLAB functions that you have open for editing.

- 2 Enter the search string in the **Find what** field.
- 3 Modify the text you want to search for by checking any or all of the following:

- **Match case** — The text must match the case of the search string.
- **Whole word** — The text must be a whole word and not part of a larger word.
- **Wrap around** — Continue searching after reaching the bottom of the editor. Otherwise, stop searching.

4 Enter the replacement string in the **Replace with** field.

5 Click the **Find Next** or **Find Previous** button to find a single occurrence of the search string.


If the text is present in the **Embedded MATLAB Editor**, it is highlighted with a gray background.



6 Click the **Replace** button to replace the highlighted text in the editor with the replacement string.

7 Click the **Replace All** button to replace every occurrence of the search string with the replacement string.

Defining Embedded MATLAB Function Arguments

Once you edit the Embedded MATLAB function, you can set the size, type, or source of an input or output argument using any of the following tools:

Tool Button	Description
 Edit Data/Ports	<p>Opens the Ports and Data Manager dialog to add or modify arguments for the current Embedded MATLAB function block (see “Ports and Data Manager” on page 16-40). You can also open this dialog by selecting Edit Data/Ports from the Tools menu.</p> <p>To define and modify input and output arguments for any Embedded MATLAB Function block in the model hierarchy, use the Model Explorer, which you can open from the Tools menu.</p>

Tool Button	Description
 Goto Diagram	Displays the Embedded MATLAB function in its native diagram without closing the Embedded MATLAB Editor .
 Update Ports	Updates the ports of the Embedded MATLAB Function block with the latest changes made to the function argument and return values without closing the Embedded MATLAB Editor .




See “Defining Inputs and Outputs” on page 16-16 for an example of defining an input argument for an Embedded MATLAB Function block.

Ports and Data Manager

The **Ports and Data Manager** provides a convenient method for defining objects and modifying their properties for an Embedded MATLAB Function block that is open and has focus. You can open the **Ports and Data Manager** directly from the Embedded MATLAB Editor for the block of interest.

The Ports and Data Manager provides the same data definition capabilities as the Model Explorer, but restricted to individual Embedded MATLAB Function blocks. To modify objects and properties for blocks across the Simulink model hierarchy, use the Model Explorer, as described in “The Model Explorer” on page 9-2.

The **Ports and Data Manager** lets you add the following objects to an Embedded MATLAB Function block:

Element	Tool	Description
Data argument		<p>Data arguments are used to communicate between the Embedded MATLAB function and the Simulink environment.</p> <p>To learn how to add data arguments and modify their properties, see “Adding Data to an Embedded MATLAB Function Block” on page 16-49.</p>
Input trigger		<p>An <i>input trigger</i> causes an Embedded MATLAB Function block to execute when a Simulink control signal changes, through a Simulink block that outputs function-call events, or through an S-function.</p> <p>To learn how to add input triggers and modify their properties, see “Adding Input Triggers to an Embedded MATLAB Function Block” on page 16-59.</p>
Function call output		<p>A <i>function call output</i> creates an output port on an Embedded MATLAB Function block, and a function that can be called from the block’s Embedded MATLAB script. When the function is called, it triggers the subsystem attached to the output port. For more information, see “Function-Call Subsystems” in the online Simulink reference.</p> <p>To learn how to add function call outputs and modify their properties, see “Adding Function Call Outputs to an Embedded MATLAB Function Block” on page 16-62.</p>

The Ports and Data Manager Dialog

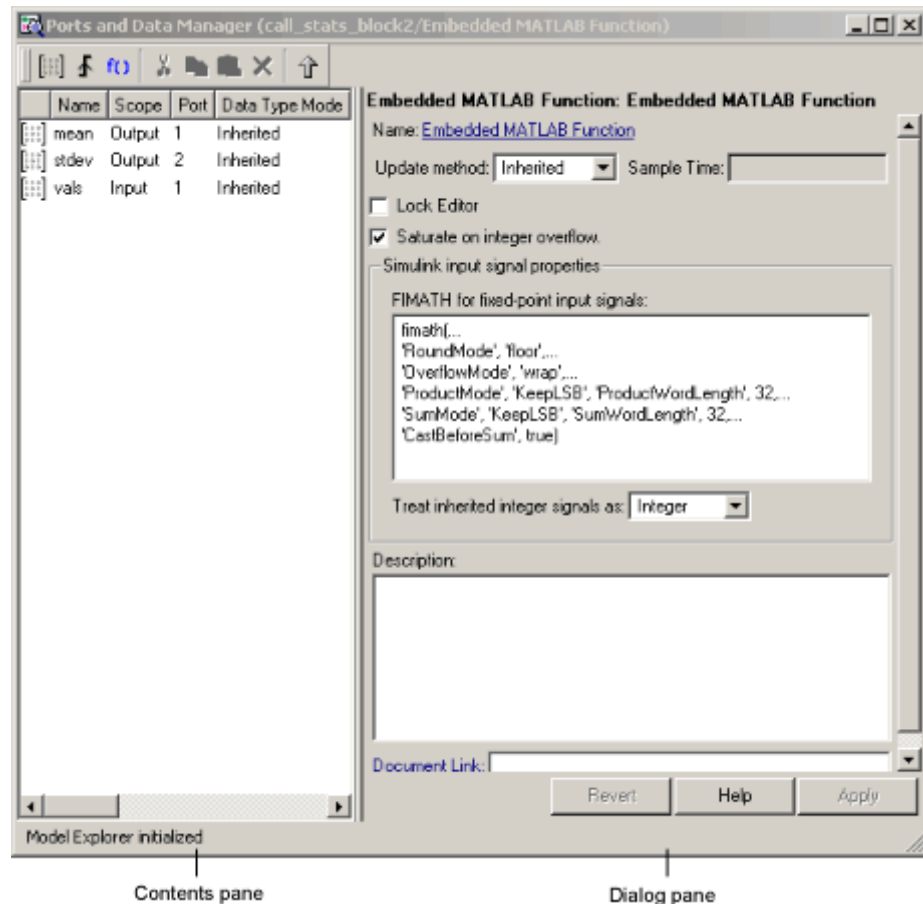
The Ports and Data Manager dialog allows you to define data arguments, input triggers, and function call outputs for Embedded MATLAB Function blocks. Using this dialog, you can also modify properties for the Embedded MATLAB Function and the objects it contains.

The dialog consists of two panes:

- Contents pane lists the objects that have been defined for the Embedded MATLAB Function block
- Dialog pane displays fields for modifying the properties of the selected object

Properties vary according to the scope and type of the object. Therefore, the **Ports and Data Manager** properties dialogs are dynamic, displaying only the property fields that are relevant for the object you add or modify.

When you first open the dialog, it displays the properties of the Embedded MATLAB Function block, as follows:



Opening the Ports and Data Manager

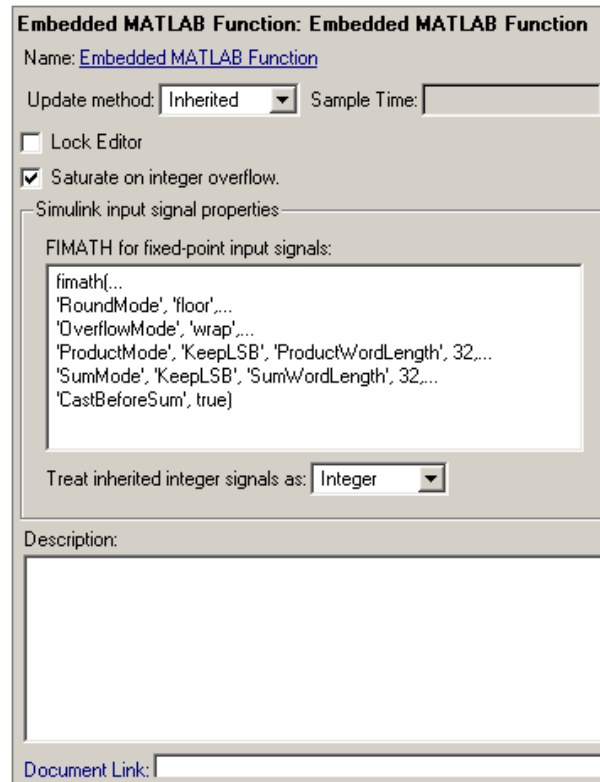
To open the **Ports and Data Manager** from the Embedded MATLAB Editor, select **Tools > Edit Data/Ports** or click the **Edit Data/Ports** icon:



The **Ports and Data Manager** appears for the Embedded MATLAB Function block that is open and has focus.

Setting Embedded MATLAB Function Properties

The Dialog pane for an Embedded MATLAB Function block looks like this:



This section describes each property of an Embedded MATLAB Function block.

Name. Name of the Embedded MATLAB Function block, following the same naming conventions as for Simulink blocks (see “Manipulating Block Names” on page 4-25.

Update method. Method for activating the Embedded MATLAB Function block. You can choose from the following update methods:

Update method	Description
Inherited (default)	<p>Input from the Simulink model activates the Embedded MATLAB Function block.</p> <p>If you define an input trigger, the Embedded MATLAB Function block executes in response to a Simulink signal or function-call event on the trigger port. If you do not define an input trigger, the Embedded MATLAB Function block implicitly inherits triggers from the Simulink model. These implicit events are the sample times (discrete or continuous) of the Simulink signals that provide inputs to the chart.</p> <p>If you define data inputs, the Embedded MATLAB Function block awakens at the rate of the fastest data input. If you do not define data inputs, the Embedded MATLAB Function block awakens as defined by its parent subsystem's execution behavior.</p>
Discrete	<p>Simulink awakens (samples) the Embedded MATLAB Function block at the rate you specify as the block's Sample Time property. An implicit event is generated by Simulink at regular time intervals corresponding to the specified rate. The sample time is in the same units as the Simulink simulation time. Note that other blocks in the Simulink model can have different sample times.</p>
Continuous	<p>Simulink wakes up (samples) the Embedded MATLAB Function block at each step in the simulation, as well as at intermediate time points that can be requested by the Simulink solver. This method is consistent with the continuous method in Simulink.</p>

Lock Editor. Option for locking the Embedded MATLAB Editor. When enabled, this option prevents users from making changes to the Embedded MATLAB Function block.

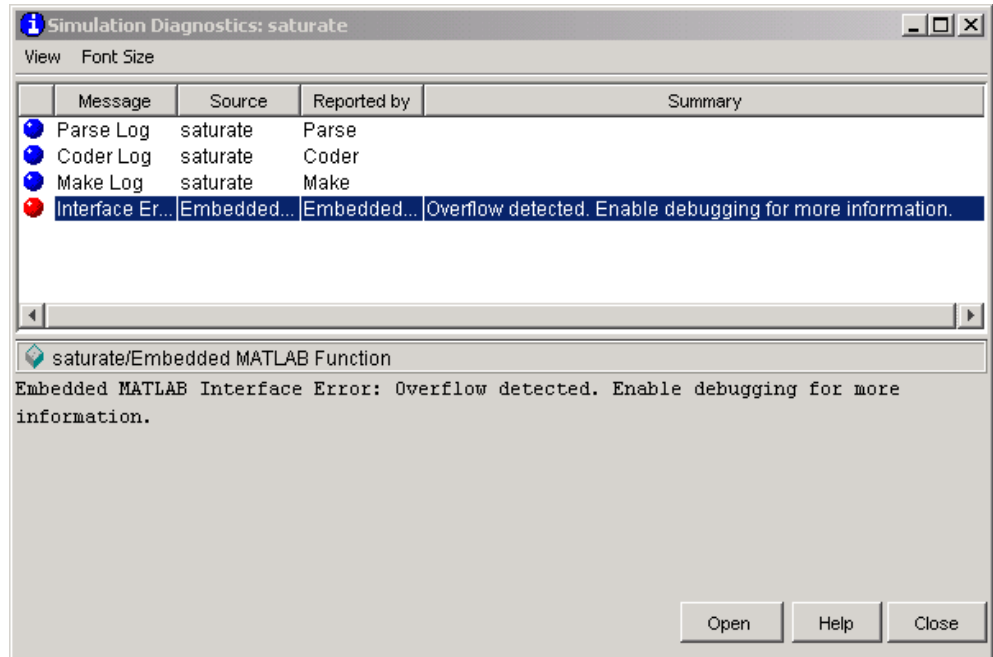
Saturate on integer overflow. Option that determines how the Embedded MATLAB Function block handles overflow conditions during integer operations, as follows:

Setting	Action When Overflow Occurs
Enabled (default)	Saturates integer by setting it to the maximum positive or negative value allowed by the word size. Matches MATLAB behavior.
Disabled	In simulation mode, generates a runtime error. For RTW code generation, the behavior depends on your C-language compiler.

When you enable **Saturate on Integer Overflow**, the Embedded MATLAB Function block adds additional checks in the generated code to detect integer overflow or underflow. Therefore, it is more efficient to disable this option if you are sure that integer overflow and underflow will not occur in your Embedded MATLAB function code.

Even when you disable this option, the code for a simulation target checks for integer overflow and underflow. If either condition occurs, simulation stops and an error is generated. If you enabled debugging for the Embedded MATLAB Function block, the debugger displays the error and lets you examine the data.

If you have not enabled debugging for the Embedded MATLAB Function block, the block generates a runtime error, as in this example:



It is important to note that the code for an RTW target does *not* check for integer overflow or underflow and, therefore, may produce unpredictable results when **Saturate on Integer Overflow** is disabled. In this situation, it is recommended that you simulate first to test for overflow and underflow before generating the RTW target.

The **Saturate on Integer Overflow** option is relevant only for integer arithmetic. It has no effect on fixed point or double-precision arithmetic.

Simulink input signal properties. Parameters that apply to Embedded MATLAB Function blocks in models that use fixed-point or integer data types. You can specify the following properties for Simulink input signals:

Property	Description
FIMATH for fixed-point input signals	<p>Defines the <code>fimath</code> object to be associated with Simulink fixed-point or integer signals that enter the Embedded MATLAB Function block as inputs. Enter an expression that evaluates to a <code>fimath</code> object, as in these examples:</p> <ul style="list-style-type: none"> Fully define the <code>fimath</code> object using the Fixed-Point Toolbox <code>fimath</code> function. Enter the variable name of a <code>fimath</code> object that is defined in the MATLAB workspace. <p>The default <code>fimath</code> object defined for this parameter emulates C-style math for a standard 32bit processor:</p> <pre>fimath(... 'RoundMode', 'floor',... 'OverflowMode', 'wrap',... 'ProductMode', 'KeepLSB', 'ProductWordLength', 32,... 'SumMode', 'KeepLSB', 'SumWordLength', 32,... 'CastBeforeSum', false)</pre> <p>This property applies to all input signals in the Embedded MATLAB Function block, not to each input individually, because signals with different <code>fimath</code> properties cannot interact. You can change the <code>fimath</code> properties of an input by casting it to a variable with the desired <code>fimath</code> properties, as follows:</p> <pre>x = fi(u, F);</pre> <p>In this example, <code>u</code> is the input, <code>x</code> is the variable, and <code>F</code> is the desired <code>fimath</code> object.</p> <p>For more information, see “Working with <code>fimath</code> Objects”.</p>
Treat inherited integer signals as	<p>Specifies whether to treat inherited integer signals as MATLAB integers or Fixed-Point Toolbox <code>fi</code> objects.</p>

Description. Description of the Embedded MATLAB Function block.

Document Link. Link to online documentation for the Embedded MATLAB Function block. To document an Embedded MATLAB Function block, set the **Document Link** property to a Web URL address or MATLAB expression that displays documentation in a suitable online format (for example, an HTML file or text in the MATLAB command window). The Embedded MATLAB Function block evaluates the expression when you click the blue **Document Link** text.

Adding Data to an Embedded MATLAB Function Block

You can define input and output data arguments for an Embedded MATLAB Function block directly in the script, or by using the Ports and Data Manager or Model Explorer. You can use the Ports and Data Manager to add data arguments to an Embedded MATLAB Function block that is open and has focus. You can also modify the properties of data arguments in the block.

You can define the following data arguments for Embedded MATLAB Function blocks:

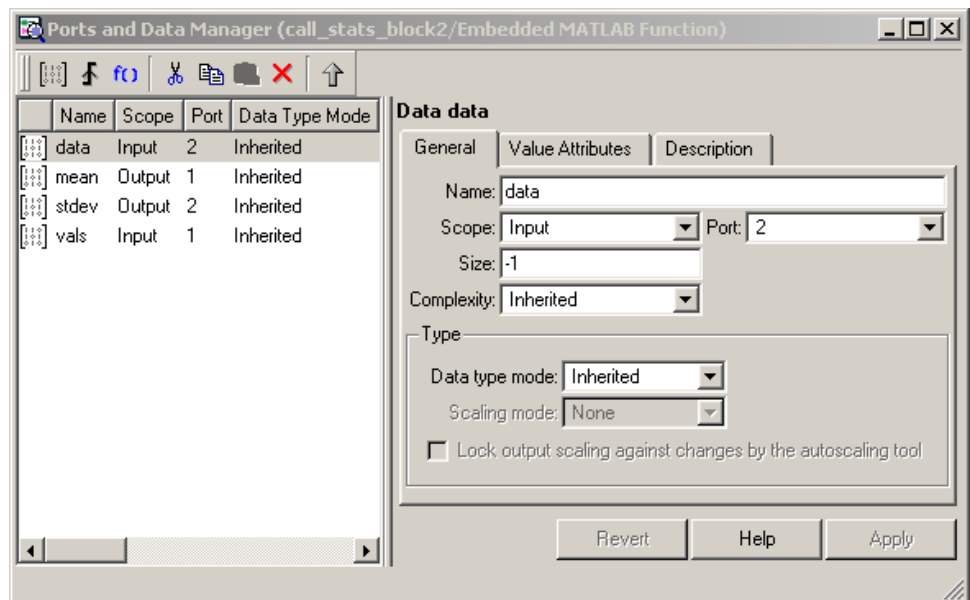
Method	For Defining	Reference
Define data directly in the Embedded MATLAB Function script	Input and output data	See “Defining Inputs and Outputs” on page 16-16.
Use the Ports and Data Manager	Input, output, and parameter data in the Embedded MATLAB Function that is open and has focus	See “Defining Data in the Ports and Data Manager” on page 16-50.
Use the Model Explorer	Input, output, and parameter data in Embedded MATLAB Function blocks at all levels of the Simulink model hierarchy	See “The Model Explorer” on page 9-2.

Defining Data in the Ports and Data Manager. To add a data argument and modify its properties, follow these steps:

- 1 In the Ports and Data Manager, click the **Add Data** icon:

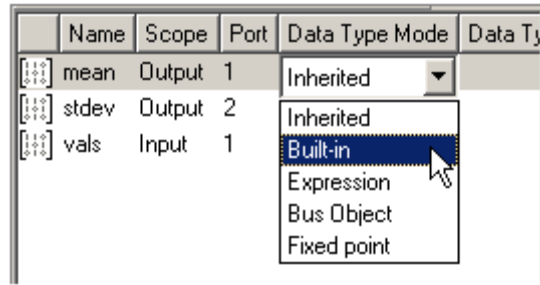


The Ports and Data Manager adds a default definition of the data argument to the Embedded MATLAB Function block and displays the Data properties dialog in its Dialog pane, as in this example.



- 2 Modify properties for the new data argument, using one of the following methods:

- In the Contents pane, select the row that contains the data argument you want to modify and then click the value of the property of interest, as in this example:



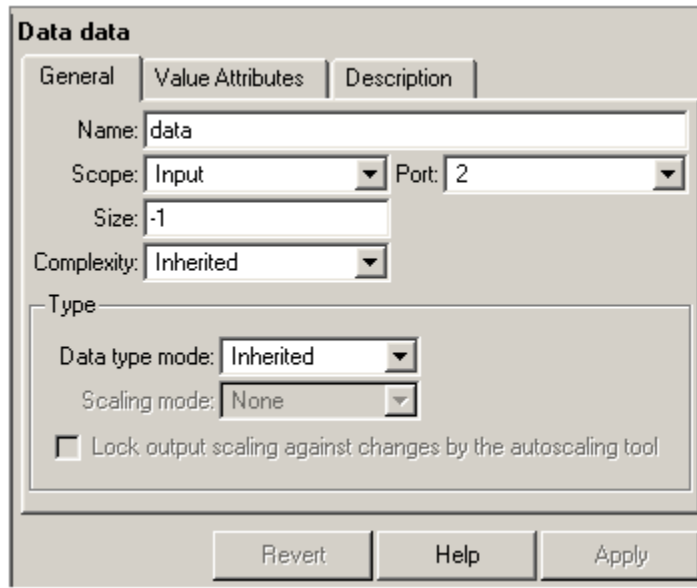
- Modify fields in the Data properties dialog, as described in “The Data Properties Dialog” on page 16-51.

The Data Properties Dialog. The Data properties dialog in the Ports and Data Manager allows you to set and modify the properties of data arguments in Embedded MATLAB Function blocks. Properties vary according to the scope and type of the data object. Therefore, the Data properties dialog is dynamic, displaying only the property fields that are relevant for the data argument you are defining.

You can open the Data properties dialog using one of these methods:

- Select a data argument in the Contents pane of the Ports and Data Manager to open the Data properties dialog in the Dialog pane.

The Data properties dialog provides a set of tabbed panes, as in this example:



Each pane lets you define different features of your data argument:

- The **General** pane lets you define the scope, size, complexity, and type of the data argument. See “Setting General Properties” on page 16-52.
- The **Value Attributes** pane lets you set a limit range, and save data argument values to the model’s base workspace. See “Setting Value Attributes Properties” on page 16-56.
- The **Description** pane lets you enter a description and link to documentation about the data argument. See “Setting Description Properties” on page 16-58.

Setting General Properties. The General tab of the Data properties dialog looks like this:

Data data

General Value Attributes Description

Name: data

Scope: Input Port: 2

Size: -1

Complexity: Inherited

Type

Data type mode: Inherited

Scaling mode: None

☐ Lock output scaling against changes by the autoscaling tool

Revert Help Apply

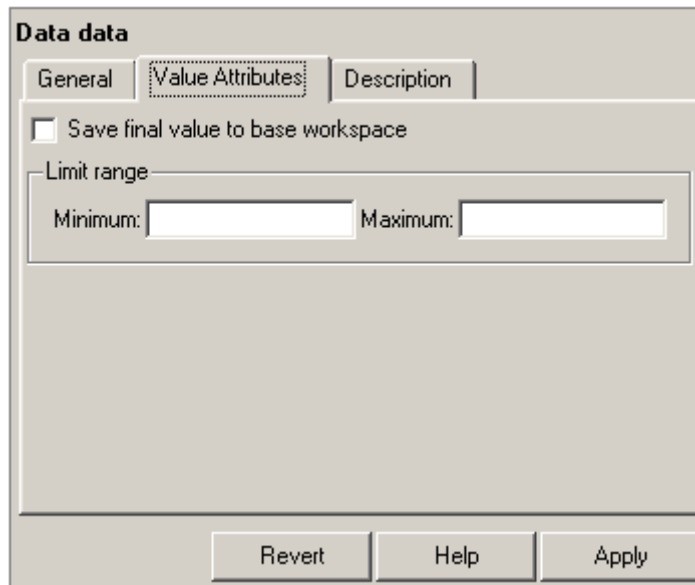
You can set the following properties in the General tab:

Property	Description
Name	Name of the data argument, following the same naming conventions used in MATLAB (see “Naming Variables” in the online MATLAB documentation.

Property	Description
Scope	<p>Where data resides in memory, relative to its parent. Scope determines the range of functionality of the data argument. You can set scope to one of the following values:</p> <ul style="list-style-type: none"> • Parameter: Specifies that the source for this data is a variable of the same name in the MATLAB or model workspace or in the workspace of a masked subsystem containing this block. If a variable of the same name exists in more than one of the workspaces visible to the block, Simulink uses the variable closest to the block in the in the workspace hierarchy (see “Working with Model Workspaces” on page 3-103). • Input: Data provided by the Simulink model via an input port to the Embedded MATLAB Function block. • Output: Data provided by the Embedded MATLAB Function block via an output port to the Simulink model. <p>For more information, see “Defining Inputs and Outputs” on page 16-16 and “Parameter Arguments in Embedded MATLAB Functions” on page 16-85.</p>
Port	Index of the port associated with the data argument. This property applies only to input and output data.
Tunable	Indicates whether the parameter used as the source of this data item is tunable (see “Tunable Parameters” on page 1-8). You must uncheck this option if you want to use the parameter where Embedded MATLAB requires a constant expression, such as zeros (see entry for zeros in “Embedded MATLAB Run-Time Function Library — Alphabetical List”). This property applies only to data with scope equal to Parameter .
Size	<p>Size of the data argument. Size can be a scalar value or a MATLAB vector of values, as described in “Specifying Argument Sizes with Expressions” on page 16-83. Size defaults to –1, which means that it is inherited, as described in “Inheriting Argument Sizes from Simulink” on page 16-81.</p> <p>For more details, see “Sizing Function Arguments” on page 16-81.</p>

Property	Description
Complexity	<p>Indicates whether the value of the data argument is a real or complex number. You can set complexity to one of the following values:</p> <ul style="list-style-type: none"> • Off: Data argument is a real number • On: Data argument is a complex number • Inherited: Data argument inherits complexity based on its scope. Input and output data inherit complexity from the Simulink signals connected to them; parameter data inherits complexity from the parameter to which it is bound.
Data type mode and Data type	<p>Data type mode lets you choose a method for specifying the type of a data argument. You can set data type mode to one of the following values:</p> <ul style="list-style-type: none"> • Inherited: Data object inherits its type based on its scope. Input and output data inherit type from the Simulink signals connected to them; parameter data inherits type from the parameter to which it is bound. See “Inheriting Argument Data Types” on page 16-72. • Built-in: Select from a list of supported data types in the Data type field, as described in “Built-In Data Types for Arguments” on page 16-74. • Expression: In the Data type field, enter an expression that evaluates to a data type. See “Specifying Argument Types with Expressions” on page 16-74. • Bus Object: Data object is a structure (see “Specifying Structures and Working with Bus Signals” on page 16-75). In the Data type field, enter the name of a Simulink.Bus object that you have created in the base workspace to define the properties of the structure. Click the Edit button to create or modify Simulink.Bus objects using the Simulink Bus Types Editor (see “Bus Editor” on page 5-80). • Fixed-point: Specify word length, scaling mode, whether the data is signed or unsigned, and whether to lock output scaling, as described in “Specifying Fixed-Point Data Properties” on page 16-79.

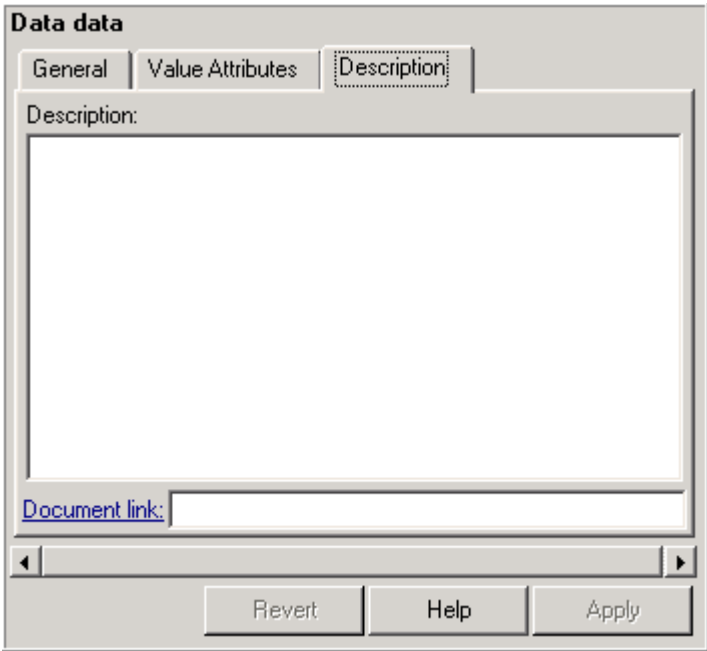
Setting Value Attributes Properties. The **Value Attributes** tab of the **Data** properties dialog looks like this:



You can set the following properties on the Value Attributes tab:

Property	Description
Save final value to base workspace	If you select this option, the Embedded MATLAB Function block assigns the value of the data argument to a variable of the same name in the model's base workspace at the end of simulation (see "Working with Model Workspaces" on page 3-103).
Limit range properties	<p>The range of acceptable values for this data object. The Embedded MATLAB Function block uses this range to validate the data object during simulation. To establish the range, specify these properties:</p> <ul style="list-style-type: none"> • Maximum — The largest value allowed for the data item during simulation. You can enter an expression or parameter that evaluates to a numeric scalar value. • Minimum — The smallest value allowed for the data item during simulation. You can enter an expression or parameter that evaluates to a numeric scalar value. <p>If you do not specify a value, the default for Maximum is <code>inf</code> and the default for Minimum is <code>-inf</code>.</p>

Setting Description Properties. The Description tab of the Data properties dialog looks like this:



You can set the following properties on the Description tab:

Property	Description
Description	Description of the data argument.
Document link	Link to online documentation for the data argument. You can enter a Web URL address or a MATLAB command that displays documentation in a suitable online format, such as an HTML file or text in the MATLAB command window. When you click the blue text that reads Document link displayed at the bottom of the Data properties dialog, the Embedded MATLAB Function block evaluates the link and displays the documentation.

Adding Input Triggers to an Embedded MATLAB Function Block

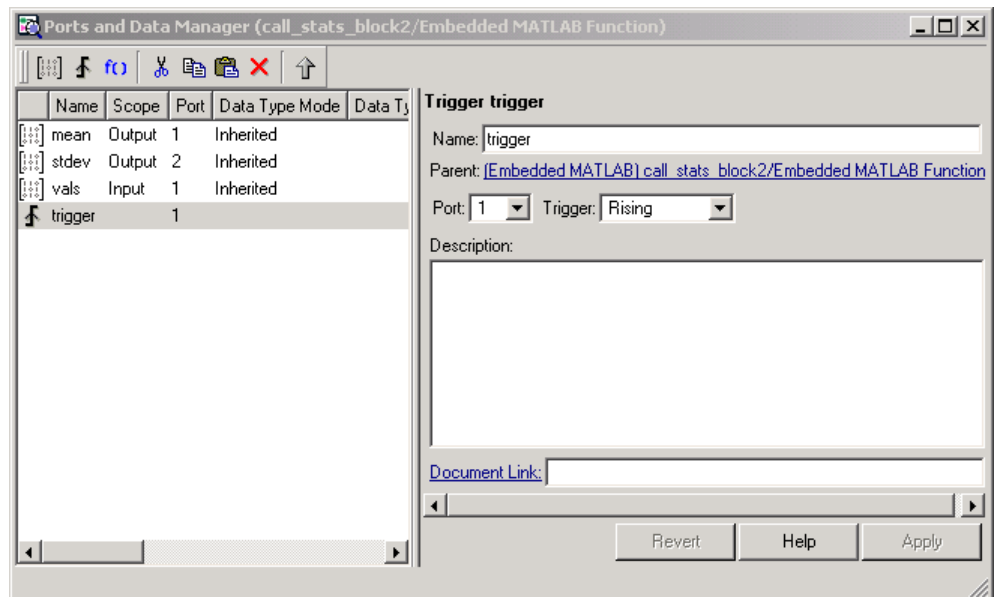
You can use the Ports and Data Manager to add input triggers to an Embedded MATLAB Function block that is open and has focus. You can also modify the properties of input triggers in the block.

To add an input trigger and modify its properties, follow these steps:

- 1 In the Ports and Data Manager, click the **Add Input Trigger** icon:

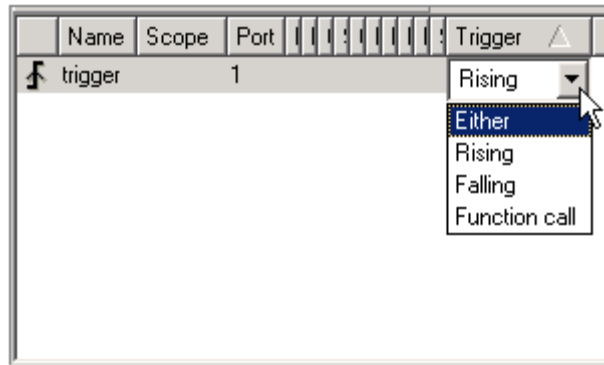


The Ports and Data Manager adds a default definition of the new input trigger to the Embedded MATLAB Function block and displays the Trigger properties dialog in its Dialog pane, as in this example.



- 2 Modify properties for the new input trigger, using one of the following methods:

- In the Contents pane, select the row that contains the input trigger you want to modify and then click the value of the property of interest, as in this example:



- Modify fields in the Trigger properties dialog, as described in “The Trigger Properties Dialog” on page 16-60.

The Trigger Properties Dialog. The Trigger properties dialog in the Ports and Data Manager allows you to set and modify the properties of input triggers in Embedded MATLAB Function blocks.

You can open the Trigger properties dialog using one of these methods:

- Select an input trigger in the Contents pane of the Ports and Data Manager to open the Trigger properties dialog in the Dialog pane.
- Right-click an input trigger in the Contents pane and select **Properties** from the submenu to open the Trigger properties dialog outside the Ports and Data Manager.

The Trigger properties dialog looks like this:

Setting Input Trigger Properties. You can set the following properties in the Trigger properties dialog:

Property	Description
Name	Name of the input trigger, following the same naming conventions used in MATLAB (see “Naming Variables” in the online MATLAB documentation).
Port	Index of the port associated with the input trigger.

Property	Description
Trigger	<p>Type of event that triggers execution of the Embedded MATLAB Function block. You can select one of the following types of triggers:</p> <ul style="list-style-type: none"> • Rising: Triggers execution of the Embedded MATLAB Function block when the control signal rises from a negative or zero value to a positive value (or zero if the initial value is negative). • Falling: Triggers execution of the Embedded MATLAB Function block when the control signal falls from a positive or zero value to a negative value (or zero if the initial value is positive). • Either: Triggers execution of the Embedded MATLAB Function block when the control signal is either rising or falling. • Function call: Triggers execution of the Embedded MATLAB Function block from a Simulink block that outputs function-call events, or from an S-function
Description	Description of the input trigger.
Document link	<p>Link to online documentation for the input trigger. You can enter a Web URL address or a MATLAB command that displays documentation in a suitable online format, such as an HTML file or text in the MATLAB command window. When you click the blue text that reads Document link displayed at the bottom of the Trigger properties dialog, the Embedded MATLAB Function block evaluates the link and displays the documentation.</p>

Adding Function Call Outputs to an Embedded MATLAB Function Block

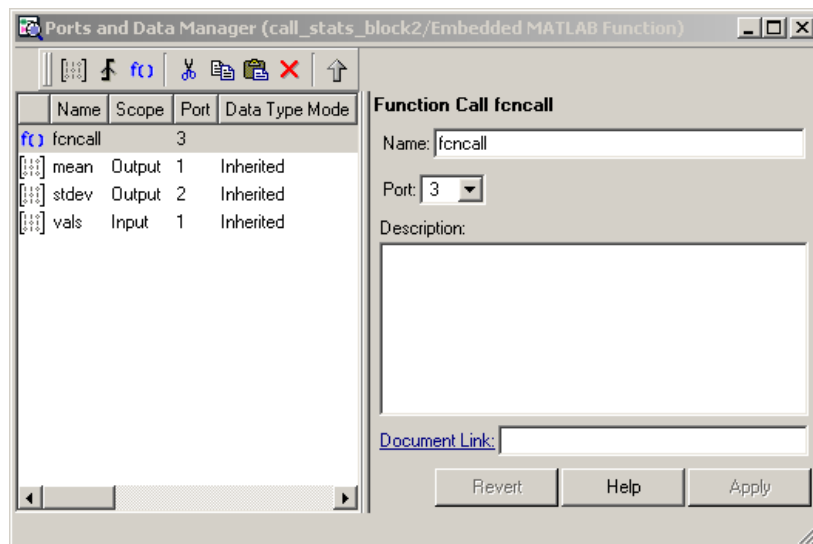
You can use the Ports and Data Manager to add function call outputs to an Embedded MATLAB Function block that is open and has focus. You can also modify the properties of function call outputs in the block.

To add a function call output and modify its properties, follow these steps:

- 1 In the Ports and Data Manager, click the **Add Function Call Output** icon:

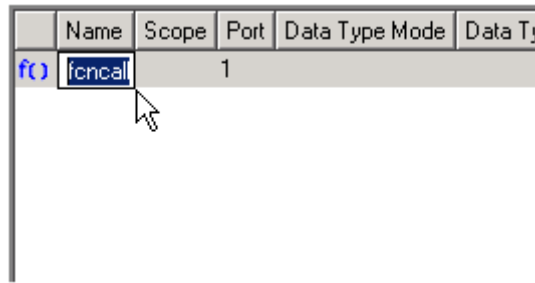


The Ports and Data Manager adds a default definition of the new function call output to the Embedded MATLAB Function block and displays the Function Call properties dialog in its Dialog pane, as in this example:



- 2 Modify properties for the new function call output, using one of the following methods:

- In the Contents pane, select the row that contains the function call output you want to modify and then click the value of the property of interest, as in this example:



	Name	Scope	Port	Data Type Mode	Data Type
f()	fncall		1		

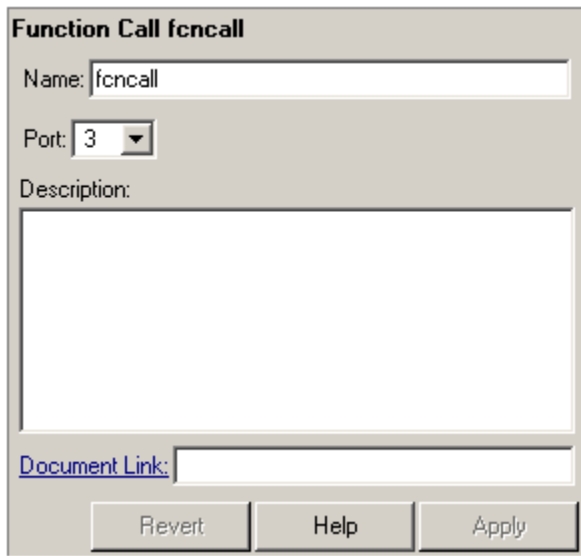
- Modify fields in the Function Call properties dialog, as described in “The Function Call Properties Dialog” on page 16-64.

The Function Call Properties Dialog. The Function Call properties dialog in the Ports and Data Manager allows you to set and modify the properties of function call outputs in Embedded MATLAB Function blocks.

You can open the Function Call properties dialog using one of these methods:

- Select a function call output in the Contents pane of the Ports and Data Manager to open the Function Call properties dialog in the Dialog pane.
- Right-click a function call output in the Contents pane and select **Properties** from the submenu to open the Function Call properties dialog outside the Ports and Data Manager.

The Function Call properties dialog looks like this:



The image shows a dialog box titled "Function Call fncall". It contains the following fields and controls:







- Name:** A text field containing "fncall".
- Port:** A dropdown menu showing "3".
- Description:** A large empty text area.
- Document Link:** A text field with a blue underlined "Document Link:" label.
- Buttons:** "Revert", "Help", and "Apply" buttons at the bottom.





Setting Function Call Output Properties. You can set the following properties in the Function Call properties dialog:

Property	Description
Name	Name of the function call output, following the same naming conventions used in MATLAB (see “Naming Variables” in the online MATLAB documentation).
Port	Index of the port associated with the function call output.
Description	Description of the function call output.
Document link	Link to online documentation for the function call output. You can enter a Web URL address or a MATLAB command that displays documentation in a suitable online format, such as an HTML file or text in the MATLAB command window. When you click the blue text that reads Document link displayed at the bottom of the Function Call properties dialog, the Embedded MATLAB Function block evaluates the link and displays the documentation.

Debugging Embedded MATLAB Functions

Use the following tools during an Embedded MATLAB function debugging session:

Tool Button	Description
	A breakpoint indicator. To set a breakpoint for a line of function code, click the hyphen character (-) in the breakpoints column for the line. A breakpoint indicator appears in place of the hyphen. Click the breakpoint indicator to clear the breakpoint.
 Build	Check for errors and build a simulation application (if no errors are found) for the model containing this Embedded MATLAB function.
 Start Simulation	Start simulation of the model containing the Embedded MATLAB function. Alternatively, press F5 , or, from the Debug menu, select Start .
 Stop Simulation	Stop simulation of the model containing the Embedded MATLAB function. You can also select Exit debug mode from the Debug menu if execution is paused at a breakpoint.
 Set/Clear Breakpoint	Set a new breakpoint or clear an existing breakpoint for the selected Embedded MATLAB code line. The presence of the text cursor or highlighted text selects the line.
 Clear All Breakpoints	Clear all set breakpoints in the Embedded MATLAB function.

Tool Button	Description
 Step	<p>Step through the execution of the next Embedded MATLAB code line. This tool steps past function calls and does not enter called functions for line-by-line execution. You can use this tool only after execution has stopped at a breakpoint. Alternatively, press F11, or, from the Debug menu, select Step.</p>
 Step In	<p>Step through the execution of the next Embedded MATLAB code line. If the line calls a subfunction, step into line-by-line execution of the subfunction. You can use this tool only after execution has stopped at a breakpoint.</p>
 Step Out	<p>Step out of line-by-line execution of the current subfunction to the line after the line that calls this subfunction. You can use this tool only after execution has stopped at a breakpoint.</p>
 Continue Debugging	<p>Continue debugging after a pause, such as stopping at a breakpoint.</p>

See “Debugging the Function in Simulation” on page 16-20 for an example using some of these debugging tools.

Typing Function Arguments

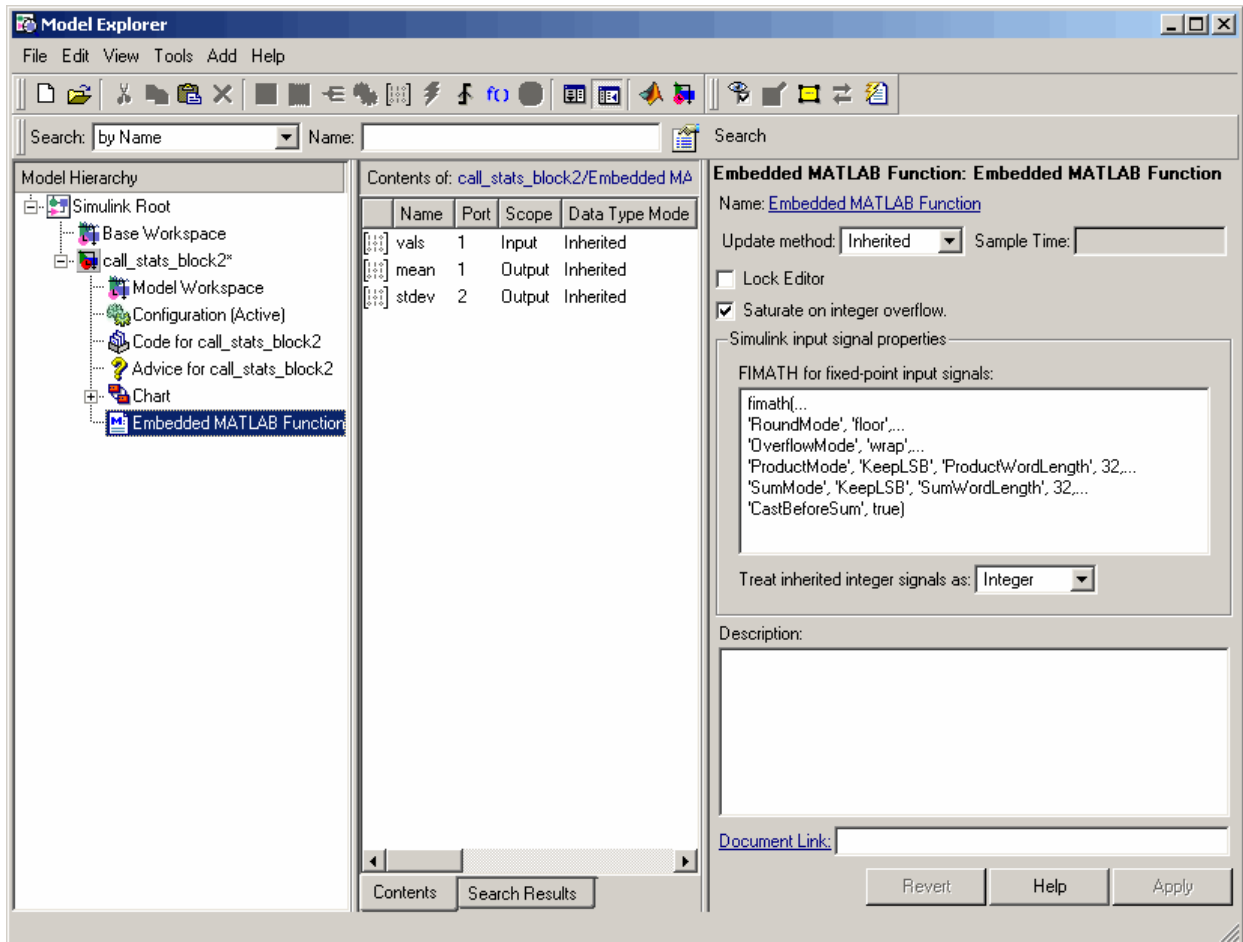
In “Programming the Embedded MATLAB Function” on page 16-8, you create two output arguments and an input argument for an Embedded MATLAB Function block by entering them in its function header. When you define arguments, Simulink creates corresponding ports on the Embedded MATLAB Function block that you can attach to Simulink signals. You can select a data type mode for each argument that you define for an Embedded MATLAB Function block. Each *data type mode* presents its own set of options for selecting a *data type*.

By default, the data type mode for Embedded MATLAB function arguments is **Inherited**. This means that the function argument inherits its data type from the incoming or outgoing Simulink signal. To override the default type, you first choose a data type mode and then select a data type based on the mode. The following procedure describes how to use the **Model Explorer** to set data types for function arguments. You can also use the **Ports and Data Manager** tool (see “Ports and Data Manager” on page 16-40).

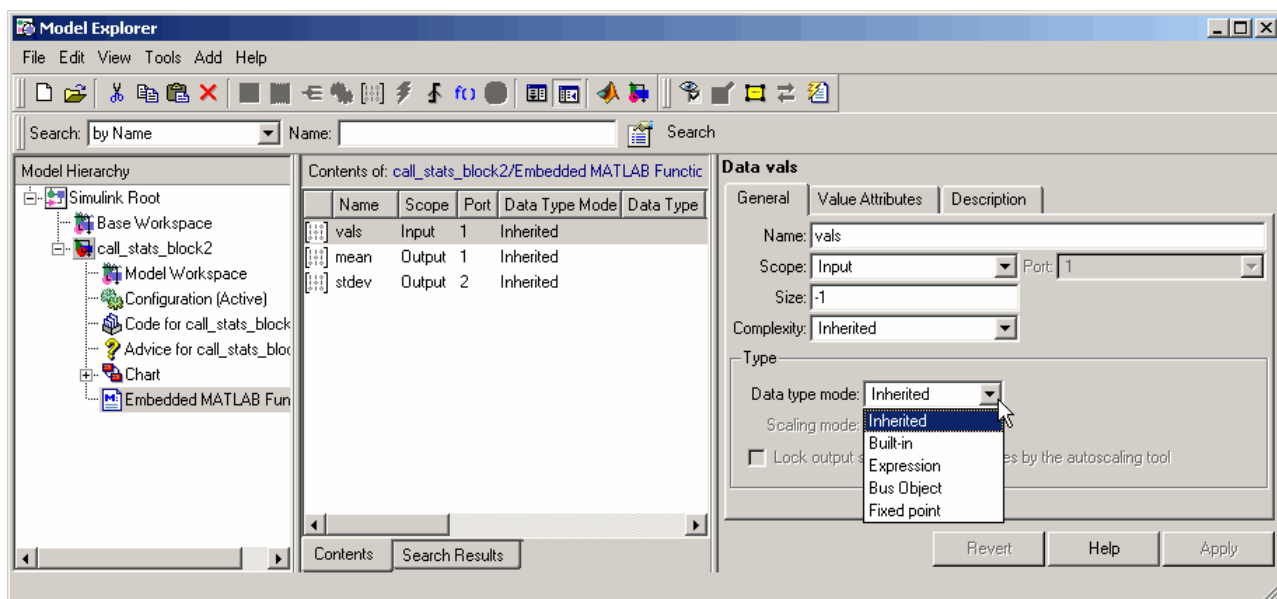
To specify the type of an Embedded MATLAB function argument:

- 1 From the **Embedded MATLAB Editor**, select **Model Explorer** from the **Tools** menu.

The **Model Explorer** appears with the Embedded MATLAB Function block highlighted in the **Model Hierarchy** pane on the left.



- 2 In the **Contents** pane, click the row containing the argument of interest and choose an option from the drop-down menu in the **Data Type Mode** column, or from the **Data type mode** field in the **Data** properties dialog on the right, as shown:



The **Data** properties dialog changes dynamically to display additional fields for specifying the data type associated with the mode.

- 3 Based on the mode you select, specify a data type as follows:

Mode	What To Specify
Inherited	<p>You cannot specify a value. The data type is inherited from previously-defined data, based on the scope you selected for the Embedded MATLAB function argument, as follows:</p> <ul style="list-style-type: none"> • If scope is Input, data type is inherited from the Simulink input signal on the designated port. • If scope is Output, data type is inherited from the Simulink output signal on the designated port. • If scope is Parameter, data type is inherited from the associated parameter, which can be defined in the Simulink masked subsystem or the MATLAB workspace. <p>See “Inheriting Argument Data Types” on page 16-72.</p>
Built-in	<p>In the Data type field of the Data properties dialog or Contents pane, select from the drop-down list of supported data types, as described in “Built-In Data Types for Arguments” on page 16-74.</p>
Expression	<p>In the Data type field of the Data properties dialog or Contents pane, enter an expression that evaluates to a data type . See “Specifying Argument Types with Expressions” on page 16-74.</p>

Mode	What To Specify
Bus Object	<p>In the Data type field of the Data properties dialog or Contents pane, enter the name of a Simulink.Bus object that you have created in the base workspace to define the properties of an Embedded MATLAB structure. See “Specifying Structures and Working with Bus Signals” on page 16-75.</p> <hr/> <p>Note You can click the Edit button to create or modify Simulink.Bus objects using the Simulink Bus Types Editor (see “Bus Editor” on page 5-80 in the Simulink User’s Guide).</p> <hr/>
Fixed point	<p>Specify the following information about the fixed point data in the Data properties dialog:</p> <ul style="list-style-type: none"> • Whether the data is signed or unsigned • Word length • Scaling mode <p>For information on how to specify these fixed point data properties, see “Specifying Fixed-Point Data Properties” on page 16-79.</p>

Inheriting Argument Data Types

Embedded MATLAB Function arguments can inherit their data types, including fixed point types, from the Simulink signals to which they are connected. Select the argument of interest in the Contents pane of the Model Explorer or Ports and Data Manager, and set data type mode using one of these methods:

- In the Dialog pane, set the **Data type mode** field in the **Data** properties dialog to **Inherited**.
- In the Contents pane, set the **Data Type Mode** column to **Inherited**.

See “Built-In Data Types for Arguments” on page 16-74 for a list of supported data types.




Note An argument can also inherit its complexity (whether its value is a real or complex number) from the Simulink signal that is connected to it. To inherit complexity, set the **Complexity** field on the **Data** properties dialog to **Inherited**.

Once you build the model, the **Compiled Type** column of the Model Explorer or Ports and Data Manager gives the actual type used in the compiled simulation application. To conveniently compile and build the model from the **Embedded MATLAB Editor**, click the **Build** icon:



In the following example, an Embedded MATLAB Function block argument inherits its data type from an input signal of type double:

Contents of: [call_stats_block2/Embedded MATLAB Function](#)

	Name	Port	Scope	Data Type Mode	Data Type	Compiled Type
	vals	1	Input	Inherited		double
	mean	1	Output	Inherited		double
	stdev	2	Output	Inherited		double

Actual compiled types

The inherited type of output data is inferred from diagram actions that store values in the specified output. In the preceding example, the variables mean and stdev are computed from operations with double operands, which yield results of type double. If the expected type in Simulink matches the inferred type, inheritance is successful. In all other cases, a mismatch occurs during build time.

Note Library Embedded MATLAB function blocks can have inherited data types, sizes, and complexities like ordinary Embedded MATLAB function blocks. However, all instances of the library block in a given model must have inputs with the same properties.

Built-In Data Types for Arguments

When you select **Built-in** for **Data type mode**, the **Data** properties dialog displays a **Data type** field that provides a drop-down list of supported data types. You can also choose a data type from the **Data Type** column in the **Contents** pane of the Model Explorer or Ports and Data Manager. Here is a list of the supported data types:

Data Type	Description
double	64-bit double-precision floating point
single	32-bit single-precision floating point
int32	32-bit signed integer
int16	16-bit signed integer
int8	8-bit signed integer
uint32	32-bit unsigned integer
uint16	16-bit unsigned integer
uint8	8-bit unsigned integer
boolean	Boolean (1 = true; 0 = false)

Specifying Argument Types with Expressions

You can specify the types of Embedded MATLAB Function arguments as expressions in the Model Explorer or Ports and Data Manager. Follow these steps:

- 1 Select **Expression** as the data type mode for your function argument, either in the Data properties dialog in the Dialog pane or Data Type Mode column in the Contents pane.

A Data type field appears in the Data properties dialog. The Contents pane also provides a Data Type column.

- 2** In the Data type field, enter an expression that evaluates to a data type. The following expressions are allowed:

- Alias type from the MATLAB workspace, as described in “Creating a Data Type Alias”.
- `fixdt` function to create a `Simulink.NumericType` object describing a fixed-point or floating-point data type
- type operator, to base the type on previously defined data

In the following example, the data type of input argument `data1` is **`int32`**. The data type of input argument `data2` is based on `data1` using the expression `type(data1)`. Click the Build icon to compile and build the model from the **Embedded MATLAB Editor**.



When the model is compiled, the actual type of `data2` appears in the Compiled Type column in the Contents pane:

Compiled data types

The screenshot shows the Embedded MATLAB Editor interface. The Contents pane on the left lists variables: `vals`, `mean`, `stddev`, `data1`, and `data2`. The `data2` row is highlighted. The Data properties dialog for `data2` is open on the right, showing the 'General' tab. The 'Data type mode' is set to 'Expression' and the 'Data type' is `type(data1)`. The 'Compiled Type' column in the Contents pane shows `int32` for `data2`.

Name	Port	Scope	Data Type Mode	Data Type	Compiled Type
<code>vals</code>	1	Input	Inherited		unknown
<code>mean</code>	1	Output	Inherited		unknown
<code>stddev</code>	2	Output	Inherited		unknown
<code>data1</code>	2	Input	Built-in	<code>int32</code>	<code>int32</code>
<code>data2</code>	3	Input	Expression	<code>type(data1)</code>	<code>int32</code>

Specifying Structures and Working with Bus Signals

This section explains how to define structures in Embedded MATLAB Function blocks that you can interface to Simulink bus signals, or define as local or persistent variables.

- “About Structures in Embedded MATLAB Function Blocks” on page 16-76
- “Rules for Defining Structures in Embedded MATLAB Function Blocks” on page 16-76
- “Defining Structure Inputs and Outputs to Interface with Bus Signals” on page 16-77
- “Defining Local and Persistent Structure Variables” on page 16-79

About Structures in Embedded MATLAB Function Blocks

Embedded MATLAB supports MATLAB structures. You can create structures in the top-level function of Embedded MATLAB Function blocks to interface with Simulink bus signals at input and output ports. Simulink buses appear inside the Embedded MATLAB Function block as structures; structure outputs from the Embedded MATLAB Function block appear as buses in Simulink.

You can also create structures as local and persistent variables in top-level functions and sub-functions of Embedded MATLAB Function blocks.

Rules for Defining Structures in Embedded MATLAB Function Blocks

Follow these rules when defining structures in Embedded MATLAB Function blocks:

- For each structure input or output in an Embedded MATLAB Function block, you must define a `Simulink.Bus` object in the base workspace to specify its type to Simulink.
- Embedded MATLAB structures cannot inherit their type from Simulink.
- Embedded MATLAB Function blocks support nonvirtual buses only (see “Working with Virtual and Nonvirtual Buses” on page 16-78).
- Structures cannot have scopes defined as **Parameter**.

Defining Structure Inputs and Outputs to Interface with Bus Signals

When you create structure inputs in Embedded MATLAB function blocks, Embedded MATLAB determines the type, size, and complexity of the structure from the Simulink input signal. When you create structure outputs, you must define their type, size, and complexity in the Embedded MATLAB function.

You can connect Embedded MATLAB structure inputs and outputs to any Simulink bus signal, including:

- Simulink blocks that output bus signals— such as Bus Creator blocks
- Simulink blocks that accept bus signals as input — such as Bus Selector and Gain blocks
- S-Function blocks
- Other Embedded MATLAB Function blocks

To define structure inputs and outputs in Embedded MATLAB Function blocks, follow these steps:

- 1** Create a Simulink bus object in the base workspace to specify the properties of the structure you will create in the Embedded MATLAB Function block.

For information about how to create Simulink bus objects, see `Simulink.Bus` in the Simulink Reference.

- 2** Open the Ports and Data Manager, as described in “Opening the Ports and Data Manager” on page 16-43.

- 3** In the Ports and Data Manager, follow these steps:

- a** Add a data object, as described in “Defining Data in the Ports and Data Manager” on page 16-50.

The Ports and Data Manager adds a data object and opens a Properties dialog box in its right-hand Dialog pane.

- b** In the Properties dialog box, enter the following information in the General tab fields:

Field	What to Specify
Name	Enter a name for referencing the structure in the Embedded MATLAB Function block. This name does not have to match the name of the bus object in the base workspace.
Scope	Select Input or Output .
Port	Enter the index of the port to be associated with the data object. By default, the Ports and Data Manager automatically assigns the next available input or output port.
Data type mode	Select Bus Object .
Bus object	Enter the name of the Simulink.Bus object in the base workspace that defines the structure.

- c To add or modify Simulink.Bus objects, click the **Edit** button to bring up the Simulink Bus Types Editor (see “Bus Editor” on page 5-80 in the Simulink User’s Guide).

- d Click **Apply**.

- 4 If your structure is an output (has scope of **Output**), define and initialize the output implicitly as a structure variable in the Embedded MATLAB function to have the same type, size, and complexity as its Simulink.Bus object, as described in “Defining Structure Variables Implicitly in Embedded MATLAB Functions”.

Working with Virtual and Nonvirtual Buses. Embedded MATLAB supports nonvirtual buses only (see “Virtual Versus Nonvirtual Buses” on page 5-11 in the Simulink User’s Guide). When Simulink builds models that contain Embedded MATLAB function inputs and outputs, it uses a hidden converter block to convert bus signals for use with Embedded MATLAB, as follows:

- Converts incoming virtual bus signals to nonvirtual buses for Embedded MATLAB structure inputs

- Converts outgoing nonvirtual bus signals from Embedded MATLAB to virtual bus signals, if necessary

Defining Local and Persistent Structure Variables

You can define structures as local or persistent variables inside Embedded MATLAB functions (see “Defining Structure Variables Implicitly in Embedded MATLAB Functions”).

Specifying Fixed-Point Data Properties

Embedded MATLAB Function blocks can represent signals and parameter values as fixed-point numbers. To simulate models that use fixed-point data in Embedded MATLAB Function blocks, you must install the Simulink Fixed Point product on your system (see “What Is Simulink Fixed Point?” in the Simulink Fixed Point online documentation).

When you select the data type mode **Fixed point**, the Type panel in the Data properties dialog displays new fields for specifying additional information about your fixed-point data, as in this example:

The image shows a MATLAB/Simulink dialog box titled "Data vals". It has three tabs: "General", "Value Attributes", and "Description". The "General" tab is selected. Inside the "General" tab, there is a "Type" section. The "Data type mode" is set to "Fixed point". There is a checkbox for "Signed" which is unchecked. The "Word length" field is empty. The "Scaling mode" is set to "None". There is a checkbox for "Lock output scaling against changes by the autoscaling tool" which is unchecked. Other fields in the "General" tab include "Name" (vals), "Scope" (Input), "Port" (1), "Size" (-1), and "Complexity" (Inherited).

You can set the following fixed-point properties:

Signed. Use this check box to indicate whether you want the fixed-point data to be signed or unsigned. Signed data can represent positive and negative quantities. Unsigned data represents positive values only.

Word length. Specify the size in bits of the word that will hold the quantized integer. Large word sizes represent large quantities with greater precision than small word sizes. Word length can be any integer between 0 and 32. If you do not specify a value, the default is 16.

Scaling mode. Specify the method for scaling your fixed point data to avoid overflow conditions and minimize quantization errors. Scaling is disabled by default. However, you can select two scaling modes:

Scaling Mode	Description
Binary point	If you select this mode, the Data properties dialog displays a field for entering fraction length that specifies the binary point location. Binary points can be positive or negative integers. If you do not specify a value, the default is 0. A positive integer entry moves the binary point left of the rightmost bit by that amount. For example, an entry of 2 sets the binary point in front of the second bit from the right. A negative integer entry moves the binary point further right of the rightmost bit by that amount.
Slope and bias	If you select this mode, the Data properties dialog displays fields for entering separate values for the slope and bias. Slope can be any <i>positive</i> real number. If you do not specify a value, the default is 1.0. Bias can be any real number. If you do not specify a value, the default value is 0.0. You can enter slope and bias as expressions that contain parameters defined in the MATLAB workspace.

Lock output scaling against changes by the autoscaling tool. Use this check box to indicate whether you want to prevent Simulink from replacing the current fixed-point type with a type chosen by the autoscaling tool. See “Automatic Scaling” in Simulink Fixed Point documentation for instructions on autoscaling fixed-point data in Simulink.

Sizing Function Arguments

You specify the size of arguments of Embedded MATLAB Function blocks in the Model Explorer or Ports and Data Manager.

Specifying Argument Size (p. 16-81)	Describes how to specify the size of data arguments
Inheriting Argument Sizes from Simulink (p. 16-81)	Explains how a data argument can inherit its size from Simulink inputs, outputs, or parameters
Specifying Argument Sizes with Expressions (p. 16-83)	Describes valid expressions for specifying the size of data arguments

Specifying Argument Size

To examine or specify the size of an argument, follow these steps:

- 1 From the **Embedded MATLAB Editor**, select **Model Explorer** or **Edit Data/Ports** from the **Tools** menu.
- 2 In the **Contents** pane, click the row that contains the data argument.
- 3 Enter the size of the argument in one of two places in the Model Explorer or Ports and Data Manager:
 - Size field of the Data properties dialog, located in the Dialog pane
 - Size column in the row that contains the data argument, located in the Contents pane

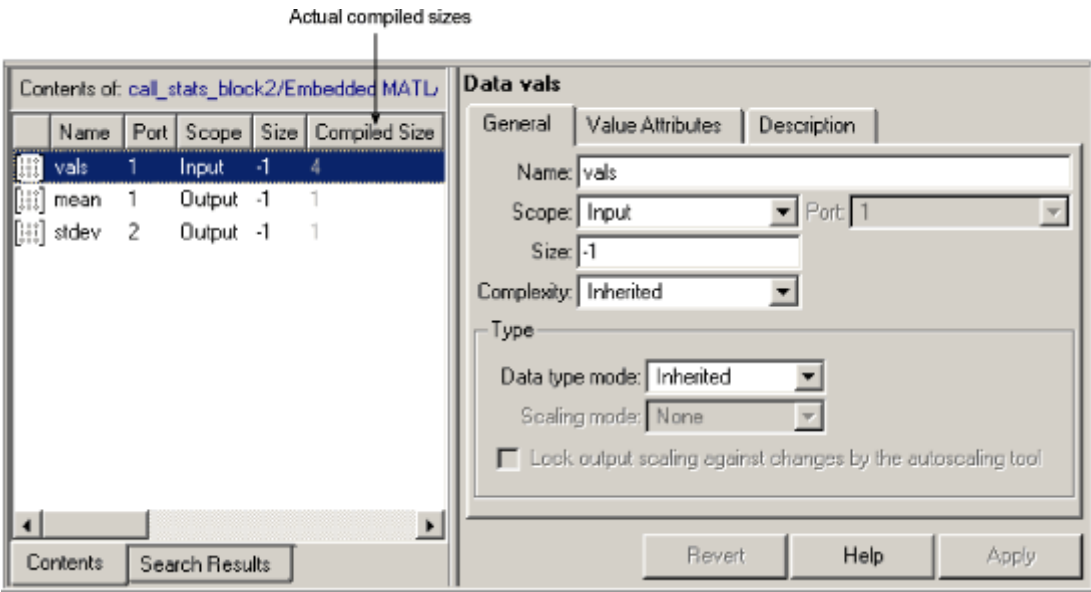
Note The default value is -1, indicating that size is inherited, as described in “Inheriting Argument Sizes from Simulink” on page 16-81.

Inheriting Argument Sizes from Simulink

Size defaults to 1, which means that the data argument inherits its size from Simulink based on its scope, as follows:

For Scope	Inherits Size
Input	From the Simulink input signal connected to the argument
Output	From the Simulink output signal connected to the argument
Parameter	From the Simulink or MATLAB parameter to which it is bound. See “Parameter Arguments in Embedded MATLAB Functions” on page 16-85.

After you compile the model, the **Compiled Size** column of the Model Explorer or Ports and Data Manager displays the actual size used in the compiled simulation application:



To conveniently compile the model from the **Embedded MATLAB Editor**, click the Build icon:



The size of an output argument is the size of the value that is assigned to it. If the expected size in Simulink does not match, a mismatch error occurs during compilation of the model.

Note No arguments with inherited sizes are allowed for Embedded MATLAB Function blocks in a library.

Specifying Argument Sizes with Expressions

The size of a data argument can be a scalar value or a MATLAB vector of values.

To specify size as a scalar, set the Size property to 1 or leave it blank. To specify size as a vector, enter an array of up to two dimensions in [row column] format where

- The number of dimensions equals the length of the vector
- The size of each dimension corresponds to the value of each element of the vector

For example, a value of [2 4] defines a 2-by-4 matrix. To define a row vector of size 5, set the **Size** field to [1 5]. To define a column vector of size 6, set the **Size** field to [6 1] or just 6. You can enter a MATLAB expression for each [row column] element in the **Size** field. Each expression can use one or more of the following elements:

- Numeric constants
- Arithmetic operators, restricted to +, -, *, and /
- Parameters defined in the MATLAB workspace or the parent Simulink masked subsystem
- Calls to the MATLAB functions min, max, and size

The following examples are valid expressions for size:

```
k+1
size(x)
min(size(y),k)
```

In these examples, k, x, and y are variables of scope **Parameter**.

Once you build the model, the **Compiled Size** column of the Model Explorer or Ports and Data Manager displays the actual size used in the compiled simulation application.

Parameter Arguments in Embedded MATLAB Functions

Parameter arguments for Embedded MATLAB Function blocks do not take their values from Simulink signals. Instead, their values come from parameters defined in a parent Simulink masked subsystem or variables defined in the MATLAB base workspace. Using parameters allows you to pass a read-only constant in Simulink to the Embedded MATLAB Function block.

Use the following procedure to add a parameter argument to a function for an Embedded MATLAB Function block.

- 1** In the **Embedded MATLAB Editor**, add an argument to the function header of the Embedded MATLAB Function block.

The name of the argument must be identical to the name of the masked subsystem parameter or MATLAB variable that you want to pass to the Embedded MATLAB Function block. For information on declaring parameters for masked subsystems in Simulink, see “Mask Editor” on page 12-16.

- 2** Bring focus to the Embedded MATLAB Function block in Simulink.

The new argument appears as an input port in the Simulink diagram.

- 3** In the **Embedded MATLAB Editor**, select **Model Explorer** or **Edit Data/Ports** from the **Tools** menu.

- 4** In the **Contents** pane, click the row that contains the new argument.

- 5** Set the scope of the data argument to **Parameter**, either in the Data properties dialog in the Dialog pane or in the Scope column in the Contents pane.

- 6** Examine the Embedded MATLAB Function block in Simulink.

The input port no longer appears for the parameter argument.

Note Parameter arguments appear as arguments in the function header of the Embedded MATLAB Function block to maintain MATLAB consistency. This lets you test functions in an Embedded MATLAB Function block by copying and pasting them to MATLAB.
