

# Basic Tutorial

---

<b>Overview of the Basic Tutorial</b>	3-2
<b>Running a Simulation on the Host PC</b>	3-3
Running a Simulation of Your Model	3-4
<b>Creating a Target Application</b>	3-6
Booting the Target PC	3-6
Entering the Simulation Parameters	3-7
Building and Downloading the Target Application	3-11
Troubleshooting the Build Process	3-12
<b>Running the Target Application on the Target PC</b>	3-12
Starting and Stopping the Target Application	3-13
<b>Signal Logging</b>	3-15
Plotting Outputs and States	3-15
Plotting Task Execution Time	3-16
<b>Signal Tracing with the Host Scope</b>	3-18
Creating a Scope Object and Selecting Signals	3-19
Closing All Scope Windows	3-25
<b>Signal Tracing with the Target Scope</b>	3-26
Creating a Target Scope Object and Selecting Signals	3-26
<b>Signal Tracing with xPC Target Functions</b>	3-31
Creating a Target Scope Object and Selecting Signals	3-31
<b>Parameter Tuning Using Simulink External Mode</b>	3-34
Setting Up Simulink in External Mode	3-35
Changing Simulink Block Parameters	3-36
<b>Parameter Tuning with xPC Target Commands</b>	3-38
Changing the Target Object Properties	3-38

## Overview of the Basic Tutorial

This tutorial explains the basic functions of xPC Target by using a simple Simulink model. The model is an oscillator with a square wave input. Since this model does not have I/O blocks, you can follow this tutorial regardless of whether you have I/O hardware on your target PC. This chapter includes the following sections.

To create and run a simulation of your model:

- “Running a Simulation on the Host PC”

To create and run a target application:

- “Creating a Target Application”
- “Running the Target Application on the Target PC”

To acquire signal data from your target application:

- “Signal Logging”
- “Signal Tracing with the Host Scope”
- “Signal Tracing with the Target Scope”
- “Signal Tracing with xPC Target Functions”

To change parameters in your target application:

- “Parameter Tuning Using Simulink External Mode”
- “Parameter Tuning with xPC Target Commands”

For an extended model with I/O blocks to explain I/O hardware and I/O drivers, see “Creating a Simulink Model with I/O Driver Blocks” on page 4-2.

For an extended model with xPC Target scope blocks to create scope objects, see “Creating a Model with xPC Target Scope Blocks” on page 4-11.

## Running a Simulation on the Host PC

You use Simulink in normal mode to observe the behavior of your model in nonreal-time. This section includes the following topics:

- “Loading a Simulink Model”
- “Running a Simulation of Your Model”

For procedures to run your target application in real-time, see “Creating a Target Application” on page 3-6.

### Loading a Simulink Model

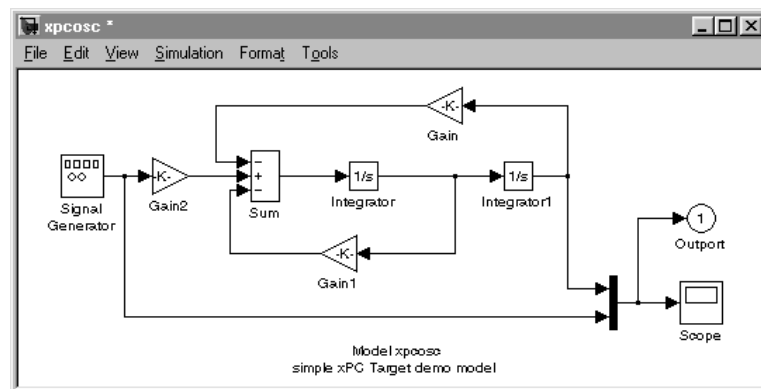
Loading a Simulink model moves information about your model, including the block parameters, into the MATLAB workspace.

After you create and save a Simulink model, you can load it back into the MATLAB workspace. This procedure uses the Simulink model `xpcosc.mdl` as an example.

- 1 In the MATLAB command window, type

```
xpcosc
```

MATLAB loads the oscillator model and displays the Simulink block diagram, as shown below.



Your next task is to run a simulation of your model in nonreal-time. See “Running a Simulation of Your Model”.

### Running a Simulation of Your Model

You run a simulation of your model in nonreal-time to observe the behavior of your model.

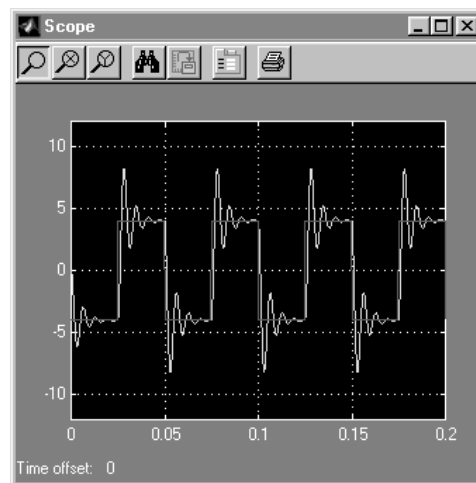
After you load your Simulink model into the MATLAB workspace, you can run a simulation. This procedure uses the Simulink model `xpcosc.mdl` as an example and assumes you have already loaded that model.

#### Simulating the Model Using the Simulink Graphical User Interface

To simulate the oscillator model, use the following procedure.

- 1 From the **Simulation** menu, click **Normal**, and then click **Start**.
- 2 Double-click the output scope block.

Simulink opens a scope window showing the output of the model.



- 3 You can either let the simulation run to the stop time, or stop the simulation manually. To stop the simulation manually, from the **Simulation** menu, click **Stop**.

### Simulating the model using the MATLAB command line interface

To simulate the oscillator model, use the following procedure.

- 1 In the MATLAB command window, type

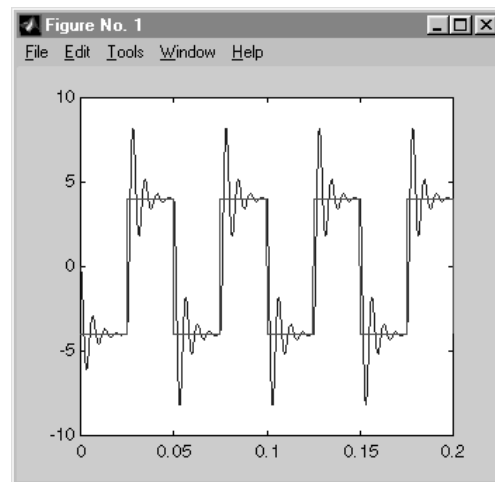
```
sim('xpcosc')
```

Simulink runs a simulation in normal mode.

- 2 After Simulink finishes the simulation, type

```
plot(tout, yout)
```

MATLAB opens a plot window and displays the output response. Since the signal generator signal is added to the output block, its response is also shown in the screen below.



Your next task is to create a target application. See “Creating a Target Application” on page 3-6.

## Creating a Target Application

You use the target application to observe the behavior of your model in real-time.

This section includes the following topics:

- “Booting the Target PC”
- “Entering the Simulation Parameters”
- “Building and Downloading the Target Application”
- “Troubleshooting the Build Process”

For procedures to simulate your model in nonreal-time, see “Running a Simulation on the Host PC” on page 3-3.

### Booting the Target PC

Booting your target computer loads and starts the xPC Target kernel. The loader then waits for xPC Target to download your target application from your host PC to your target PC.

After you have configured xPC Target using the Setup window, and created a target boot disk for that setup, you can boot your target PC. You need to boot the target computer before building the target application because the build process automatically downloads the target application to your target PC.

- 1 Insert the target boot disk into your target PC drive.
- 2 Turn on the target PC or press the reset button.

The target PC displays the following screen.

Loaded App: <b>1MB free</b>	
Memory: <b>61MB</b>	
Mode: <b>loader</b>	
Logging: -	
StopTime: -	
SampleTime: -	
AverageTET: -	
Execution: -	
	-----
	* xPC Target 1.0, (c) 1996-99 The MathWorks Inc. *
	-----
	System: Host-Target Interface is RS232 (COM1/2)
	System: COM1 detected

Your next task is to enter the simulation and real-time run parameters for Real-Time Workshop. See “Entering the Simulation Parameters” on page 3-7

**Possible Problem.** When booting the target PC, it might display the message:

```
xPC Target 1.0 loading kernel . . @@@@
```

The target PC displays this message when it cannot read and load the kernel from the target boot disk. The probable cause is a bad disk.

**Solution.** The solution is to reformat the disk or use a new preformatted floppy disk and create a new target boot disk.

## Entering the Simulation Parameters

The simulation or real-time run parameters are part of Simulink and Real-Time Workshop. They give information to Real-Time Workshop for how it builds the target application from your Simulink model.

After you load a Simulink model and boot your target PC, you can enter the simulation parameters for Real-Time Workshop. This procedure uses the Simulink model `xpcosc.mdl` as an example and assumes you have already loaded that model.

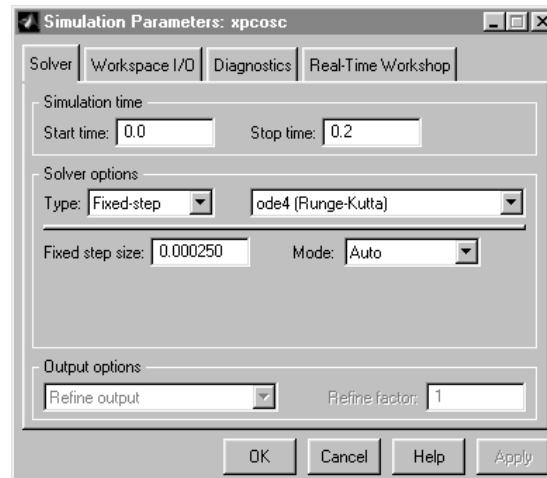
- 1 In the Simulink block window, and from the **Simulation** menu, click **Parameters**. In the **Simulation Parameters** dialog box, click the **Solver** tab.

The Solver property sheet opens.

- 2 In the **Start time** box, enter **0** seconds. In the **Stop time** box, enter **0.2** second.
- 3 From the **Type** list, choose **Fixed-step**. From the integration algorithm list choose **ode4 (Runge-Kutta)**. In the **Fixed step size** box, enter 0.00025 second (250 microseconds).

If you find that 0.000250 second results in overloading the CPU on your target PC, try a slower **Fixed step size** such as 0.002 second.

The Solver property sheet should look like the screen shown below.



- 4 At the top of the **Simulation Parameters** dialog box, click the **Workspace I/O** tab.

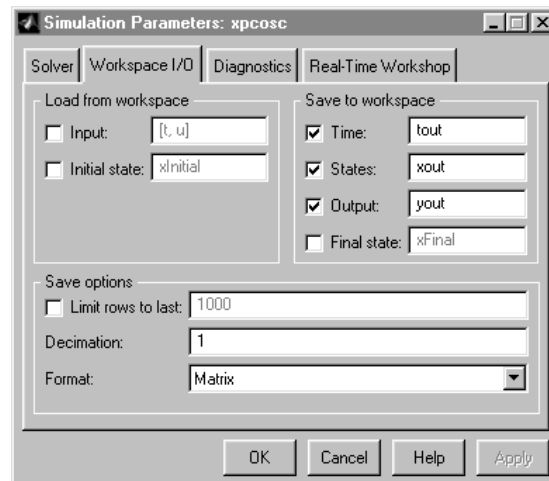
The **Workspace I/O** property sheet opens. This property sheet indicates which model signals are logged during the simulation of your model or the real-time run of your target application.

To save (log) data from signals other than the state values, you need to add output blocks to your Simulink model.

- 5 In the **Save to workspace** section, choose the **Time**, **State**, and **Output** check boxes.



The Workspace I/O property sheet looks like the screen shown below.



- 6 At the top of the **Simulation Parameters** dialog box, click the **Real-Time Workshop** tab.

The Real-Time Workshop property sheet opens.

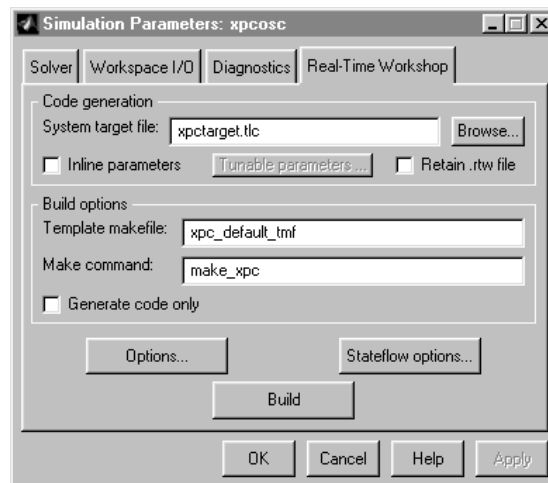
- 7 Click the **Browse** button.

The System Target File Browser opens.

- 8 Select the following Target Language Compiler file.  
xpctarget.tlc xPC Target

### 9 Click **Ok**.

The System target file `xpctarget.tlc`, the Template makefile `xpc_default_tmf`, and the Make command `make_xpc` are automatically entered into the property sheet. The Real-Time Workshop property sheet should now look like the screen shown below.



### 10 Click **OK**.

Your next task is to create (build) the target application. See “Building and Downloading the Target Application” on page 3-11.

## Building and Downloading the Target Application

You use the xPC Target *build process* to generate C code, compile, link, and download the target application to your target PC.

After you enter your changes in the **Simulation Parameters** dialog box, you can build the target application. This procedure uses the Simulink model `xpcosc.mdl` as an example, and assumes you have loaded that model.

- 1 In the Simulink block window, and from the **Tools** menu, click **RTW Build**.

After the compiling, linking, and downloading process, a target object is created with properties and associated methods (functions). The default name of the target object is `tg`. For more information about the target object, see “Target Object Properties and Commands” on page 7-4.

On the host computer, the following lines are displayed after a successful build process.

```
### Starting Real-Time Workshop build procedure for model: xpcosc
. . .
### Successful completion of xPC Target build procedure for model: xpcosc
```

The target PC displays the following information.

Loaded App: <b>xpcosc</b>	* xPC Target 1.0, (c) 1996-99 The MathWorks Inc. *
Memory: <b>61MB</b>	-----
Mode: <b>RT, single</b>	System: Host-Target Interface is RS232 (COM1/2)
Logging: <b>t x y tet</b>	System: COM1 detected
StopTime: <b>0.2 d</b>	System: download started...
SampleTime: <b>0.00025</b>	System: download finished
AverageTET: <b>-</b>	System: initializing application...
Execution: <b>stopped</b>	System: initializing application finished

If you do not have a successful build, see “Troubleshooting the Build Process” on page 3-12.

Your next task is to run the target application in real-time on your target computer. See “Running the Target Application on the Target PC” on page 3-12.

### Troubleshooting the Build Process

If the host PC and target PC are not properly connected or you have not correctly entered the properties, the download process is terminated after about 5 seconds with a time-out error.

To correct the problem, open the Setup window. In the MATLAB command window, type

```
xpcsetup
```

Check and, if necessary, make changes to the communication properties, update the properties, and recreate the target boot disk. For information on the procedures, see either “Setting the Environment for Serial Communication” on page 2-12 or “Setting the Environment for Network Communication” on page 2-19, and then see “Creating a Target Boot Disk for the Target PC” on page 2-22.

### Running the Target Application on the Target PC

During the build process, a target object was created with properties and associated commands (methods). You control the target application by changing the target object properties with the target object commands.

For more detailed information about the target object properties and commands (methods), see “Target Object Properties and Commands” on page 7-4.

This section includes the following topic:

- “Starting and Stopping the Target Application”

## Starting and Stopping the Target Application

You run your target application in real time to observe the behavior of your model with generated code.

After xPC Target downloads your target application to your target PC, you can run the target application. This procedure uses the Simulink model `xpcosc.mdl` as an example, and assumes you have created and downloaded the target application for that model. The default name of the target object is `tg`.

- 1 Start the target application. In the MATLAB command window, type either `start(tg)` or `+tg`

On the host screen, and in the MATLAB command window, the status of the target object changes from stopped to running.

xPC Object

Connected	= Yes
Application	= xpcosc
Mode	= Real - Time Single-Tasking
Status	= running

On the target screen, the Simulation line changes from stopped to running and the AverageTET line periodically updates with a new value.

Loaded App: xpcosc	System: COM1 detected
Memory: 61MB	System: download started...
Mode: RT, single	System: download finished
Logging: t x y tet	System: initializing application...
StopTime: 0.2 d	System: initializing application finished
SampleTime: 0.00025	System: execution started (sample time: 0.000250)
AverageTET: 8.608e-006	System: execution stopped at 0.200000
Execution: stopped	minimal TET: 0.000008 at time 0.003500
	maximal TET: 0.000011 at time 0.000500

After running for 0.2 second, the target application stops.

- 2 Change the final time between runs. For example, to change the stop time to 1000 seconds, type either

`set(tg, 'StopTime', 1000)` or `tg.StopTime = 1000`

- 3 Change the sample time between runs. For example, to change the sample time to 0.01 second, type either

`set(tg, 'SampleTime', 0.01)` or `tg.SampleTime = 0.01`

Although you can change the sample time in between different runs, you can only change the sample time without rebuilding the target application under certain circumstances.

If you choose a sample time that is too small, a CPU overload can occur. If a CPU overload occurs, the target object property CPU Overload changes to detected. In that case, change the **Fixed step size** in the Solver property sheet to a larger value.

- 4 Restart the simulation. Type either

`start(tg)` or `+tg`

- 5 Stop running the target application. Type either

`stop(tg)` or `-tg`

Your next task is to log and observe the signals from your target application in real-time. See one of the following topics:

- “Signal Logging” on page 3-15
- “Signal Tracing with the Host Scope” on page 3-18
- “Signal Tracing with the Target Scope” on page 3-26
- “Signal Tracing with xPC Target Functions” on page 3-31

## Signal Logging

Signal logging is the process for acquiring signal data during a real-time run, and after the run reaches its final time or you manually stop the run, transferring the data to the host PC and plotting the data.

Plotting the time, state, and output signals is possible only if you added output blocks to your Simulink model before the build process, and in the **Simulation Parameters** dialog box, checked the output boxes on the I/O-Workspace property sheet.

Plotting the task execution time is possible only if you checked the **LogTET** check box in the **Code Generation Options** dialog box. For more information, see “Workspace I/O Properties” on page 6-4 and “Code Generation Dialog Box” on page 6-8.

This section includes the following topics:

- “Plotting Outputs and States”
- “Plotting Task Execution Time”

### Plotting Outputs and States

You plot the outputs and states of your target application to observe the behavior of your model, or to determine the behavior when you vary the input signal.

After you run a target application, you can plot the state and output signals. This procedure uses the Simulink model `xpcosc.mdl` as an example, and assumes you have created and downloaded the target application for that model.

- 1 In the MATLAB command window, type either  
`start(tg)` or `+tg`

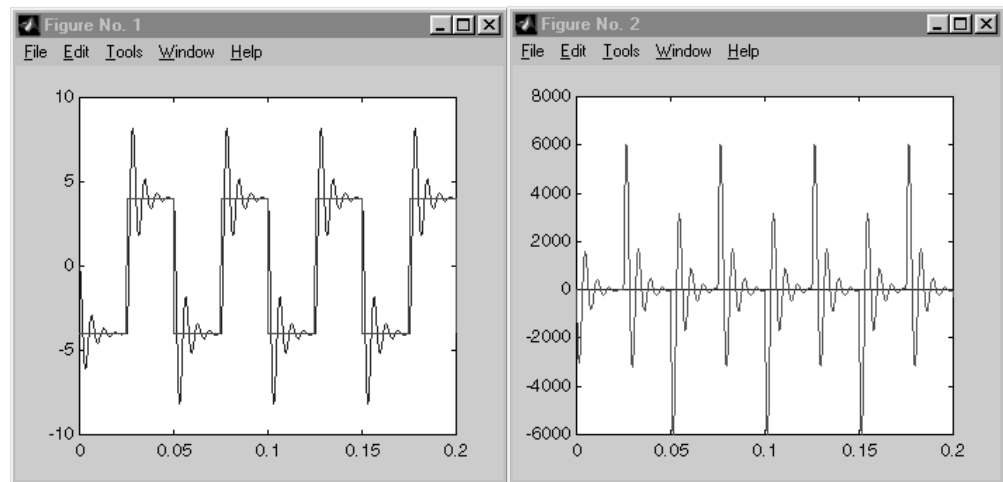
The target application starts and runs until it reaches the Final Time.

The outputs are those signals connected to Simulink output blocks. The model `xpcosc.mdl` has just one output block labeled 1 and there are two states. This output block shows the signal leaving the block labeled **Integrator1** and **Signal Generator**.

- 2 Plot the signals from the output block and the states. Type

```
plot(tg.TimeLog, tg.OutputLog)
figure
plot(tg.TimeLog, tg.StateLog)
```

The plots shown below are the result of a real-time execution. To compare this plot with a plot for a nonreal-time simulation, see “Running a Simulation of Your Model” on page 3-4.



## Plotting Task Execution Time

The task execution time (TET) is the time to calculate the signal values for the model during each sample interval. If you have subsystems that run only under certain circumstances, plotting the TET would show when subsystems were executed and the additional CPU time required for those executions.

After you run a target application, you can plot the task execution time. This procedure uses the Simulink model `xpcosc.mdl` as an example, and assumes you have created and downloaded the target application for that model.

- 1 In the MATLAB command window, type either  
`start(tg)` or `+tg`

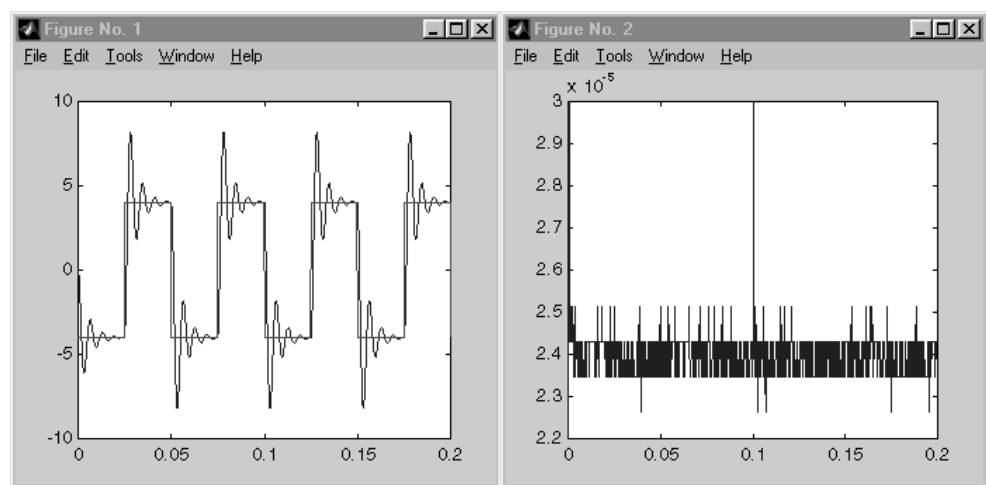
The target application starts and runs until it reaches the final time.



- 2 Plot the signals from the output block and the task execution time (TET).  
Type

```
plot(tg.TimeLog, tg.OutputLog)
figure
plot(tg.TimeLog, tg.TETLog)
```

The plots shown below are the result of a real-time run. The output is shown on the left and the task execution time is shown on the right.



- 3 Get information about the average task execution time. Type either  
`get(tg, 'AvgTET')` or `tg.AvgTET`

MATLAB displays the following information.

```
ans =  
0.000009
```

In the example above, the minimum TET was 8  $\mu$ s, the maximum TET 11  $\mu$ s, and the average TET 8  $\mu$ s. This means that the real-time task has taken about 3 % of the CPU performance (Average TET of 8  $\mu$ s / Sample time of 250  $\mu$ s).

## Signal Tracing with the Host Scope

Signal tracing is the process of acquiring and visualizing signals during a real-time run. It allows you to acquire signal data and visualize it on your target PC or upload the signal data and visualize it on your host PC while the target application is running.

Signal logging differs from signal tracing. With signal logging you can only look at a signal after a run is finished, and the entire log of the run is available. For information on signal logging, see “Signal Logging” on page 3-15.

With xPC Target, signal tracing is called *Scope* because this function is similar to using a digital oscilloscope. You can access the Scope functions indirectly through a Scope window, or directly by using xPC Target commands.

This section includes the following topics:

- “Creating a Scope Object and Selecting Signals”
- “Exporting Data from the Host Scope Window”
- “Closing All Scope Windows”

For procedures on using xPC Target commands for scopes, see “Signal Tracing with xPC Target Functions” on page 3-31.

## Creating a Scope Object and Selecting Signals

Opening a Scope window allows you to view signals with a graphical user interface (GUI).

After you create, download, and start running a target application, you can view signals. This procedure uses the Simulink model `xpcosc.mdl` as an example, and assumes you are running the target application for that model.

- 1 In the MATLAB command window, type  
`xpcscope`

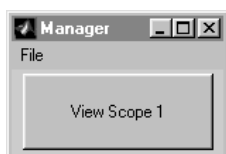
The Manager window opens. This window is the root-window of the Scope graphical interface.



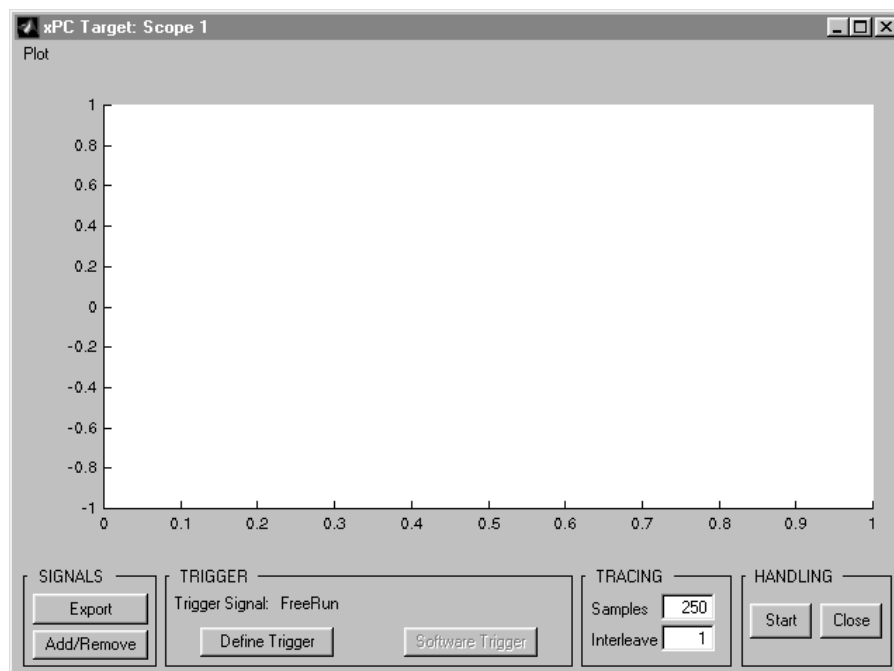
At this point, the window is empty because you need to define a specific scope.

- 2 From the **File** menu, click **New Scope**.

On the host PC, a new scope button appears on the Manager window and a new Scope window opens.



If the Scope window is in the background, on the Manager window, click the **View Scope 1** button. The Scope window moves to the foreground.



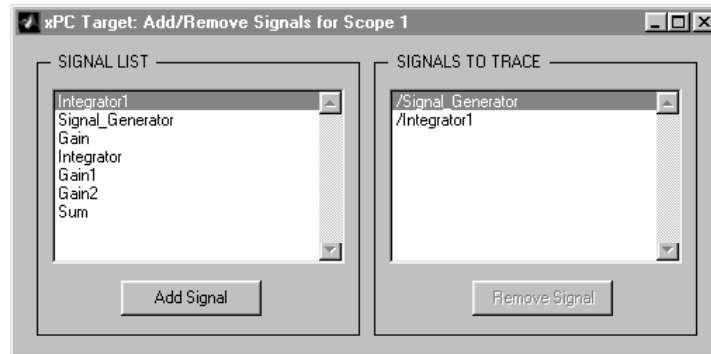
The scope window uses most of the area for signal plotting. At the bottom, there are controls to specify the scope functions.

The target PC displays the following message.

Scope: 1, created, type is host

- 3 In the Scope window, click the **Add/Remove** button

The **Add/Remove Signals** dialog box opens. It allows you to specify which signals to trace.

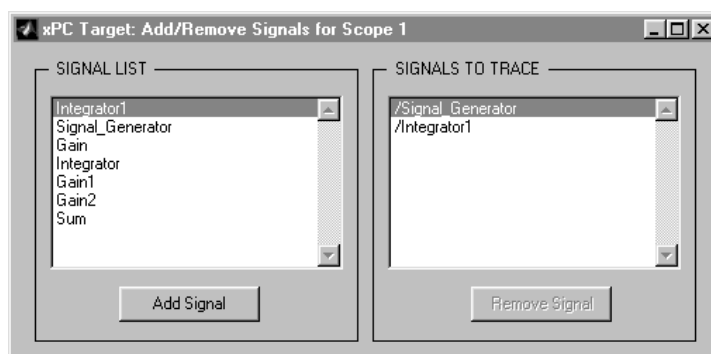


The **Signal list** box, displays all of the available signals from the target application. The names of the signals correspond to the block names within the Simulink model `xpcosc.mdl`. The block name indicates the output signal from that block.

Click a block name to highlight it, and then click the **Add Signal** button to move the signal to the **Signal trace** box on the right of the window. The **Signal trace** box contains the signals to be traced by this scope.

- 4 From the **Signal list** box, click **Integrator 1**, and then click the **Add Signal** button. Click **Signal Generator**, and then click the **Add Signal** button.

Changes to the **Add/Remove Signals** dialog box are shown below. The signals to trace can be removed by clicking the block name in the **Signal trace** box, and then clicking the **Remove Signal** button.



During the next steps, you can leave the **Add/Remove Signals** dialog box open, or close and reopen it without restrictions.

You can now start the scope. You also need to start a run before the signals are visible in the scope window. If you use a scope, set the final time to a value high enough to ensure the target application is running during the entire signal tracing session. The final time is set by changing the target object property `StopTime`.

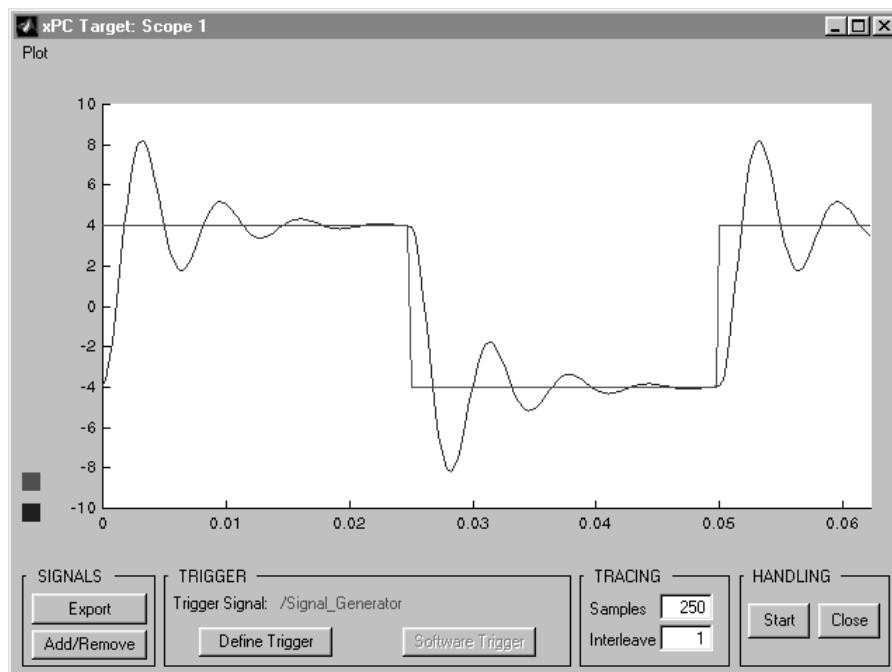
- 5 In the Scope window, click the **Start** button. In the MATLAB command window, type either  
`start(tg)` or `+tg`

The target application starts running.

You can start the scope and the target application in any order. The target application does not have to be running to start the scope or make changes to the scope properties. While the scope is running, the **Start** button on the Scope window changes to a **Stop** button.

If a target application is running and you start a scope, the host scope window acquires one data package, and then updates the signal graph. The time to collect one data package is equal to the number of samples multiplied by the sample time.

If you are using a host scope, there is a delay between collecting data packages because of the communication overhead from your target to host computers. If you are using a target scope, the target scope window is updated faster than when using a host scope.



## Exporting Data from the Host Scope Window

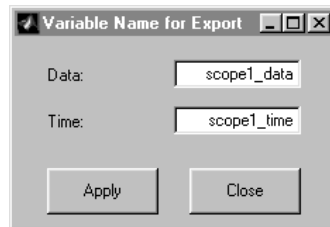
Exporting data from an xPC Target scope window gives you more flexibility than signal logging, which requires you to add output blocks to your Simulink model and activate the logging of signals. You can select which signals to collect, and you can capture unexpected outputs during a run.

This procedure uses a scope window as a graphical interface to move data from the last data package collected to the MATLAB workspace. For information on exporting data using a scope object and the properties `Data` and `Time`, see “Scope Object Properties and Commands” on page 7-12.

After you start and stop a run with a host or target scope, you can move data from the last data package collected to the MATLAB workspace. This procedure uses the Simulink model `xpcosc.mdl` as an example, and assumes you have completed a run with the target application.

- 1 In the xPC Target Scope window, and from the **Plot** menu, click **Variable Name for Export**.

The **Variable Name for Export** dialog box opens.



- 2 In the **Data** and **Time** text boxes, enter the name of the MATLAB variables to contain the data from the scope data package. Click the **Apply** button, and then click the **Close** button.

The default name for the time vector is `scopen_time`, and the default name for the signal vector is `scopen_data` where `n` is the scope number.

- 3 In the Scope window, click the **Export** button. You can export data regardless of whether a scope is started or stopped.
- 4 In the MATLAB command window, type  
`whos`

MATLAB displays a list of variables and their description. For example

Name	Size	Bytes	Class
<code>ans</code>	1x1	13958	xpc object
<code>scope1_data</code>	250x2	4000	double array
<code>scope1_time</code>	250x1	2000	double array
<code>t</code>	801x1	6408	double array
<code>tg</code>	1x1	14002	xpc object
<code>x</code>	801x2	12816	double array
<code>y</code>	801x1	6408	double array

Grand total is 5828 elements using 59592 bytes

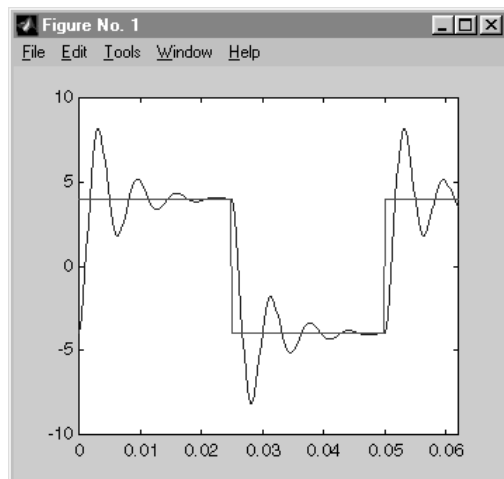
You can now save or further analyze the data using the MATLAB variables.



- 5 Plot the MATLAB variables. Type

```
plot(scope1_time, scope1_data)
```

MATLAB plots the data in a new window.



## Closing All Scope Windows

This procedure closes all scope windows that are open. It is useful since a scope can have a scope window, **Add/Remove** dialog box, and a **Trigger** dialog box opened at the same time.

- 1 Select the Manager window, then use one of the following procedures:

- From the **File** menu, click **Close All Scopes**.
- From the **File** menu, click **Close Scope Manager**.

A message box opens asking if you want to save the current scope state.

- 2 Use one of the following procedures:

- If you do not want to save the scope state, click **No**.
- If you want to save the scope state, click **Yes**. The **Save Scopes** dialog box opens. Enter the name of a file, and then click **Save**.

## Signal Tracing with the Target Scope

Signal tracing is acquiring signals while a real-time execution is running. It allows you to acquire signals and visualize them on the target PC.

This section includes the following topic:

- “Creating a Target Scope Object and Selecting Signals”

For a discussion of signal tracing and a brief description of a scope object, see “Signal Tracing with the Host Scope” on page 3-18.

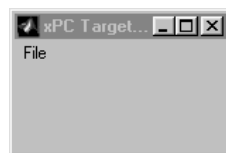
### Creating a Target Scope Object and Selecting Signals

Opening a target scope window allows you to select and view signals using a graphical user interface.

After you create, download, and start running a target application, you can view signals. This procedure uses the Simulink model `xpcosc.mdl` as an example, and assumes you are running the target application for that model.

- 1 In the MATLAB command window, type  
`xpctgscope`

The target scope Target Manager window opens. This window is the root-window of the Scope graphical user interface.



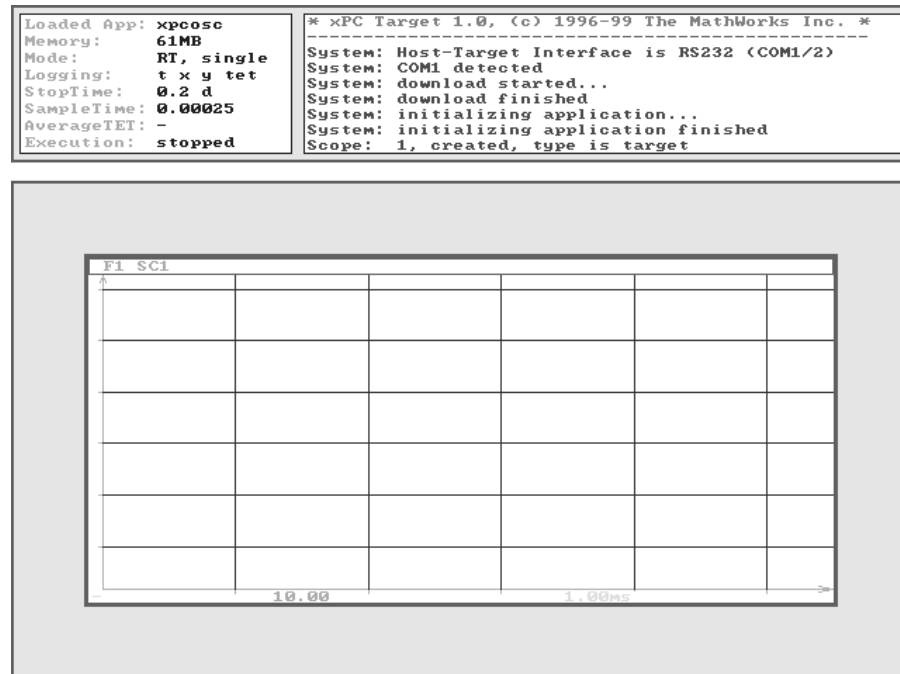
At this point, the window is empty because a specific scope has not been defined.

- 2 From the **File** menu, click **New Scope**.

On the host PC, a new scope button appears on the Target Manager window.

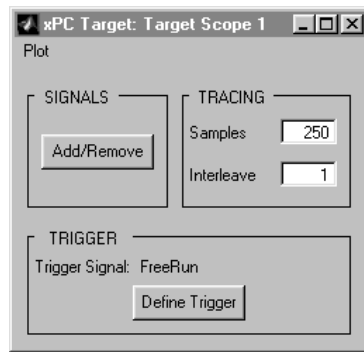


And on the target PC, a new target scope window opens.



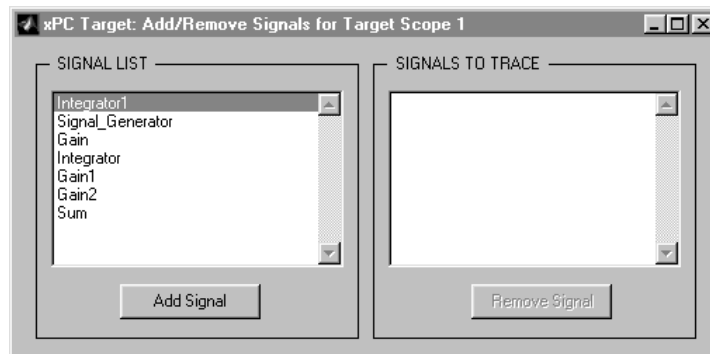
- 3 In the Target Manager window, right-click the scope button, and then click **Properties**.

The Target Scope 1 window opens.



4 Click the **Add/Remove** button.

The **Add/Remove Signals** dialog box opens. This dialog box allows you to specify which signals to trace.

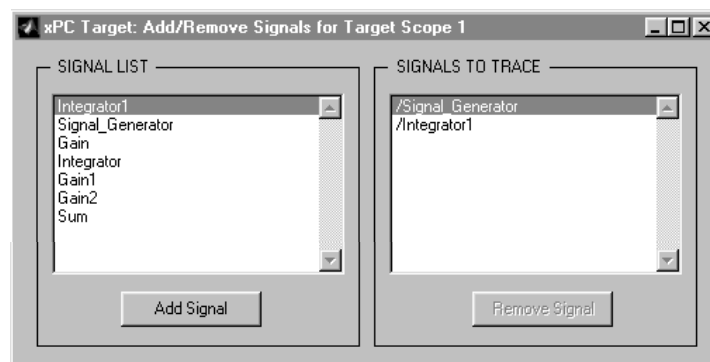


The **Signal list** box, displays all of the available signals from the target application. The names of the signals correspond to the block names within the Simulink model `xpcosc.mdl`. The block name indicates the output signal from that block.

Click a block name to highlight it, and then click the **Add Signal** button to move the signal to the **Signals to trace** box on the right of the window. The **Signals to trace** box contains the signals to be traced by this scope.

- 5 From the **Signal list** box, click **Integrator 1**, and then click the **Add Signal** button. Click **Signal Generator**, and then click the **Add Signal** button.

On the host PC, changes to the **Add/Remove Signals** dialog box are shown below. The signals to trace can be removed by clicking the block name in the **Signals to trace** box, and then clicking the **Remove Signal** button.



The target PC displays the following messages.

```
Scope: 1, created, type is target
Scope: 1, signal 1 added
Scope: 1, signal 0 added
```

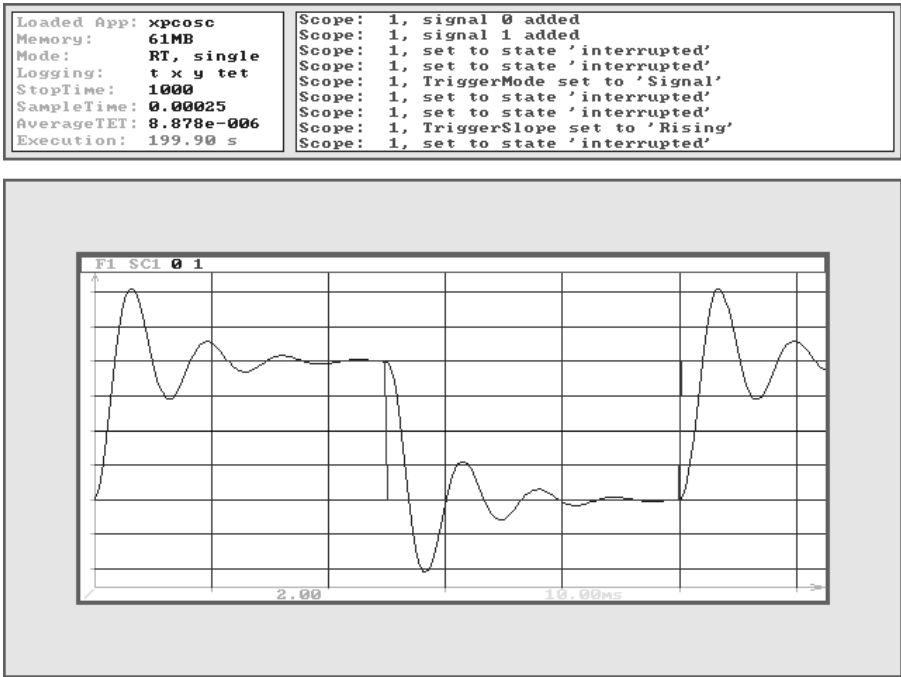
The line above the graph gives information about the target scope object. The string SC1 means that this graph corresponds to the scope object with an identifier equal to 1. The colored number 0 and number 4 are the signals added to this target scope. When you start signal tracing, the color of the traces corresponds to the color of the signal numbers.

- 6 Starting the target scope is slightly different than starting the host scope. In the Target Manager window, right-click the **Scope 1** button, and then click **Start**.

You also need to start running a target application before the signals are visible in the scope window. Type either

```
start(tg) or +tg
```

The plot frame on the target PC displays the signal traces and updates at a rate equal to the time to collect one data package, as shown below.



## Signal Tracing with xPC Target Functions

Signal tracing is acquiring signals while a real-time execution is running. It allows you to acquire signals and visualize them on the target PC.

This section describes how to signal trace using xPC Target functions instead of using the xPC Target graphical interface.

This section includes the following topic:

- “Creating a Target Scope Object and Selecting Signals”

For a discussion of signal tracing with a graphical user interface and a brief description of a scope object, see “Signal Tracing with the Host Scope” on page 3-18.

### Creating a Target Scope Object and Selecting Signals

Creating a target scope object allows you to select and view signals using the xPC Target functions.

After you create and download, the target application, you can view output signals. This procedure uses the Simulink model `xpcosc.mdl` as an example, and assumes you have build the target application for that model.

- 1 Increase the stop time. For example, to increase the stop time to 1000 seconds, in the MATLAB command window, type

```
tg.stoptime=1000
```

- 1 Start running your target application. Type either `start(tg)` or `+tg`

The target PC displays the following message.

```
System: simulation started (sample time: 0.0000250)
```

- 2 To get a list of parameters, type either `set(tg, 'ShowSignals', 'on')` or `tg.ShowSignals='on'`

The MATLAB command window displays a list of the target objects properties for the available signals. For example, the signals for the model `xpcosc.mdl` are shown below.

```
ShowSignals      =    On
Signals          =    0    : Integrator1
                  1    : Signal Generator
                  2    : Gain
                  3    : Integrator
                  4    : Gain1
                  5    : Gain2
                  6    : Sum
```

The signal numbers (0, 1 . . .6) are not properties of the target object. However, the Parameter identifiers (P0, P1, . . .P6) are properties of the target object.

- 3 To create a scope object displayed on the target PC with an identifier of 1 and an object name of `sc1`, type

```
sc1= addscope(tg, 'target', 1)
```

Or you could type

```
addscope(tg, 'target', 1)
sc1 = getscope(tg, 1)
```

- 4 To list the properties of the target scope object, type

```
sc1
```

The MATLAB command window displays a list of the scope object properties.

```
xPC Scope Object
```

```
Application      = xpcosc
ScopeId          = 1
Status           = Interrupted
Type             = Target
NumSamples       = 250
Decimation       = 1
TriggerMode      = FreeRun
```



```

TriggerSignal      = - 1
TriggerLevel       = 0
TriggerSlope       = Either
TriggerScope       = 1
Mode               = Redraw (Graphical)
YLimit             = Auto
Grid               = On
StartTime          = - 1.000000
Data               = Matrix (250 x 0)
Time               = Matrix (250 x 1)
Signals            = no Signals defined

```

- 5 Add signals to the scope object. For example, to add the Integrator1 and Signal Generator, type

```
addsignal(sc1, [0, 1])
```

The target PC displays the following messages.

```
Scope: 1, signal 0 added
```

```
Scope: 1, signal 1 added
```

After you add signals to a scope object, the signals are not shown on the target screen until you start the scope object.

- 6 Start the scope object. For example, to start the scope sc1, type either

```
start(sc1) or +sc1
```

The target screen plots the signals after collecting each data package. During this time you can observe the behavior of the signals while the scope is running.

- 7 Stop the scope. Type either

```
stop(sc1) or -sc1
```

The signals shown on the target PC stop updating while the target application continues running, and the target PC displays the following message.

```
Scope: 1, set to state 'interrupted'
```

## Parameter Tuning Using Simulink External Mode

You use Simulink external mode to connect your Simulink block diagram to your target application. The block diagram becomes a graphical user interface to your target application where you can change block parameter values and have those changes also made in your running target application.

This section includes the following topics:

- “What Is External Simulation Mode?”
- “Setting Up Simulink in External Mode”

Alternately, you can use xPC Target commands instead of using the Simulink window to change model parameters. You usually use these functions to program and batch process a series of runs. For an introduction to these commands, see “Parameter Tuning with xPC Target Commands” on page 3-38

### What Is External Simulation Mode?

External simulation mode is a feature of the Real-Time Workshop (RTW). It offers an easy way to change parameters in a target application regardless whether a target application is running or not:

- If a target application is not running, it allows you to prepare a model with a new set of parameters before the next run.
- If a target application is running, it allows you to change parameters and immediately see what effect changing parameters has on the behavior of your generated code.

After installing the Real-Time Workshop, the Simulink window has new menu commands and dialog boxes that relate to the external simulation mode:

- On the **Simulation** menu there are two new commands: **Normal** and **External**.

By selecting normal mode, Simulink is able to run a nonreal-time simulation of your model on your host PC. See “Running a Simulation on the Host PC” on page 3-3. You start and stop the nonreal-time simulation by using the **Start** and **Stop** commands from the **Simulation** menu.

By selecting external mode, Simulink is able to make a connection to your target application running on the target PC. This communication channel allows you to use a Simulink block diagram as a graphical user interface to the target application.

The menu item **Parameters** opens the **Simulations Parameters** dialog box and selects the **RTW Workshop** tab.

- On the **Tools** menu there are three new commands: **RTW Build**, **RTW Options**, and **External Mode Control Panel**.

## Setting Up Simulink in External Mode

You set up Simulink in external mode to establish a communication channel between your Simulink block window and your target application.

After you download your target application to your target PC, you can connect your Simulink model to the target application.

- 1 In the Simulink block window, and from the **Simulation** menu, click **External**.

A check mark appears next to the menu item **External**, and Simulink external mode is activated.

- 2 In the Simulink block window, and from the **Simulation** menu, click **Connect to Target**.

All of the current Simulink model parameters are downloaded to your target application. This downloading guarantees the consistency of the parameters between the host model and the target application.

The target PC displays the following message.

```
ExtM: Updating # parameters
```

Your next task is to change parameters using Simulink external mode, see “Changing the Target Object Properties” on page 3-38.

## Changing Simulink Block Parameters

In Simulink external mode, wherever you change parameters in the Simulink block diagram, Simulink downloads those parameters to the target application while it is running. This feature lets you change parameters in your program without rebuilding the Simulink model to create a new target application.

After you have downloaded a target application to your target PC, and set up Simulink in external mode, you can change parameters in your target application by changing parameters in your Simulink model. This procedure uses the Simulink model `xpcosc.mdl` as an example, and assumes you have created and downloaded the target application for that model.

- 3 From the **Simulation** menu, click **Start Real-Time Code** or in the MATLAB command window, type either

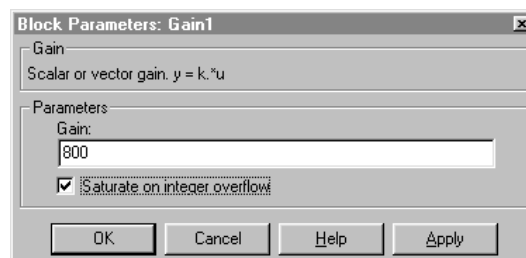
`start(tg)` or `+tg`

The target application begins running on the target PC, and the target PC displays the following message.

System: execution started (sample time: 0.000250)

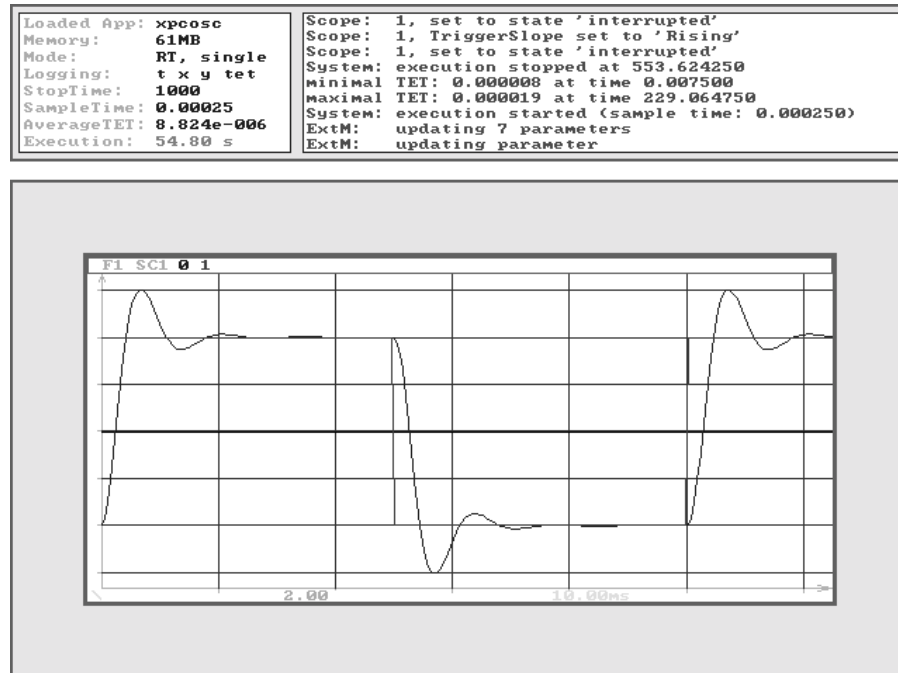
- 4 From the Simulation block diagram, click the block labeled **Gain1**.

The **Block Parameters: Gain1** parameter dialog box opens.



- 5 In the **Gain** text box, enter 800, and click **OK**.

As soon as you change a model parameter and click the **OK** button on the **Block Parameters: Gain1** dialog box, all of the changed parameters in the model are downloaded to the target application, as shown below.



- 6 From the **Simulation** menu, click **Disconnect from Target**.

The Simulink model is disconnected from the target application. Now, if you change a block parameter in the Simulink model, there is no effect on the target application. Connecting and disconnecting Simulink works regardless of whether the target application is running or not.

- 7 From the **Simulation** menu, click **Stop Real-Time Code**, or in the MATLAB command window, type either

`stop(tg)` or `-tg`

The target application on the target PC stops running, and the target PC displays the following messages.

```
System: simulation stopped
minimal TET: 0.000023 at time 1313.789000
maximal TET: 0.000034 at time 407.956000
```

## Parameter Tuning with xPC Target Commands

You use the xPC Target functions to change block parameters. You do not need to set Simulink in external mode, and you do not need to connect Simulink with the target application. You enter the xPC Target functions in the MATLAB command window, and they work whether the target application is running, or it is stopped.

This section includes the following topic:

- “Changing the Target Object Properties” on page 3-38

Alternately, you can also use the Simulink window as a graphical interface to the target application instead of using xPC Target commands. You use this graphical interface to interactively change parameters during a run. For an introduction to Simulink external mode, see “Parameter Tuning Using Simulink External Mode” on page 3-34.

### Changing the Target Object Properties

You can download parameters to the target application while it is running or between runs. This feature lets you change parameters in your program without rebuilding the Simulink model.

After you download a target application to the target PC, you can change block parameters using xPC Target functions. This procedure uses the Simulink model `xpcosc.mdl` as an example, and assumes you have created and downloaded the target application for that model.

- 1 In the MATLAB command window, type either  
`start(tg)` or `+tg`

The target PC displays the following message.

```
System: execution started (sample time: 0.001000)
```

- 2 Display a list of parameters. Type either  
`set(tg, 'ShowParameters', 'on')` or `tg.ShowParameters='on'`  
`tg`

The MATLAB command window displays a list of the target objects properties.

```
ShowParameters = On
Parameters=PROP VALUE      PARAMETER NAME      BLOCK NAME
      P0  0.000000      Initial Condition  Integrator1
      P1  4.000000      Amplitude          Signal Generator
      P2  20.000000     Frequency          Signal Generator
      P3  1000000.00     Gain              Gain
      P4  0.000000      Initial Condition  Integrator
      P5  400.000000     Gain              Gain1
      P6  1000000.00     Gain              Gain2
```

The Parameter identifiers (P0, P1, . . . P6) are properties of the target object. However, the Signal numbers (0,1, . . .) are not properties of the target object.

- 3 Change the gain. For example, to change the Gain1 block, type either  
`set(tg, 'p5', 800)` or `tg.p5=800`

As soon as you change a parameter, the changed parameters in the target object are downloaded to the target application. The target PC displays the following message.

```
Param: parameter 5 updated
```

And the plot frame updates the signals after running the simulation with the new parameters.

- 4 Stop the target application. In the MATLAB command window, type either  
`stop(tg)` or `-tg`

The target application on the target PC stops running, and the target PC displays the following messages.

```
System: simulation stopped
minimal TET: 0.000023 at time 1313.789000
maximal TET: 0.000034 at time 407.956000
```

