Simulink® for System and Algorithm Modeling

Modeling Logical Systems

The MathWorks
Training Services

© 2015 The MathWorks, Inc.

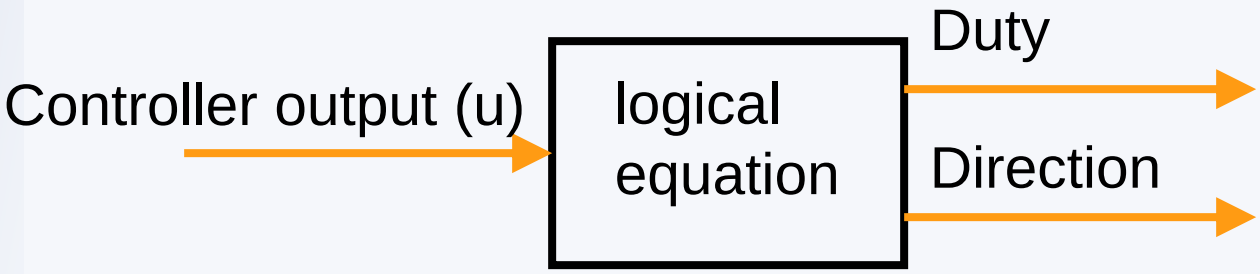
Outline

- Defining the system and identifying its components
- Modeling the system with equations
- Building a block diagram for the model
- Defining and configuring signal viewers
- Setting the configuration parameters
- Introducing zero crossing
- Variable-step versus fixed-step solvers
- Modifying the signal viewer parameters
- Simulating the model and analyzing the results
- Defining the maximum step size
- Modeling the system with Embedded MATLAB
- Simulating and comparing results

© 2006 The MathWorks, Inc.

Defining the System and Identifying Its Components

- The electronic throttle PI controller outputs a signal.



- The variables in this system are the controller output (u), duty, and direction:
  - The controller output, u, is the system input.
  - Duty and direction are the system outputs.
  - There are no states or parameters to this system. Both duty and direction are computed directly from the input signal.

© 2006 The MathWorks, Inc.

Modeling the System with Equations

- The system input is `u`. `Duty` and `Direction` are the system outputs.
- The duty is constrained to fall within the range of $0 <= duty <= 1$.

$$Duty = \begin{cases} |u| & if \quad 0 <= |u| <= 1 \\ 1 & if \quad |u| > 1 \end{cases}$$

$$Direction = \begin{cases} 1 & if \quad u >= 0 \\ -1 & if \quad u < 0 \end{cases}$$

© 2006 The MathWorks, Inc.

Building a Block Diagram for the Model

- Start by adding a From File block to the model. Label the block output u. This is the system input signal.
- Next, connect a Sign block to the u signal.
- Use Switch blocks to implement the two if statements.
- Limit the `duty` signal with a Saturation block.
  (0<= duty cycle <= 1)
- Connect the system outputs to Outport blocks.

$$Duty = \begin{Bmatrix} |u| & if & 0 <= |u| <= 1 \\ 1 & if & |u| > 1 \end{Bmatrix}$$

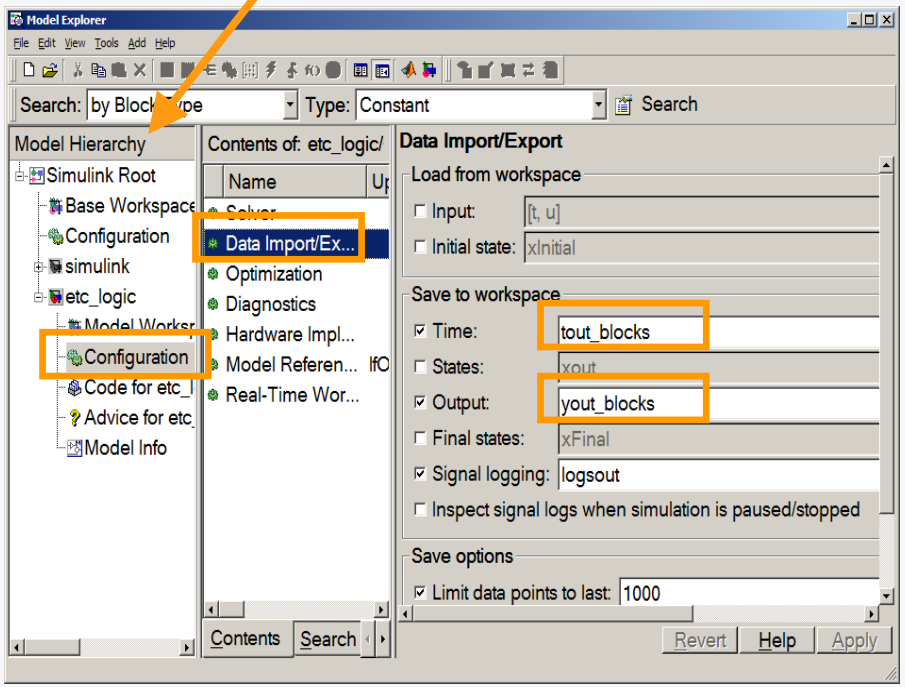
$$Direction = \begin{Bmatrix} 1 & if & u >= 0 \\ -1 & if & u < 0 \end{Bmatrix}$$

```
>> etc_logic
```

© 2006 The MathWorks, Inc.

Defining the Block Diagram I/O

- Set the **File Name** parameter for the From File block to `pi_output.mat`.
- To define the Outport block parameters, open the **Model Explorer** from the **View** menu of the model. Select the model you are working from **Model Hierarchy** in the **Model Explorer**.
- Check that **Time** and **Output** are selected.
- Modify the **Save to workspace** variable names for **Time** and **Output**.

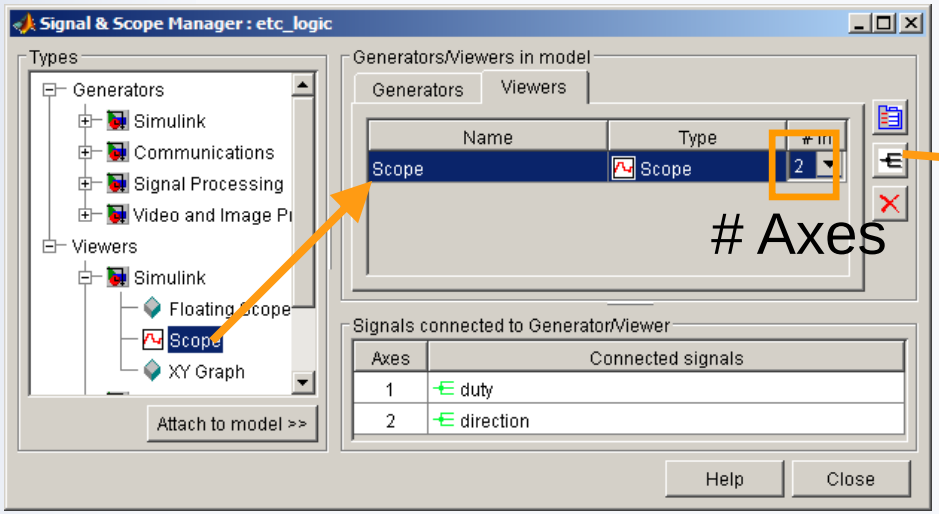


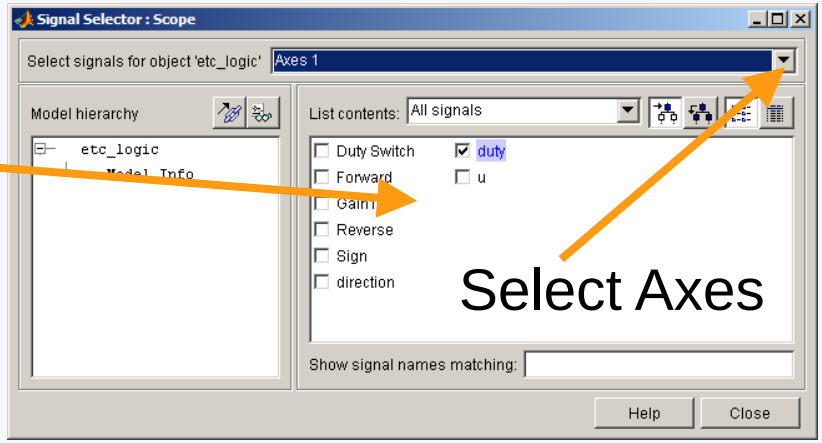
© 2006 The MathWorks, Inc.

Adding Signal Viewers

The Signal & Scope Manager lets you create and view signals without using blocks:

- Select **Tools > Signal and Scope Manager**.
- Go to **Types > Viewers > Simulink > Scope**.
- Create a Simulink Scope viewer with two axes.
- Connect the Duty signal to the first axis and the direction signal to the second axis of the viewer.

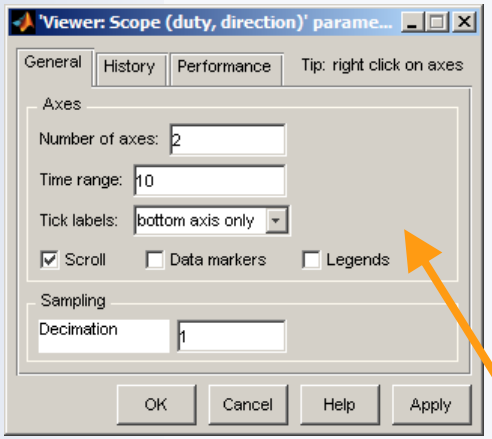


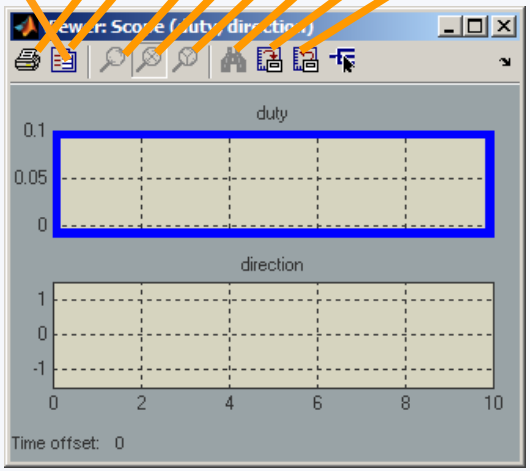


© 2006 The MathWorks, Inc.

Modifying the Viewer Parameters

- Open the Viewer by double-clicking the icon.



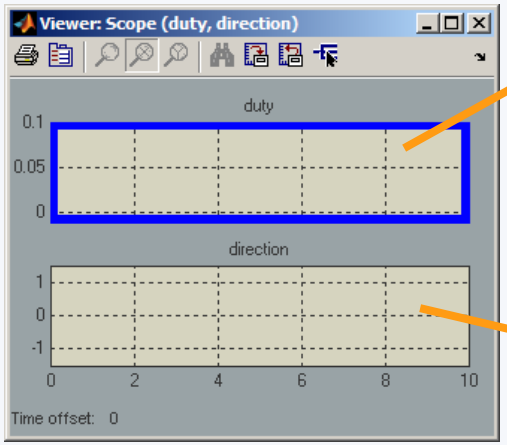


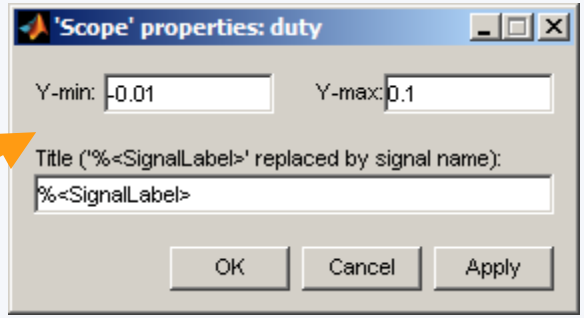
Print
Scope parameters
Zoom in
Zoom in x direction
Zoom in y direction
Autoscale
Save axes settings
Restore axes settings

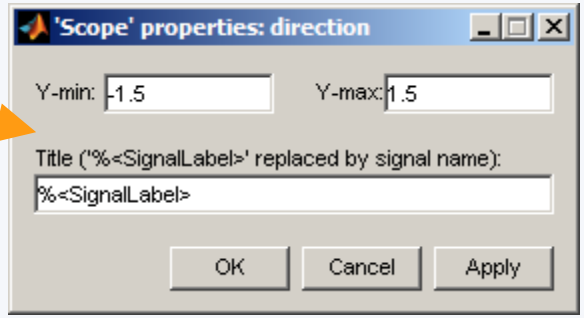
© 2006 The MathWorks, Inc.

Controlling the Vertical Limits of the Viewer

- Open the axis properties dialog box by selecting **Axes properties** from the context menu after right-clicking in the axis.
- Set the vertical axis limits using the **Y-min** and **Y-max** fields.







© 2006 The MathWorks, Inc.

Introducing Zero Crossings

- Simulink works by establishing a dialog between the system and the solver.
- Simulink can carry messages from the system to the solver.
- One important message is “There has been a zero crossing!”
- Zero crossings:
  - A signal changes sign.
  - A block changes mode.
- Missing zero crossings can result in inaccurate representation:
  - Bouncing of a ball

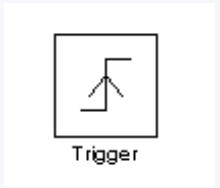
© 2006 The MathWorks, Inc.

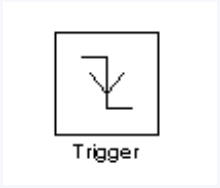
Zero Crossing Types

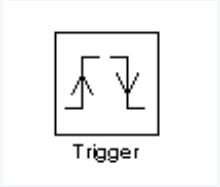
- Blocks with zero-crossing support:

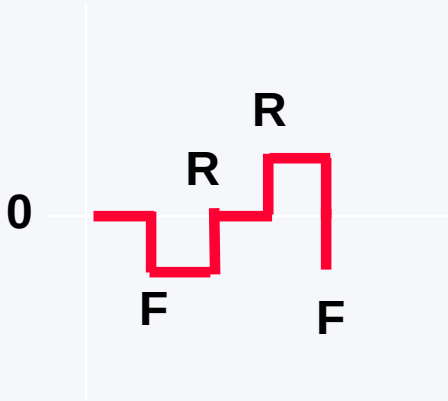
Abs	Backlash	MinMax	Dead Zone
Sign	Integrator	Switch	Relational Operator
Relay	Saturation	Step	Hit Crossing

- Each block defines its own zero crossings.
- Zero crossings can be
  - Rising — Signal rises to or through zero or signal leaves zero and becomes positive.
  - Falling — Signal falls to or through zero or signal leaves zero and becomes negative.
  - Either







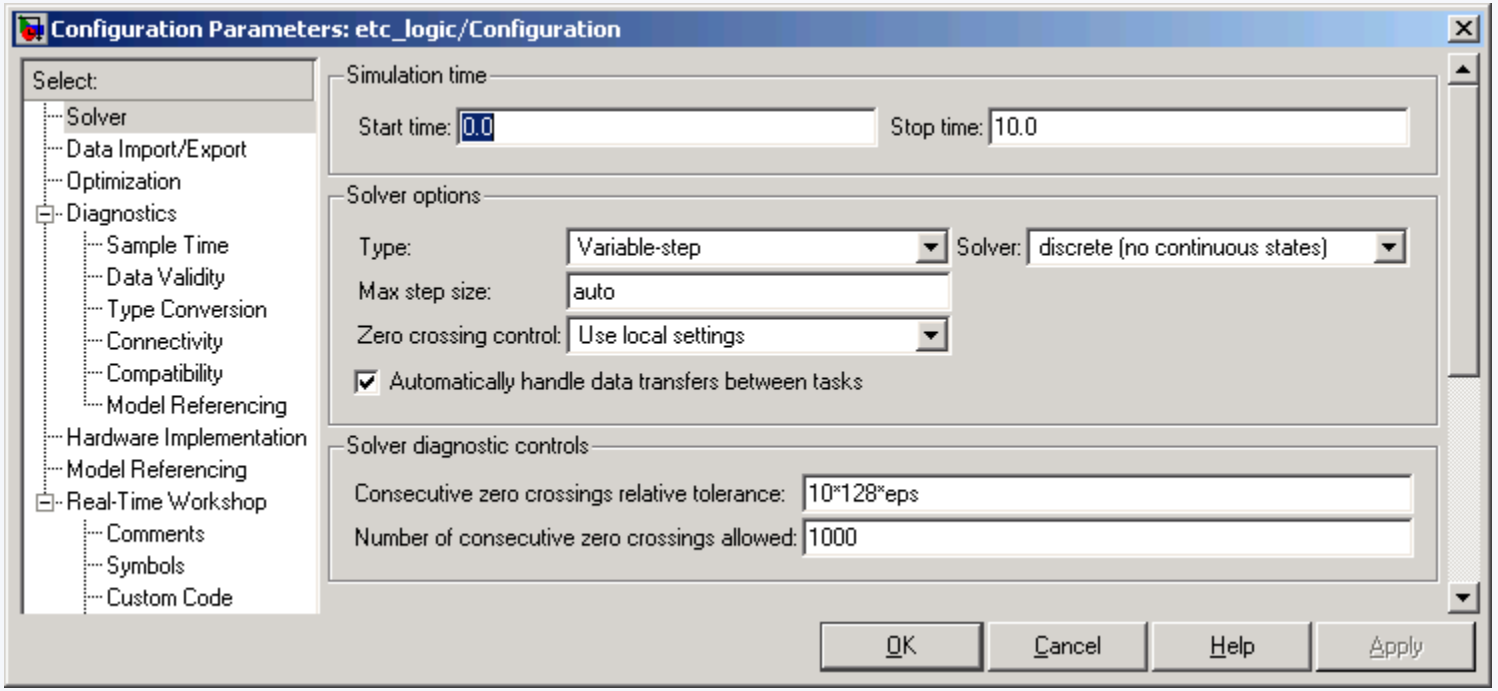


© 2006 The MathWorks, Inc.

Selecting a Solver for Zero-Crossing Detection

Select **Simulation → Configuration Parameters → Solver**

- For solver options:
  - Select a `Variable-Step`, `discrete` solver.
  - Set **Zero crossing** control to `Use local settings`



© 2006 The MathWorks, Inc.

Working with Zero Crossings

- Discrete blocks cause discontinuities.
  - Event detection is not required.
  - Events happen only at update times!
- Zero-crossing detection can be turned off for an entire system or individual blocks:
  - Configuration Parameters dialog box
  - Individual block parameters dialog box
- It is possible to create models whose responses go through zero several times in an infinitesimal period, and cause repeated detection of zero crossings at the same step, practically bringing simulation to a stop.

```
>> chatter
```

© 2006 The MathWorks, Inc.

Simulating the Model and Analyzing the Response

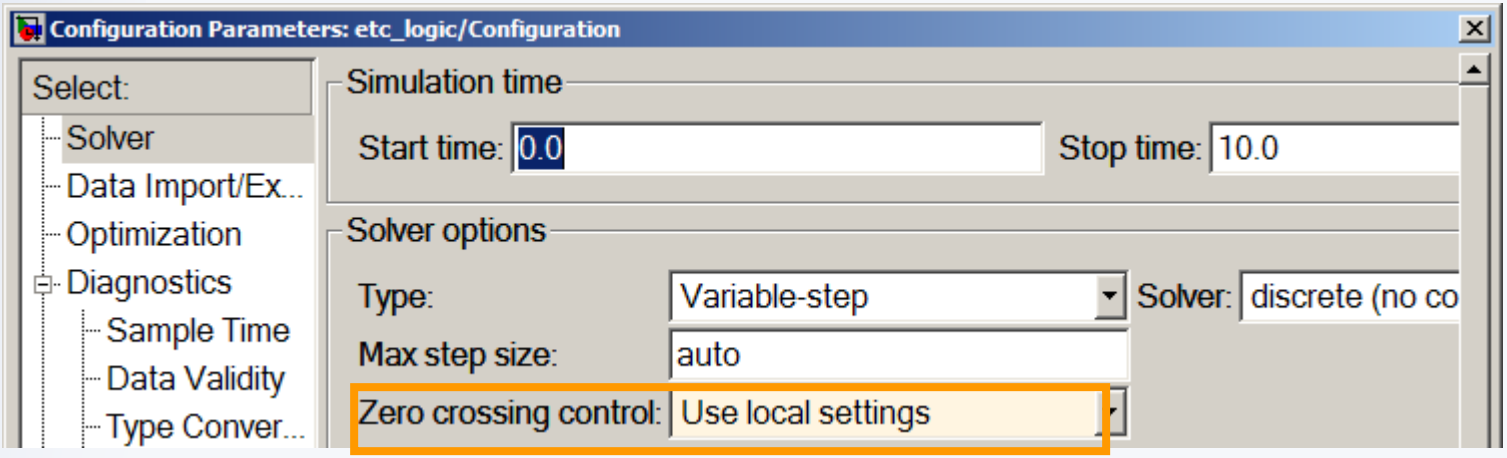
- To simulate the model from the command line, type

```
>> [t,x,y] = sim('etc_logic')
```

- To visualize the system output,

```
>> plot(t,y(:,1),t,y(:,2))
>> legend('Duty','Direction')
```

- Turn **Zero crossing control** off for the model and simulate again.
  - Is there a difference in the results?



© 2006 The MathWorks, Inc.

Defining the Max Step Size

- If the **Max step size** is `auto` and there are no zero crossings or continuous states in the model, Simulink computes a step size of

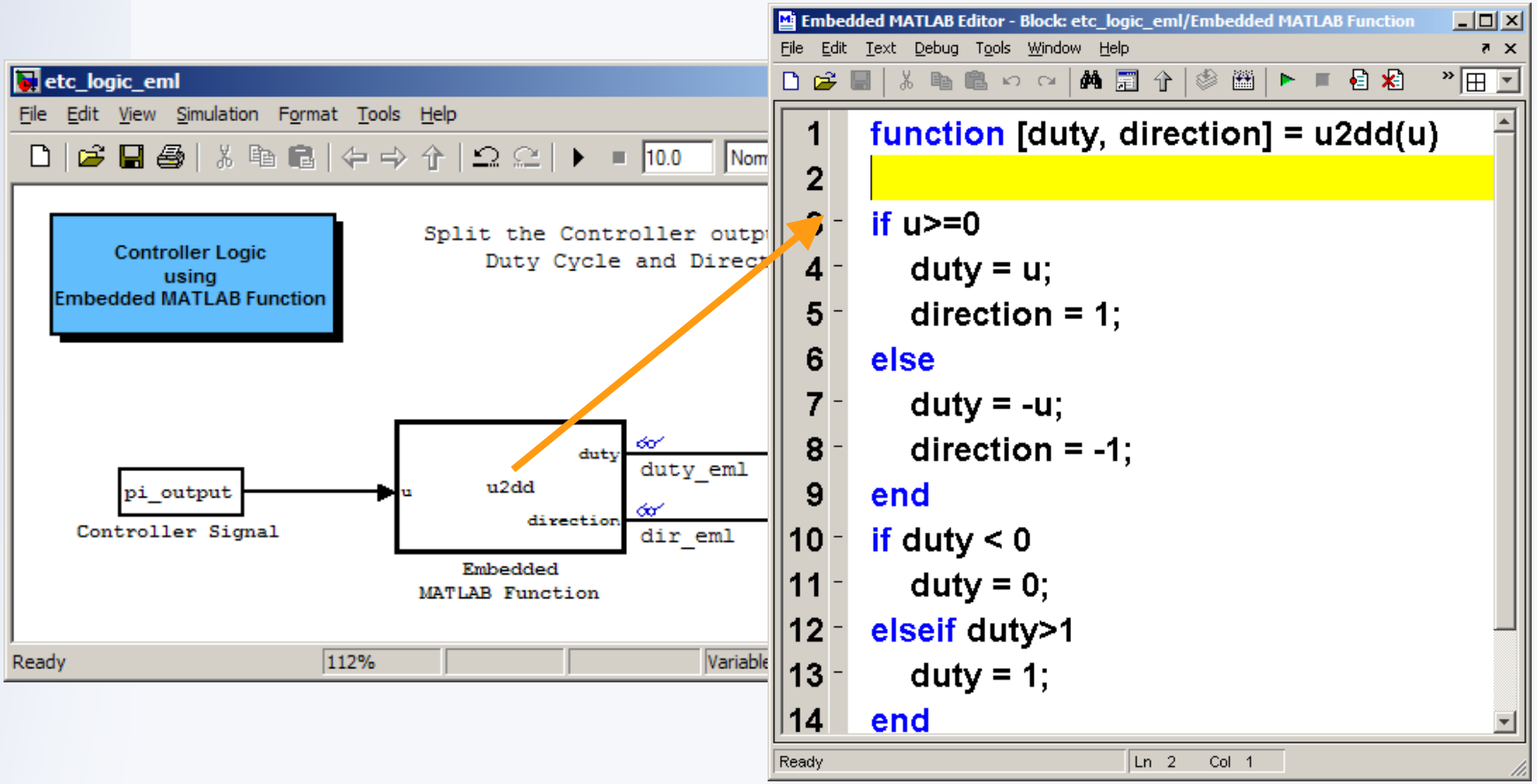
$$h = \frac{t_{stop} - t_{start}}{50}$$

- This is why you have 51 points in the simulation when zero-crossing detection is turned off.
- You can force a smaller step size by specifying it.

```
>> set_param('etc_logic','ZeroCrossControl','Disable')
>> [t_off,x_off,y_off]= sim('etc_logic');
>> length(t_off)
```

© 2006 The MathWorks, Inc.

Modeling Logic with Embedded MATLAB Function Blocks

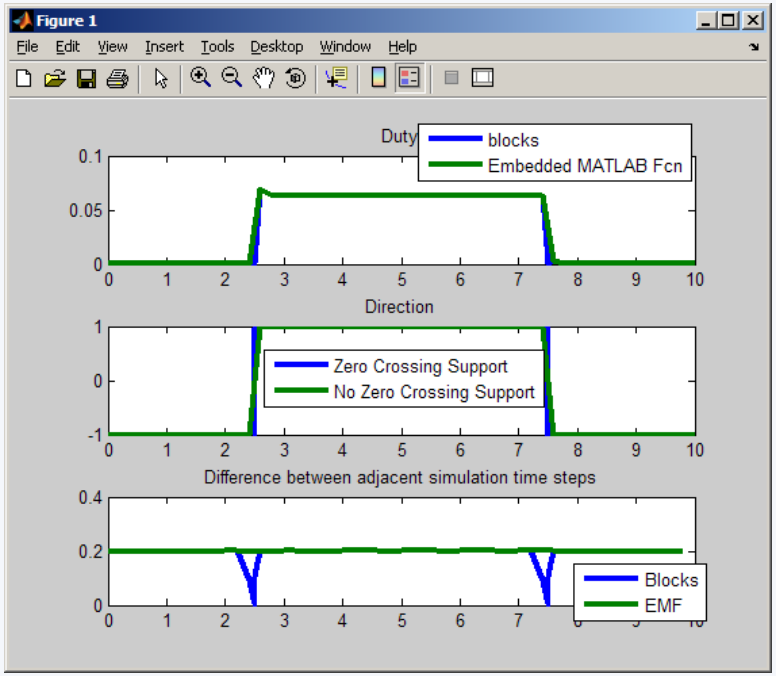



```
>> etc_logic_eml
```

© 2006 The MathWorks, Inc.

Simulating the Models and Comparing the Results

- Simulate the both versions of the logical systems with zero-crossing detection turned on.
- Compare the system outputs using MATLAB.
- Are there differences?
- Why?
- How can you make them behave the same?




```
>> edit etc_logic_analysis
>> etc_logic_analysis
```

© 2006 The MathWorks, Inc.

Summary

- Defining the system and identifying its components
- Modeling the system with equations
- Building a block diagram for the model
- Defining and configuring signal viewers
- Setting the configuration parameters
- Introducing zero crossing
- Variable-step versus fixed-step solvers
- Modifying the signal viewer parameters
- Simulating the model and analyzing the results
- Defining the maximum step size
- Modeling the system with Embedded MATLAB
- Simulating and comparing results

© 2006 The MathWorks, Inc.