

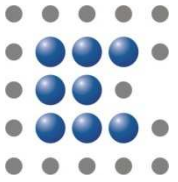
MIL / SIL Tutorial



<http://www.emmeskay.com/>
info@emmeskay.com

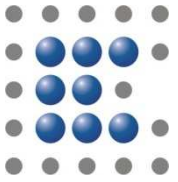
EMMESKAY, INC.
47119, Five Mile Road
Plymouth, MI 48170 USA
Phone: (734) 207 – 5564
FAX: (734) 207 – 5556

EMMESKAY SYSTEMS SOLUTIONS Pvt. Ltd.
20 Kannadasan Salai, T. Nagar,
Chennai 600 035 INDIA
Phone: +91 – 44 – 4207 0231
FAX: +91 – 44 – 4207 0243



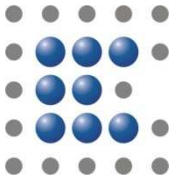
Overview

- Introduction
- Power Window Model
- MIL
- SIL
- HIL
- Review



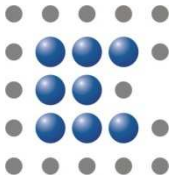
Introduction

- Goals of this tutorial
 - Describe differences between MIL, SIL & HIL
 - Give a walkthrough of how to actually go from MIL to SIL using Real-Time Workshop
- Things not covered here
 - Details of how to create and run HIL (too many real-time simulator vendors to demonstrate while remaining tool-neutral)



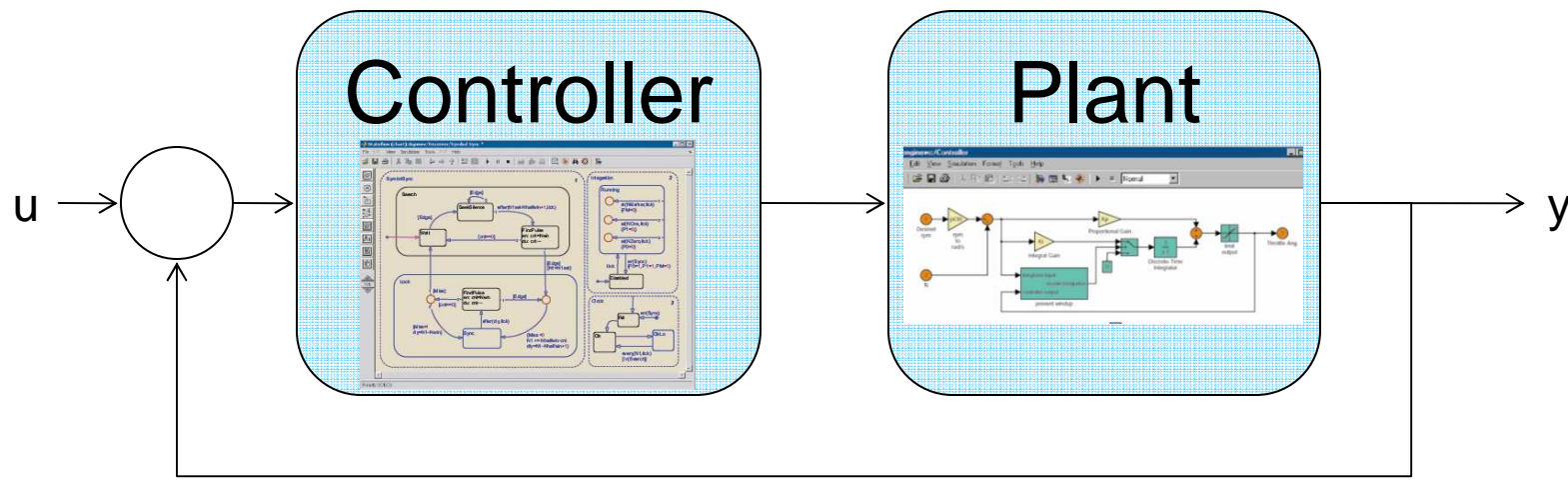
Introduction

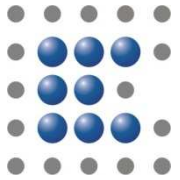
- Different stages of the development process (V-diagram) require different kinds of models
 - *MIL = Model-in-the-Loop*
 - *SIL = Software-in-the-Loop*
 - *HIL = Hardware-in-the-Loop*
- What are the differences in these models?
- How do we create them?



Introduction: MIL

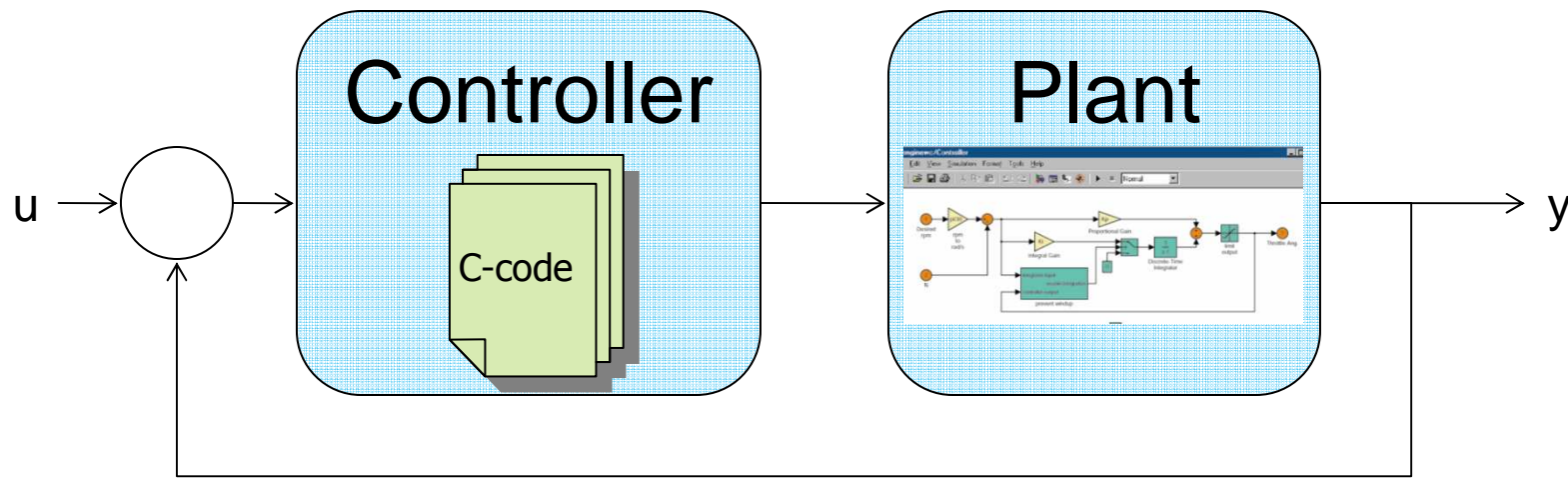
- Model exists entirely in native simulation tool (e.g., Simulink / Stateflow)
- Good for control algorithm development





Introduction: SIL

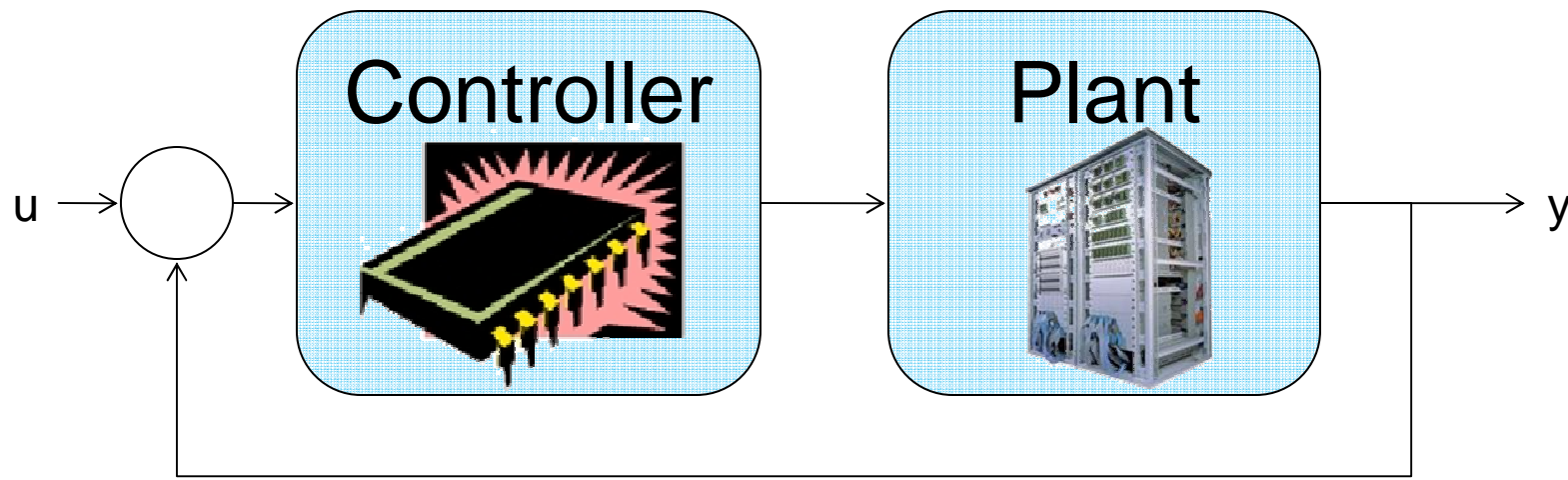
- Part of the model exists in native simulation tool (e.g., Simulink / Stateflow), and part as executable C-code (e.g., S-function)
- Good for testing controller implementations in C-code

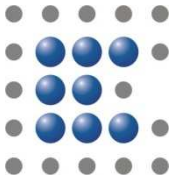




Introduction: HIL

- Part of the model runs in a real-time simulator, and part *may* exist as physical hardware (e.g., ECU)
- Good for testing interactions with hardware





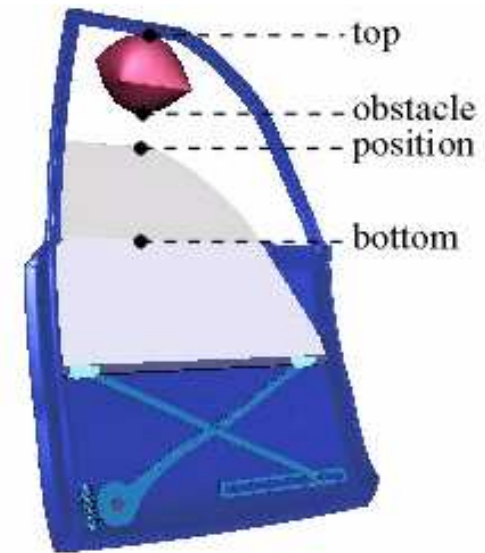
Overview

- Introduction
- **Power Window Model**
- MIL
- SIL
- HIL
- Review



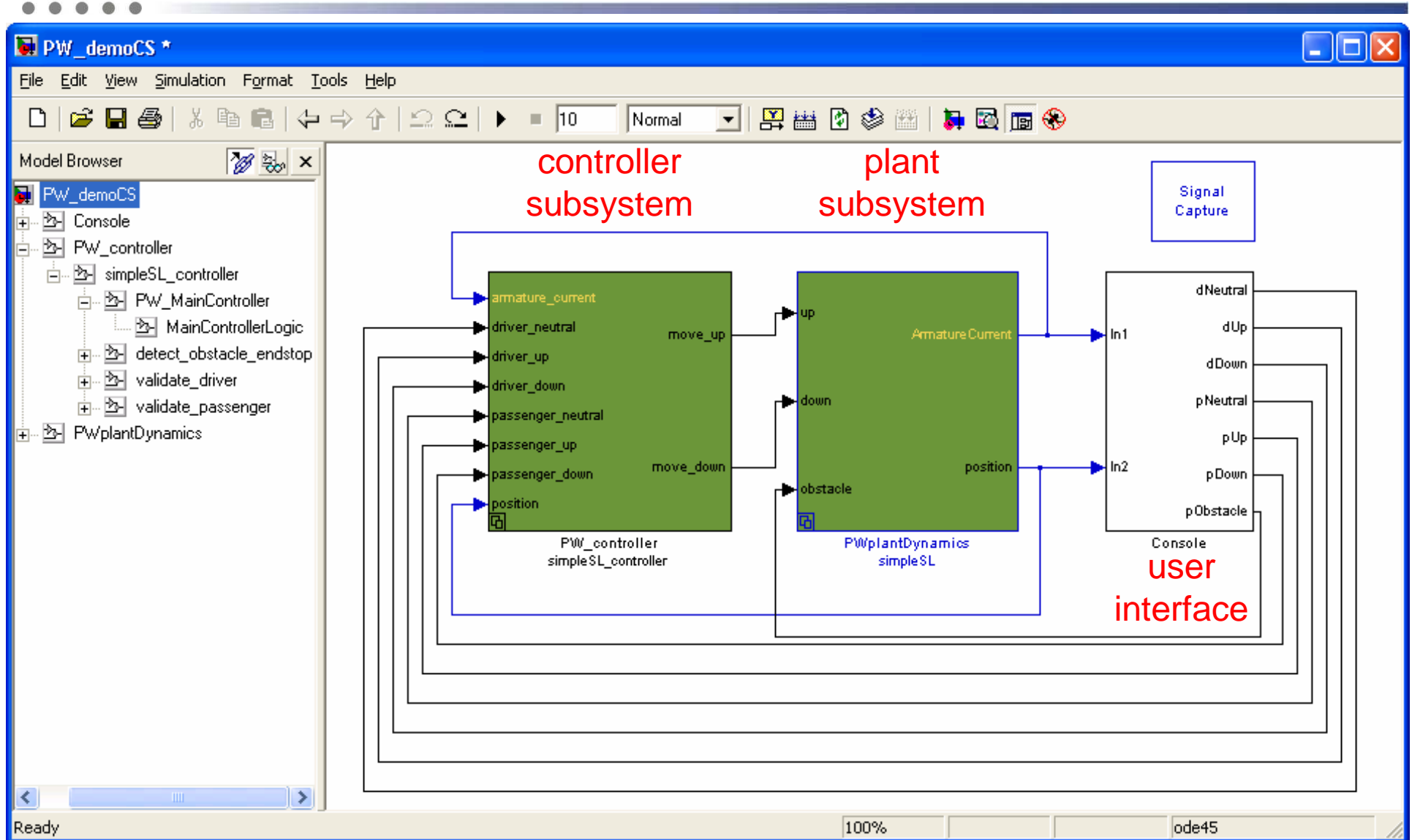
Power Window Model

- Simulink includes a demo model of a power window (PW) and its controller (see Simulink / Automotive Applications section of the Demo tab in Help system)
- We will use this model to demonstrate how to go from MIL to SIL



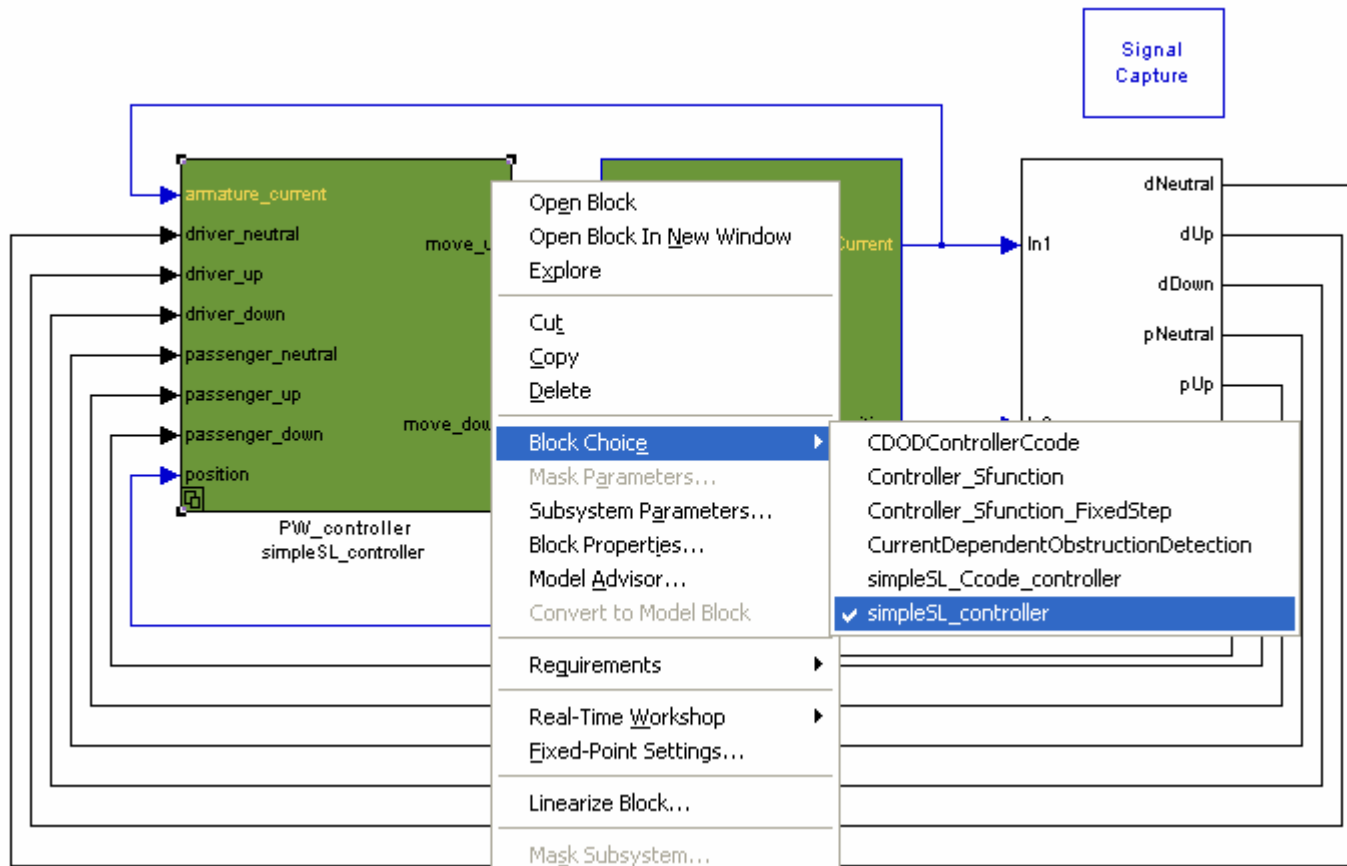


Power Window Model



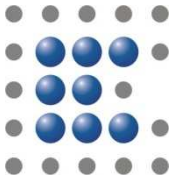


PW Model: Configurable Subsystems

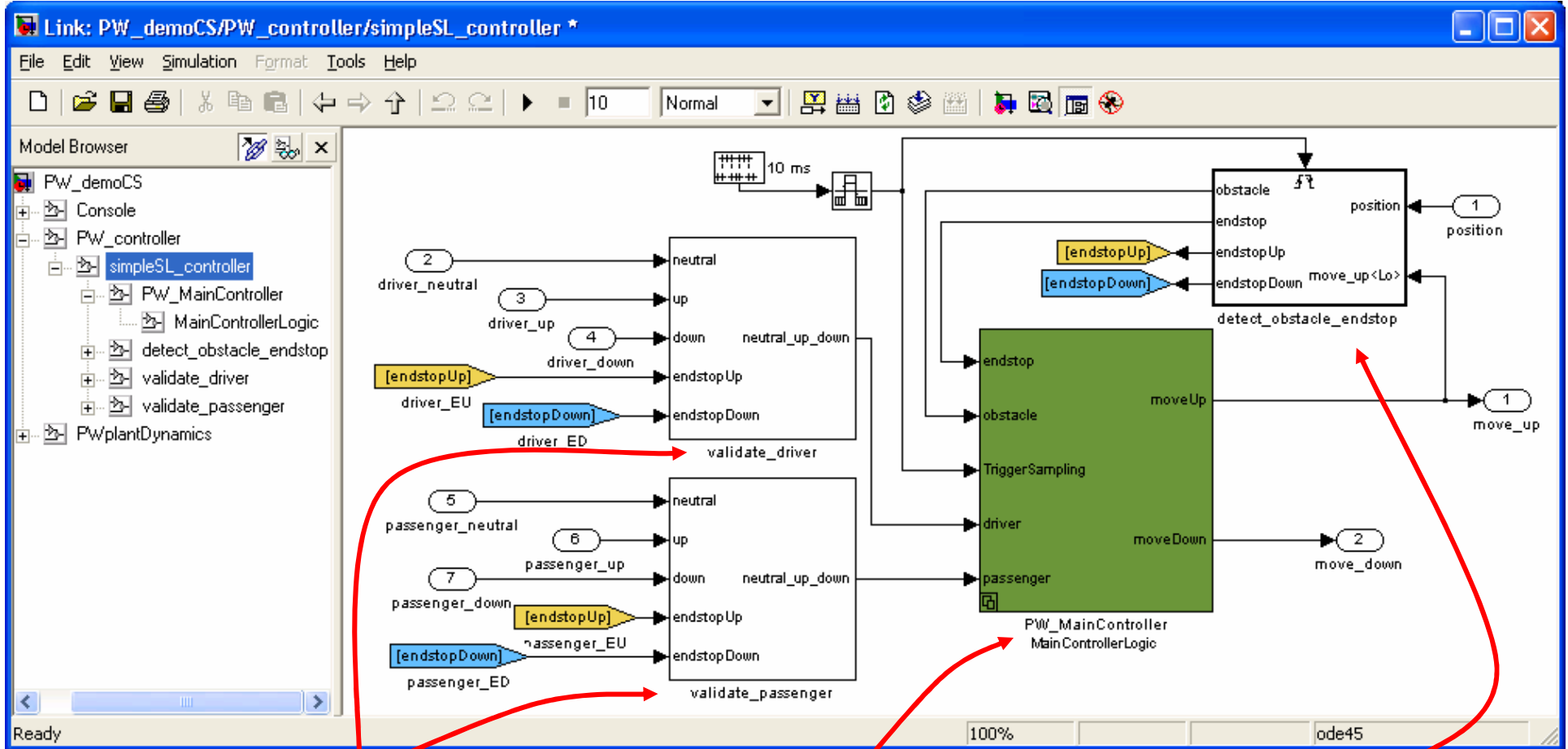


PW model uses configurable subsystems (linked libraries)

right-click on a subsystem and select “Block Choice” to see other available subsystem choices



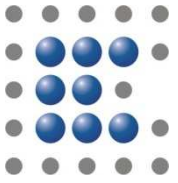
PW Model: Controller Subsystem



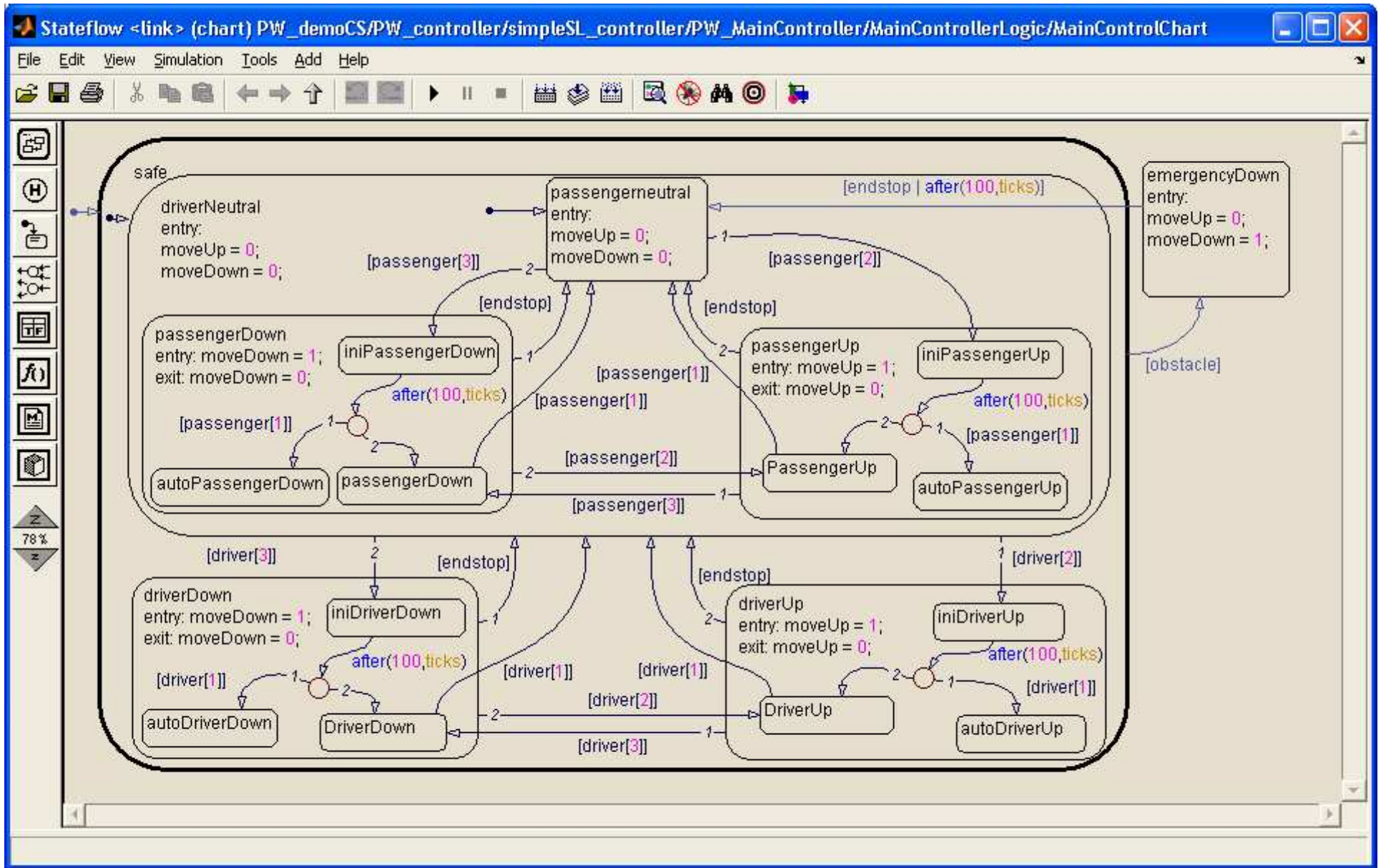
preprocess command signals from driver and passenger

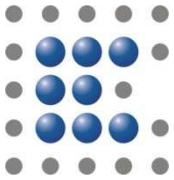
main logic (Stateflow chart)

triggered subsystem periodically checks if there's an obstacle

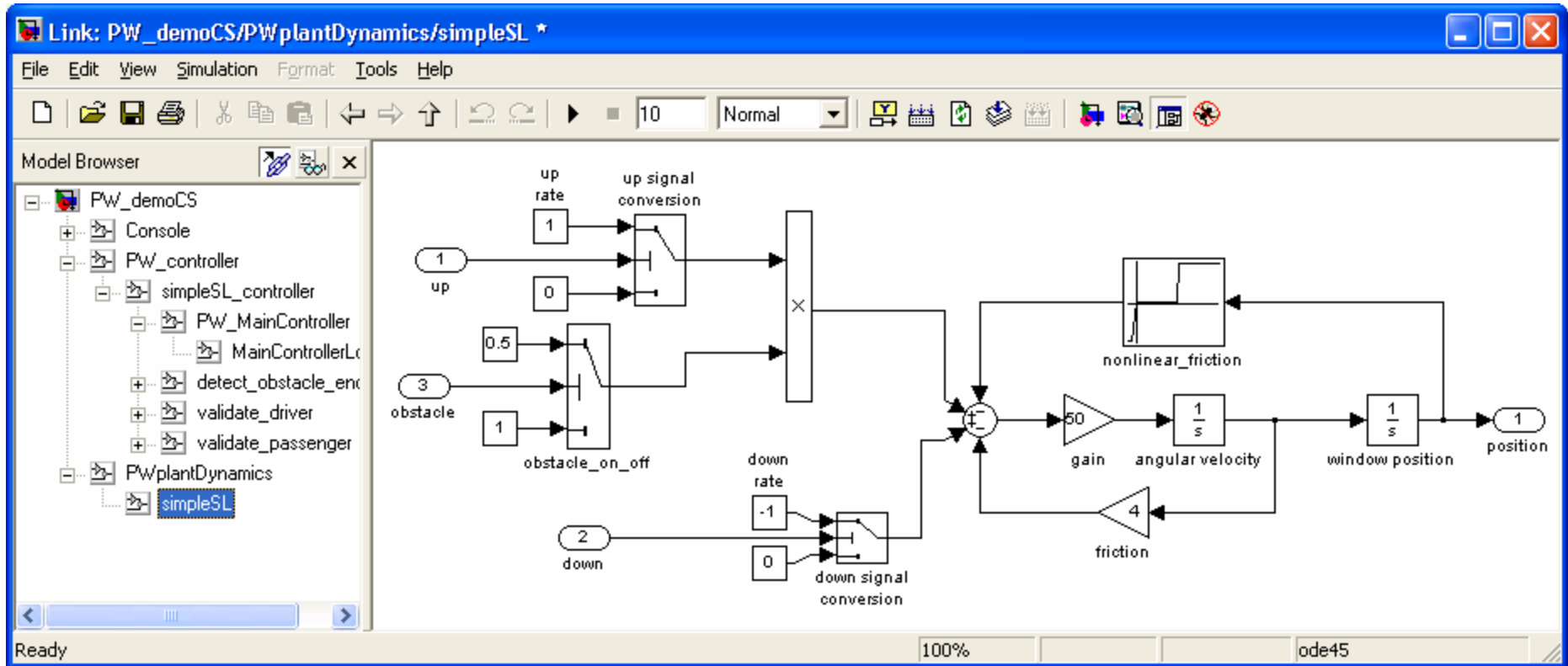


PW Model: Stateflow Chart

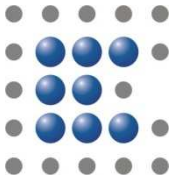




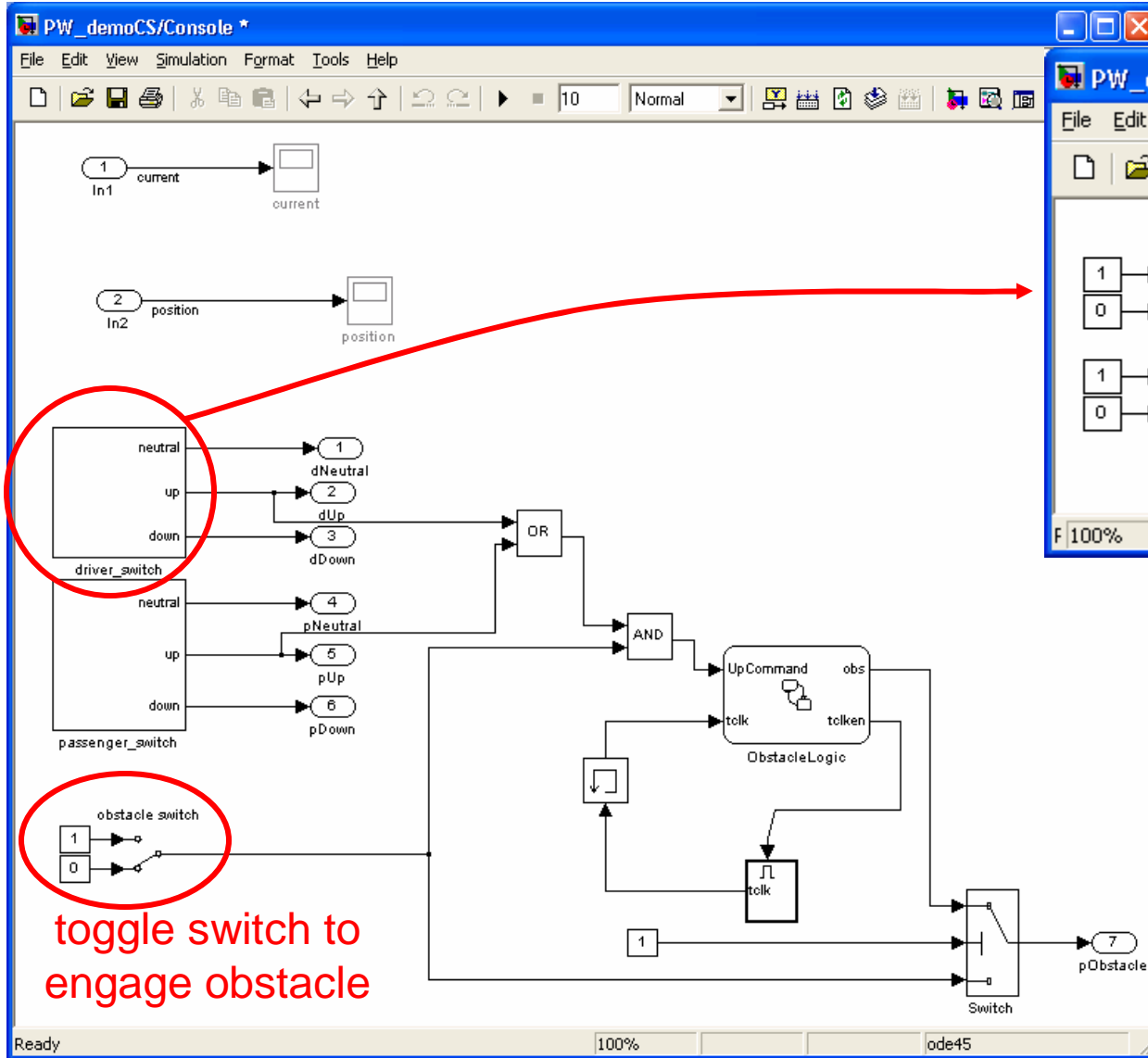
PW Model: Plant Subsystem



simple plant model integrates
acceleration and velocity to
predict window position

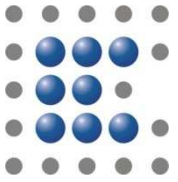


PW Model: Console



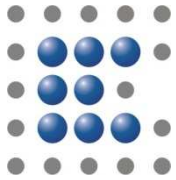
toggle switch to engage obstacle

toggle switches to signal up and down commands from driver / passenger



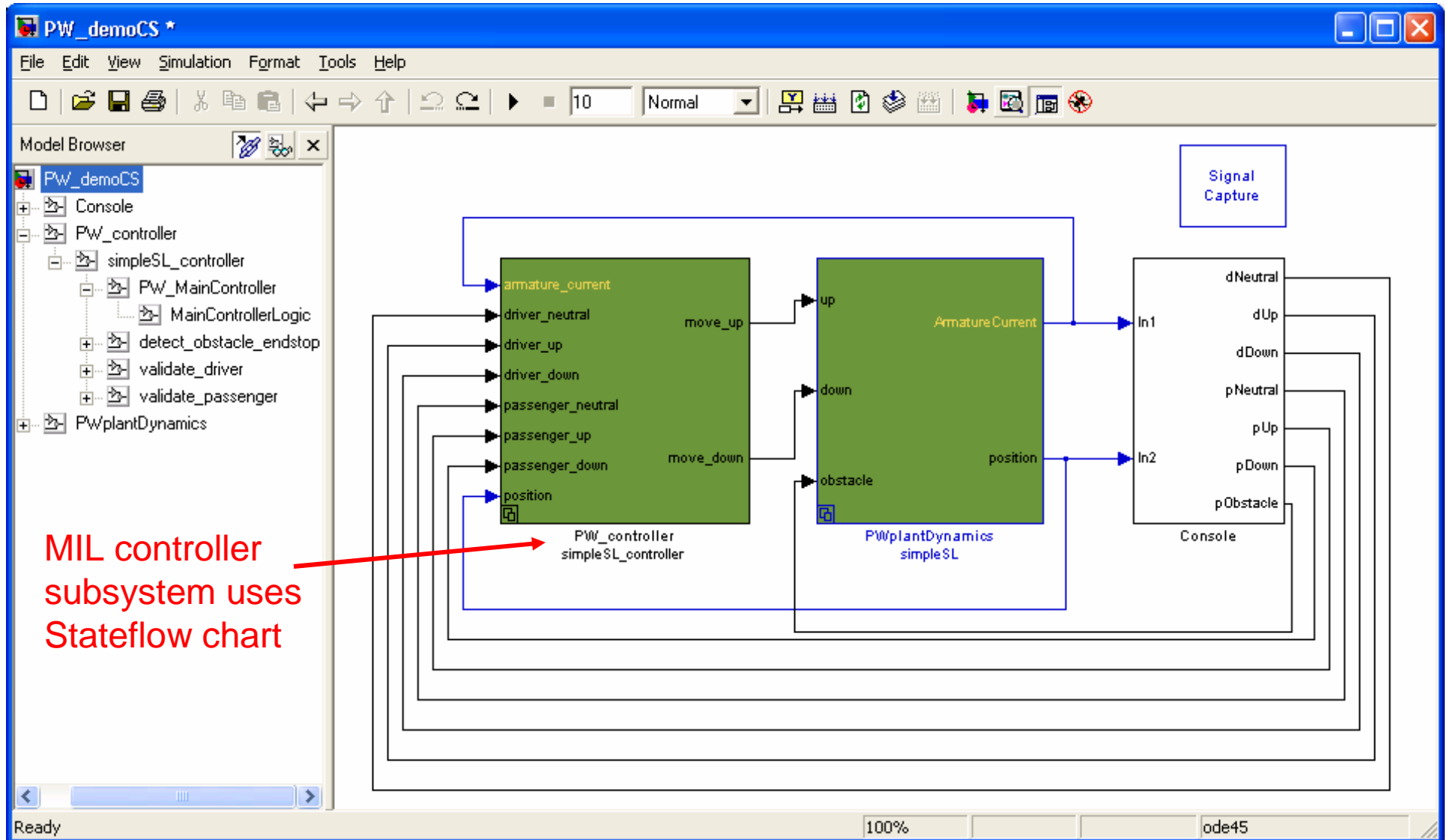
Overview

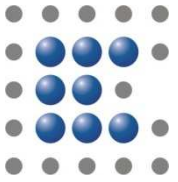
- Introduction
- Power Window Model
- **MIL**
- SIL
- HIL
- Review



Model-in-the-Loop (MIL)

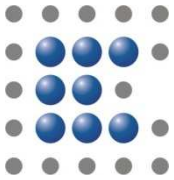
Run the existing model in Simulink for MIL



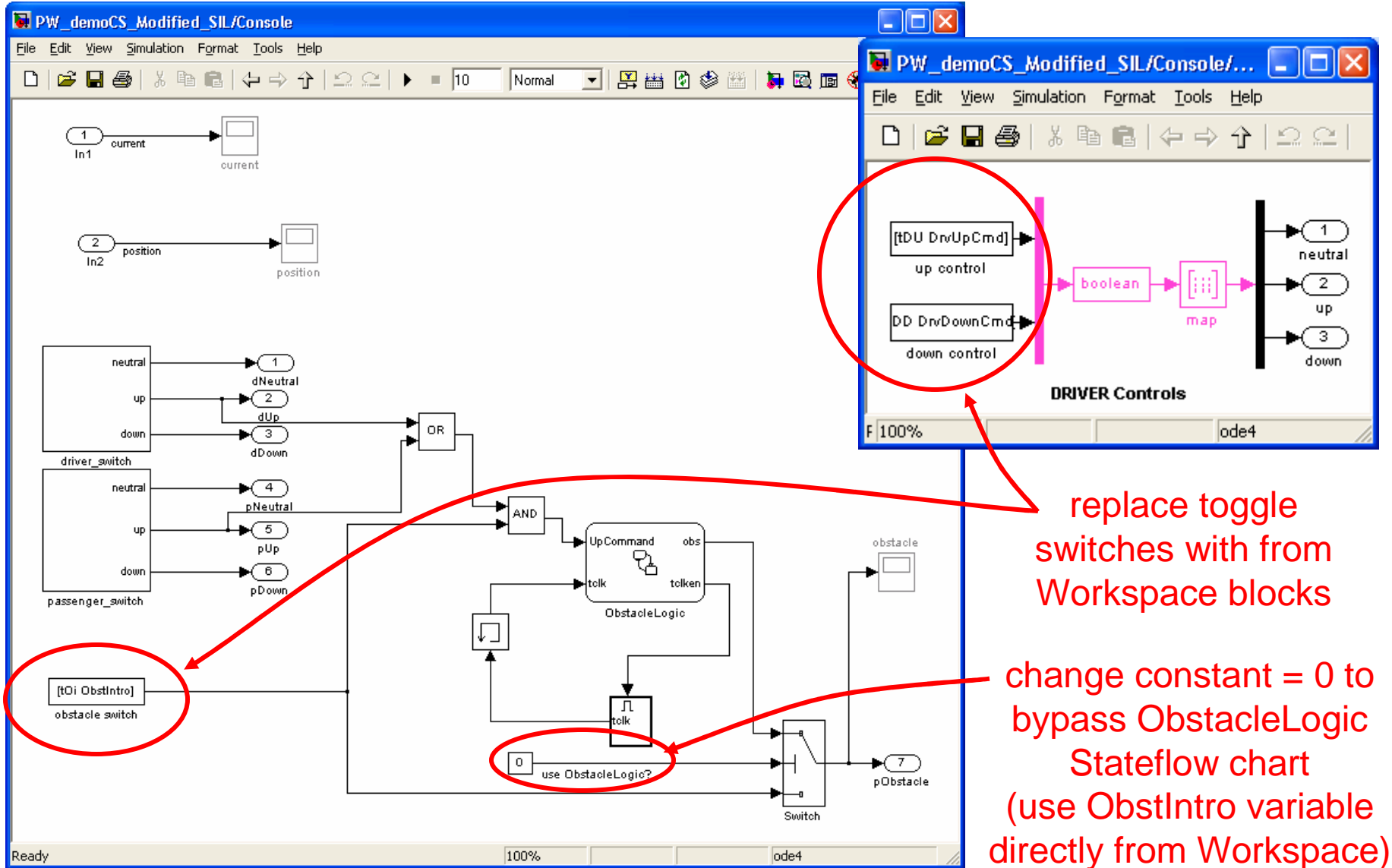


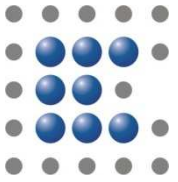
Model-in-the-Loop (MIL)

- Controlling the model
 - Original model controls inputs via 5 toggle switches
 - obstacle
 - driver up
 - driver down
 - passenger up
 - passenger down
 - Not very convenient for automated testing
- Enhancements
 - Replace 5 toggle switches with “from Workspace” blocks so you can run automated model testing through an m-script



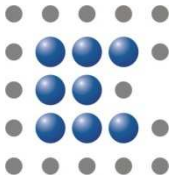
Model-in-the-Loop (MIL)





Model-in-the-Loop (MIL)

- Sample script will:
 - open specific test model
 - select specific subsystems from library
 - set specific input conditions (test vector)
 - execute simulation
 - compare results to “known good”



Model-in-the-Loop (MIL)

```
model = 2; % model configuration
input = 2; % input conditions
output = 2; % expected results
plotit = 1; % show plot of results?
```

```
% set model configuration
```

```
switch model
```

```
case 1
```

```
    % original Mathworks model
```

```
    modelname = 'PW_demoCS.mdl';
```

```
    open(modelname)
```

```
    set_param('PW_demoCS/PW_controller','BlockChoice','simpleSL_controller');
```

```
    set_param('PW_demoCS/PW_controller/simpleSL_controller/PW_MainController','BlockChoice','MainControllerLogic');
```

```
case 2
```

```
    % model modified to shortcut console Stateflow diagram
```

```
    modelname = 'PW_demoCS_Modified.mdl';
```

```
    open(modelname)
```

```
    set_param('PW_demoCS_Modified/PW_controller','BlockChoice','simpleSL_controller');
```

```
    set_param('PW_demoCS_Modified/PW_controller/simpleSL_controller/PW_MainController','BlockChoice','MainControllerLogic');
```

```
case 3
```

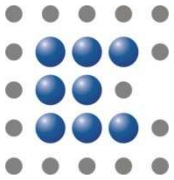
```
    % model using S-function representation of controller subsystem
```

```
    modelname = 'PW_demoCS_Modified_SIL.mdl';
```

```
    open(modelname)
```

```
    set_param('PW_demoCS_Modified_SIL/PW_controller','BlockChoice','Controller_Sfunction');
```

```
end
```



Model-in-the-Loop (MIL)

% set input conditions

```
switch input
case 1
    % passenger raises at 2s, driver lowers at 9s
    tDU = [0;10];
    DrvUpCmd = [0;0];

    tDD = [0;6;6;9;9;10];
    DrvDownCmd = [0;0;1;1;0;0];

    tPU = [0;2;2;7;7;10];
    PassUpCmd = [0;0;1;1;0;0];

    tPD = [0;10];
    PassDownCmd = [0;0];

    tOi = [0;10];
    ObstIntro = [0;0];
case 2
    % passenger raises at 2s, obstruction at 3s
    tDU = [0;10];
    DrvUpCmd = [0;0];

    tDD = [0;10];
    DrvDownCmd = [0;0];

    tPU = [0;2;2;2.1;2.1;10];
    PassUpCmd = [0;0;1;1;0;0];

    tPD = [0;10];
    PassDownCmd = [0;0];

    tOi = [0;3;3;10];
    ObstIntro = [0;0;1;1];
end
```

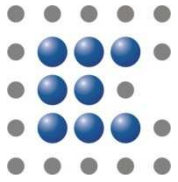
% run simulation

```
sim(modelname);
```

% check outputs for pass / fail

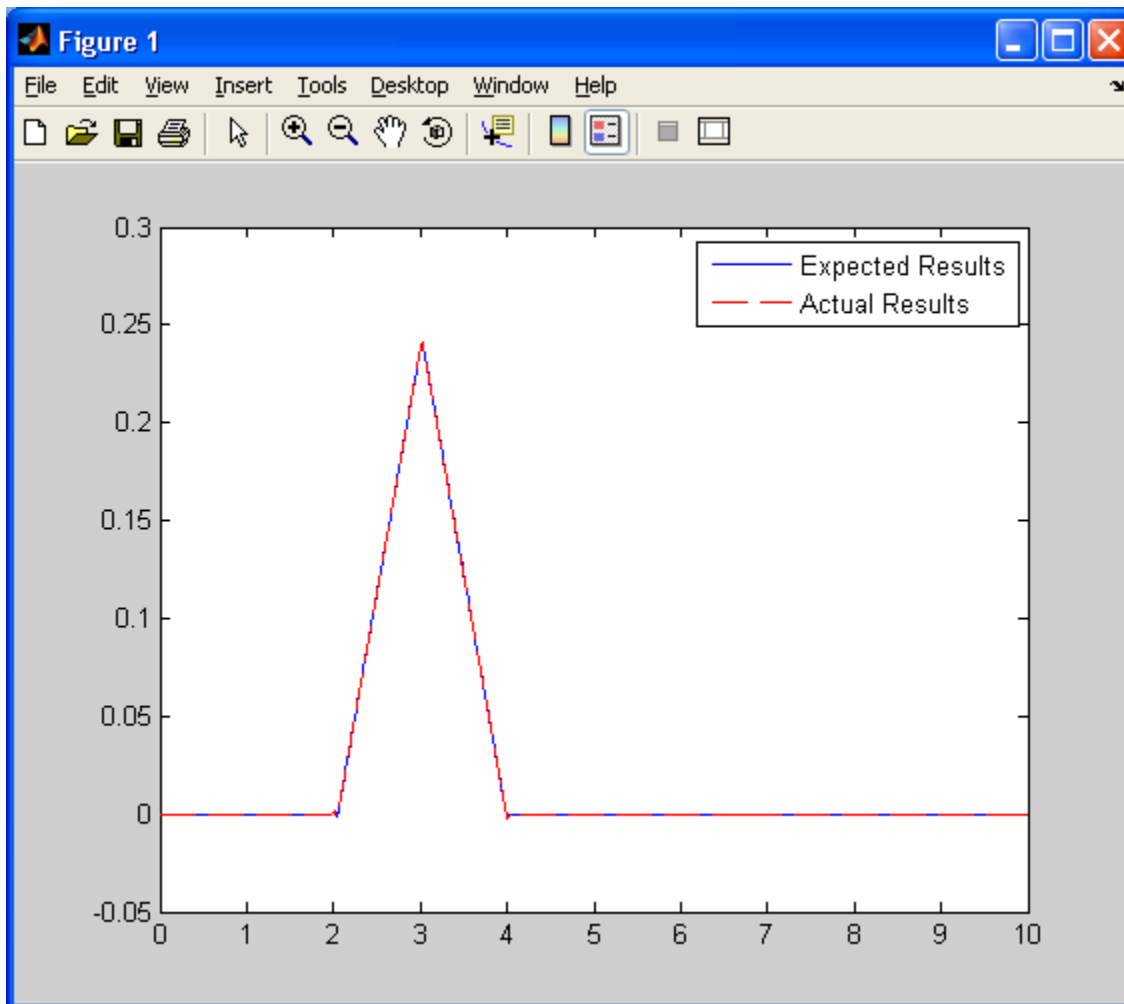
```
switch output
case 1
    % window raised at 2s up to 0.4m and lowers at 9s back to 0m
    expected = load('ExpectedResult1.mat');
    rms_error = minimum_rms( ...
        [position.time, position.signals.values], ...
        [expected.position.time, expected.position.signals.values]);
    if rms_error < 0.01, result = 1, else, result = 0, end
    if plotit
        plot(expected.position.time, expected.position.signals.values,'b', ...
            position.time, position.signals.values, 'r--')
        legend('Expected Results','Actual Results')
    end
case 2
    % window raised at 2s up to 0.25m and lowers at 3s back to 0m
    expected = load('ExpectedResult2.mat');
    rms_error = minimum_rms( ...
        [position.time, position.signals.values], ...
        [expected.position.time, expected.position.signals.values]);
    if rms_error < 0.01, result = 1, else, result = 0, end
    if plotit
        plot(expected.position.time, expected.position.signals.values,'b', ...
            position.time, position.signals.values, 'r--')
        legend('Expected Results','Actual Results')
    end
end
```

Emmeskay function for comparing time traces

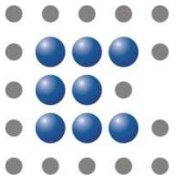


Model-in-the-Loop (MIL)

Running the script for test case shown above results in the following:



```
>> PW_openloop  
  
result =  
  
    1
```



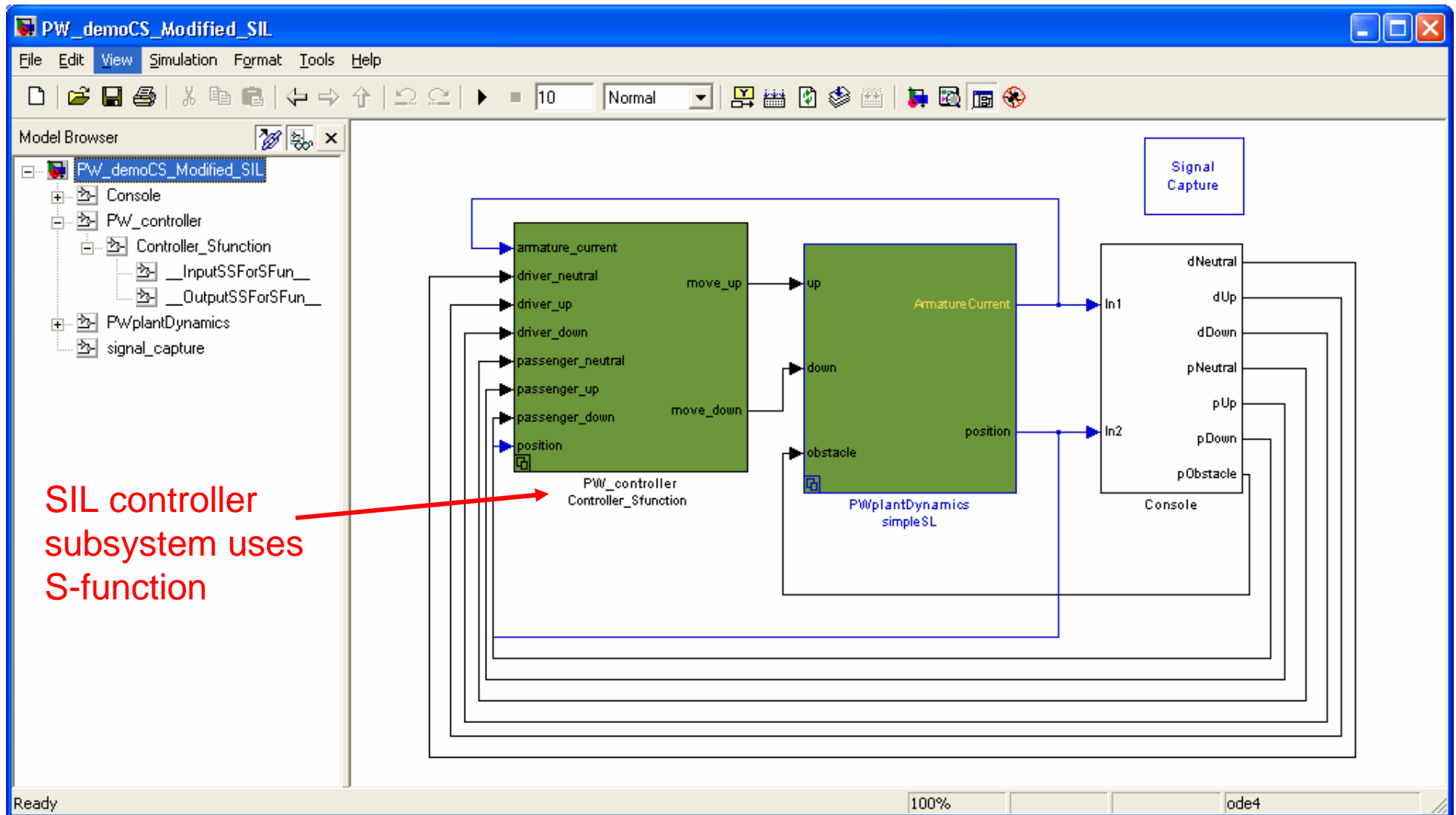
Overview

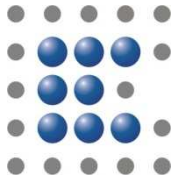
- Introduction
- Power Window Model
- MIL
- **SIL**
- HIL
- Review



Software-in-the-Loop (SIL)

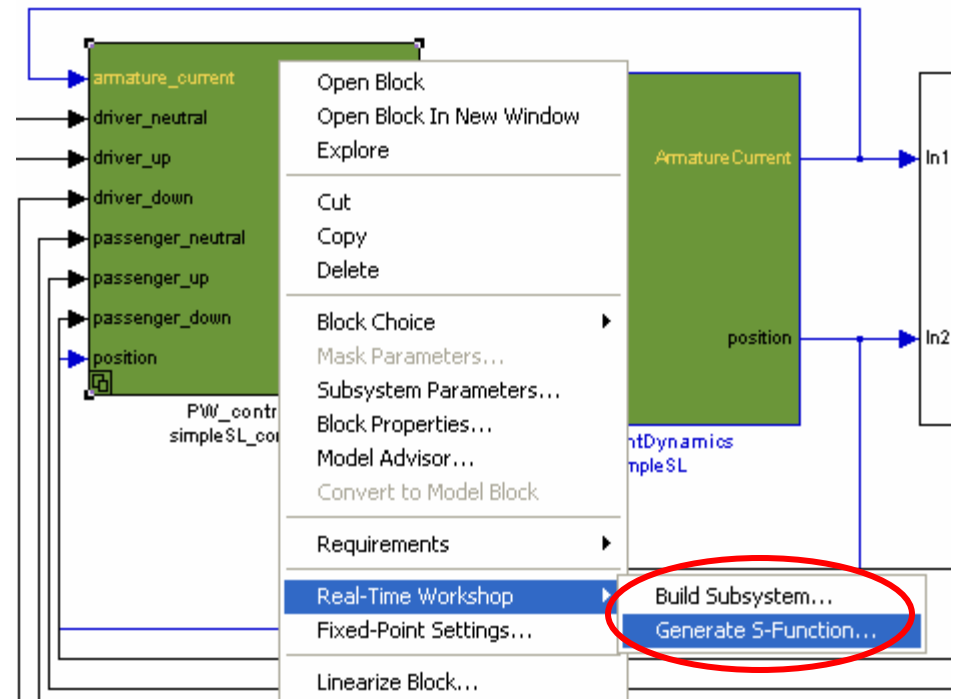
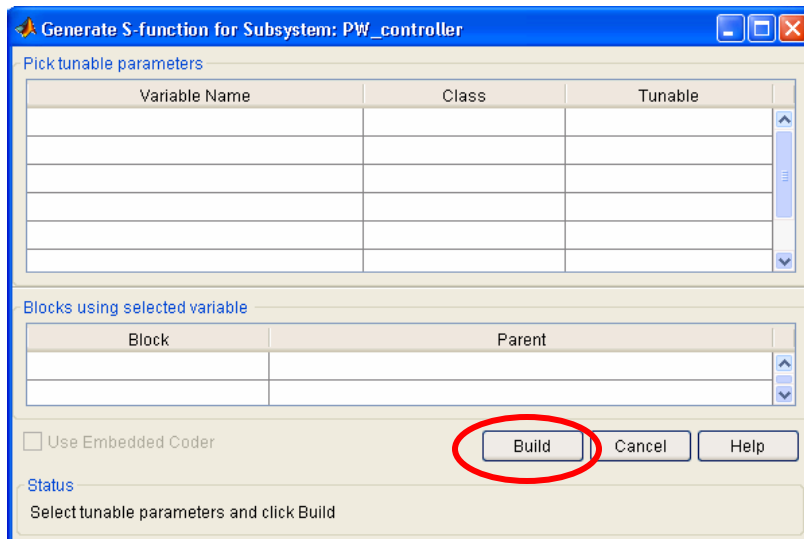
Convert controller to S-function for SIL

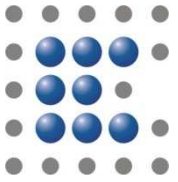




Software-in-the-Loop (SIL)

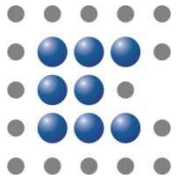
- How do you create the S-function for the controller subsystem?
 - Right-click on controller subsystem and select “Real-Time Workshop / Generate S-Function”
 - Click “Build”





Software-in-the-Loop (SIL)

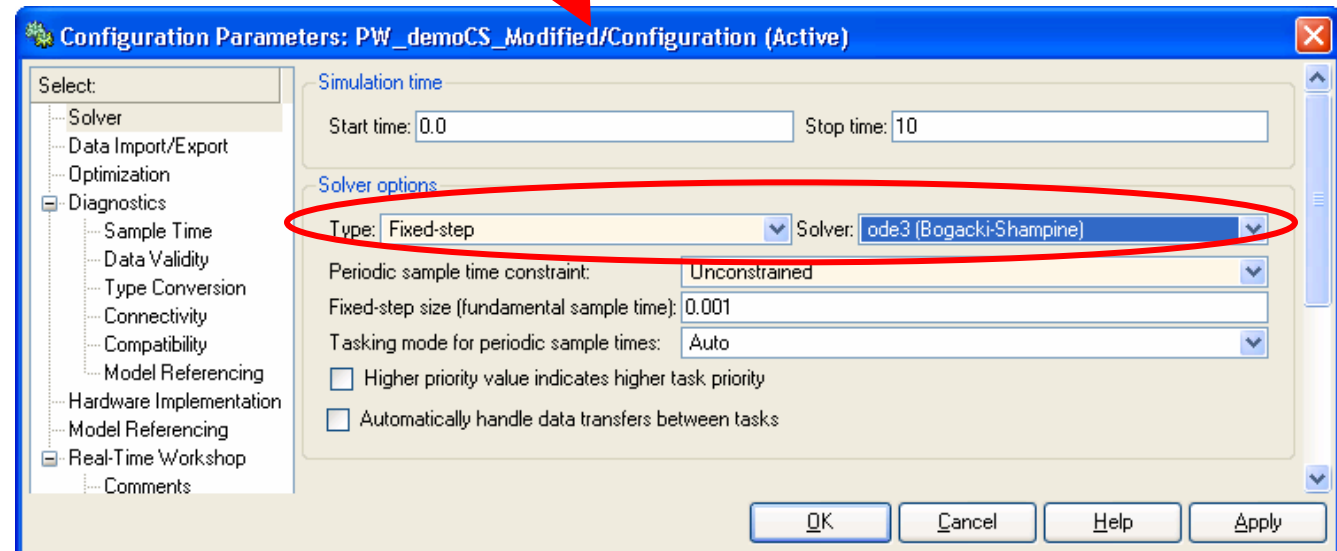
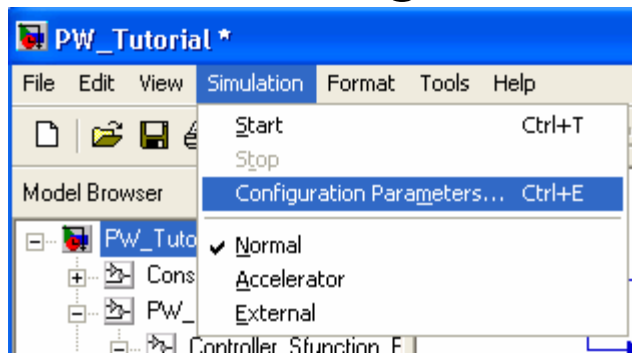
- Is that all there is to it?
 - No, not really
 - We want to eventually run this in a HIL system
 - HIL means you run the model in real-time
 - Therefore, must get this model to run with a *fixed-step solver*



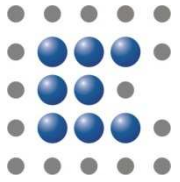
Software-in-the-Loop (SIL)

1. Create S-function

a. Change model to fixed-step solver

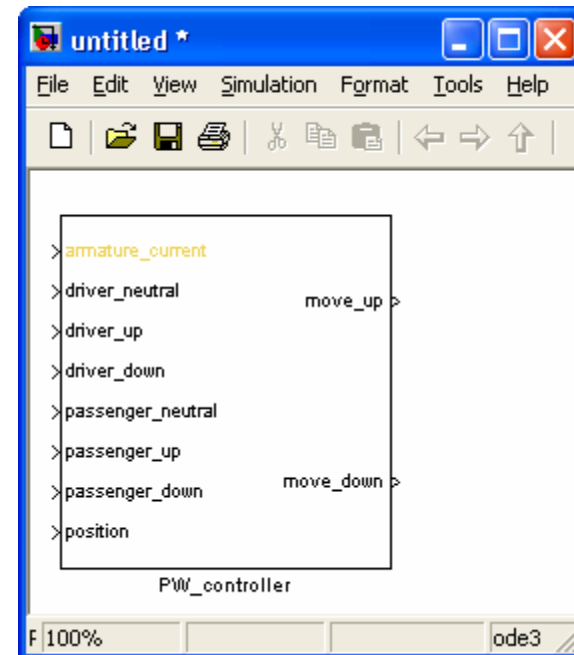


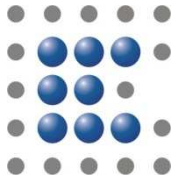
Note: The original model won't run with fixed-step solvers as is. See "Potential Gotcha #1" below.



Software-in-the-Loop (SIL)

1. Create S-function
 - a. Change model to fixed-step solver
 - b. Build S-function from controller subsystem
 - c. A new model will open in Simulink



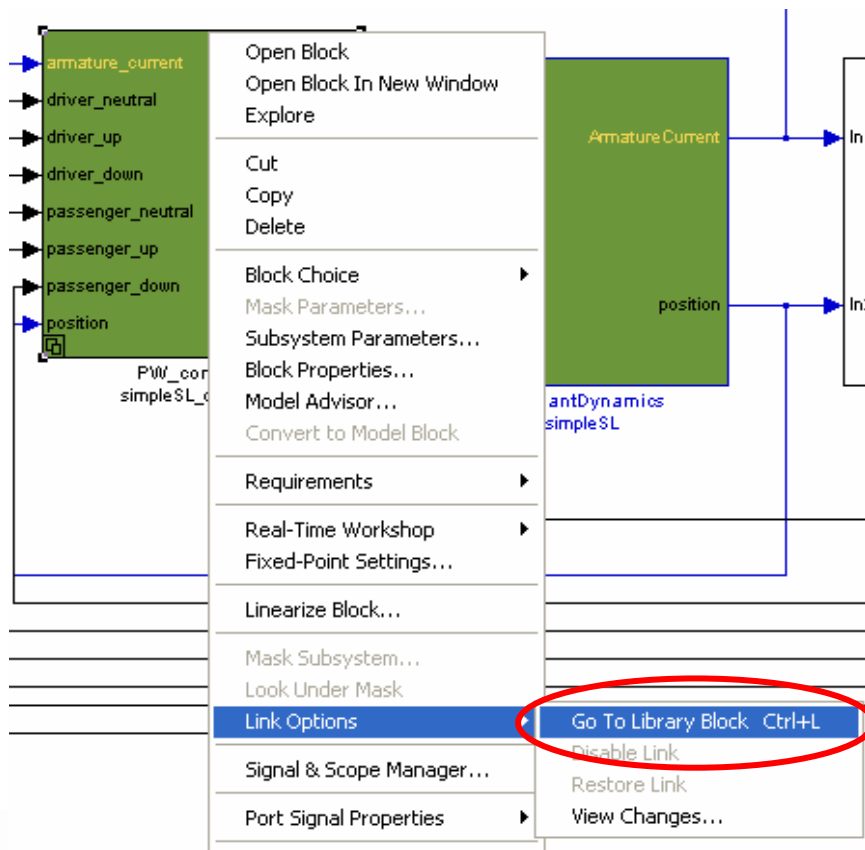


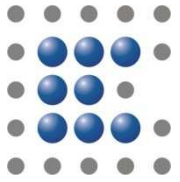
Software-in-the-Loop (SIL)

2. Bring S-function into library

a. Copy new model

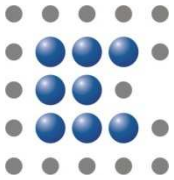
b. Right-click controller: “Go To Library Block”





Software-in-the-Loop (SIL)

2. Bring S-function into library
 - a. Copy new model
 - b. Right-click controller: “Go To Library Block”
 - c. In library, select “Edit / Unlock Library”
 - d. Paste new model into library
 - e. Rename new model



Software-in-the-Loop (SIL)

The screenshot shows a software development environment window titled "Library: Lib_PWcontrolCS". The window has a menu bar (File, Edit, View, Format, Help) and a toolbar. A "File" menu is open, showing options like "Can't Undo", "Cut", "Copy", "Paste", "Delete", "Select All", "Copy Model To Clipboard", "Find...", "Open Block", "Explore", "Block Choice", "Mask Parameters...", "Block Parameters...", "Block Properties...", "Convert to Model Block", "Create Subsystem", "Mask Subsystem...", "Look Under Mask", "Link Options", "Unlock Library" (highlighted with a red circle), "Refresh Model Blocks", and "Update Diagram".

The main workspace displays several block diagrams. A "PW_controller" block is highlighted in cyan. Below it, there are several "Controller_Sfunction" blocks, including "simpleSL_Code_controller", "Controller_Sfunction", and "Controller_Sfunction_FixedStep". Each block contains a list of parameters: armature_current, driver_neutral, driver_up, driver_down, passenger_neutral, passenger_up, passenger_down, and position. Some parameters have "move_up" or "move_down" labels next to them. A "Signal Capture" block is also visible in the top right.

At the bottom of the window, there is a status bar with the text "Allow the block library to be modified", "100%", and "Locked".

Unlock library

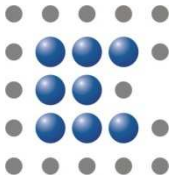
Paste in new S-function model

Double-click name to rename it

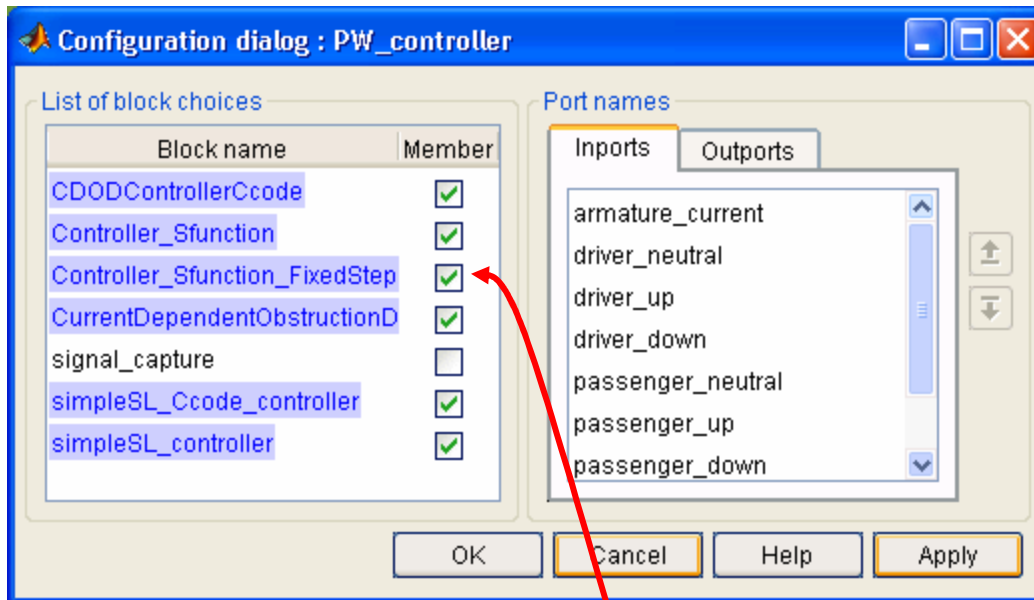


Software-in-the-Loop (SIL)

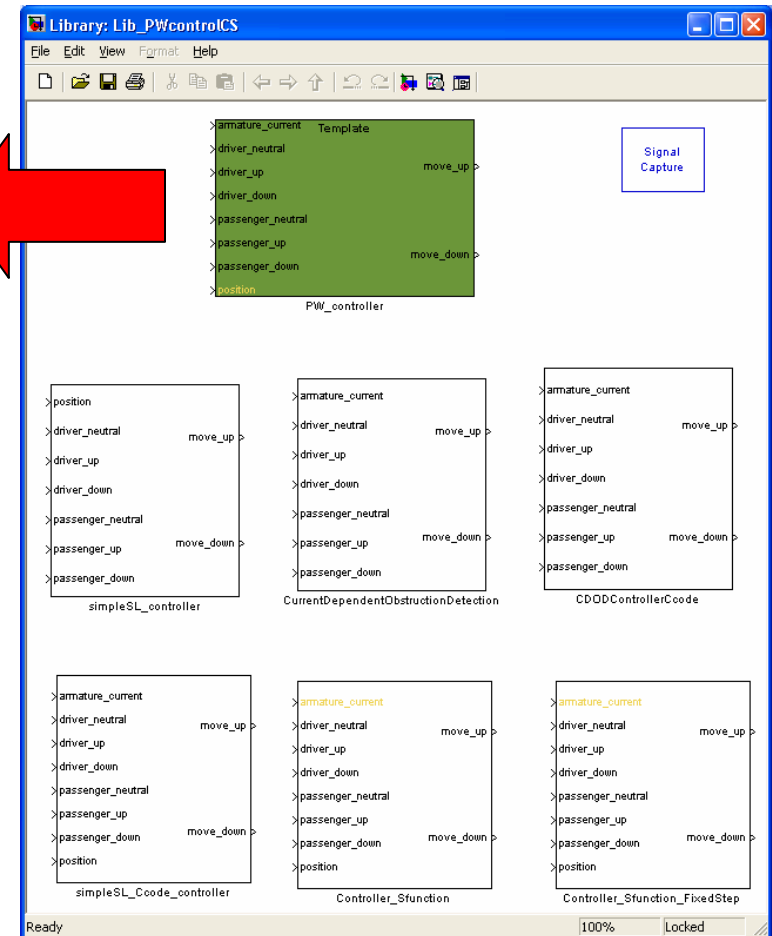
2. Bring S-function into library
 - a. Copy new model
 - b. Right-click controller: “Go To Library Block”
 - c. In library, select “Edit / Unlock Library”
 - d. Paste new model into library
 - e. Rename new model
 - f. Double-click PW_controller
 - g. Check new model to make it available as a block choice and click OK
 - h. Save library and close it

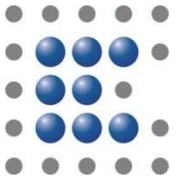


Software-in-the-Loop (SIL)



Check new model to include it as a library block choice

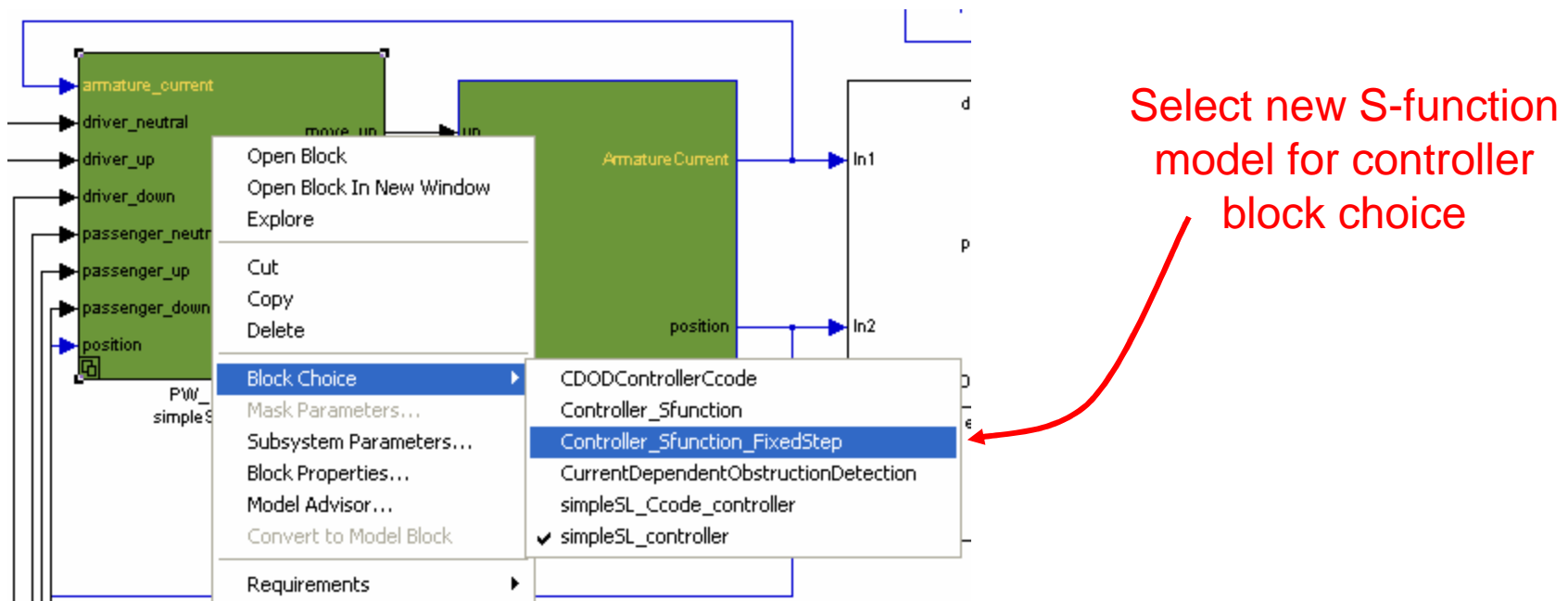


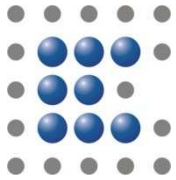


Software-in-the-Loop (SIL)

3. Run the SIL system

- a. Select the new S-function for the controller subsystem
- b. Simulate the model

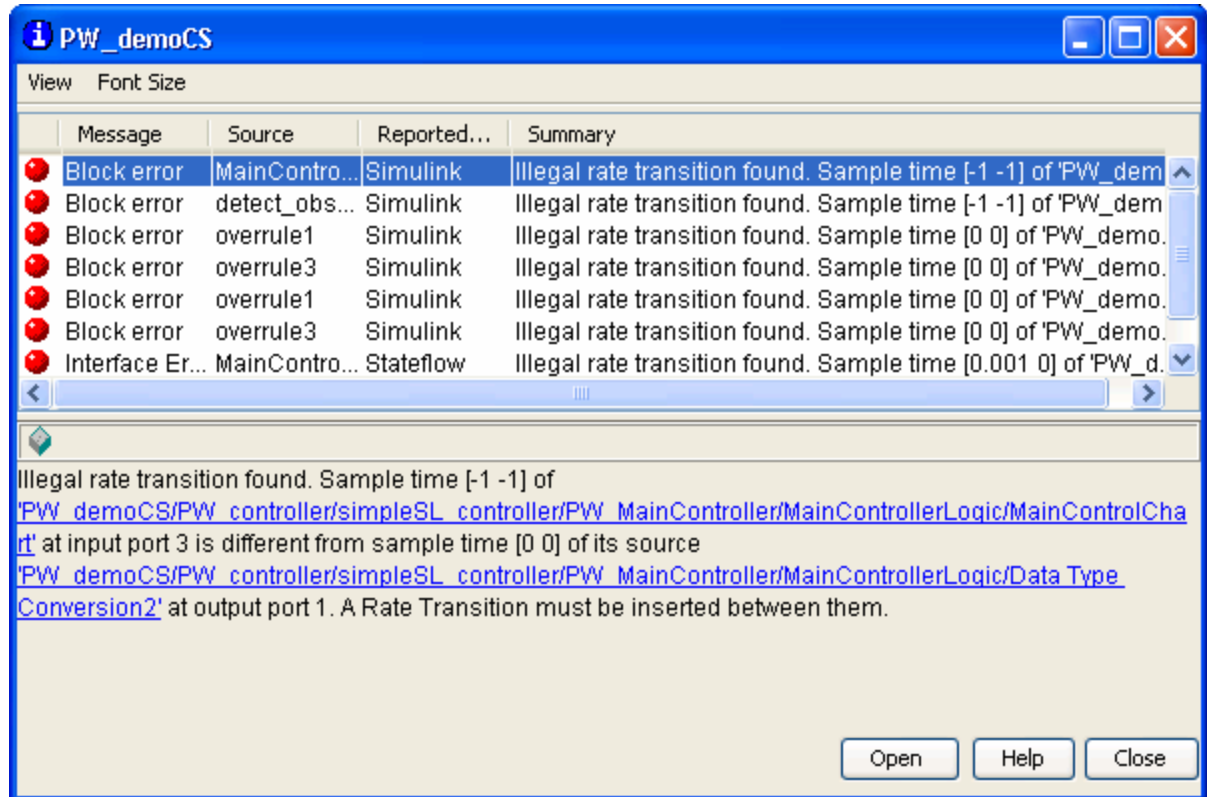


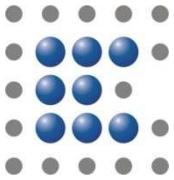


Software-in-the-Loop (SIL)

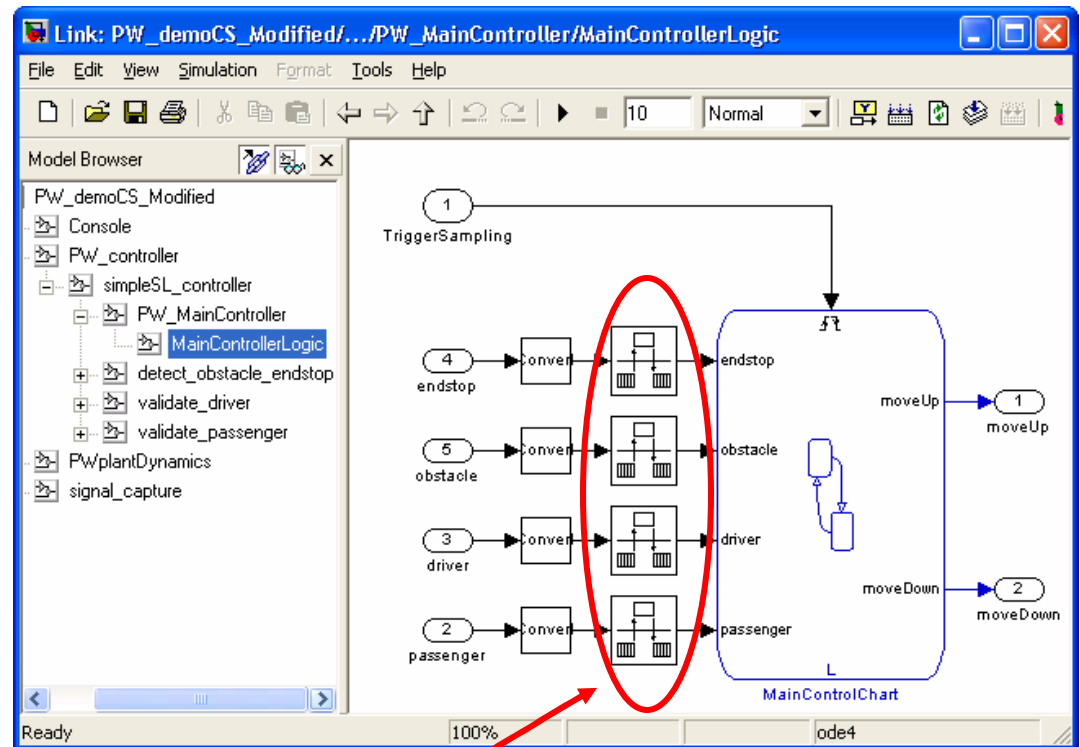
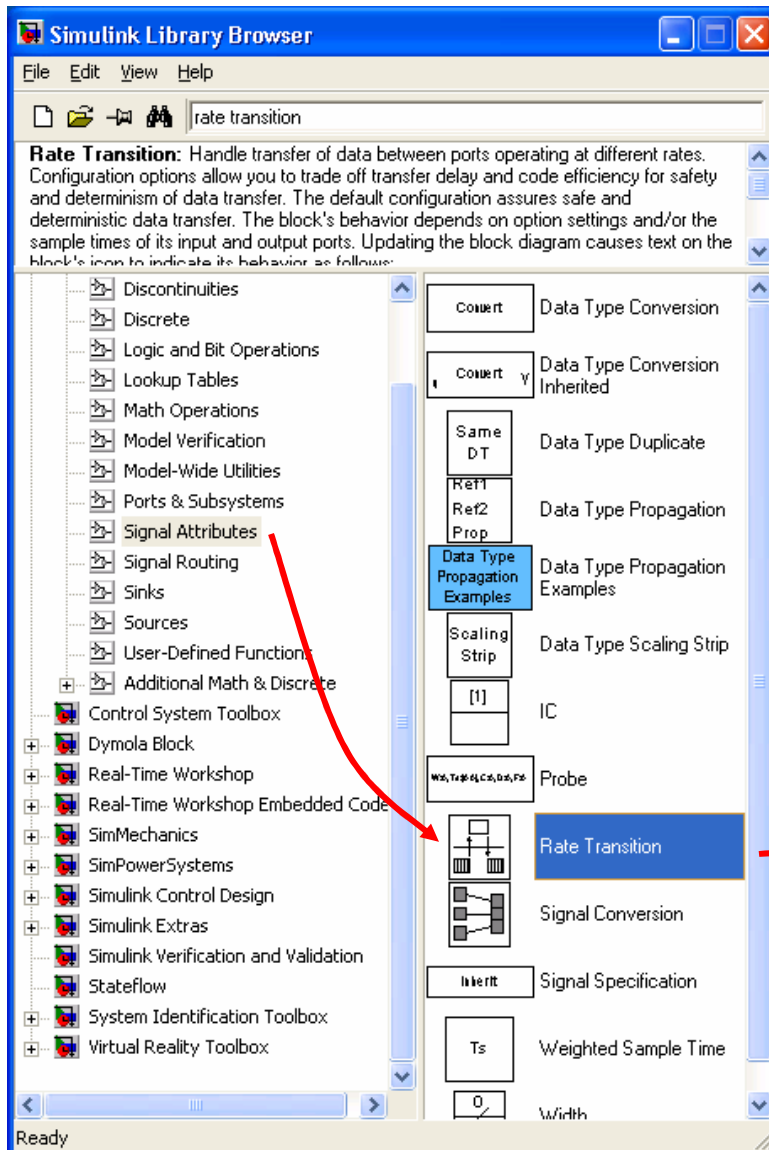
- Potential Gotcha #1
 - When trying to run MIL with a fixed step solver, you get this error:

The problem is that original model does not run properly with fixed step solvers because of the triggered subsystems. You must insert “Rate Transition” blocks to let the subsystems communicate in fixed step. Fixing the two triggered subsystems in the simpleSL_controller will allow the model to run in fixed-step.

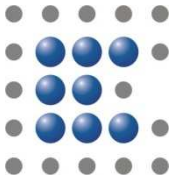




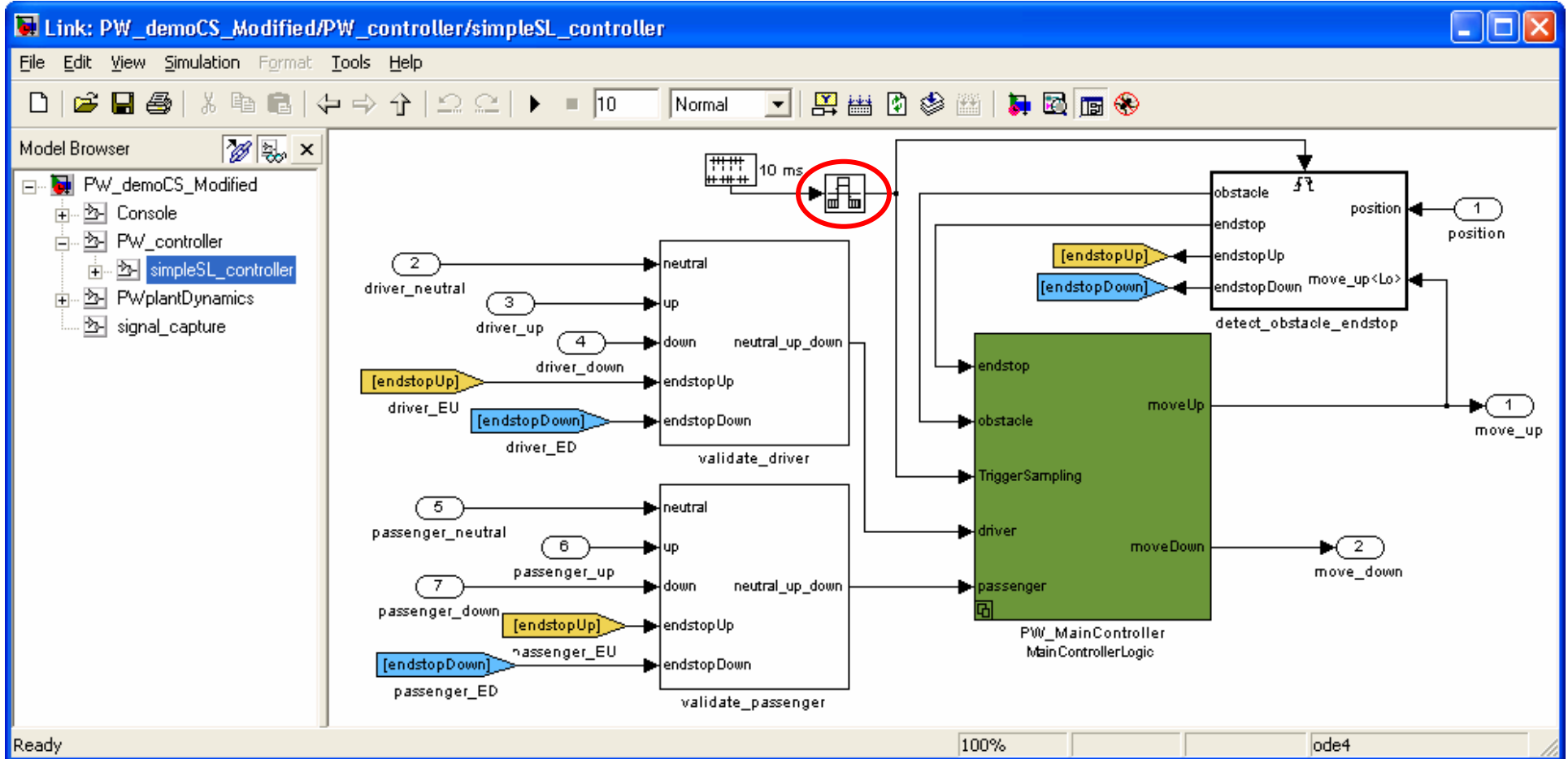
Software-in-the-Loop (SIL)



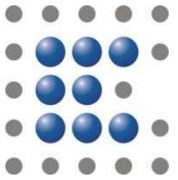
Triggered subsystem #1 is in the controller's MainControllerLogic subsystem. Source library is in Lib_MainControllerCS.



Software-in-the-Loop (SIL)



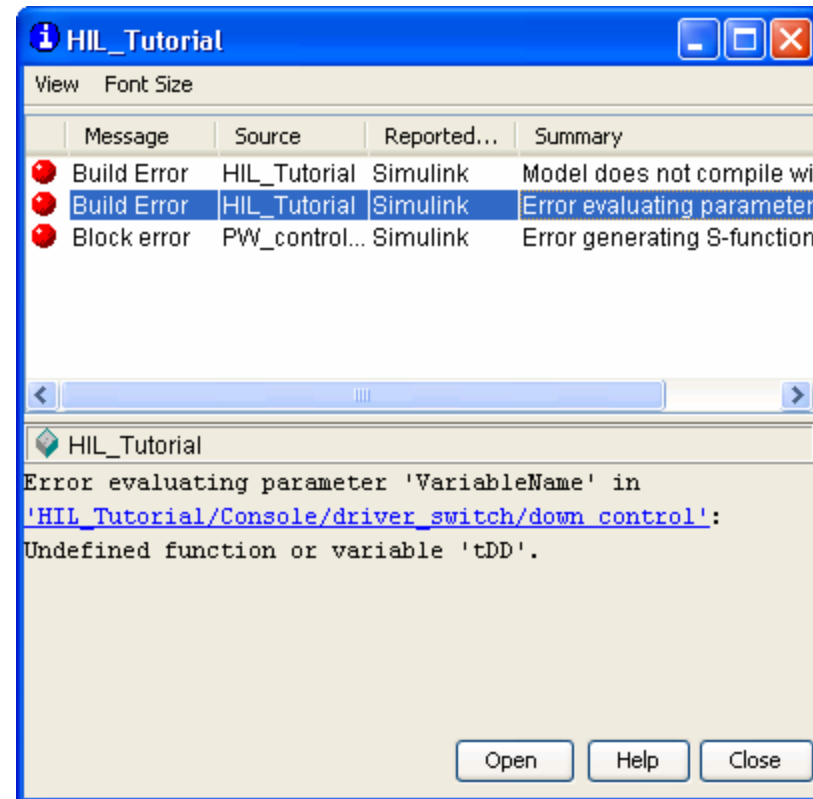
Triggered subsystem #2 is in the controller's top level.

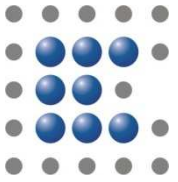


Software-in-the-Loop (SIL)

- Potential Gotcha #2:
 - When running “Generate S-function”, you get the following error:

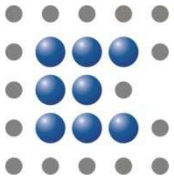
The problem is that the variables used in the “from Workspace” blocks were not declared first. Run the script (see MIL) to define values for these variables and try again.





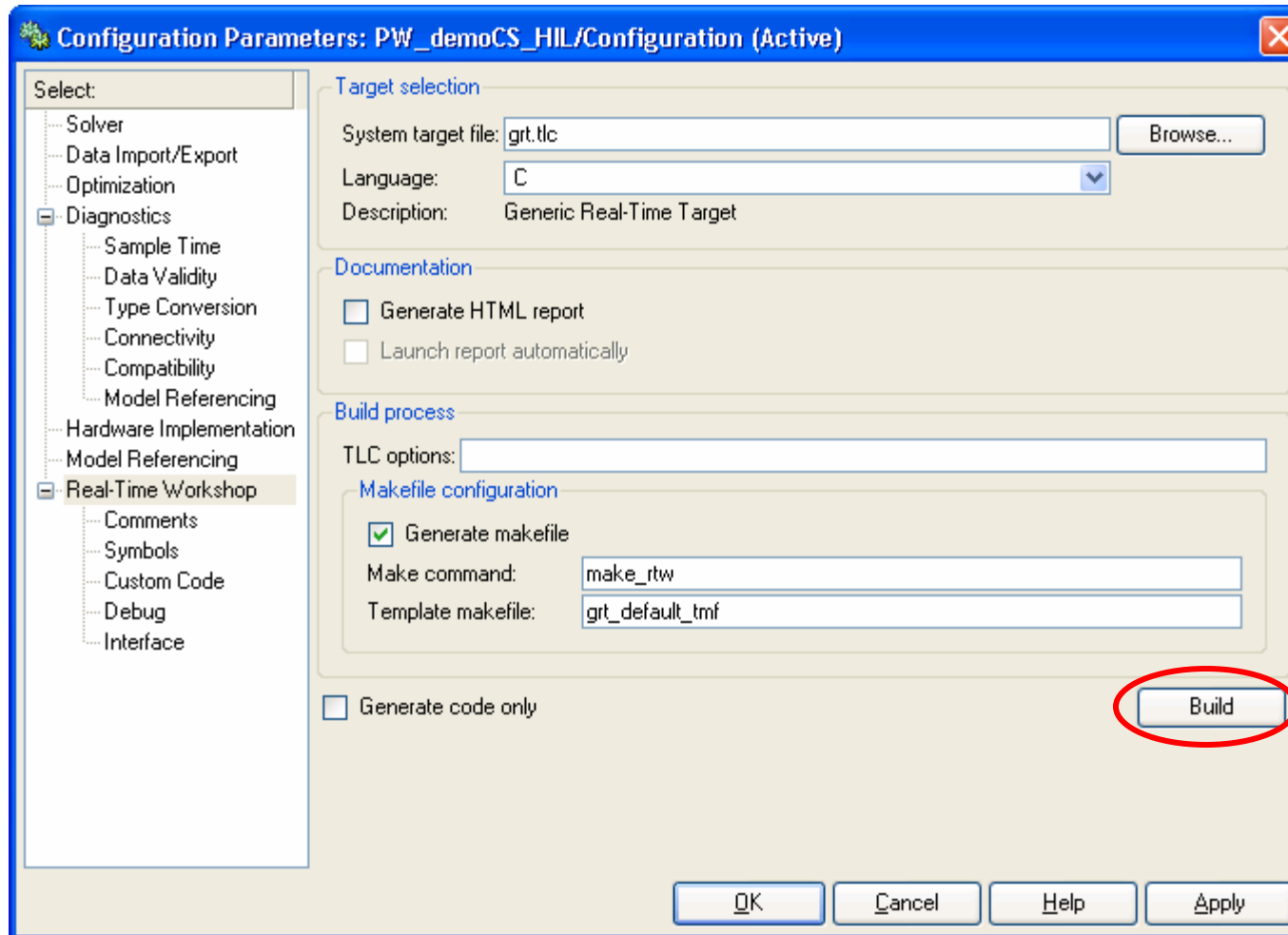
Overview

- Introduction
- Power Window Model
- MIL
- SIL
- **HIL**
- Review



Hardware-in-the-Loop (HIL)

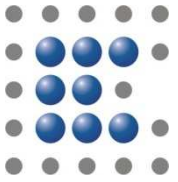
Build real-time system on simulator for HIL





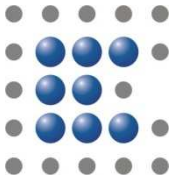
Hardware-in-the-Loop (HIL)

- Use Real-Time Workshop to build a version of the model that will run on real-time simulator.
- Use software tools to control the inputs to the real-time simulation.
- Automate testing, if desired.
- Contact us to know more about your hardware and software options.

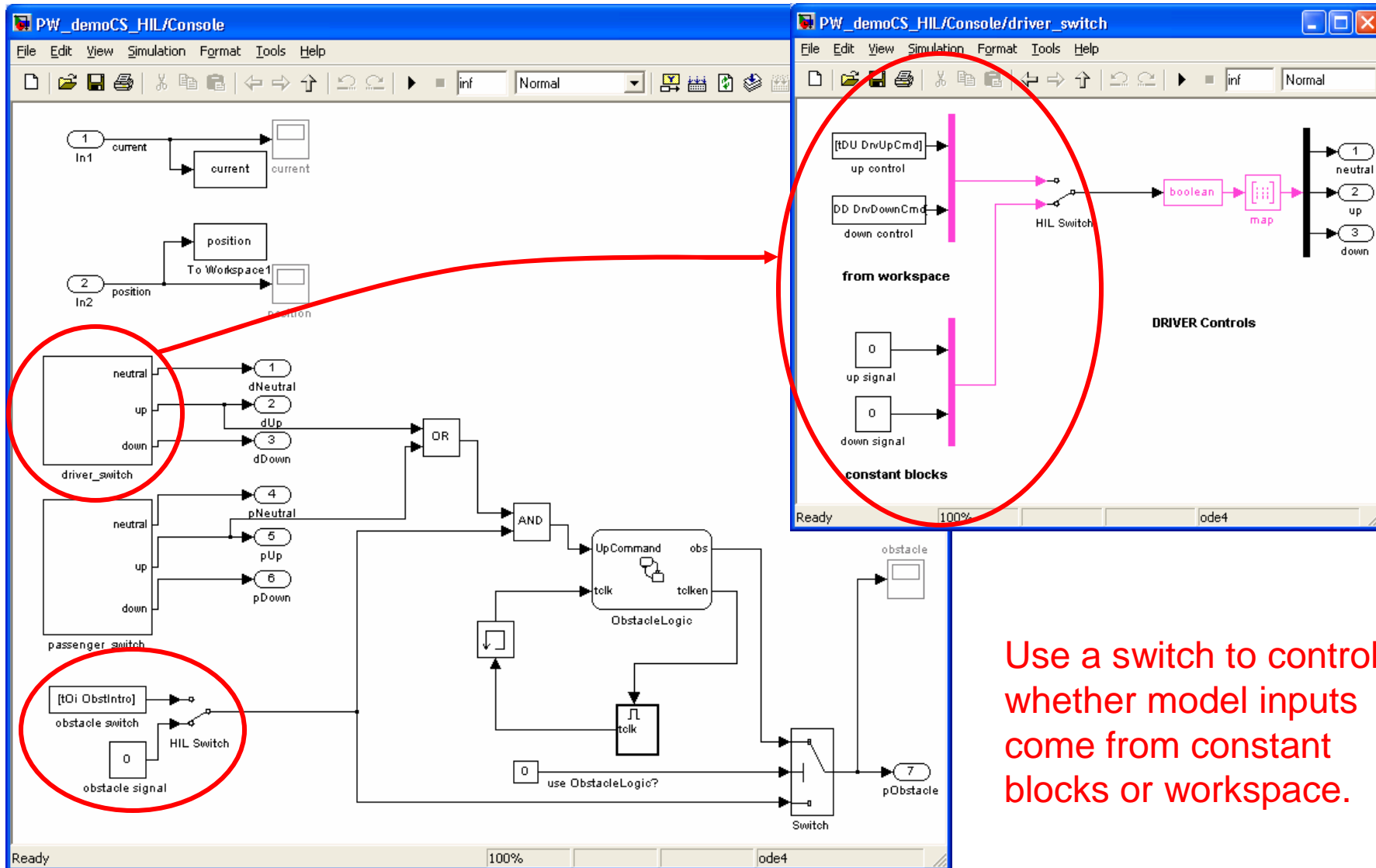


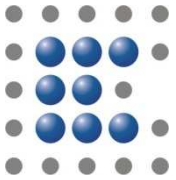
Hardware-in-the-Loop (HIL)

- How will your software control the model inputs?
 - Some software can only change the value of constant blocks
 - Some software can only change the value of workspace signals
- We want to build a model that will work with either setup
- **Start from MIL model again** to extract the source model blocks, not the S-function



Hardware-in-the-Loop (HIL)

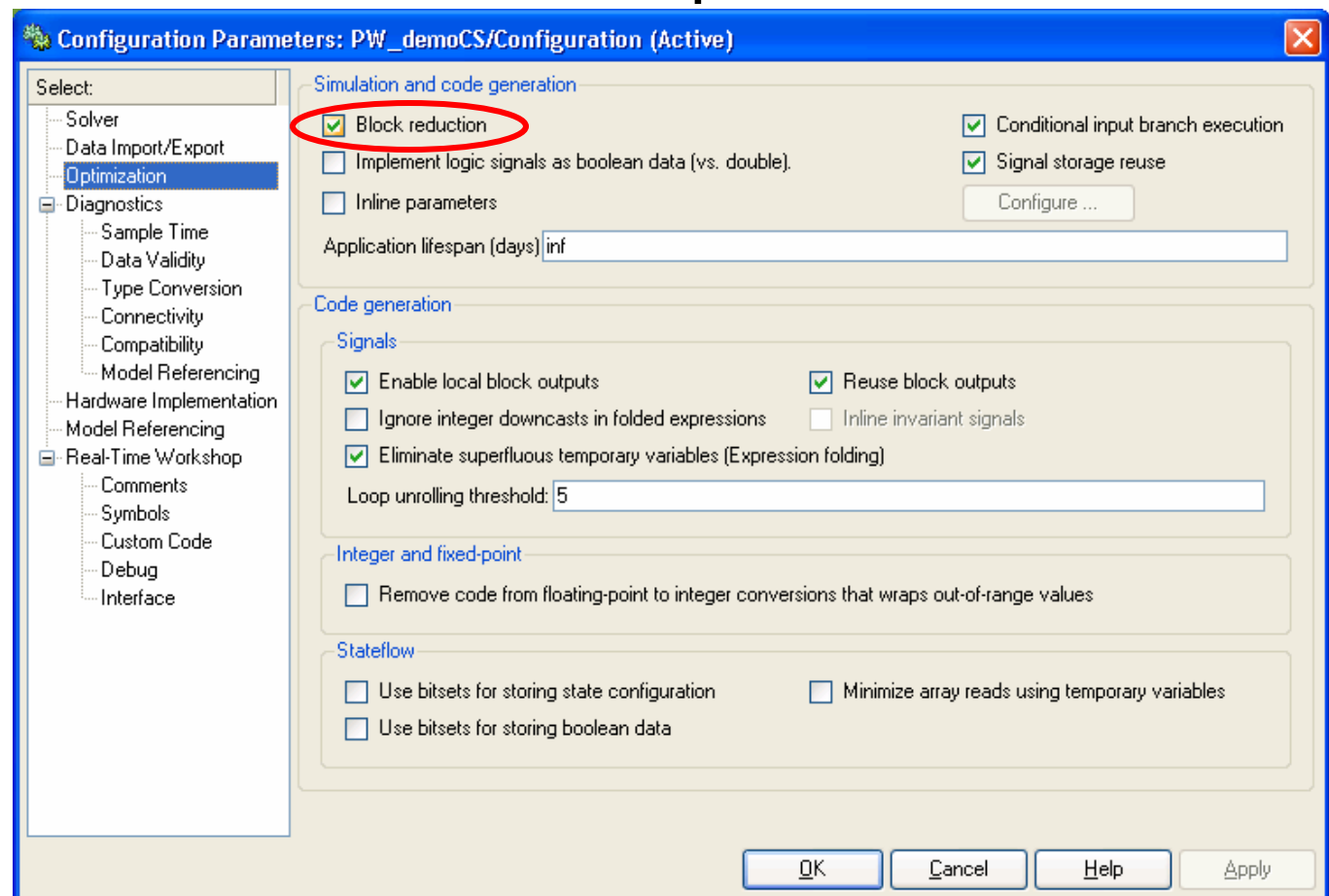




Hardware-in-the-Loop (HIL)

1. Build the model for Real-Time
 - a. Turn off “Block reduction” option

This option will restructure the model for simulation speed, but that prevents us from controlling specific blocks directly from a real-time simulator (HIL). It must be turned off to build a HIL model.



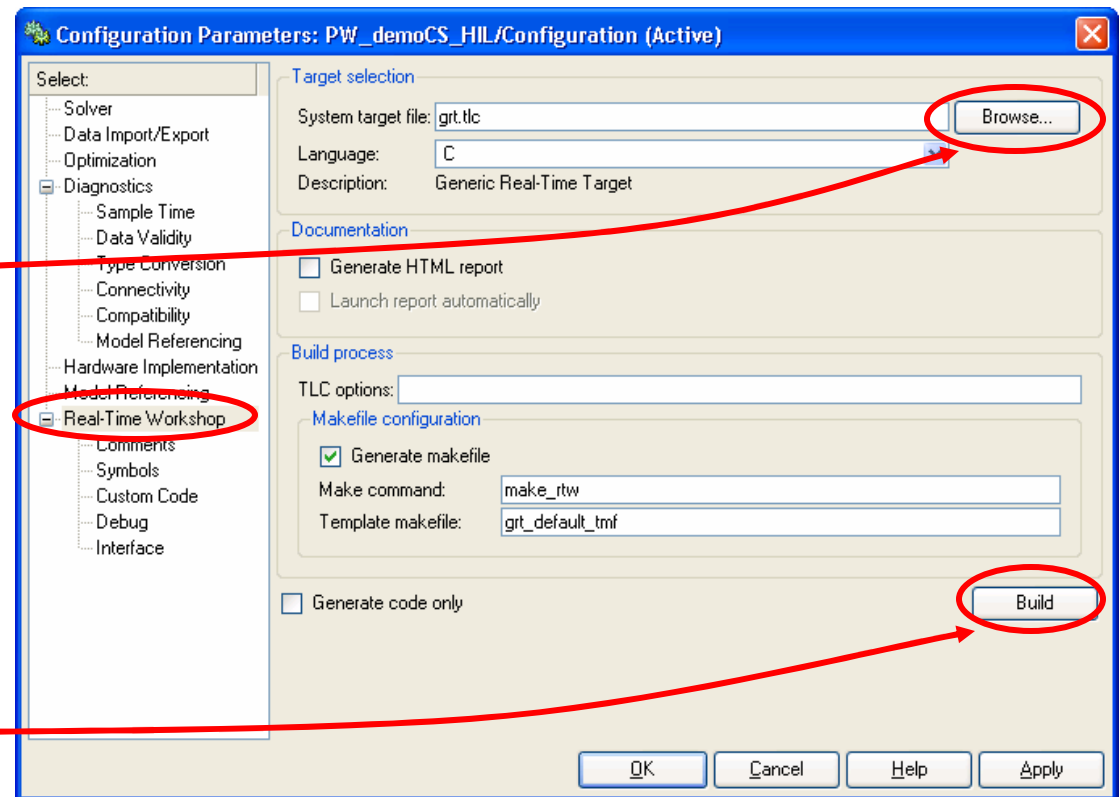


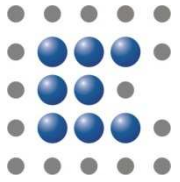
Hardware-in-the-Loop (HIL)

1. Build the model for Real-Time
 - a. Turn off “Block reduction” option
 - b. Specify the target system
 - c. Click Build

Use the Browse button to find your target platform for the build.

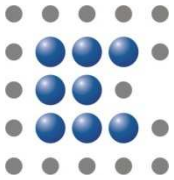
Click Build to generate the HIL files.





Hardware-in-the-Loop (HIL)

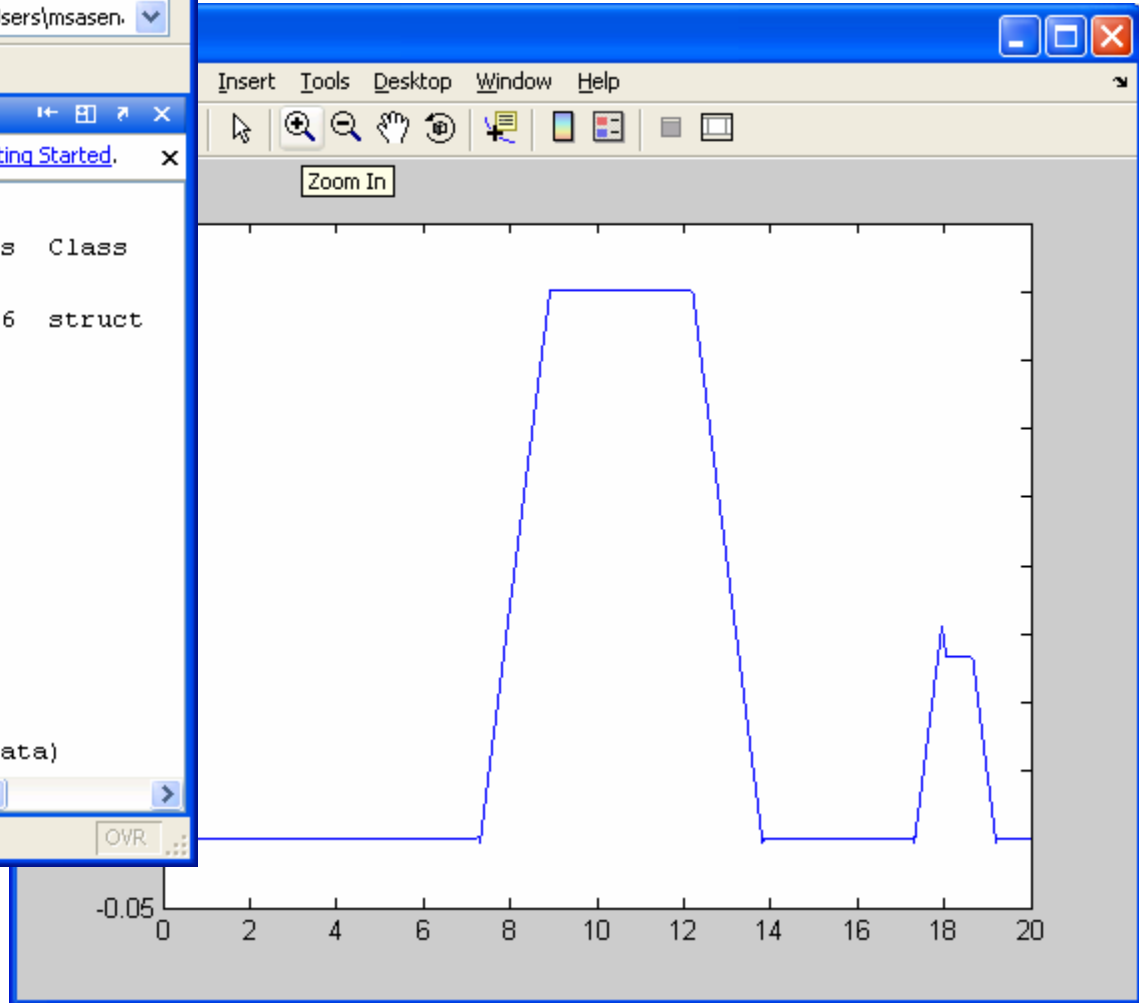
2. Run model and hardware together in real-time simulator
 - a. Could use any number of platforms, so we won't show this step
3. Analyze results
 - a. Store signals from test
 - b. Read into MATLAB
 - c. Compare to expected results
 - d. Report as "Pass" / "Fail"

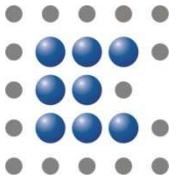


Hardware-in-the-Loop (HIL)

```
MATLAB 7.5.0 (R2007b)
File Edit Debug Desktop Window Help
C:\Users\msasen.
Shortcuts How to Add What's New
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
>> whos
Name          Size          Bytes  Class
test1data     1x1           324136 struct
>> test1data
test1data =
      X: [1x1 struct]
      Y: [1x1 struct]
  Description: [1x1 struct]
    RTProgram: [1x1 struct]
      Capture: [1x1 struct]
>> plot(test1data.X.Data,test1data.Y.Data)
```

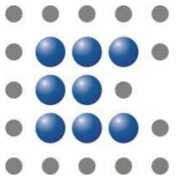
After loading .mat file into workspace...





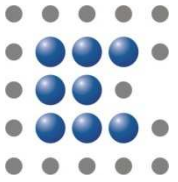
Overview

- Introduction
- Power Window Model
- MIL
- SIL
- HIL
- **Review**



Review

- Started with a simple Power Window model
- Modified it for automated testing
 - from workspace blocks
 - testing script
- Modified it for real-time application
 - fixed-step solver
 - rate transition blocks
 - created multiple input conditions to run on different control software



Review

- MIL: ran the model in automated testing to validate the control algorithm
- SIL: built S-functions of controller to validate the C-code implementation
- HIL: built MIL for real-time simulator to validate the hardware performance



Thank you
and
good luck!!