Coupled Tanks Simulink Files

Kerianne Gross

March 12, 2014

This document provides an introductory overview of the Coupled Tanks Simulink Model and Verification. It is assumed the reader has a basic understanding of MATLAB, Simulink, AADL, AGREE, and formal methods analysis, but may not have knowledge of Simulink Design Verifier.

Contents

[List of Figures 4](#_Toc432079661)

[Comments on the Model 5](#_Toc432079662)

[Before you run the files 5](#_Toc432079663)

[File Structure 6](#_Toc432079664)

[Simulation of System Level Model 7](#_Toc432079665)

[Controller 8](#_Toc432079666)

[Tank 1 Controller 8](#_Toc432079667)

[Tank 2 Controller 9](#_Toc432079668)

[Environment 10](#_Toc432079669)

[Actuators 10](#_Toc432079670)

[Tank 1 11](#_Toc432079671)

[Tank 2 11](#_Toc432079672)

[Sensors 11](#_Toc432079673)

[Assumptions 14](#_Toc432079674)

[Properties 14](#_Toc432079675)

[Persistent Variables 14](#_Toc432079676)

[Verification Files 15](#_Toc432079677)

[Properties and Assumptions used to Analyze the Subsystems 17](#_Toc432079678)

[Actuator 17](#_Toc432079679)

[Assumptions 17](#_Toc432079680)

[Properties/Guarantees (Proof Objectives) 17](#_Toc432079681)

[Controller for Tank 1 17](#_Toc432079682)

[Assumptions 17](#_Toc432079683)

[Properties/Guarantees (Proof Objectives) 17](#_Toc432079684)

[Controller for Tank 2 17](#_Toc432079685)

[Assumptions 17](#_Toc432079686)

[Properties/Guarantees (Proof Objectives) 18](#_Toc432079687)

[Sensor 18](#_Toc432079688)

[Assumptions 18](#_Toc432079689)

[Properties/Guarantees (Proof Objectives) 18](#_Toc432079690)

[Tank 1 18](#_Toc432079691)

[Assumptions 18](#_Toc432079692)

[Properties/Guarantees (Proof Objectives) 19](#_Toc432079693)

[Tank 2 19](#_Toc432079694)

[Assumptions 19](#_Toc432079695)

[Properties/Guarantees (Proof Objectives) 19](#_Toc432079696)

[Verification 21](#_Toc432079697)

# List of Figures

Figure 1: Menus to “Add Selected Folders and Subfolders” to the Path 5

Figure 2: Location of constants file in the folder 5

Figure 3: File structure of the model 6

Figure 4: System Level Simulation of the Model 7

Figure 5: Plot of the liquid height in Tank 1 (Yellow) and liquid height in Tank 2 (magenta) versus time 7

Figure 6: Controller Model 8

Figure 7: Tank 1 Controller Model 8

Figure 8: Tank 2 Controller Model 9

Figure 9: Environment Level Model 10

Figure 10: Actuator Model 10

Figure 11: Tank 1 Model 11

Figure 12: Tank 2 Model 11

Figure 13: Sensor Model 11

Figure 14: Example of Property File (from the Sensor Property File) 12

Figure 15: Persistent Variable Example (from the Tank 1 Controller Property File) 13

Figure 16: Location of Simulink Design Verifier Blocks in Library Browser 14

Figure 17: Verification File Example (from Sensor verification file) 14

Figure 18: Design Verifier Menu Navigation for Analysis 18

Figure 19: Simulink Design Verifier Log Window 18

# Comments on the Model

* Files were created in Matlab 2012b
* In order to run the verification files, you must have Simulink Design Verifier installed
* The other files do not require Simulink Design Verifier

# Before you run the files

* Right click on the folder in the Matlab Current Folder Window, and follow the menus to “Add to Path” and “Selected Folders and Subfolders”

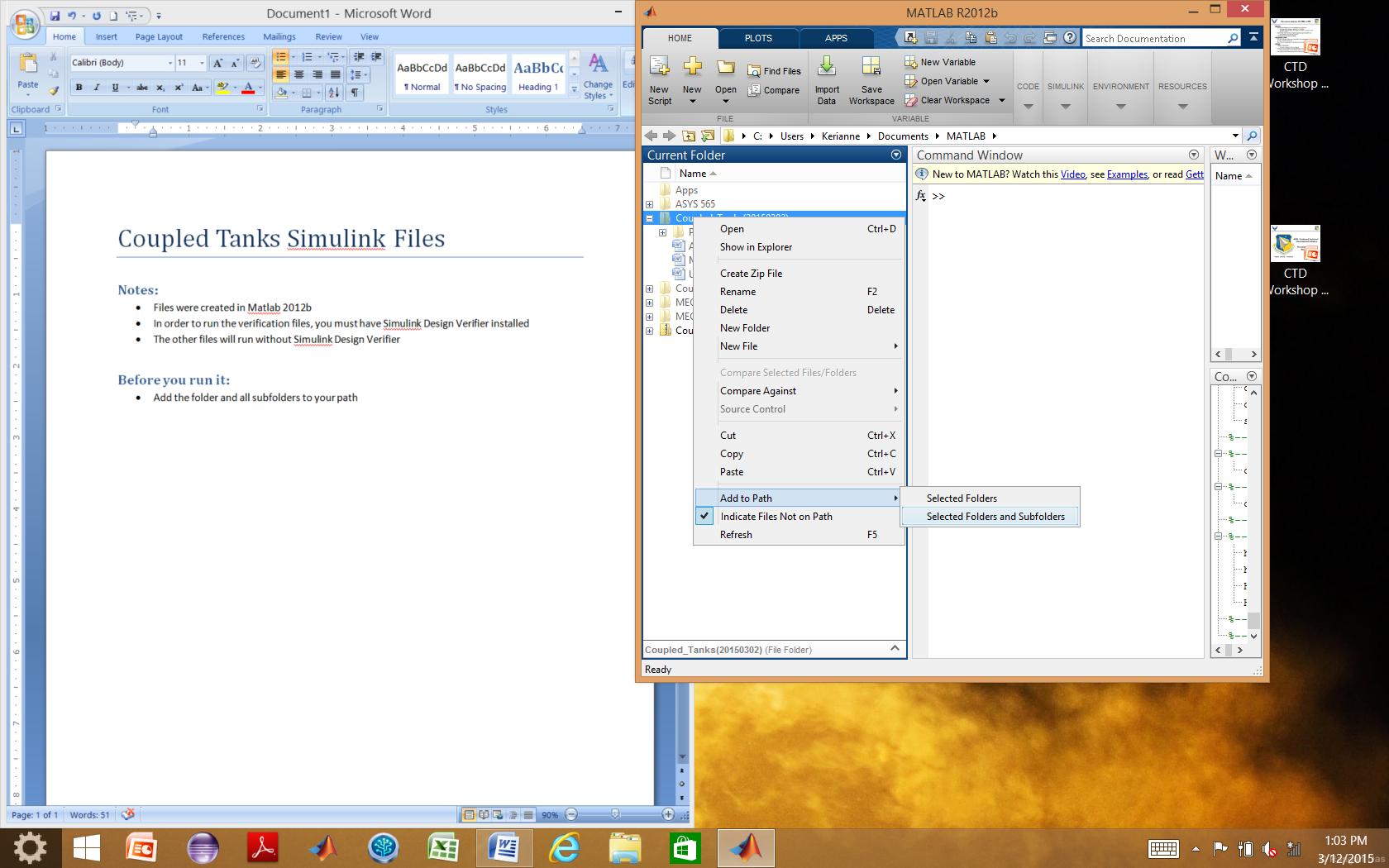


Figure 1: Menus to “Add Selected Folders and Subfolders” to the Path

* Run the “constants.m” file prior to any of the other files

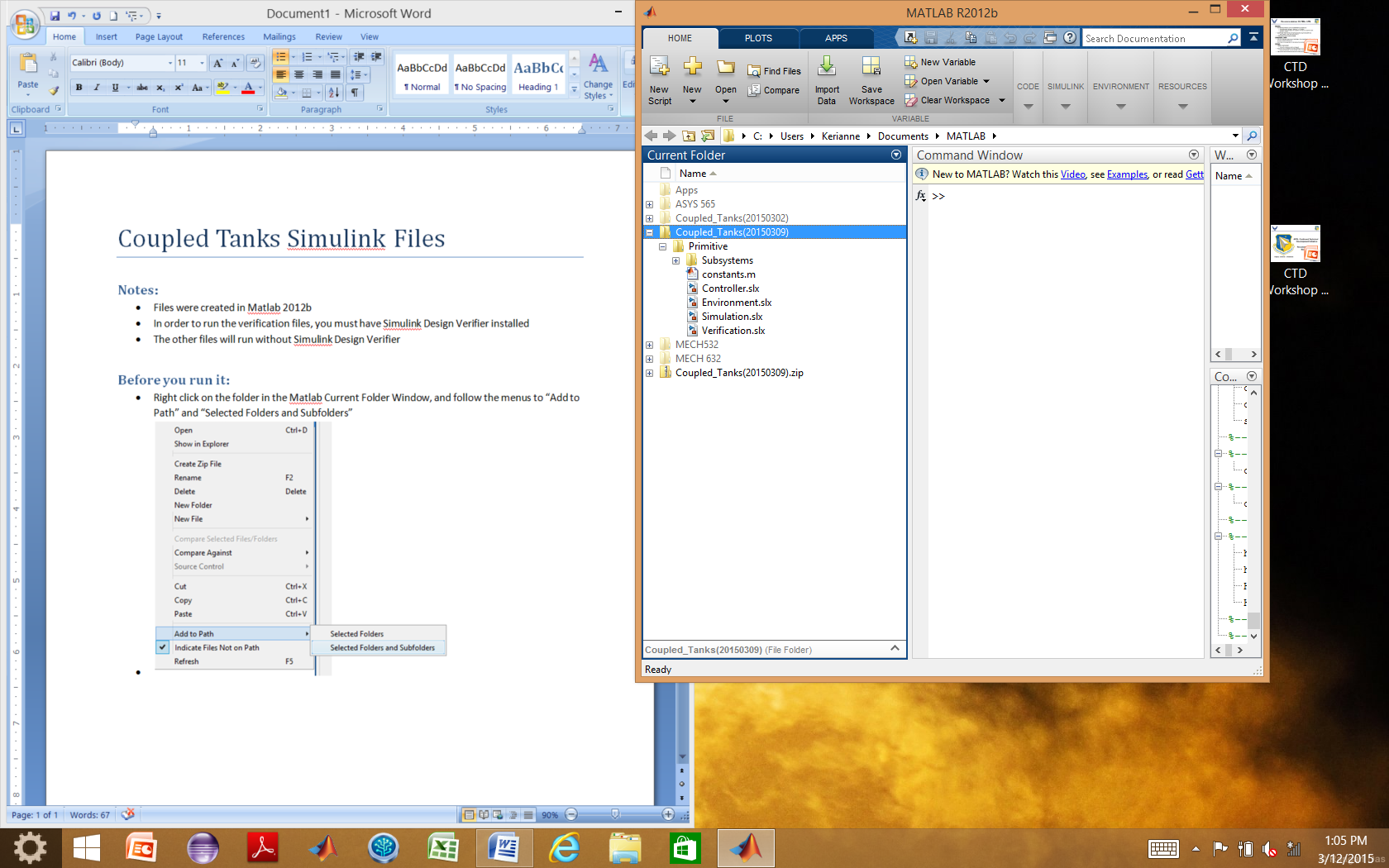


Figure 2: Location of constants file in the folder

# File Structure

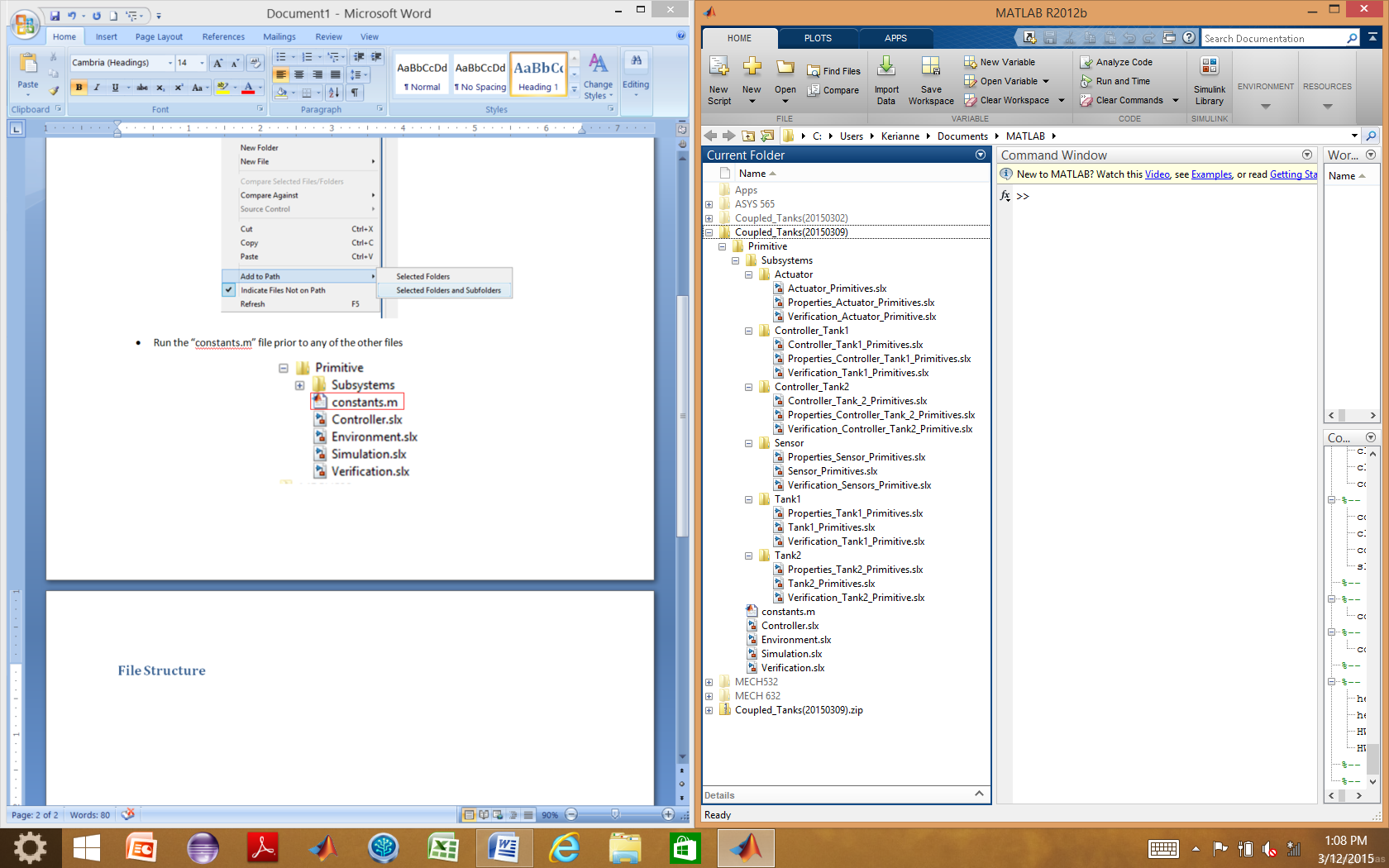


Figure 3: File structure of the model

* System level files are in the main folder, “Primitive” because this model is implemented using includes all of the formally specified primitives
* Subsystem level files are in the subsystem folders
* The **Simulation.slx** is a system level file that references the **Controller.slx** and **Environment.slx**
* The **Controller.slx** file references:
  + Controller\_Tank1\_Primitives.slx
  + Controller\_Tank2\_Primitives.slx
* The **Environment.slx** file references:
  + Actuator\_Primitives.slx
  + Sensor\_Primitives.slx
  + Tank1\_Primitives.slx
  + Tank2\_Primitives.slx
* The **Verification.slx** is a system level file that references **Simulation.slx** to do analysis on the system
* Each of the **Properties files of the subsystems** contains Properties that match the Guarantees and Assumptions that match Assumptions developed using AGREE in the AADL architecture phase
* Each of the subsystem Verification files references that subsystem model and the subsystem properties

# Simulation of System Level Model

* As stated in the File Structure Section, **Simulation.slx** is a system level file that references the **Controller.slx** and **Environment.slx**
* It ports the signals appropriately between the controller and environment and includes a scope to view the change in the height of the liquid in each tank
* A unit delay is placed between the signals from the environment to the controller to prevent an algebraic loop by accounting for the delay in time steps in the simulation
* Simulation.slx is shown in Figure 4

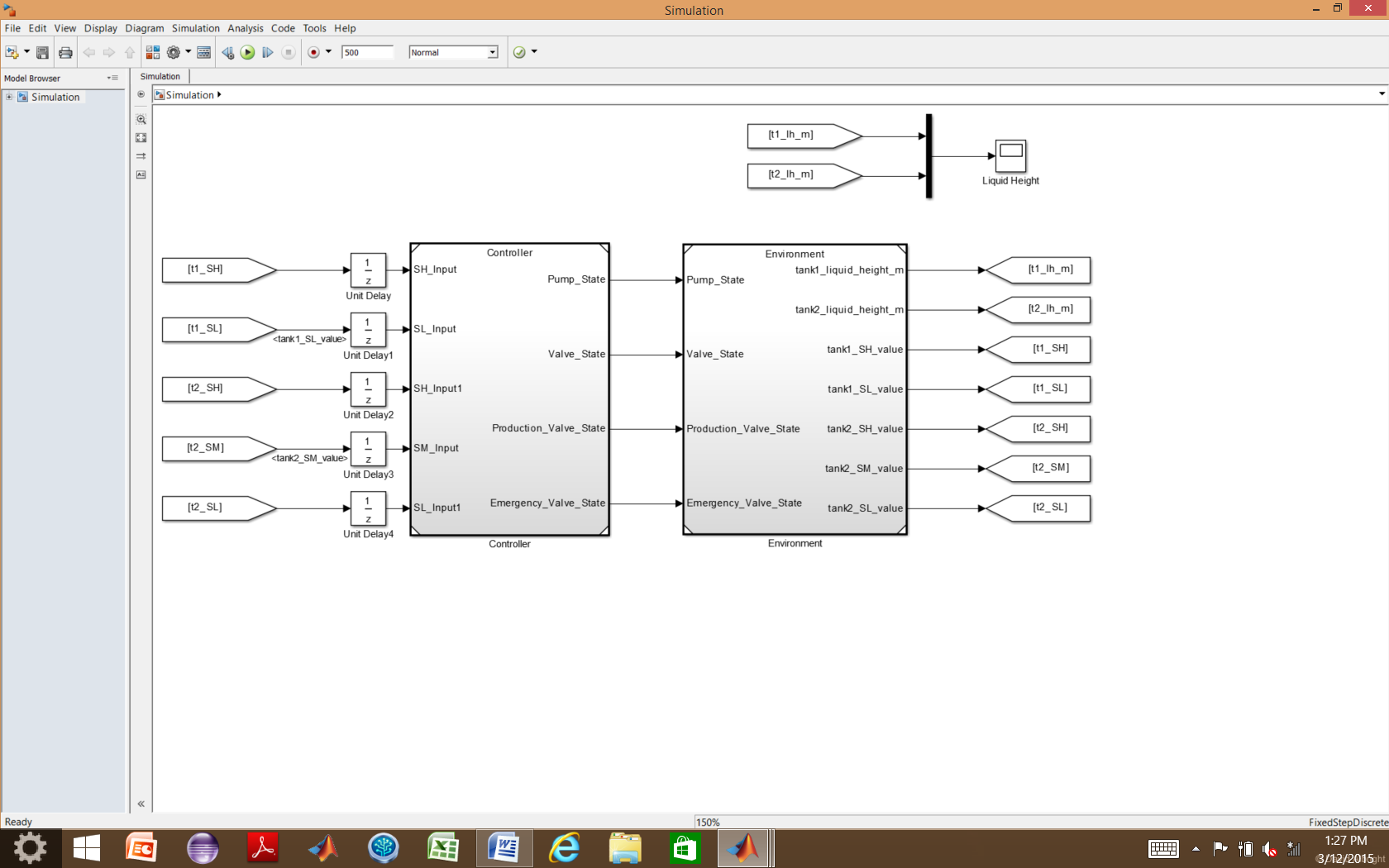


Figure 4: System Level Simulation of the Model



Figure 5: Plot of the liquid height in Tank 1 (Yellow) and liquid height in Tank 2 (magenta) versus time

# Controller

* The **Controller.slx** file references **Controller\_Tank1\_Primitives.slx** and **Controller\_Tank2\_Primitives.slx** and appropriately routes signals between the two
* Sensor Inputs are from the environment, but do not actually reference anything at this level

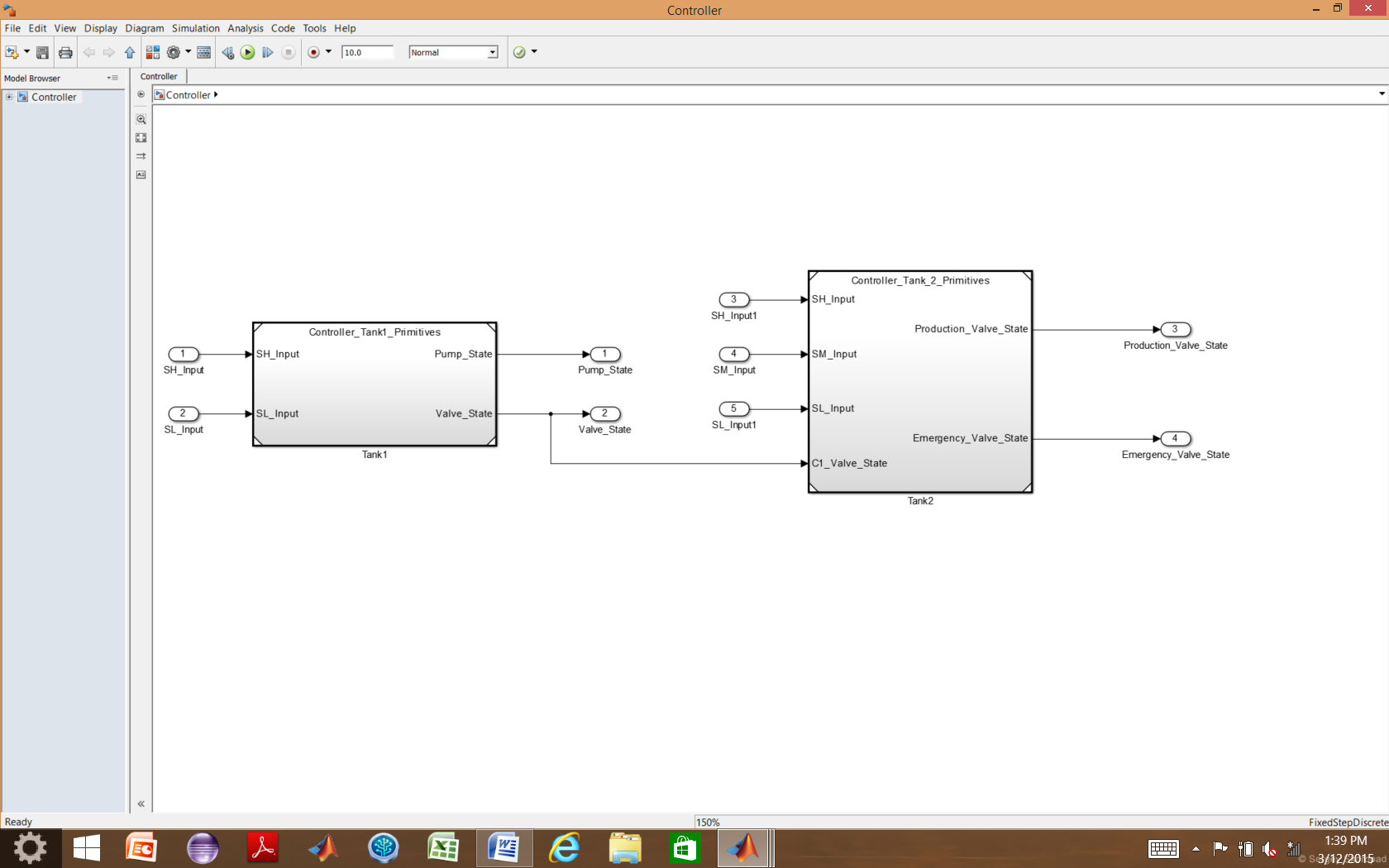


Figure 6: Controller Model

## Tank 1 Controller

The tank 1 controller implemented with Simulink primitives is shown in Figure 7. Unit delays are utilized to create the “previous state” of the pump and valve, used by the controller.

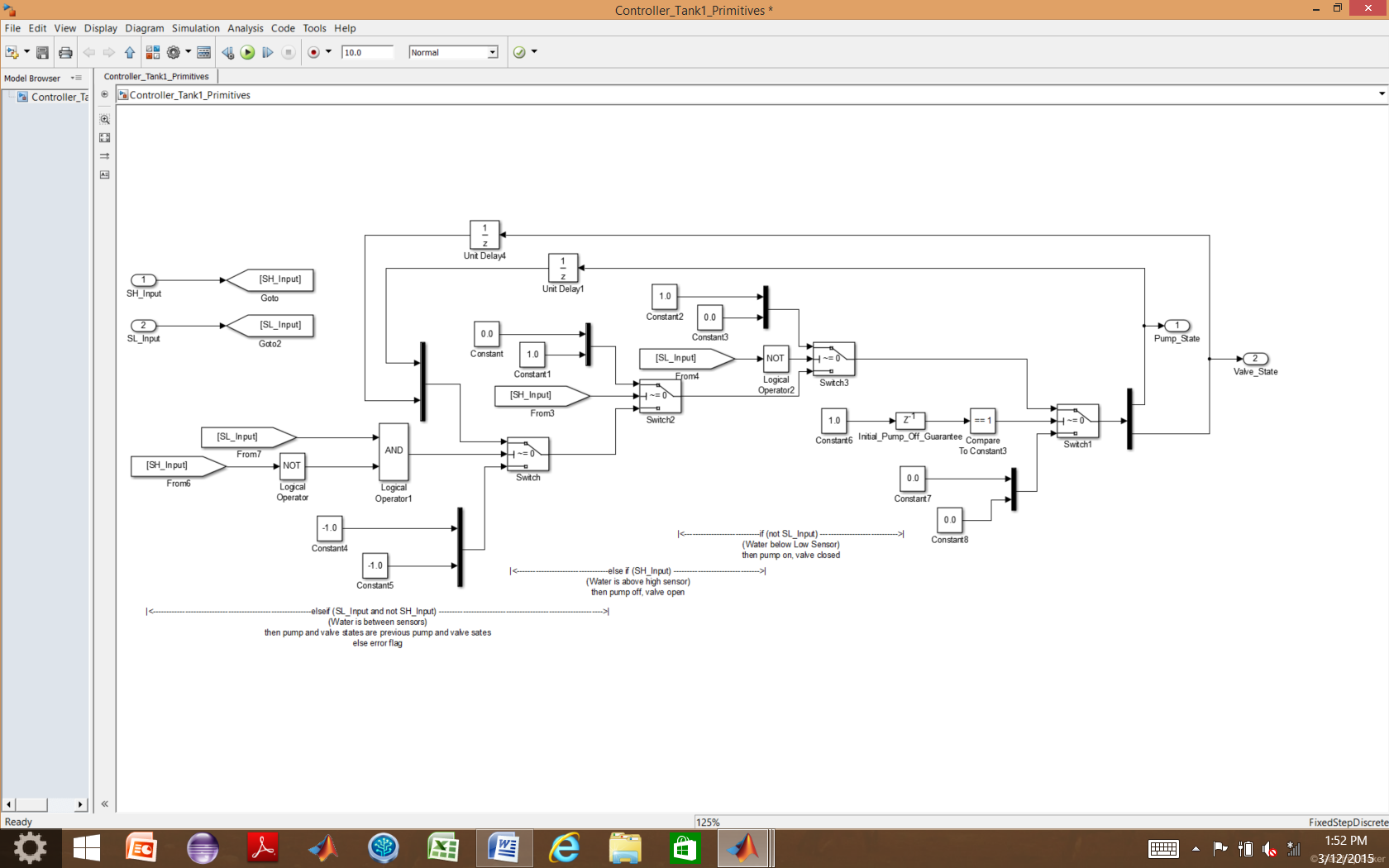
****

Figure 7: Tank 1 Controller Model

## Tank 2 Controller

The tank 2 controller implemented with Simulink primitives is shown in Figure 8. Delay blocks are used to compute the “previous state” of the Emergency and Production valve for use by the controller in the next time step.

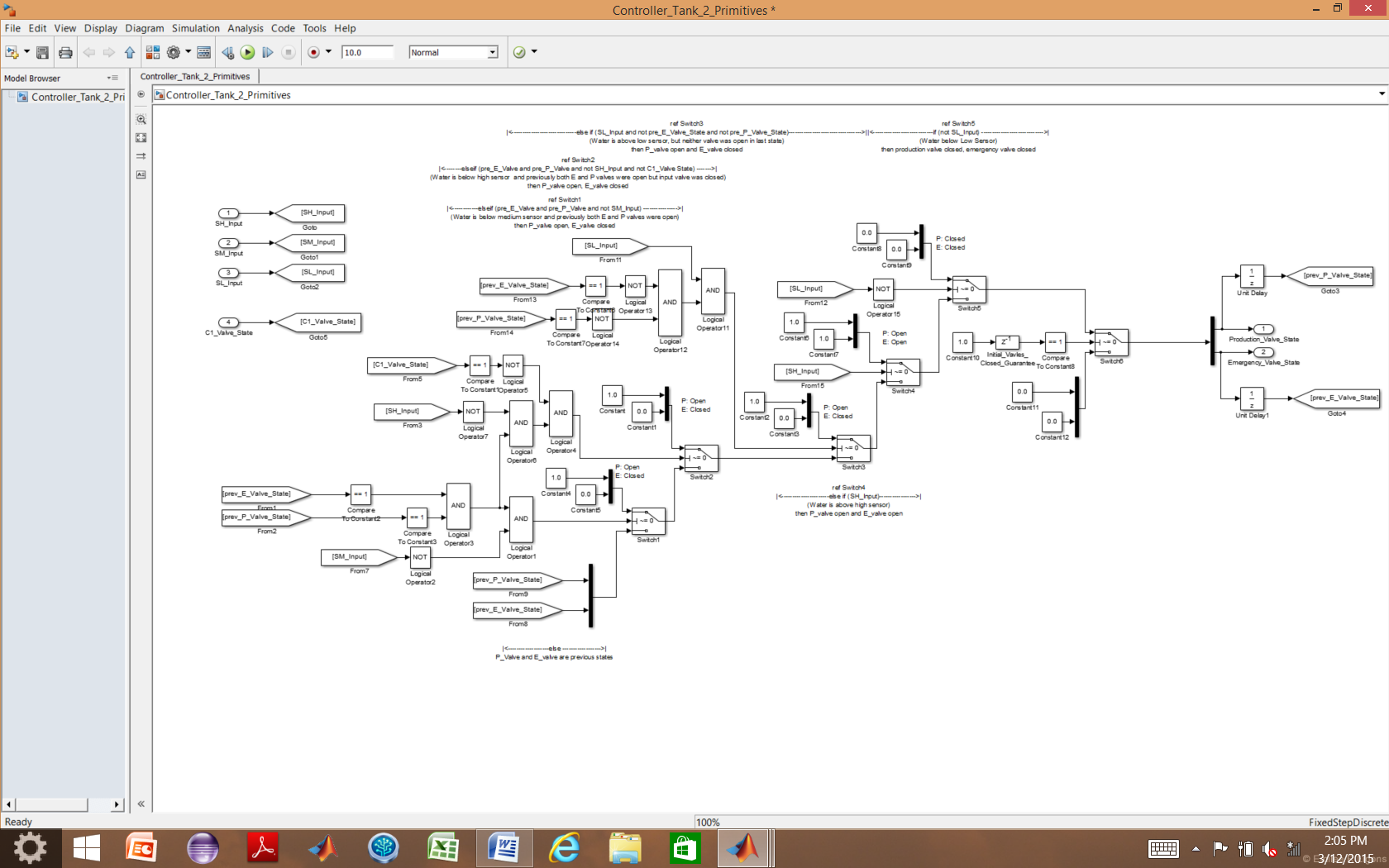


Figure 8: Tank 2 Controller Model

# Environment

The **Environment.slx** file references:

* Actuator\_Primitives.slx four times to model the output flow rate of the Pump, Valve, Production Valve, and Emergency Valve based on the signals each receives from the controller, the time increment, and the flow rate of each component (listed in the constants.m file)
* Tank1\_Primitives.slx to model the liquid height of Tank1 based on the flow rates of the Pump and Valve
* Tank2\_Primitives.slx to model the liquid height of Tank2 based on the flow rates of the Valve, Production Valve, and Emergency Valve
* Sensor\_Primitives.slx five times to model the each of the five sensor reactions (true or false) to the current height of the liquid based on its location (defined by the constants.m file) in its respective tank
* Unit Delay blocks are incorporated between the Tanks and the Sensors

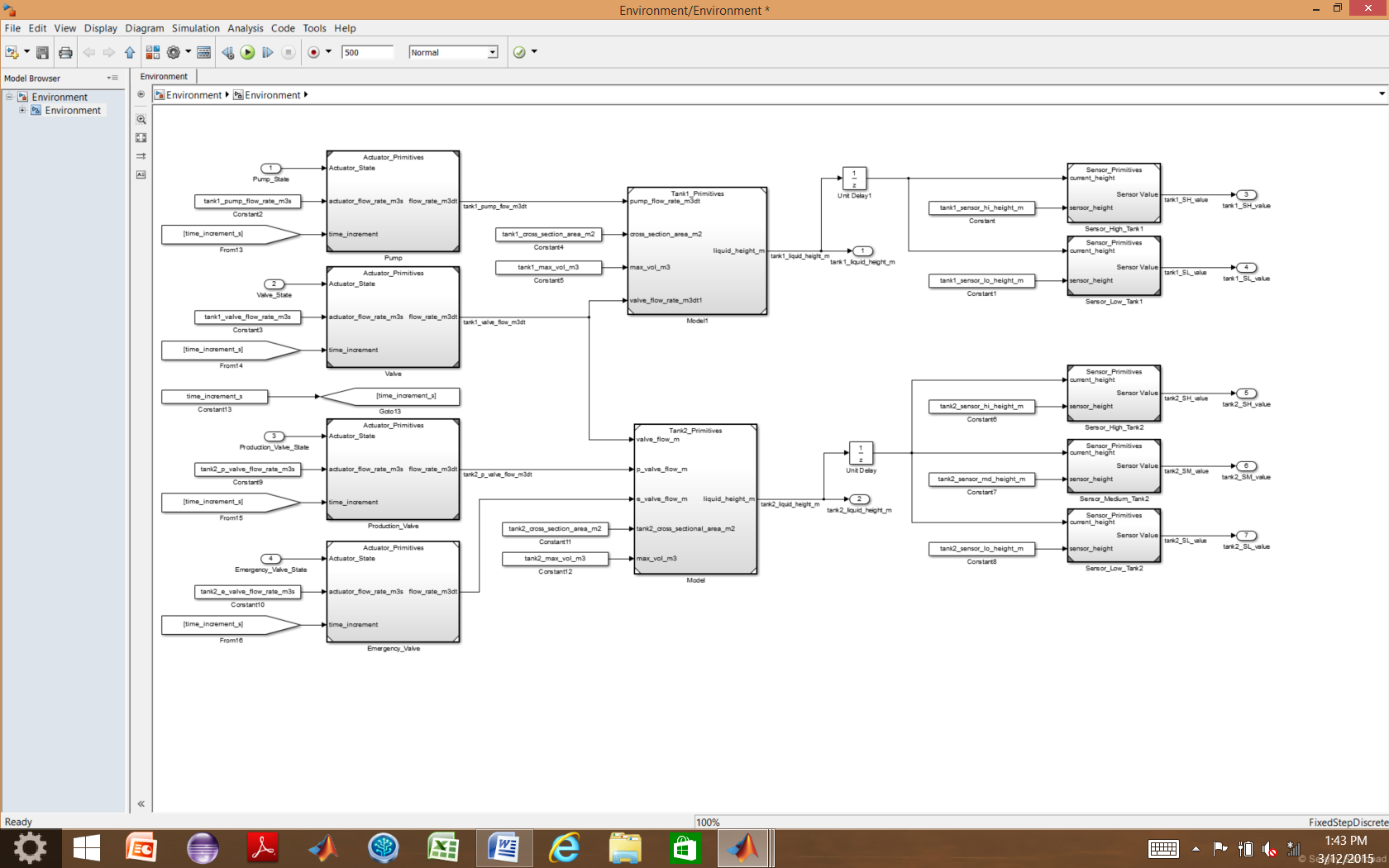


Figure 9: Environment Level Model

## Actuators

The same actuator model, shown in Figure 10, is used to model the performance of the Pump, Valve, Production Valve, and Emergency Valve.

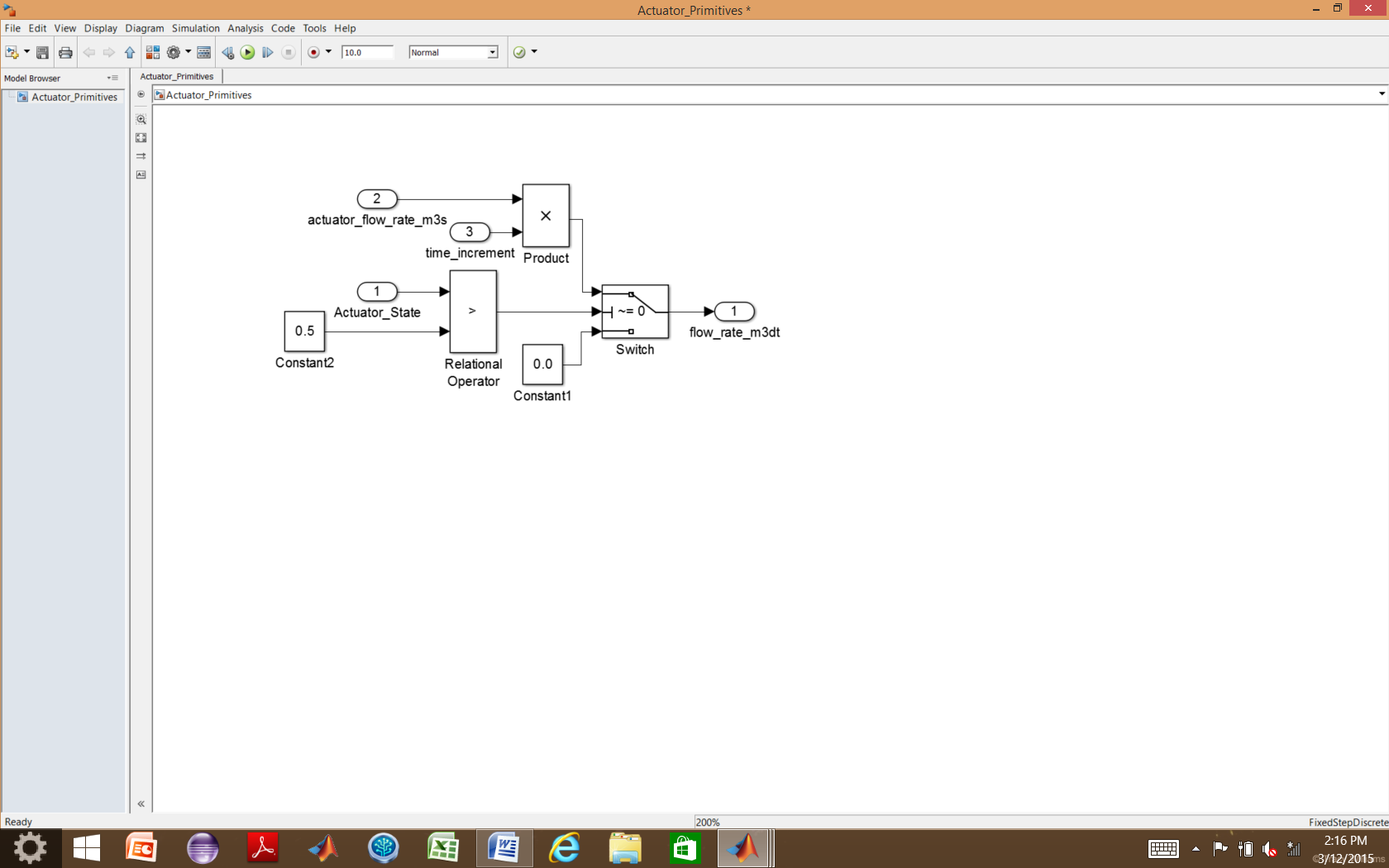


Figure 10: Actuator Model

## Tank 1

The model used to simulate Tank 1 is shown in Figure 11. A unit delay is used to calculate the “previous height” used in the next time step’s calculations.

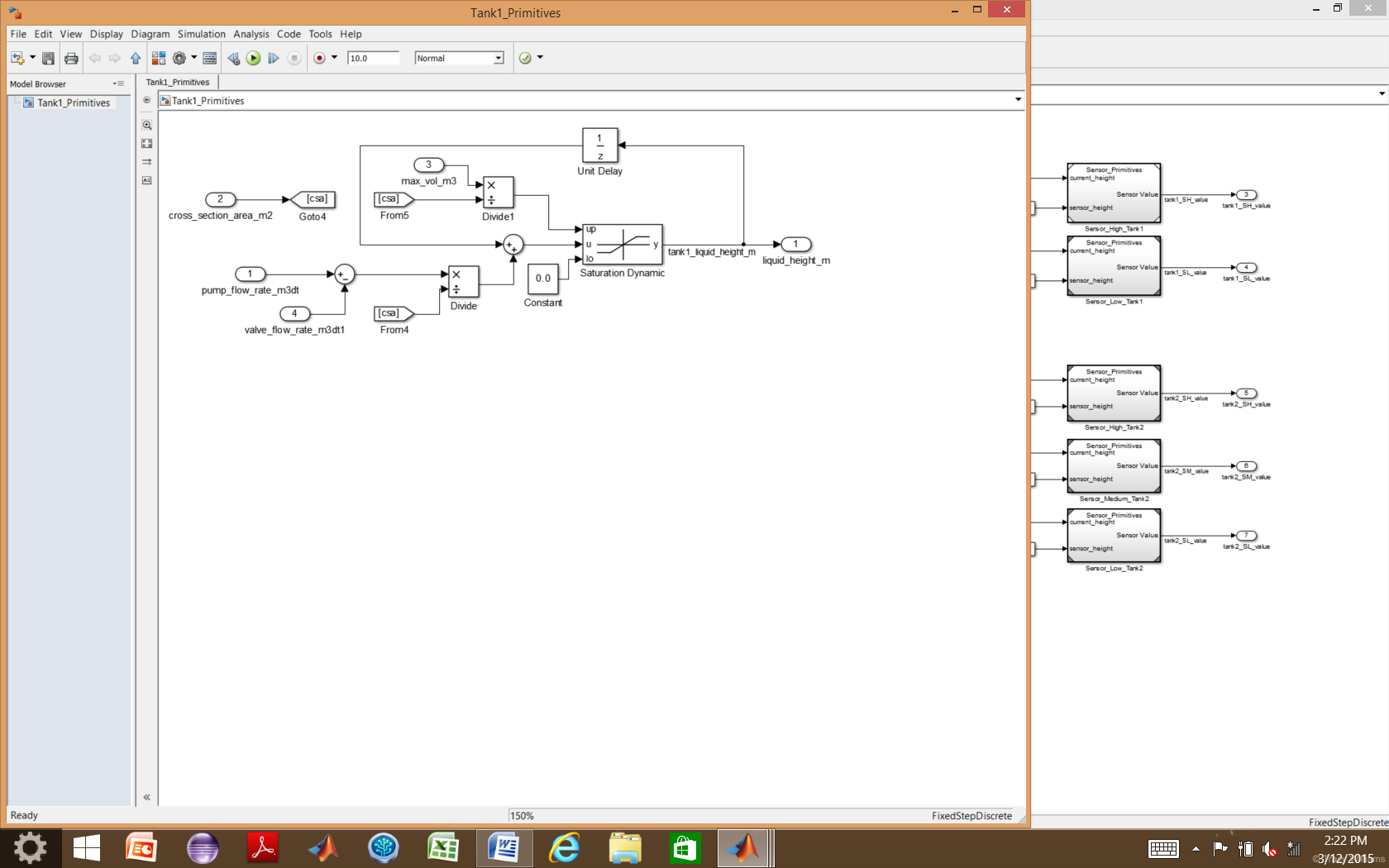


Figure 11: Tank 1 Model

## Tank 2

The model used to simulate Tank 2 is shown in Figure 12. A unit delay is used to calculate the “previous height” used in the next time step’s calculations.

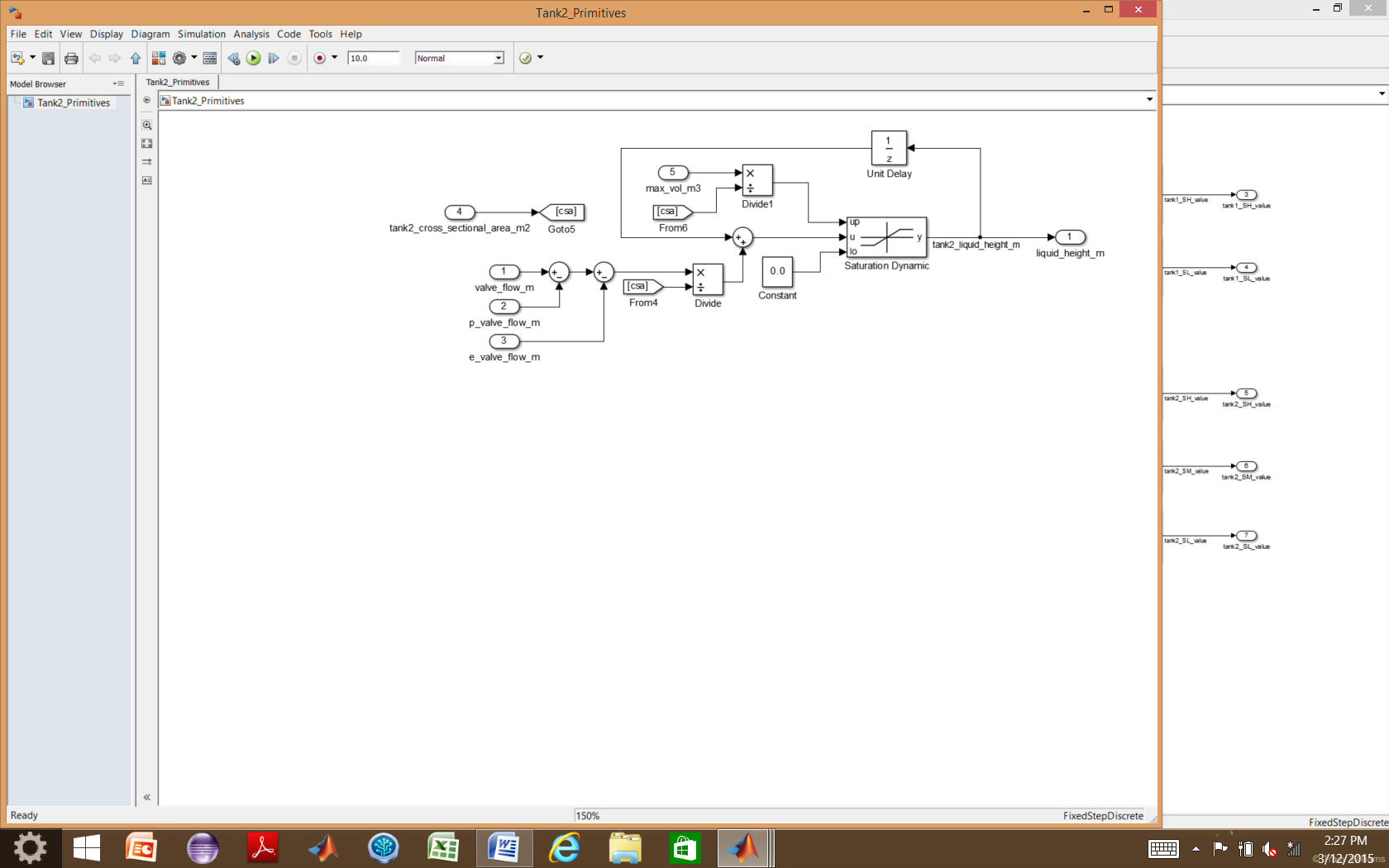


Figure 12: Tank 2 Model

## Sensors

The model used to simulate all five of the sensors used in the environment is shown in Figure 13.

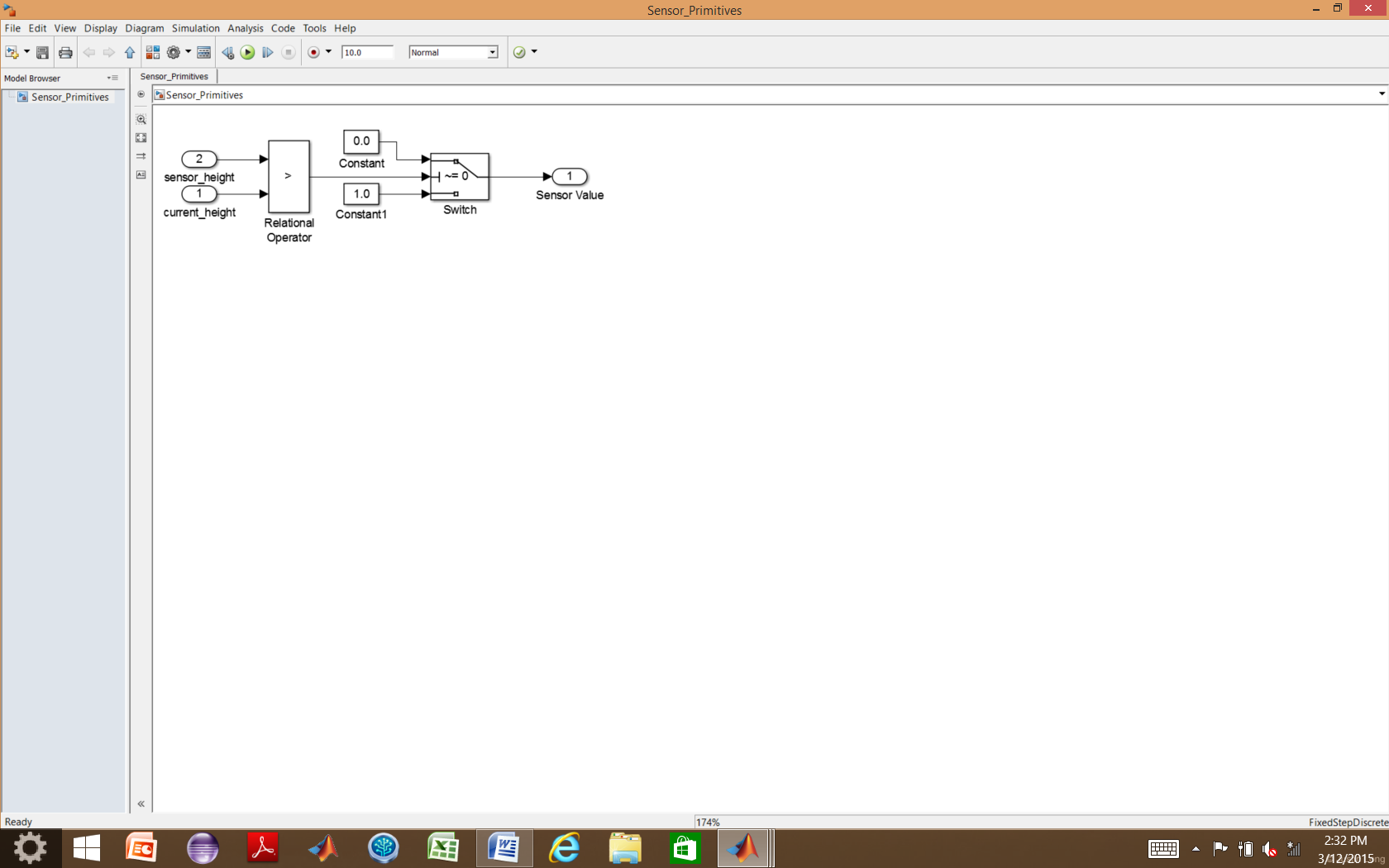


Figure 13: Sensor Model

Properties Files

A property file exists in Simulink as a MATLAB function block for each subsystem and is used to conduct verification of the subsystem. The properties files contain MATLAB functions, as seen in Figure 14.

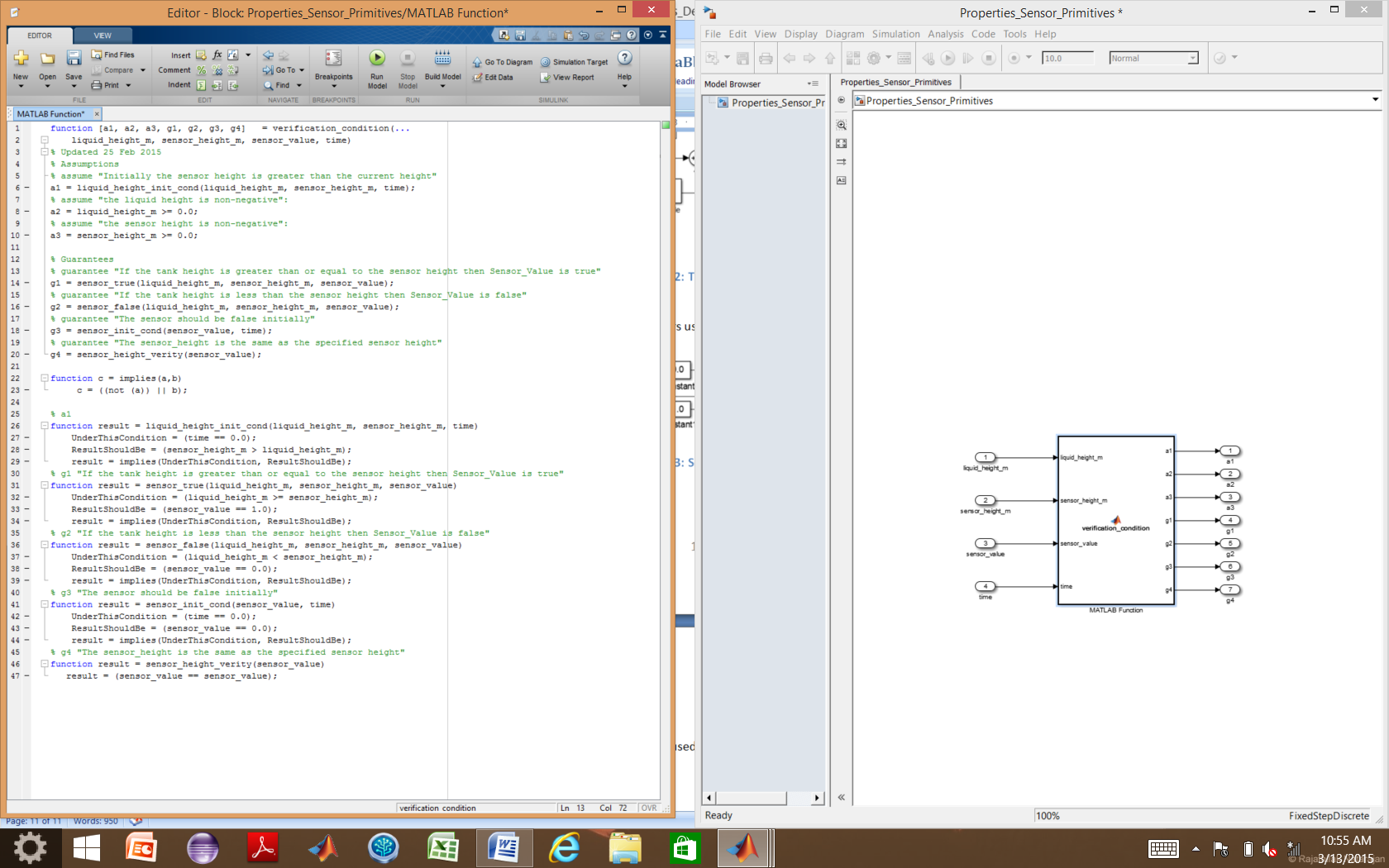


Figure 14: Example of Property File (from the Sensor Property File)

Each property file contains several functions. The main function accepts variables of interest, typically the inputs and outputs of the property in question, (in this example, the inputs to the main function for the sensor properties file are the inputs and outputs of the sensor model), and outputs the result of the analysis of the assumptions and properties for that subsystem. The other functions support analysis of the properties.

## Assumptions

If the design of the system is done in the VVAS envisioned approach of formal requirements specification and analysis, followed by formal architecture specification and analysis, followed by modeling and analysis, the assumptions specified in the property file should match the assumptions for the subsystem documented in the AGREE annex of the AADL architectural implementation in the previous design phase. The assumptions constrain the model checking and should be used with caution. In the example, assumptions are used to constrain the value of the liquid height and sensor height to non-negative numbers and to establish that the expected initial condition of the tank is that the liquid height will be below the sensor.

## Properties

If the architecture is developed in AADL with assumptions and guarantees written in the AGREE annex, the properties of the property file should match the guarantees of the AGREE file for that subsystem. For this reason, the properties are named “g1, g2, etc.” The properties describe performance characteristics that the model is expected to hold, and the objective of the analysis is to prove that the properties are satisfied by the model.

## Persistent Variables

When the previous value of a variable, a counter, or time tracking of a condition is needed in order to analyze a property of the system, persistent variables are used. Persistent variables hold their value in memory until the next time the function is called, but differ from global variables in that they are local to the function where they are defined. An example of the use of persistent variables in the Properties file for the Tank 1 Controller is displayed in Figure 15.

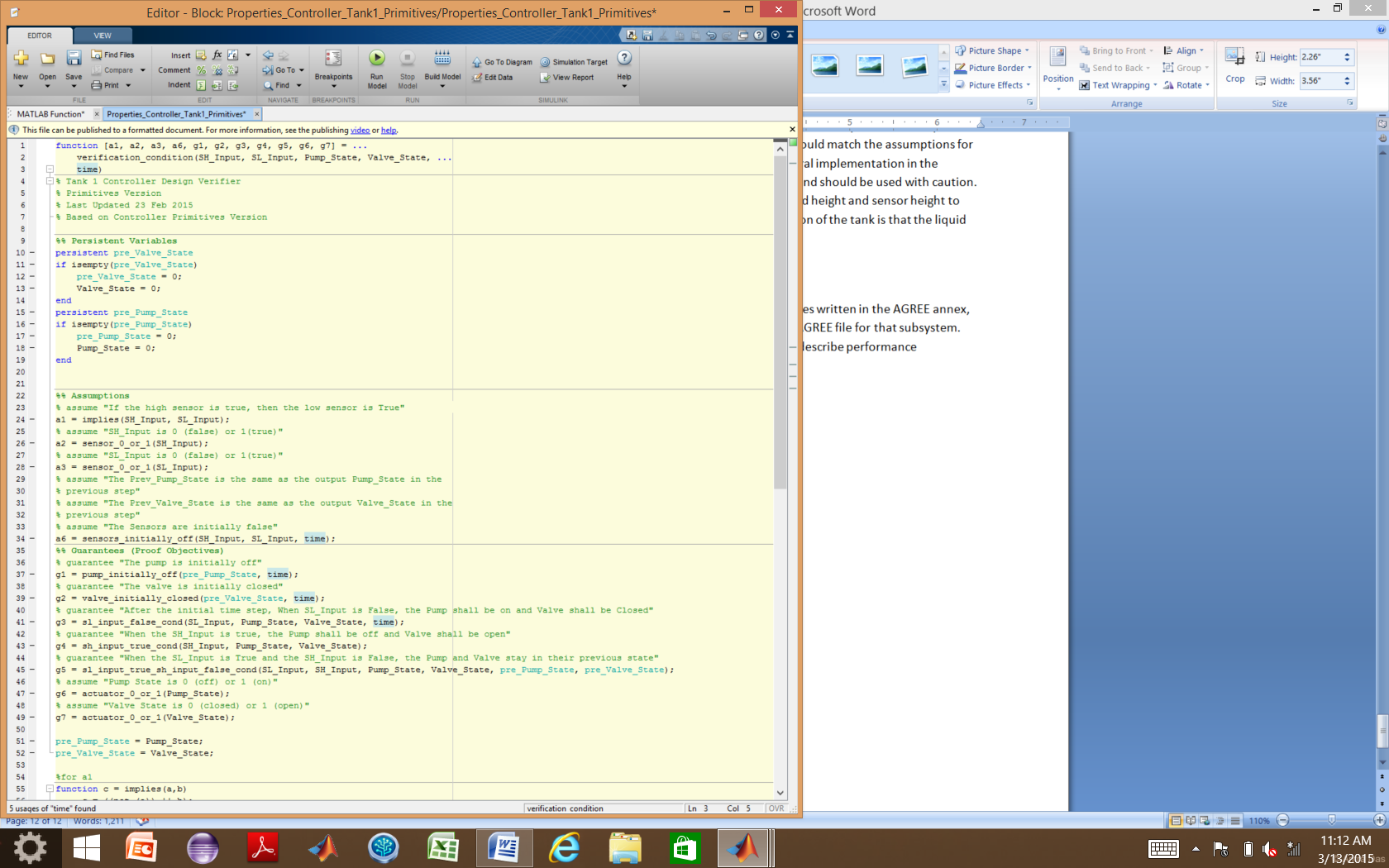
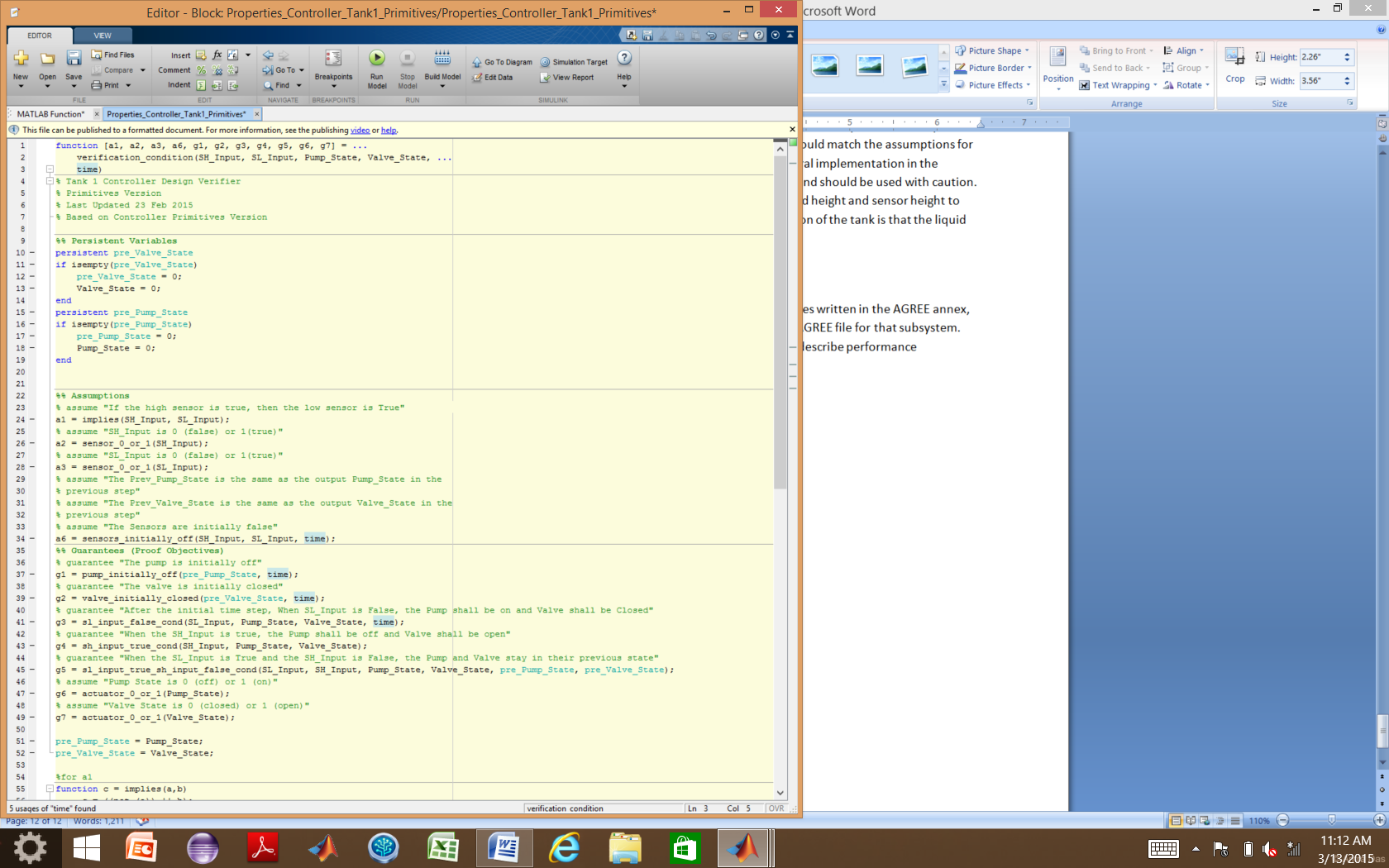


Figure 15: Persistent Variable Example (from the Tank 1 Controller Property File)

To use a persistent variable, it must first be declared, and then given an initial value using an *if statement* and the *isempty* function. The value of that variable should be updated to reflect the new state of the variable at the end of the function call.

# Verification Files

Verification requires a license for Simulink Design Verifier. With Simulink Design Verifier, the blocks used to do formal analysis of the subsystem model are the “Assumption” and “Proof Objective” blocks, shown in Figure 16.

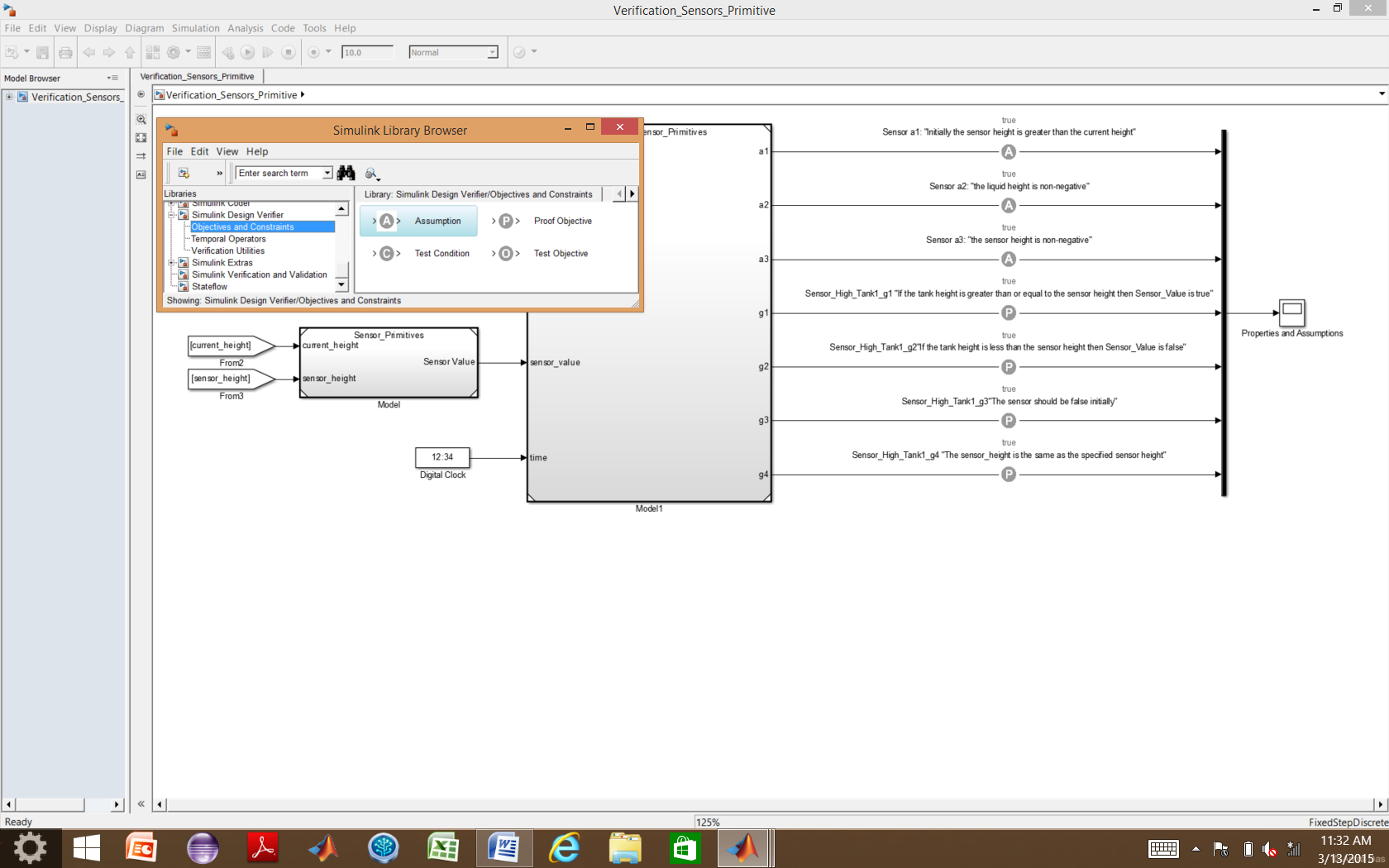


Figure 16: Location of Simulink Design Verifier Blocks in Library Browser

The verification files reference the model of the system and the properties model. The variables of interest (usually the inputs and outputs of the model block) are connected to the inputs of the properties block. The “Assumption” blocks are connected to the outputs of the property file corresponding to the assumptions “a1, a2, etc.” and the “Proof Objective” blocks are connected to the outputs of the property file corresponding to the properties “g1, g2, etc.” An example of a verification file for the sensor is shown in

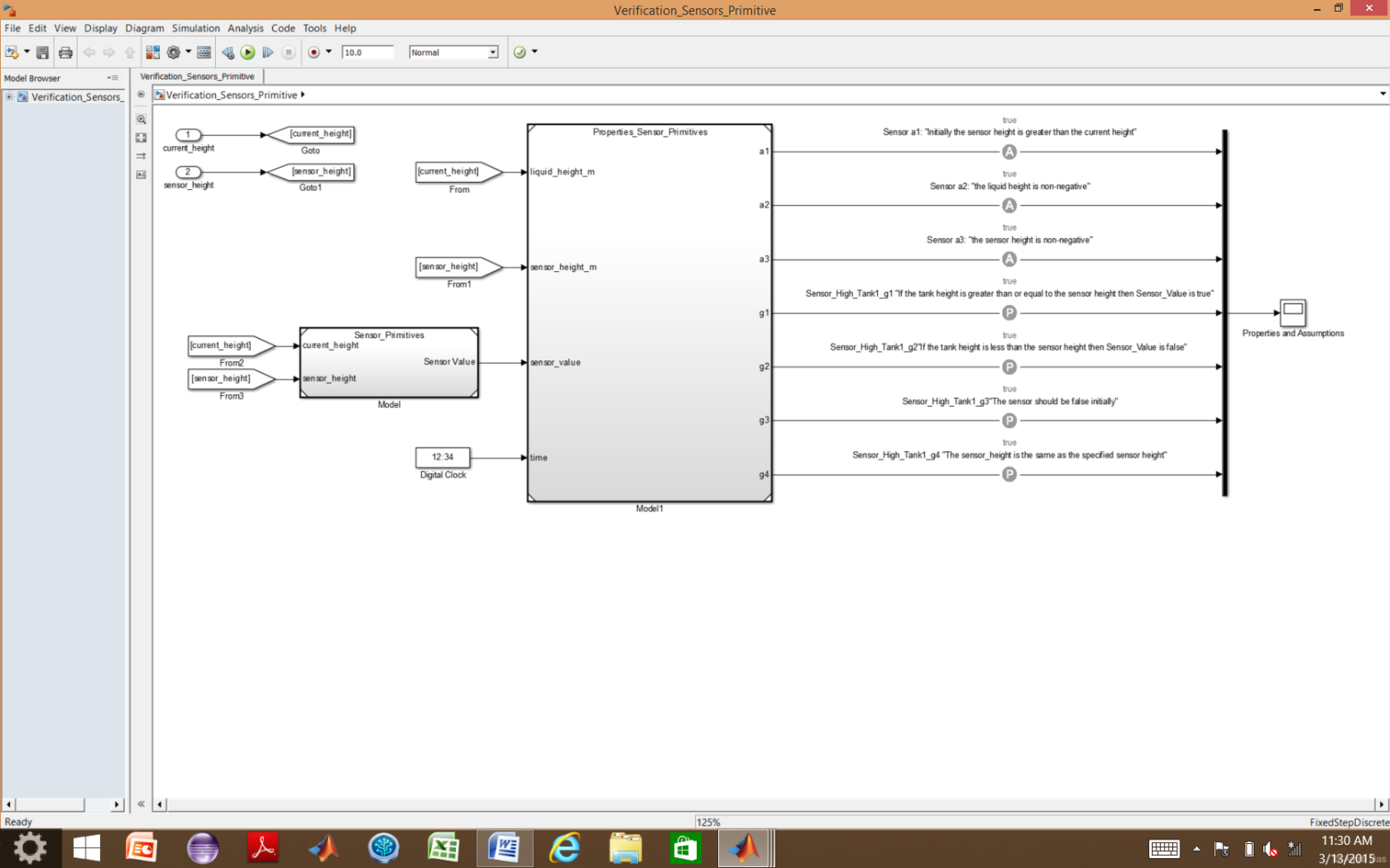


Figure 17: Verification File Example (from Sensor verification file)

# Properties and Assumptions used to Analyze the Subsystems

## Actuator

### Assumptions

* a1: "The actuator state is either 0 or 1"
* a2: "The time increment is positive"
* a3: "The actuator flow rate is positive"

### Properties/Guarantees (Proof Objectives)

* g1: "If the actuator is receiving an ON|OPEN signal then the flow through the actuator is the max flow rate"
* g2: "If the actuator is receiving an OFF|CLOSED signal then the flow rate through the actuator is 0.0"
* g3: "If there is flow through the actuator, then the actuator is receiving an ON|OPEN signal"
* g4: "If the flow is 0.0, then the pump is receiving on OFF|CLOSED signal"

## Controller for Tank 1

### Assumptions

* a1: "If the high sensor is true, then the low sensor is True"
* a2: "SH\_Input is 0 (false) or 1(true)"
* a3: "SL\_Input is 0 (false) or 1(true)"
* a4: "The Sensors are initially false"

### Properties/Guarantees (Proof Objectives)

* g1: "The pump is initially off"
* g2: "The valve is initially closed"
* g3: "After the initial time step, When SL\_Input is False, the Pump shall be on and Valve shall be Closed"
* g4: "When the SH\_Input is true, the Pump shall be off and Valve shall be open"
* g5: "When the SL\_Input is True and the SH\_Input is False, the Pump and Valve stay in their previous state"
* g6: "Pump State is 0 (off) or 1 (on)"
* g7: "Valve State is 0 (closed) or 1 (open)"

## Controller for Tank 2

### Assumptions

* a1: "If the high sensor is true, then the mid sensor is True" :
* a2: "If the mid sensor is true, then the low sensor is True"
* a3: "SH\_Input is 0 (false) or 1(true)"
* a4: "SM\_Input is 0 (false) or 1(true)"
* a5: "SL\_Input is 0 (false) or 1(true)"
* a8: "The sensors are all initially false (The tanks are initially empty)”

### Properties/Guarantees (Proof Objectives)

* g1: "The output emergency valve is initially closed"
* g2: "The output production valve is initially closed"
* g3: "When SL\_Input is False, the valves shall be Closed"
* g4: "When SM\_Input is False and the SL\_Input is true, the production valve shall be open and the emergency valve shall be closed"
* g5: "When the SH\_Input is true, both valves will be open"
* g6: "When the SM\_Input is True and the SH\_Input is False and the emergency valve was previously open, the valves shall be Open"
* g7: "When the SM\_Input is True and the SH\_Input is False and the emergency valve was previously closed, the production valve shall be open and the emergency valve shall be closed"
* g8: "Production Valve State is 0 (closed) or 1 (open)"
* g9: "Emergency Valve State is 0 (closed) or 1 (open)"

## Sensor

### Assumptions

* a1: "Initially the sensor height is greater than the current height"
* a2: "the liquid height is non-negative":
* a3: "the sensor height is non-negative":

### Properties/Guarantees (Proof Objectives)

* g1: "If the tank height is greater than or equal to the sensor height then Sensor\_Value is true"
* g2: "If the tank height is less than the sensor height then Sensor\_Value is false"
* g3: "The sensor should be false initially"
* g4: "The sensor\_height is the same as the specified sensor height"

## Tank 1

### Assumptions

* a1: "Pump flow rate is positive"
* a2: "Pump flow rate is initially 0"
* a3: "Tank cross section area is positive"
* a4: "Tank cross section area is constant"
* a5: "Valve flow rate is positive"
* a6: "Tank volume is positive"
* a7: "Tank volume is constant"
* a8: "Initial Height of the liquid is 0.0"
* a9: “valve\_flow\_rate is 0 or max flow rate”
* a10: “pump flow rate is 0 or max flow rate”

### Properties/Guarantees (Proof Objectives)

* g1: "The initial volume of tank 1 should be equivalent to the specified initial tank 1 volume"
* g2: "If the current tank 1 volume plus the tank 1 net flow is greater than the tank 1 maximum volume then tank 1 is full"
* g3: "If the current tank 1 volume plus the tank 1 net flow is less than the tank 1 minimum volume then tank 1 is empty"
* g4: "If the current tank 1 volume plus the tank 1 net flow is greater than empty and less than the tank 1 maximum volume then the tank 1 volume changes by the tank 1 net flow":
* g5: "Current height of tank 1 is equal or less than the max tank 1 height"
* g6: "Current height of tank 1 is not negative"

## Tank 2

### Assumptions

* a1: "Valve flow rate is positive"
* a2: "Tank cross section area is positive"
* a3 "Tank cross section area is constant"
* a4: "P valve flow rate is positive"
* a5: "E valve flow rate is positive"
* a6: "Tank volume is positive"
* a7: "Tank volume is constant"
* a8: "Initial Height of the liquid is 0.0"
* a9: "valve\_flow\_rate is 0 or max flow rate"
* a10: "production valve flow rate is 0 or max flow rate"
* a11: "emergency valve flow rate is 0 or max flow rate"

### Properties/Guarantees (Proof Objectives)

* g1: "The initial volume of tank 2 should be equivalent to the specified initial tank 2 volume"
* g2: "If the current tank 2 volume plus the tank 2 net flow is greater than the tank 2 maximum volume then tank 2 is full"
* g3: "If the current tank 2 volume plus the tank 2 net flow is less than the tank 2 minimum volume then tank 2 is empty"
* g4: "If the current tank 2 volume plus the tank 2 net flow is greater than empty and less than the tank 2 maximum volume then the tank 2 volume changes by the tank 2 net flow"
* g5: "Current height of tank 2 is equal or less than the max tank 2 height"
* g6: "Current height of tank 2 is not negative"

# Verification

Prior to conducting verification, the verification file should be run and no errors or warnings should appear. Once the verification model has been run, the results can be previewed using a scope block, however the scope may not provide a complete picture. To run the analysis, follow the menus in the model window as shown in Figure 18, finally selecting “Model.”

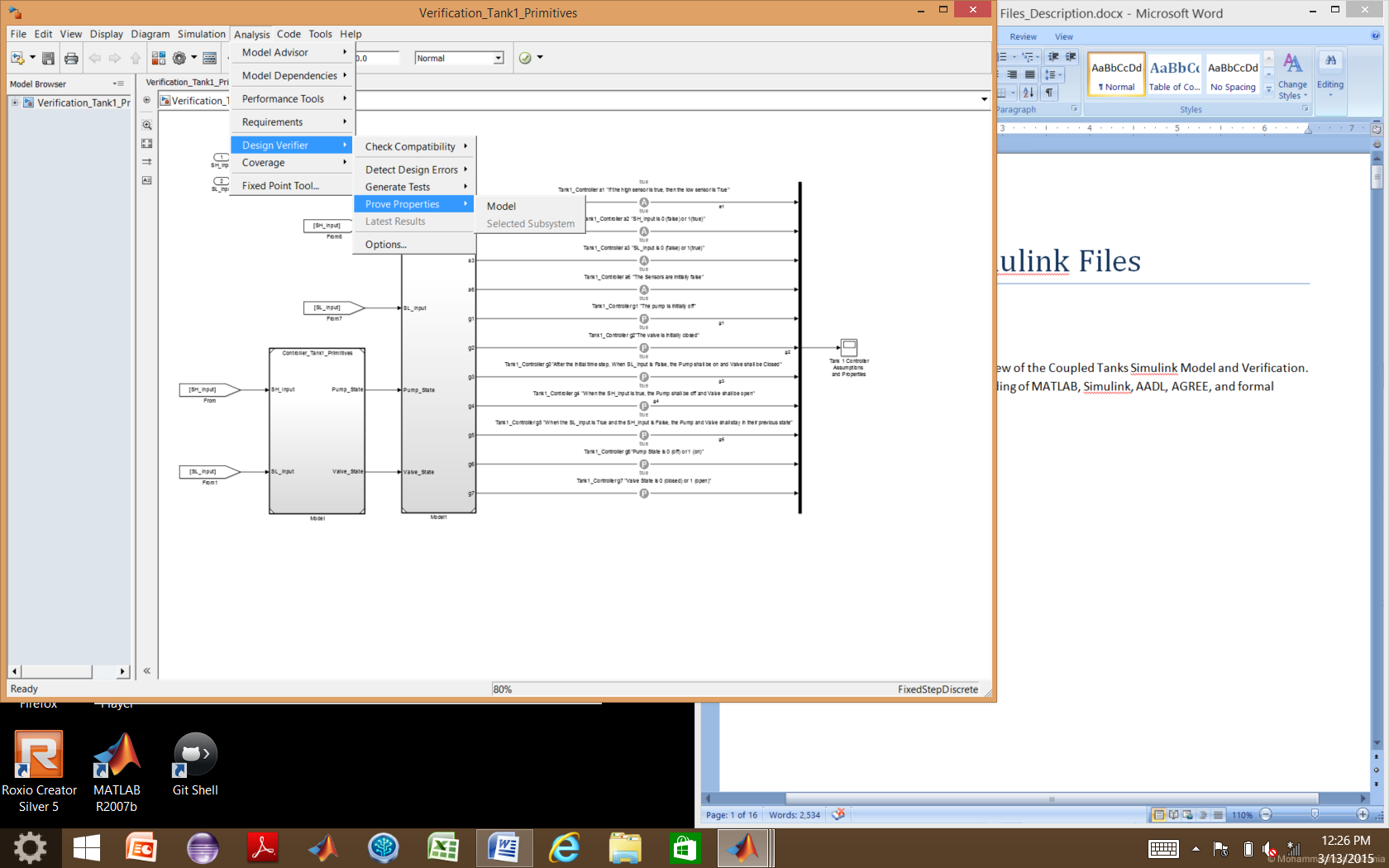


Figure 18: Design Verifier Menu Navigation for Analysis

Once you select “Model,” the Log Window shown in Figure 19 will appear, showing the progress of the analysis. In this example, all properties of Tank 1 are satisfied (proven). In the case that a property is falsified, Simulink Design Verifier will usually generate a counter example, which aids you in tracking down how the property is violated. Clicking on the “Generate detailed analysis report,” creates an HTML file which houses more information on the analysis such as counter examples for falsified properties.

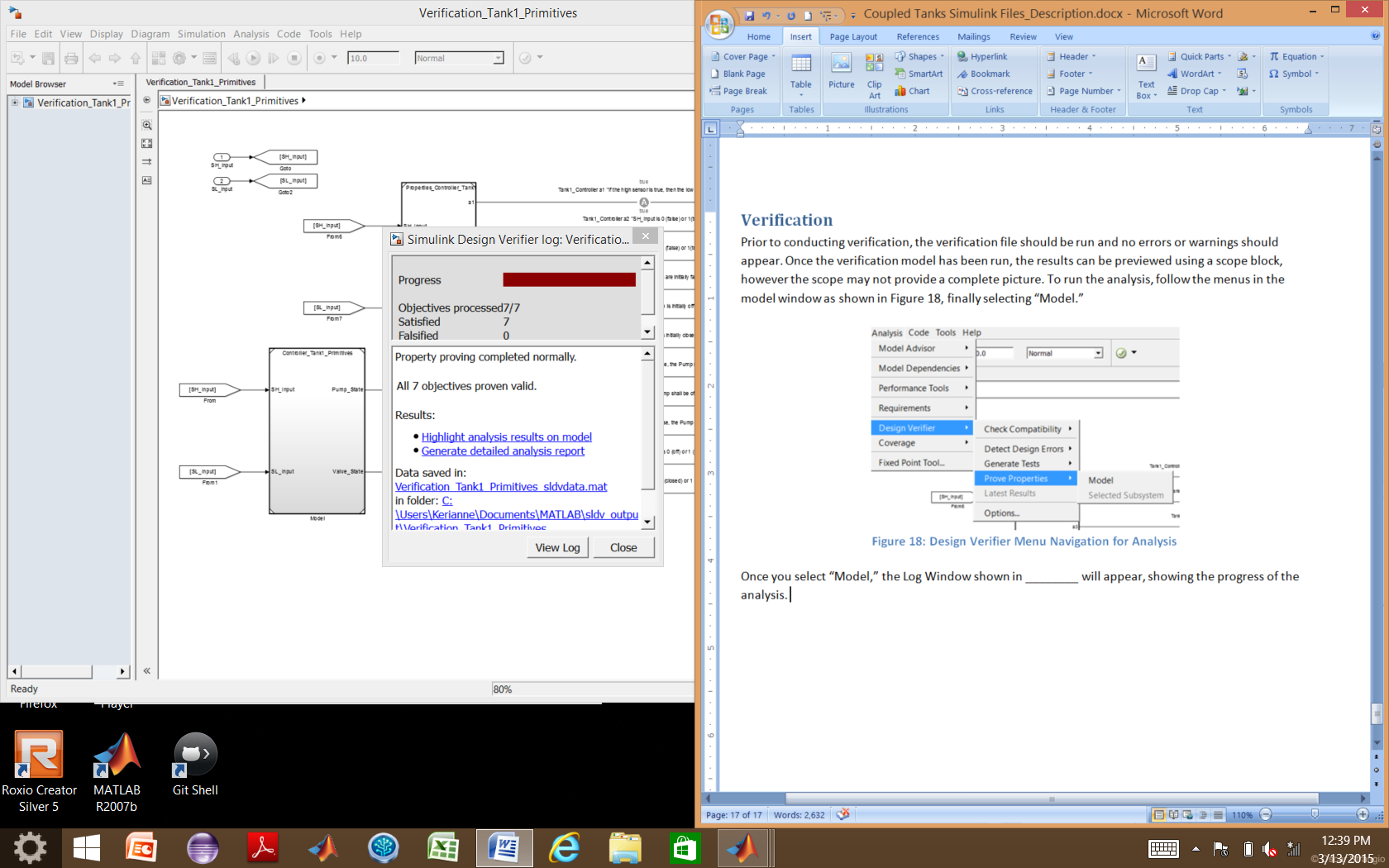


Figure 19: Simulink Design Verifier Log Window