An Introduction to Writing S-Functions

University of Michigan
EECS 461

Overview

MATLAB S-functions are an effective way to embed object code into a Simulink model. These S functions can be written in a few languages, C being the one most relevant for our purposes. This tutorial is meant to serve as a guide – almost a walk-through – that explains what all you need to know regarding S functions and how to use them in your final project Simulink models.

In lab 1, we wrote a simple adder program. We followed this up in lab 8 where we used Simulink blocks to represent the Digital Input and Output blocks and to shift and add integers. We will now combine concepts from labs 1 and 8 and utilize the best of both worlds. Understanding how and when to use S-functions in your final project is almost guaranteed to make your project significantly more understandable to both you and the GSI who helps you debug your project.

Tutorial

1. The first step is to make sure you are in the correct directory.

   Make sure you are working from the H drive. Create a folder called Adder anywhere on your H drive.

2. The second step is to select a C compiler in MATLAB.

   <u>Follow the steps below if you are using a CAEN computer</u>
   1. At the MATLAB prompt, enter "mex -setup"
   2. Select "Microsoft Visual C++ 2015 Professional (C)" as the compiler.

   This Microsoft compiler cannot generate object code for the MPC5643L. However, the compiler should enable you to complete most of your debugging outside the EECS 461 lab.

   <u>Follow the steps below if you are using an EECS 461 lab computer</u>
   1. At the MATLAB prompt, enter "mex -setup"
   2. Select "Microsoft Windows SDK 7.1 (C)" as the compiler.

3. Create a new Simulink model. Insert 8 Constant Blocks, 2 Muxs, 1 Demux, and 5 display blocks. Insert the S-Function Builder block located under “User-Defined Functions”. Your Simulink model should look like Figure 2.

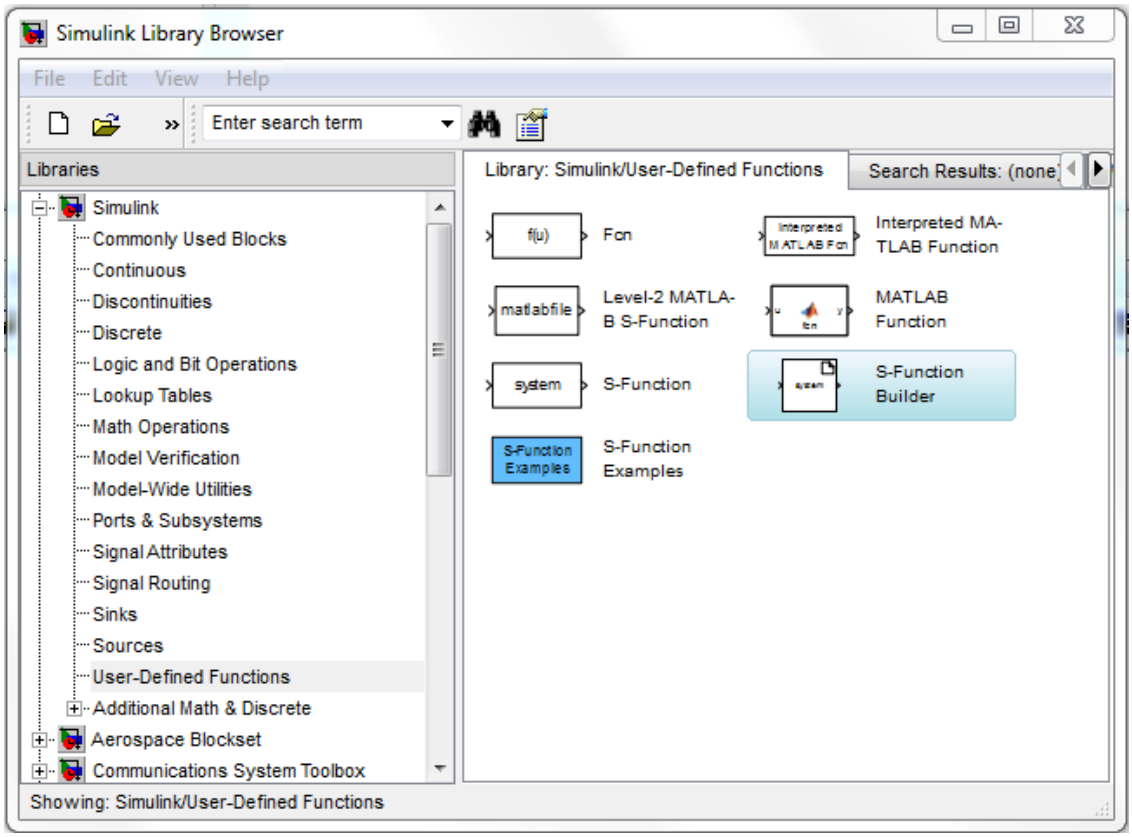

Figure 1

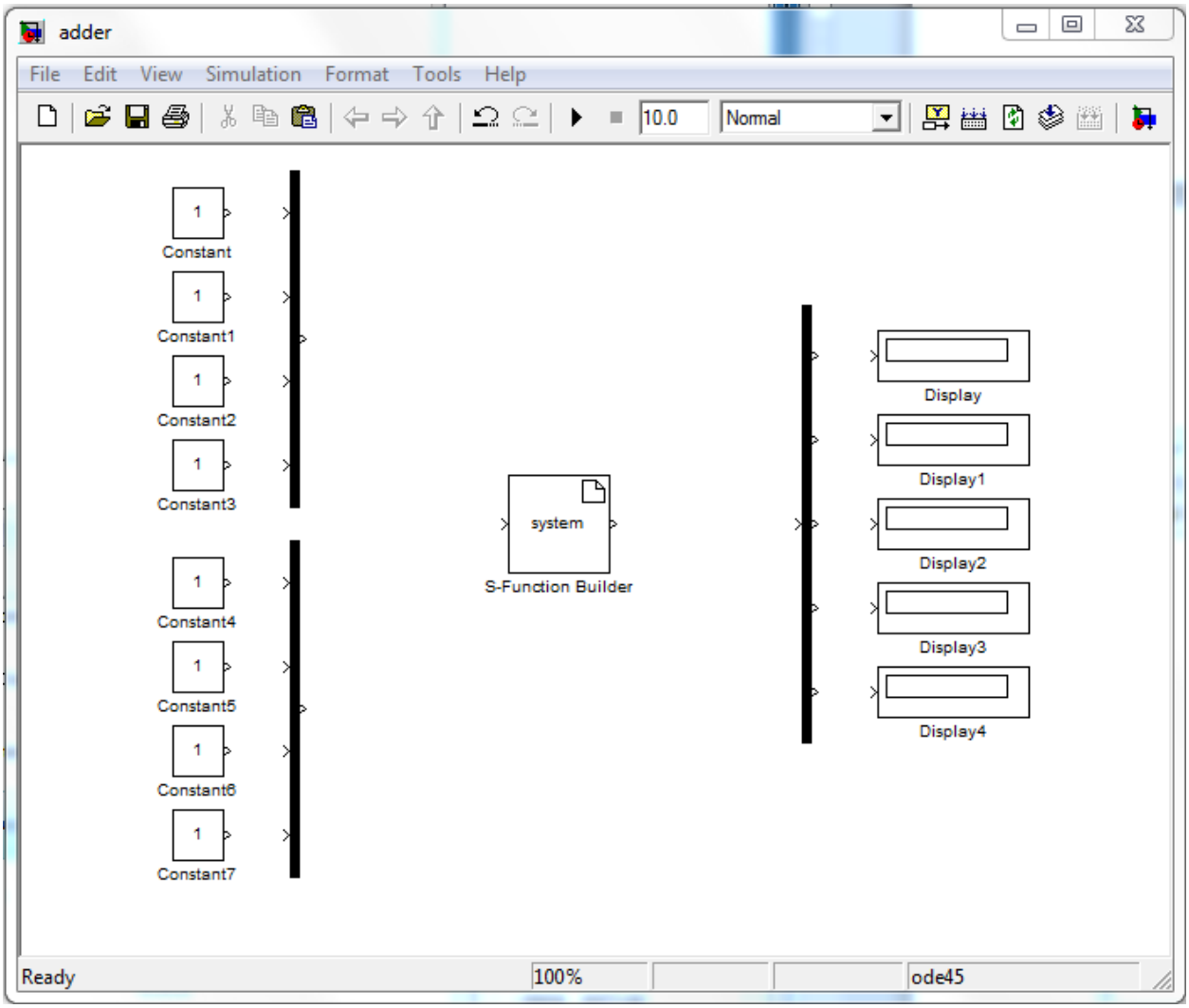
adder
File Edit View Simulation Format Tools Help
10.0
Normal
1
Constant
1
Constant1
1
Constant2
1
Constant3
1
Constant4
1
Constant5
1
Constant6
1
Constant7
system
S-Function Builder
Display
Display1
Display2
Display3
Display4
Ready
100%
ode45

Figure 2

4. Double-click on the S-Function Builder block. In the text box labeled “S-function name,” give your new S-Function a suitable name. This name has to be different from your model name.
5. Click on the “Data Properties” tab and then on the “Input ports” tab. This is where you will specify the names, types and sizes of your function’s inputs. Create two input ports, num1 and num2. These should be 1-D vectors and have 4 rows, as seen in the figure below. These settings mean that your S-Function will receive 2 vector inputs, each containing 4 elements. See figure 3 for reference.

   In the Final Project you will have many input signals to your S-functions. You may choose to group lines together and have multiple rows for each vector input.

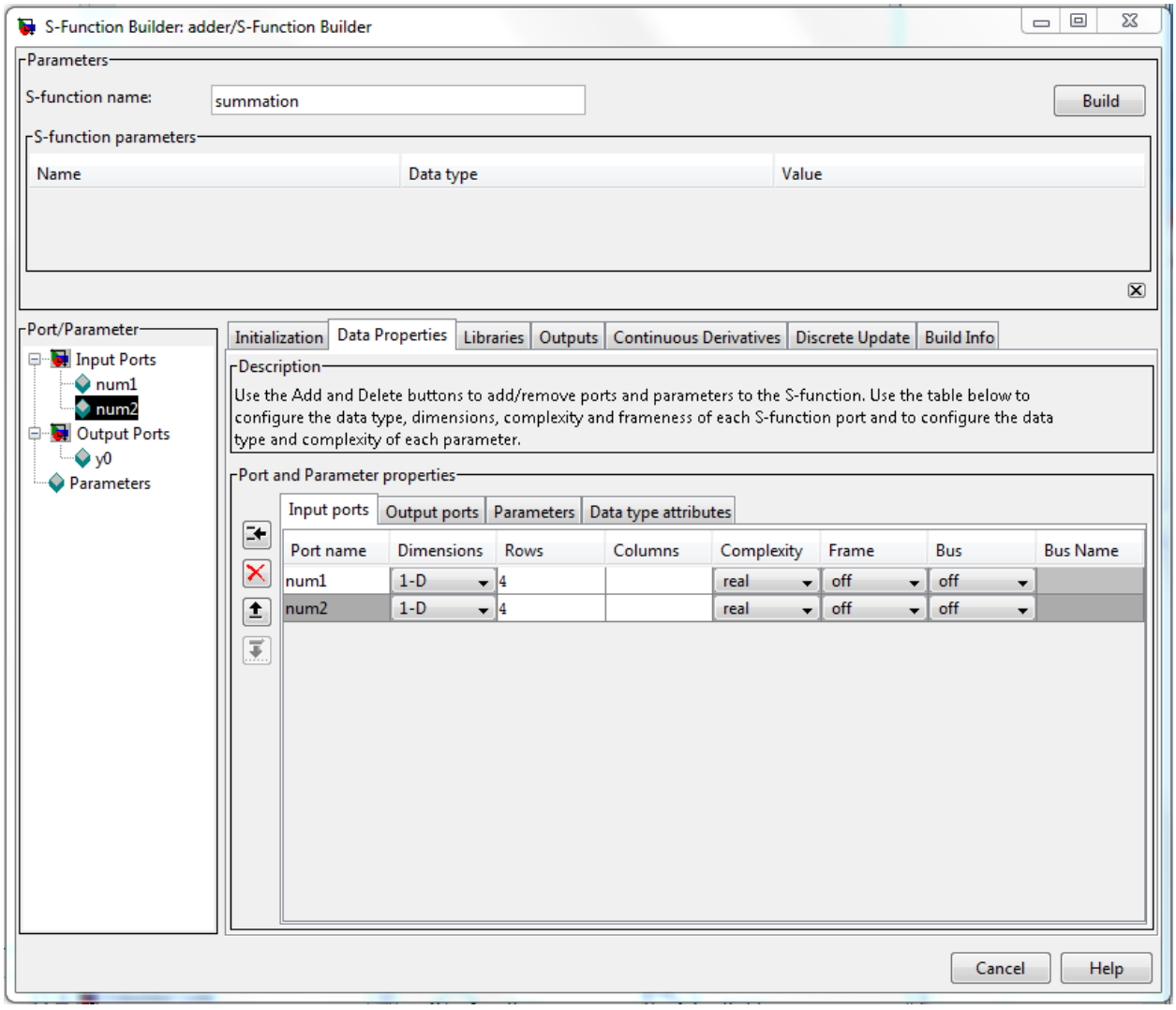

Figure 3

6. Now click on the “Output ports” tab. This is where you will specify the names, types and sizes of your function’s outputs. Create an output port, sum, a vector containing 5 elements.
7. Click on the “Data type attributes” tab and select a data type of “boolean” for inputs and outputs. See figure 4 for reference.

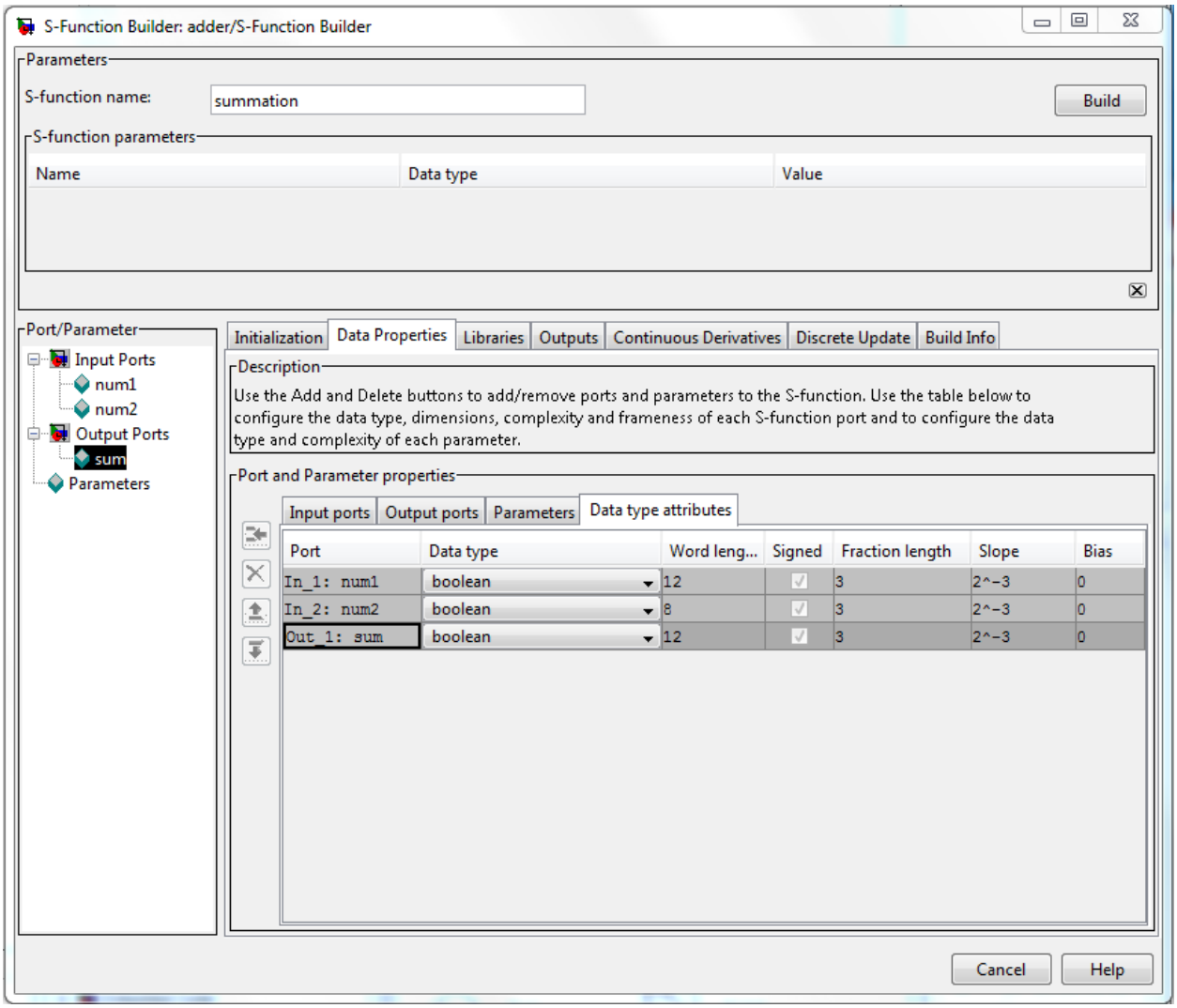

Figure 4

8. Click on the “Outputs” tab. This is where you will write the C code that connects your function’s inputs to its outputs. Each element of the input vector can be accessed by indexing into the vector the same way you would do in C. For example, if you wanted the first output element to be the logical OR of the first input element of each input, you would write:

   sum[0] = num1[0] | num2[0];

   Now implement your code for a 4 bit adder. See figure 5 for reference.

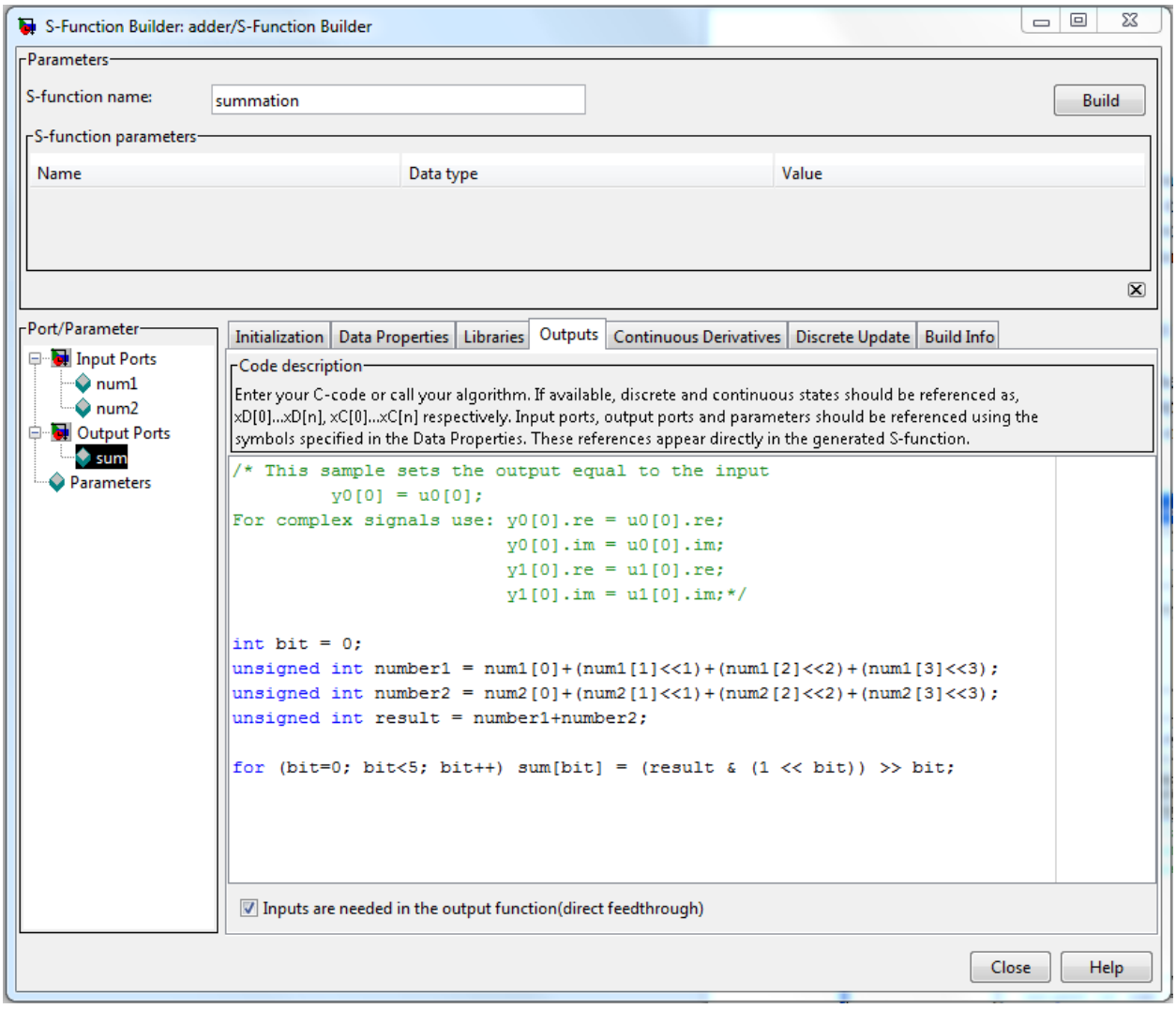

Figure 5

9. You are now ready to build your function. Click on the “Build Info” tab. Check the boxes named “Show compile steps,” “Create a debuggable MEX-file” and “Generate wrapper TLC”. Now click “Build.” If the build process was successful, your window should look similar to figure 6.

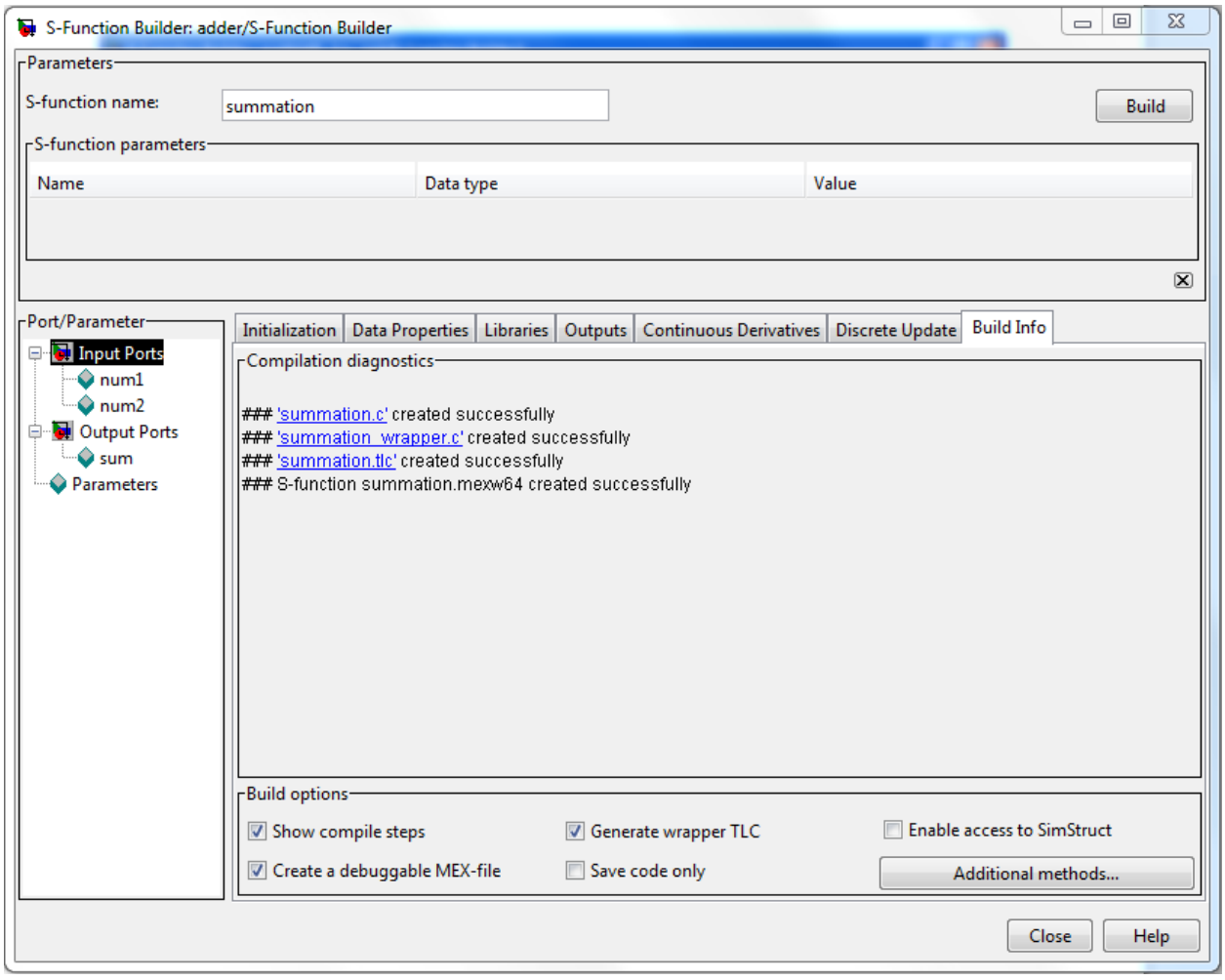

Figure 6

10. Your S-Function block should now be updated with the input and output ports. Now complete the model as shown in figure 7. You will have to change the data types of the constant blocks to "boolean". See figure 8 for reference.

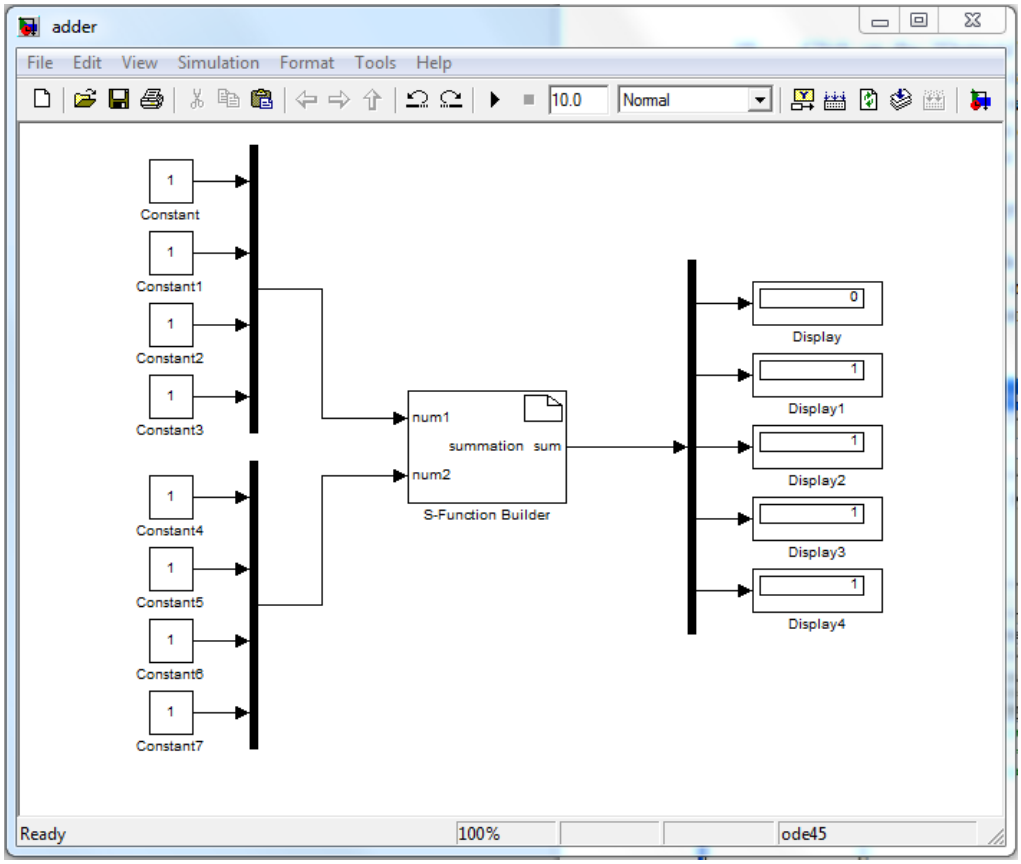

Figure 7

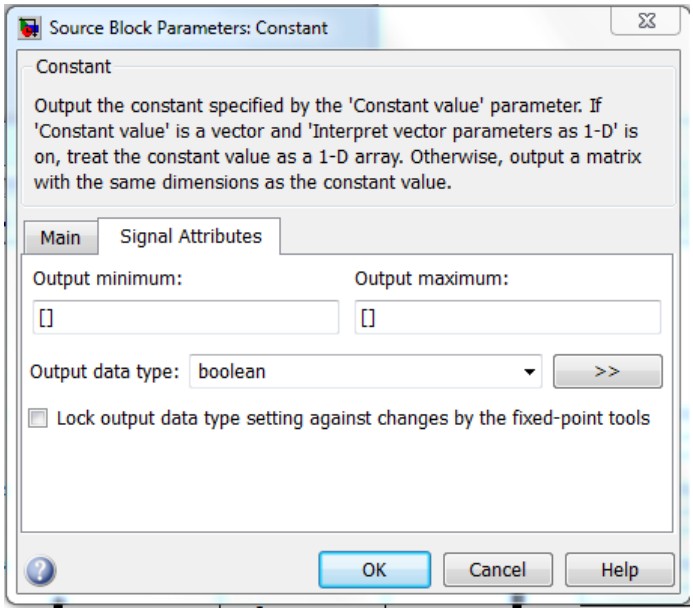

Figure 8

The preceding steps can all be done either in a CAEN lab, or in the EECS 461 lab. In order to create an S-function that may be used for autogenerate code that may be downloaded to the MPC5643L, you must be in the EECS 461 lab, and using the Windows SDK 7.1(c) compiler.

OPTIONAL: Using Motor Control toolbox

11. Rebuild your s-function in the EECS 461 lab, thus using the Microsoft Windows SDK 7.1 (C) compiler.
12. Open the .c file resulting from the S-function (called summation.c in the present example), and insert the line #include <stdbool.h>" immediately before or after the autogenerated #define lines of code, and before any other lines of code.
13. You will now incorporate the Motor Control Toolbox blocks provided on the lab computers in your model. Replace all of the Constant Blocks with Digital Input Blocks. Replace all the Display Blocks with digital Output Blocks. Select the appropriate pin for each block by double clicking on each block. Next, add the RAppID Initialization Block to your model. See Figure 9 for reference.
14. You will need to set a fixed time step. Go to Simulation and select "Configuration Parameters...". For Fixed-step size, enter ".001". Click "OK".
15. You may build your model by pressing *ctrl-B*. This will generate an executable linkable file (.elf).

The reason to include stdbool.h is either to use boolean variables in your S function, or because the S function builder uses booleans when it creates a .c file (it does this with mux blocks, for example). Please note that you need to redo this step every time to edit the S function.

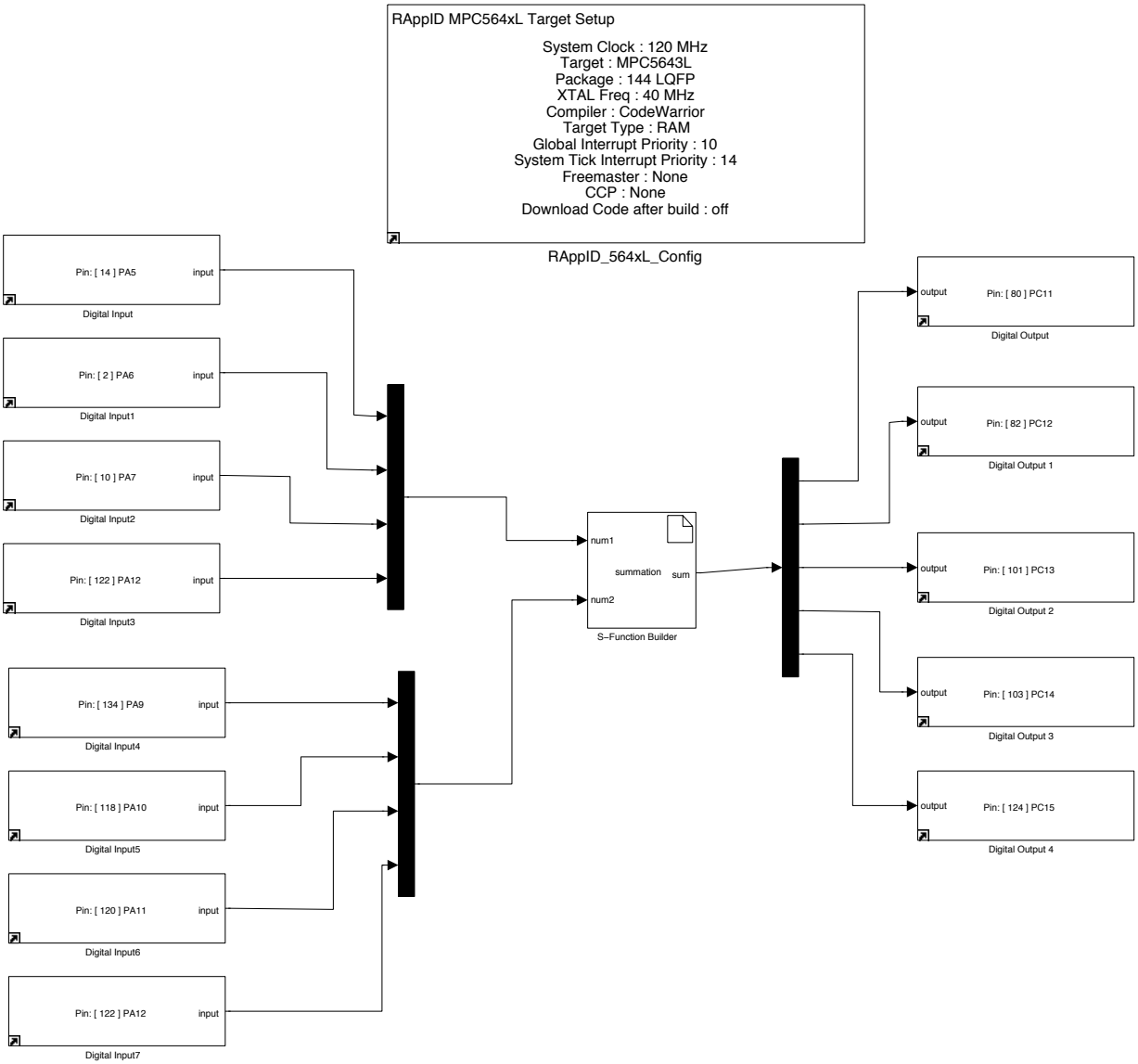
RAppID MPC564xL Target Setup
System Clock : 120 MHz
Target : MPC5643L
Package : 144 LQFP
XTAL Freq : 40 MHz
Compiler : CodeWarrior
Target Type : RAM
Global Interrupt Priority : 10
System Tick Interrupt Priority : 14
Freemaster : None
CCP : None
Download Code after build : off
RAppID_564xL_Config
Pin: [14] PA5
input
Digital Input
Pin: [2] PA6
input
Digital Input1
Pin: [10] PA7
input
Digital Input2
Pin: [122] PA12
input
Digital Input3
Pin: [134] PA9
input
Digital Input4
Pin: [118] PA10
input
Digital Input5
Pin: [120] PA11
input
Digital Input6
Pin: [122] PA12
input
Digital Input7
num1
summation
sum
num2
S-Function Builder
output
Pin: [80] PC11
Digital Output
output
Pin: [82] PC12
Digital Output 1
output
Pin: [101] PC13
Digital Output 2
output
Pin: [103] PC14
Digital Output 3
output
Pin: [124] PC15
Digital Output 4

Figure 9