

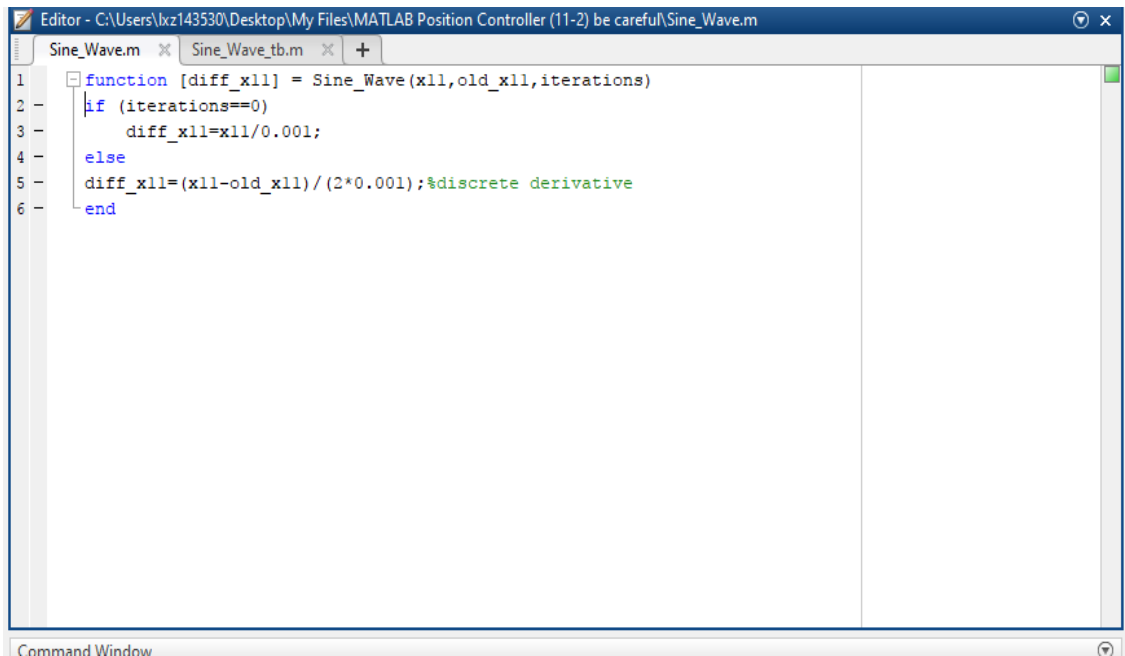
How to Implement MATLAB Code on LabVIEW

Table of Contents

Table of Contents	1
1. Create the MATLAB code and test bench	2
2. Convert MATLAB code to C using MATLAB coder	3
3. Building the Shared Object	7
4. Copying the Shared Object file to the myRIO	17
5. Implementing Shared Object file onto LabVIEW code.....	20

1. Create the MATLAB code and test bench

MATLAB code:



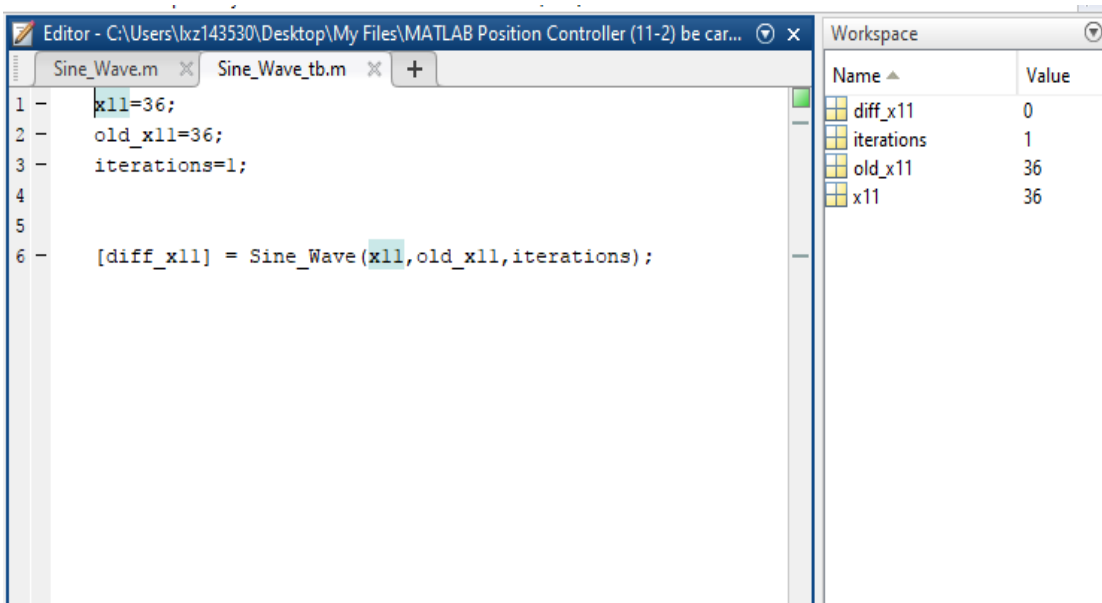
```

Editor - C:\Users\kxz143530\Desktop\My Files\MATLAB Position Controller (11-2) be careful\Sine_Wave.m
Sine_Wave.m  Sine_Wave_tb.m  +
1  function [diff_x11] = Sine_Wave(x11,old_x11,iterations)
2  if (iterations==0)
3      diff_x11=x11/0.001;
4  else
5      diff_x11=(x11-old_x11)/(2*0.001);%discrete derivative
6  end

```

Format for function: `function [output 1, output 2, ...] = name [input1, input2, ...];`

Test bench:



```

Editor - C:\Users\kxz143530\Desktop\My Files\MATLAB Position Controller (11-2) be car...
Sine_Wave.m  Sine_Wave_tb.m  +
1  x11=36;
2  old_x11=36;
3  iterations=1;
4
5
6  [diff_x11] = Sine_Wave(x11,old_x11,iterations);

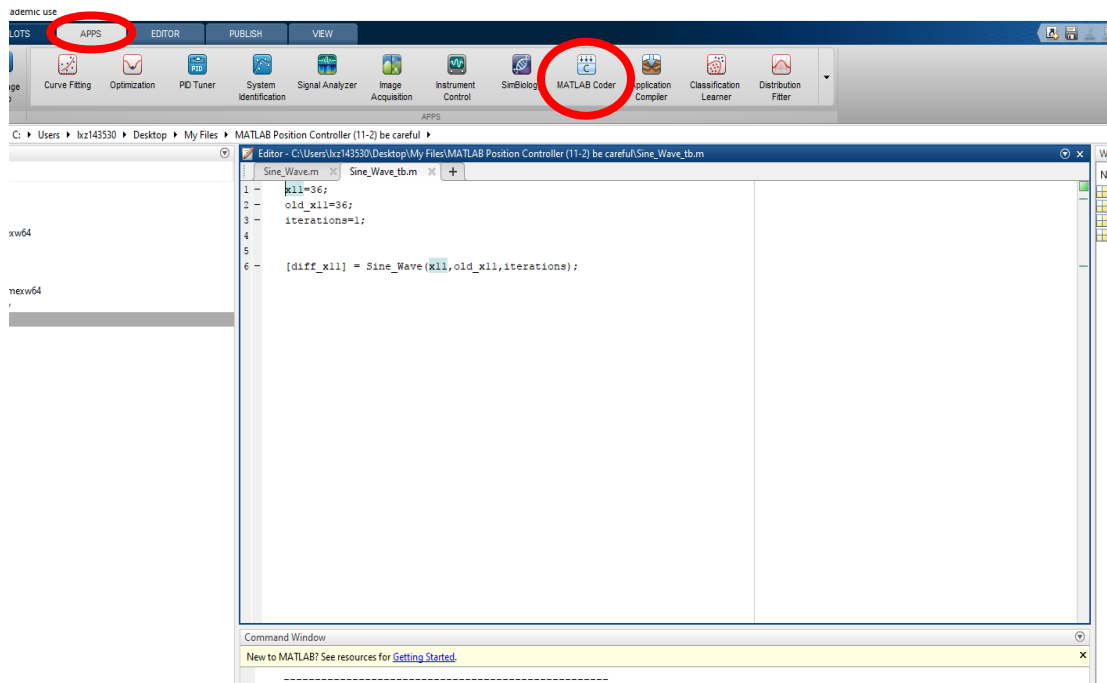
```

Name	Value
diff_x11	0
iterations	1
old_x11	36
x11	36

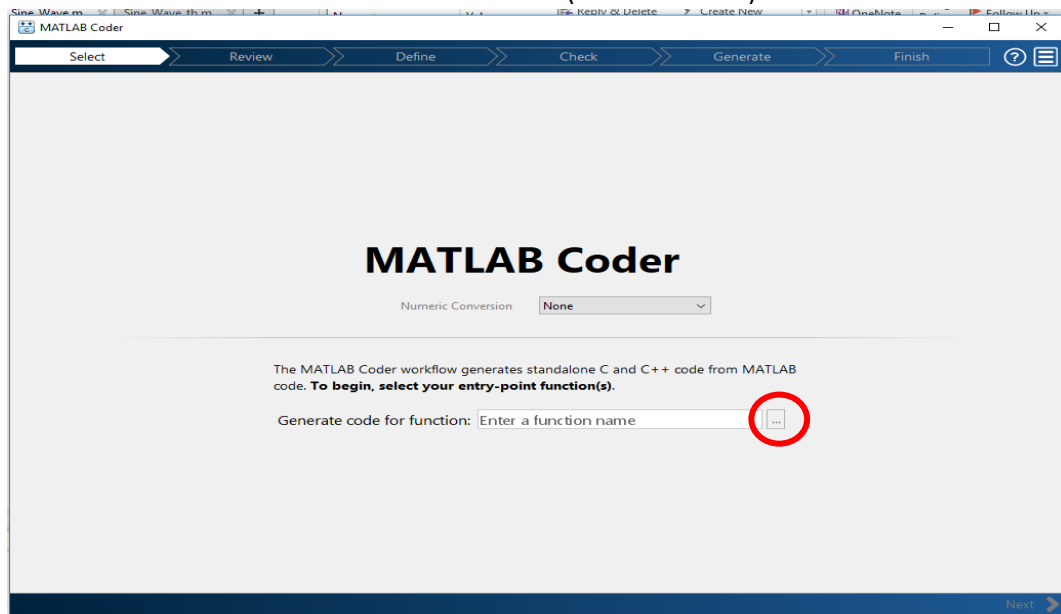
Initialize inputs (these can be chosen arbitrarily as they have no effect on output when this code runs on LabVIEW). After initialization of inputs, call the function.

2. Convert MATLAB code to C using MATLAB coder

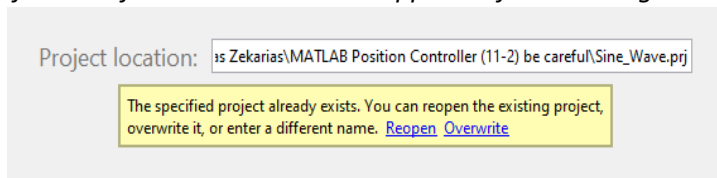
a) To convert from MATLAB to C open the MATLAB Coder under the APPS tab.



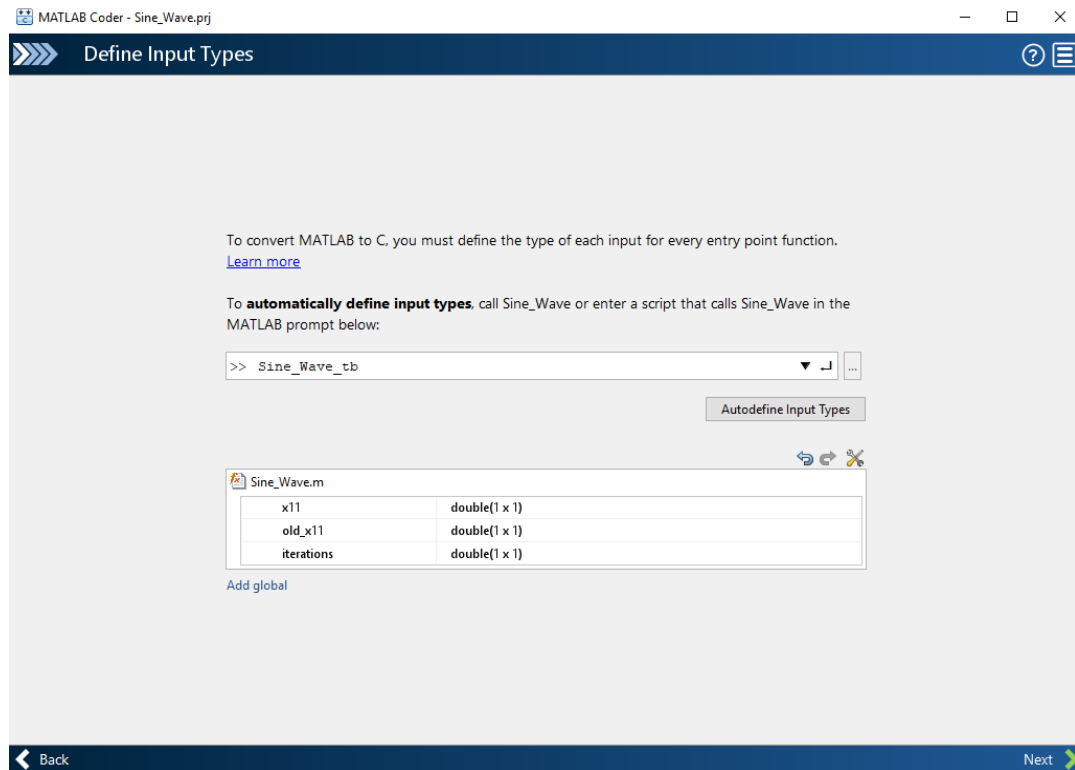
b) Click "... " and find the .m file for MATLAB code (not test bench) then click next.



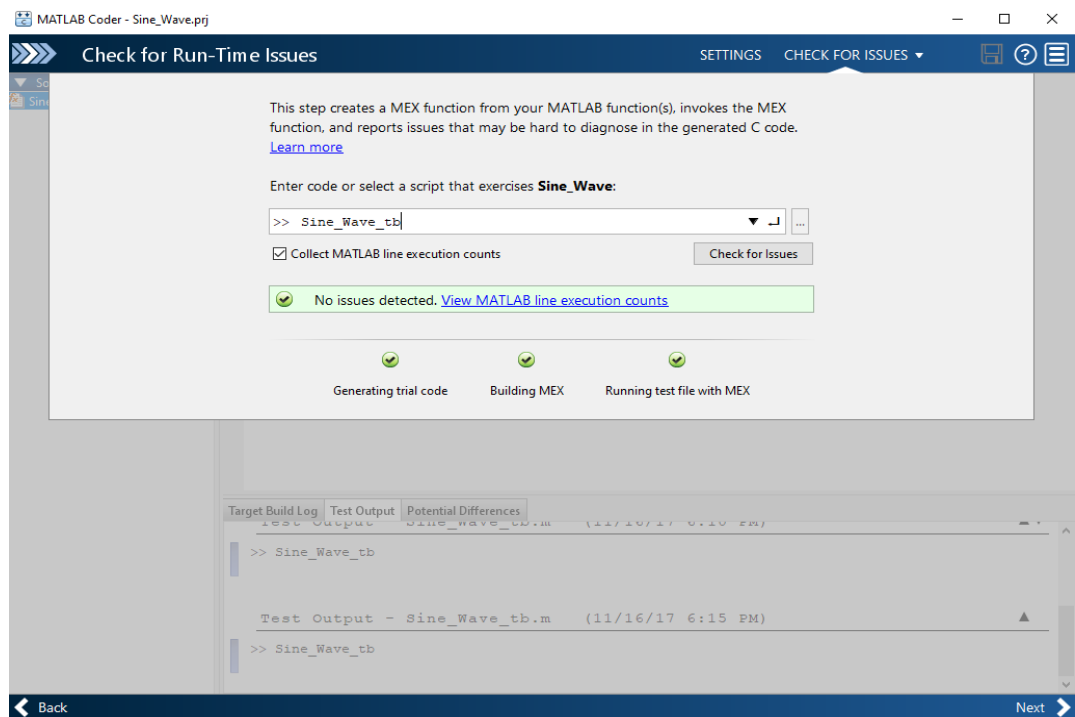
If the notification shown below appears after selecting the desired .m file, click "Overwrite".



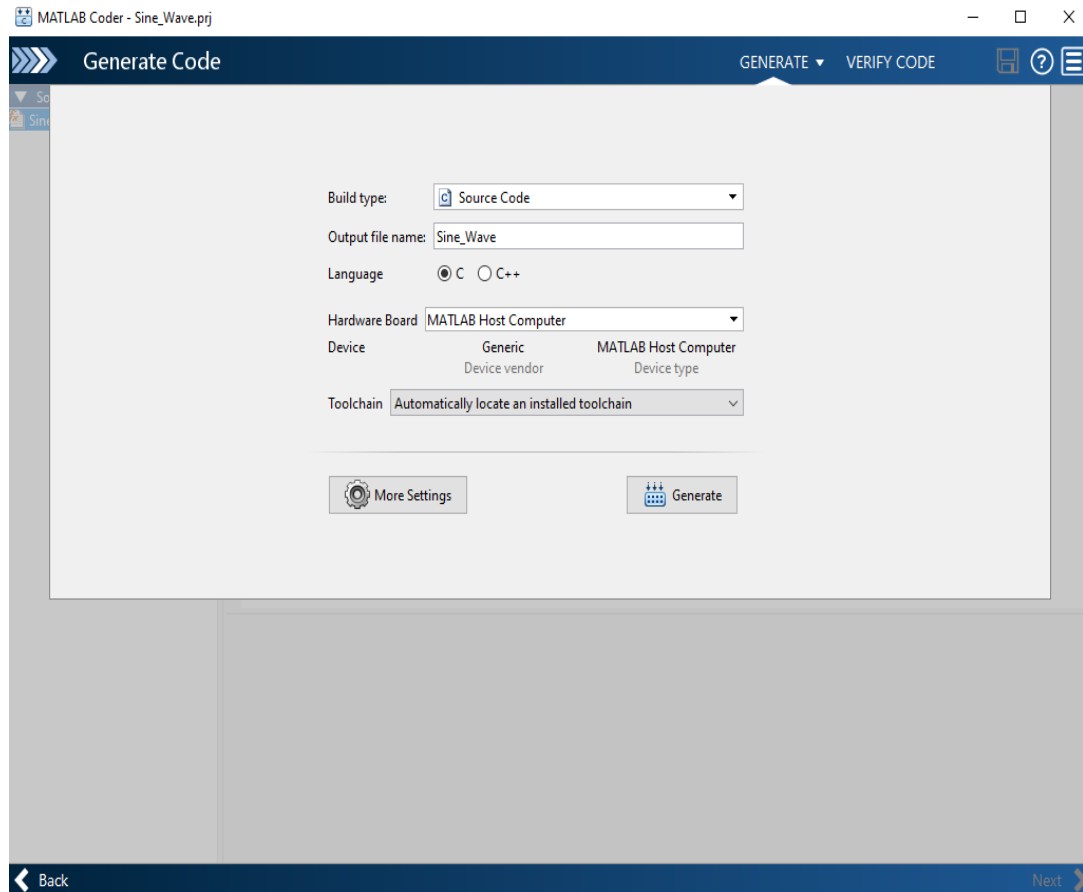
- c) Select “...” and find the test bench .m file then, select “Autodefine Input Types”. Once the input types are defined, click next.



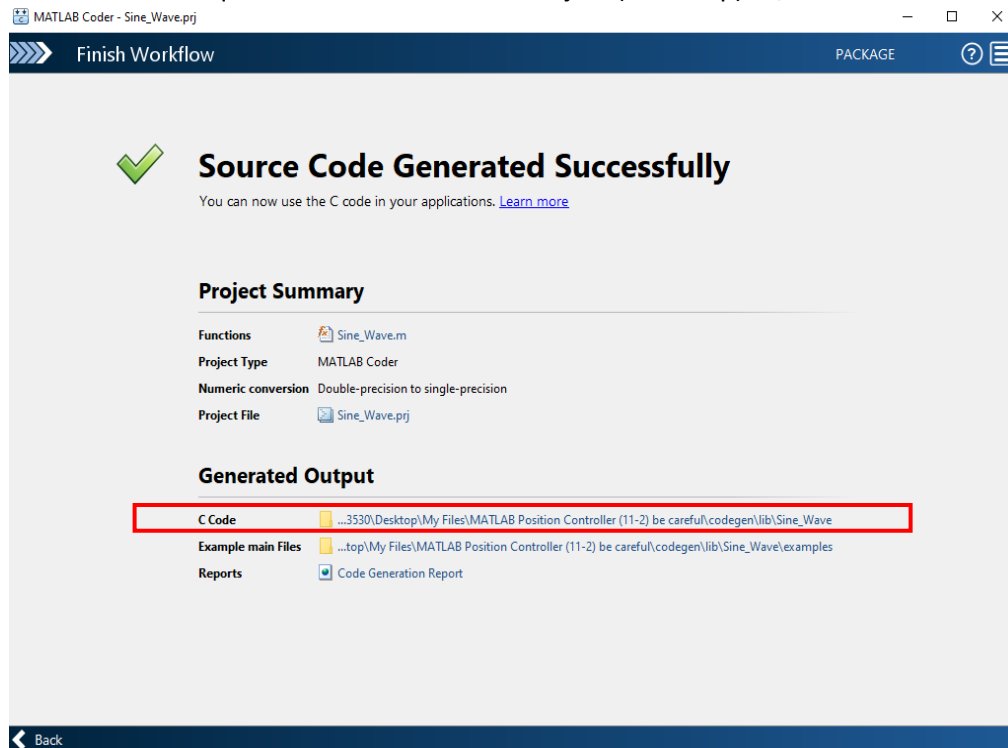
- d) Select “Check for Issues”, once it is done click next. It should look like this once it is done.



- e) Click “Generate”, once the C source code is successfully generated a notification stating “Source Code generation succeeded” on the bottom of the page will appear. After that click next.



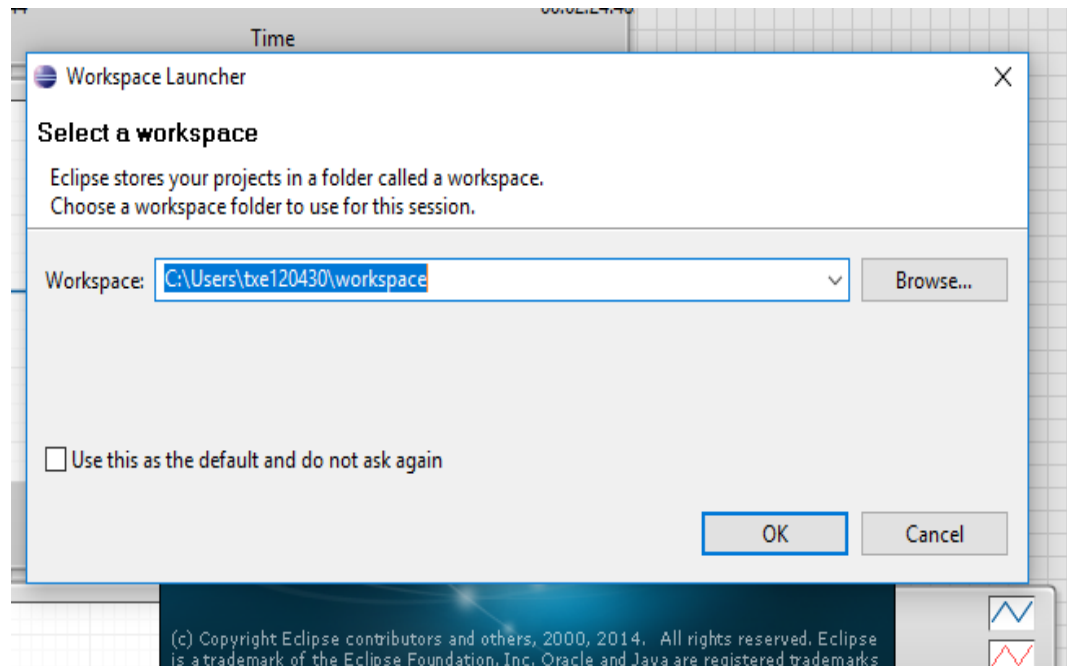
- f) To find where the relevant .c and .h files are located click on the “C Code” folder location. These files are required to build the Shared Object (next step) so, do not close this window.



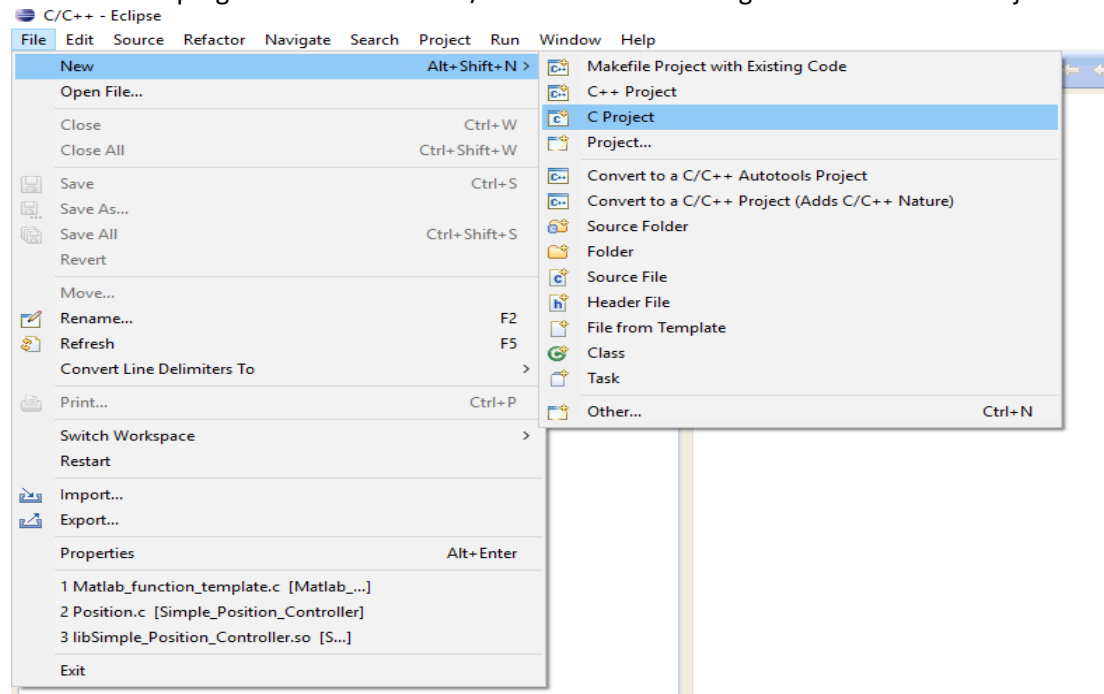
3. Building the Shared Object

This step will convert the C code to a Shared Object

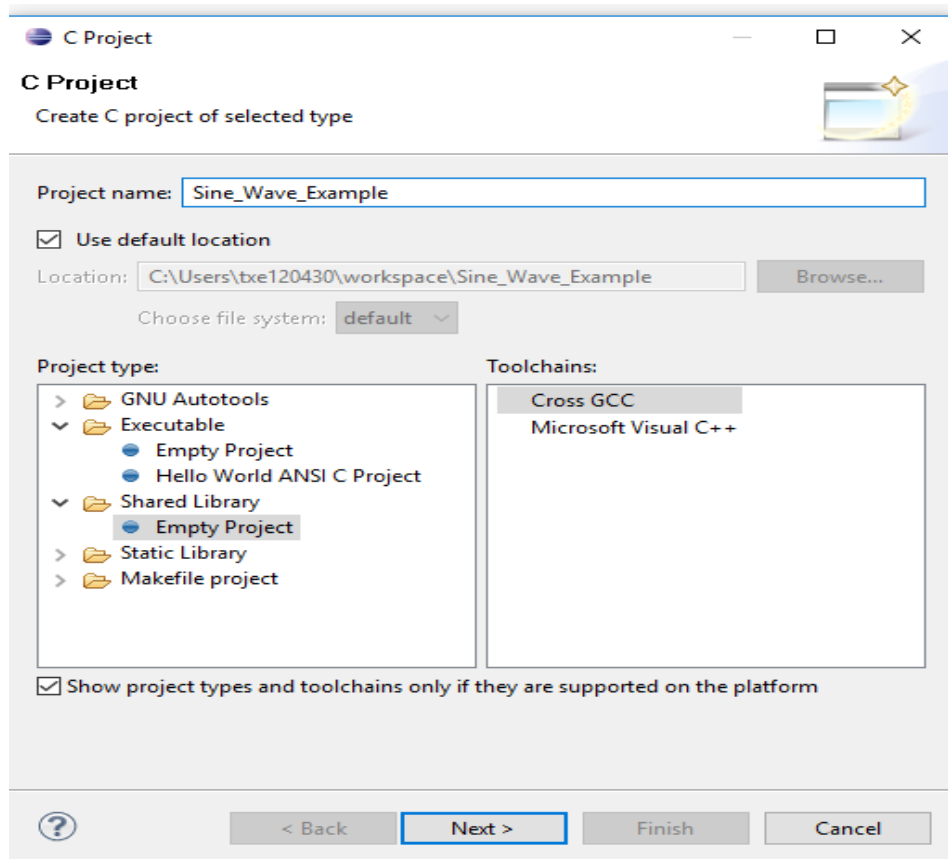
- Launch C & C++ Development Tools for NI Linux Real-Time 2014, Eclipse Edition.
- Once the program is open, this window will pop up. By default the “Workspace Launcher” will select the workspace folder under the user’s netID. Click “OK” once the desired workspace folder is selected.



- Look at the top right and make sure “C/C++” is selected then go to File->New->C Project.

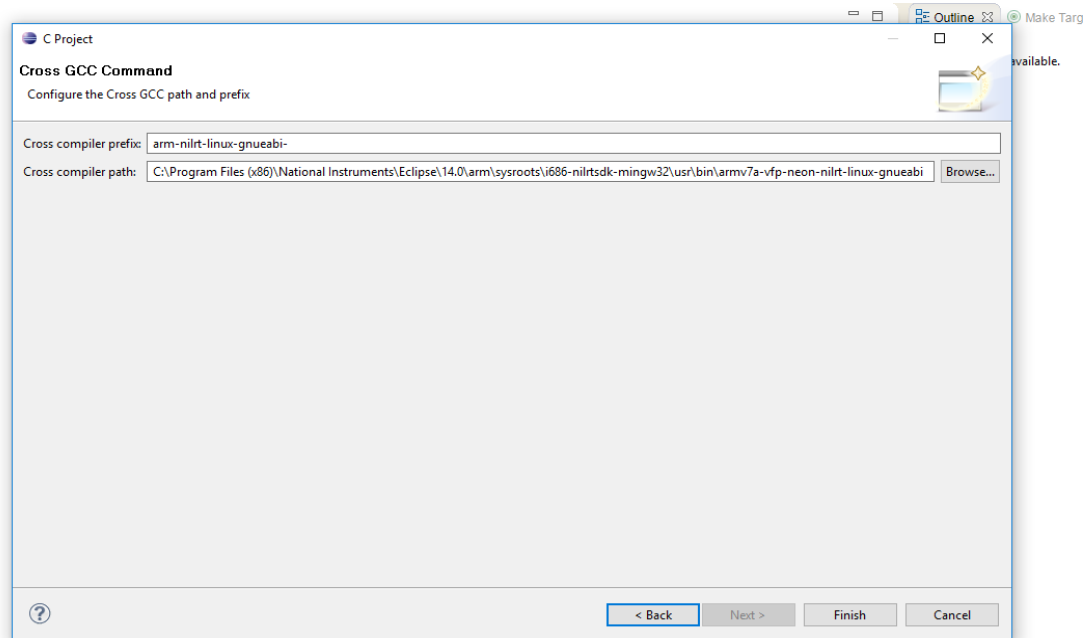


- d) Enter the desired Project name and select “Empty Project” under “Shared Library”, then click next.



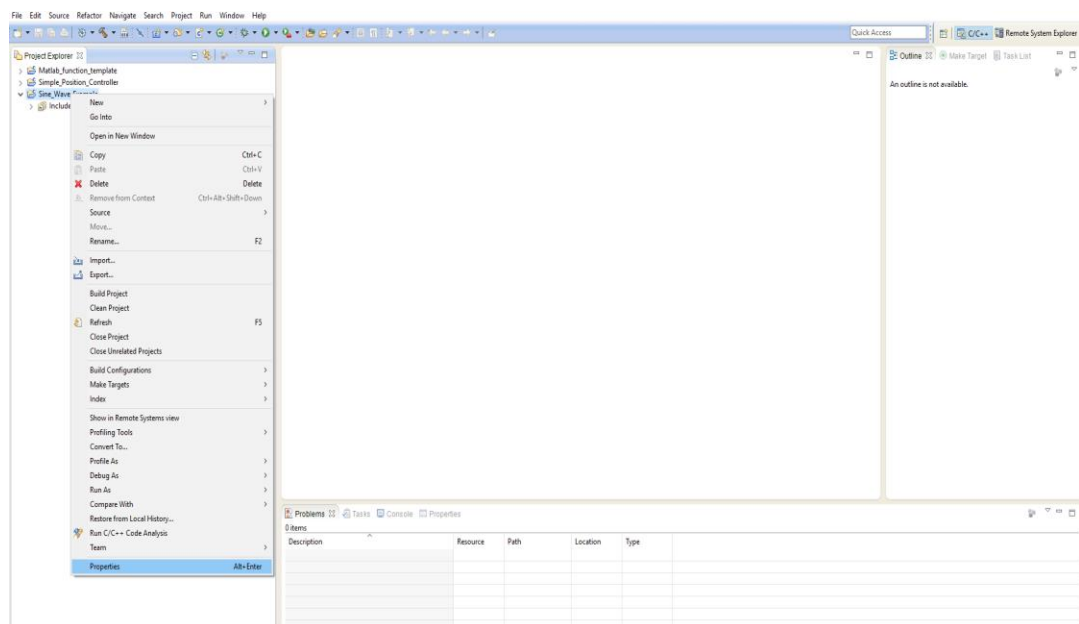
There is nothing that needs to be altered on the “Select Configurations” window so, click next again

- e) Under “Cross compiler prefix” enter: arm-nilrt-linux-gnueabi-
 Under “Cross compiler path” enter:
 C:\Program Files (x86)\National Instruments\Eclipse\14.0\arm\sysroots\i686-nilrtsdk-mingw32\usr\bin\armv7a-vfp-neon-nilrt-linux-gnueabi
 Then click finish.

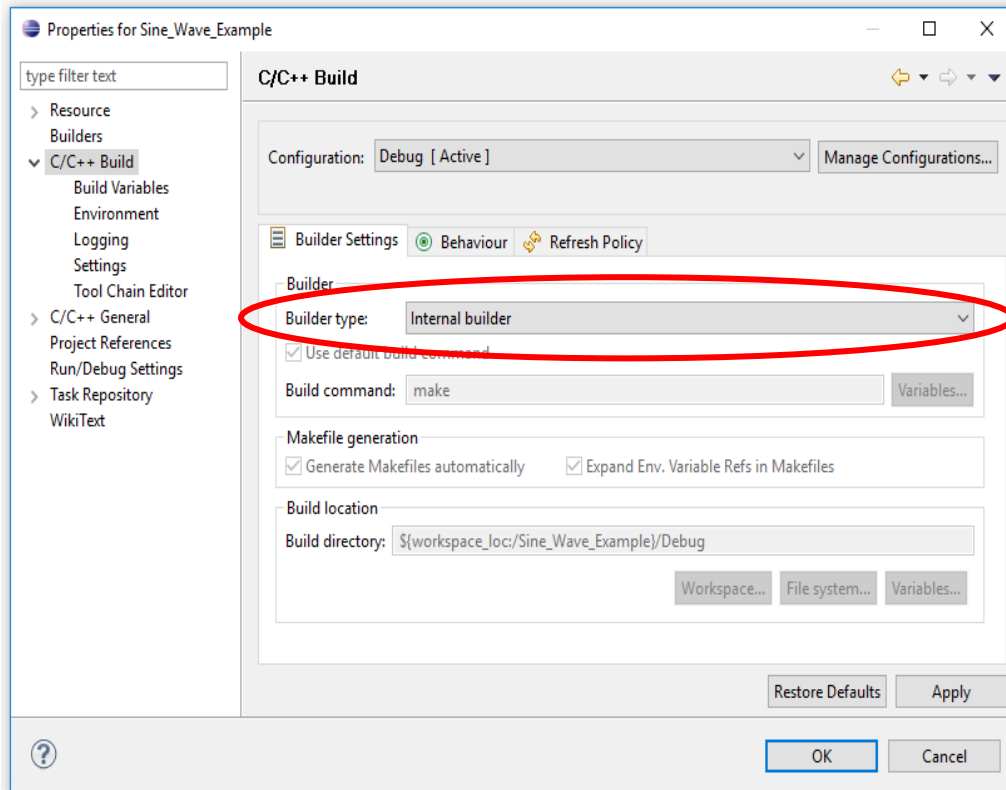


onsole Properties

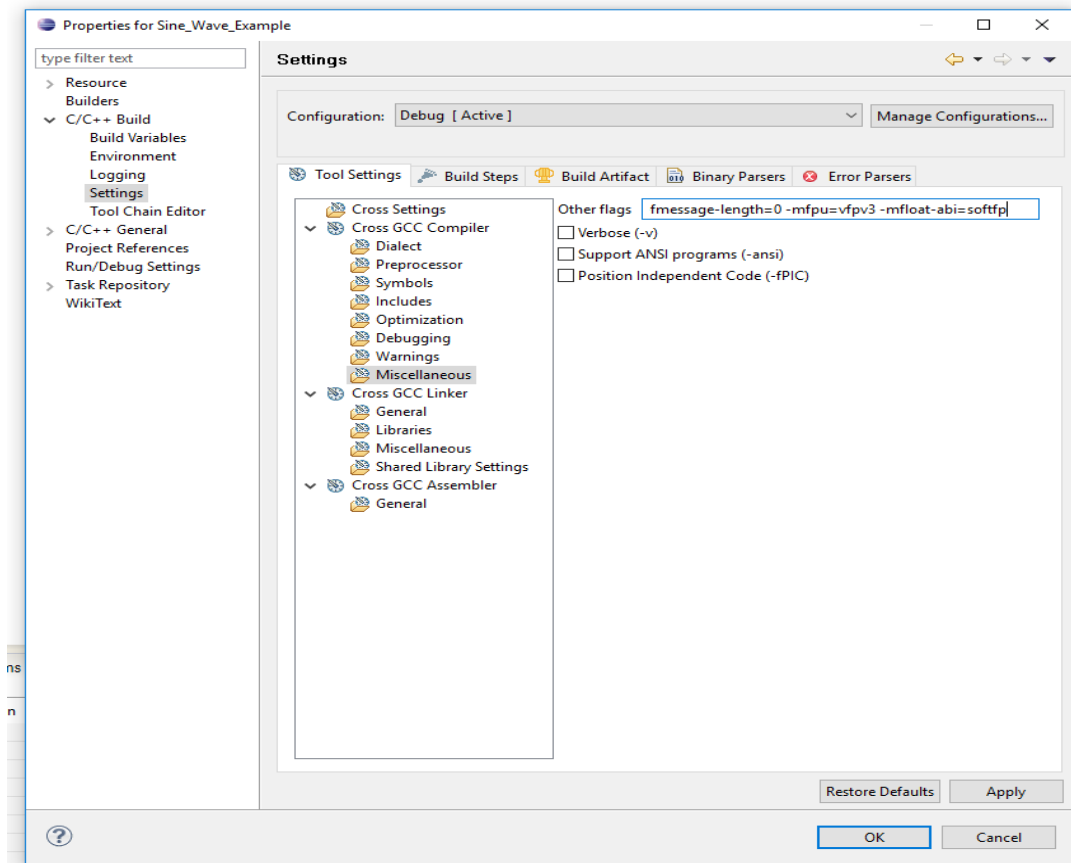
- f) Now right click the project folder under “Project Explorer” and click on “Properties”.



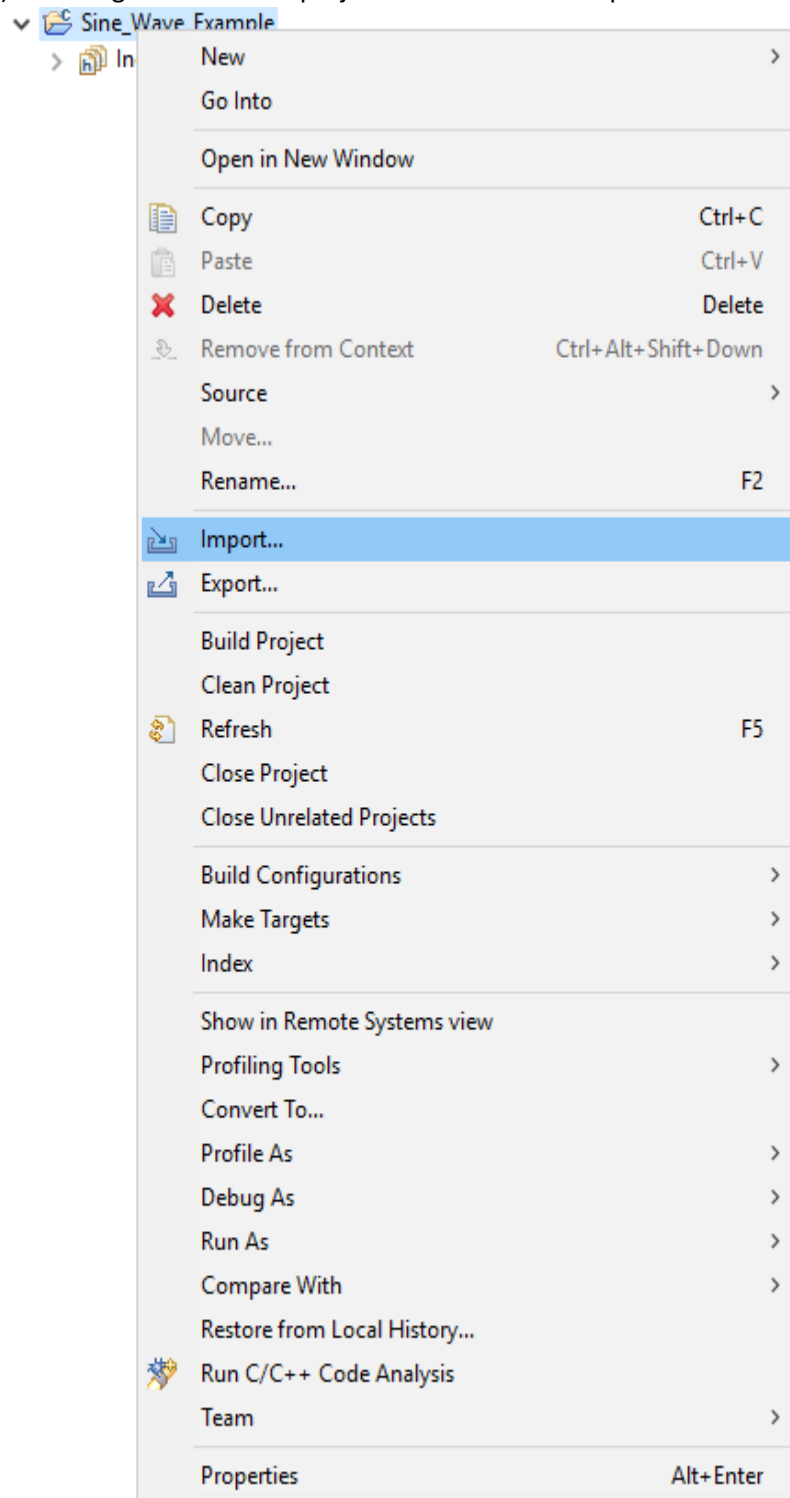
g) Go to C/C++ Build and change the “Builder type” to internal builder, then click apply.



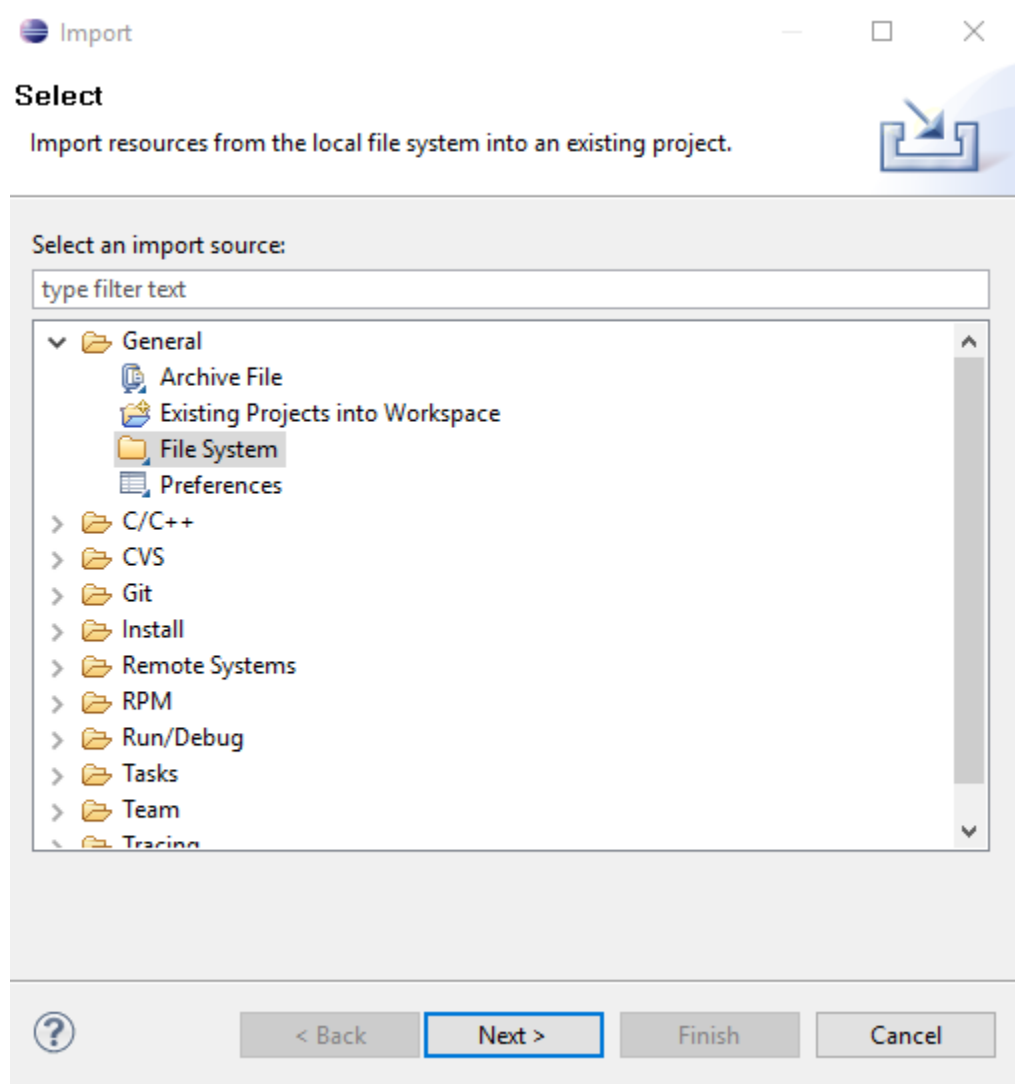
- h) Now go to “Settings” under “C/C++ Build” then go to “Miscellaneous” under “Cross GCC Compiler”. In the text box next to “Other flags” after `-c -fmessage-length=0` add a space and enter: `-mfpu=vfpv3 -mfloat-abi=softfp`. Then click OK.



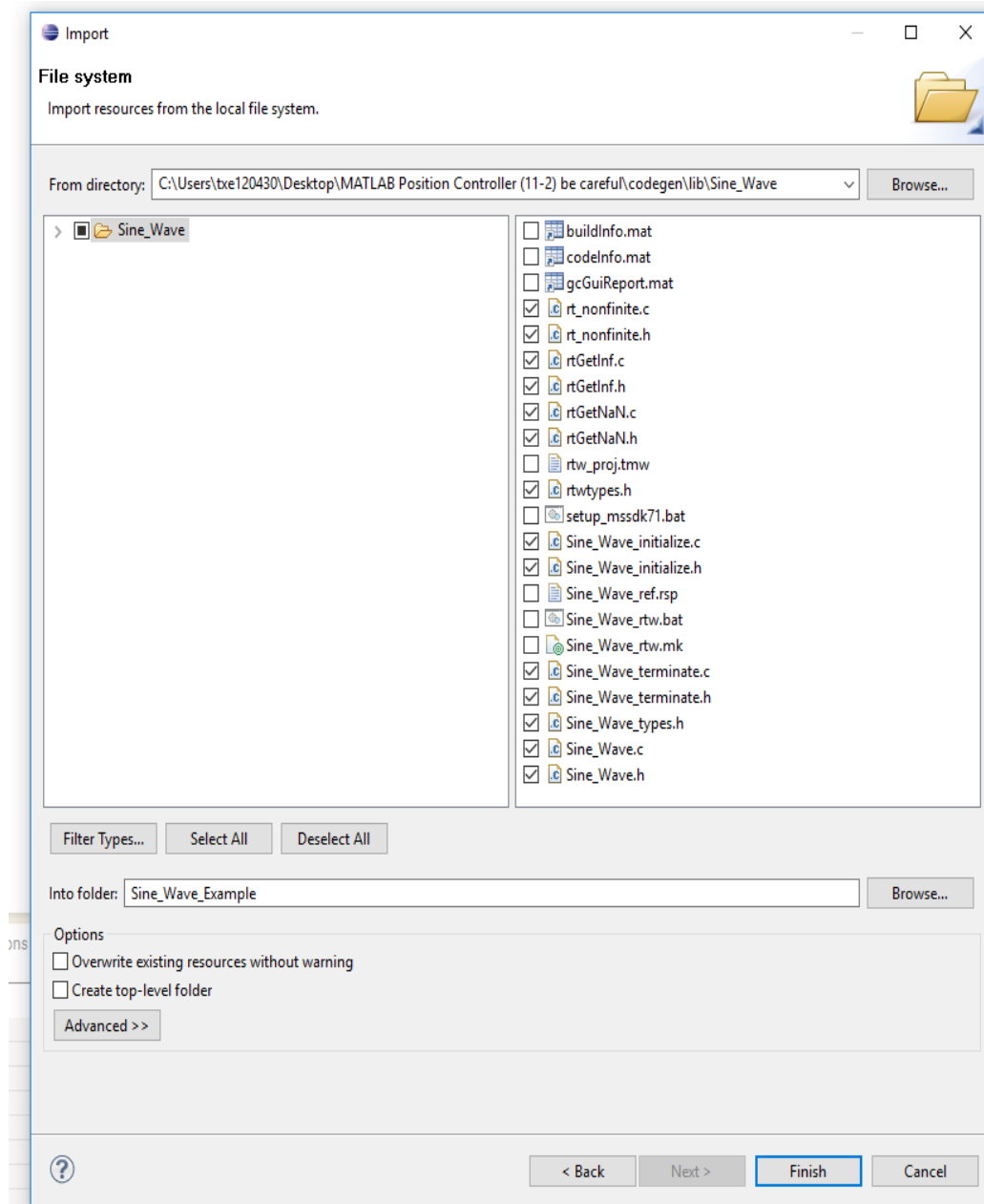
l) Right click on the project folder and click import.



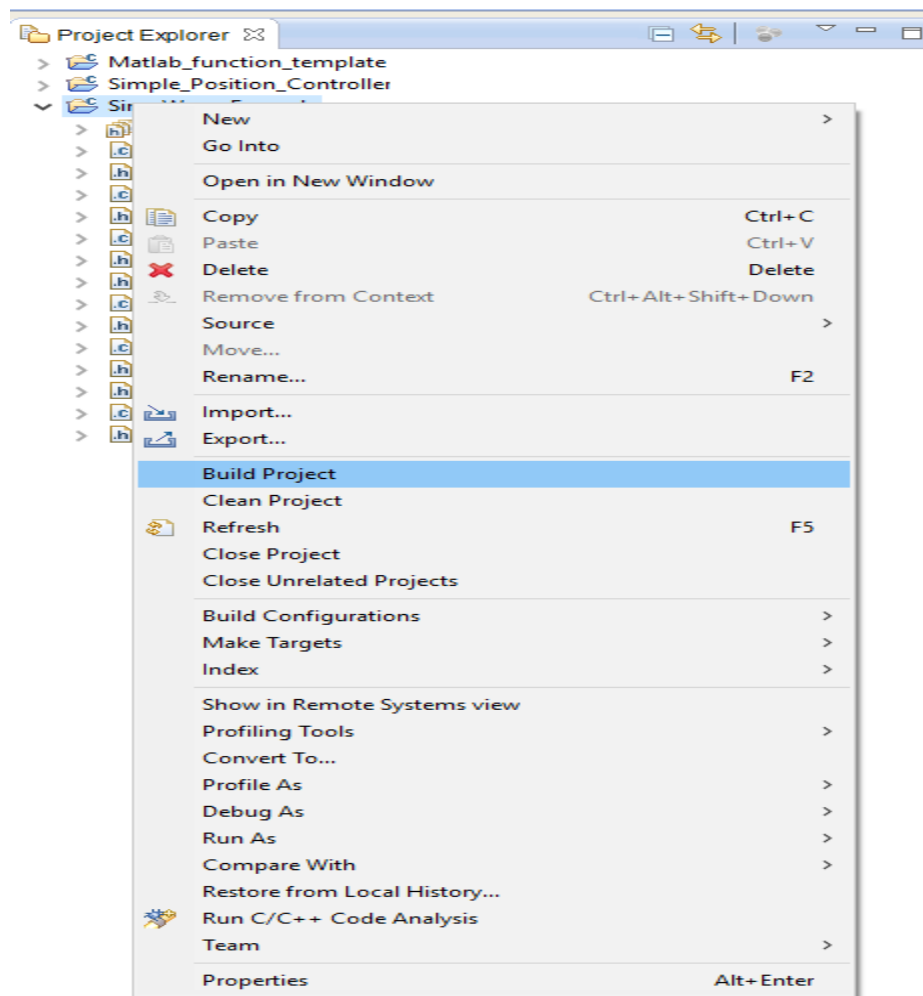
- j) Select "File System" under "General", then click next.



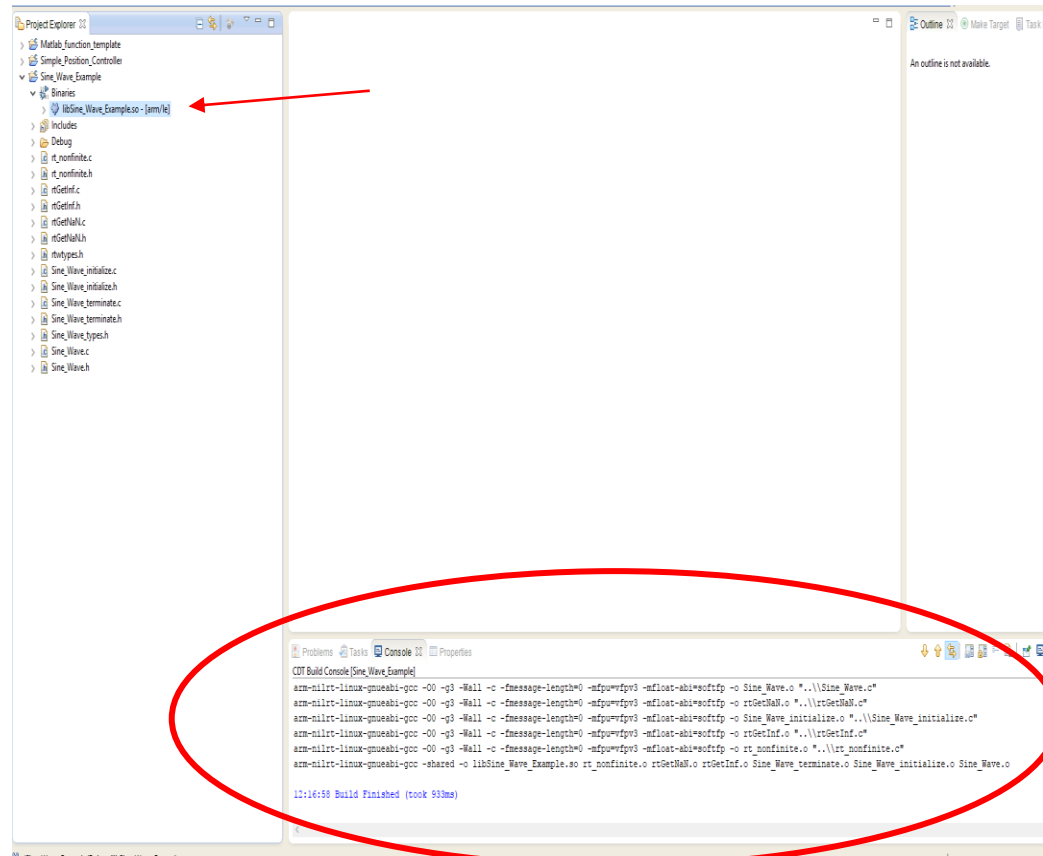
k) Select all the .c and .h files; they should be under ...codegen/lib/function_name. To find the exact folder location refer to step 2f. Once all the appropriate files are selected, click Finish.



l) Now, build the project.



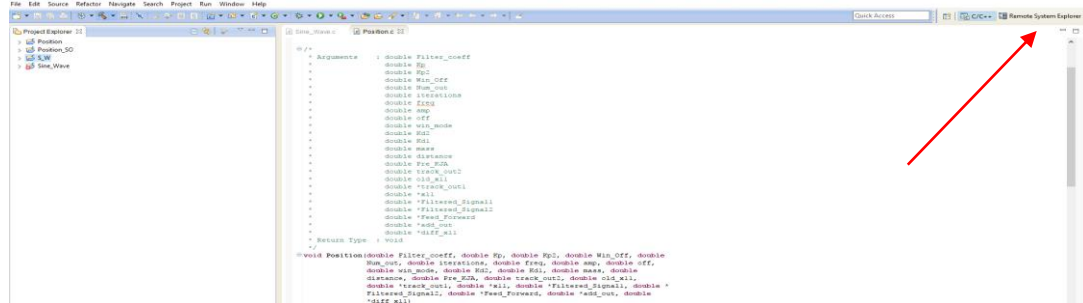
m) Once the project is built successfully, the console should look like this and the .so file should be under “Binaries”. Copy the .so file so that it can be pasted to the myRIO (next step). Only copy the .so file, do not copy the files contained in the .so file.



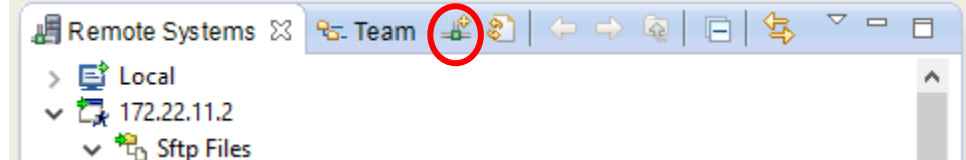
4. Copying the Shared Object file to the myRIO

Steps b-d are not needed if the myRIO is located under "Remote Systems" (to see if the myRIO is located under "Remote Systems" and look for the myRIO's IP Address). If the user does not know the myRIO's IP Address, it can be found by launching NI Max desktop application and selecting "Remote Systems" on left side of the window.

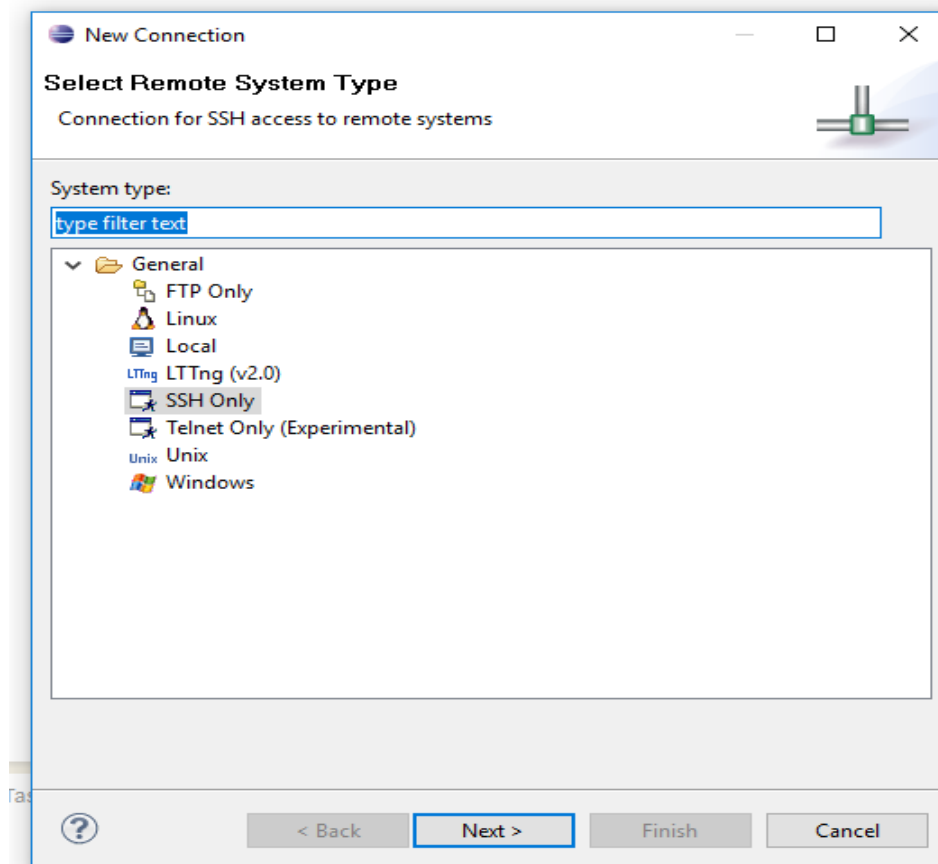
- a) Click "Remote System Explorer" on the right hand side of the window.



- b) Click the "Define a connection to remote system" button (circled in red).



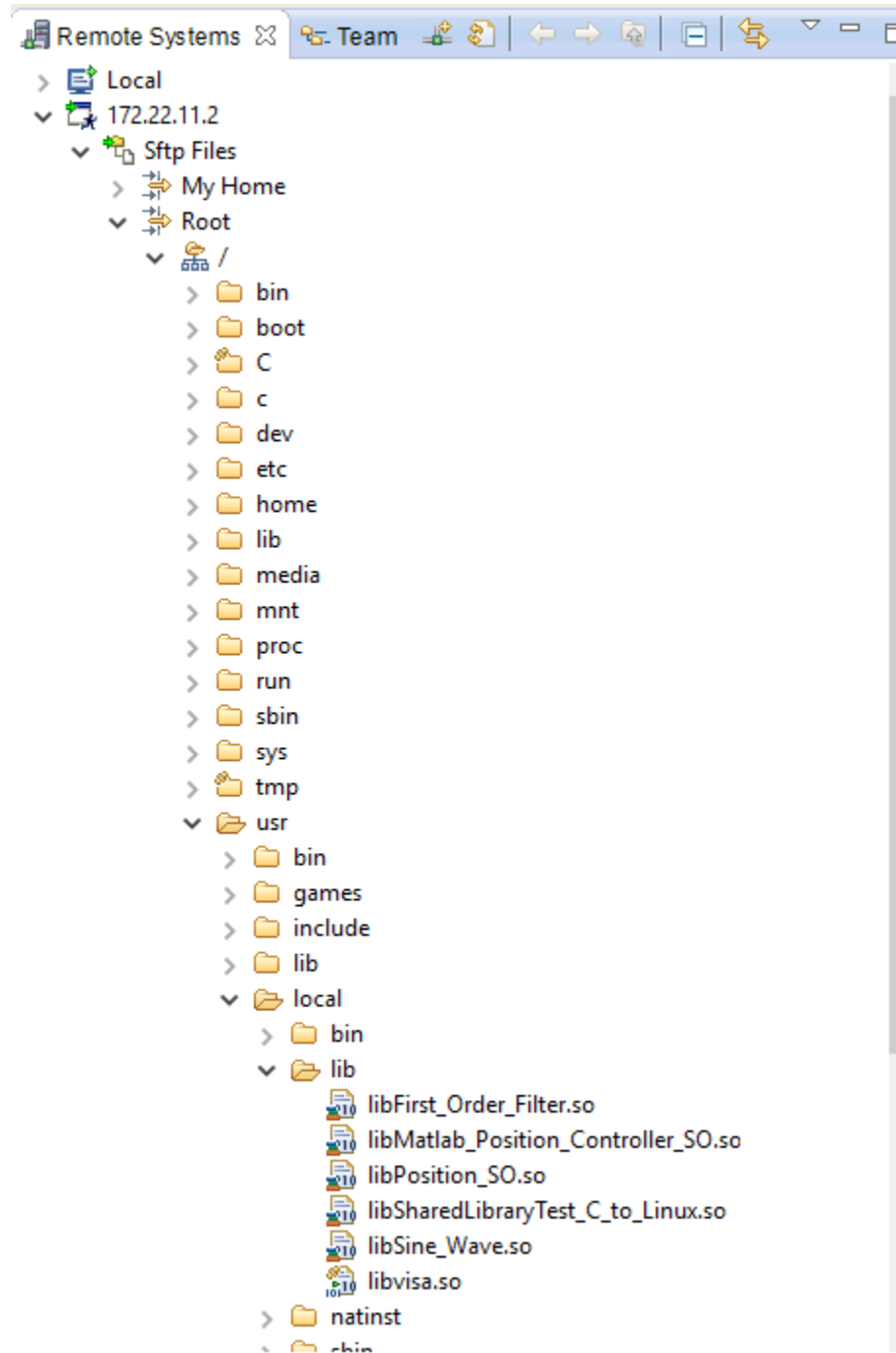
- c) Select "SSH Only" then click next



- d) Enter the IP Address of the myRIO as the “Host name” and “Connection name”, and then click finish.

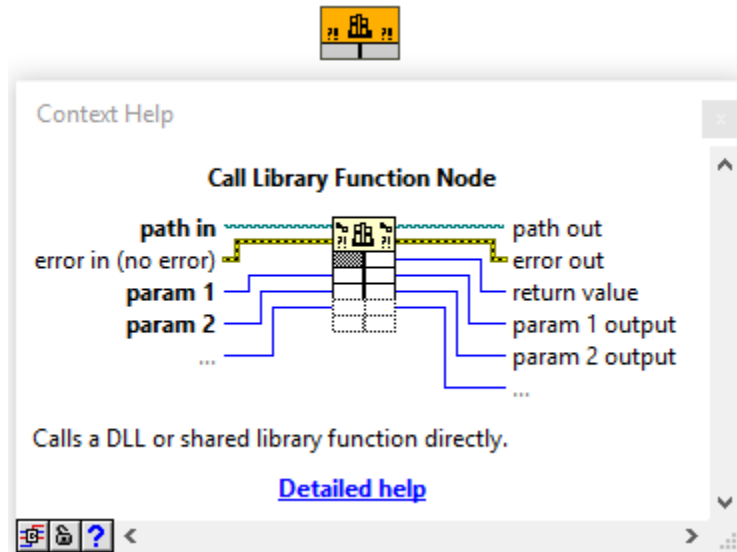
The screenshot shows a Windows-style dialog box titled "New Connection" with a standard minimize, maximize, and close button set in the top right corner. Below the title bar, the main heading is "Remote SSH Only System Connection", followed by the subtitle "Define connection information". The dialog contains several input fields: "Parent profile:" is a dropdown menu currently showing "BELAP79023"; "Host name:" is a text box containing "172.22.11.2" with a dropdown arrow on the right; "Connection name:" is a text box also containing "172.22.11.2"; and "Description:" is an empty text box. Below these fields, there is a checked checkbox labeled "Verify host name" and a blue hyperlink that reads "Configure proxy settings". At the bottom of the dialog, there is a row of four buttons: a help button (a circle with a question mark), a "< Back" button, a "Next >" button, and a "Finish" button which is highlighted with a blue border. To the right of the "Finish" button is a "Cancel" button.

- e) Now double click on the myRIO's IP address and go to Sftp Files->Root, then enter the myRIO's login information (currently the username and password are both "admin"). Then go to /usr/local/lib and paste the .so file to this folder.

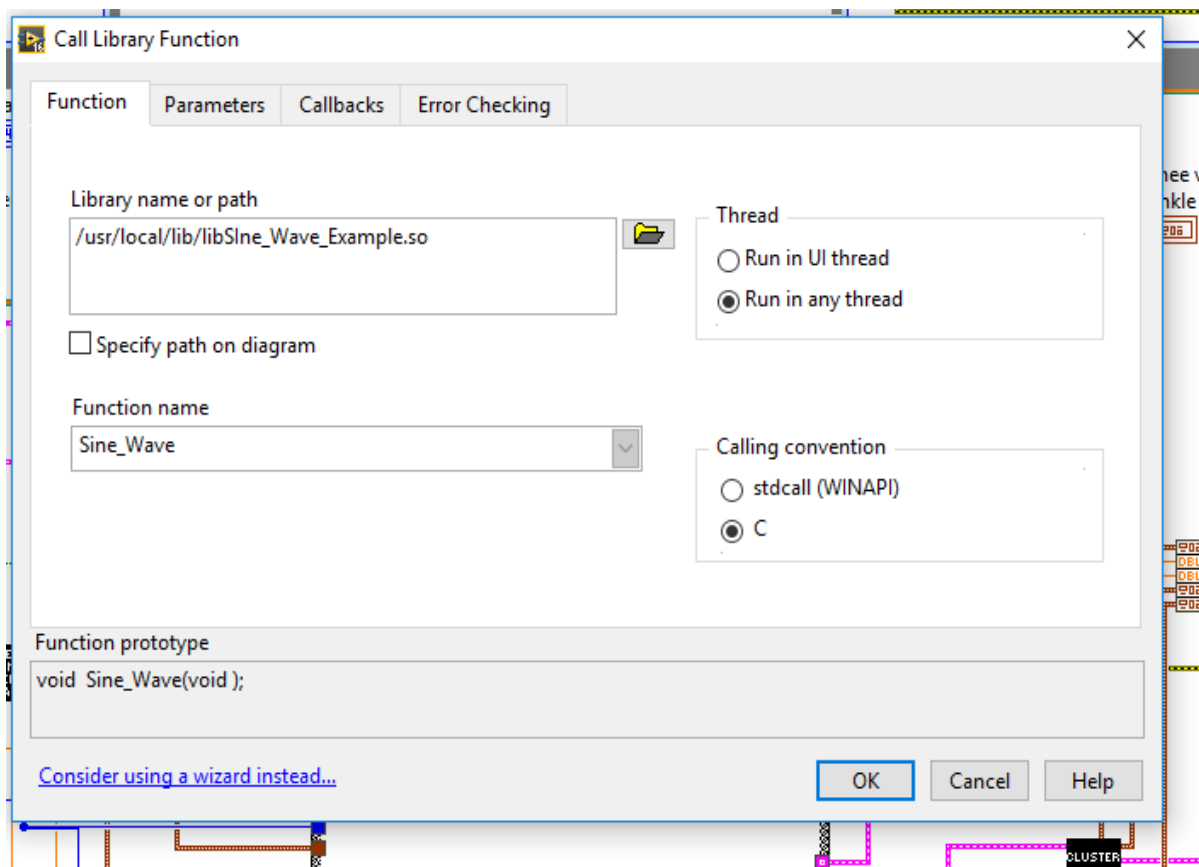


5. Implementing Shared Object file onto LabVIEW code

- a) Go to LabVIEW and add the Call Library Function Node to the Block Diagram.

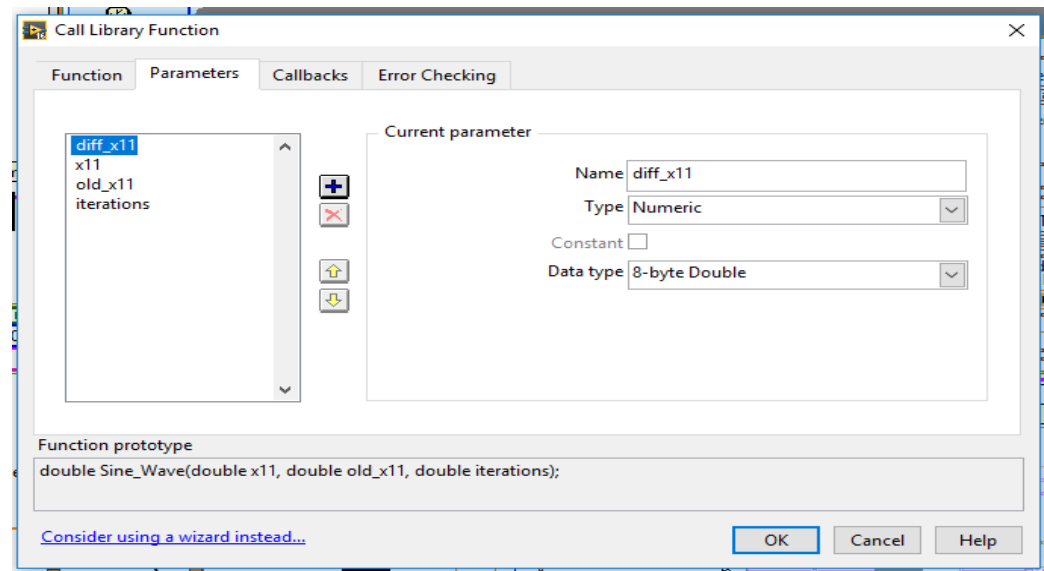


- b) Double click on the Call Library Function Node and go to the "Function" tab. Then, under "Thread" select "Run in any thread". The "Function name" should be the same function name that was used in MATLAB. The Library path is where the .so file is located in the myRIO. After filling out the information under the "Function" tab select the "Parameter" tab.

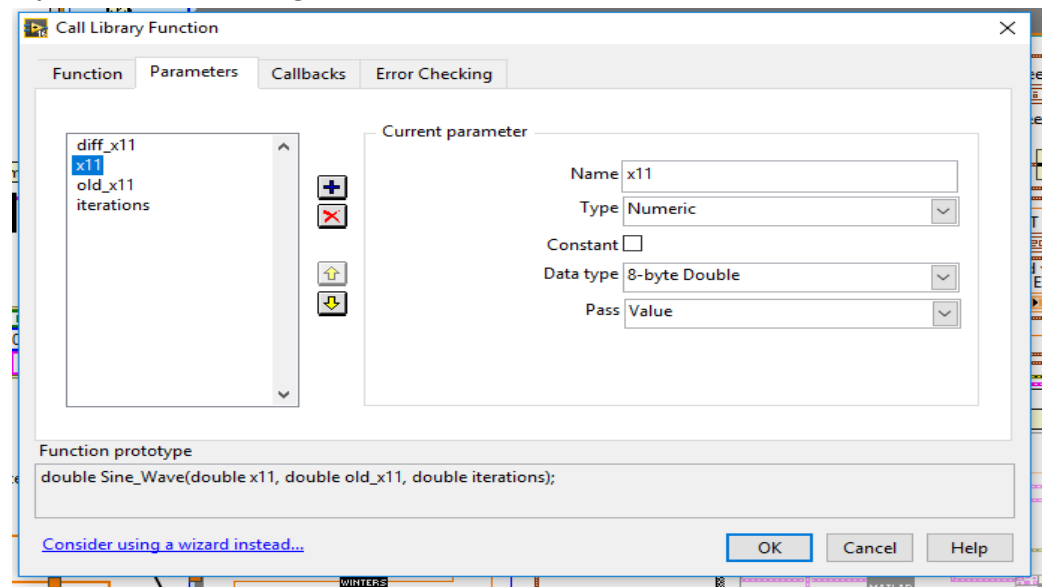


- c) Add the inputs and outputs of the MATLAB code. In “Sine_Wave” MATLAB code used in this tutorial, diff_x11 is the only output (return parameter) and x11, old_x11, and iterations are all inputs. Since the “Sine_Wave” MATLAB code only has one output, that output is treated as the return parameter. However, if the MATLAB code had multiple outputs then, the return parameter would have a “Type” of “void” (will not be used). For inputs, “Pass” is “Value” and for outputs (when there is more than one output), “Pass” is “Pointer to Value”. In addition, for the return parameter there is not a “Pass” option. Below are examples of the settings for each parameter type (return, input, and output):

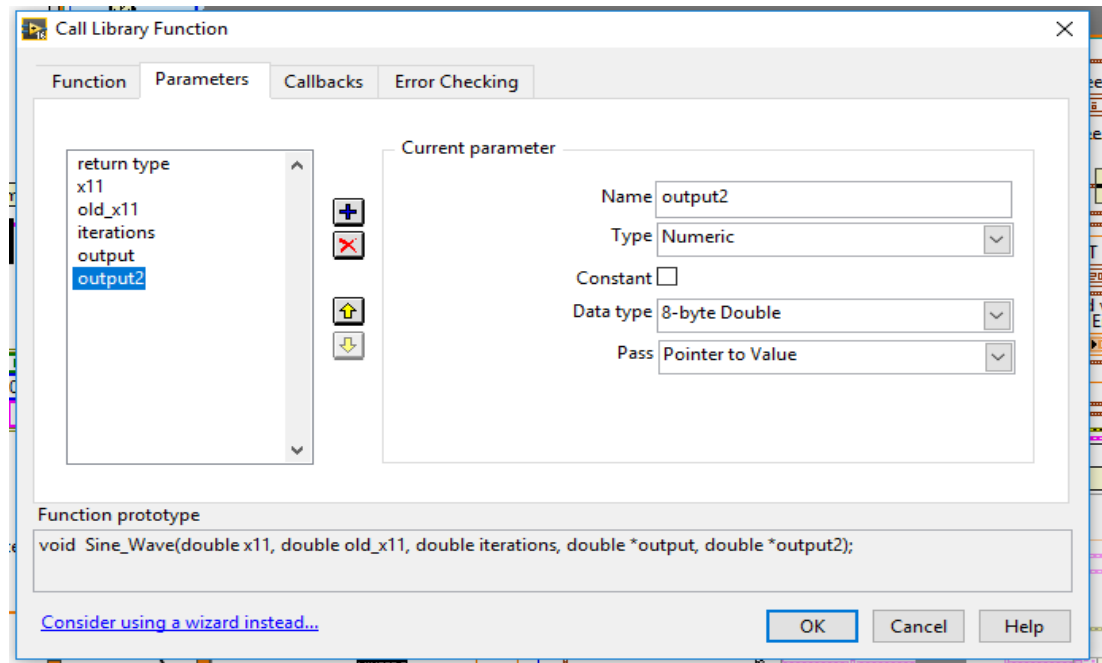
Return Parameter Settings



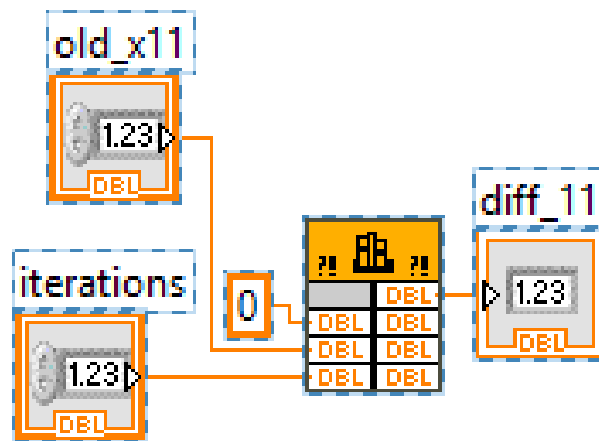
Input Parameter Settings



Output Parameter Settings (for Multiple Outputs)



- d) Now connect the inputs to the corresponding constants or controls then, connect the outputs to the corresponding indicators like so. The LabVIEW block below has one output (the return parameter) and three inputs.



The tutorial is now complete! To test the implementation of the MATLAB code on LabVIEW, click run.