**Implement Nakagami Distribution of MATLAB in NetSim without using .m file**

**Software Used:** NetSim Standard v10 (64-bit), Visual Studio 2015, MATLAB 2011a (64-bit)

In this example we will replace the default Rayleigh Fading (part of the path loss calculation) used in NetSim, with a Fading Power calculated using the Nakagami Distribution from MATLAB

**Procedure:**

1. Create a MATLAB\_Interface.c file inside the IEEE802\_11 folder which can be found in the path <NetSim\_Install\_Direcotry>/src/Simulation/. Write the following code inside the MATLAB\_Interface.c file:-

/\*

\*

\* This is a simple program that illustrates how to call the MATLAB

\* Engine functions from NetSim C Code.

\*

\*/

#include <windows.h>

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include "engine.h"

#include "mat.h"

#include "mex.h"

char buf[100];

Engine \*ep;

int status;

mxArray \*h=NULL, \*i=NULL, \*j=NULL,\*k=NULL;

mxArray \*out;

double \*result;

double fn\_netsim\_matlab\_init()

{

/\*

\* Start the MATLAB engine

\*/

if (!(ep = engOpen(NULL))) {

MessageBox ((HWND)NULL, (LPCWSTR)"Can't start MATLAB engine",

(LPCWSTR) "MATLAB\_Interface.c", MB\_OK);

exit(-1);

}

engEvalString(ep,"desktop");

return 0;

}

double fn\_netsim\_matlab\_run()

{

//write your own implementation here

int nakagami\_shape=5,nakagami\_scale=2;

engPutVariable(ep,"h",h);

sprintf(buf,"h=ProbDistUnivParam('nakagami',[%d %d])",nakagami\_shape,nakagami\_scale);

status=engEvalString(ep,buf);

engPutVariable(ep,"i",i);

sprintf(buf,"i=random(h,1)");

status=engEvalString(ep,buf);

out=engGetVariable(ep,"i");

result=mxGetPr(out);

return \*result;

}

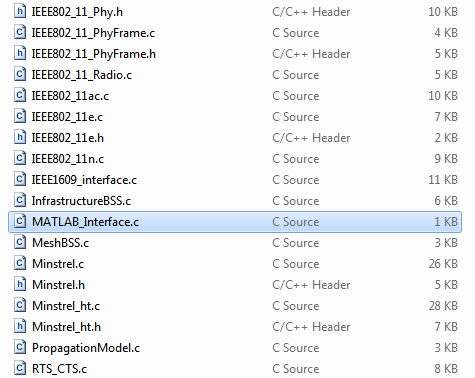
double fn\_netsim\_matlab\_finish()

{

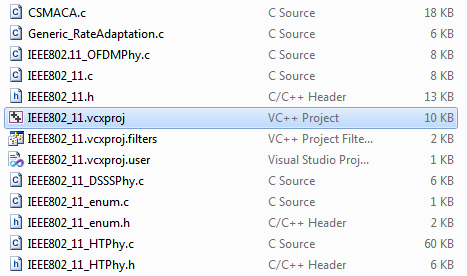
status=engEvalString(ep,"exit");

return 0;

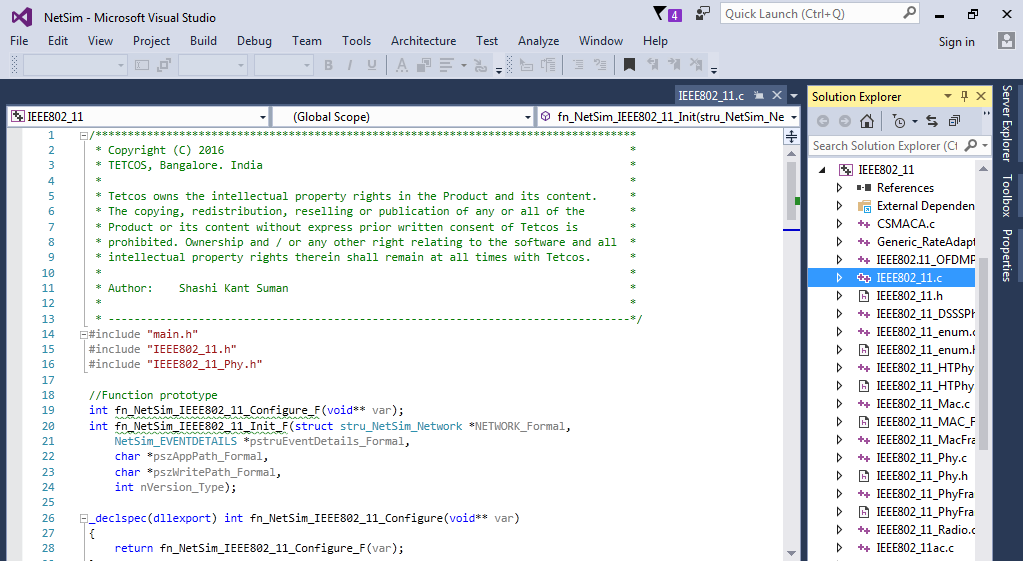
}

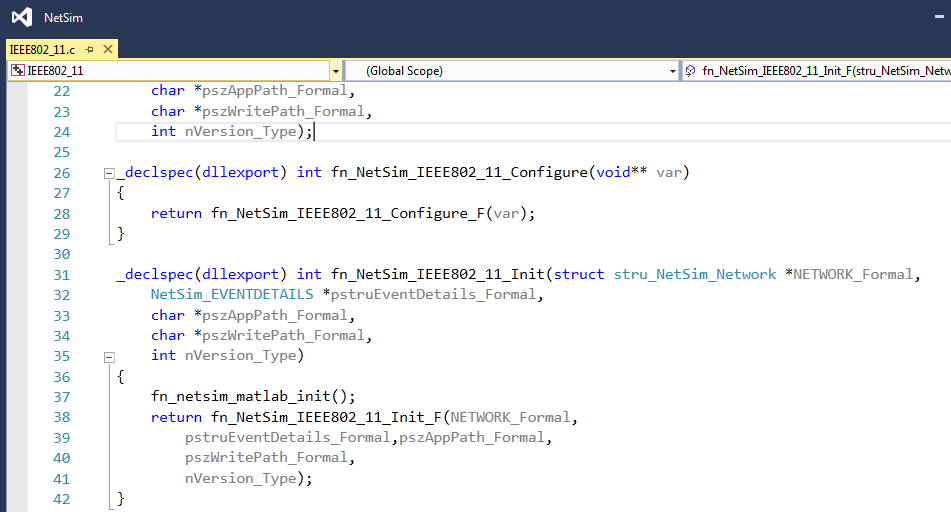


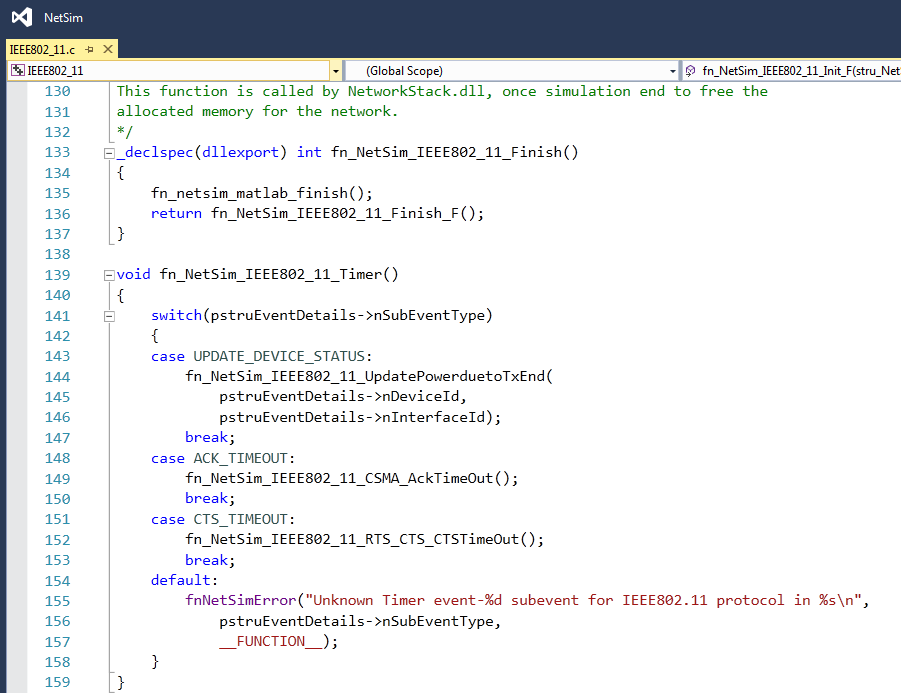
1. Now open IEEE802\_11 project file, inside the IEEE802\_11 folder.



1. Right click on “IEEE802\_11 Project” present in “Solution Explorer” window and select Add 🡪 Existing Item and select the MATLAB\_Interface.c file.
2. MATLAB\_Interface.c file contains the following functions
   1. fn\_netsim\_matlab\_init() - Opens the MATLAB Engine
   2. fn\_netsim\_matlab\_run() - Communicates with MATLAB Command Window
   3. fn\_netsim\_matlab\_finish() - Closes the MATLAB Engine
3. In the Solution Explorer double click on the IEEE802\_11.c file.



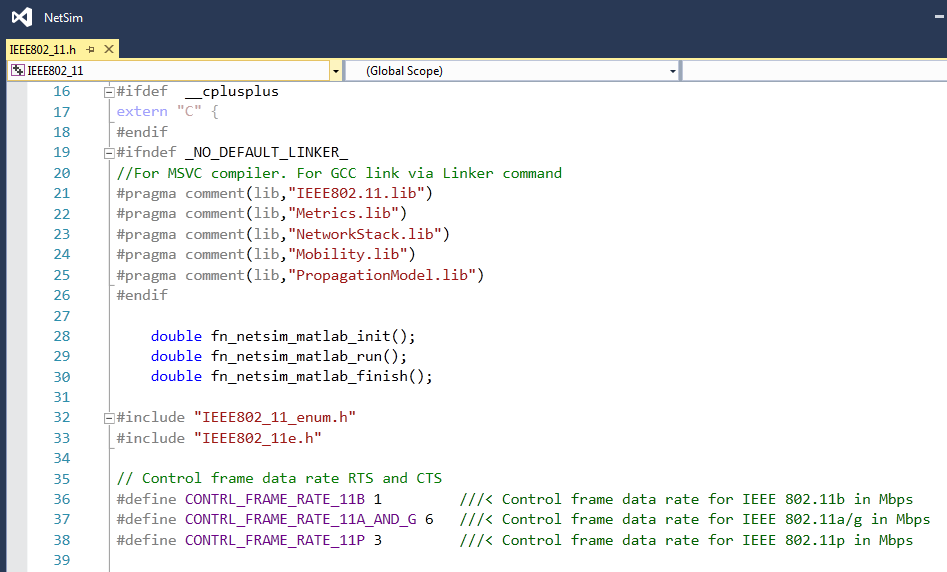
1. Add a call to fn\_netsim\_matlab\_init() inside the fn\_NetSim\_IEEE802\_11\_Init() function.
2. Similarly add a call to fn\_netsim\_matlab\_finish() inside the fn\_NetSim\_IEEE802\_11\_Finish() function.



1. In the Solution Explorer double click on the IEEE802\_11.h file. Add definitions of the following functions

double fn\_netsim\_matlab\_init();

double fn\_netsim\_matlab\_run();

double fn\_netsim\_matlab\_finish();

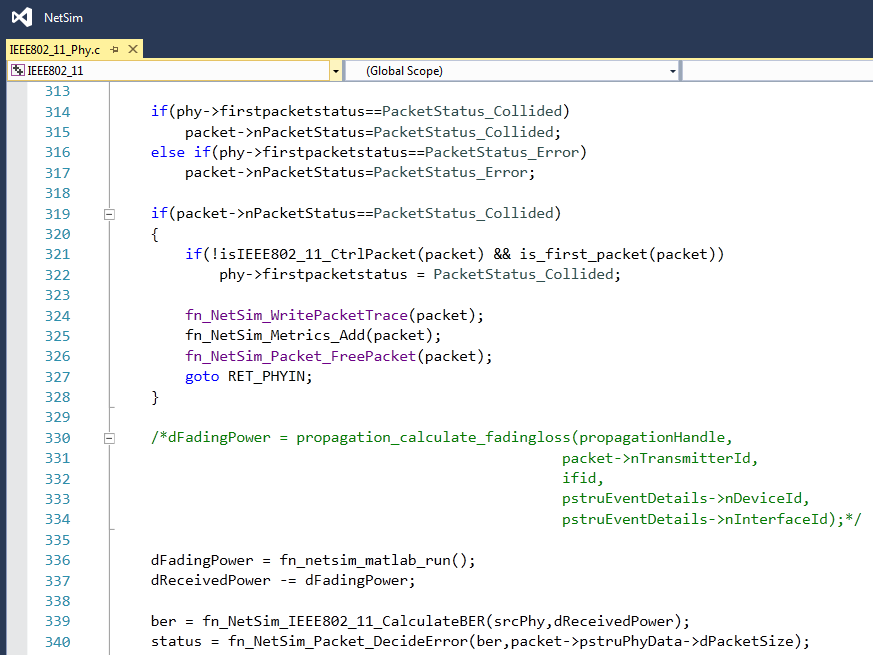
1. In the Solution Explorer double click on the IEE802\_11\_PHY.c file.
2. Inside fn\_Netsim\_IEEE802.11\_PHYIn() function comment the lines,

dFadingPower = propagation\_calculate\_fadingloss(propagationHandle,

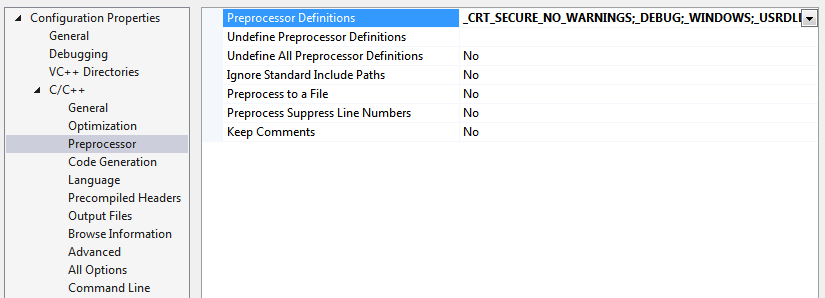
packet->nTransmitterId,ifid, pstruEventDetails->nDeviceId,

pstruEventDetails->nInterfaceId);

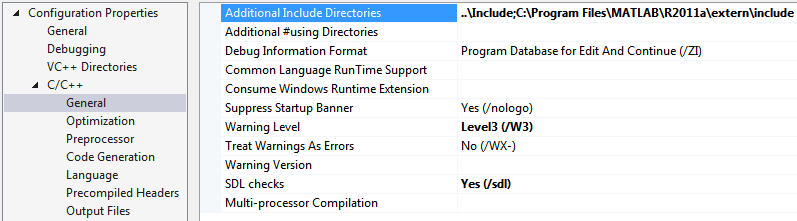
Make a call to the fn\_netsim\_matlab\_run() function by adding the following line,

 dFadingPower = fn\_netsim\_matlab\_run();

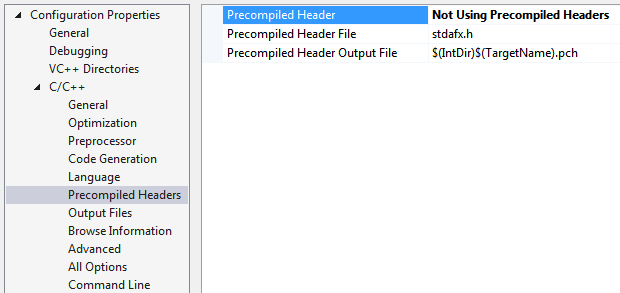
1. Right click on the IEEE802\_11 project and select PROPERTIES in the solution explorer. Go to Preprocessor under C/C++ Properties and edit Preprocessor Definitions by adding **\_CRT\_SECURE\_NO\_WARNINGS**.



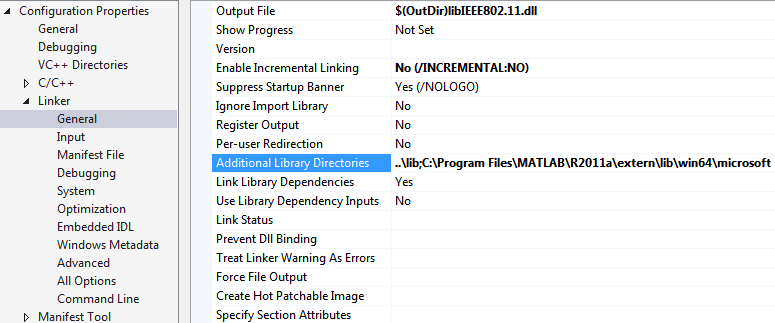
1. To compile a MATLAB engine application in the Microsoft Visual Studio 2015 environment, Right click on the IEEE802\_11 project and select PROPERTIES in the solution explorer. Once this window has opened, make the following changes:
   1. Under C/C++ 🡪 General, add the following directory to the field ADDITIONAL INCLUDE DIRECTORIES:

<Path where MATLAB is installed>\extern\include

* 1. Under C/C++ 🡪 Precompiled Headers, set PRECOMPILED HEADERS as "Not Using Precompiled Headers".

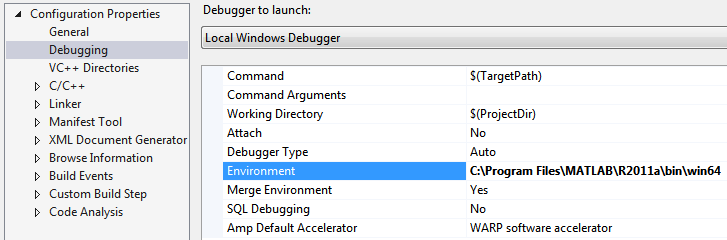


* 1. Under Linker 🡪 General, add the directory to the field ADDITIONAL LIBRARY DIRECTORIES:

<Path where MATLAB is installed>\extern\lib\win64\microsoft

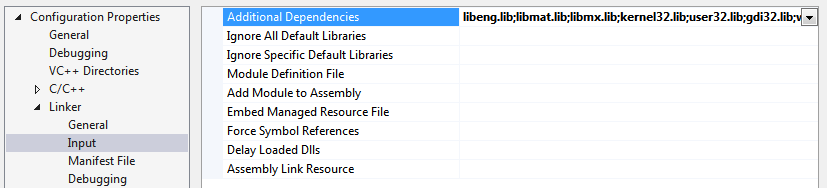
* 1. Under Configuration Properties 🡪Debugging, Add the following Target path in the ENVIRONMENT:

<Path where MATLAB is installed>\bin\win64



1. Under Linker 🡪 Input, add the following names to the field marked ADDITIONAL DEPENDENCIES:

libeng.lib, libmx.lib, libmat.lib;



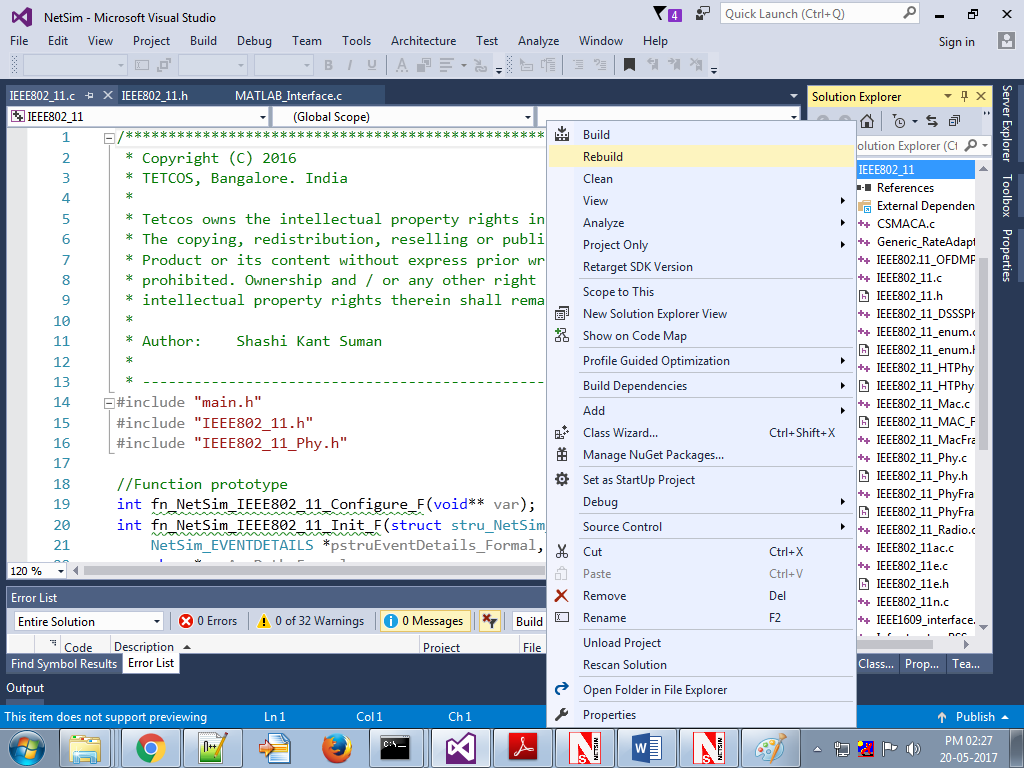
1. Make sure that the following directory is in the environment variable PATH:

<Path where MATLAB is installed>\bin\win64

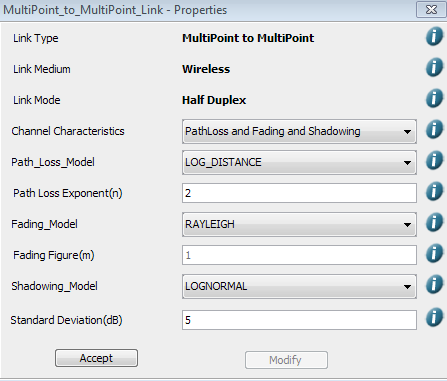
**NOTE:** To do step 14, check the Windows system path by clicking on Start 🡪 Right click on Computer 🡪 Properties 🡪 Advanced System Settings 🡪 Environment variables 🡪 System Variables 🡪 Open "Path" for editing.

Note: If the machine has more than one MATLAB installed, the directory for the target platform must be ahead of any other MATLAB directory (for instance, when compiling a 64-bit application, the directory in the MATLAB 64-bit installation must be the first one on the PATH).

1. Now Right Click on IEEE802\_11 project and select Rebuild.



1. Now replace the newly built libIEEE802.11.dll from the DLL folder, into the NetSim bin folder. Please ensure you rename the original libIEEE802.11.dll file to retain a copy of the original file.
2. Run NetSim in Administrative mode. Create a Network scenario involving IEEE802\_11 say MANET, and set the Fading Figure value in the Multipoint to Multipoint Link properties to 1, to ensure that Rayleigh fading is set.



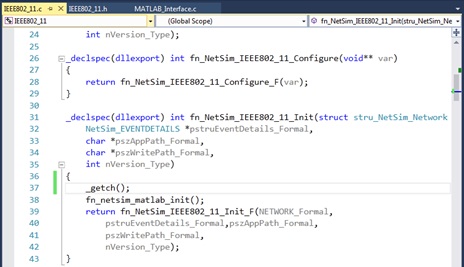
1. Perform Simulation. You will find that once the Simulation starts MATLAB command window starts and gets closed once the simulation is over.

Note: On Windows systems, engOpen opens a COM channel to MATLAB. The MATLAB software you registered during installation starts. If you did not register during installation, enter the following command at the MATLAB prompt:

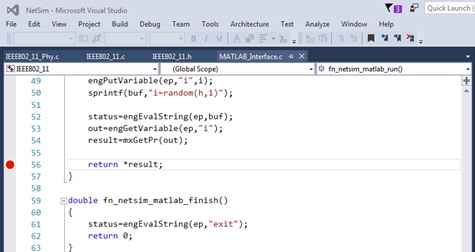
!matlab -regserver

### Debug and understand communication between NetSim and MATLAB

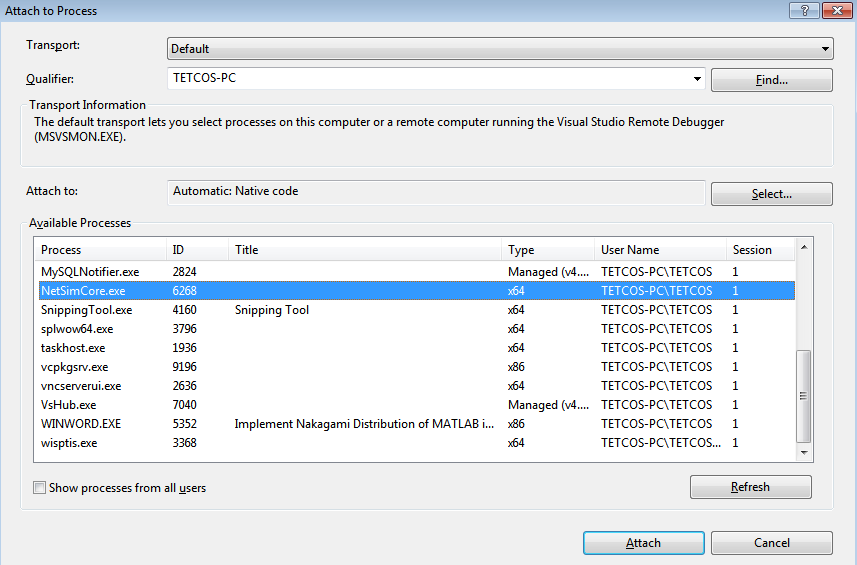
1. In the Solution Explorer of Visual Studio double click on IEEE802\_11.c and Add a \_getch() inside the fn\_NetSim\_IEEE802\_11\_Init() function.



1. In the Solution Explorer double click on MATLAB\_Interface.c file and place a breakpoint inside the fn\_netsim\_matlab\_run() function before the return statement.

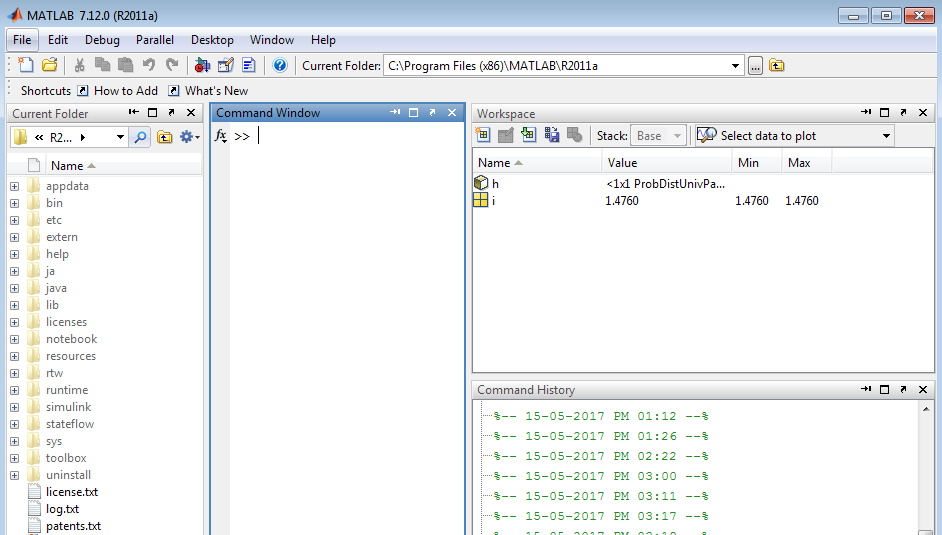


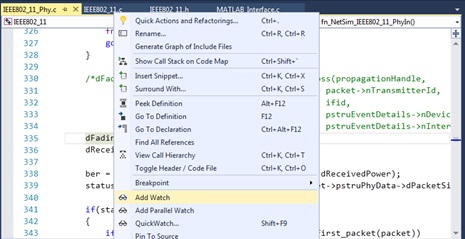
1. Rebuild the Dll and replace in bin path.
2. Now run the NetSim Scenario. The simulation window stops for user interrupt.
3. In Visual studio, go to Debug 🡪 Attach to Process.
4. From the list of Processes select NetSimCore.exe and click on Attach.



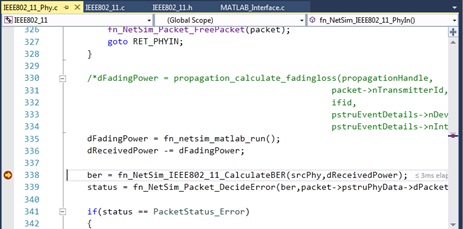
1. Now go to the Simulation window and press Enter.
2. MATLAB Command Window and MATLAB Desktop Window will start and breakpoint in Visual Studio gets triggered.



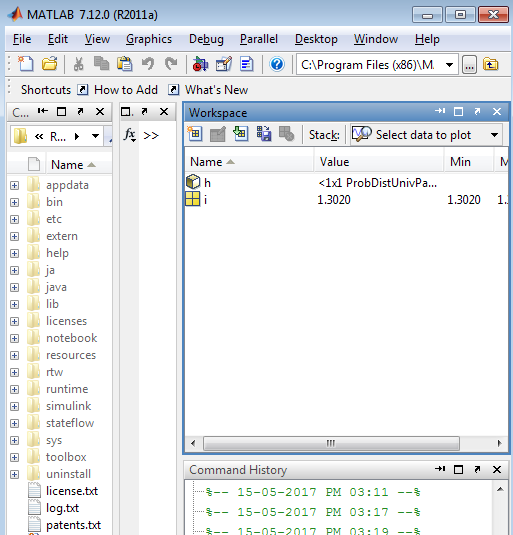
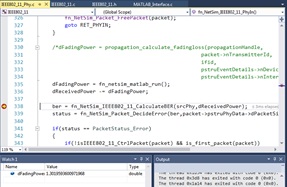
1. Now when debugging (say, by pressing F5 each time) you will find the computation taking place in the MATLAB Workspace.
2. This value of i obtained from MATLAB is used to calculate fading power instead of the Rayleigh Fading Model.
3. Add a watch to the dFadingPower variable in the IEEE802\_11\_Phy.c file. For this, right click on the variable dFadingPower and select “Add Watch” option. You will find a watch window containing the variable name and its value in the bottom left corner.



1. Now place another breakpoint after the line dFadingPower = fn\_netsim\_matlab\_run()



1. Now when debugging (say by pressing F5 each time) you will find that the watch window displays the value of dFadingPower whenever the control reaches the recently set breakpoint. You will also find that the value of dFadingPower in the Visual Studio Watch window and the value of i in the MATLAB workspace window are similar.

****