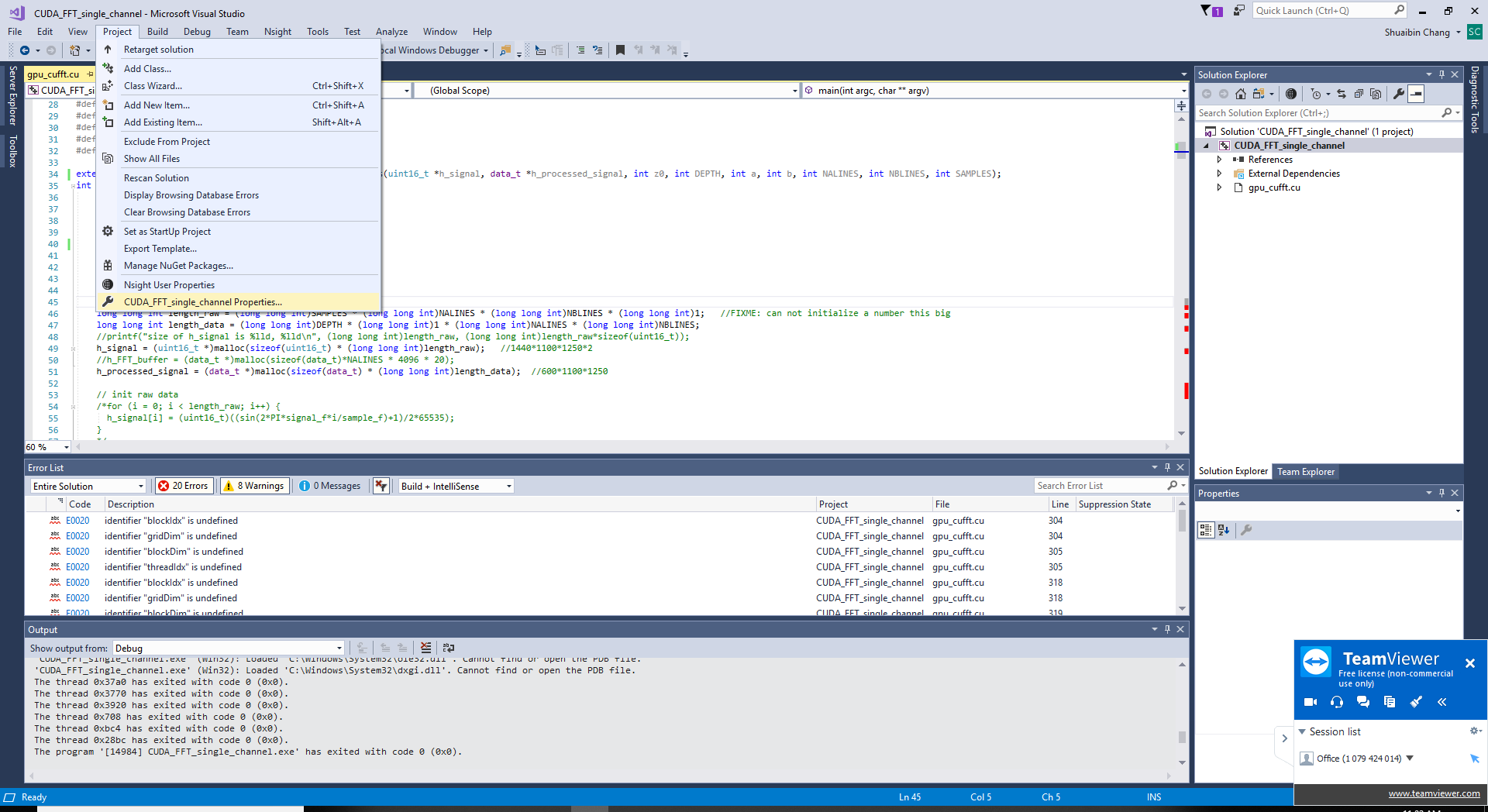
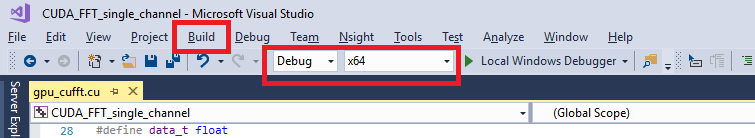
This is the manual for coding in CUDA and make DLL files.

When it comes to CUDA, NEVER start from scratch. Always start from a working example and modify it to suit your application.

You can copy this project and start from here.

This project was tested to successfully run on Visual studio 2017 community version, 64bit system and software. The GPU used was NVIDIA Quadro RTX4000. GPU driver was from CUDA Toolkit 10.1.

This example should be ready to build DLL file. If you start from another project that is not for building DLL, make the following changes to build DLL file:

1. extern "C" \_\_declspec(dllexport) in front of the function declaration that’s supposed to export in the DLL file. And put the declaration at the top before any other functions, including main()
2.  Open Project-> <project name> Properties to make following changes
3. Configuration Properties-> General-> Configuration Type, choose DLL if you want build a DLL file
4. Configuration Properties-> VC++ Directories-> Include Directories, add the path of the CUDA Toolkit inc folder, like, C:\ProgramData\NVIDIA Corporation\CUDA Samples\v10.1\common\inc
5. Configuration Properties-> CUDA C/C++-> Common-> Additional Include Directories, again add the CUDA Toolkit inc folder just like in step 4
6. Configuration Properties-> Linker-> General-> Additional Library Directories, add **$(CudaToolkitLibDir)**  there, use ; to separate different items
7. Configuration Properties-> Linker-> Input-> Additional Dependencies, add **cifft.lib** there, use ; to separate different libraries.
8. Click apply
9.  Sometimes Release has problems finding some CUDA functions, use Debug instead. Make sure you are choosing x64 for 64bit application and x86 for 32bit application. When building DLL files, use Build-> build/rebuild solution, DO NOT use the local windows debugger as DLL can not directly execute as .exe files

The generated DLL files/exe files can be found in <project folder>/x64/Debug. You can use it in any 64bit/32bit application, depending on your choice of x64/x86 when building the DLL file. For example, you can use it in LabVIEW to call some external C functions for faster execution.

This is a description of data format in this code.

h\_data and d\_data: 1d array with 16bit unsigned integer

FFT\_buffer: 1d array with 32bit float

d\_processed: 1d array with 16bit unsigned integer

---------Shuaibin(Stephan) Chang

01/27/2020