**Using a Pre-Built Executable with Intel Redistributable Libraries**

To install CrunchTope, run the CrunchSetup-Windows.exe installation executable from

<https://github.com/CISteefel/CrunchTope/releases/tag/v2.10>

to install both CrunchTope and the Exercises. The installer should modify the System Registry to add the location of CrunchTope that you have chosen to the PATH Environment Variable. This way, CrunchTope can be run from any folder/directory. The default will be to put the executable and short course exercises it into C:\Software\CrunchTope, but in the window immediately after the License Agreement, you have an opportunity to change the Destination Folder. The issue may be that you as a User may not have rights run software in the C:\Software directory (this might require Administrative Privilege).

So that CrunchTope can find the needed libraries, you should install the Intel oneAPI Runtime package:

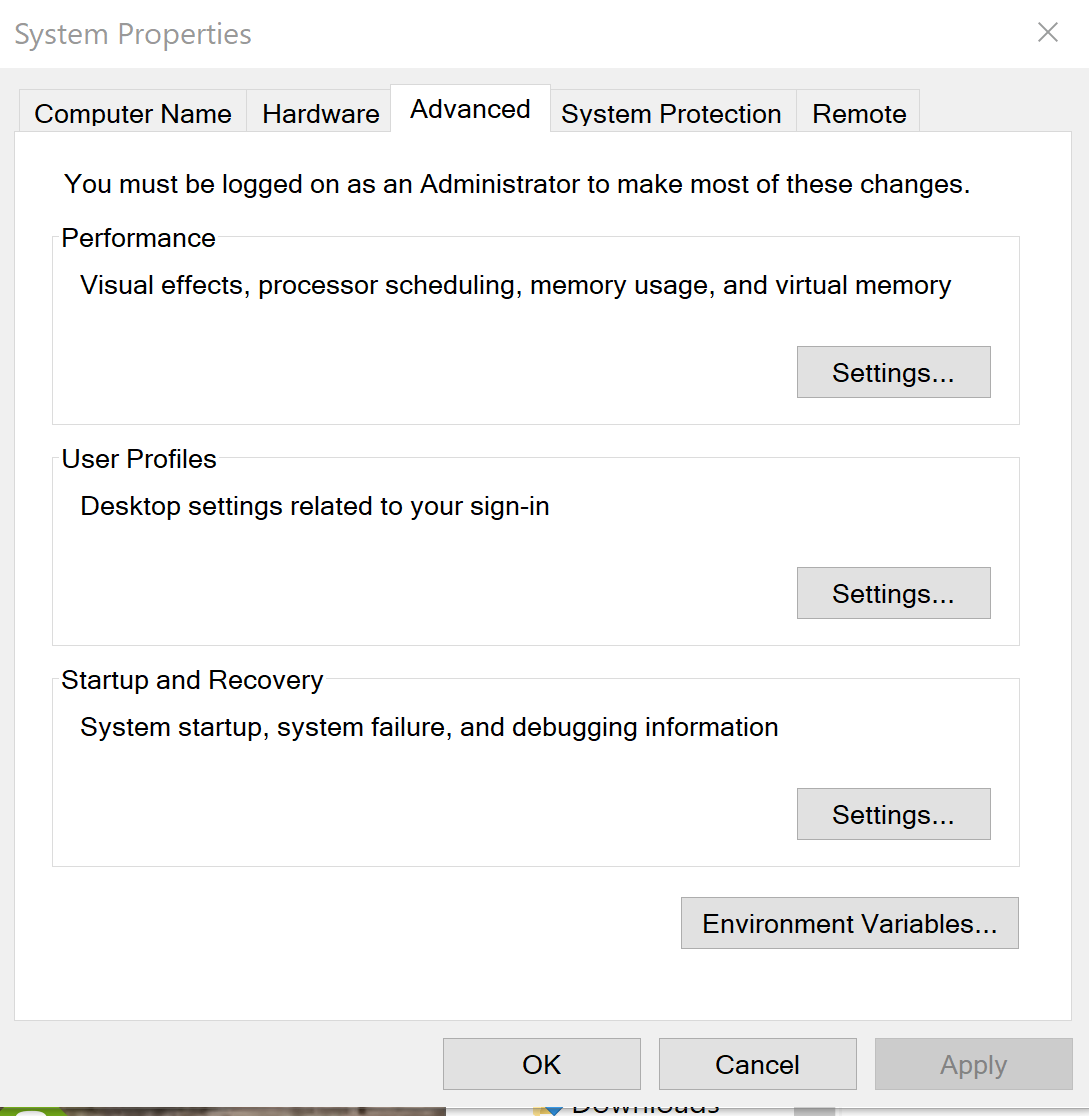
[Intel® oneAPI DPC++/C++ Compiler Runtime for Windows](https://registrationcenter-download.intel.com/akdlm/irc_nas/18850/w_dpcpp_cpp_runtime_p_2022.2.0.9553.exe)

[Intel® Fortran Compiler Runtime for Windows](https://registrationcenter-download.intel.com/akdlm/irc_nas/18910/w_ifort_runtime_p_2022.2.0.9553.exe)

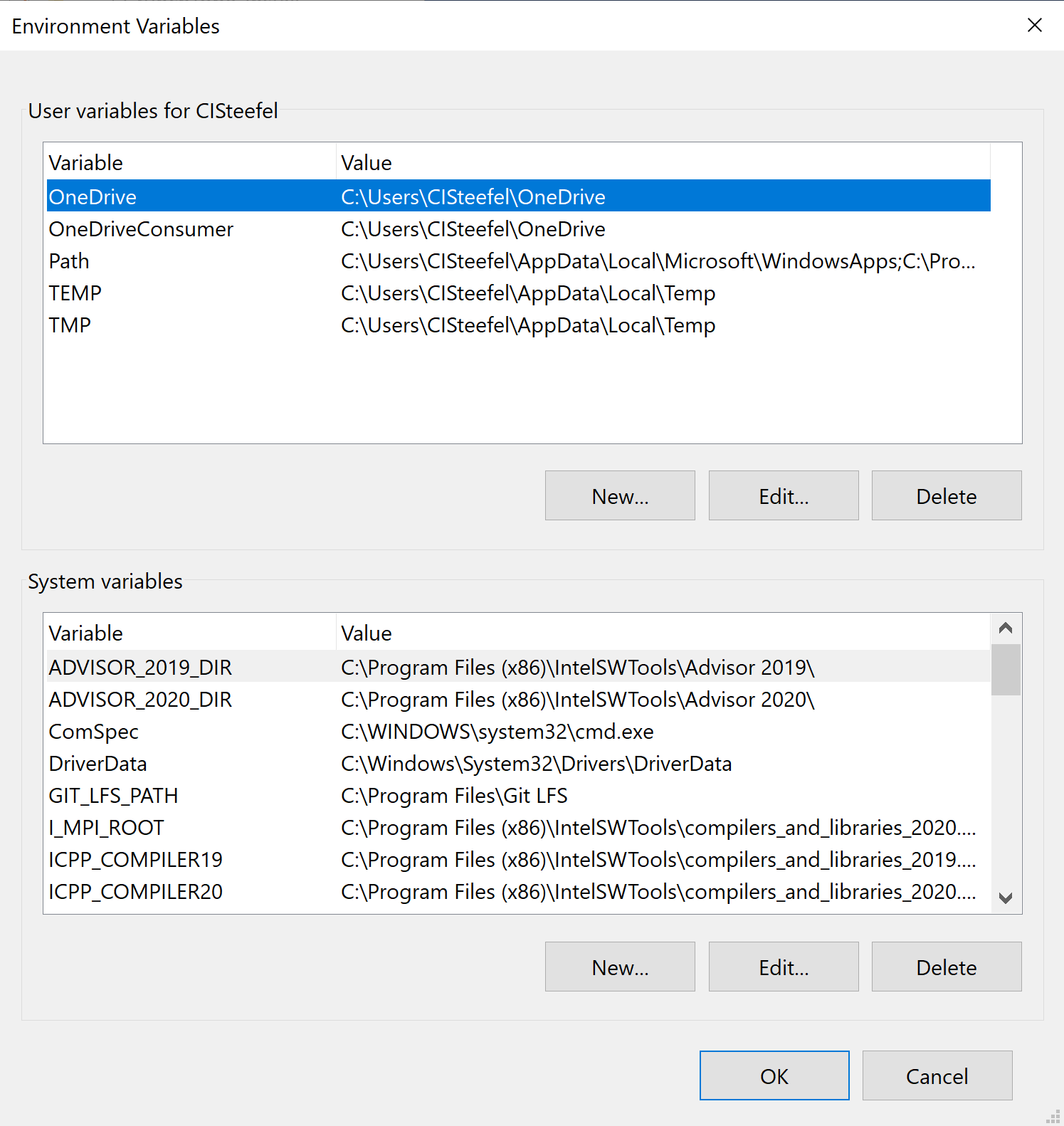
Ideally this should be done before installing Crunch.

Otherwise, you can do this manually with:

Type in the Start panel “Environment Variables” and you should a best match with “Edit the system environment variables” in Control Panel. You should see:



Here you want to click on “Environment Variables”.



Then go to “System Variables” in this panel and scroll down to Path and use the Edit button and add the location of the CrunchTope executable, e.g.,

C:\Software\CrunchTopeRun

Or whatever folder you have chosen. Best to NOT put spaces in the address (even though Windows allows this), since Cygwin (Unix) will have trouble with this.

Then when click OK and then again OK and then OK, this will be saved. Then when you run a cmd.exe command from start, the Command Window should find the executable from ANY folder.

**Instructions for Building PETSc and CrunchFlow on Windows**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Building PETSc on Windows**

* Install Microsoft Visual Studio 2022.
* Install Intel oneAPI Base Kit and oneAPI HPC (Fortran is in the HPC libraries)

Since the Configure scripts for PETSc really work well only with UNIX or Linux type systems, the recommended approach (see <http://www.mcs.anl.gov/petsc/documentation/installation.html#Windows>) is to install Cygwin. There may be other Unix emulators that work, not sure.

Make sure you also get the entire Python distribution (this is not part of the Default installation) and you get the Gnu Make routine within the *Devel* package. Since Cygwin includes a “link” command that can interfere with the Intel compilers, you should carry out this command in a Cygwin BASH shell:

mv /usr/bin/link.exe /usr/bin/link-cygwin.exe

It turns out that much of the difficulty in getting PETSc to build easily is due to the failure to find the right Environmental Variables and compilers in the Cygwin BASH shell. The key is to follow these steps.

Easiest way is to go to the Start panel in Windows 11 and “search apps, settings, and documents”. Under Apps, scroll down to the Intel oneAPI icon and choose “*Intel oneAPI command prompt for Intel 64 for Visual Studio*”. This will start a Command window that has all of the oneAPI environment variables set (for compilers, for Math Kernel Library, for MPI, etc.).

Otherwise, to do this manually, open a Command window and navigate to:

C:\Program Files (x86)\Intel\oneAPI\

and give the command:

setvars.bat intel64

This will set the various Intel oneAPI flags.

After these are set, run from within the same Command window where you have just set the Intel Environment Variables the command:

C:\cygwin64\bin\mintty.exe -

This launches a bash shell in the Cygwin Unix environment, but it has to be done from the same Windows Command window where the environment variables were set.

If everything has worked correctly, the Cygwin bash shell should have inherited the Environment Variable settings from running the “setvars.bat” script. Test for this by now running within the same Cygwin terminal from a directory other than the one actually containing the files so as to test whether the compilers are in the system search paths:

which icx

which ifx

The location of these compilers should be echoed (the full path shown), if not, the paths have not been set correctly. If not (i.e., you get a message like “No ifx found in …”,), then you will need to add the location of these files manually. Or better yet, repeat the procedures above until you get them right.

Then, change directories to where you want to install PETSc, typically within C:\software. But you should use the Unix address (Unix cannot handle the spaces in Windows) when in the Cygwin terminal:

cd /cygdrive/c/software

Next, install PETSc on your machine from <http://www.mcs.anl.gov/petsc/index.html>. Execute this command in your Cygwin terminal

git clone [https://gitlab.com/petsc/petsc.git --branch v3.21.6](https://gitlab.com/petsc/petsc.git --branch%20v3.21.6)

followed by

export PETSC\_DIR=/cygdrive/c/software/petsc

**!!! Petsc-3.22 does not work with CrunchFlow !!!**

We need to change the arguments in the Fortran calls for it to work, so for now use petsc-3.21.6 or earlier (the “git” command should create the “petsc directory).

From your Cygwin terminal when using a Windows-downloaded version of “git”, you can execute the command:

git config --global core.autocrlf input

to avoid error messages about DOS scripts (related to Windows line endings). You should get the proper Unix-based line endings coming over.

In the Python scripts below, the value set for PETSC\_ARCH will override what is set elsewhere (e.g., in Windows Environment Variables, or in .bashrc). One can create as many PETSC\_ARCH as needed, since each configure build will create a separate directory with that name. The user can then switch between these various PETSC\_ARCH options, using either the Windows Environment Variable setting for PETSC\_ARCH, or in the user’s .bashrc profile, or at the command line.

**Debug Version**

Here we configure the “no MPI” version (--with-mpi=0).

For statically linked libraries, use:

./configure PETSC\_ARCH=oneAPI-noMPI-opt \

--with-cc=/cygdrive/c/software/petsc/lib/petsc/bin/win32fe/win32fe\_icx \

--with-fc=/cygdrive/c/software/petsc/lib/petsc/bin/win32fe/win32fe\_ifx \

--with-cxx=0 \

--with-mpi=0 \

--with-blaslapack-dir=/cygdrive/c/PROGRA~2/Intel/oneAPI/mkl/latest/lib \

--with-debugging=1 \

--with-shared-libraries=0

**Optimized Version**

./configure PETSC\_ARCH=oneAPI-noMPI-opt \

--with-cc=/cygdrive/c/software/petsc/lib/petsc/bin/win32fe/win32fe\_icx \

--with-fc=/cygdrive/c/software/petsc/lib/petsc/bin/win32fe/win32fe\_ifx \

--with-cxx=0 \

--with-mpi=0 \

--with-blaslapack-dir=/cygdrive/c/PROGRA~2/Intel/oneAPI/mkl/latest/lib \

--with-debugging=0 \

--with-shared-libraries=0

**For MPI version (optimized version)**

./configure PETSC\_ARCH=mpi-oneAPI-opt \

--with-cc=/cygdrive/c/software/petsc/lib/petsc/bin/win32fe/win32fe\_icx \

--with-fc=/cygdrive/c/software/petsc/lib/petsc/bin/win32fe/win32fe\_ifx \

--with-cxx=0 \

--with-mpi-include=/cygdrive/c/PROGRA~2/Intel/oneAPI/mpi/latest/include \

--with-mpi-lib=/cygdrive/c/PROGRA~2/Intel/oneAPI/mpi/latest/lib/impi.lib \

--with-mpiexec=/cygdrive/c/PROGRA~2/Intel/oneAPI/mpi/latest/bin/mpiexec.exe \

--with-blaslapack-dir=/cygdrive/c/PROGRA~2/Intel/oneAPI/mkl/latest/lib \

--with-debugging=0 \

--with-shared-libraries=0 \

If the configure.py step is successful, you will see a line at the bottom like:

Configure stage complete. Now build PETSc libraries with:  
   make PETSC\_DIR=/cygdrive/c/software/petsc PETSC\_ARCH=oneAPI-noMPI-opt all

which will then build petsc. Then you will see:

Now to check if the libraries are working do:

make PETSC\_DIR=/cygdrive/c/software/petsc PETSC\_ARCH=oneAPI-noMPI-opt check

which you should copy and paste to execute. If all successful, you should see:

Running PETSc check examples to verify correct installation

Using PETSC\_DIR=/cygdrive/c/software/petsc and PETSC\_ARCH=oneAPI-noMPI-opt

C/C++ example src/snes/tutorials/ex19 run successfully with 1 MPI process

Fortran example src/snes/tutorials/ex5f run successfully with 1 MPI process

Completed PETSc check examples

to move on to building Crunch.

**Building CrunchFlow on Windows**

Building a version of CrunchFlow using only a CYGWIN shell is similar to a building on a Linux or UNIX machine. Here you would use the Makefile included in the GitHub source distribution.

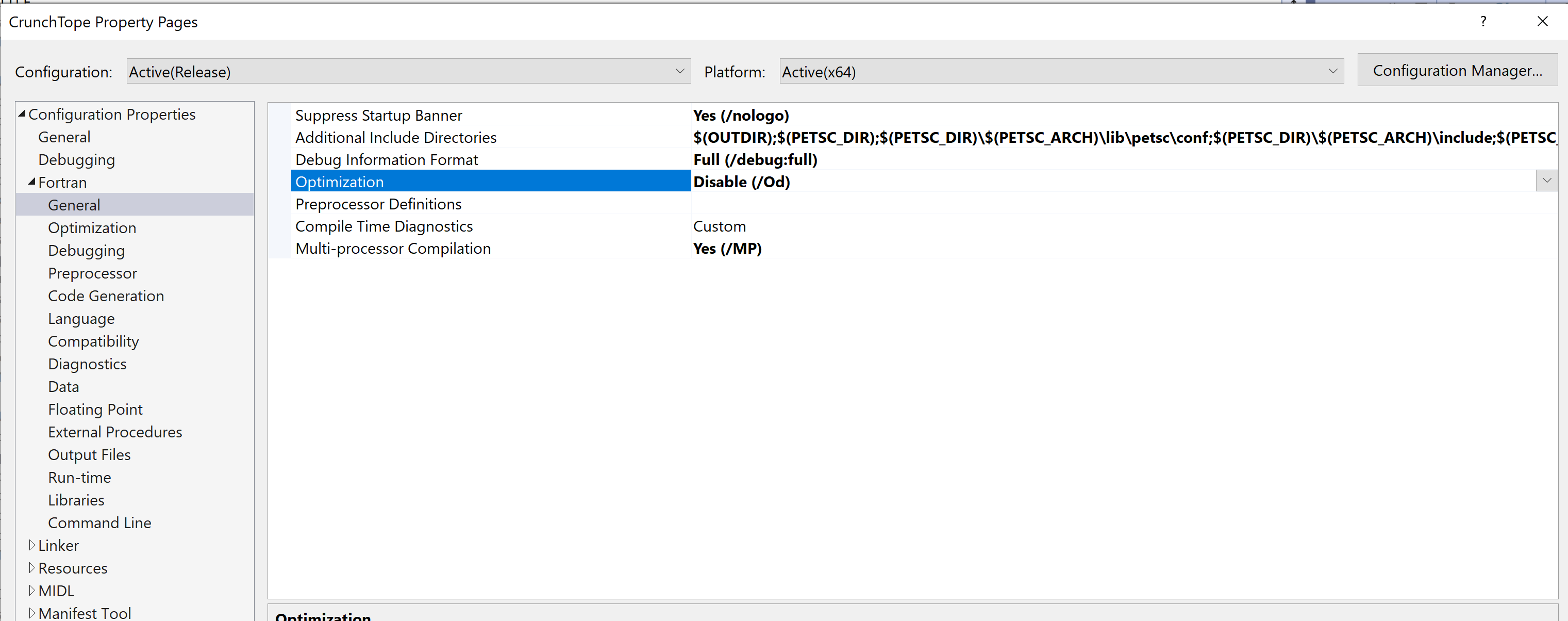
First, download CrunchTope from the GitHub repository:

git clone <https://github.com/CISteefel/CrunchTope.git>

To build CrunchFlow within the Microsoft Visual Studio Environment, you can get the Intel Visual Fortran project file from the GitHub repository for CrunchTope:

<https://github.com/CISteefel/CrunchTope>

This way, the files that should go in the current release will be properly identified by the Project file. However, there may be some fine-tuning of the Project Settings required, so the desired options are given below. These correspond to what one should see when selecting the Project dropdown menu, selecting CrunchFlow Properties.



Right click on the down arrow all the way on the right and choose “Edit” to bring up this under “Additional Include Directories”:

$(OUTDIR)

$(PETSC\_DIR)

$(PETSC\_DIR)\$(PETSC\_ARCH)\lib\petsc\conf

$(PETSC\_DIR)\$(PETSC\_ARCH)\include

$(PETSC\_DIR)\include\petsc\finclude

$(PETSC\_DIR)\lib\petsc\conf

$(PETSC\_DIR)\include

$(PETSC\_DIR)\include\petsc

$(PETSC\_DIR)\include\petsc\mpiuni

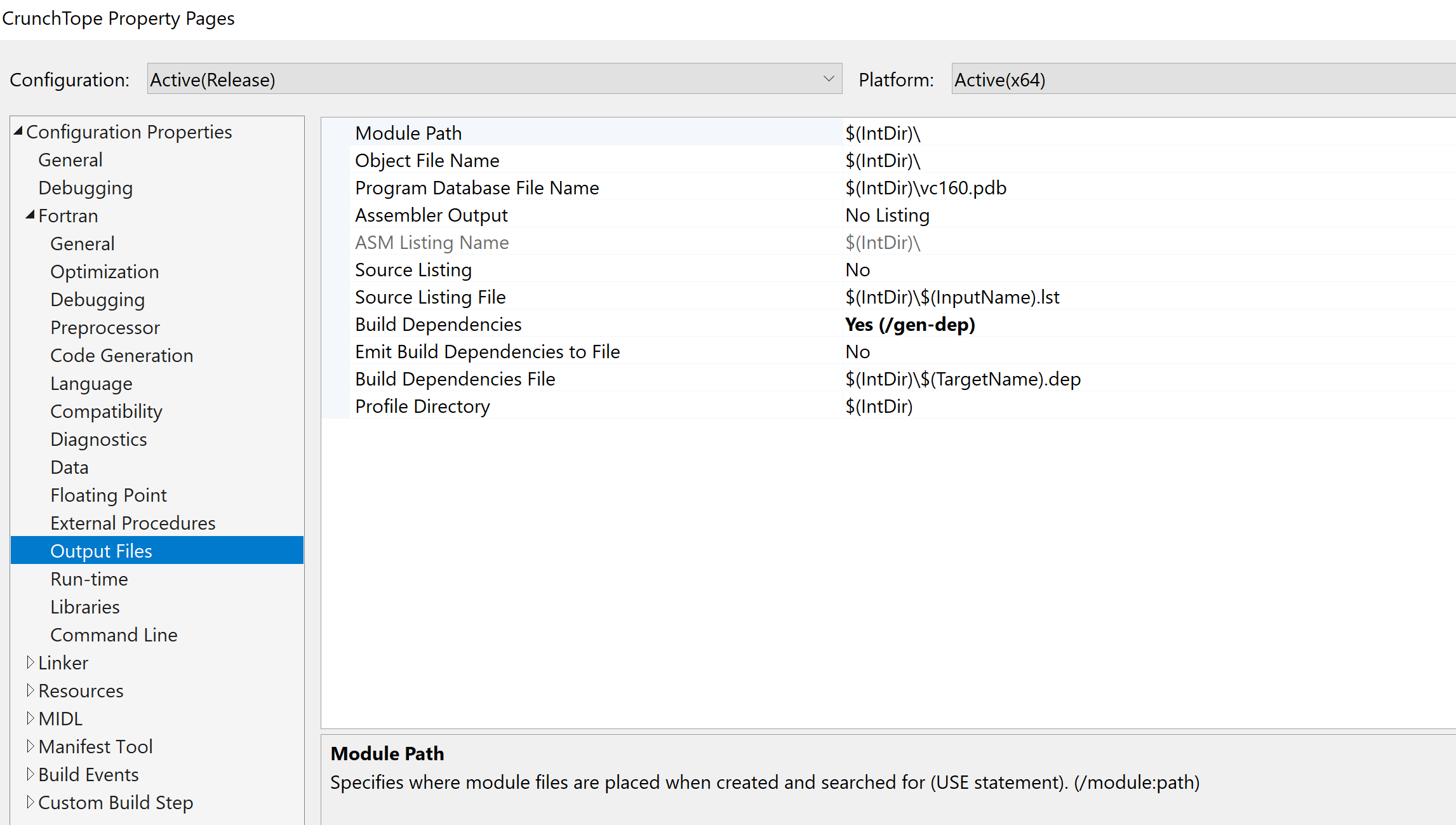
$(PETSC\_DIR)\$(PETSC\_ARCH)\lib

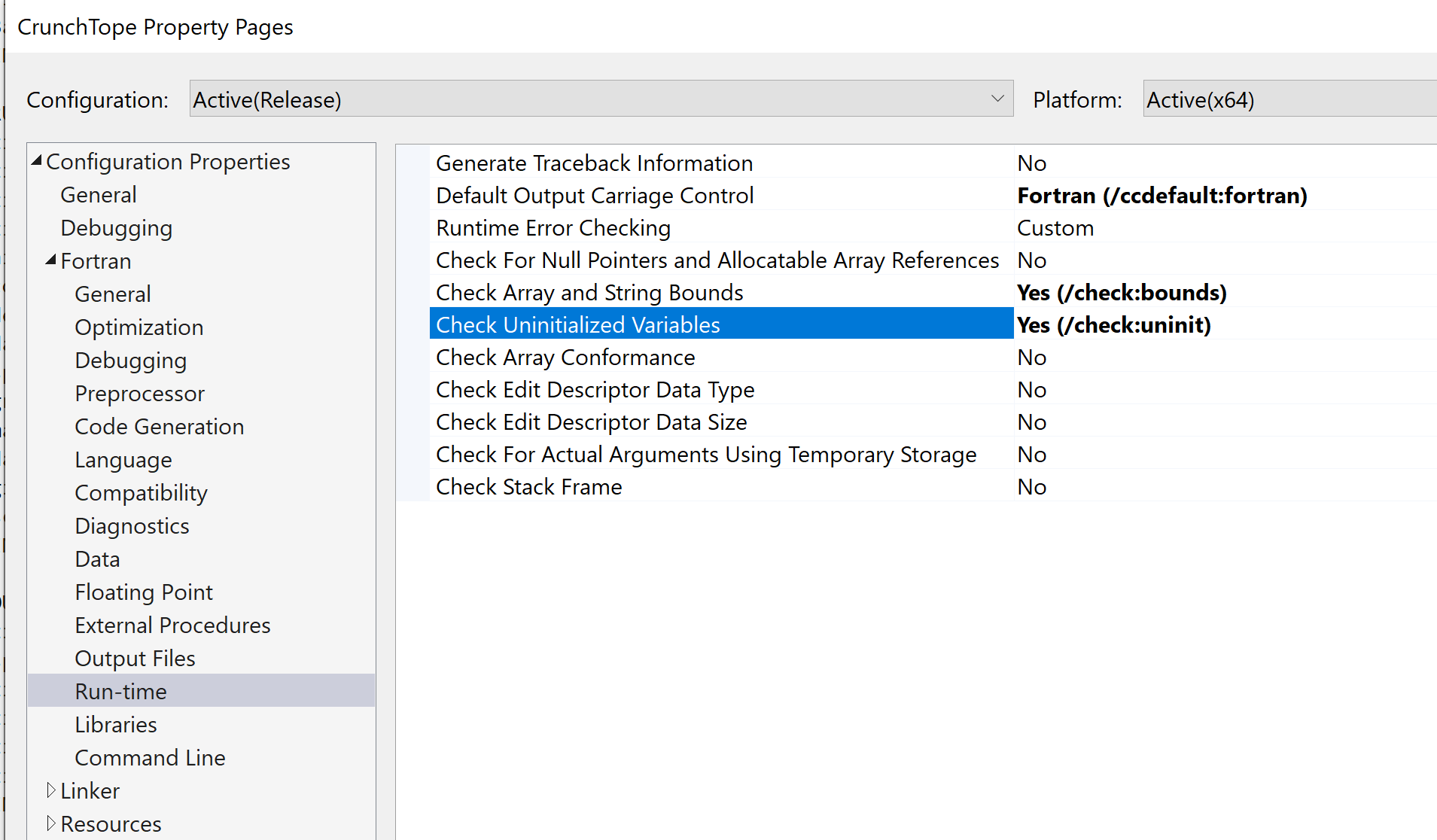
$(I\_MPI\_ONEAPI\_ROOT)\lib\release

$(I\_MPI\_ONEAPI\_ROOT)\include

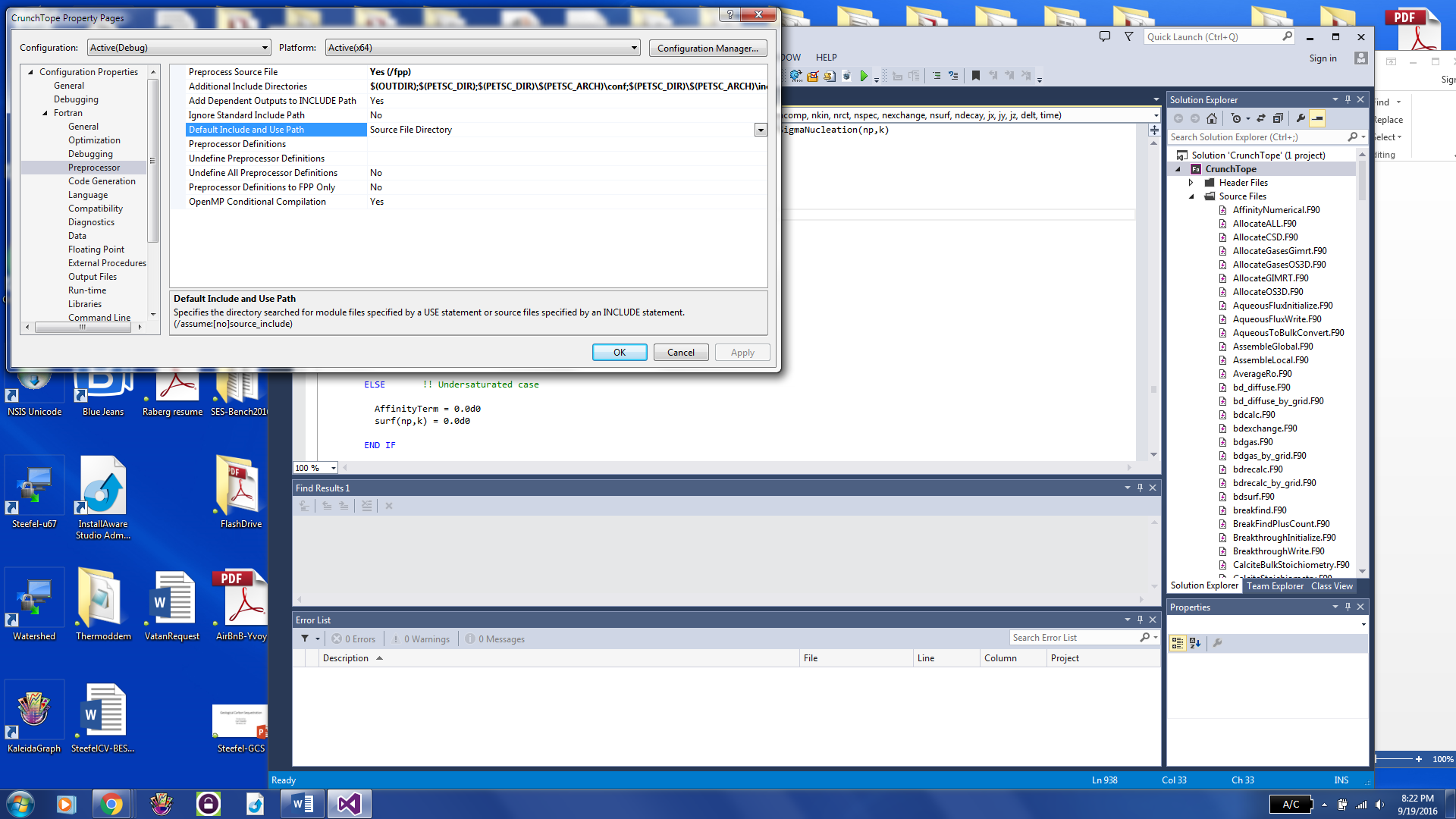
$(ONEAPI\_ROOT)\mkl\2022.1.0\lib\intel64

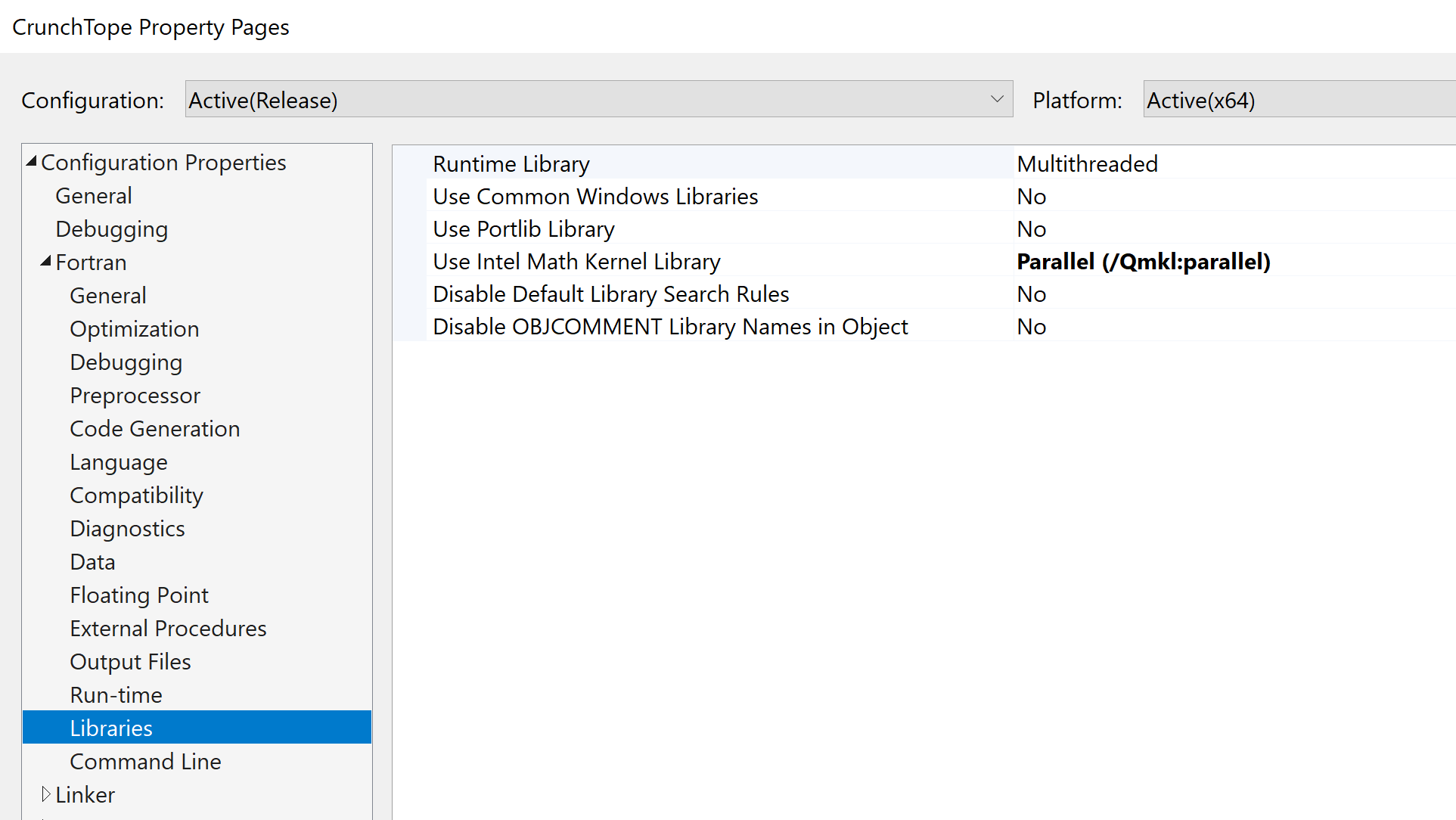
Output file window (choose “Generate Dependencies”)



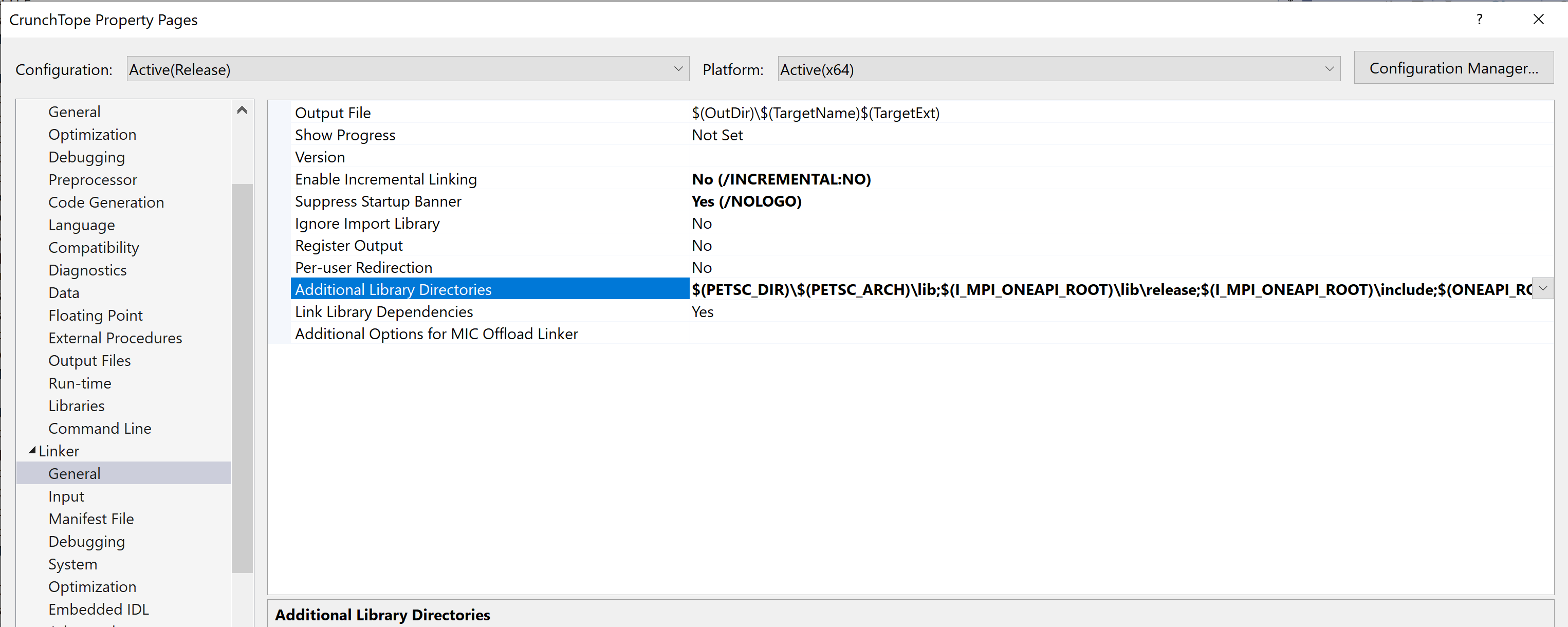


Fortran, Preprocessor





Then on the “Linker” tab, for “General”



And right-click on “Additional Include Directories”

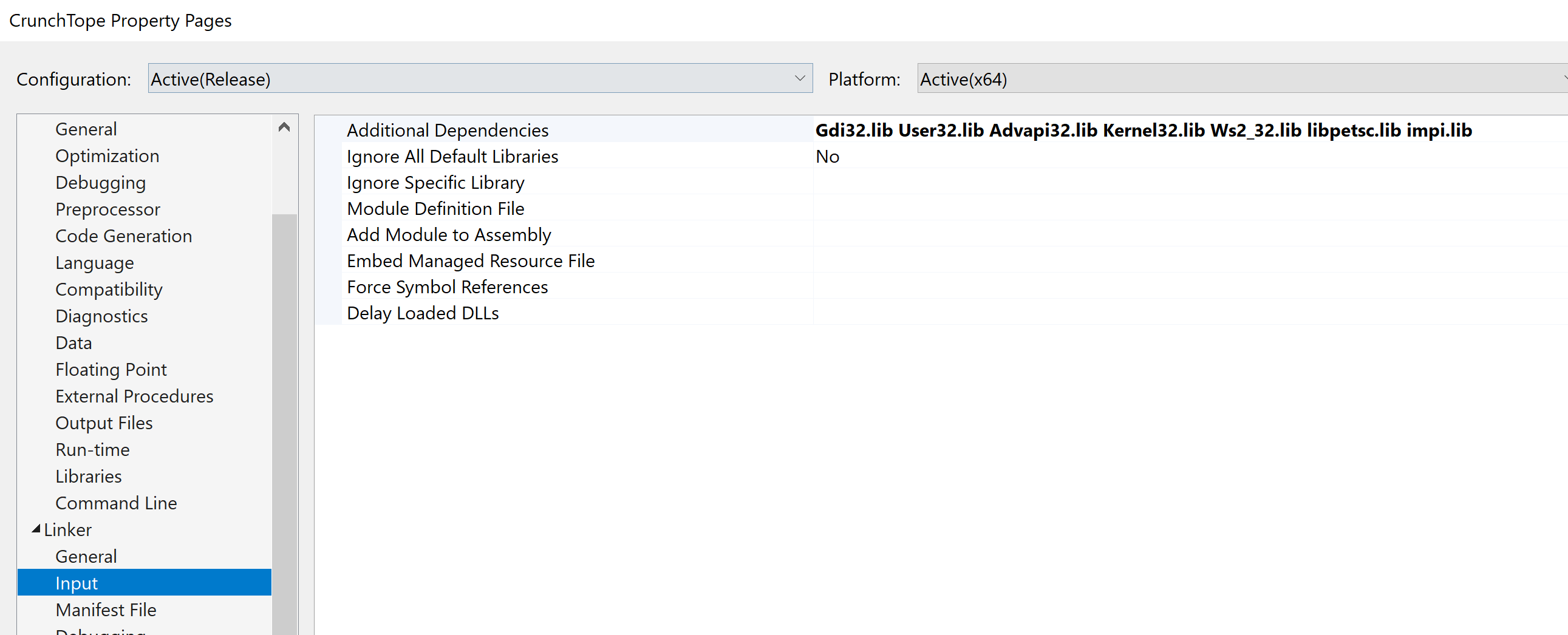
$(PETSC\_DIR)\$(PETSC\_ARCH)\lib

$(I\_MPI\_ONEAPI\_ROOT)\lib\release

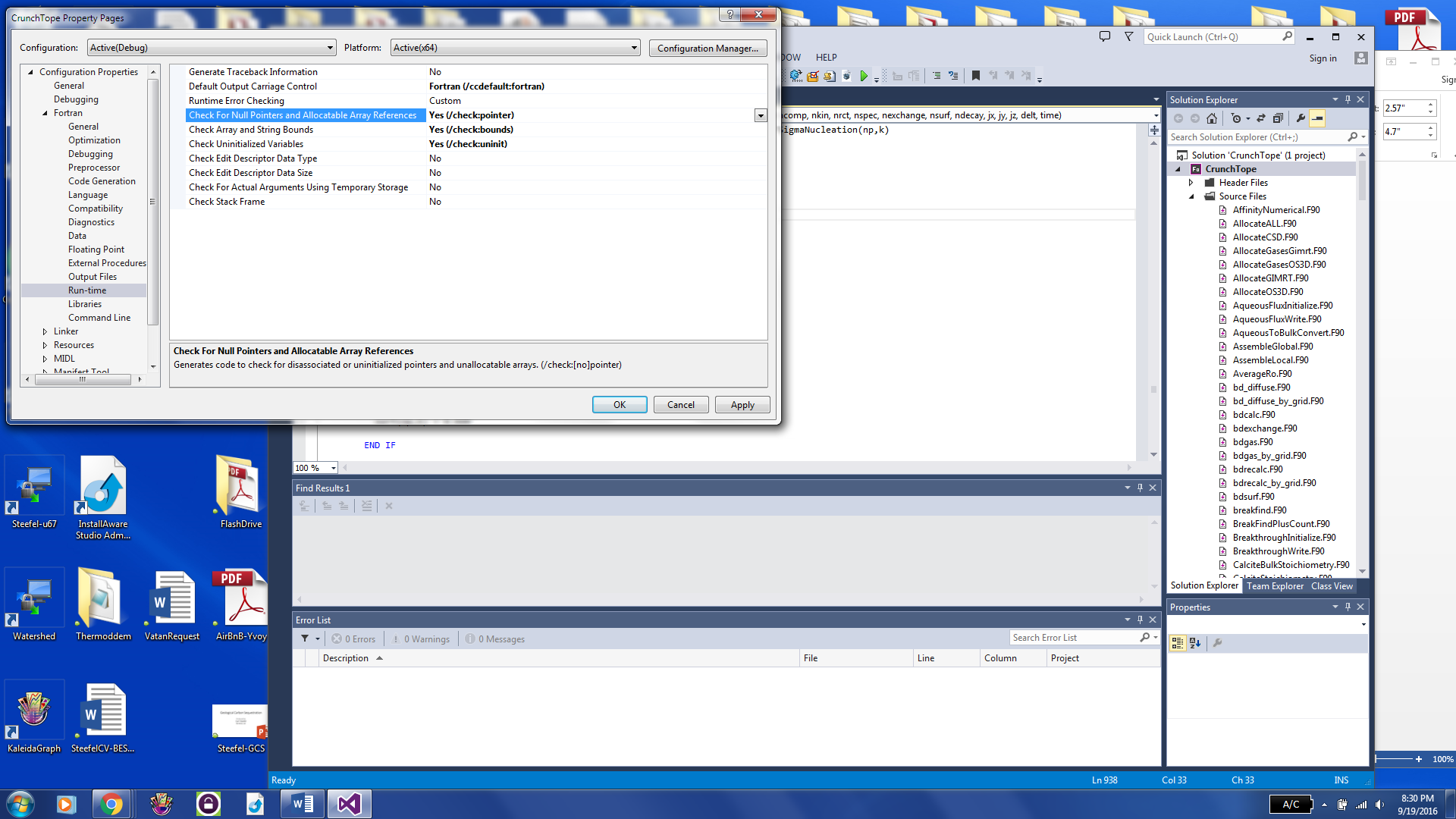
$(I\_MPI\_ONEAPI\_ROOT)\include

$(ONEAPI\_ROOT)\mkl\2022.1.0\lib\intel64

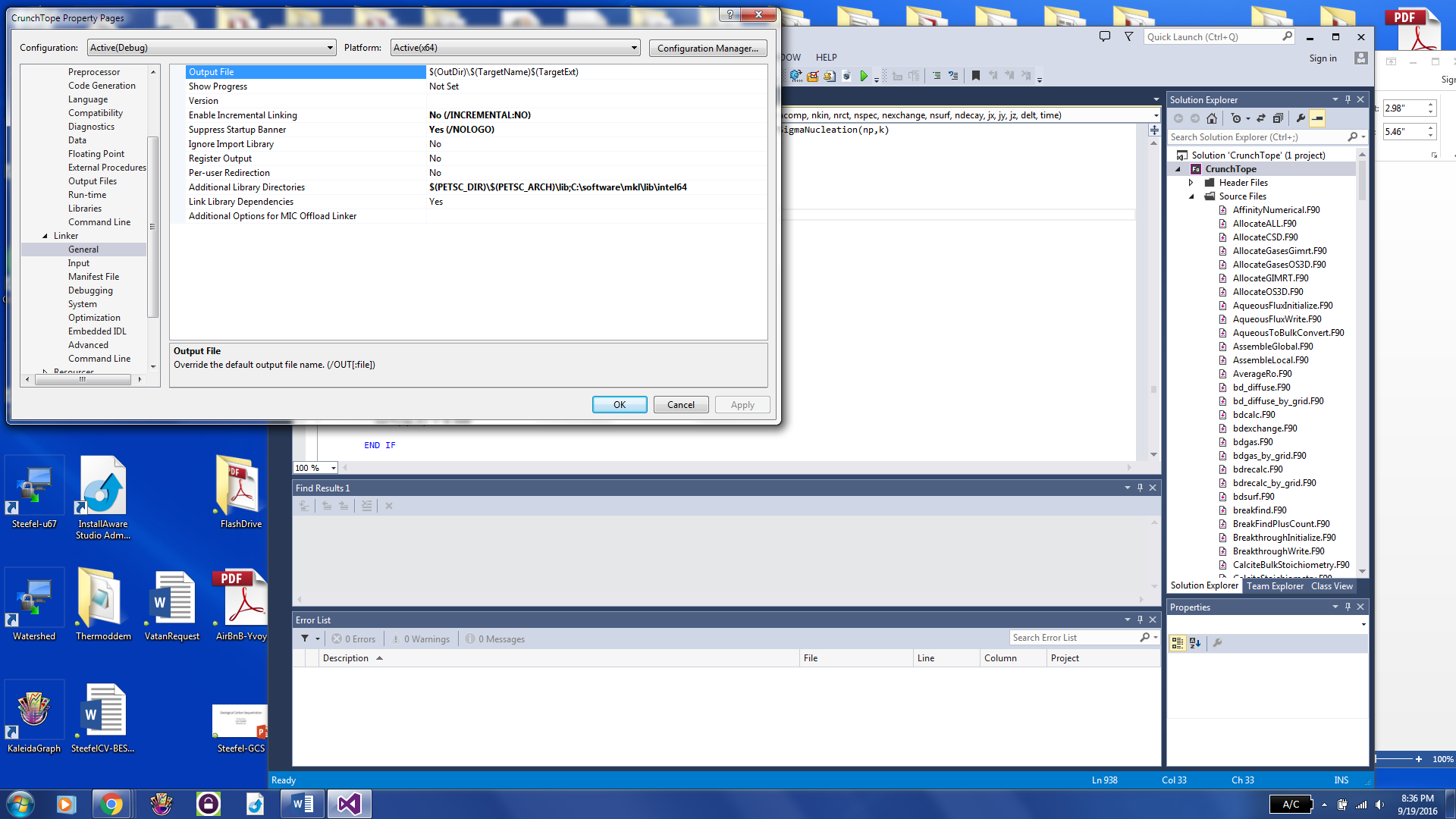
And then “Input”



Fortran, Run-time (for debugging, subscript and unitialized variable checking are useful—turn these off for optimized code)



Linker, General

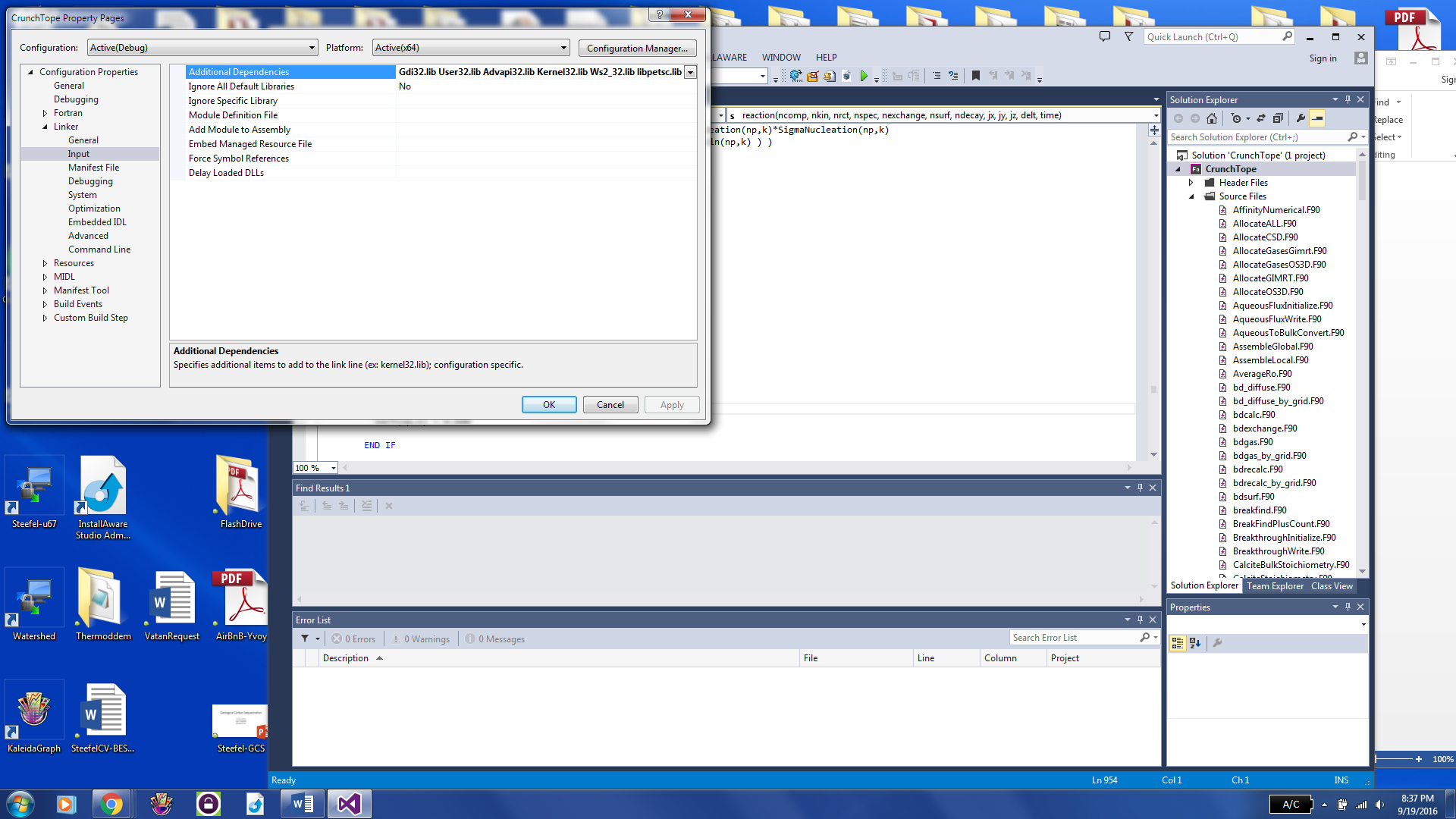


With Additional Library Directories (click on field and select Edit):

$(PETSC\_DIR)\$(PETSC\_ARCH)\lib

C:\software\mkl\lib\intel64

Linker, Input



Other screens should not require modification.

**Building PETSc and CrunchTope on Windows**

Previously it was necessary to purchase Intel C++ and Fortran compilers, but now all of this available free of charge thanks to Intel.

If you want to use the compilers within Microsoft Visual Studio, then you should install Visual Studio 2019, since later versions may not work. People tell me, however, that they fixed the problem with later versions of Visual Studio and Intel oneAPI Fortran. Visual Studio 2019 can be found at:

[Visual Studio 2019 Download](https://my.visualstudio.com/Downloads?q=visual%20studio%202019&wt.mc_id=o~msft~vscom~older-downloads)

Then follow this with installs of (in sequence):

[Intel® oneAPI Base Toolkit](https://www.intel.com/content/www/us/en/developer/tools/oneapi/toolkits.html#base-kit)

[Intel® oneAPI HPC Toolkit](https://www.intel.com/content/www/us/en/developer/tools/oneapi/toolkits.html%23hpc-kit)