VC++ UDF Studio 2022R1 Tutorial

Open Chinese Version

Contents

1. Supported Platform Software Versions

Table 1. Supported platform software versions

Platform versions	Support or not
Win XP~Win10 (x86/x64)	✓
Fluent 6.3~2022R1(x86/x64)	✓
Visual Studio 2008 SP1~2013 Professional or Ultimate (Chs. or Eng.)	✓
Visual Studio 2015 or higher	✗
Intel Visual Fortran 2011~2018 (Optional to install)	✓
Intel Visual Fortran 2019 or higher	✗
Matlab2014a ~2021b (Optional to install)	✓

* Win10 + Visual Studio 2010 Ultimate+ Fluent 17.0 (or higher) are recommended

2. Academic Version vs. Enterprise Version

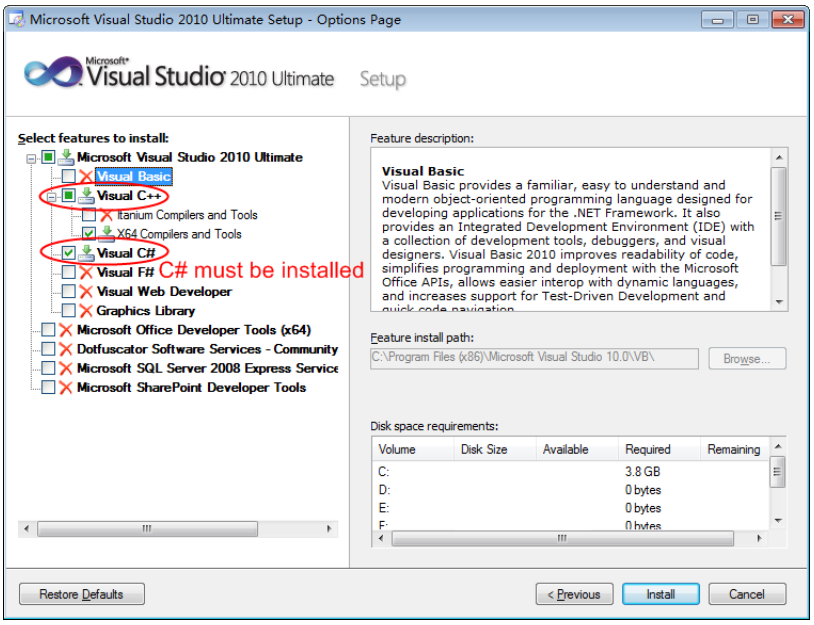
Table 1. Capabilities of academic version vs. enterprise version

Capabilities	Academic Version		Enterprise Version	
	Trial	Registered	Trial	Registered
Compile and debug (serial, single precision)	✓ max. 2 macros	✓ unlimited	✓ max. 2 macros	✓ unlimited
Compile and debug (serial, double precision)	✗	✓	✗	✓
Compile and debug (parallel, single/double precision)	✗	✓	✗	✓
Call C++/Win32 API/MFC functions	✓	✓	✓	✓
Get zone ID from name by UDF	✓	✓	✓	✓
Interrupt iteration by UDF	✓ 2d, 3d serial only	✓ unlimited	✓ 2d, 3d serial only	✓ unlimited
Call 3rd-party library	✓	✓	✓	✓
Set 3rd-party library folders	✗	✗	✓ max. 1 folder	✓ unlimited
Add user menu in Fluent	✗	✗	✓ max. 2 submenus	✓ unlimited
Drive Fluent to iterate by UDF	✗	✗	✓ max. 1 iteration	✓ unlimited
Call Scheme/TUI by UDF	✗	✗	✗	✓
Start by Workbench	✗	✓	✗	✓
Call CoolProp functions (optional to purchase)	✓ max. 1 funcion	✓ unlimited	✓ max. 1 funcion	✓ unlimited
Call Fortran procedures (optional to purchase)	✓ max. 1 procedure	✓ unlimited	✓ max. 1 procedure	✓ unlimited
Call Matlab functions (optional to purchase)	✓ max. 1 function matrix argument forbidden	✓ unlimited	✓ max. 1 function matrix argument forbidden	✓ unlimited

3. Important Notes on Platform Software Installation

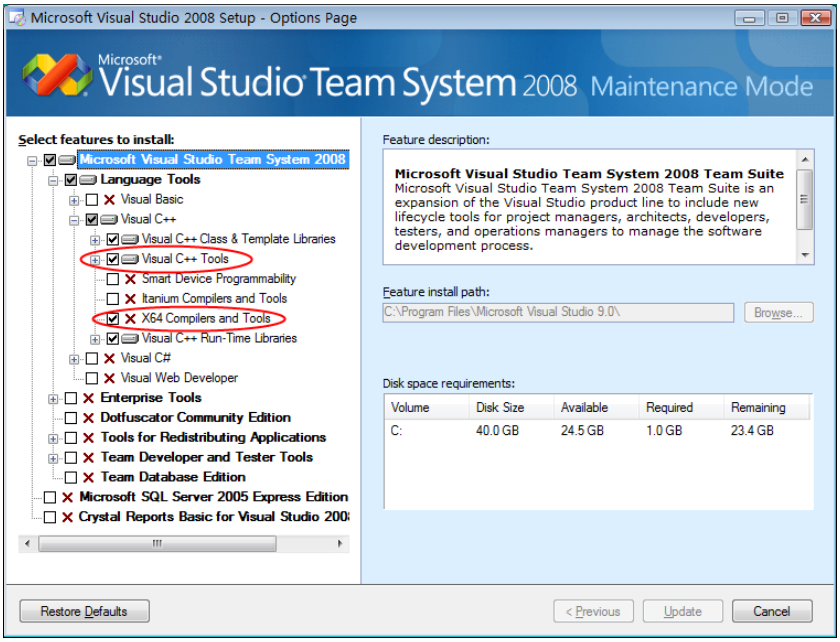
3.1. Notes on Visual Studio Installation

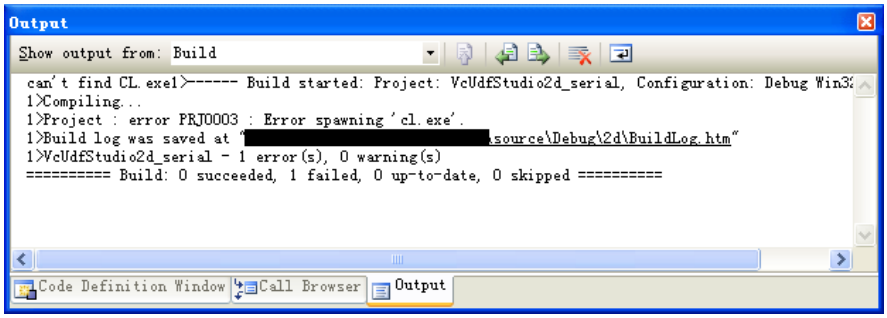
1. Visual C# is recommended to be installed along with Visual C++, though it is not necessary for some Visual Studio version (e.g. VS2008 SP1 ultimate edition). But for VS2010, installation of Visual C# is a must. Otherwise, error will occur when starting Visual Studio.



For VS2008, please assure “Visual C++ Tools” will be installed. And for VS2008 standard edition, Visual C# is also necessary to avoid the “can’t find cl.exe” error. In addition, VS 2008 Service Pack 1 must be installed based on the original version.

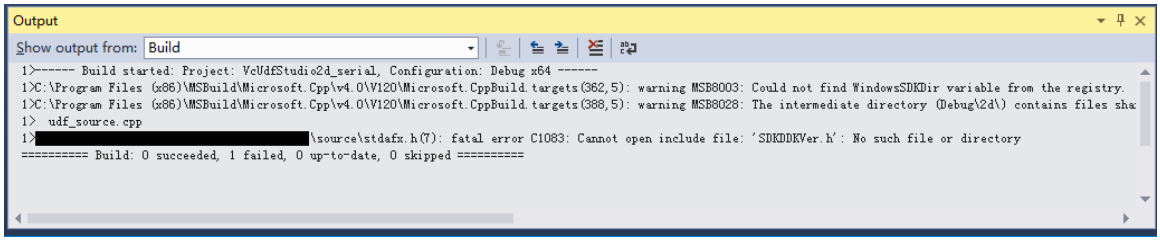
Besides, you have to install 64bit Fluent and assure that “X64 compilers and Tools” will be installed for 64 bit Windows.





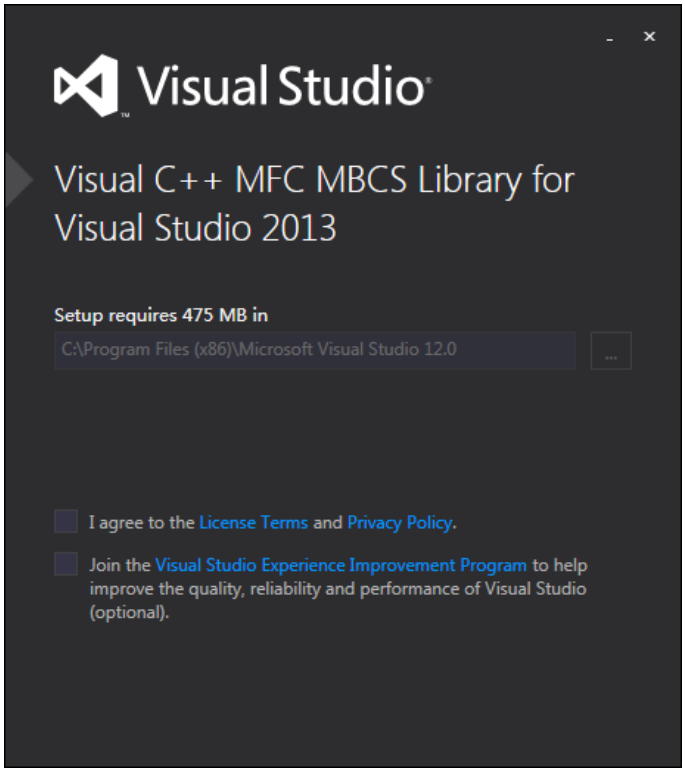
2. If you prefer Visual Studio 2013, please install the latest Visual Studio 2013 update5. Errors may occur if using other earlier 2013 version, such as "cannot open include file 'afxv_cpu.h'" or even chaos in VC++UdfStudio menu.

   Secondly, please first assure the internet connection during installation. Otherwise, "WindowSDKDir" variable may be lost as following figure shows.

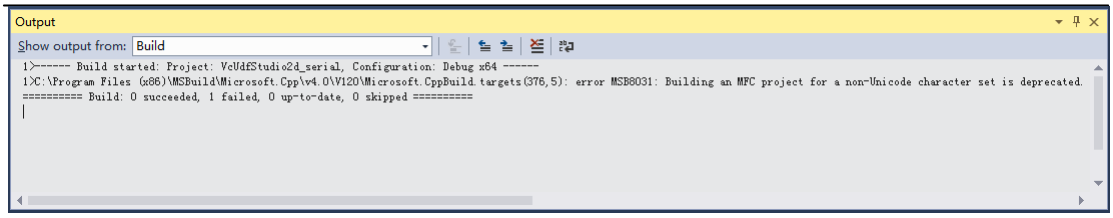


   Last of all, please download and install "Visual C++ MFC Multi-Byte-Character-Set Library" first for Visual Studio 2013.

   Website: https://www.microsoft.com/en-US/download/details.aspx?id=40770

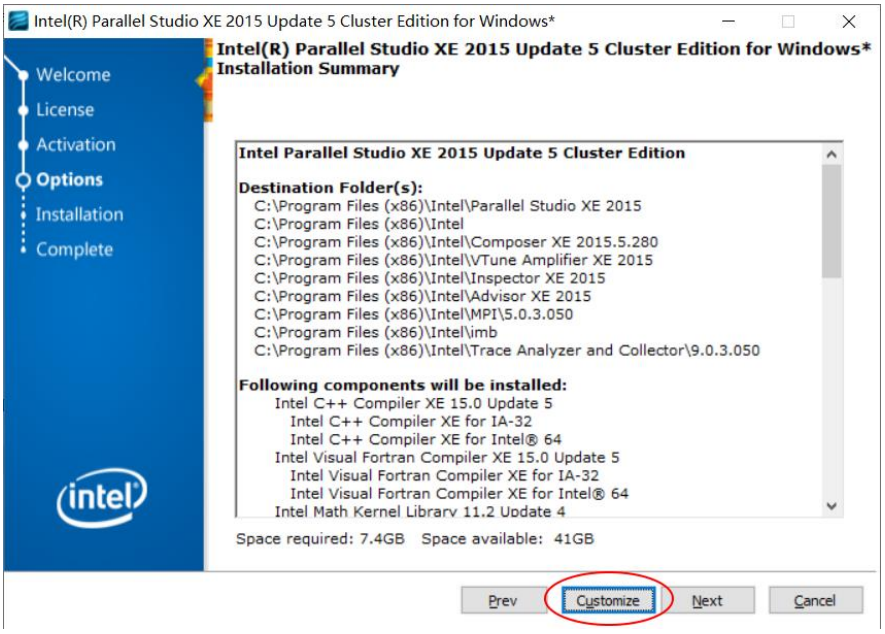


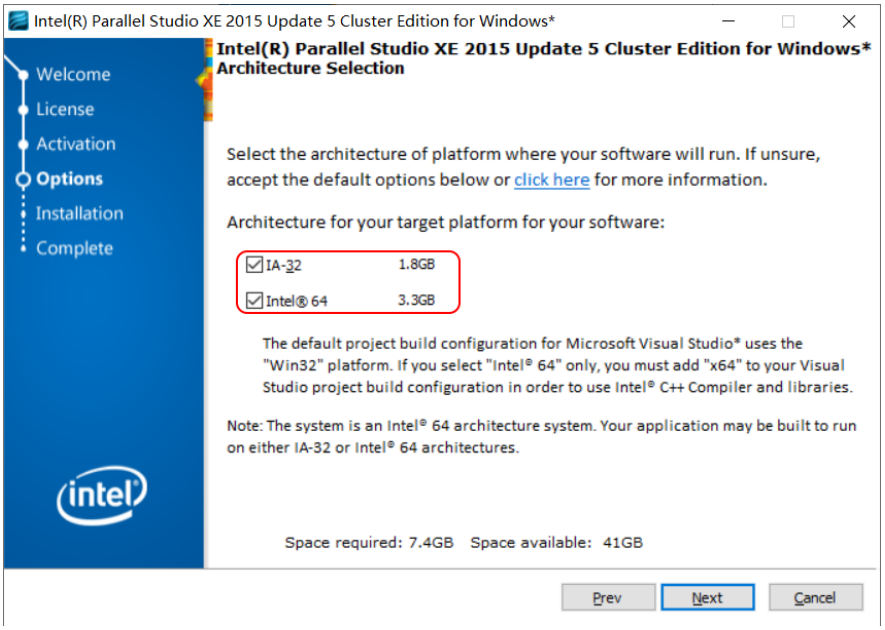
   Otherwise, following error will occur if using Visual Studio 2013.



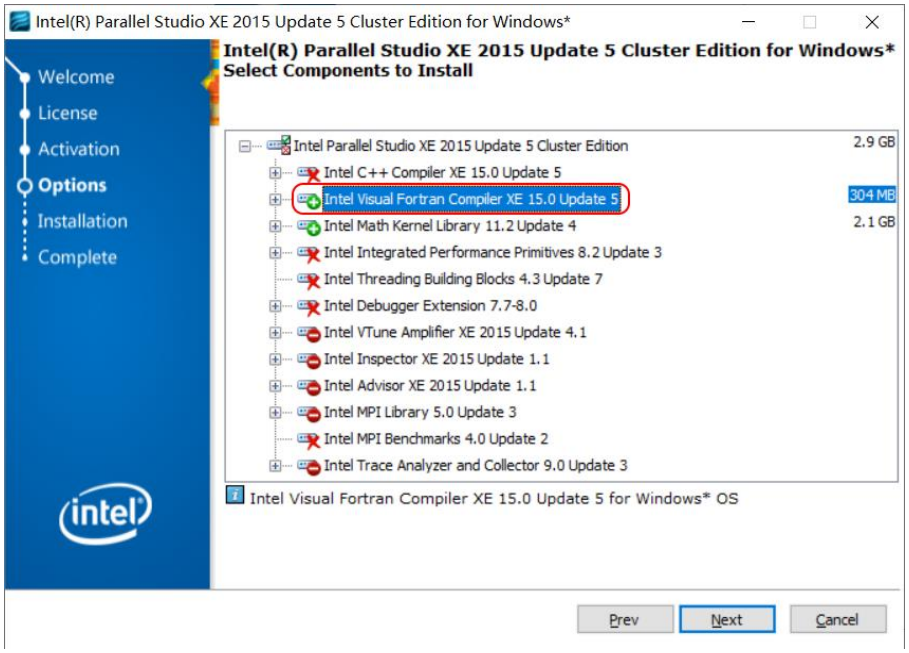
3.2. Notes on Intel Visual Fortran Installation

1. If you want to try the “calling Intel Fortran” capability (or you have purchased this function), then you need to install Intel Visual Fortran besides Visual Studio. The supported versions are listed in table 1. Note that 32/64 bit Fortran compiler should be consistent with the FLUENT architecture (32 or 64bit). For example, if you are using 64bit Fluent, then please install 64bit Fortran compiler.

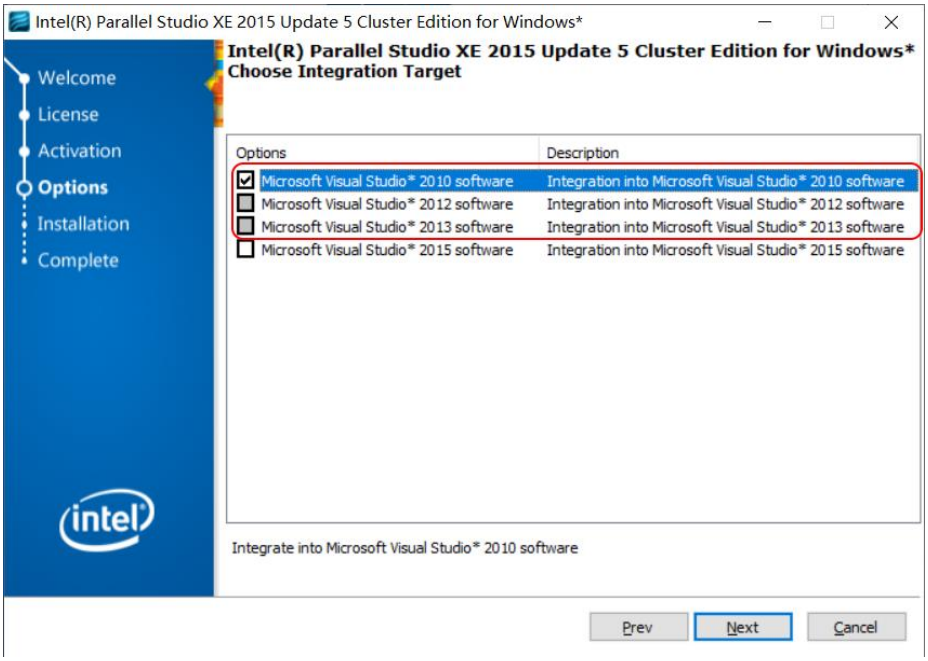




2. Intel Visual Fortran has become a component of Intel Parallel Studio XE package in later version. When selecting customized installation, please check the “Intel Visual Fortran Compiler” option. Others can be chosen as you want.

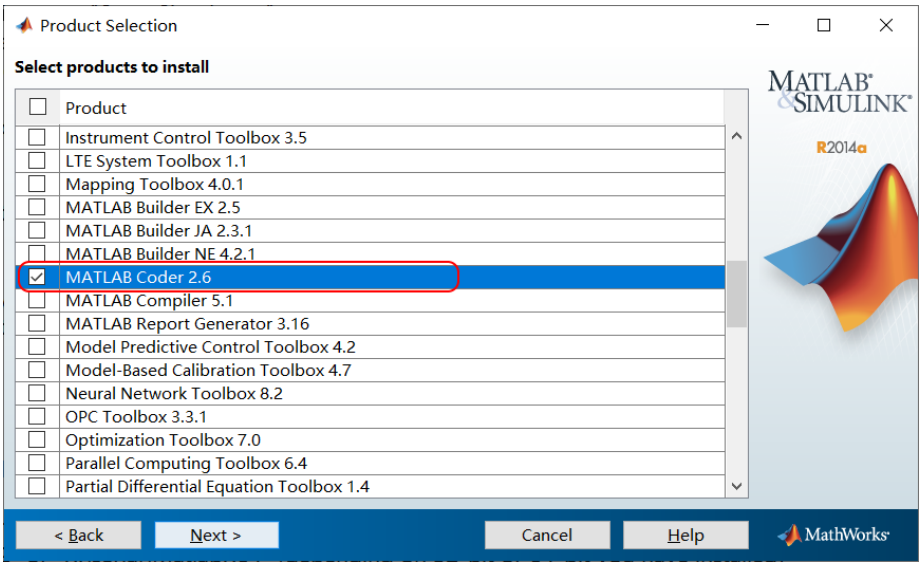


3. In addition, “Integration into Microsoft Visual Studio” should be checked according to the Visual Studio version you installed. Note, Visual Studio may have to be installed first before this step.



3.3. Notes on Matlab Installation

1. If you want to try the “calling Matlab” capability (or you have purchased this function), then you need to install Matlab besides Visual Studio. The supported versions are listed in table 1. Note that Matlab Coder installation is a must, while others are optional.

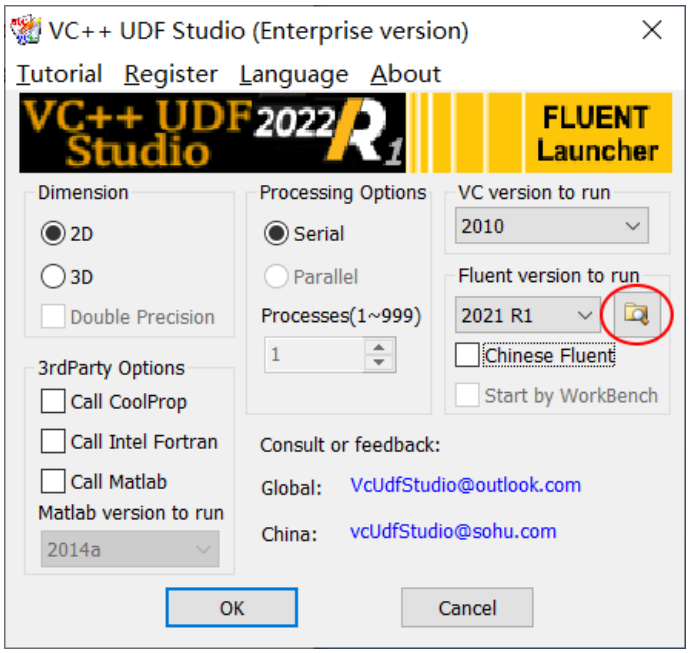


4. Basic steps of UDF Compilation and debug

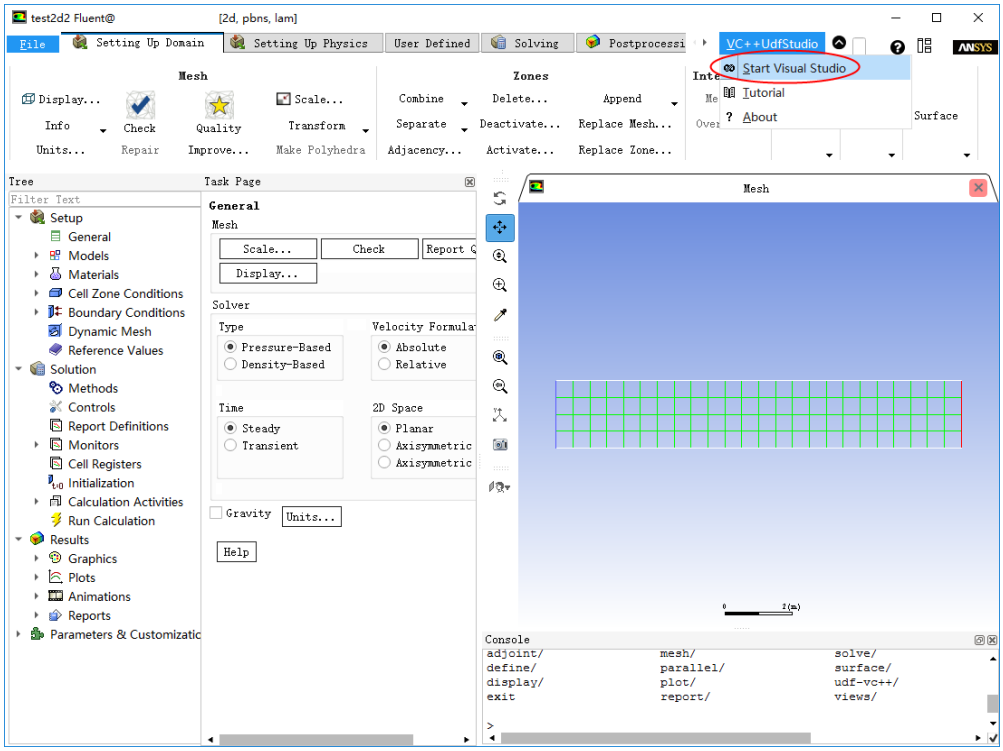
1. Assure "Visual C#", "Visual C++" and “Visual C++ Tools” checked when selecting Visual Studio components to be installed (See previous section "Notes on Visual Studio Installation” for detailed requirements).

2. Install Service Pack 1 if you are using Visual Studio 2008. Other Visual Studio version can skip this step.

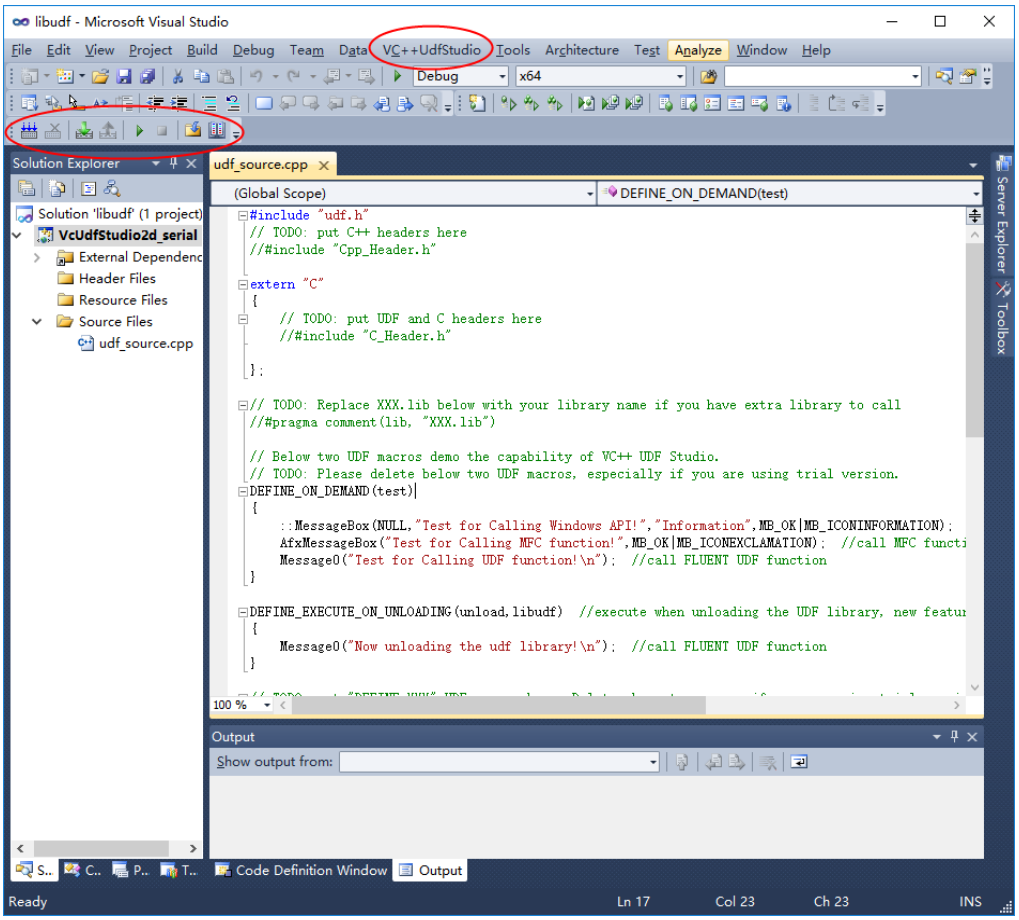
3. Start the “VC++ UDF Studio” launcher, and select the FLUENT version and VC version you’d like to use and click “OK”. Click the "Browse" button to select a FLUENT installation directory whose corresponding version is not in the list.

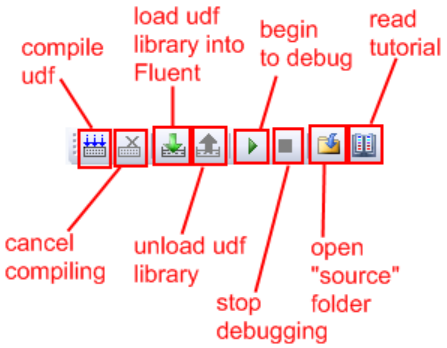


4. Read a Fluent case then click “Start Visual Studio” menu and begin to code UDF.

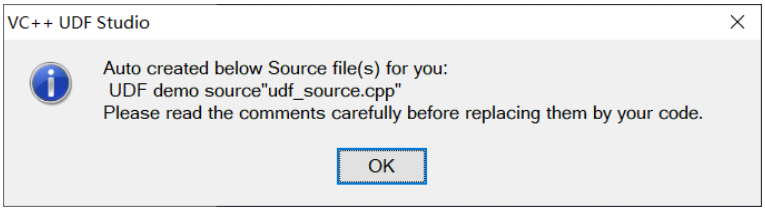


5. “VC++ UDF Studio” toolbar and menu will be shown in the Visual Studio. In the meantime, a folder called “source” will be created in the case directory containing all the source codes.

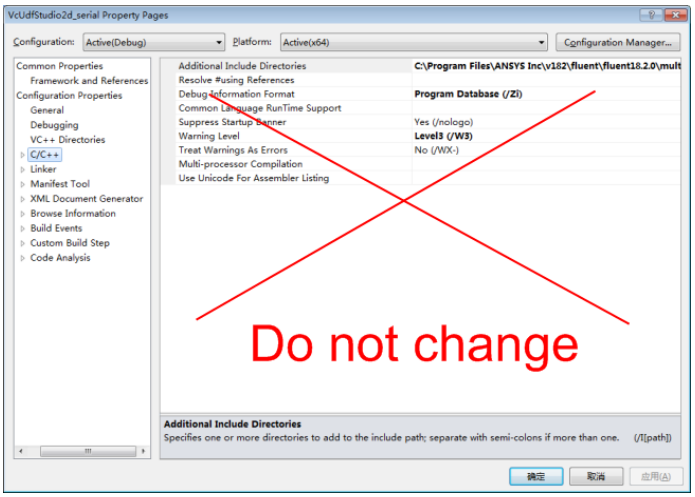




If the UDF source “udf_source.cpp” file does not exist in the “source” folder, a demo file will be created for you. If “udf_source.cpp” file exists in the “source” folder, then it will be opened directly.



Project files “*.vcproj” (VS2008) or “*.vcxproj”(VS2010 and later version) will be auto deleted when Visual Studio closes. Therefore, please never change the project settings. If you want to link additional library "XXX.lib", you can add #pragma comment(lib, "XXX.lib") in “udf_source.cpp”.

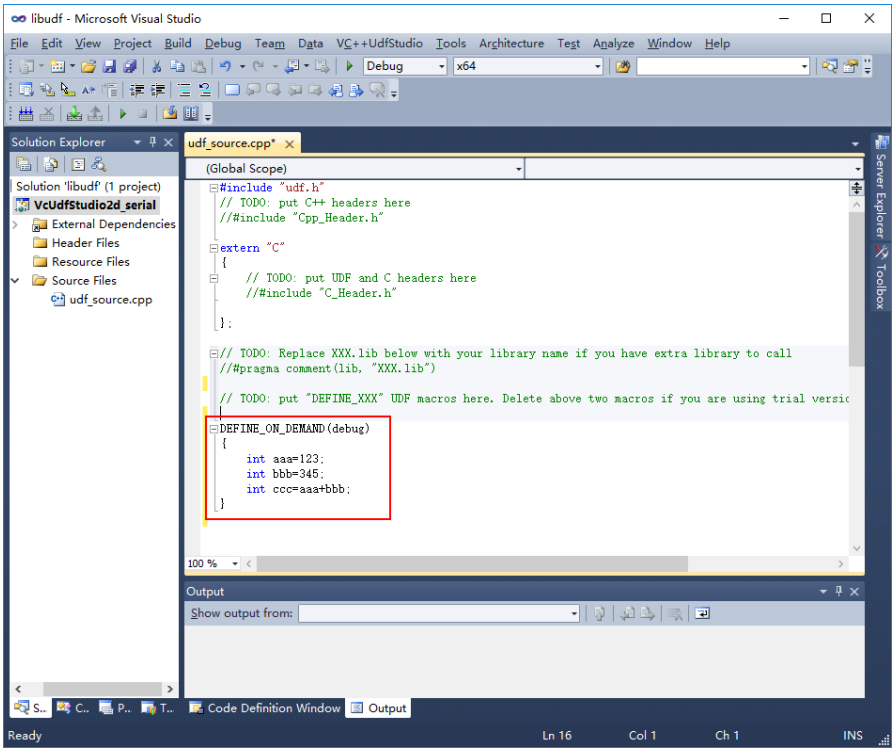


Note: All project files and intermediate folders (e.g., *.sln, *.suo, *.vcproj, *.vcxproj, *.user, *.filters, *.ncb, *.sdf, Debug folder, Release folder) will be auto deleted after Visual Studio closed EXCEPT necessary source files, such as udf_source.cpp, for_source.f90 and *.m.

6. Edit UDF source code. Add following test code to the end of “udf_source.cpp” file. (If you are using trial version, please delete the two demo micros, i.e., DEFINE_ON_DEMAND and DEFINE_EXECUTE_ON_UNLOADING. Otherwise, macro over limitation error will be reported).

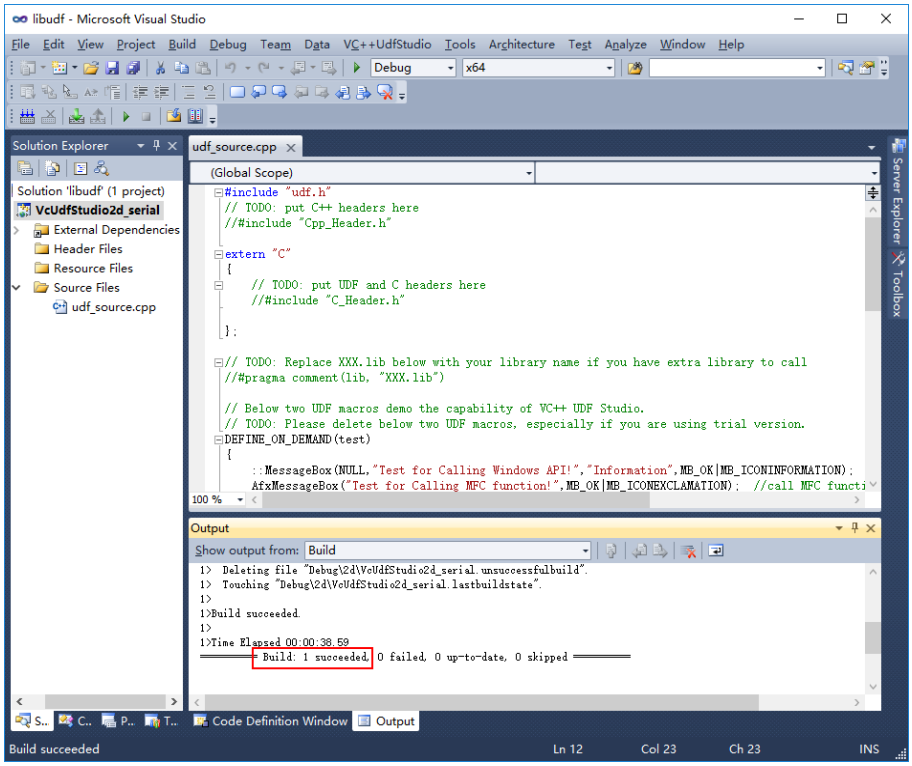
```
DEFINE_ON_DEMAND(debug)
{
    int aaa=123;
```

```
    int bbb=345;
    int ccc=aaa+bbb;
}
```

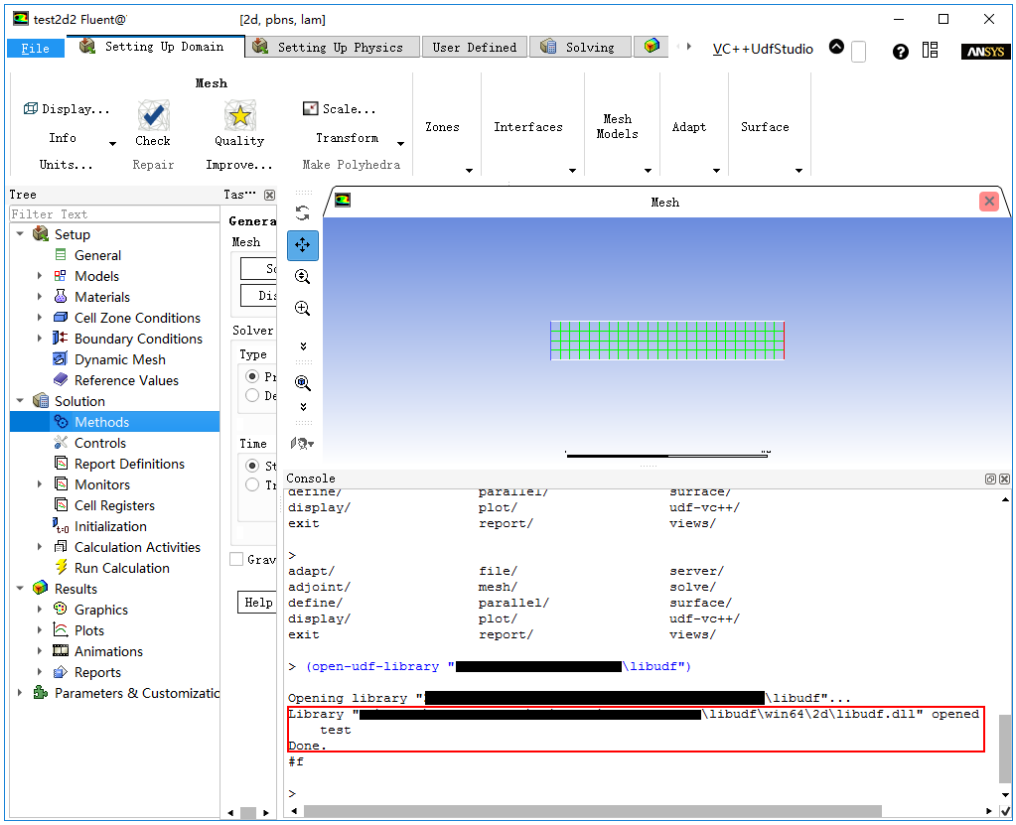


7. Click the “Build UDF library” button (or hotkey “F7”) to compile the UDF source.

Compiling will succeed no syntax errors found (shown in below figure). You can’t load or debug until the source code has been successfully built.



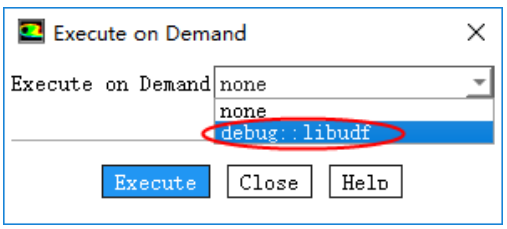
8. Click ”Load UDF library to Fluent” button to load the library to FLUENT. Then, the FLUENT console should show message that “libudf” library is opened.

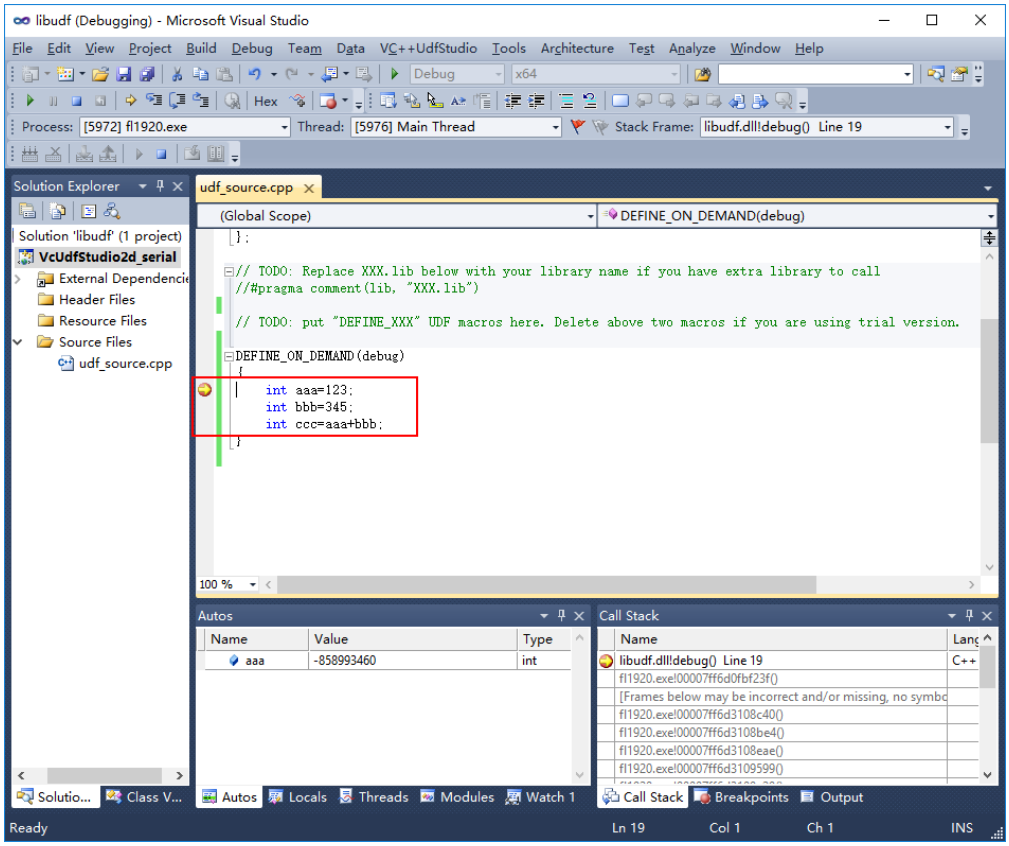


9. Set a breakpoint before “int aaa=123;” (mouse hover on this line and press hotkey “F9”) and press “Start debugging UDF library” button. Of course, directing clicking this button is also OK without loading the library first. This tool will auto load the library first for you. But anyway, UDF library must be generated successfully.

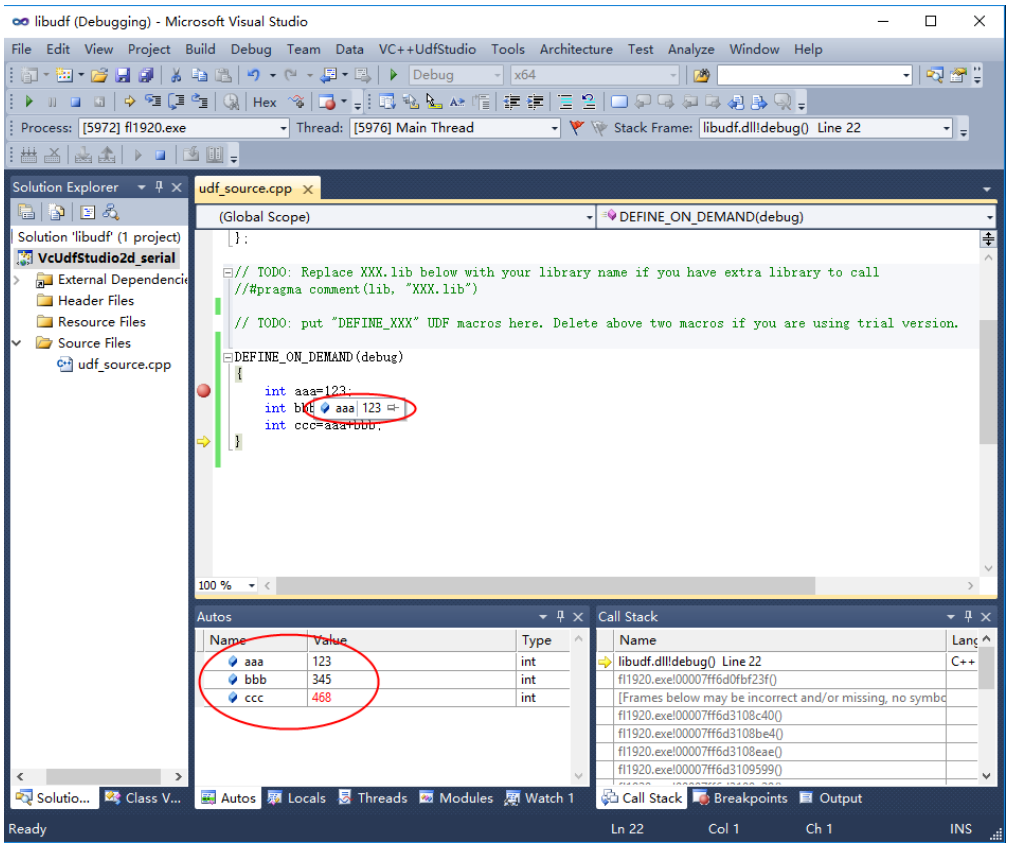
10. Execute “debug::libudf” in Fluent. Visual Studio will auto stop at the breakpoint. Now you can watch all the variable values. Note: Some macros are only called during Fluent iteration, such as DEFINE_SOURCE, DEFINE_PROFILE, etc. Therefore, you not only need to set the breakpoint and push the “debug” button but also have to start Fluent iteration.

Note: Please first understand when Fluent calls the macro you want to debug. If Fluent doesn’t call the macro, breakpoints in the macro won’t work. For example, DEFINE_SOURCE is called during iteration and DEFINE_INIT is called when initialization. If you have set a breakpoint in DEFINE_SOURCE but haven’t started iteration, program won’t stop at the breakpoint.

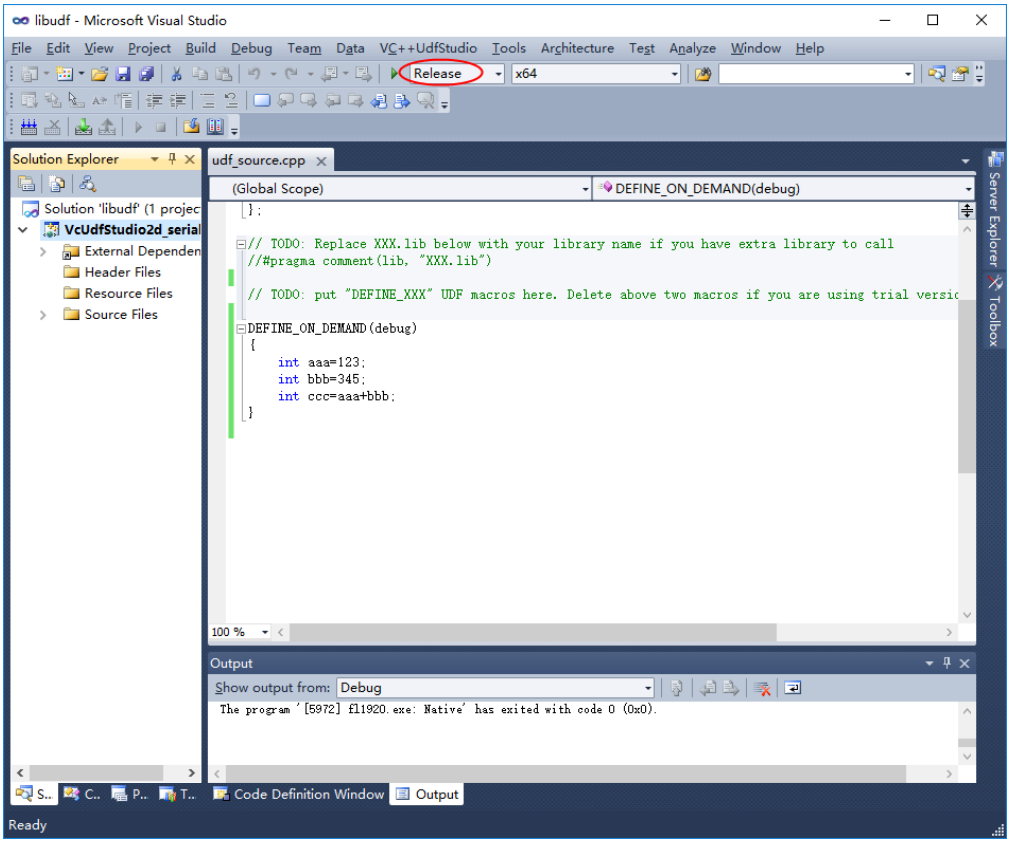




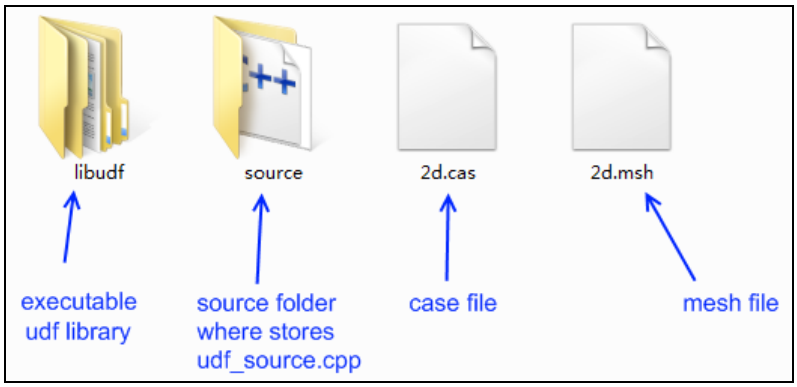
11. Step over the code line by line (or “F10” hotkey). You can observe all variable values just in the usual debugging way.



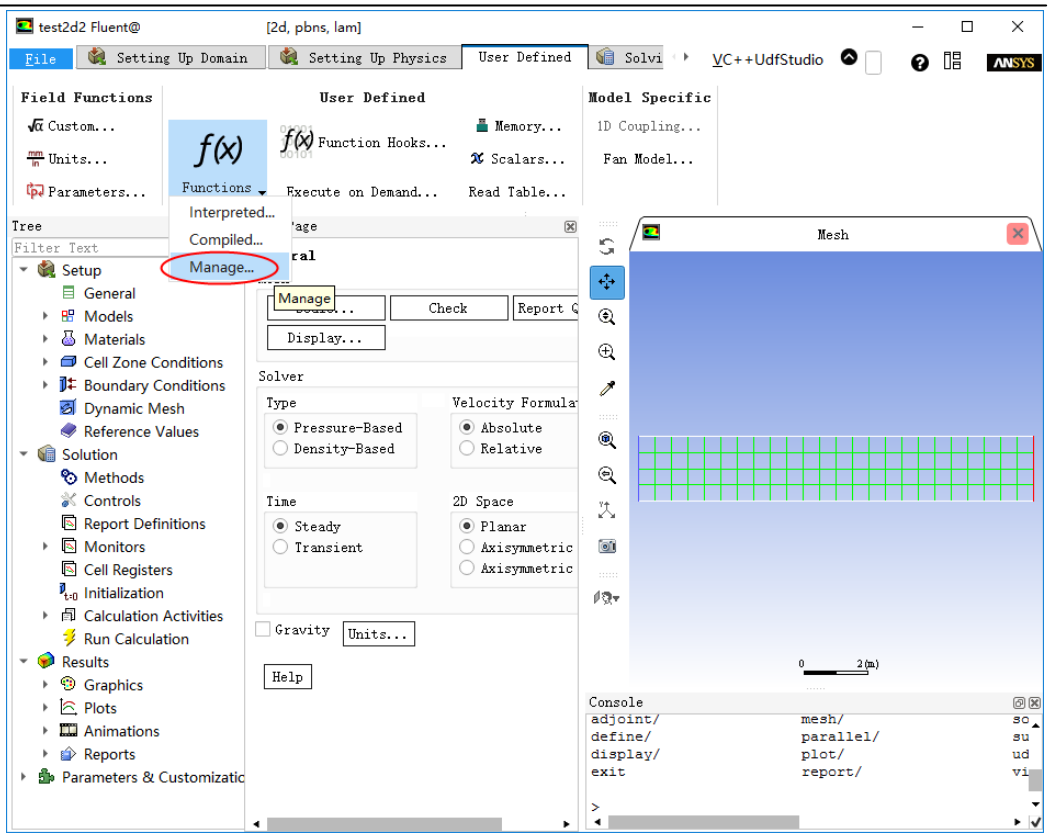
12. After all bugs removed, you can change from Debug to Release and re-compile it in release mode.

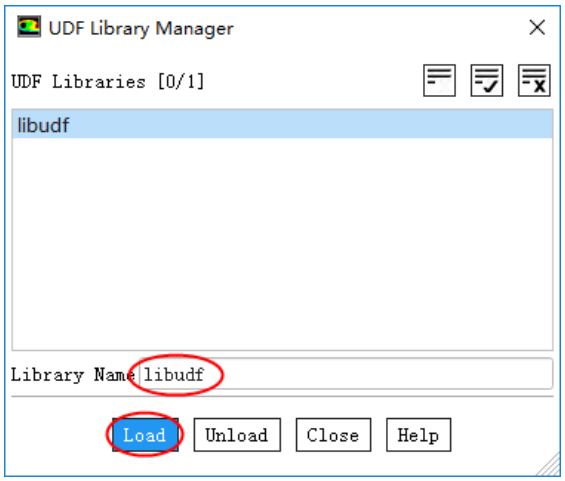


13. Now, your case folder may look like below, where "libudf" folder contains the release version of your UDF library and "source" folder stores the UDF source file "udf_source.cpp".



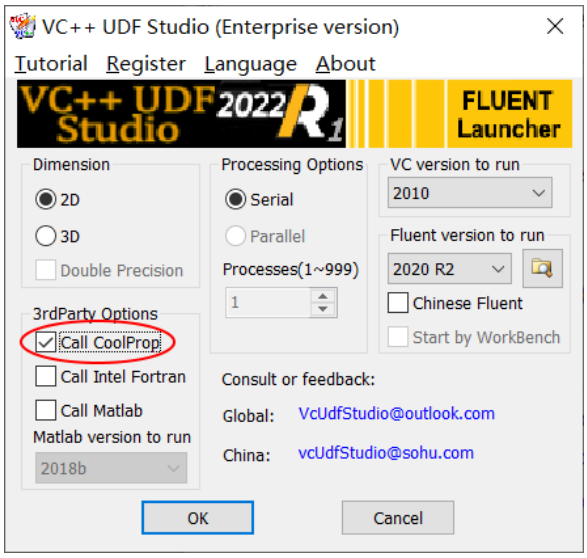
14. After successfully compiling your release version of UDF library, you needn't start Fluent from VC++ UDF Studio launcher if you only want to run the case (not modify/compile UDF source). Just start Fluent in usual way and load the UDF library by "Define->User-Defined->Functions->Manage" menu and type "libudf" in the "Library Name" edit box and press "Load" button.



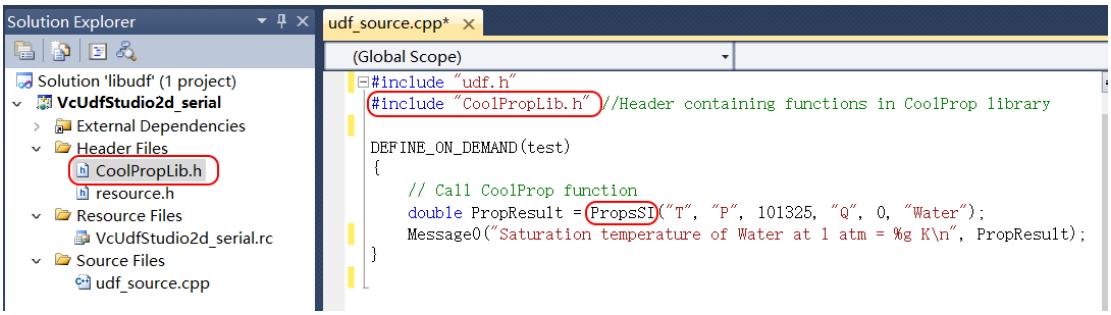


5. Basic steps of Calling CoolProp

1. Check the “Call CoolProp” Option in the launcher (Note: If you have purchased registered version without the “Call CoolProp” feature, then this checkbox will be disabled. You can uninstall this software and re-install it so as to recover to trial version and enable this checkbox).



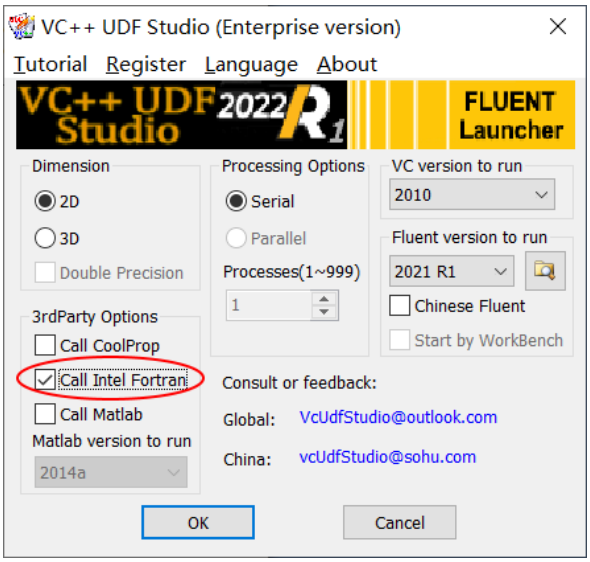
2. Click Ok to start Fluent and read a case. Then start Visual Studio from the menu (See previous section "Basic steps of UDF Compilation and debug"). You will see that "CoolPropLib.h" has been auto added to the project folder. What you should do is to add #include "CoolPropLib.h" into your "udf_source.cpp" file. Then you can use CoolProp functions in the source file "udf_source.cpp".



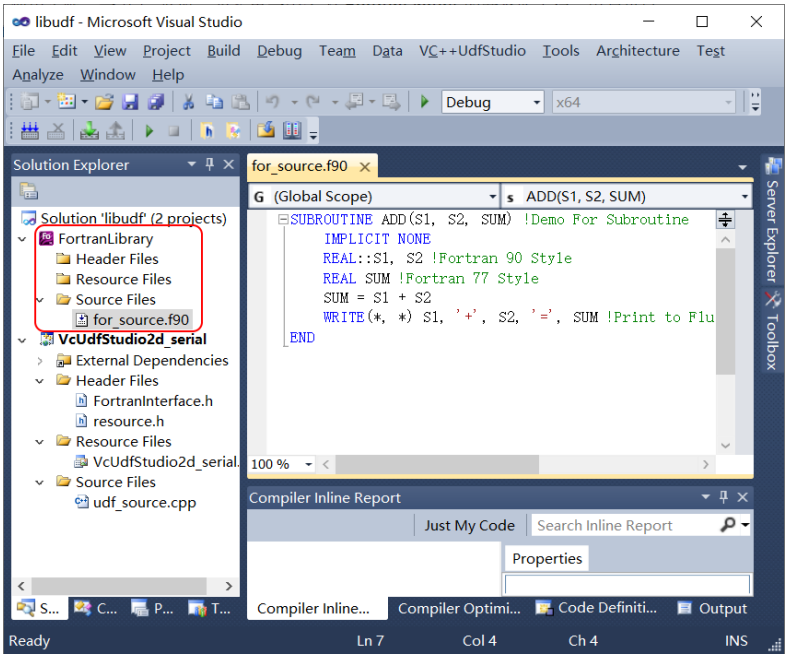
6. Basic steps of Calling Intel Fortran

1. Install supported version of Intel Visual Fortran (See previous section "Notes on Intel Visual Fortran Installation" for detailed requirements).

2. Check the "Call Intel Fortran" Option in the launcher (Note: If you have purchased registered version without the "Call Intel Fortran" feature, then this checkbox will be disabled. You can uninstall this software and re-install it so as to recover to trial version and enable this checkbox).



3. Click Ok to start Fluent and read a case. Then start Visual Studio from the menu (See previous section "Basic steps of UDF Compilation and debug"). A new static library project named "FortranLibrary" will be added to the solution. The user can add Fortran functions in the source file "for_source.f90".



4. Edit the head file "FortranInterface.h" according to the functions in "for_source.f90". This head file should include the C declaration of the Fortran functions to be called. Table 3 is an example of Fortran and C/C++ corresponding common types. Note that the Fortran arguments should be called by reference. Therefore, the C declaration corresponds to pointer type. For example the demo Fortran source:

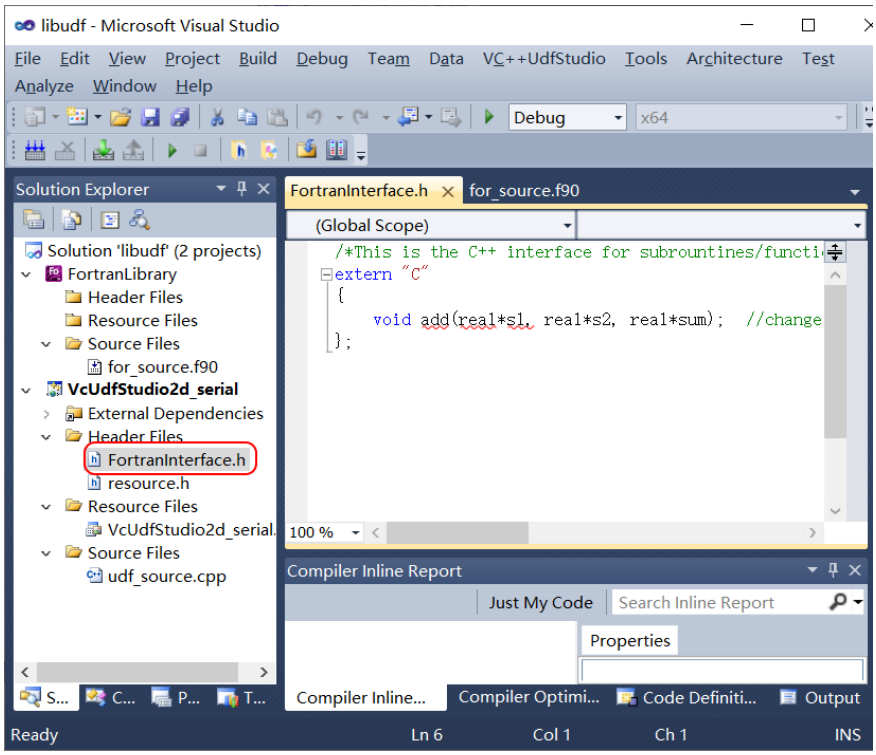
```
SUBROUTINE ADD(S1, S2, SUM)
REAL::S1, S2
REAL::SUM
```

Corresponding C declaration is:

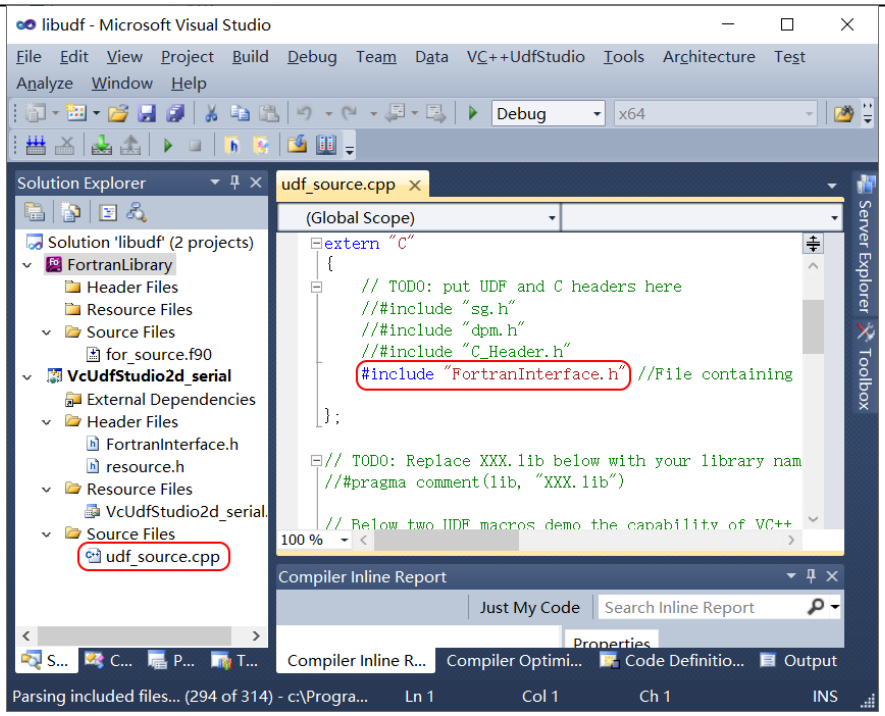
```
void add(real*s1, real*s2, real*sum);
```

Table 3. Fortran and C/C++ common corresponding types

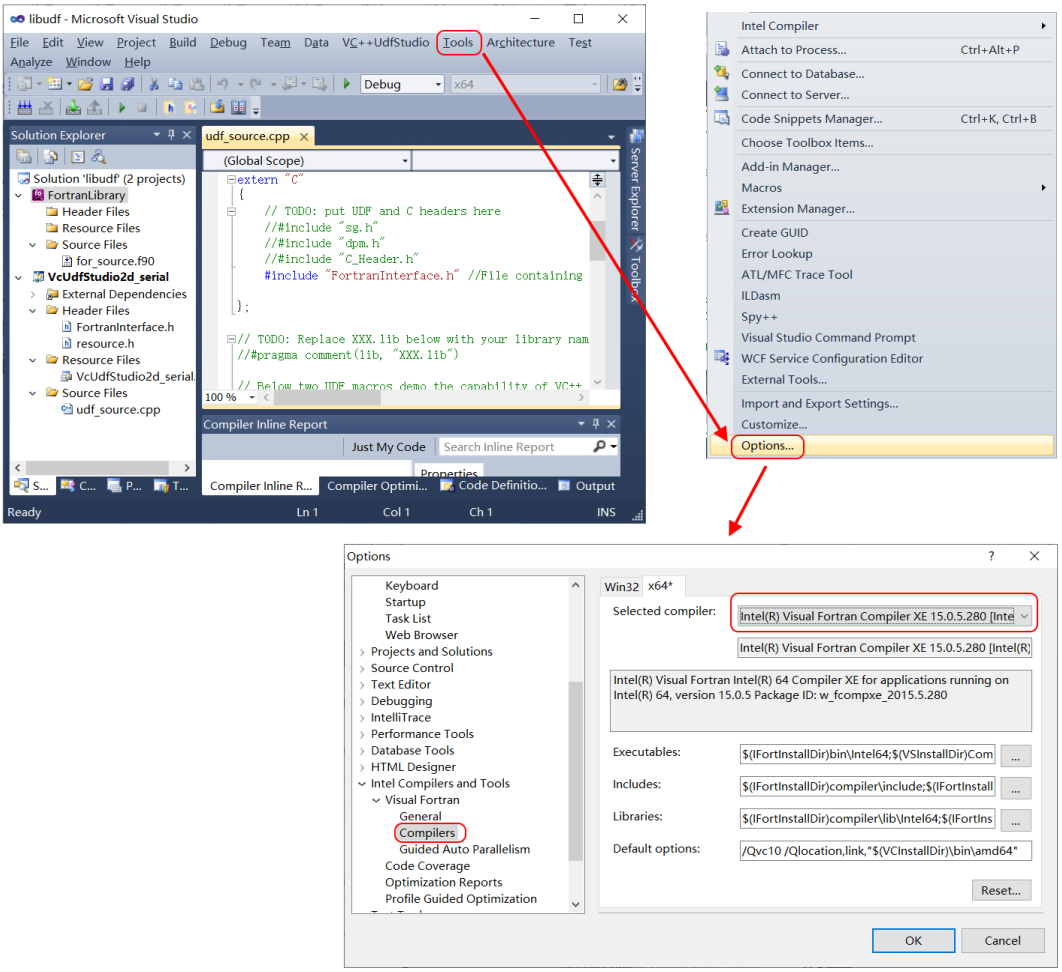
FORTRAN	C/C++
byte	unsigned char
integer*2	short int
integer	long int or int
integer iabc(2,3)	int iabc[3][2];
logical	long int or int
logical*1	bool (C++, one byte)
real	float (can be real type in Fluent UDF)
real*8	double (can be real type in Fluent UDF)
real*16	long double
complex	struct{float realnum; float imagnum;}
double complex	struct{double dr; double di;}
character*6 abc	char abc[6];
character*6 abc(4)	char abc[4][6];
parameter	#define *PARAMETER value*



5. Add *#include "FortranInterface.h"* to udf_source.cpp file so that Fortran functions can be called.



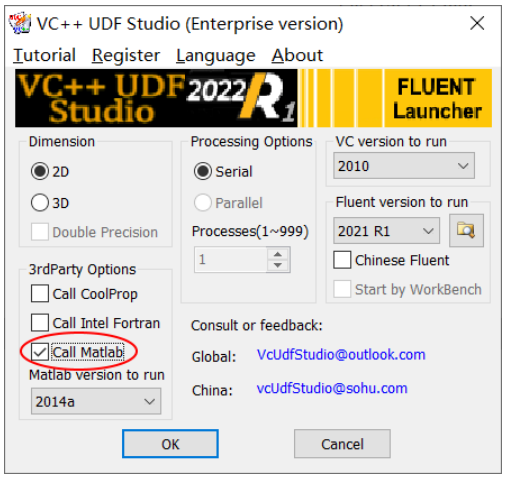
6. Click the “build” button to compile. If you have installed multiple Intel Fortran versions, then you can set the ready-to-use version in *Tools->Options->Intel(R) Visual Fortran* menu.



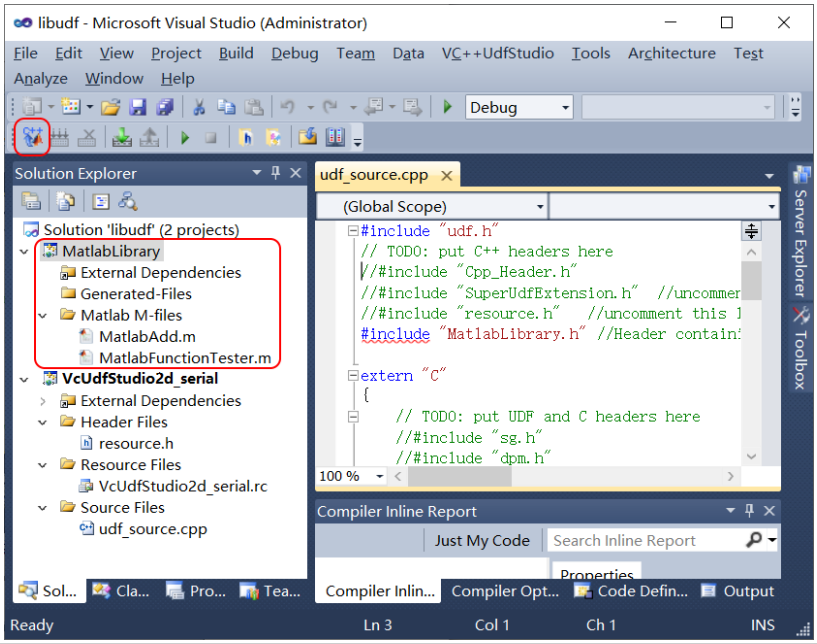
7. Basic steps of Calling Matlab

1. Install supported version of Matlab (See previous section "Notes on Matlab Installation" for detailed requirements).

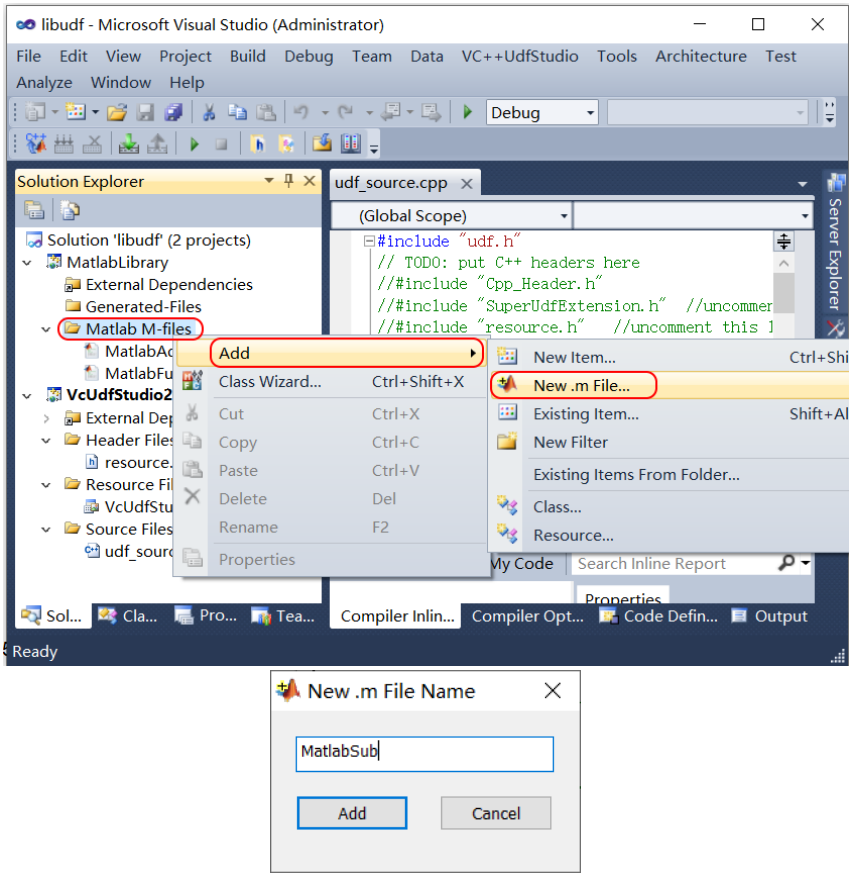
2. Check the "Call Matlab" Option in the launcher and select the Matlab version to use (Note: If you have purchased registered version without the "Call Matlab" feature, then this checkbox will be disabled. You can uninstall this software and re-install it so as to recover to trial version and enable this checkbox). In addition, if you are changing the Matlab version that has been called before, please run the launcher as administrator.



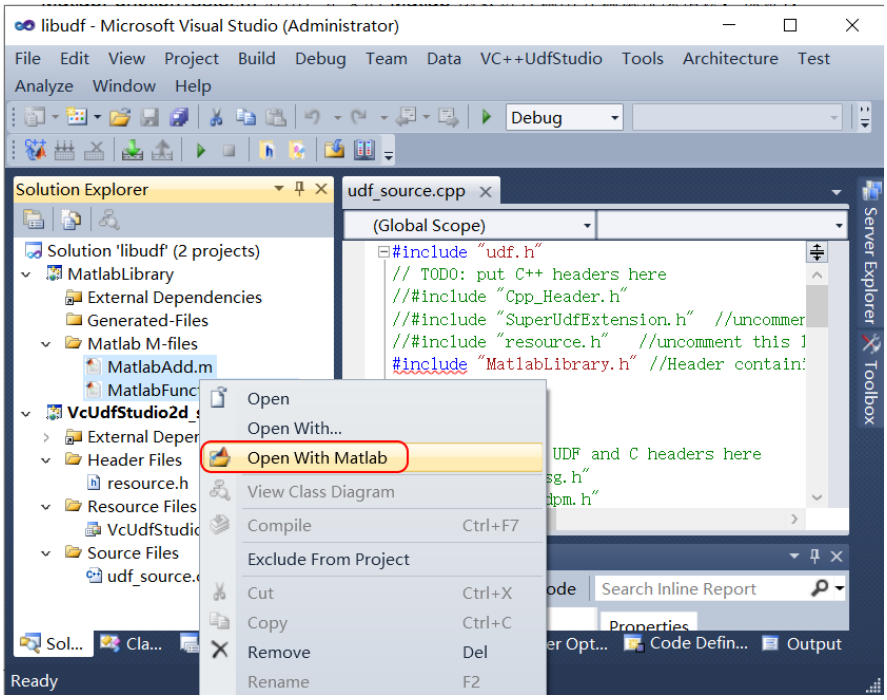
3. Click Ok to start Fluent and read a case. Then start Visual Studio from the menu (See previous section "Basic steps of UDF Compilation and debug"). A new toolbar button "Convert .m files to Cpp" and a new static library project named "MatlabLibrary" will be shown for the solution. The user can add Matlab functions in the folder "Matlab M-files". Note that the MatlabFunctionTester.m file is designed to debug Matlab functions in Matlab environment, which cannot be deleted (detailed operations will be introduced later).

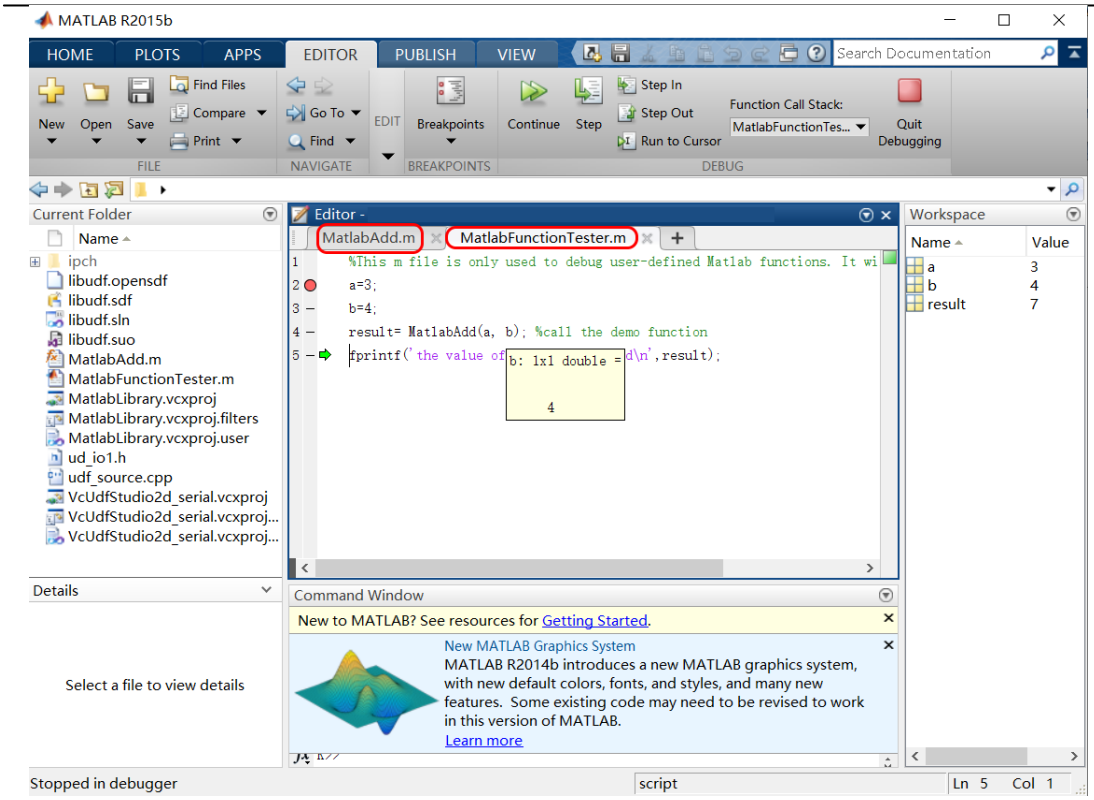


4. Right click on the “Matlab M-files” folder so as to add new m file, i.e, new Matlab function (because Matlab requires separate file for each function).

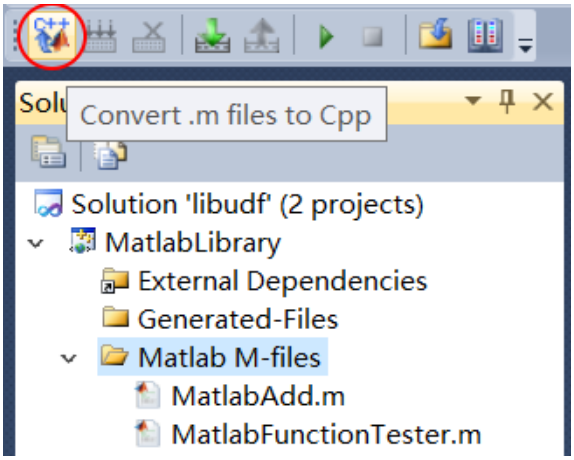


5. Right click on m file/files and select “Open with Matlab” so that the user can debug Matlab functions that are called by “MatlabFunctionTester.m” in Matlab. After removing all bugs, close Matlab.

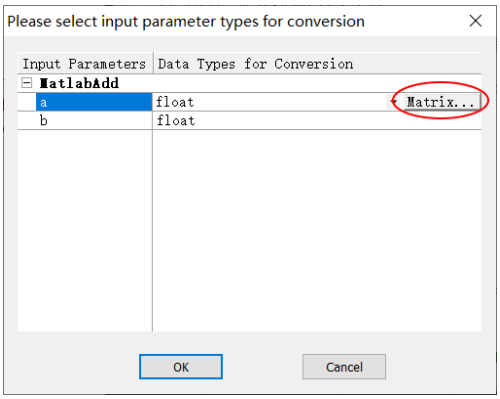


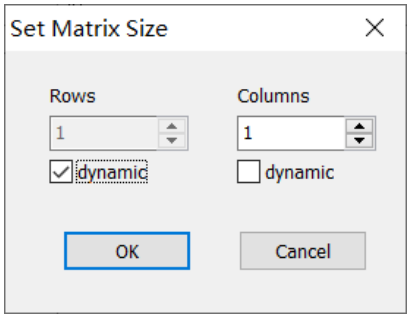


6. Click the “Convert .m files to Cpp” toolbar button in order to convert Matlab functions (except the MatlabFunctionTester.m file) to C/C++ files. Note that some Matlab functions are not supported for conversions, such as plot, eval, etc.

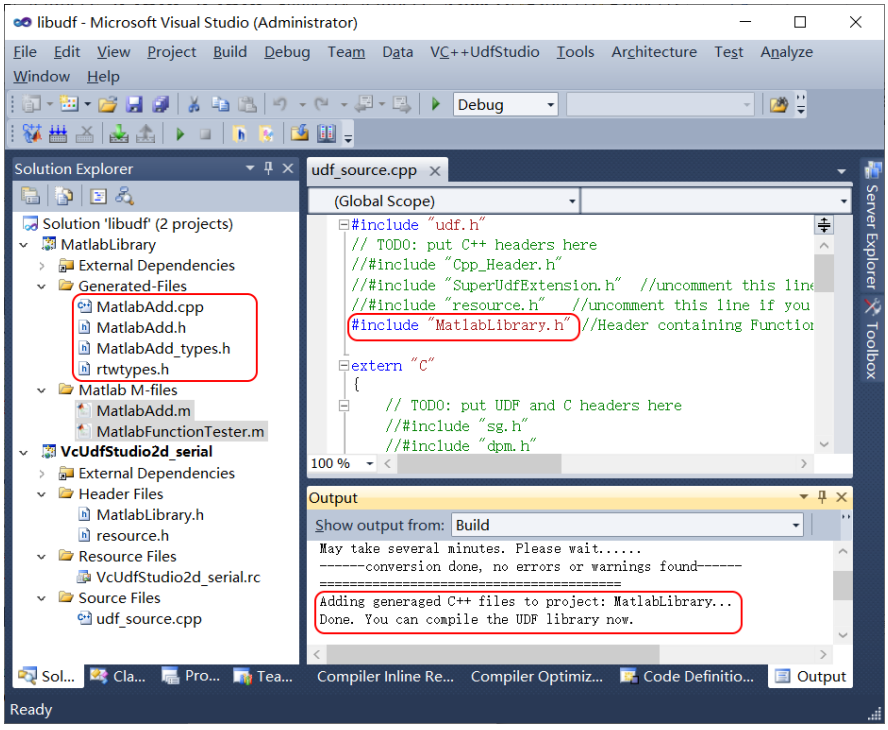


7. Select type for each function arguments in the comboboxes. Default type is scalar float, not matrix. If you want to specify matrix type, please click the “Matrix…” button to set matrix dimensions in the dialog. Check the dynamic box if the size is dynamic. Click OK to convert.

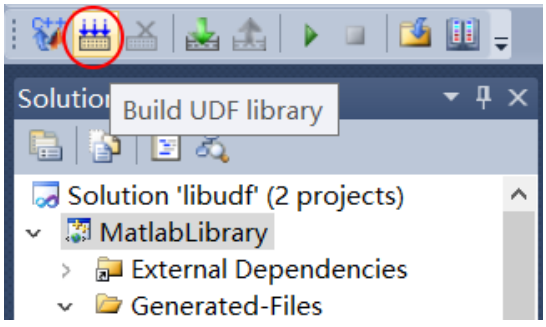




8. If conversion succeeds, the software will auto add the generated C++ files to "Generated-Files" folder. All the Matlab function declarations are in the MatlabLibrary.h file. **Note**: remember to add #include "MatlabLibrary.h" in the UDF source file "udf_source.cpp" before calling any Matlab function.

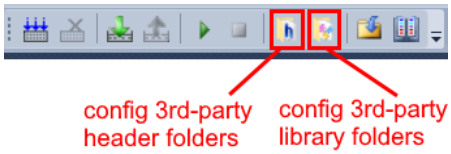


9. Click "Build UDF library" toolbar button to compile the UDF library. Later operations can be found in previous section "Basic steps of UDF Compilation and debug".

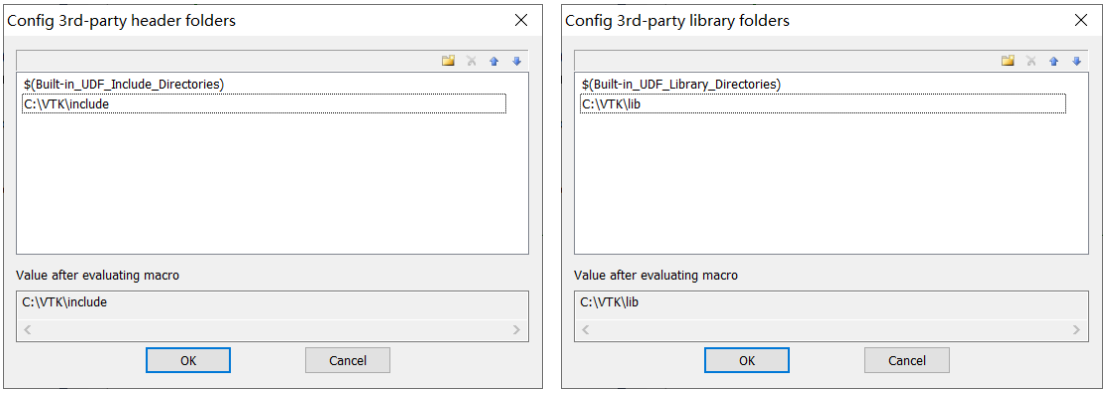


8. Set up 3rd-Party Directories (Enterprise Version Only)

1. You can use below buttons to set 3rd-party header directories and library directories.



2. Below two figures show the dialogs after pushing these buttons. Additional 3rd-party header directories and library directories can be added by manual input or browse. $(Build-in_UDF_Include_Directories) and $(Build-in_UDF_Library_Directories) represent the necessary header directories and library directories required by VC++ UDF Studio. They are forbidden to be modified, only allowed to be moved their relative positions with additional 3rd-party directories.

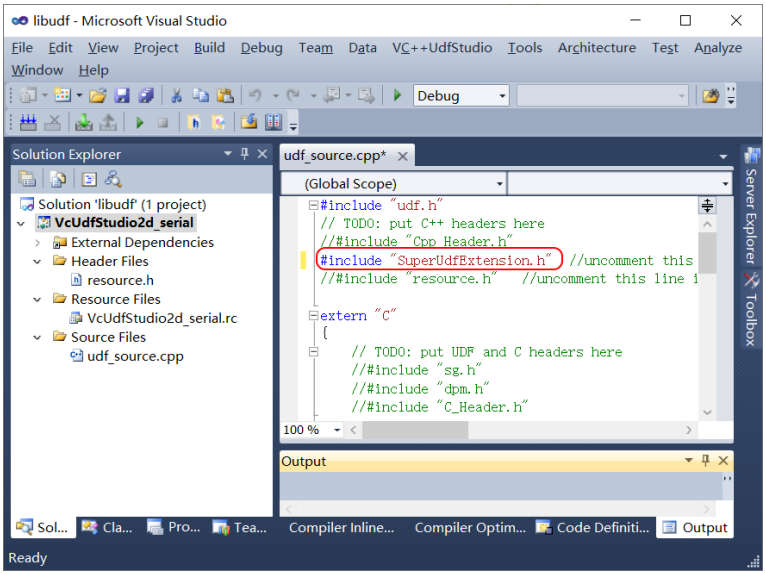


9. SuperUDF Extension Library

8.1. Enable SuperUDF Extension Library

1. This tool has packed some useful functions in the form of 3rd-party library for users' convenient calling. As below figure shows, the user just needs to remove the comments of following line.

```
#include "SuperUdfExtension.h"
```



8.2. Extension Library Function List

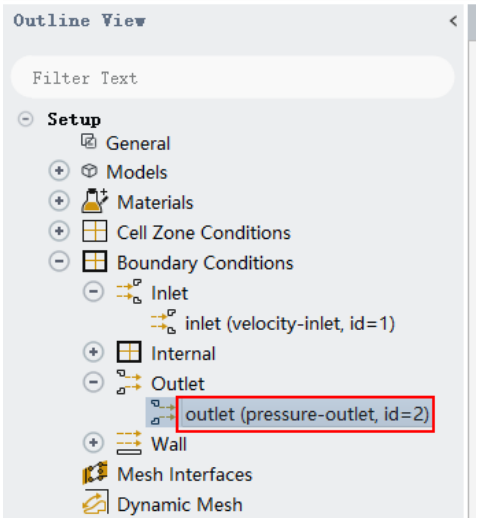
1. `void SuperUdf_Initialize(HMODULE hLibudfDllModule)`

This function is used for the SuperUdf library initialization, where "hLibudfDllModule" is the module handle of udf library. You can use AfxGetInstanceHandle() to get it (see the example in next section).

Note: This function has to be called before other SuperUdf extension functions. The best place to call it is in the "DEFINE_EXECUTE_ON_LOADING" macro.

2. `int SuperUdf_GetZoneIdByName(char* strZoneName)`

Get zone ID according to zone or boundary name. For example the case in below figure, if we call SuperUdf_GetZoneIdByName("outlet"), the function will return 2.



This function is mainly used to improve the robustness of UDF source code. As we know, LookUp_Thread(domain, zone_ID) is the common way to get Thread, where zone_ID is a key parameter. However, it will change with the input mesh. Many users have to inquire the zone ID manually and revise/recompile the UDF source code each time the input mesh changes, which is very inconvenient. After using this function, we can set a fixed name for the zone when we draw the mesh and thus the UDF source code needn't be changed anymore.

Note: This function can only be called on serial or host. It will return -1 on node. A workaround is that we can call it on serial or host and then call host_to_node_int to send the value to node.

3. `void SuperUdf_Interrupt()`

This function is used to interrupt steady or unsteady iteration. You can put this function in DEFINE_ADJUST or DEFINE_EXECUTE_AT_END so that you can call this function to stop the iteration when your criterion reaches.

4. `HWND SuperUdf_GetFluentMainWnd();`

Get the handle of Fluent main window (Enterprise version only, see programming guide).

5. `void SuperUdf_Steady_Iterate(int nTimes)`

Drive Fluent to iterate n steps in steady case (Enterprise version only, see programming guide).

6. void SuperUdf_ExecuteConsoleCommand(char* strAnsiConsoleCommand)

   Drive Fluent to perform TUI or scheme command (Enterprise version only, see programming guide).

7. void SuperUdf_AddUserMenu(UINT uMenuResourceID)

   Insert user menu in Fluent (Enterprise version only, see programming guide).

8. void SuperUdf_EnableMenuItem(UINT uTargetMenuID, BOOL bEnabled)

   Enable or disable user menu item (Enterprise version only, see programming guide).

9. void SuperUdf_SetMenuBmpAndFun(MenuItemBmpAndFun menuBmpAndFuns[], ULONG nCount)

   Set the bitmaps and click action functions (Enterprise version only, see programming guide).

10. void SuperUdf_SetMenuSelectCallBack(MENUSELECTPROC UserCallBackFunction)

    Set the call back function of select menu. Menu items can be dynamically disabled or enabled in the call back function (Enterprise version only, see programming guide).

8.3. Extension Library Function Example

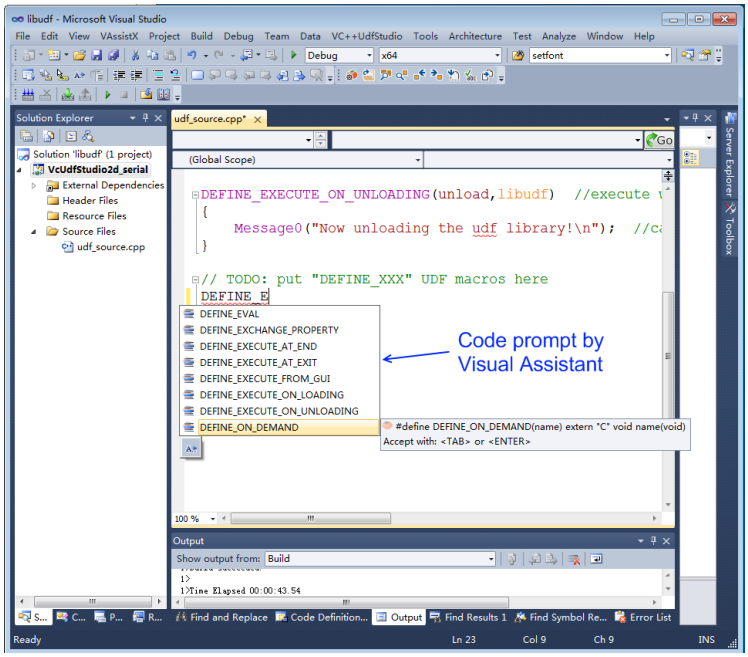
Below is an example of extended functions in academic version (Enterprise version extended function example is shown in programming guide).

```
#include "udf.h"
#include "SuperUdfExtension.h"

DEFINE_ON_DEMAND(GetOutletId)
{
    int outlet_id;
    face_t f;
    Thread*tf;
    Domain*domain=Get_Domain(1);
#if !RP_NODE
    outlet_id=SuperUdf_GetZoneIdByName("outlet"); //get the id of zone whose name is "outlet"
#endif
    host_to_node_int_1(outlet_id);

#if !RP_HOST
    if(-1==outlet_id)
        Message("Can't get the ID on myid=%d\n",myid);
    else
    {
        tf=Lookup_Thread(domain, outlet_id);
        Message("myid=%d, outlet id=%d\n",myid, outlet_id);
        begin_f_loop(f,tf)
        {
            if(PRINCIPAL_FACE_P(f,tf))
            {
                // loop over faces on "outlet"
            }
        }
        end_f_loop(f,tf)
    }
```

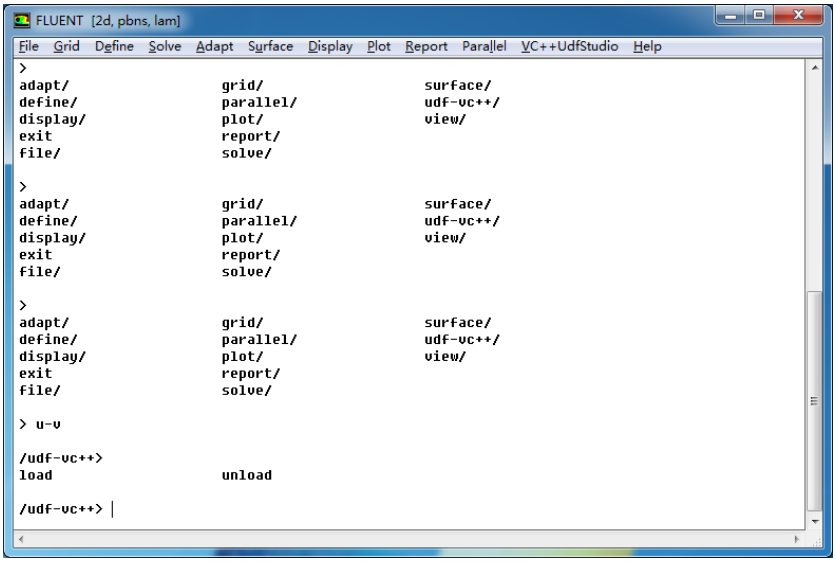
```
#endif
}

DEFINE_EXECUTE_ON_LOADING(load,libudf)
{
    SuperUdf_Initialize(AfxGetInstanceHandle());
}
```

10.Tips

1. “Visual assistant” (www.wholetomato.com) is highly recommended to install, which has a lot of extended functions (such as code completion, braces matching, user-defined keyword colors).

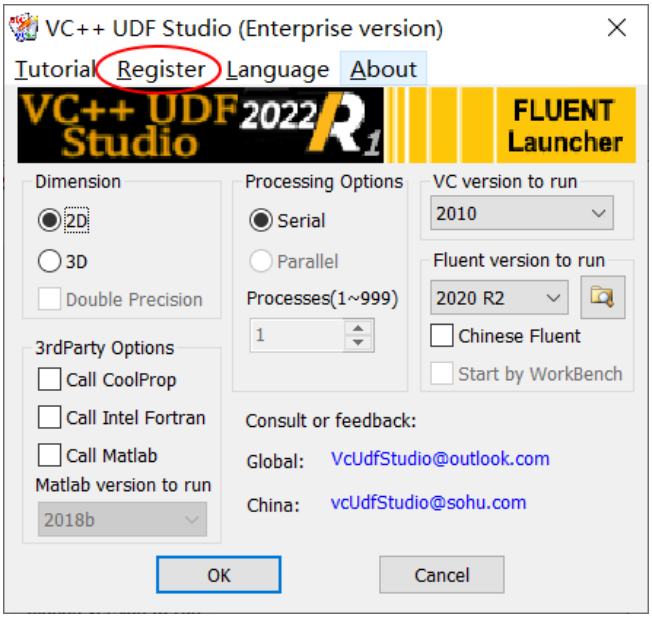


2. VC++UdfStudio menu in Fluent can be loaded or unloaded by TUI command *udf-vc++/load* or *udf-vc++/unload*

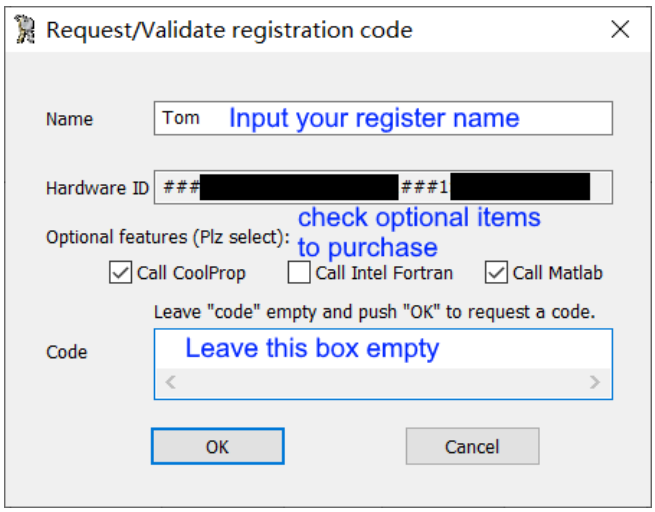


11.How to Register

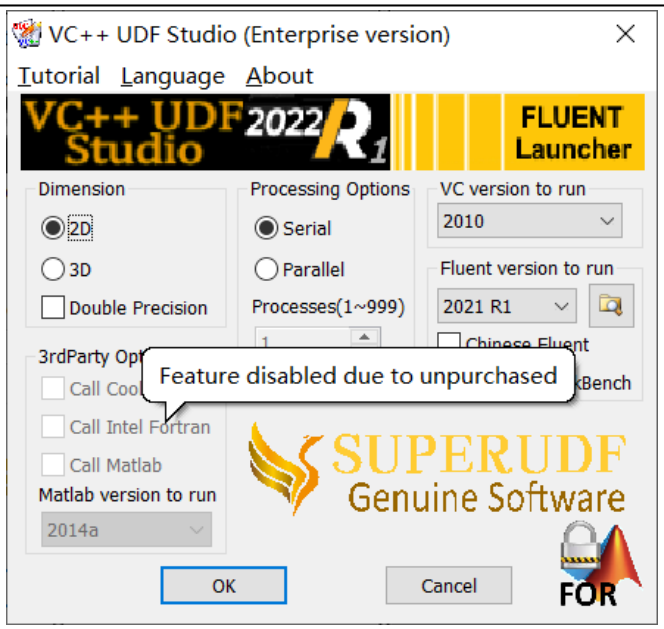
1. Open launcher and click "Register" menu.



2. Input your username and leave the "Code" text box empty. Check the optional features you want to purchase (call Intel Fortran or call Matlab) then click OK. All your user name and hardware information will be put into the text file “user.ini”.



3. Contact vcUdfStudio@outlook.com (International) or vcUdfStudio@sohu.com (China) and pay for the software. Then send the “user.ini” file as the email attachment. After receiving returned email with register code, run the Launcher as administrator and click "Register" menu again. Input the user name and register code and all functions are available now (Note that after successful registration, corresponding feature will be disabled if you haven’t purchased “call Intel Fortran” or “call Matlab” feature. You can uninstall this software and re-install it so as to recover to trial version and try these features).


VC++ UDF Studio (Enterprise version)
Tutorial Language About
VC++ UDF Studio 2022 R1
FLUENT Launcher
Dimension
2D
3D
Double Precision
Processing Options
Serial
Parallel
Processes(1~999)
VC version to run
2010
Fluent version to run
2021 R1
Chinese Fluent
Feature disabled due to unpurchased
3rdParty Opt
Call Intel Fortran
Call Matlab
Matlab version to run
2014a
SUPERUDF
Genuine Software
OK
Cancel
FOR