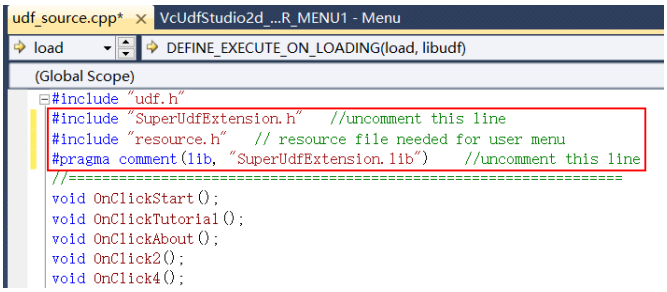
SuperUDF Library Programming Guide 22.1 Open Chinese Version

VC++ Udf Studio 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 two lines.

```
#include "SuperUdfExtension.h"     //uncomment this line
#include "resource.h"     // resource file needed for user menu
#pragma comment(lib, "SuperUdfExtension.lib")     //uncomment this line
```



Extension Library Function List:

1. `void SuperUdf_Initialize(HMODULE hLibudfDllModule)`

   hLibudfDllModule---- module handle of udf library. Call AfxGetInstanceHandle()to get it.

   Return Value----void

***Description*:** 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-1 in later 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)`

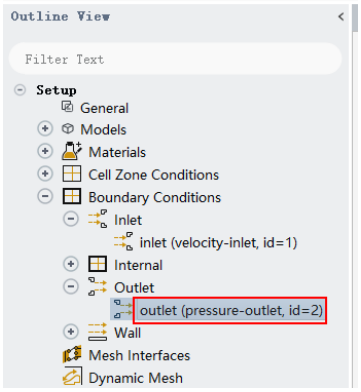
   strZoneName----zone or boundary name

   Return Value----zone ID corresponding to the name

***Description*:** 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 (reference to example-1).

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. `HWND SuperUdf_GetFluentMainWnd();`

Return Value---- handle of the Fluent main window.

Description: Get the handle of Fluent main window. For example, if you call MessageBox (Window API function), which requires the parent window handle as the input parameter, then you can call SuperUdf_GetFluentMainWnd() to get the handle of Fluent main window as the parent window.

Example:

```
DEFINE_ON_DEMAND(msgbox)
{
    HWND hFluentWnd=SuperUdf_GetFluentMainWnd();
    ::MessageBox(hFluentWnd, "Test", "Information",MB_OK);    // MessageBox is a Windows API function
}
```

4. void SuperUdf_Steady_Iterate(int nTimes)

nTimes----`iterations for steady case`

Return Value----void

Description: Drive Fluent to iterate n times in steady case. This can be only used in steady case, and trial version only allows 1 iteration.

Example:

```
SuperUdf_Steady_Iterate(1000);    //drive Fluent to iterate 1000 steps
```

5. void SuperUdf_ExecuteConsoleCommand(char* strAnsiConsoleCommand)

strAnsiConsoleCommand----TUI or Scheme command

Return Value----void

Description: Drive Fluent to perform TUI or scheme command (not available in trial version)

Example:

```
SuperUdf_ExecuteConsoleCommand("/solve/dual-time-iterate 1000 20");
```

The above is a TUI command example, which can drive Fluent to perform unsteady iterations (1000 time

steps with 20 iterations in each time step).

SuperUdf_ExecuteConsoleCommand("(write-case \"d:\\test.cas\")");

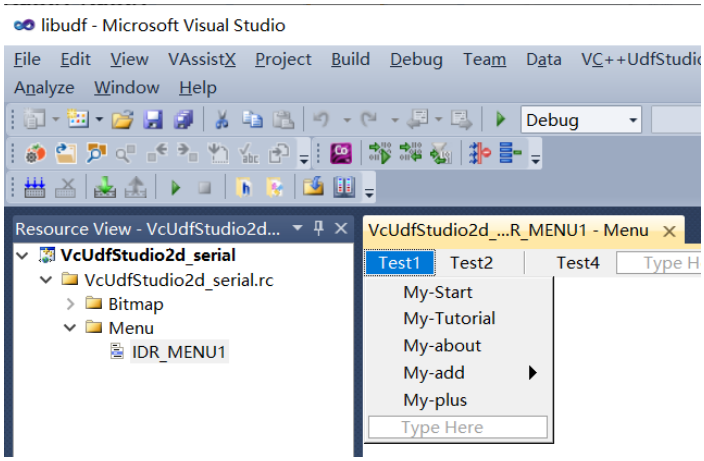
The above is a Scheme command example, which will save current case as d:\test.cas.

6. void SuperUdf_AddUserMenu(UINT uMenuResourceID)

uMenuResourceID----resource ID of the user menu

Return Value----void

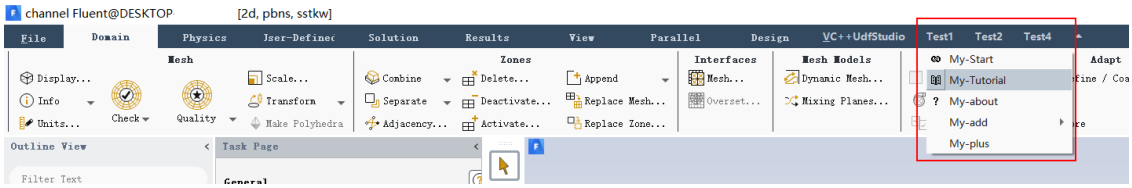
Description: Insert user menu in Fluent. Before calling this function, you need to design a menu resource using menu designer in Visual Studio[1-2]. Set the menu resource ID as the input parameter of this function. Then, the menu will appear in Fluent. The position is just before the "Help" menu. Below figure shows a menu designer case including some sub-menus.



Example:

SuperUdf_AddUserMenu(IDR_MENU1);

Above source will insert the menu into Fluent (resource ID IDR_MENU1 in menu designer), as below figure shows. Menu icon has to be added by other functions introduced later.



7. void SuperUdf_EnableMenuItem(UINT uTargetMenuID, BOOL bEnabled)

uTargetMenuID----resource ID of a menu item.

bEnabled----enable the submenu if set TRUE. Disabled and grayed if set FALSE.

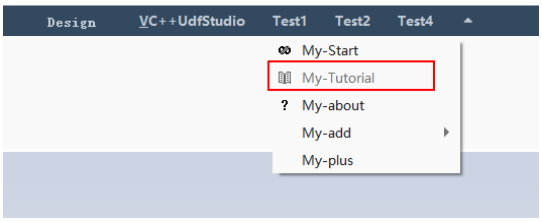
Return Value----void

Description: Enable or disable user menu item.

Example:

SuperUdf_EnableMenuItem(ID_TEST1_TUTORIAL, *FALSE*);

Above source will disable the "My-Tutorial" menu item and make it gray.



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

menuBmpAndFuns----MenuItemBmpAndFun type array, where MenuItemBmpAndFun is strut type. The declaration is as below

```
typedef struct
{
    UINT MenuItemID;    //menu item ID
    HBITMAP hBitmap;    //menu bitmap handle
    void(*fcn)();     // pointer to a menu item response function returning void
}MenuItemBmpAndFun;
```

Where,

MenuItemID----ID of the menu item.

hBitmap----handle of the menu icon. You can use LoadBitmap function to load an icon resource.

fcn----void type function pointer for the responding function.

nCounts----MenuItemBmpAndFun array size.

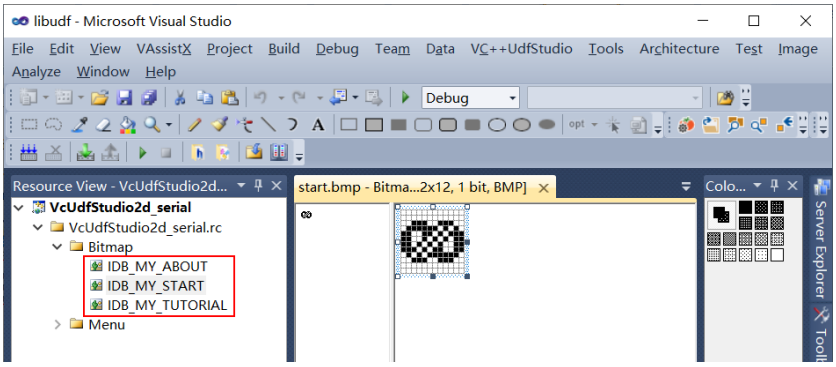
Return Value----void

Description: Set the bitmaps and callback functions when user clicking the menu item. You can set the hBitmap as NULL if there is no bitmap for the menu item. While, responding function is a must to set. Otherwise, there will be no action when you click the menu item. *Note*: please call this before *SuperUdf_AddUserMenu* function.

Example:

```
void OnClickStart();
void OnClickTutorial();
void OnClickAbout();
void OnClick4();
DEFINE_EXECUTE_ON_LOADING(load, libudf)
{
    HMODULE hModule=AfxGetInstanceHandle();
    SuperUdf_Initialize(hModule);   //Call this before any other SuperUdf functions
    MenuItemBmpAndFun bmpAndFun[]={
    {ID_TEST1_START,::LoadBitmap(hModule,MAKEINTRESOURCE(IDB_MY_START)), OnClickStart},
    {ID_TEST1_TUTORIAL,::LoadBitmap(hModule,MAKEINTRESOURCE(IDB_MY_TUTORIAL)), OnClickTutorial},
    {ID_TEST1_ABOUT,::LoadBitmap(hModule,MAKEINTRESOURCE(IDB_MY_ABOUT)), OnClickAbout},
    {ID_TEST4, NULL, OnClick4}    // if no bitmap, use NULL
        SuperUdf_SetMenuBmpAndFun(bmpAndFun, sizeof(bmpAndFun)/sizeof(MenuItemBmpAndFun));
      SuperUdf_AddUserMenu(IDR_MENU1);
};
```

Above source can add bitmaps whose resource ID are IDB_MY_START, IDB_MY_TUTORIAL and IDB_MY_ABOUT. Please add these resources by Visual Studio resource view. OnClickStart，OnClickTutorial and OnClickAbout are the responding functions.



9. void SuperUdf_SetMenuSelectCallBack(MENUSELECTPROC UserCallBackFunction)

UserCallBackFunction----Callback function when mouse selects or hovers over the menu item. MENUSELECTPROC is a callback type. The declaration is

typedef void (CALLBACK MENUSELECTPROC)()*

Actually this is a function pointer returning void.

Return Value----void

Description: Set the call back function when mouse selects or hovers over the menu item. You can enable or disable menu item dynamically in this function. Other purpose is not recommended.

Note: Never call Message, Message0 or CX_Message is serial Fluent (2021R1 or higher version), otherwise serial verison of Fluent may be corrupted. Serial (2020R2 or lower) or Paralllel version of Fluent is not limited by this requirement.

Example:

```
void CALLBACK MenuSelectProc();
DEFINE_EXECUTE_ON_LOADING(load, libudf)
{
    HMODULE hModule=AfxGetInstanceHandle();
    SuperUdf_Initialize(hModule);
    ……
    SuperUdf_AddUserMenu(IDR_MENU1);
    SuperUdf_SetMenuSelectCallBack(MenuSelectProc);
}
void CALLBACK MenuSelectProc()
{
    if(my_value==TRUE)    // my_value is a variable to judge
            SuperUdf_EnableMenuItem(ID_TEST4, TRUE);
    else
```

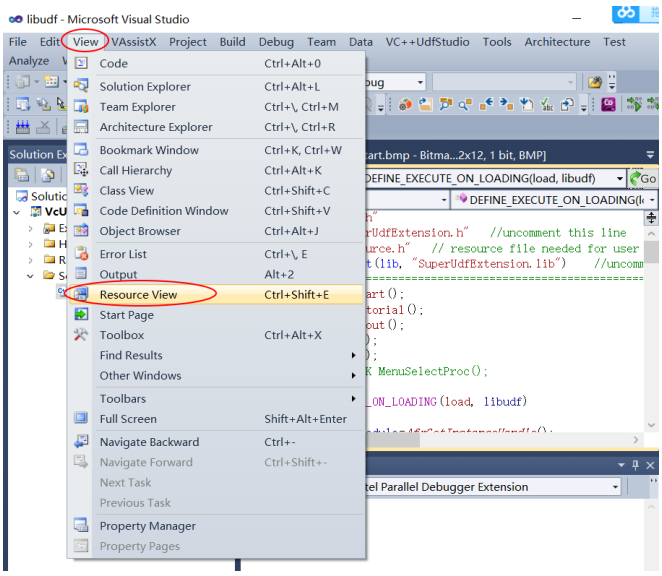
```
        SuperUdf_EnableMenuItem(ID_TEST4, FALSE);
    }
```

Example-1: Get zone ID from zone/boundary name

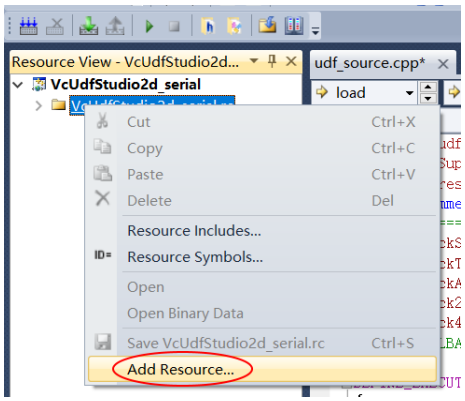
```
#include "udf.h"
#include "SuperUdfExtension.h"
#pragma comment(lib, "SuperUdfExtension.lib")

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());
}
```

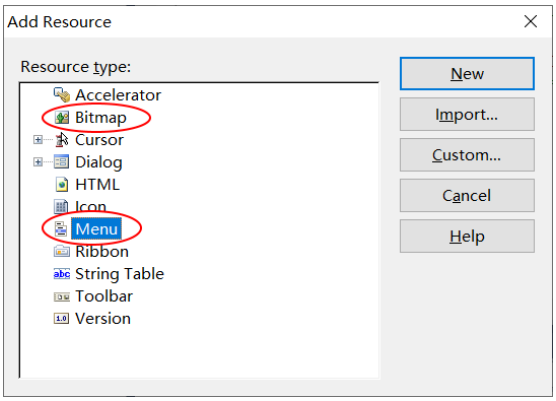
Example-2: Insert User Menu in Fluent

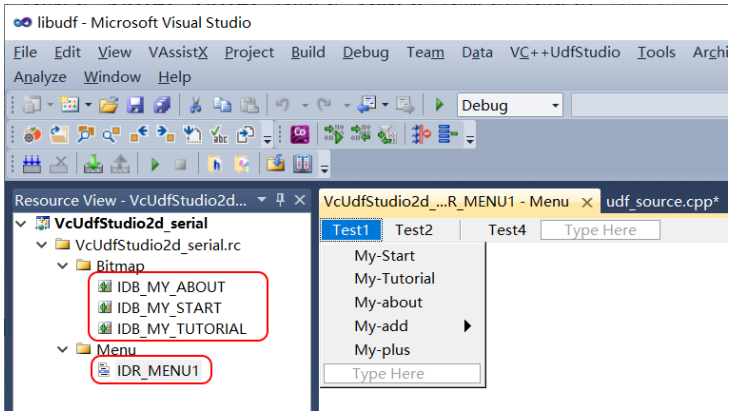
First, open Visual Studio “Resource View”. If it is hidden, you can click “View->Resource View” menu to show it.



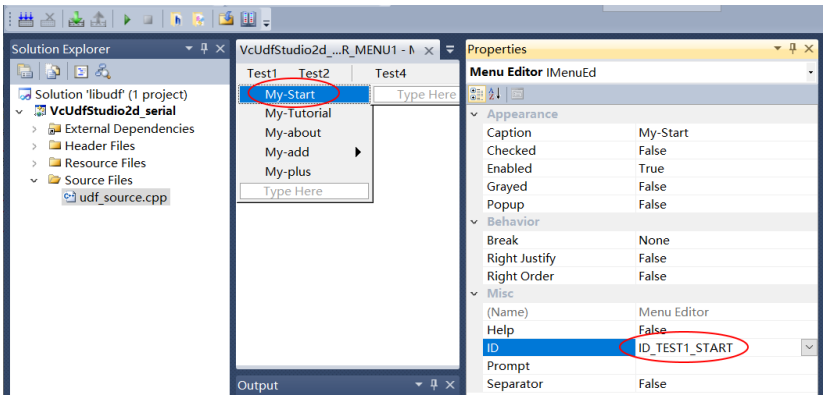
Right-click on the project folder in “Resource View” and click “Add Resource…” menu item. Add three bitmaps resource, with ID “IDB_MY_START”, “IDB_MY_TUTORIAL” and “IDB_MY_ABOUT”. You can either “New” or “Import” existing bitmaps. We recommend black-and-white bitmaps with 12*12 pixel size.







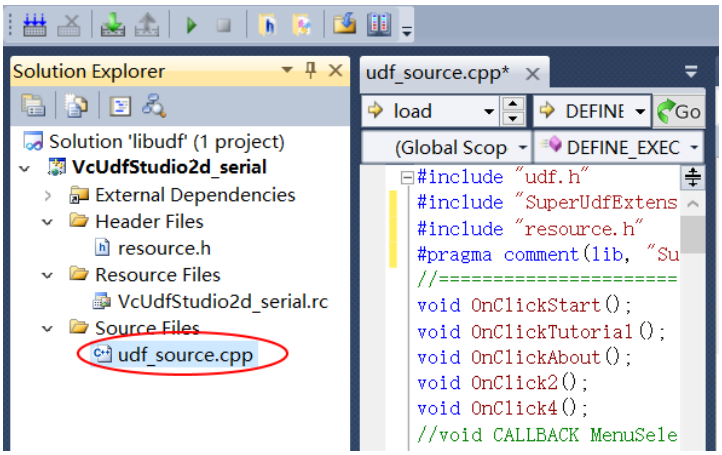
Add a menu resource named “IDR_MENU1” and the menu item property can be modified in the right “Property Manger”. If the “Property Manger” is hidden, then double click a menu item to show it.



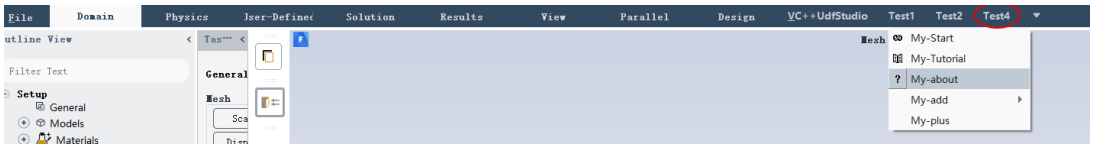
Set the menu item text and ID as below table.

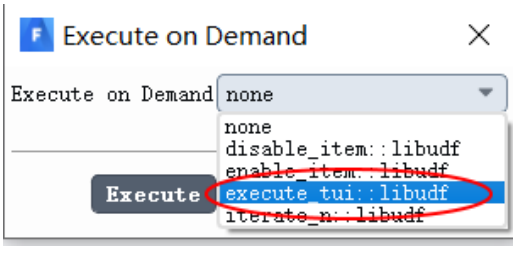
Menu Item Text	*Menu item ID*
My-Start	*ID_TEST1_START*
My-Tutorial	*ID_TEST1_TUTORIAL*
My-about	*ID_TEST1_ABOUT*
Test4	*ID_TEST4*
Others	*Name them as you wish because we won't set their callback functions here*

Go back to Solution Explorer and double-click "udf_source.cpp" to open the UDF source.



Put below content in the source "udf_source.cpp" file. There will be a user menu appear in Fluent after compilation and loading. However, you will find that "Test4" menu is disabled but can be enabled after you execute "execute_tui::libudf" manually. This is because we have set "my_value" to "FALSE" at the beginning and menu is enabled/disabled in the "MenuSelectProc" function according to "my_value". In "execute_tui::libudf" DEFINE_ON_DEMAND macro, "my_value" is set to "TRUE", therefore the menu item is enabled after executing this macro when we select or mouse-hover over it.





Udf source “udf_source.cpp” as below:

```
#include "udf.h"
#include "SuperUdfExtension.h"
#include "resource.h"     // resource file needed
#pragma comment(lib, "SuperUdfExtension.lib")
//=================Only Registered version can pass, because trial version only allow 2 macros==================
BOOL my_value=FALSE;   // variable enable or disable Menu items
void OnClickStart();
void OnClickTutorial();
void OnClickAbout();
void OnClick4();
void CALLBACK MenuSelectProc();

DEFINE_EXECUTE_ON_LOADING(load, libudf)
{
	HMODULE hModule=AfxGetInstanceHandle();
	SuperUdf_Initialize(hModule);

	MenuItemBmpAndFun bmpAndFun[]={
		{ID_TEST1_START,::LoadBitmap(hModule,MAKEINTRESOURCE(IDB_MY_START)), OnClickStart},
		{ID_TEST1_TUTORIAL,::LoadBitmap(hModule,MAKEINTRESOURCE(IDB_MY_TUTORIAL)), OnClickTutorial},
		{ID_TEST1_ABOUT,::LoadBitmap(hModule,MAKEINTRESOURCE(IDB_MY_ABOUT)), OnClickAbout},
		{ID_TEST4, NULL, OnClick4}   // if no bitmap, use NULL
	};
	SuperUdf_SetMenuBmpAndFun(bmpAndFun, sizeof(bmpAndFun)/sizeof(MenuItemBmpAndFun));
	SuperUdf_AddUserMenu(IDR_MENU1);
	SuperUdf_SetMenuSelectCallBack(MenuSelectProc);
}

void OnClickStart()
{
	HWND hFluentGUIMainWnd=SuperUdf_GetFluentMainWnd();
	MessageBox(hFluentGUIMainWnd, "start clicked", "",    MB_OK | MB_ICONINFORMATION | MB_TOPMOST);
}

void OnClickTutorial()
{
	HWND hFluentGUIMainWnd=SuperUdf_GetFluentMainWnd();
	MessageBox(hFluentGUIMainWnd, "tutorial clicked", "", MB_OK | MB_ICONINFORMATION | MB_TOPMOST);
}

void OnClickAbout()
{
	HWND hFluentGUIMainWnd=SuperUdf_GetFluentMainWnd();
	MessageBox(hFluentGUIMainWnd, "about clicked", "", MB_OK | MB_ICONINFORMATION | MB_TOPMOST);
}

void OnClick4()
{
	HWND hFluentGUIMainWnd=SuperUdf_GetFluentMainWnd();
	MessageBox(hFluentGUIMainWnd, "4 clicked", "", MB_OK | MB_ICONINFORMATION | MB_TOPMOST);
}

DEFINE_ON_DEMAND(disable_item)
{
	SuperUdf_EnableMenuItem(ID_TEST4, FALSE);
```

```
    SuperUdf_EnableMenuItem(ID_TEST1_TUTORIAL, FALSE);
}

DEFINE_ON_DEMAND(enable_item)
{
    SuperUdf_EnableMenuItem(ID_TEST4, TRUE);
    SuperUdf_EnableMenuItem(ID_TEST1_TUTORIAL, TRUE);
}

DEFINE_ON_DEMAND(execute_tui)
{
    SuperUdf_ExecuteConsoleCommand("/solve/initialize/initialize-flow yes");
    my_value=TRUE;
}

DEFINE_ON_DEMAND(iterate_n)
{
    SuperUdf_Steady_Iterate(1000);
}

void CALLBACK MenuSelectProc()
{
    if(my_value==TRUE)   // my_value is a variable to judge
        SuperUdf_EnableMenuItem(ID_TEST4, TRUE);
    else
        SuperUdf_EnableMenuItem(ID_TEST4, FALSE);
}
```

Reference:

[1] SunXin, Detailed Introduction for VC++ (in Chinese), Publishing House of Electronics Industry

[2] Charles Petzold, Programming Windows (5th edition), Microsoft Press