CELEBRITY_C++, C++ Interface for *Expert*

CELEBRITY_C++, the C++ interface for Silvaco's ***Expert*** layout tool, is now an exclusive customization language that expands the tool's handling of high-level customization, such as a design application working with ***Expert's*** database. Users of the interface develop dynamic link libraries (DLLs) with Microsoft Visual C++ that are stored in the Silvaco install directory. ***Expert*** loads these DLL files and implements the user-defined commands in the menu bar.

Expert is already customizable with Silvaco's LISA/xi scripting language. However, ***CELEBRITY_C++*** extends these features. The following are some benefits of ***CELEBRITY_C++***:

- Enables functional access to the ***Expert*** layout database and commands from Visual C++.
- Permits high-level customization through the use of functions and classes that are prepared in Visual C++
- Easy implementation of graphic-user-interfaces (GUIs) such as dialog boxes, form windows, and toolbars.
- Compiles the source code to a fast-running dynamic link library (DLL) file.
- DLLs are easy to distribute and install.

Using *CELEBRITY_C++*

1) Create a new Visual C++ project

Take the following steps to create a new Visual C++ project:

1. Click File > New from Visual C++ menu bar to launch the "Create a new file/ project" dialog.
2. Click the "MFC AppWizard (dll)" icon and specify the project name.

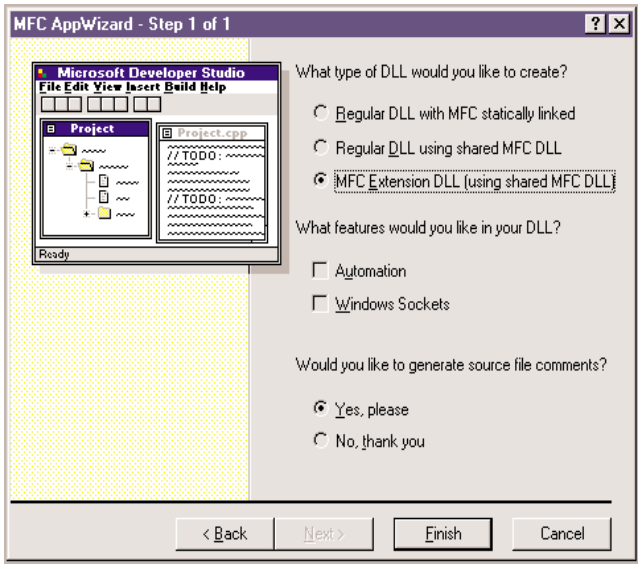

Figure 2. MFC AppWizard.

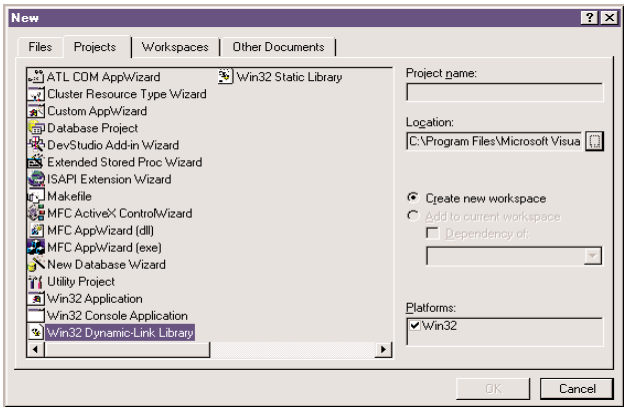

Figure 1. Create a new file/project dialog.

3. Choose "MFC expanded DLL (using MFC common DLLs)" radio button
4. Click "Finish" to complete the operation and open a new Visual C++ project.

2) Project Setting

Certain library and header files are required to access Expert database and commands. Take the following steps to specify these files:

1. Click Build > Setting Active from the menu bar and select "<project name> - Win32 Release".
2. Expand the compressed file <SILVACO_INST_DIR>\examples\expert\plugset.zip to the ExpApi and Include folders.
3. Click Tools > Options from the menu bar and specify the path to the header files in the "Directories" page.
4. Click Project > Setting from the menu bar to set the following:

```
Output directory:
<SILVACO_INST_DIR>\lib\expert\<version>\x86
-nt\PlugIns

Executable for debug session:
<SILVACO_INST_DIR>\etc\GuiAppStarter.exe

Parameters for Executable: -lib-dir-name
Expert -exe-name Expert

Object / Library module:
<SILVACO_INST_DIR>\lib\expert\<version>\x86-nt\
```

5. Add the following lines in the Visual C++ project file StdAfx.h:

```
#include <afxtempl.h>

#include <afxwin.h>
```

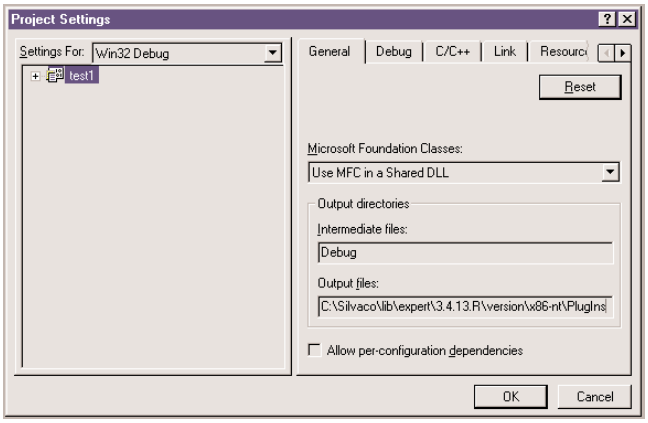

Figure 3. Project Setting.

3) Writing source code

It is necessary to write commands that add the menu items to the DLL file. First, a new C++ source code file is necessary in order to write the program.

You can add menu items by generating CCustomMenu constructor, as illustrated in the following example:

```
void File1Out();
void File1OutAll();
void File2Out();
void FileIn();


CUSTOMENU_ITEM Items1[] =
{
  CUSTOMENU_ITEM("Current Cell(&L)",
&File1Out),
  CUSTOMENU_ITEM("All Child Cells(&L)",
&File2OutAll)
};


CUSTOMENU_ITEM Items2[] =
{
  CUSTOMENU_ITEM("Current Cell(&L)",
&File2Out)
};
CUSTOMENU_ITEM Items[] =
{
  CUSTOMENU_ITEM("Output File1(&L)",
Items1, EA_countof(Items1)),
  CUSTOMENU_ITEM("Output File2(&D)",
Items2, EA_countof(Items2)),
  CUSTOMENU_ITEM(),
  CUSTOMENU_ITEM("Load Files(&L)", &FileIn)
};
CCustomMenu cm(10, "Output Files", Items,
EA_countof(Items));
```

Menu items are defined as arrays of CUSTOMENU_ITEM. Add extended menus by specifying the name of another CUSTOMENU_ITEM and the number of items instead of the name of function. The function EA_countof()counts the number of items in an CUSTOMENU_ITEM array. Commands used in CUSTOMENU_ITEM should be defined in advance.

Finally, generate a constructor of CCustomMenu class by using the CUSTOMENU_ITEM array. The parameters of CCustomMenu::CCustomeMenu() are the position in the menu bar, the name of menu, the name of CUSTOMENU_ITEM array, and the number of items in the CUSTOMENU_ITEM array.

4) Compiling the program

When the source code is complete, run Build > Build to compile the program into a DLL. When compiling is complete, start ***Expert*** to check if the program is working as intended. Place ready-to-deliver DLL files in the <SILVACO>\ lib\expert\<version>\x86-nt\PlugIns directory.

Conclusion: Comparing CELEBRITY_C++ and LISA / xi Scripts

LISA / xi scripts are also used to customize ***Expert***. xi is an extension of the ***LISA*** (Language for Interfacing Silvaco Applications) scripting language. ***CELEBRITY_C++*** performs most of the same features as a ***LISA*** / xi script. The option is dependent on the project's objective and scale.

LISA / xi scripts are easy to develop but are proprietary to ***Expert***. This makes them ideally suited for simple or small utility development. Despite its simplicity, ***LISA*** / xi is very flexible.

CELEBRITY_C++ is better suited for the development of larger, more complex applications. ***CELEBRITY_C++*** makes more efficient use of the processor, and is not modifiable in its compiled form.

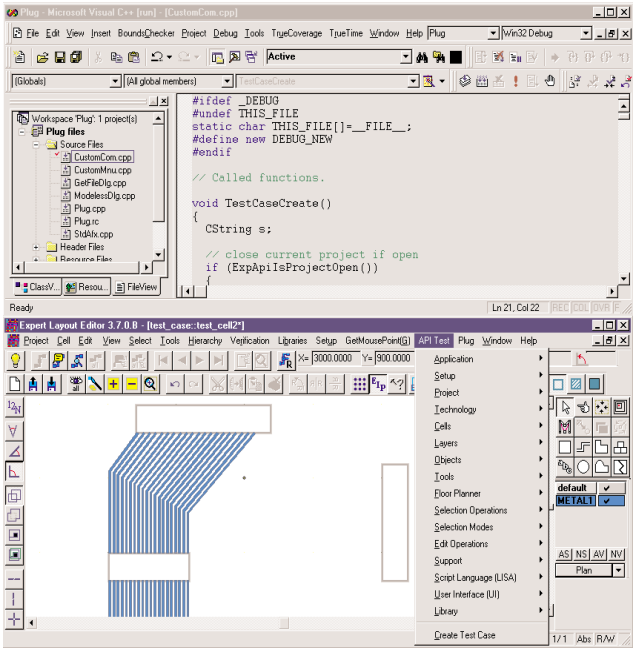
Figure 4. Using users DLL in ***Expert***.