C/C++ Development

Last Modified on 01/18/2017 5:47 pm CST

For C/C++ application development, you'll need two sets of files:

- Header files
- Import libraries

The NumXL SDK files for C/C++ development and examples can be found on the **download page**. For the latest development files and examples, you can pull the SDK **project on GitHub**.

1. Compiler Dependency

The header files of NumXL SDK fully adhere to the ANSI standard C syntax. So in principle, any development tool with a compatible ANSI compiler can be used with NumXL SDK.

For the Import Libraries, the C/C++ SDK package already includes libraries files for Visual Studio 2010 and 2013 for 32 and 64-bit platforms.

Although, the included import libraries are readily usable in a wide range of development tools, we have included the module definition source files for the rare occasion where you may need to build your own libraries.

2. (*Optional*) Build From Sources

The C/C++ SDK package (or the project on GitHub) includes the NRE DLL module definition (*.def) source files (under src folder). So, to create import libraries, issue the following commands for each source file:

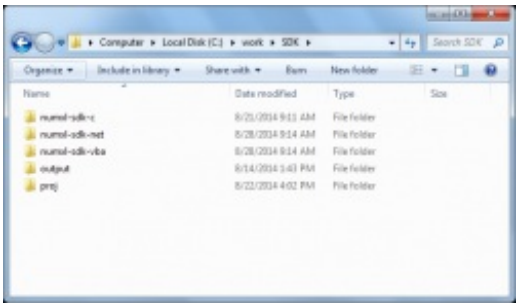
```
LIB /MACHINE:x86 /DEF:<.DEF file name> /OUT:<.lib file name and path>
LIB /MACHINE:x64 /DEF:<.DEF file name> /OUT:<.lib file name and path>
```

Note: The **LIB.exe** is a front end utility to the Linker program and it is usually part of your compiler installation.

Going forward, we are using Visual Studio 2013. So if you use a different development tool, you can use the information here, but map the steps to match the UI of your development tool.

3. Directory Structure

To setup your development folders, we recommend the following structure:



- **numxl-sdk-c** folder contains all files in the SDK.
- **output** folder contains the executable binaries in NumXL and the SDK.
- **proj** folder is where you maintain the source code for your custom application.

4. Hello World!" Application

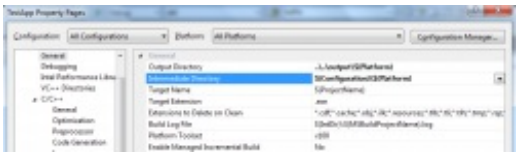
Go to the project folder (e.g.), create a new project:

1. Project type: Visual C++ Projects → Win32 Console Application
2. Open the new project, and click on "Project Properties" under .

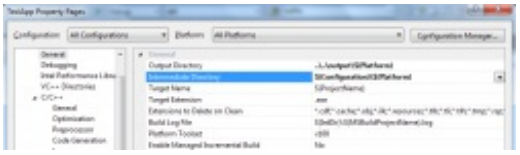
Open the project properties dialog, set the Configuration to "All Configurations", and platform to "All Platforms".

Project Configurations

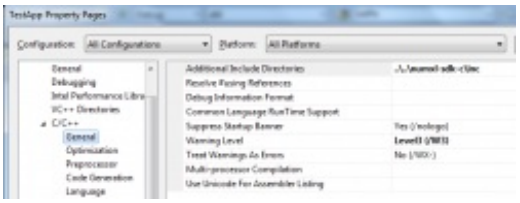
1. Select "General" from the left taskbar, and set the followings:
   - Output Directory: path to the SDK binaries executables (e.g. ....\output\$(Platform))
   - Intermediate Directory: local path relative to your project (e.g. $(Configuration)\$(Platform))



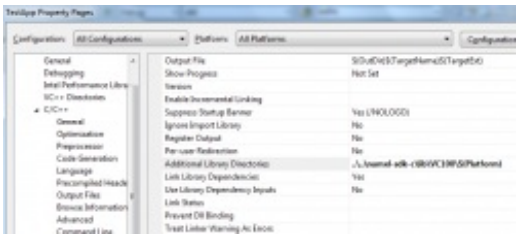
2. Next, under "C/C++" settings, select "General":
   - Additional Include Directories: path to NumXL SDK header files folder



3. Under the linker settings, select "General":
   - Additional Library Directories: path to NumXL SDK import libraries folder (e.g. numxl-sdk-c\lib\VC100\$(Platform))



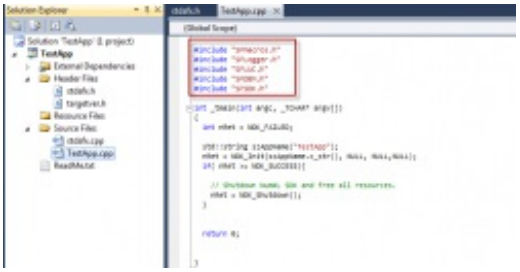
4. Under the linker settings, select "Input":
   - Additional Dependencies: SFSDK.lib, SFLOG.LIB, SFLUC.LIB, SFDBM.LIB



5. Click "OK" and close the project properties dialog box.

Source code

1. In the main source file (e.g. TestApp.cpp>), add the following header files: SFMacros.h, SFLogger.h, SFLUC.h, SFDBM.h, and SFSDK.h



2. Then, initialize the SDK by call NDK_Init.

```
// The AppName must match the configuration file basename.
// In this example, we must have HellpApp.conf in the output directory
std::wstring szAppName(L"HelloApp");

int nRet = NDK_Init( szAppName.c_str(),  // we have HelloApp.conf
                     NULL,                // use the license key found in Li
cense.lic file
                     NULL,                // use the activation code in Lice
nse.lic file
                     NULL                 // use the temp directory found in
 the current user's profile
                   );
if( nRet == NDK_SUCCESS){



}
else
{
  // something went wrong
}
```

3. Finally, shutdown the SDK by calling NDK_Shutdown to free resources.

```
int nRet = NDK_Init( szAppName.c_str(), NULL, NULL, NULL, NULL);
if( nRet == NDK_SUCCESS){

....
  nRet = NDK_Shutdown();// Close log file and free all resources.
}
```

4. Done! You may use next any NumXL SDK function as needed.

Remarks

1. NumXL SDK APIs (arguments and behavior) are designed to match their counterpart NumXL worksheet functions as much as possible: (1) APIs are stateless, (2) robust, and (3) they can generate a wealth of logging information to help with issues raised during development and integration.
2. In essence, the NumXL SDK exports its functions using C-API interface and reports its status via error codes as a return value.
3. The caller application does not get passed any exception generated during the course of the function call.

See Also

[template("related")]