ICS ELECTRONICS
division of Systems West Inc.

IEEE 488 APPLICATION BULLETIN

Converting 32-bit GPIB Test Programs to 64-bit Programs

Introduction

Now that ICS has released a true 64-bit GPIB-32.DLL, many engineers want to write 64-bit applications to take advantage of the larger address space available with 64-bit operating systems. Another reason to convert existing programs is the eventual end of 32-bit platforms at some point in the future. However the process of converting 32-bit GPIB applications to 64-bit applications is new and confusing to many engineers. This Application Bulletin attempts to clear up the mystery surrounding the conversion process by giving the user some conversion guidelines and by including some conversion examples for Visual Basic and Visual C++. The directions in this Application Bulletin will apply to other 32-bit programs that use external libraries.

Obtaining the Example Files

The files used for this bulletin can be found on Example Programs page on ICS's website at http://www.icselect.com/ab_prgms.html. Download the AB48-44_Programs file to your computer and unzip the file to a temporary folder. Within the zip file are all of the files needed to build a 32 or 64-bit GPIB application. In addition to these files there are 32-bit and 64-bit example projects for Visual Basic and Visual C++. You should have the files shown in Figure 1.

Examining the Downloaded Files

The unzipped file is broken down into three folders; 32bit, 64bit and Example Programs.

The 32bit Folder contains all of the files needed to build a 32-bit Visual Basic or Visual C++ application. The GPIB-32.DLL is required to run ICS's 488-USB2 and 488-LPCI GPIB controllers. Visual C++ uses two files: the GPIB-32.lib library file to link the application to the GPIB-32.DLL and the ICSdecl.h file which contains the C function calls to the GPIB-32.DLL. Visual Basic requires just the GPIB-32.vb file to make the function calls to the GPIB-32.DLL.

```
32bit Folder
        GPIB-32.DLL
        GPIB-32.lib
        GPIB-32.vb
        ICS_Spy.exe
        ICSdecl.h
64bit Folder
        GPIB-32.DLL
        GPIB-32.lib
        GPIB-32.vb
        ICS_Spy.exe
        ICSdecl.h
Example Software Folder
        32Bit Examples Folder
                VB 2005 - Visual Basic Project
                VC 2005 - Visual C++ Project
        64Bit Examples Folder
                VB 2005 - Visual Basic Project
                VC 2005 - Visual C++ Project
```

Figure 1 Unzipped Download Files

The last file, ICS_Spy.exe, is a utility program that is used to assist in troubleshooting hardware and software problems. ICS_SPY is most useful in monitoring the calls made to the GPIB-32.DLL. The ICS_SPY has a help menu for the which explains how to use the program.

The 64bit Folder is a similar to the 32bit folder but contains all of the 64-bit equivalent files. The GPIB-32.DLL in this folder is a 64-bit DLL and cannot be used interchangeably with the

32-bit DLL in the 32bit folder. Both DLLS are named GPIB-32.DLL to maintain the standard naming conventions. **Care should be taken to not mix-up the two files**. GPIB-32.vb and ICSdecl.h are identical files to their 32-bit count parts and are interchangeable. ICS_Spy.exe is a 64-bit application which works identical to the 32-bit version. Both programs can monitor calls to either GPIB-32.DLL.

The Example Software Folder is broken down into two sub-folders. One for the 32-bit examples and another for the 64-bit examples. Each sub-folder contains a Visual Basic 2005 project and Visual C++ 2005 project. The 32-bit projects will be used later to show how to make a 32-bit application into a 64-bit application.

Steps to Convert a 32-bit Visual C++ Project

For this you will need the example program located at "..\Example Software\32-bit Examples\ VC 2005" and a copy of the 64bit GPIB-32.lib located within the "..\64Bit" Folder.

Start by opening the project in Visual Studio and attempting to compile. If the project compiles correctly then move on to the first step. The following steps should work for any Visual Studio software that is 2005 or later. Microsoft may make changes to the interface system making these instructions inaccurate for future releases of Visual Studio. If your project does not compile look over the errors you are receiving for hints as to why your system may not be handling this project correctly.

Step 1. Locate the 64bit GPIB-32.lib file located within "..\64bit Folder" and copy it into the example program's working directory ("..\Example Software\32-bit Examples\VC 2005\vc32_GPIBKybd\vc32_GPIBKybd") This should replace the existing 32-bit version of GPIB-32.lib that was already within the folder.

Step 2. Open the 'Configuration Manager' by click the 'Build' menu then go to the bottom of the list where you will see "configuration manager...".

Step 3. Within the drop down menu in the upper right hand corner, under "Active Solution platform:" select '<New...>', in the popup window make sure the new platform is set to 'x64' then click "Okay".

Step 4. Click "Close", recompile and run your application. If you have errors then continue reading the following sections to ensure no mistakes were made.

Steps to Convert a 32-bit Visual Basic Project

For this you will need the example program located at "..\Example Software\32-bit Examples\ VB 2005".

Start by opening the project in Visual Studio and attempting to compile. If the project compiles correctly then move on to the first step. The following steps should work for any Visual Studio software that is 2005 or later. Microsoft may make changes to the interface system making these instructions inaccurate for future releases of Visual Studio. If your project does not compile look over the errors you are receiving for hints as to why your system may not be handling this project correctly.

Step 1. Open the 'Configuration Manager' by click the 'Build' menu then go to the bottom of the list where you will see "configuration manager...".

Step 2. Within the drop down menu in the upper right hand corner, under "Active Solution platform:" select '<New...>', in the popup window make sure the new platform is set to 'x64' then click "Okay".

Step 3. Click "Close", recompile and run your application. If you have errors then continue reading the following sections to ensure no mistakes were made.

Step 4. Confirm that your application is targeting the correct CPU type. Click 'Project' then 'properties' from the drop down menu. In the new window on the left click 'Compile' then the 'Advanced Compile Options...'. The last option will be 'Target CPU' and should be set to 'x64'.

Some Important Facts about 64-bit applications

1. Some older applications use functions that Windows no longer supports or has changed the function for 64-bit applications. These types of errors are common and must be fixed to build your program.

2. A 32-bit application requires the use of a 32-bit DLL and lib/obj file, and a 64-bit application requires the use of a 64-bit dll and lib/obj file. When changing a program over to 64-bit make sure the project settings are pointing to the correct set of files.

3. 64-bit applications can only be built and ran on a 64-bit operating system, but 32-bit applications can be built and ran on either a 32-bit or on a 64-bit operating system.

4. The GPIB-32.DLL may be installed in possibly two locations on a 64-bit operating system. The 64-bit version is installed in the System32 folder, while the 32-bit version is installed in the sysWOW64 folder. If you are having difficulties linking to the correct DLL then copy the DLL you wish to use into your project's working directory. Based on Microsoft's default search path this will become the default GPIB-32.DLL.

Frequently Ask Questions:

I'm receiving LNK Error's when compiling my 64-bit application, however my code compiles fine for my 32-bit application?

The most likely cause to this type of LNK Error is an incorrect .lib file. The GPIB-32.Lib file is specific to 32-bit or 64-bit. The 64-bit version of the library cannot be used with a 32-bit application and the reverse is true. Copying the correct .lib file into your project should fix this issue.

How can I tell if I have the correct .lib in my project?

There is not a good way to tell if the .lib file you are using is built for a 32-bit or 64-bit application. The easiest way to handle this problem is to take a copy of the .lib from the ICS Dev files folder and copy over the .lib in question. Using updated .lib files with older GPIB-32.DLL's will not cause any problems.

I need to have one project build both a 32-bit and 64-bit versions of my project do I need to switch the 32-bit and 64bit .lib files before each compile?

Yes, since your program is linking to GPIB-32.Lib this file must match the target OS of your project. There are two ways to get around manually switching these files back and forth. The first is a change to the code contained within the ICS_Decl.h file. At the top you will see the line 'pragma comment(lib, "GPIB-32.LIB")' replace this line with the follow

```
#if _WIN64
        #pragma comment(lib, "GPIB-32(x64).LIB")
#else
        #pragma comment(lib, "GPIB-32(x86)LIB")
#endif
```

Change the name of the 32-bit GPIB-32.lib within your working directory to "GPIB-32(x86).LIB", and your 64bit GPIB-32.lib to "GPIB-32(x64).LIB". This Code will automatically select the appropriate library file based on the target operating system.

The second way to automate this change is by making a change within the project definitions. Since project definitions are directly connected to the solution platform, changes may be made for the 64-bit application without affecting the 32-bit version. Before changes to the project definitions are made two new folders should be made to contain the 32-bit and 64-bit library file. Create a "32bit" folder within the working directory along with a "64bit" folder. Place the GPIB-32.lib files within their respective folders. Place the 32bit .lib within the "32bit" folder and the 64bit .lib within the "64bit" folder. Now open up your project definitions and navigate to 'linker', 'input' ,'additional dependencies' and type in the location for the GPIB-32.lib that applies to your project. Save the settings and do the same for the other platform's project definitions. Remember to remove 'Pragma Comment(lib,"GPIB-32.lib") from ICS_Decl.h and your project should link correctly to both .lib without having to change the name of the .lib files.

Summary

This application note provided working examples on how to convert a 32-bit application to a 64-bit application in Visual Studio 2005. The directions also apply to Visual Studios 2008 and 2010. The examples can be used with other GPIB test programs and as guidance when converting other 32-bit programs with library files to 64-bit programs.

There are many issues that may arise while developing a 64-bit application. For debugging questions and further information visit Microsoft Support at: http://windows.microsoft.com/en-US/windows-vista/32-bit-and-64-bit-Windows-frequently-asked-questions