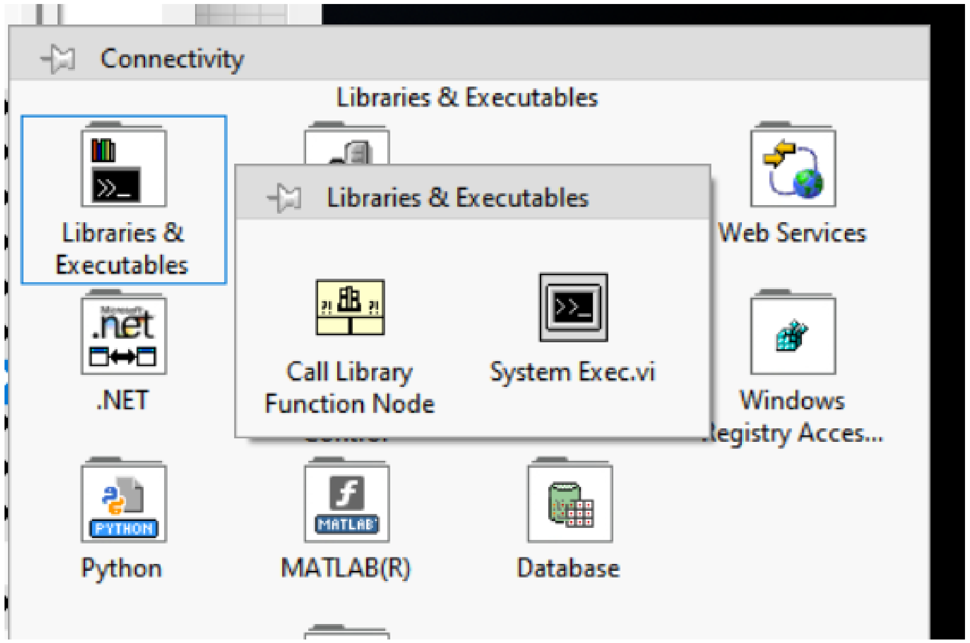
Connection with other languages

LabVIEW can call external, compiled codes with two options:

1). Using the command line with **System Exec.vi**
2). Call dynamic linked libraries (DLLs) with the **Call Library Function node**

These 2 functions can be found at **Connectivity>>Library & Executables** palette.

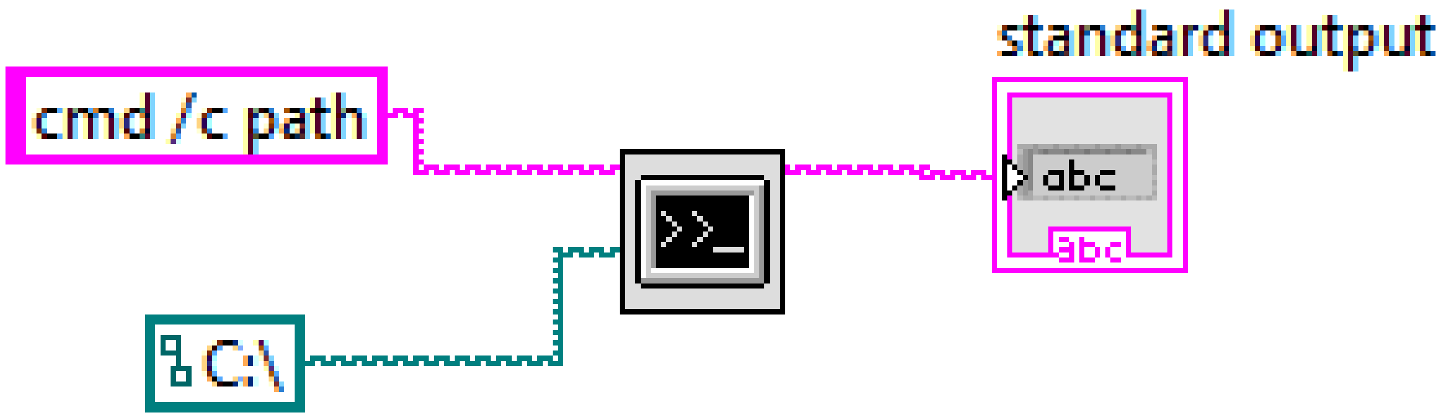


The following procedure shows how to use the System Exec VI:

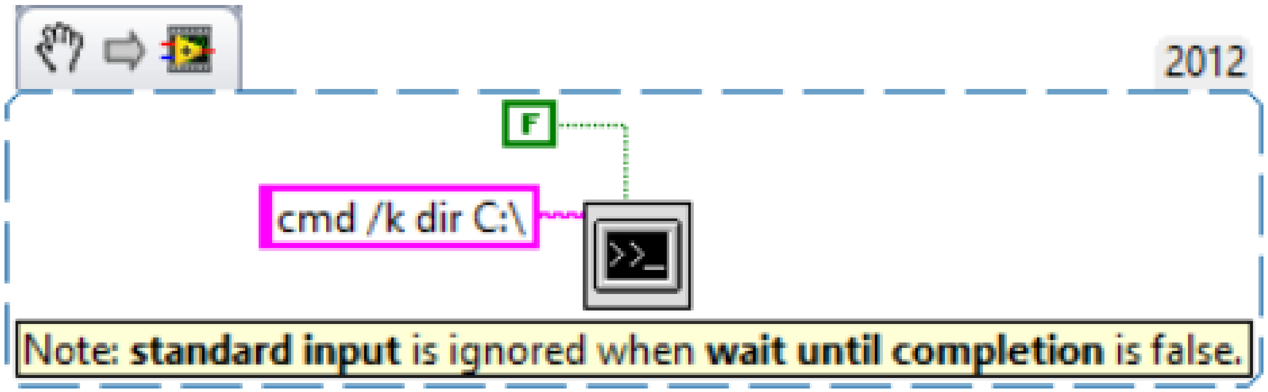
1. On the block diagram, place a System Exec VI
2. Right-click the **command line** terminal and select **Create Constant**. Input the argument needed to launch the command line (e.g. for Windows, cmd). Enter the entire command as you would type it at the DOS prompt, including all parameters.
3. Right-click the **standard input** terminal and select **Create Constant**. This terminal allows you to redirect input to the program that you are running. For example, if you were running a batch file with a "pause" statement, you could input an end of line through the "standard input" to simulate pressing "Enter" on the keyboard.
4. Determine whether you would like LabVIEW to open the command prompt window.
   By default, the command prompt window does not open. LabVIEW will open it in the background, run the command, and return the result. This can be adjusted by using /k before the code is sent to the command line (see example below).
   In certain circumstances, it is necessary to include the new line character (\r\n) after the command.
5. Determine whether you would like to halt code execution until the command prompt returns a result. Configure **wait until completion?** parameter accordingly.
   By default, LabVIEW will wait until the result is returned before continuing with the program.
6. (Optional) Continue configuring the VI, using LabVIEW content Help as a guide for the other non-required parameters.
7. (Optional) To view the command line output in LabVIEW, right-click the **standard output** and select **Create Indicator**.

The following code executes the dir command on the C:\ directory and outputs the result to a String Indicator. The command line does not open, and the program halts code execution until the command has returned results.

Assignment 1:



Alternatively, the implementation shown below will open a command prompt window and execute the dir command on the C:\ within the command prompt. The /k argument forces the command prompt window to stay open so that you can see the results of the dir command. Using this method, LabVIEW and the command prompt window will run asynchronously. In other words, after starting the command window and passing the command to it, LabVIEW will continue executing the next VI in your code while the command prompt window responds to the dir command.



Note: **standard input** is ignored when **wait until completion** is false.

Call A DLL (C++)

https://knowledge.ni.com/KnowledgeArticleDetails?id=kA03q000000YGggCAG&l=en-US

There are multiple ways to import code from other program languages into LabVIEW. To determine which method you should use, consider the following:Is your library a **C/C++ DLL** , **Microsoft .NET Assembly/.NET DLL or ActiveX DLL**? To determine what type of library you have, you can check the Portable Executable (PE) header for the DLL files.

If you are using a Microsoft .NET Assembly, continue to the *Import .NET Assembly Functions with Constructor Node* section below.

If you are using an **ActiveX DLL** *go to section calling Active X.*

Do you have a **header (*.h)** file for your C/C++ DLL?

If you do not have a header file, continue to the *Manually Configure DLL Functions Using Call Library Function Node* section.

What **data types** does your function(s) utilize?

If the function(s) you want to call utilizes supported data types, you can proceed with using the Import Shared Library Wizard. Continue to the *Import Functions with Import Shared Library Wizard* section.

If your function(s) use complex or unsupported data types, you can use the Call Library Function node to import and format the functions manually. Continue to the *Configure DLL Functions Using Call Library Function Node* section.

Import DLL Functions with Import Shared Library Wizard

If you have a header file and you are using supported data types, you can use the Import Shared Library Wizard to import your DLL functions into LabVIEW. This tool parses the header file, lists the functions in the shared library, converts data types in the shared library to LabVIEW data types, and generates a wrapper VI for each function. The wizard saves the VIs in a LabVIEW project library. Continue to the *Use Header File with the Import Shared Library Wizard* section.

A full set of instructions to configure the Import Shared Library Wizard is available in the LabVIEW Help. Briefly, you can start your import by:

1. Launch LabVIEW and navigate to **Tools** >> **Import...** >> **Shared Library (.dll)** to launch the Import Shared Library Wizard.
2. Select **Create VIs for a shared library** and then **Next**
3. Input the file paths for the **Shared Library (.dll) File** and **Header (.h) File**.
4. Continue configuring each page as needed, importing your desired functions, and selecting **Next**.
5. When finished with configuration, select **Finish** to create your LabVIEW Project library (.lvlib) file.

For an example of how to import a DLL using the Import Shared Library Wizard, follow Example: Importing Functions from a Shared Library File from the LabVIEW Help.

Assignment 2:

In this assignment, you will need to import the DDL generated with C++, to LabVIEW. This is a set of functions that is used to control the Deformable Lens that has 32 actors, a key component for an adaptive optics (AO) system.
The DLL consists of the **UsbHvDriverWfc.dll** and the C++ header file **actuatorsdriverplugin.h** (attached0, provided by Dynamic Optics.
Shows the list of all the imported LabVIEW Functions., in the LabVIEW Block Diagram.

Note: the 3 functions cannot be imported and MSUT be excluded:

getPluginInfo();
pd_getProperties();
pd_getInfor().