

# Import er4commplib into labVIEW

## Revision History

Date	Version	Description
22/05/2024	1.0	First version of document

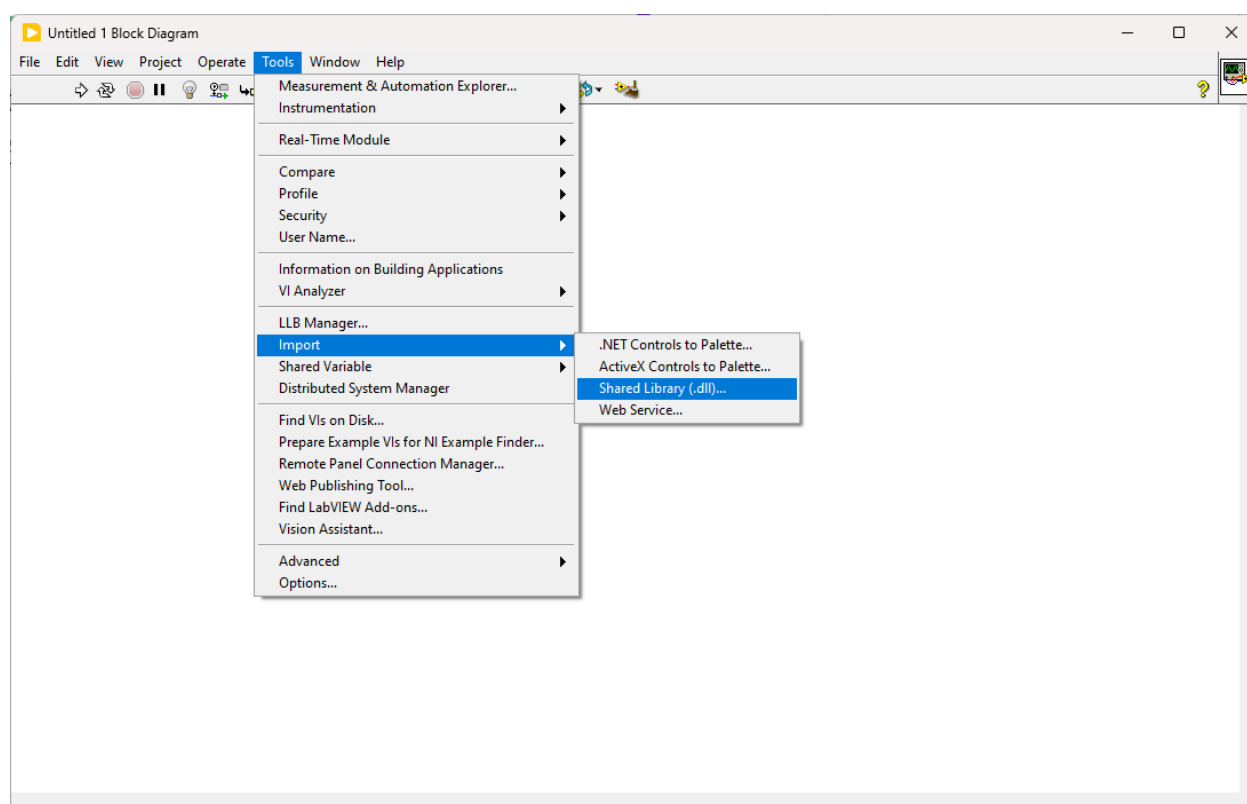


# Import C++ dll in labview

The Labview compatibility between Elements devices is guaranteed by the utilization of a dynamic C++ library directly importable with the labview wizard.

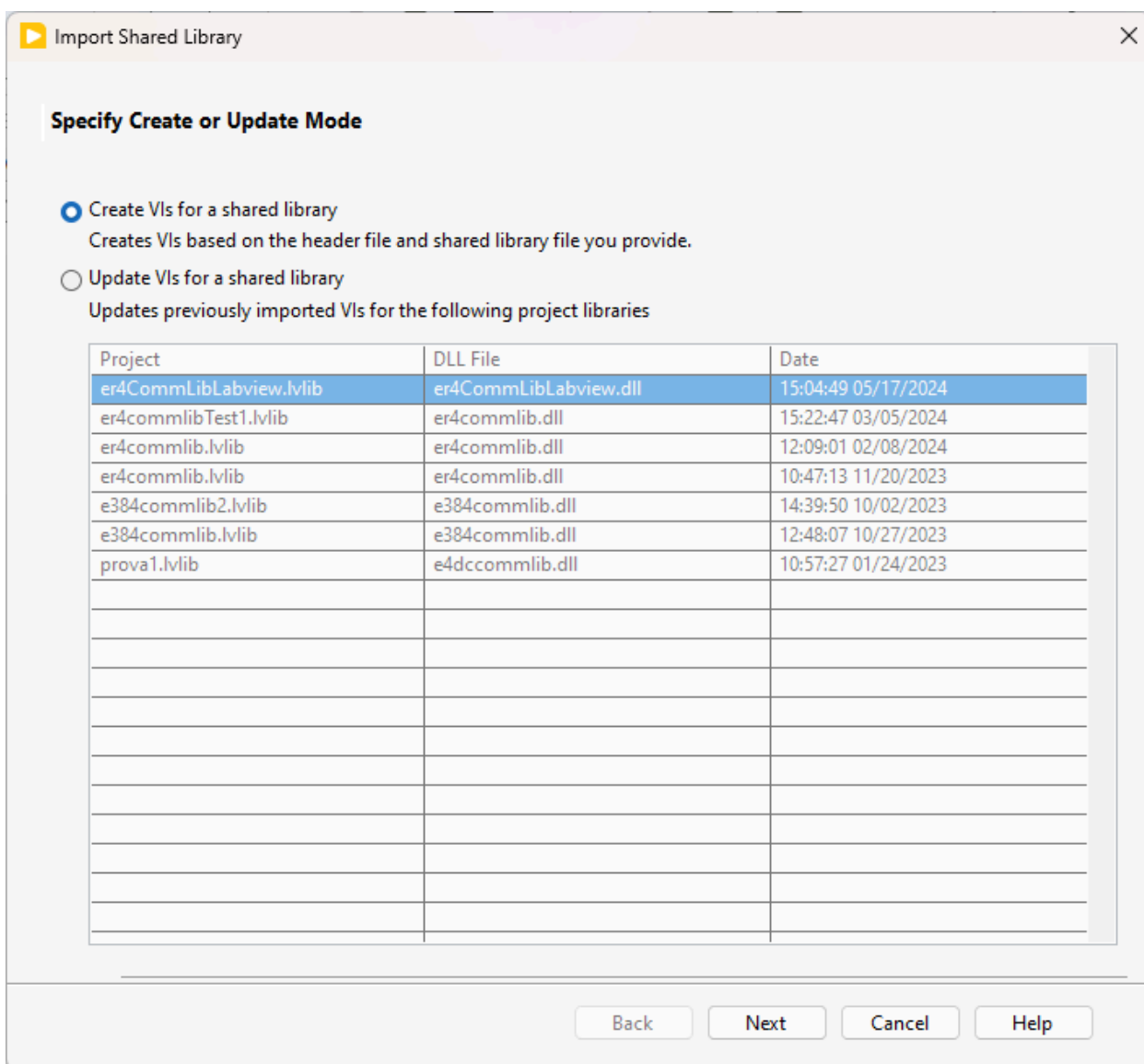
## Steps

Create a new VI then click on tools > Import > Shared Library (.dll).



if you never imported our dll click on Create VIs for a shared library.

If you already have a lvlib and you only want to update it you can click on update and select the appropriate lvlib file, this is more practical as most fields will already be filled.



Choose the path of the dll to import and make sure to include the er4commplib\_labview header file and to click on “Shared library file is not on the local machine”.



Import Shared Library

Select Shared Library and Header File

Shared Library (.dll) File

C:\ElemLibraries\er4CommLib\lib\release\er4CommLibLabview.dll

☒ Shared library file is not on the local machine

Header (.h) File

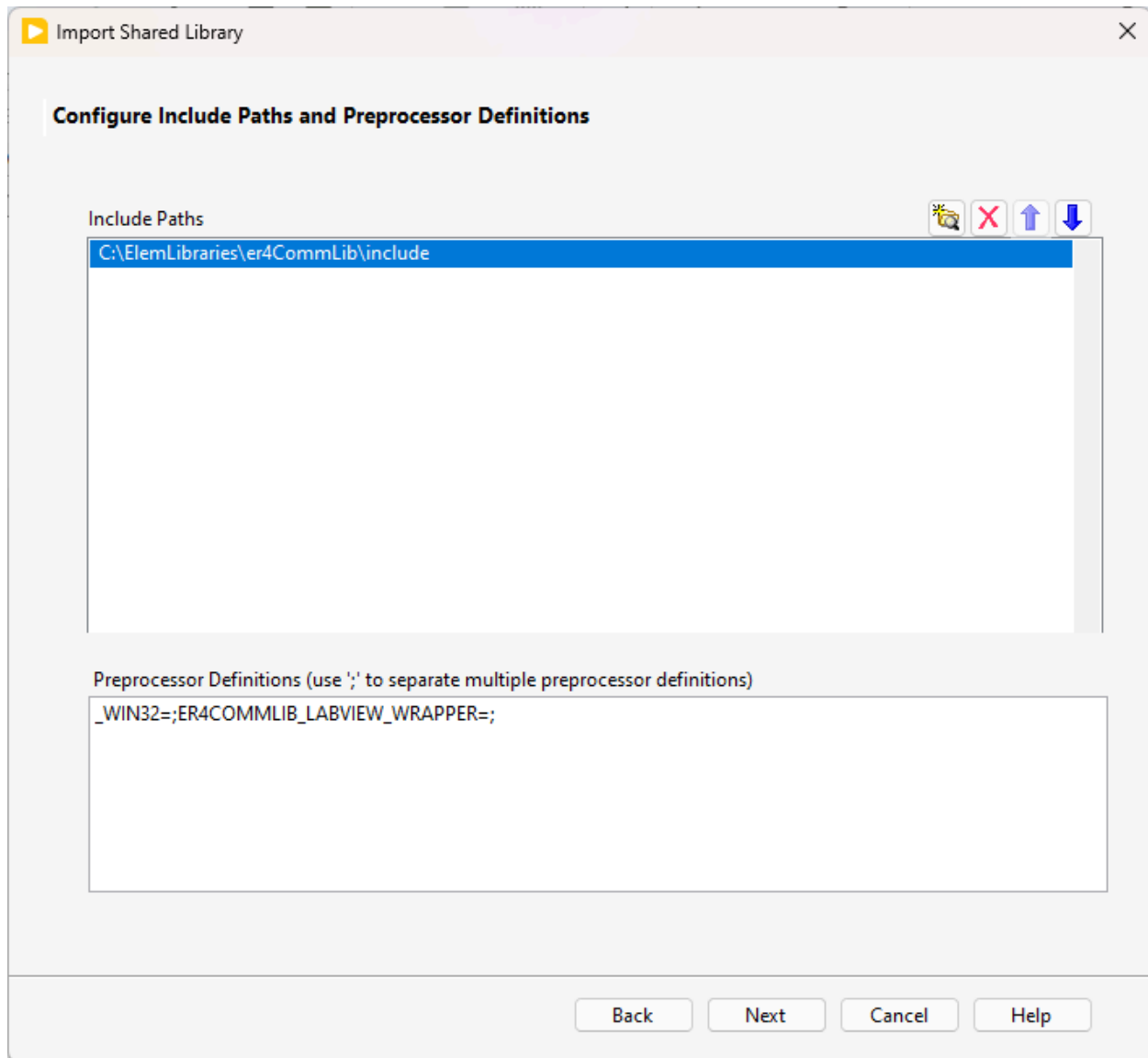
C:\ElemLibraries\er4CommLib\include\er4commLib\_labview.h

Back Next Cancel Help

Then under “Include Paths” choose the path to the library include directory and under preprocessor definitions write the following:

Unset

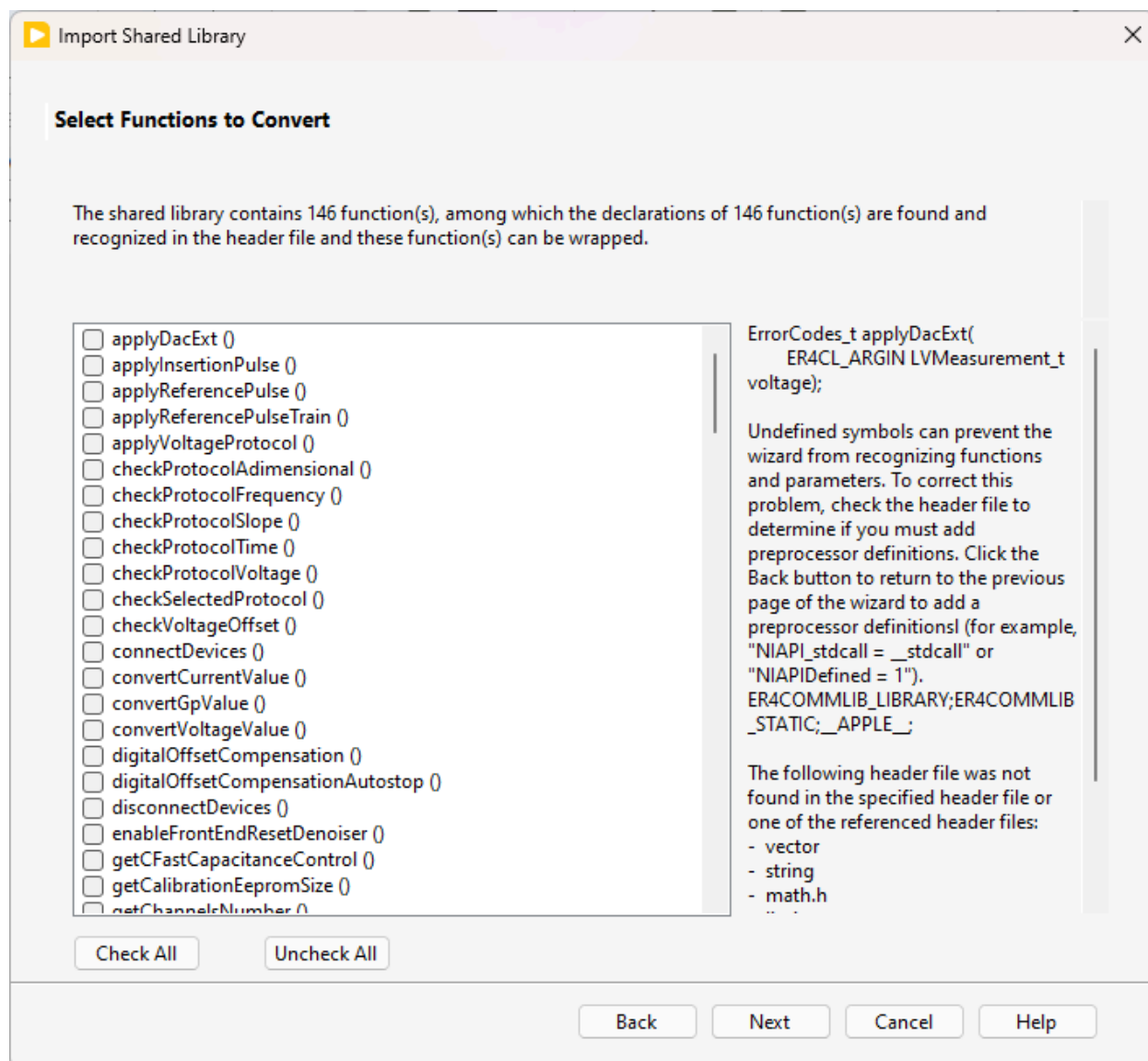
```
_WIN32=;ER4COMMLIB_LABVIEW_WRAPPER=;
```



The parsing of the header file could take a few seconds, if it gets stuck for several minutes and/or Labview crashes please contact our support team.

After the completion of the parsing process you will see a list of functions.

Click check all to generate the corresponding VIs.



In the next step use the default settings or modify them as you wish.



Import Shared Library

### Configure Project Library Settings

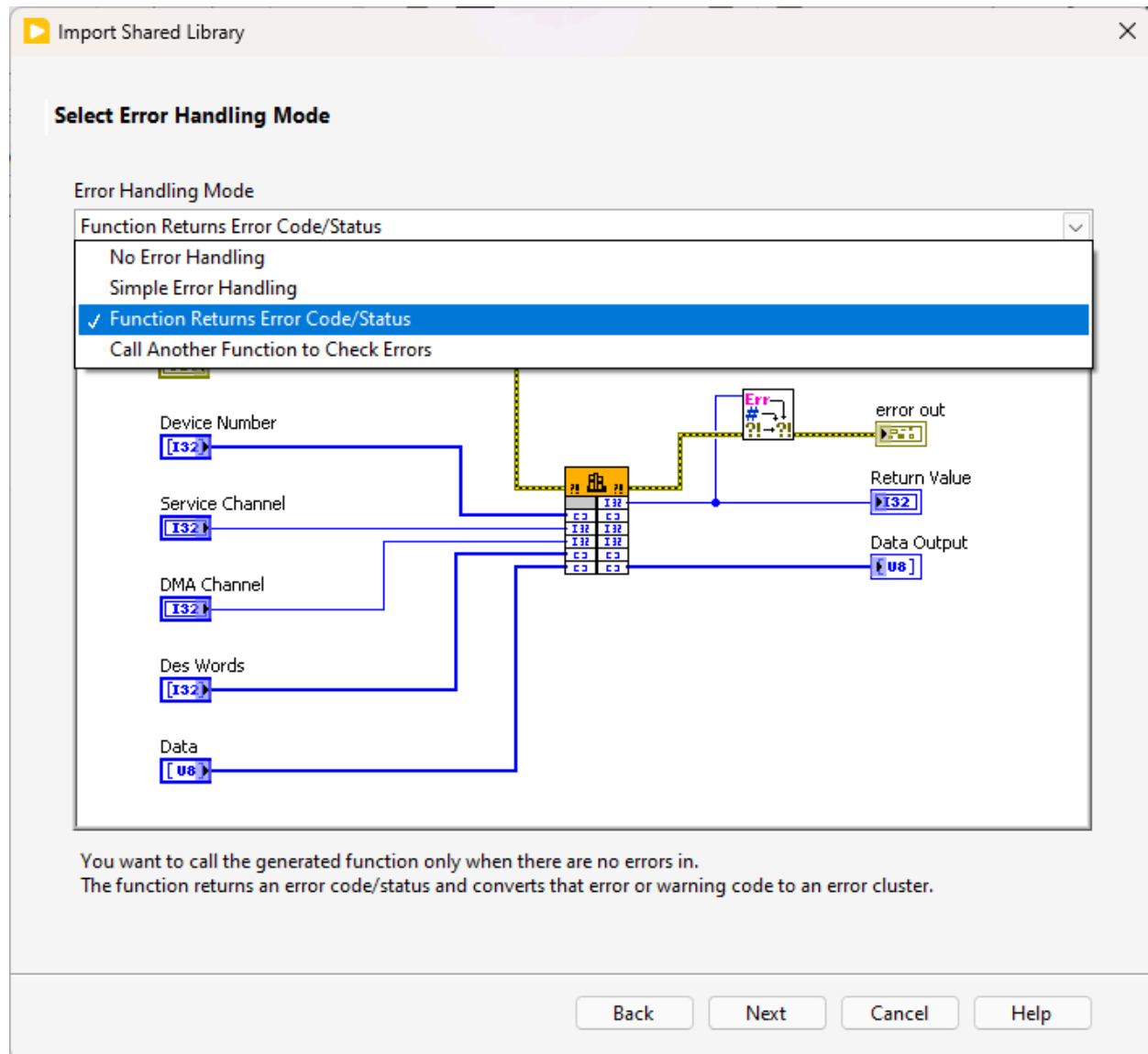
Project Library Name (.lvlib)  
er4CommLibLabview.lvlib

Project Library Path  
C:\Program Files\National Instruments\LabVIEW 2022\user.lib\er4CommLibLabview

☐ Copy the shared library file to the destination directory.

Back Next Cancel Help

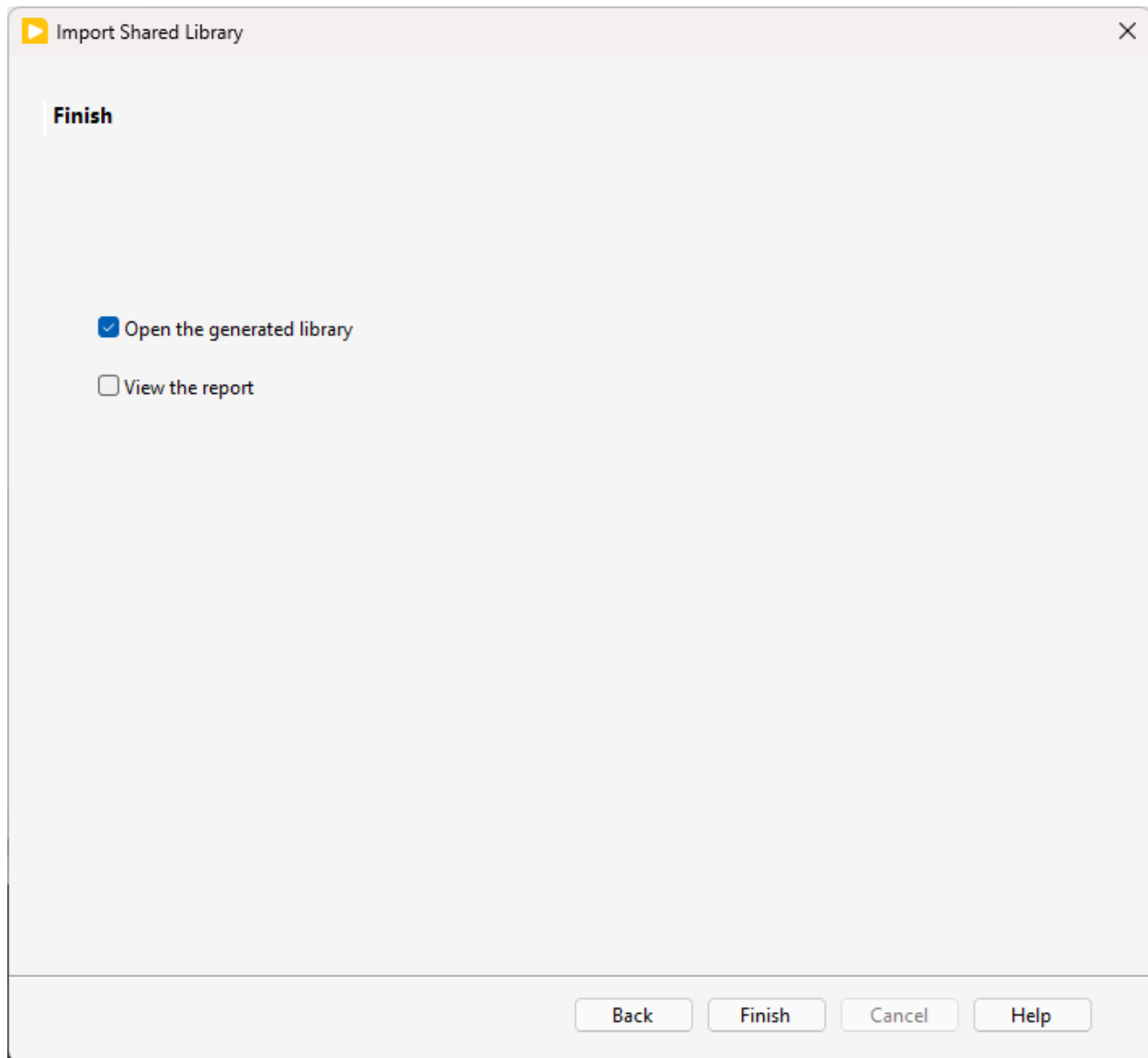
Select the Function Returns Error Code/Status in the drop down menu.



Under the Configure VIs and Controls select “Run in any thread” instead of “Run in UI thread”, this would enable the application to be multi-threaded instead of single-threaded. This would result in a more fluid experience.

Click next another couple of times and the Generation step should begin.

After the generation process is complete make sure to enable “View the report”



If everything was successful no errors should be listed in the report.

To correctly read data from the device open the read data vi and modify it accordingly to the following screenshot.



Call Library Function

Function Parameters Callbacks Error Checking

return type  
dataToRead  
dataRead  
buffer

Current parameter

Name buffer

Type Array

Constant ☐

Data type Signed 16-bit Integer

Dimensions 1

Array format Array Data Pointer

Minimum size 4194304

Function prototype

```
uint32_t readData(uint32_t dataToRead, uint32_t *dataRead, int16_t *buffer);
```

[Consider using a wizard instead...](#)

OK Cancel Help



---

## Usage

A simple example of the basic APIs to be used is shown in the file `main.cpp`. The only exception is that the method `detectDevices` is not implemented in the `labVIEW` wrapper: by calling the `connectDevices` method with no arguments the program will automatically try to connect to the available device, or it will return an error if no devices are plugged.