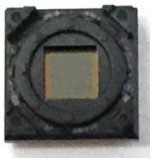
NSP32 SDK®
Reference Manual
for LabVIEW



ver 1.7

nanoLambda

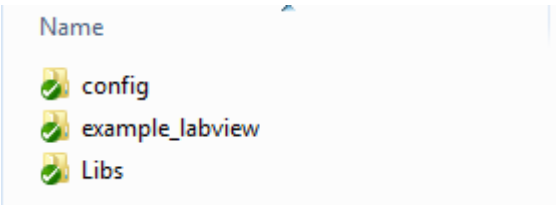
IMPORTANT NOTICE

nanoLambda Korea and its affiliates (nanoLambda) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to nanoLambda's terms and conditions of sale supplied at the time of order acknowledgment. This development kit is only for the purpose of performance evaluation and application development. nanoLambda assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using any components of the development kit. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards. nanoLambda does not warrant or represent that any license, either express or implied, is granted under any nanoLambda patent right, copyright, mask work right, or other nanoLambda intellectual property right relating to any combination, machine, or process in which nanoLambda products or services are used. Information published by nanoLambda regarding third-party products or services does not constitute a license from nanoLambda to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from nanoLambda under the patents or other intellectual property of nanoLambda. Reproduction of nanoLambda information in nanoLambdaI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. nanoLambda is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions. Resale of nanoLambda development kit and any of its components is not allowed. Decompiling, disassembling, reverse engineering or attempt to reconstruct, identify or discover any source code, underlying ideas, techniques or algorithms are not allowed by any means whatever. nanoLambda products are not authorized for use in safety-critical applications (such as life support) where a failure of the nanoLambda product would reasonably be expected to cause severe personal injury or death. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of nanoLambda products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by nanoLambda. Further, Buyers must fully indemnify nanoLambda and its representatives against any damages arising out of the use of nanoLambda products in such safety-critical applications.

Table of Contents

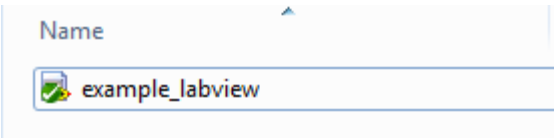
LabVIEW Example Project (‘example_LabVIEW.vi’)

An example project has a directory structure as below:

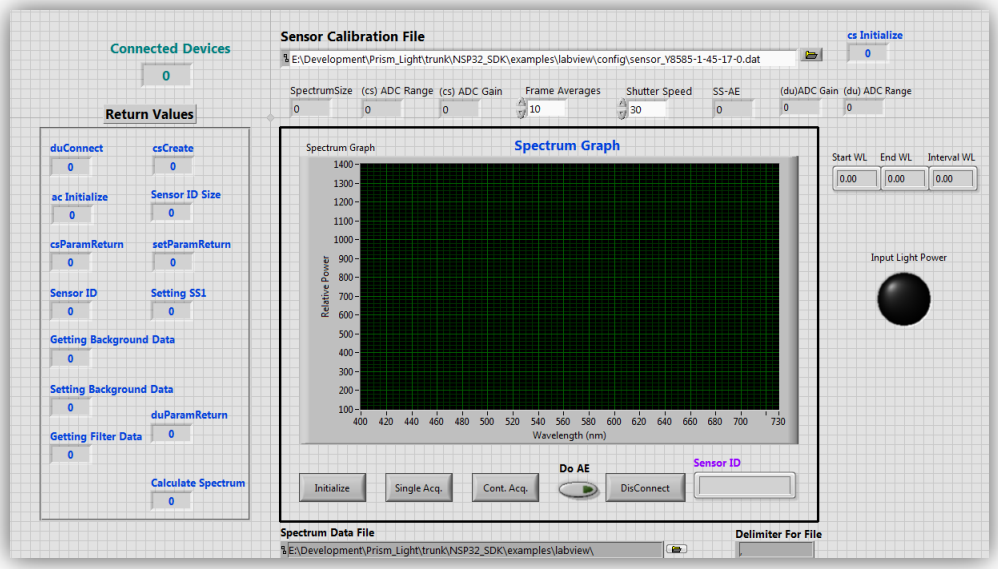


- "`config`" folder contains the sensor data file for the specific sensor.
- "`example_labview`" folder contains the labview exmaple.
- "`Libs`" folder contains all the Dynamic (.dll) libraries

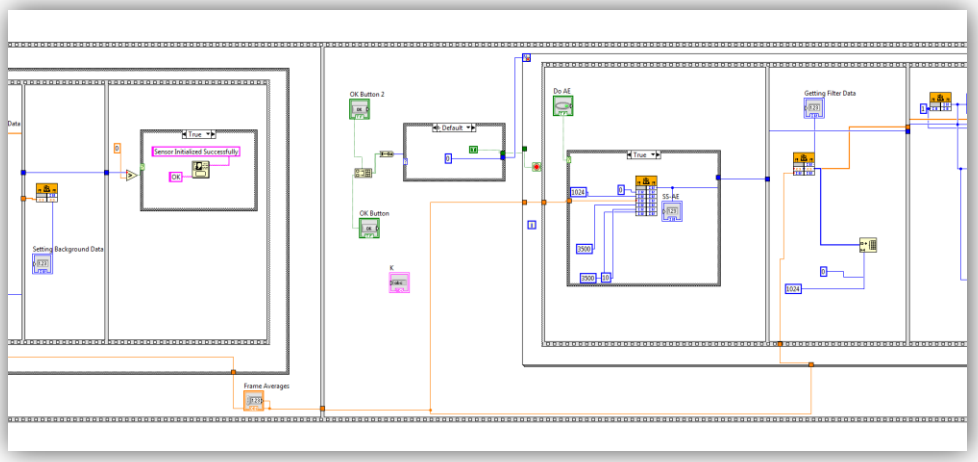
Start with an example project named with ‘example_labview.vi’ under "example_labview" folder:



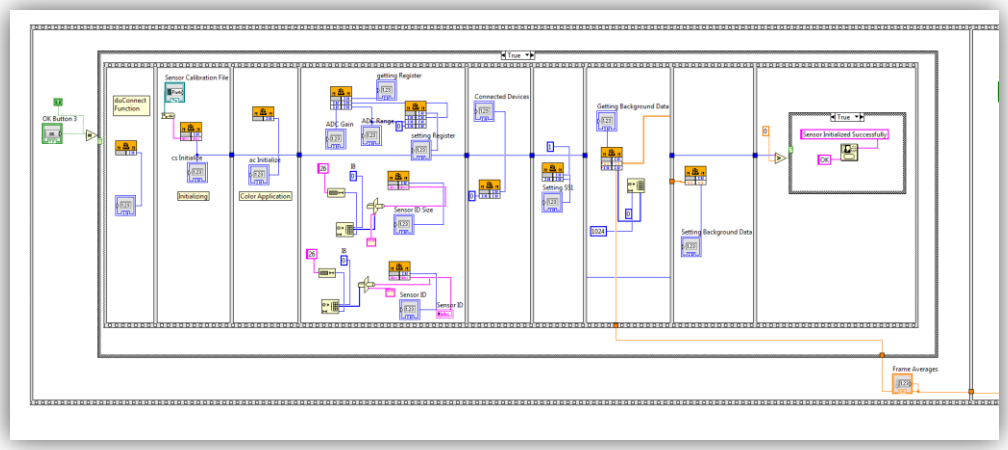
Front Panel



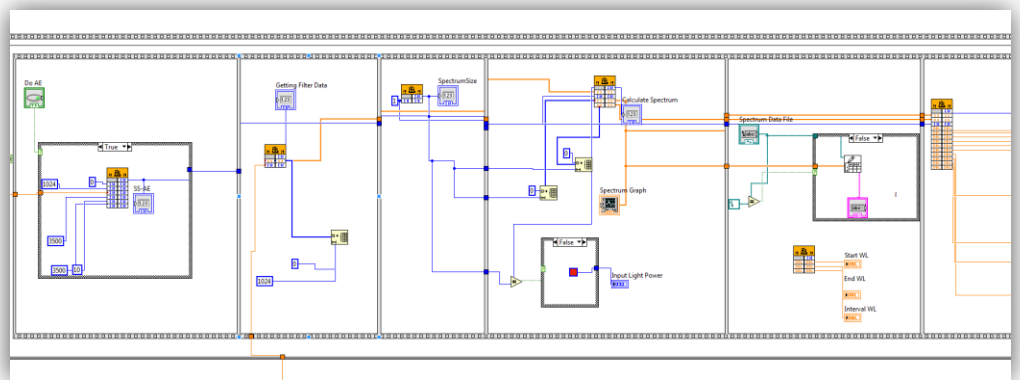
Back Panel



Initialization part



Main processing part



Connect API with LabVIEW (Example Project)

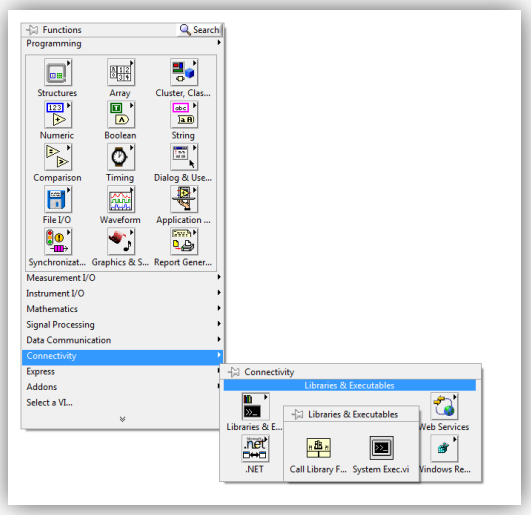
In initialization part, some of the important functions used are:

1) Connecting to the physical NSP32 device (CrystalPort.dll)
2) Creating a core open source object. (CrystalCore.dll)
3) Loading the sensor data file and registering it. (CrystalCore.dll)
4) Getting the sensor Id from the physical NSDP32 device (CrystalPort.dll)
5) Getting the sensor Id from the sensor data file (CrystalCore.dll)
6) Getting sensor parameters (ADC gain and ADC range) from the sensor data file (CrystalCore.dll)
7) Getting sensor parameters (ADC gain and ADC range) from the NSP32 device (CrystalPort.dll)
8) Setting sensor parameters (ADC gain and ADC range) to physical NSP32 device (CrystalPort.dll)
9) Activating NSP32 physical device with the index. Default is 0. (CrystalPort.dll)
10) Acquiring background data with shutter speed 1. (CrystalPort.dll)
11) Setting background data. (CrystalCore.dll)

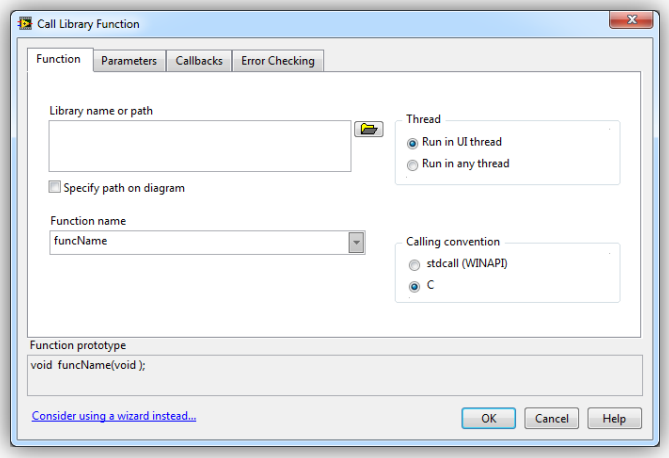
In Main Processing part, some of the important functions used are

1) Finding optimal shutter speed with AE (CrystalPort.dll)
2) Acquiring the Filter data at specific shutter from NSP32 device (CrystalPort.dll)
3) Calculating the spectrum data from the earlier acquired filter data. (CrystalCore.dll)
4) Find Spectrum Size (CrystalCore.dll)
5) Find the spectral information of the specific NSDP32 device (CrystalCore.dll)
6) Disconnecting the NSP32 device (CrystalPort.dll)

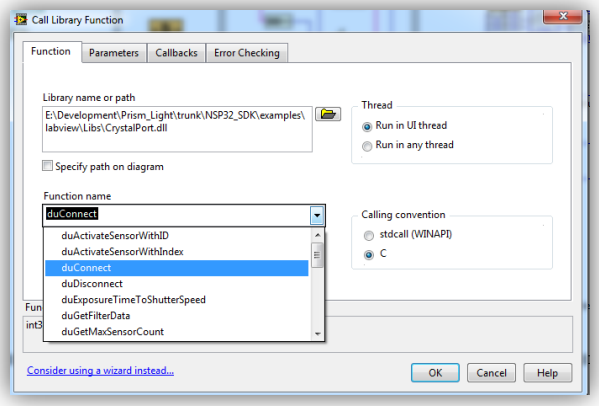
Right click on the back panel and then select "*Connectivity->Libraries& Executable->Call Library Function (CLF)*"



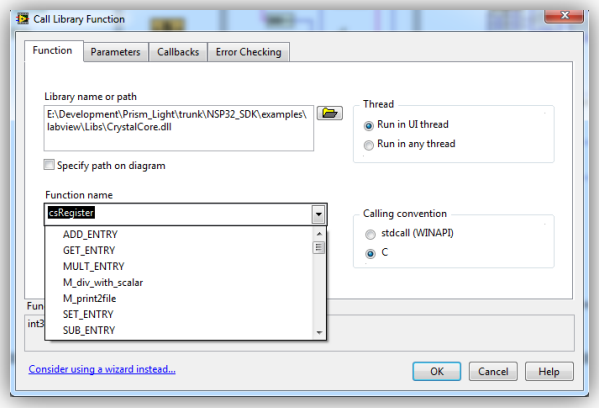
One VI  will be selected. Double click on it.



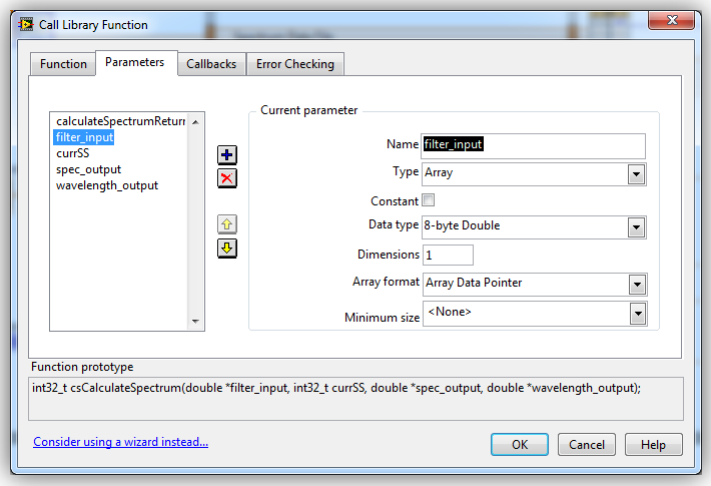
Select the path of the "`CrystalPort.dll`" or "`CrystalCore.dll`" depending on the function you want to use. Please make sure all other DLLs such as "`CrystalBase.dll`" and other are also in the same path. After select "`CrystalPort.dll`" or "`CrystalCore.dll`", all the available functions for interface with LabVIEW will be appear in the Drop down list of '*Function name*' like this.
For `CrystalPort.dll`,



For `CrystalCore.dll`,



Select the specific function and click '*Parameters*' tab from the top. For example, let's select the function "`csCalculateSpectrum`". After selecting the function, open the header file ('`nsp_spectrum_api.h`') from the NSP32_SDK/include folder and refer to that function's declaration, and set the correct return and arguments parameters in '*Current parameter*' section manually.



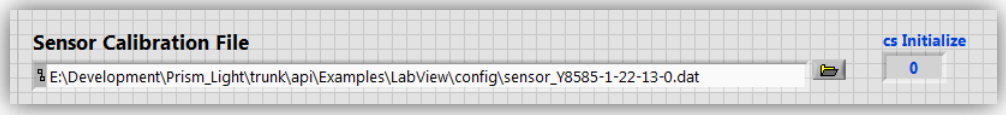
For more information, double click on the any CLF and check the return and arguments settings.

LabVIEW Example Project – Simple Tutorial.

First three steps must be done before running the application. For successful initialization of sensor , '*sensor_[SENSOR_ID].dat*'(for example, *'sensor_Y8585-1-85-85-0.dat*') file path is needed.

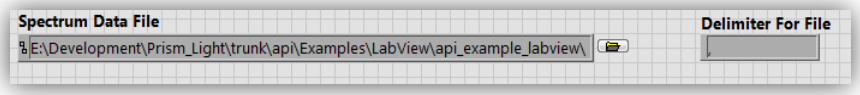
GUI Controls

1) File path for NSP32 spectral sensor calibration data:



Enter the path of the NSP32 spectral sensor calibration file. e.g. '`E:\LabVIEWProjects\config\sensor_[SENSOR_ID].dat`'. This file must be available in the "`config`" folder in the USB memory stick, shipped with sensor. "`csInitialize`" is an indicator which will show the return from the "`csRegister`" function. For more details about functions like "`csRegister`", please refer to " *NSP32 SDK Reference Manual for Cpp-v1.7.pdf*".

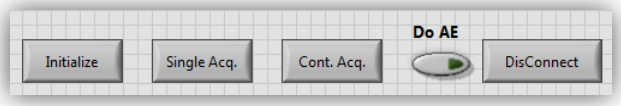
2) File path for Spectrum data dump in file: (Optional)



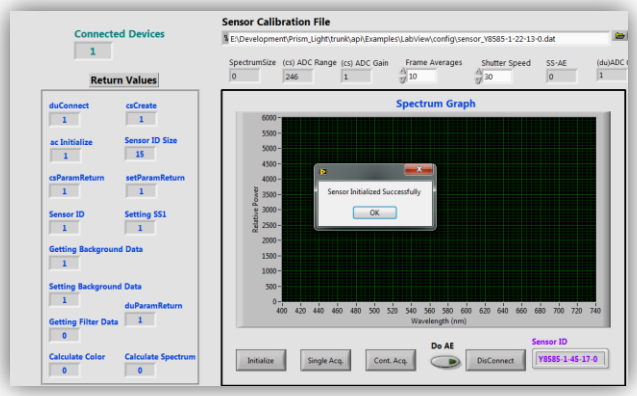
This is an optional function and can be utilized if you want to save the spectrum data in file. For example, if you want to save the data in .csv format then first select the path of the .csv file. The default delimiter for CSV format file is COMMA(',). But you can change the delimiter to other character like TAB character('\t') according to your preference.

Now you can run the example from LabVIEW. Run in Continuous mode. For that please press this button.  . This button is available on left top.

3) Control buttons for initialization, single and continuous spectrum acquisitions:

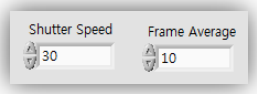


For the initialization of the sensor, please press "*Initialize*" button. If initialization is succeed, then you will see a message box displaying the successful message.



If you presses "*Single Acq.*" or "*Cont. Acq.*" button before sensor initialization is done will result in fatal error. After initialize sensor, you can press either "*Single Acq.*" or "*Cont. Acq.*" for single acquisition or for continuous acquisition of spectrum data.

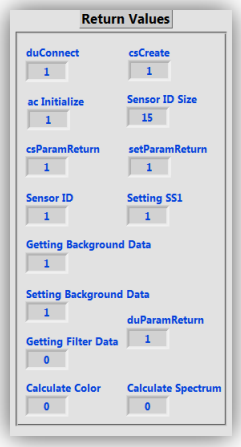
4) Controls for setting '*Shutter Speed*' and '*Frame Average*':



From this control panel, you can set the '*Shutter speed*' and '*Frame average*'. You have to adjust '*Shutter Speed*' to get correct spectrum data under a specific input light condition. And if you increase '*Frame average*' number, then you will have better S/N spectrum.
Also you can enable '*Do AE*' button, this button will find the optimal shutter speed for the specific measurement conditions and then acquire the data with that optimal shutter speed.

AE Disabled AE Enabled

5) Indicators for API functions:



These all are indicators for different functions available in the API.

6) LED indicator for input light power status:
This LED indicator will display the status of input light source. If the power of input light source is too weak or strong, then spectrum data will not reliable one(distorted).

LED indicator	Description
	Black means process not yet started
	Gray means, input light intensity is too low. [Solution] Please increase shutter speed or input light intensity.
	Green means, Sensor response is good.
	Red means, sensor response saturated. [Solution] Please decrease shutter speed or input light intensity.

Spectrum Acquisition Examples

Fluorescent (in-door)