

PhotoniQ Windows GUI Example Description

This download includes a sample Microsoft Visual Studio project application that communicates with the PhotoniQ through the provided drivers. The drivers have been converted to C++ code (originally LabVIEW code), to cut out LabVIEW from the GUI interface entirely. The project was developed on Visual Studio 2022 on Windows 10 using C++14 for 64-bit systems.

The GUI was developed in C++ using the Microsoft Windows Api. To run the program, ensure that the sub system used is set to Windows (Project Properties -> Linker -> System), the “Windows Kits\10\Lib” directory is included in additional library directories (Project Properties -> Linker -> General), and that the “Windows Kits\10\Lib\10.0.22621.0\um\x64\hid.lib” and “Windows Kits\10\Lib\10.0.22621.0\um\x64\setupapi.lib” directories are included in additional dependencies (Project Properties -> Linker -> Input).

The drivers include four pairs of header and source code files and one additional header file for global variables used between all driver source files.

The Initialize files initializes and validates an interface with the PhotoniQ and gets the capabilities of the device for use by the Control Interface and Data Interface.

The Control Interface executes a control operation to a previously initialized PhotoniQ. Many helper functions are included to make sending control operations as easy as possible and documentation is included to explain what each command does.

The Data Interface is used to signal to the PhotoniQ to generate reports which are then parsed into events. Event data can then be written to a file. The Data Interface can also be signaled to stop generating reports based on certain conditions being met such as a specified number of triggers being read, an amount of time passing, or upon the change of a Boolean value. The contents of event data can be found in any PhotoniQ user manual.

The Close header and source file combines two separate driver functions of the LabVIEW drivers into one, dealing with both closing the connection to the PhotoniQ and also handling errors. A connection can be closed without throwing an error and should be done at the end of any program that interfaces with a PhotoniQ.

The GUI itself is created using multiple helper files which handle most of the functionality of the controls, indicators, and static text. The messaging handling is done in the Main Window source file along with the creation of the front panel itself. The GUI allows for the user to easily update the config table of the PhotoniQ, altering values such as the integration period, trigger rate, and the trigger source. Start and stop acquiring with the push of a button. When acquiring, the channel data will be displayed on a bar graph, and the trigger count, event count, and timestamp will be displayed in indicator textboxes.

This project is meant to give a developer a starting point for creating a C++ GUI that fulfills their needs and to make altering and adding to the GUI an easy process. The helper files for controls and various helper functions that are provided in said files, should reduce the necessary Windows Api knowledge one needs to be able to develop the GUI further substantially, and the examples of controls and indicators already provided should allow a user to pick up the project with ease.