

DLP® Advanced Light Control Software Development Kit for Lightcrafter™ Evaluation Modules User's Guide

1 Introduction

This document will guide you through the build process for the DLP® Advanced Light Control (ALC) Software Development Kit (SDK) for Lightcrafter™ evaluation modules (hereto referred as DLP ALC SDK). Please note that these instructions apply to versions 2.0 and later of the DLP ALC SDK. It will also demonstrate by example how to build the source code for the related TI Designs ([TIDA-00254](#), [TIDA-00361](#), [TIDA-00362](#)) by walking through the build procedure for TIDA-00254.

The following pieces of software are required to build the DLP ALC SDK:

1. CMake v3.1: <https://cmake.org/download/>
2. Qt v5.3.2 (for the MinGW 4.8 compiler): <http://download.qt.io/archive/qt/5.3/5.3.2/>
3. OpenCV v2.4.10: <https://sourceforge.net/projects/opencvlibrary/files/opencv-win/2.4.10/>
4. Doxygen v1.8.11: <http://www.stack.nl/~dimitri/doxygen/download.html>
5. TI DLP SDK v2.0: <http://www.ti.com/tool/DLP-ALC-LIGHTCRAFTER-SDK>

Be sure to download the proper file types for your system. These downloads are large and, depending on your web connection, can take a long time. Throughout this guide we will be using a 32-bit Windows 7 system. The file names and approximate sizes in this case are:

- *cmake-3.5.0-win32-x86.msi* (15 Mb)
- *qt-opensource-windows-x86-mingw482_opengl-5.3.2.exe* (737 Mb)
- *opencv-2.1.10.exe* (367.3 Mb)
- *doxygen-1.8.11-setup.exe* (24 Mb)
- *DLPSDK-2.0-windows-installer.zip* (7kB)

2 Installing Doxygen

This section guides the user through the installation of Doxygen. Doxygen is used to help generate documentation from the source code.

1. Using the link in [Section 1](#), download the Windows binary distribution. Click on the “ftp” link highlighted in [Figure 1](#).

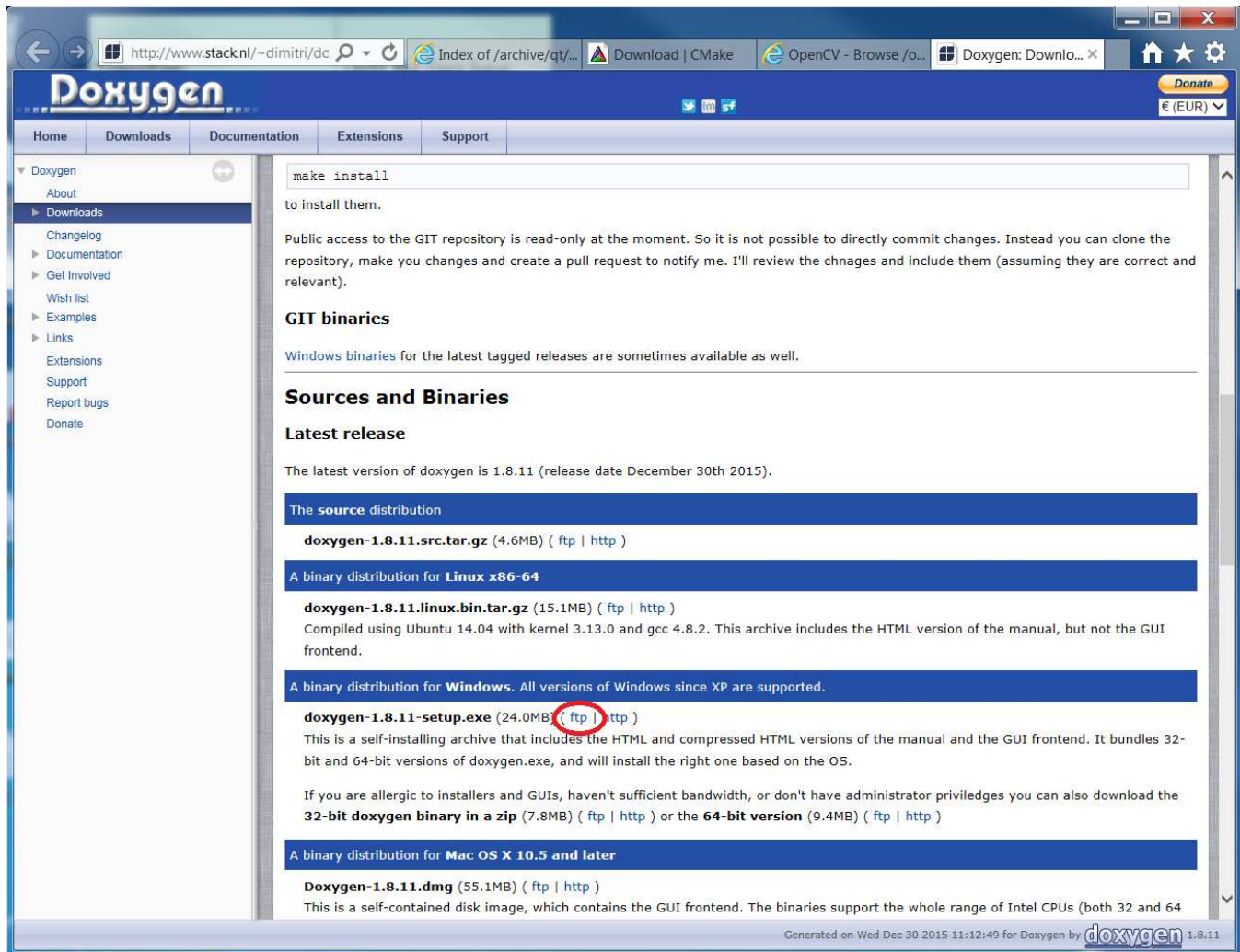


Figure 1. Doxygen Website

2. Run the downloaded file and follow the prompts through the Doxygen installer, leaving the options as their default values unless your system requires them to be changed.
3. Doxygen is now installed.

3 Installing Qt

This section guides the user through the installation of Qt ("cute"). Qt is an integrated development environment (IDE) used in a wide variety of applications. The Texas Instruments DLP ALC SDK uses the MinGW compiler bundled with Qt.

1. Download `qt-opensource-windows-x86-mingw482_opengl-5.3.2.exe` from the link given in [Section 1](#). [Figure 2](#) shows the Qt website and indicates which file to download.

Note: The DLP ALC SDK uses C++ 11 and will not compile using an out of date compiler.

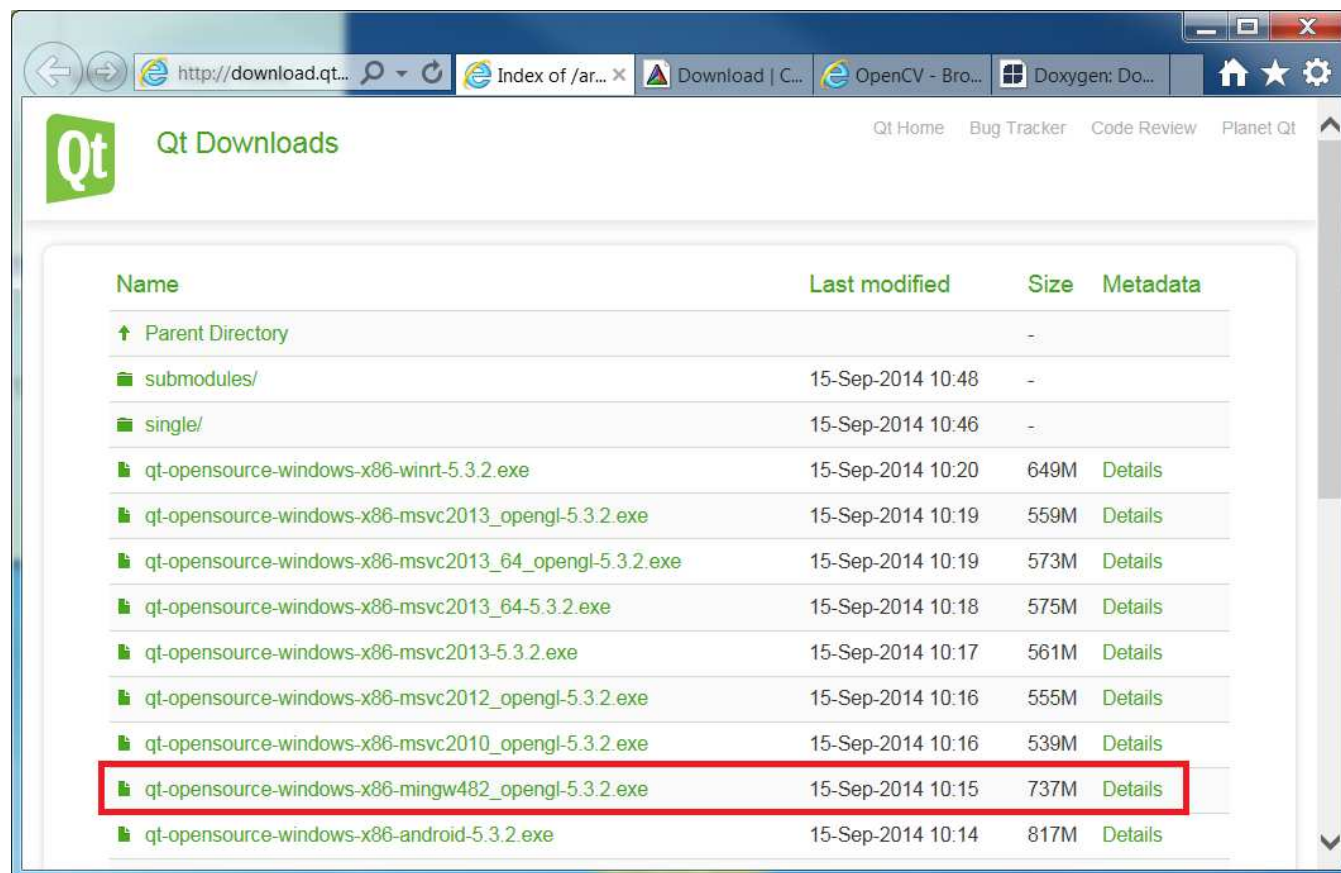


Figure 2. Qt Website

2. Run the downloaded executable file. Click the "Next" button shown in [Figure 3](#).

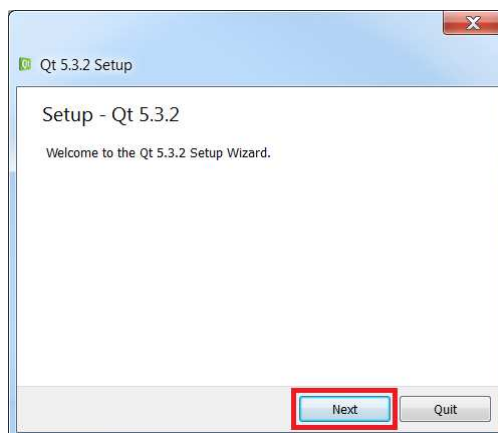


Figure 3. Qt Initial Run

3. Select an installation path as shown in [Figure 4](#) and click “Next.”

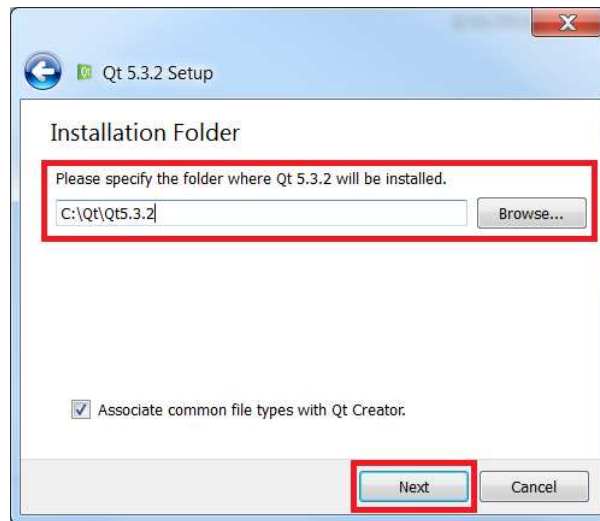


Figure 4. Qt Install Folder

4. In the “Select Components” window, expand the “Tools,” drop-down. As indicated in [Figure 5](#), check the box next to MinGW 4.8.2. Click “Next.”

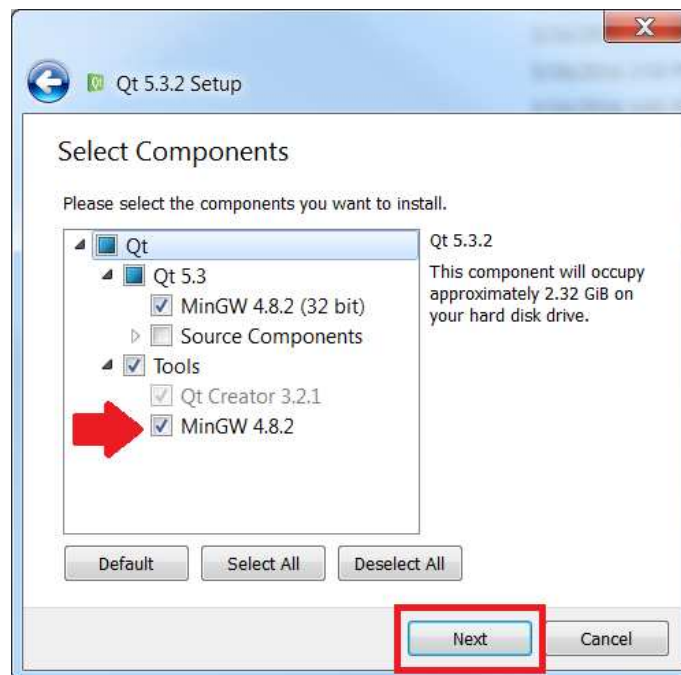


Figure 5. Install Components

- As shown in Figure 6, read and accept the license agreement and click “Next.”

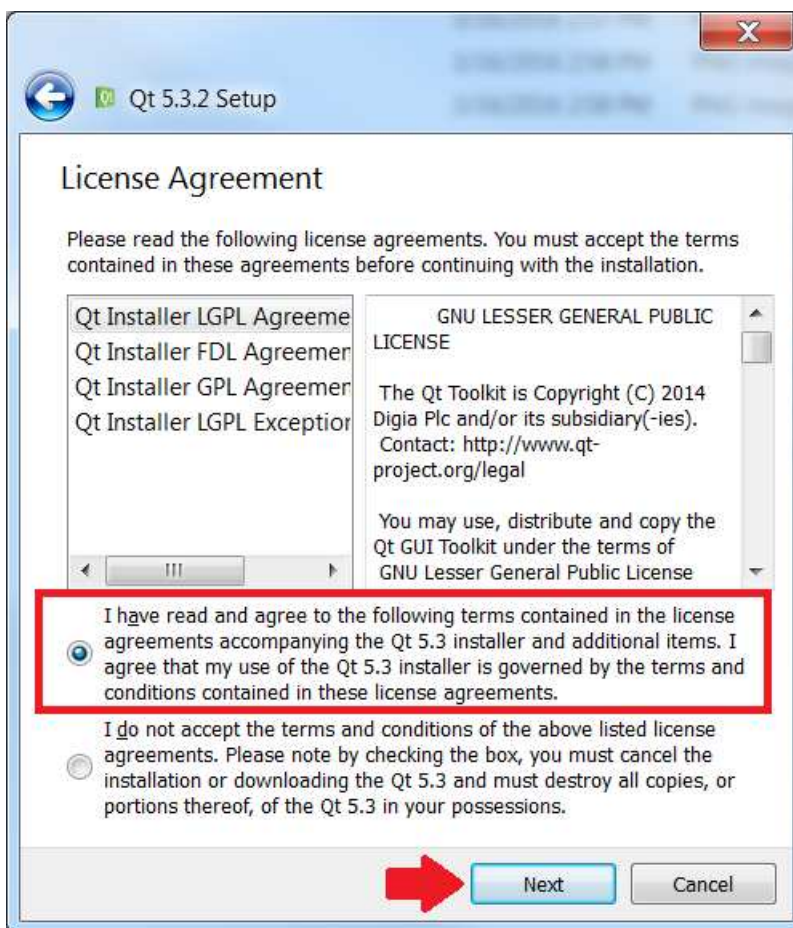


Figure 6. Qt License Agreement

6. If required by your system, select a different location in the Start Menu where you would like to create shortcuts. [Figure 7](#) shows the default value. Once completed, click “Next.”

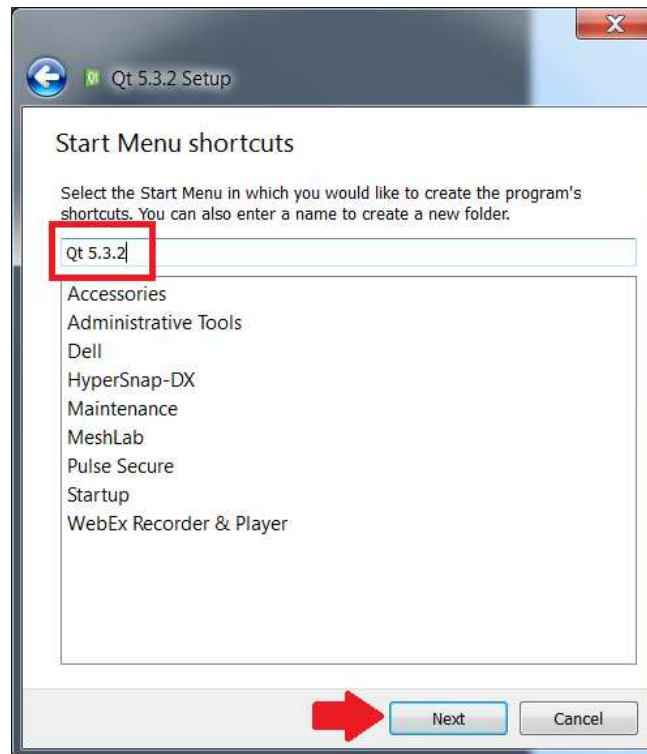


Figure 7. Qt Start Menu Shortcuts

7. Qt is now ready to install, click “Install” as shown in [Figure 8](#). Qt will begin installing. Wait for the installation to complete.

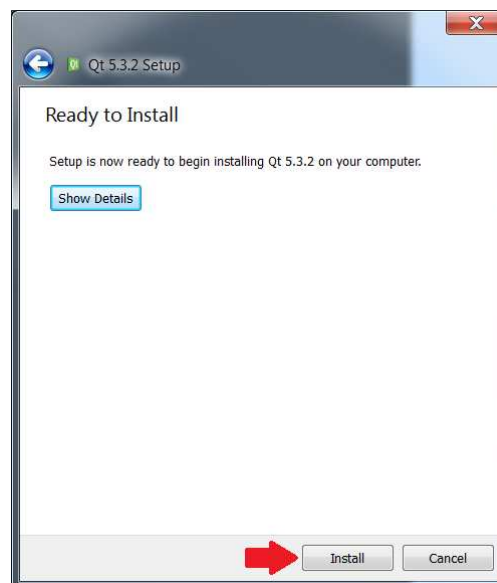


Figure 8. Qt Ready to Install

8. Once Qt has finished installing, click “Finish.” Uncheck the boxes for launching Qt Creator and opening the readme as in [Figure 9](#).

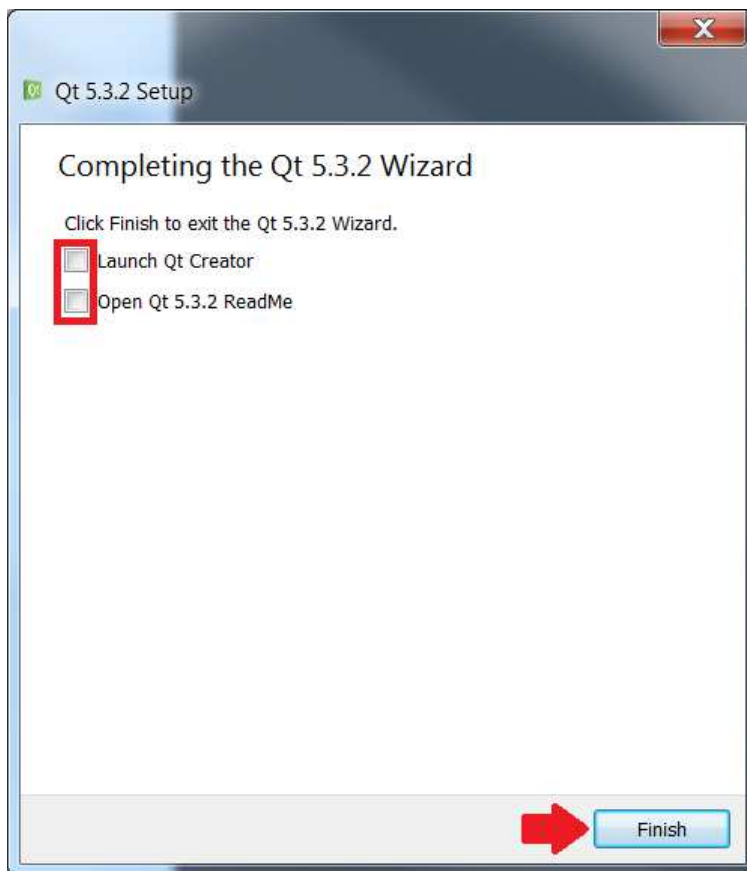


Figure 9. Qt Install Complete

9. Qt is now installed.

4 Configuring the System to Use MinGW

This section guides the user through modifying environment variables to add the MinGW compiler installed with Qt used by the DLP ALC SDK.

1. Using File Explorer, navigate to the Qt installation location and find the MinGW binary folder location. Copy the path. In this example, shown in [Figure 10](#), the path is C:\Qt\Qt5.3.2\Tools\mingw482_32\bin

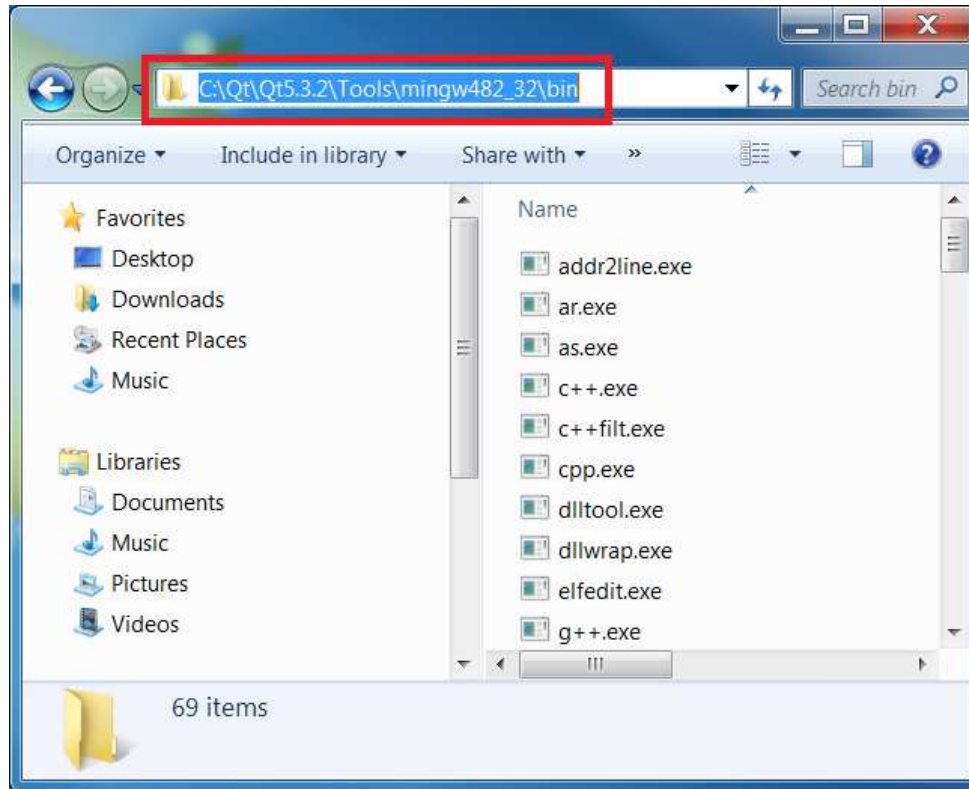


Figure 10. MinGW Path

2. In the Start Menu, search for “Edit the System Environment” and select “Edit the system environment variables.” Refer to [Figure 11](#).

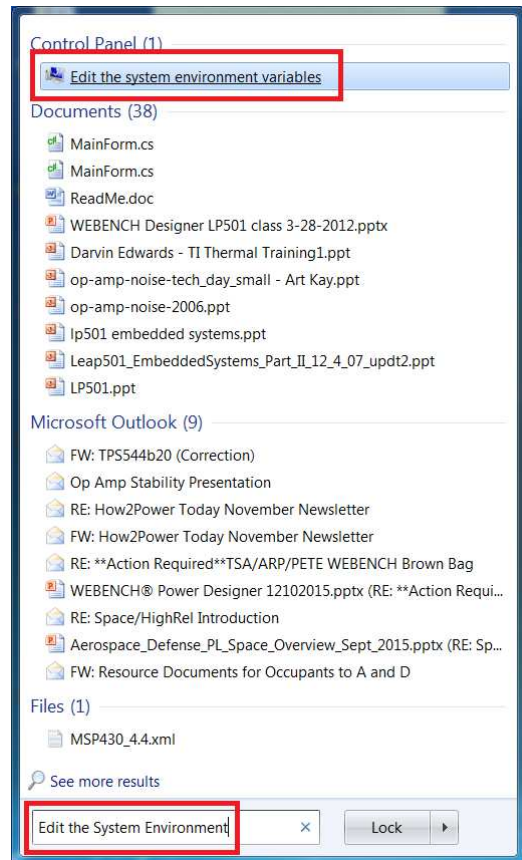


Figure 11. System Environment Variables

3. Once the system properties window opens, select the “Environment Variables” button in the “Advanced” tab as shown in [Figure 12](#).

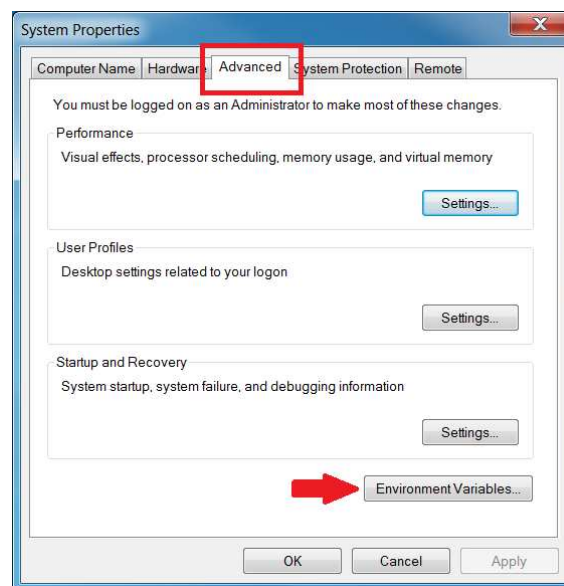


Figure 12. Open Environment Variables

4. In the system variables field, browse to the entry labeled “Path” or “PATH.” Highlight the entry and click “Edit” as shown in [Figure 13](#).

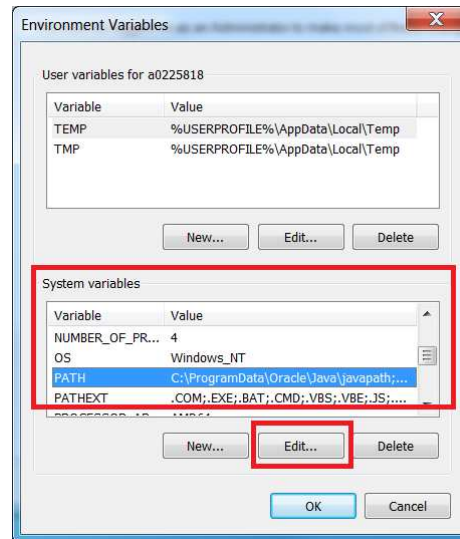


Figure 13. Edit PATH Environment Variable

5. Insert the MinGW path that was copied earlier at the end of the “Variable value” field as shown in [Figure 14](#). Click “OK” to make the changes.
Note: Make sure that the entries in this field are separated by a semicolon.

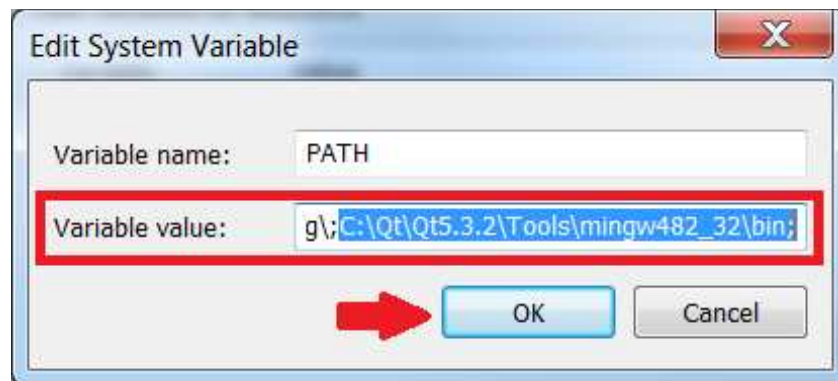


Figure 14. Insert MinGW Path

6. Click “OK” to get out of the Environment Variables window. Qt and MinGW have been installed and configured for the remaining steps in this build process.
Note: At this stage, restart the computer for the changes to take effect.

5 Installing CMake

This section guides the user on how to install CMake, which is used to build OpenCV and the DLP ALC SDK.

1. Download CMake from the link in [Section 1](#). Select the Windows installer as shown in [Figure 15](#).

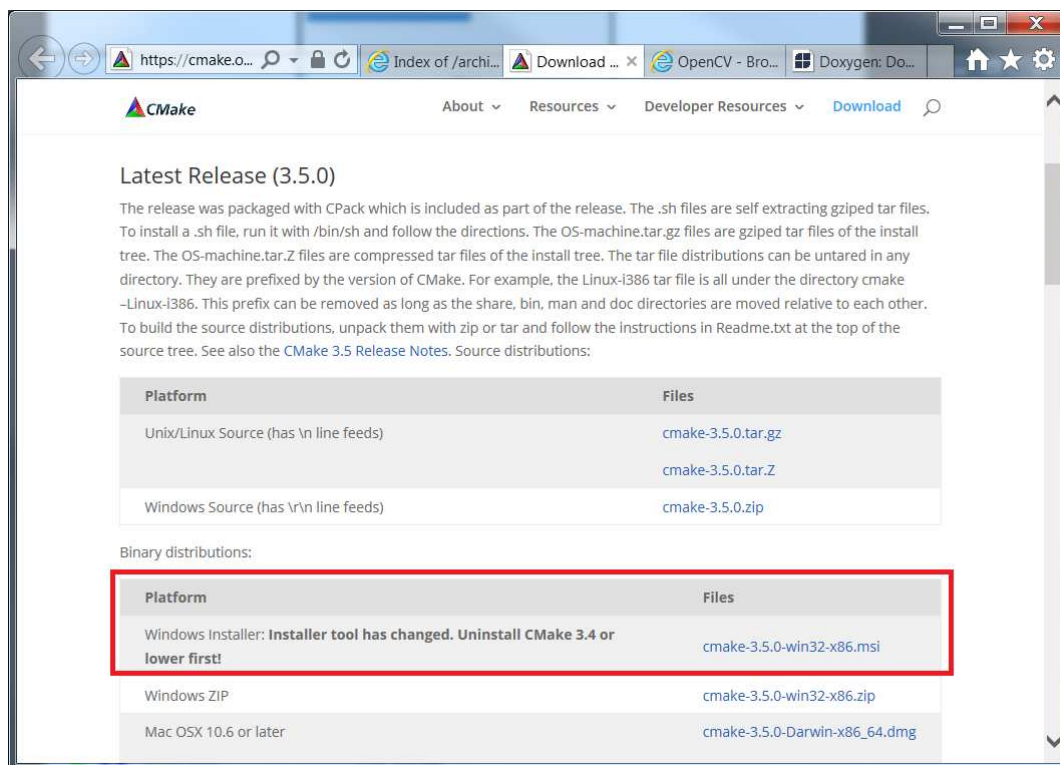


Figure 15. CMake Website

2. Once the download is complete, run the executable. Click "Next" as shown in [Figure 16](#).

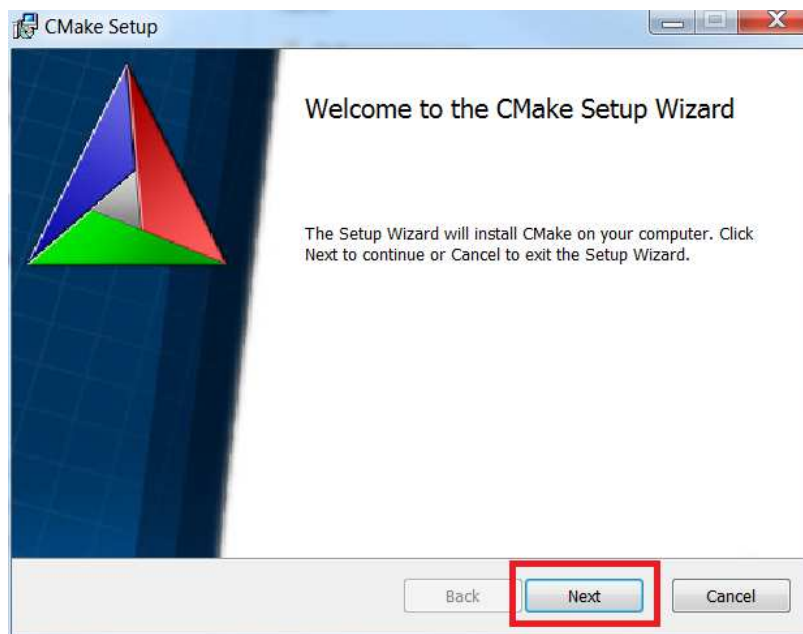


Figure 16. Open CMake Installer

3. Read and accept the license agreement. In the Install Options window, select the radio button indicating “Do not add CMake to the system PATH” as shown in Figure 17.

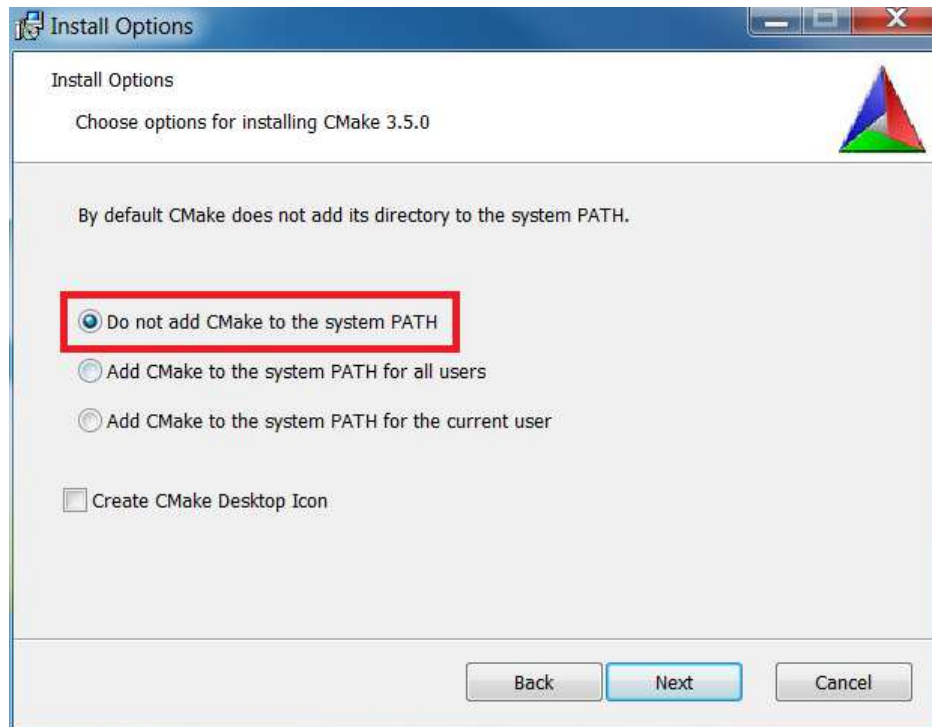


Figure 17. CMake Install Options

4. Choose a destination folder for CMake, click “Next” as shown in Figure 18.

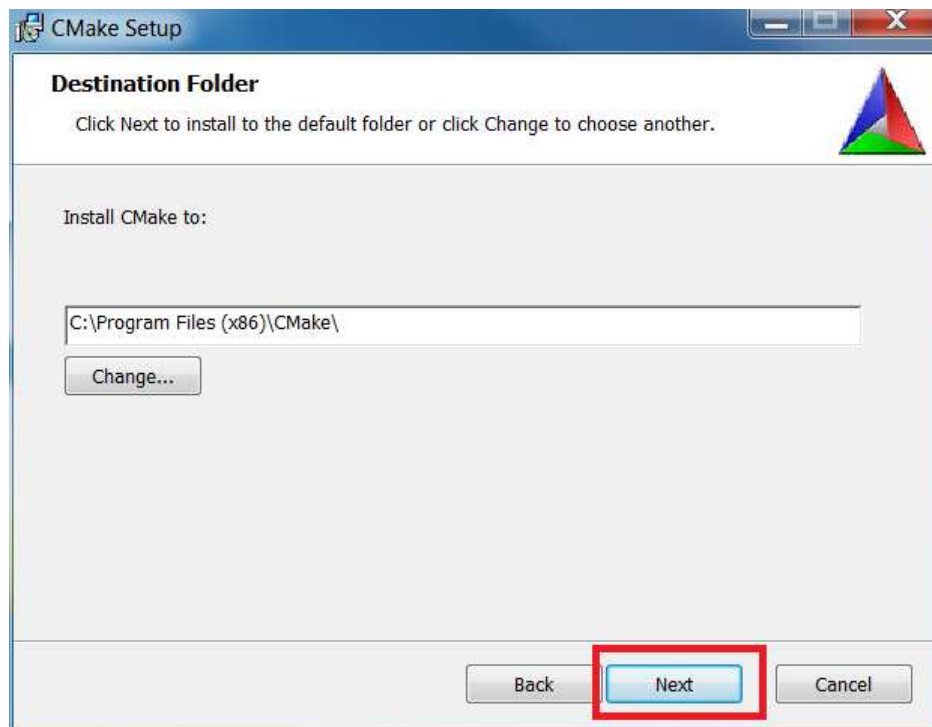


Figure 18. CMake Destination Folder

5. Click “Install” as shown in [Figure 19](#) and let CMake run.

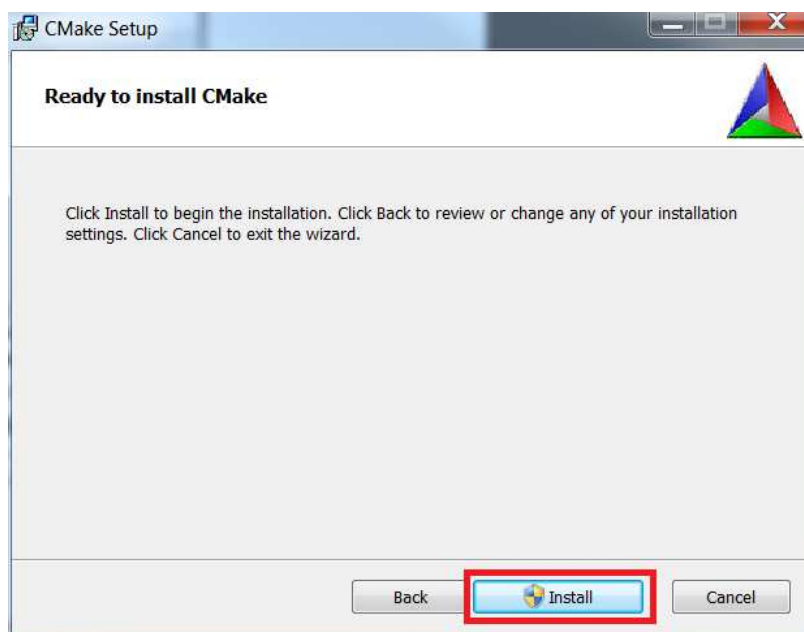


Figure 19. CMake Ready to Install

6. Click “Finish” as shown in [Figure 20](#) and CMake is now installed.

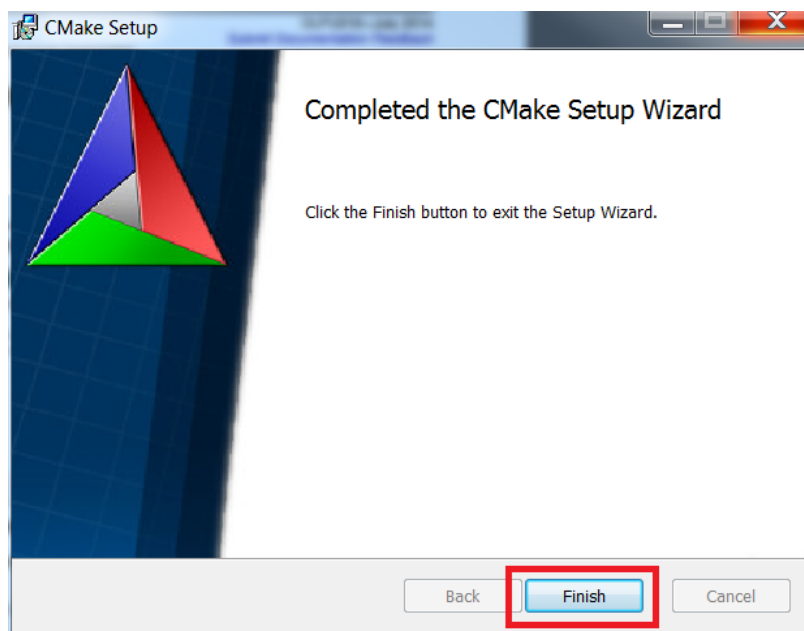


Figure 20. CMake Finished

6 Building OpenCV

OpenCV is an open source library dedicated to computer and machine vision applications. It is used by the DLP ALC SDK for its algorithms. This section guides the user on building OpenCV using CMake.

1. Download OpenCV from the link in [Section 1](#). [Figure 21](#) shows the link on the download page. Once the download is complete, run the executable.

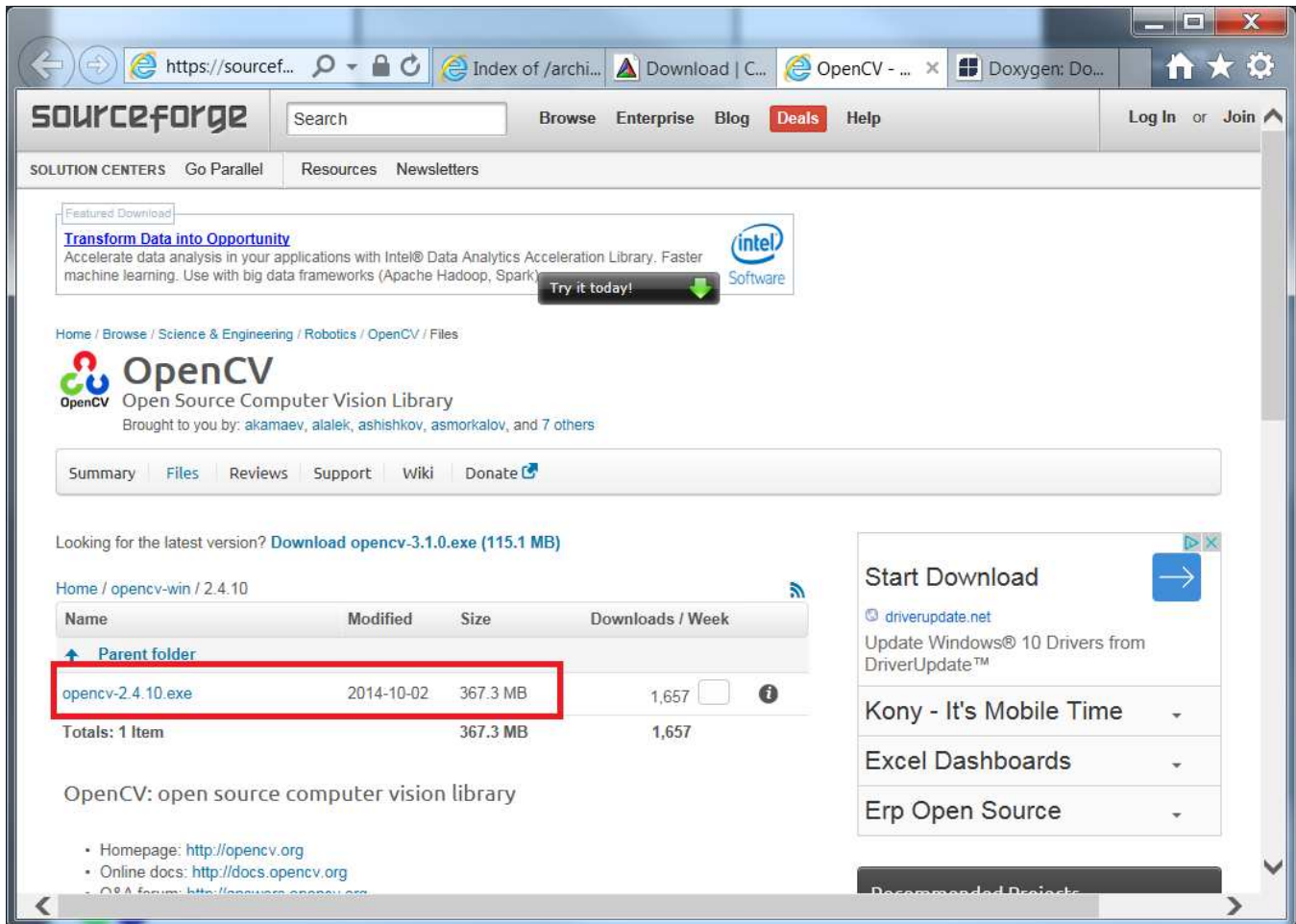


Figure 21. OpenCV Website

2. Choose a suitable place to extract the files and select “Extract” as shown in [Figure 22](#).

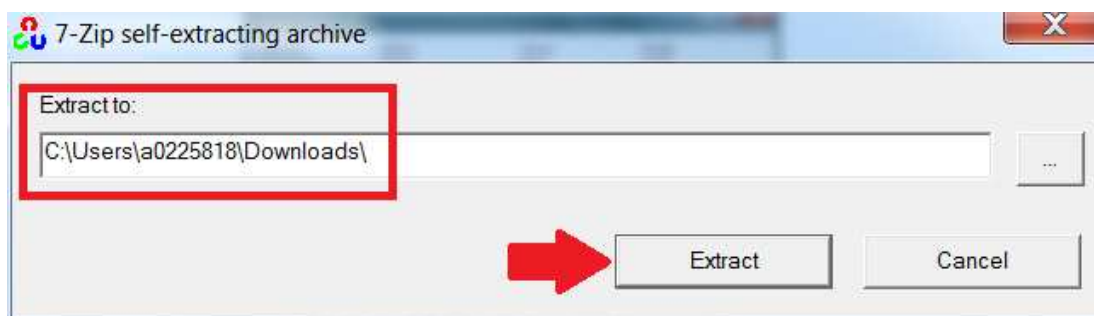


Figure 22. Extract OpenCV

3. Once the files have extracted, open the Start Menu. Search for and select “CMake (cmake-gui)” as shown in [Figure 23](#).



Figure 23. Find CMake to Build OpenCV

4. When the GUI is first opened, it should look like [Figure 24](#).

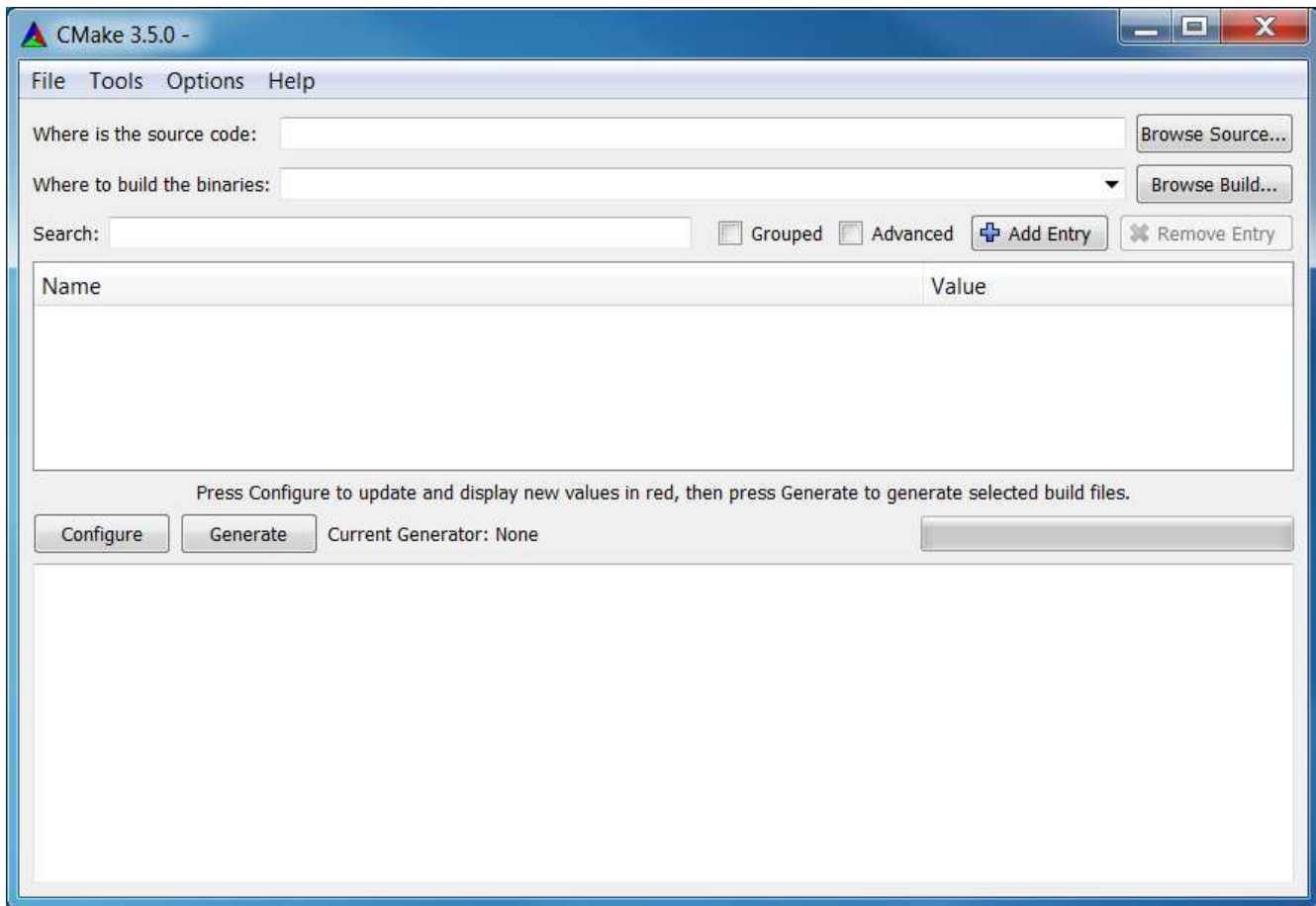


Figure 24. Default CMake GUI

5. Check the “Grouped” box and browse for the path of the OpenCV source files. The path should be of the form C:/Users/<username>/Downloads/opencv/sources as shown in [Figure 25](#).

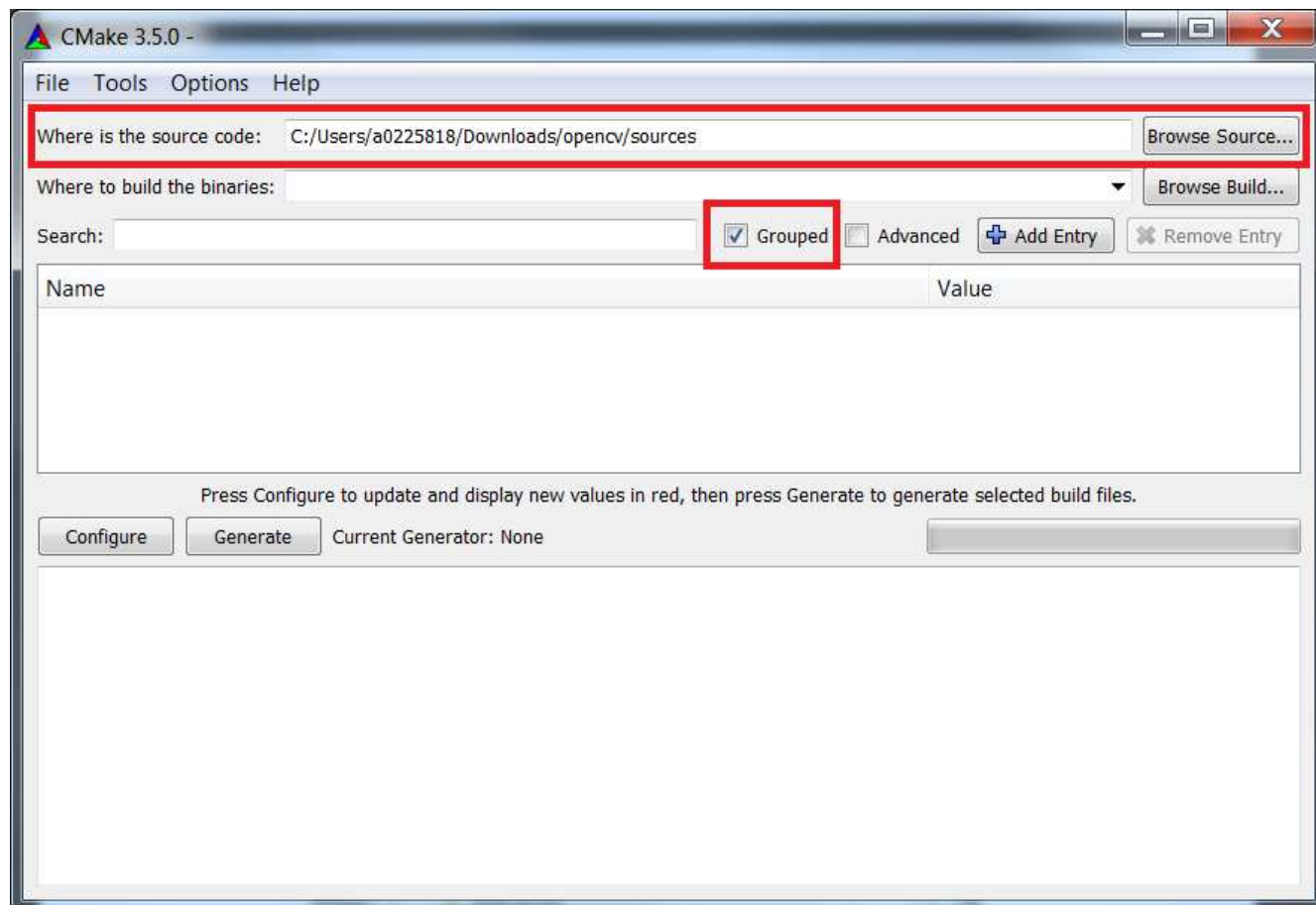


Figure 25. Set Source Path

6. Click “Browse Build” and create a new folder for the built OpenCV files as shown in Figure 26.

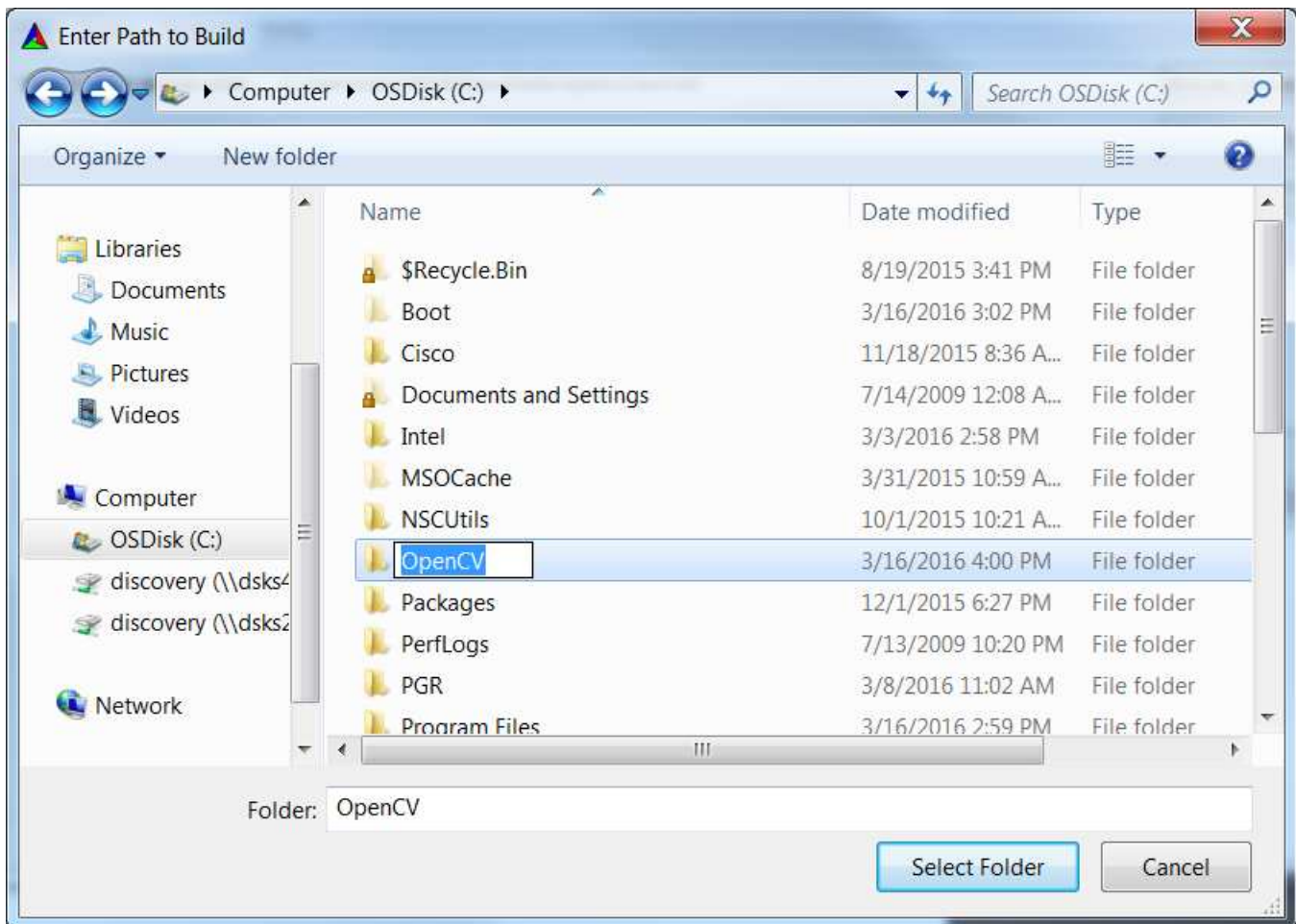


Figure 26. Build New Folder

7. Then click “Configure” in the CMake GUI. In the drop down, find “MinGW Makefiles” as shown in [Figure 27](#).

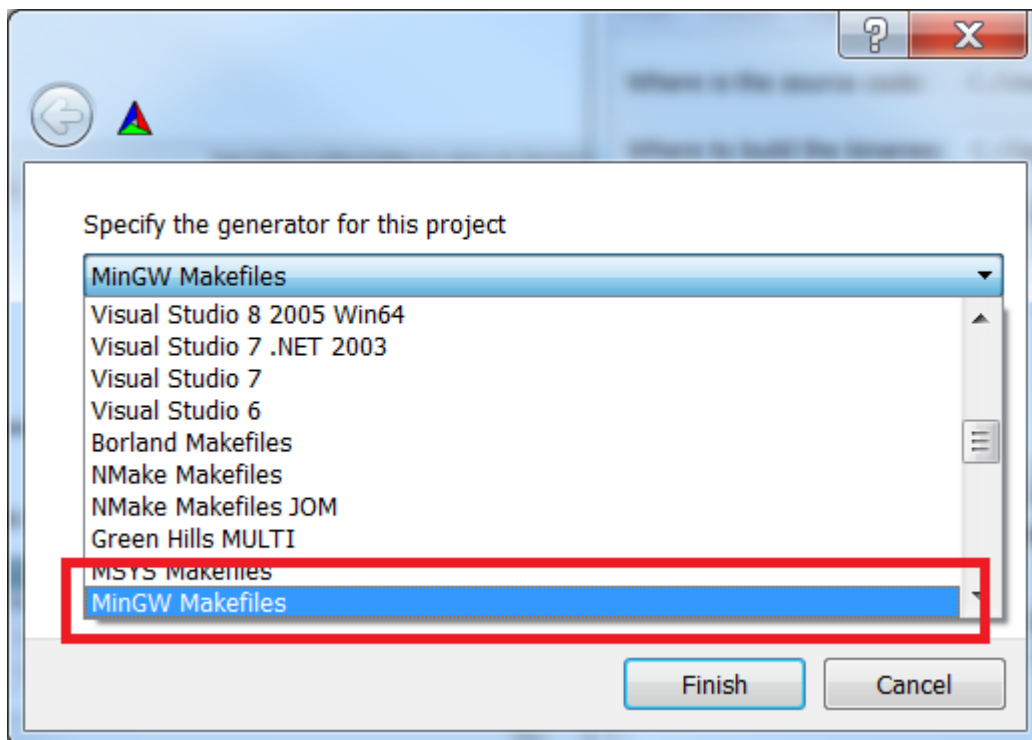


Figure 27. CMake Specify Generator

8. Select the “Use default native compilers” radio button and then select "Finish" as shown in [Figure 28](#). CMake will now configure the OpenCV files. It should configure with no errors.

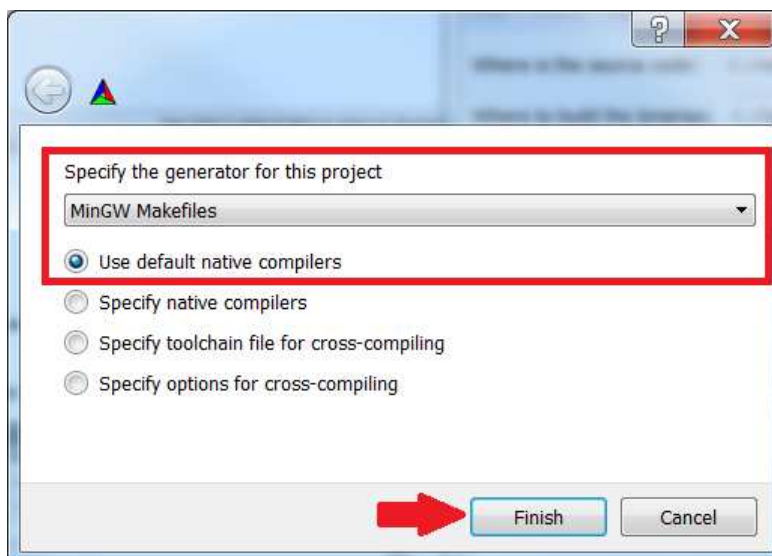


Figure 28. CMake Configure Generators

9. If an error occurs immediately, check the PATH variable and make sure that it includes the MinGW path from [Section 4](#). Also make sure the computer was restarted after that change. If the error persists, use the “Specify native compilers” option as shown in [Figure 29](#). Click “Next” to continue. [Figure 30](#) shows the compiler selection menu. Browse for both the C and C++ compilers in the Qt MinGW binary folder. For the C compiler, select gcc.exe and for the C++ compiler select the g++.exe as shown in [Figure 31](#). Once this is complete, the compilers window should look like [Figure 32](#). CMake will try configuring again.

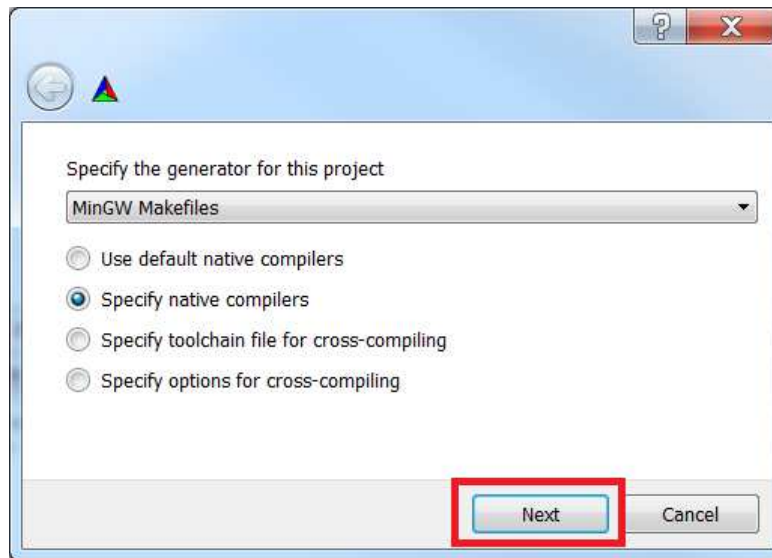


Figure 29. CMake Specify Compilers

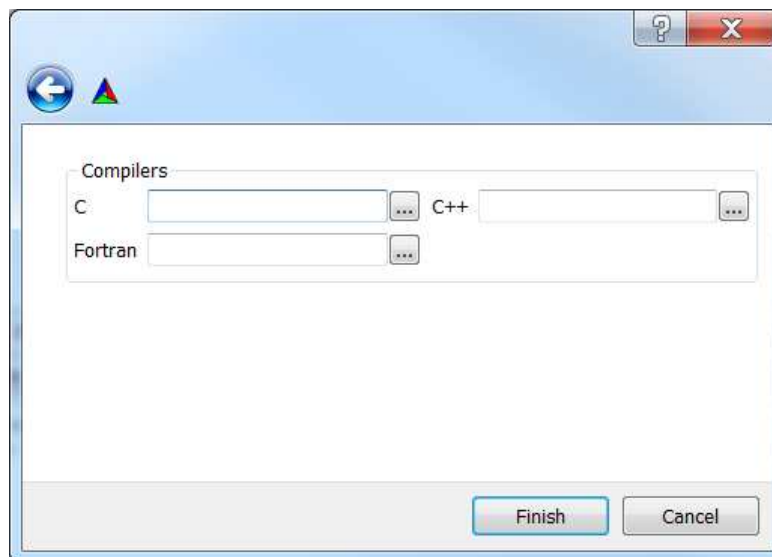


Figure 30. CMake Specify Compiler Menu

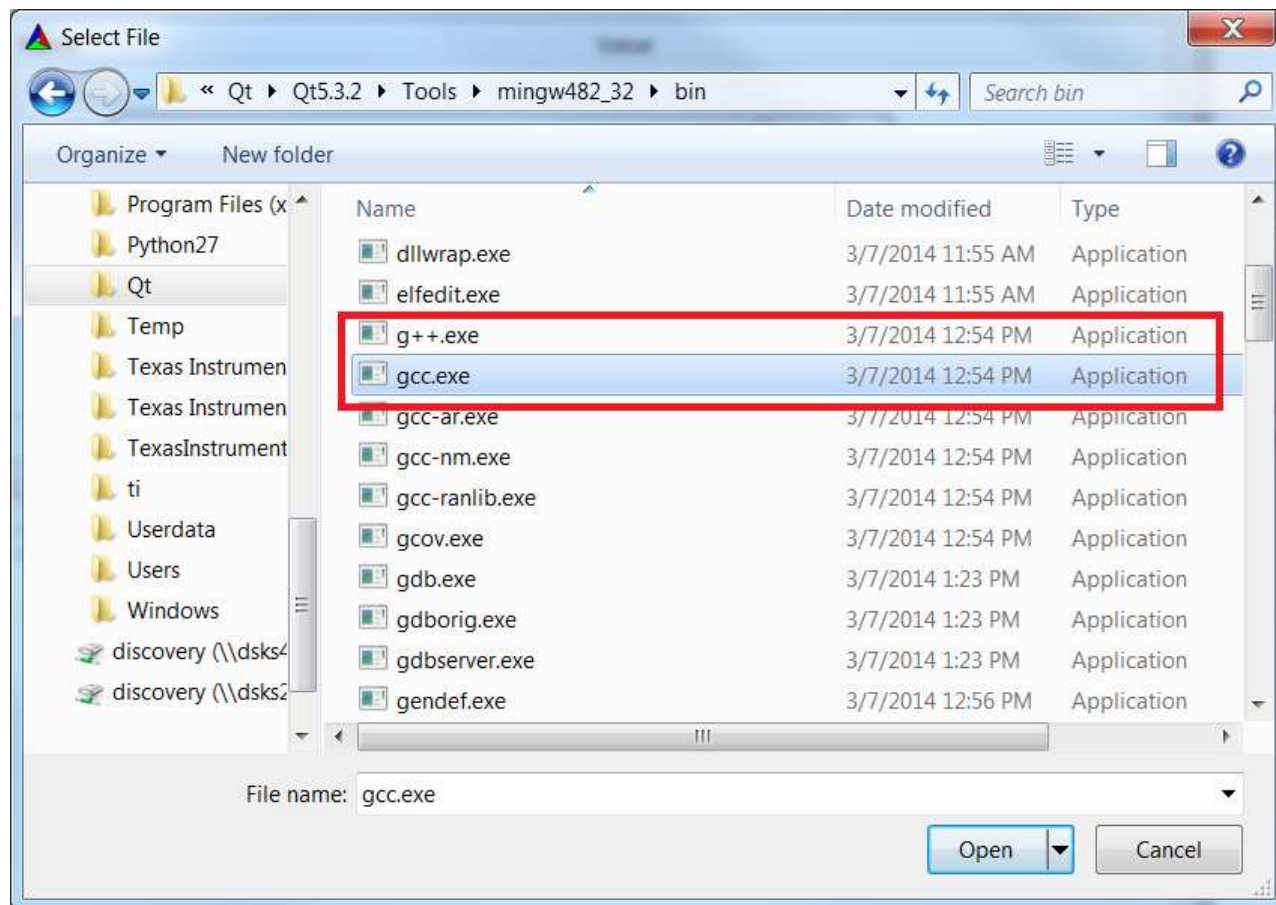


Figure 31. Browse for Compilers

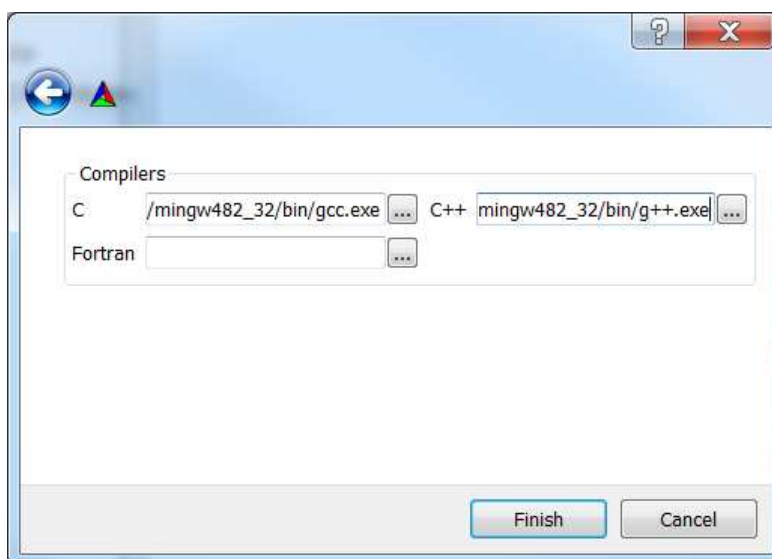


Figure 32. CMake Compilers Selected

10. Once the configuration is done, the CMake GUI should look like [Figure 33](#).

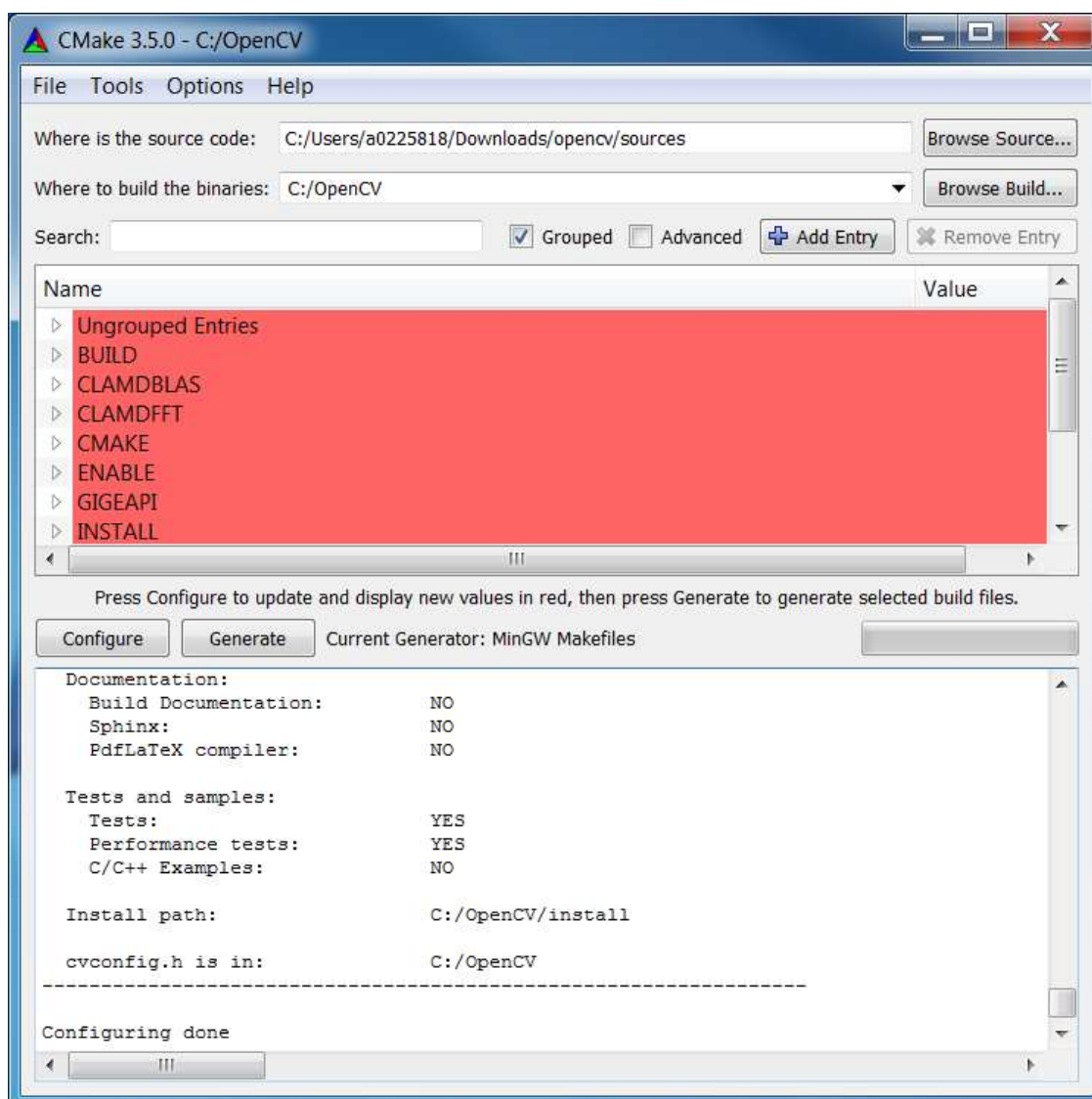


Figure 33. OpenCV Post Configuration

11. Find CMAKE_CONFIGURATION_TYPES as shown in Figure 34. Type "Release" into that space. Click on the "Generate" button. Allow CMake to run and finish generating the files. There should be no errors.

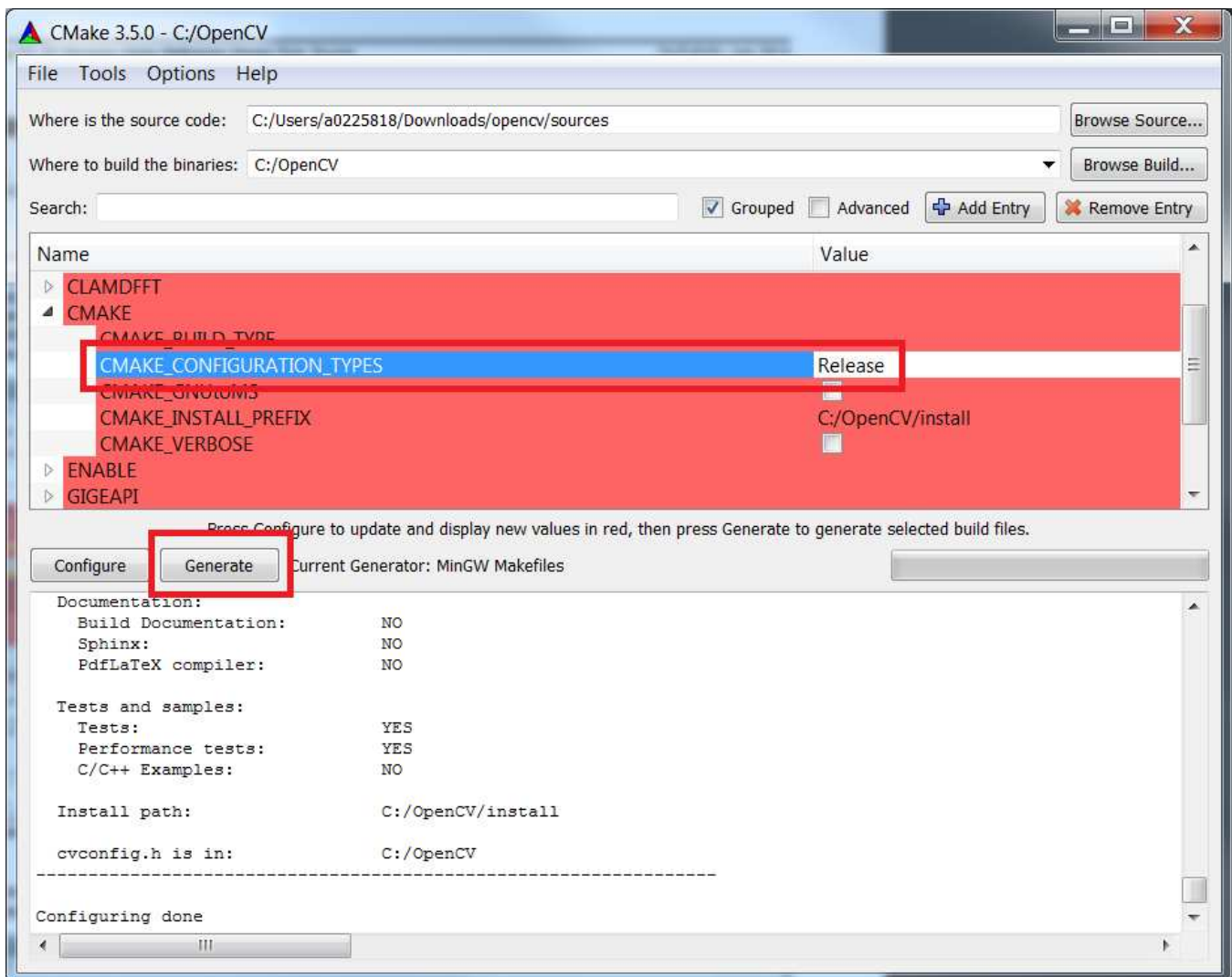


Figure 34. Change Configuration Type to Release

12. Next, in File Explorer, browse to where the OpenCV files were built. In the file path line, type "cmd" as shown in [Figure 35](#). This will open the command line window in the folder selected.

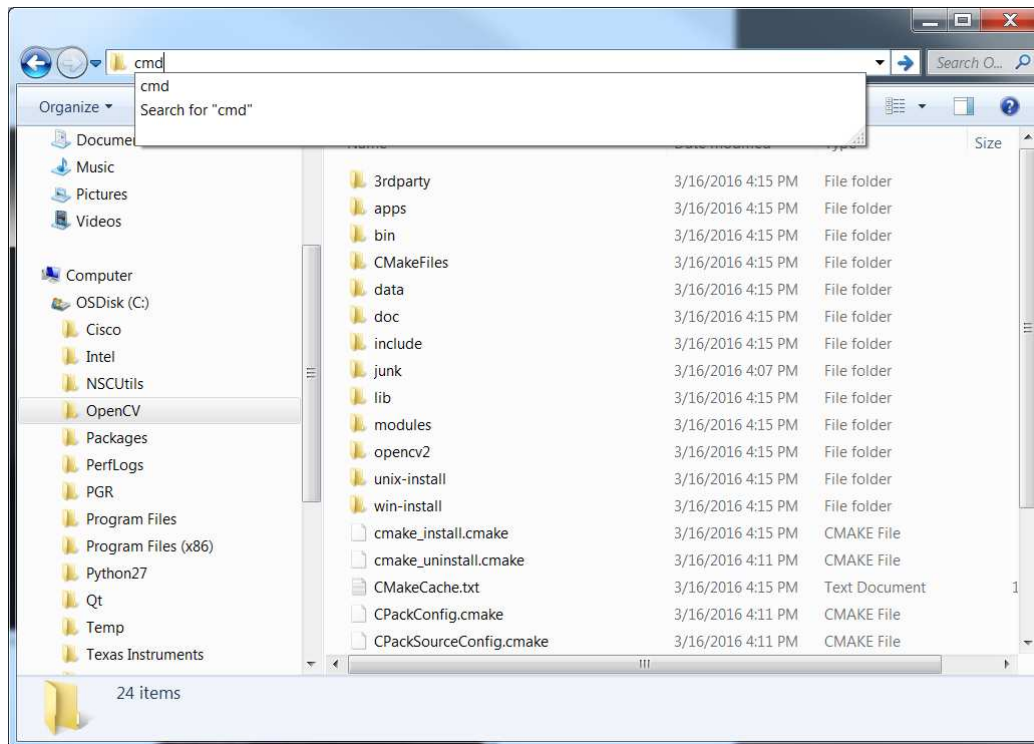


Figure 35. Open Command Line in OpenCV Window

13. As shown in [Figure 36](#), type the command "mingw32-make" and hit enter.

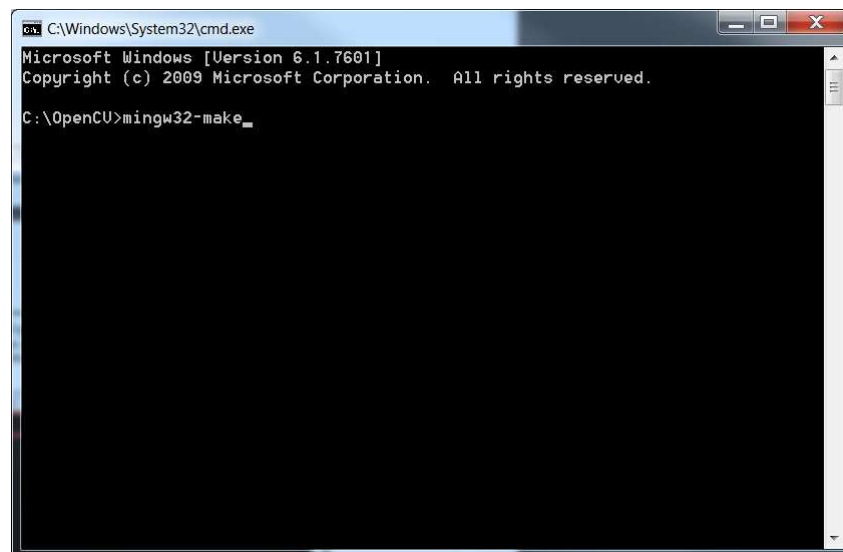
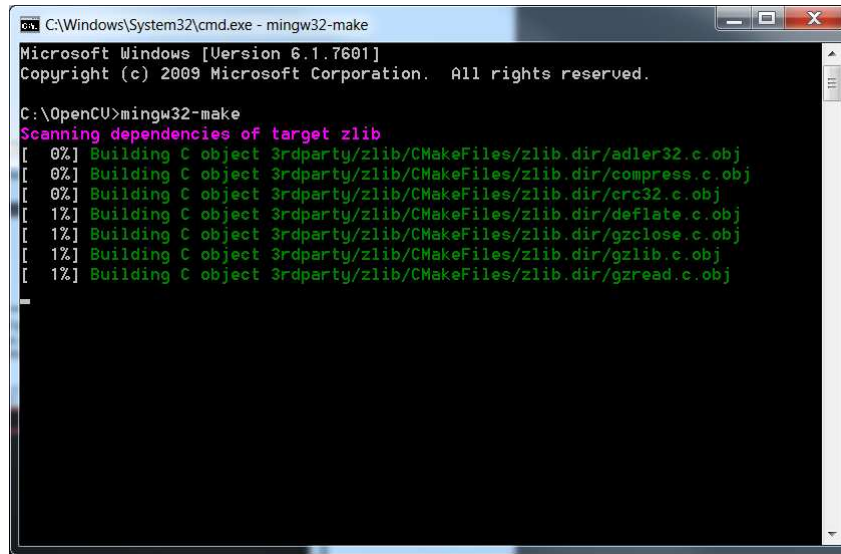


Figure 36. OpenCV CMD MinGW32-make

14. Allow the make command to run. This can take a substantial amount of time depending on your system's processing power. The command line will show progress as in [Figure 37](#).



```

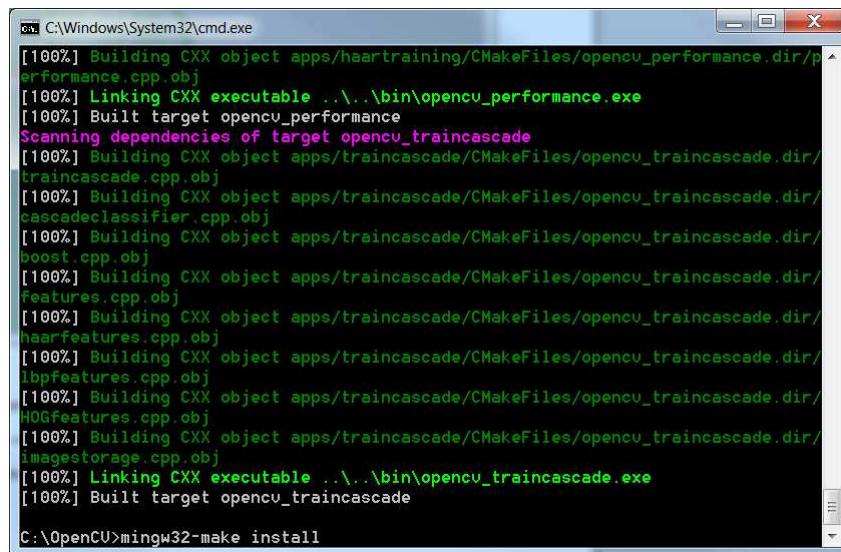
C:\Windows\System32\cmd.exe - mingw32-make
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\OpenCV>mingw32-make
Scanning dependencies of target zlib
[ 0%] Building C object 3rdparty/zlib/CMakeFiles/zlib.dir/adler32.c.obj
[ 0%] Building C object 3rdparty/zlib/CMakeFiles/zlib.dir/compress.c.obj
[ 0%] Building C object 3rdparty/zlib/CMakeFiles/zlib.dir/crc32.c.obj
[ 1%] Building C object 3rdparty/zlib/CMakeFiles/zlib.dir/deflate.c.obj
[ 1%] Building C object 3rdparty/zlib/CMakeFiles/zlib.dir/gzclose.c.obj
[ 1%] Building C object 3rdparty/zlib/CMakeFiles/zlib.dir/gzlib.c.obj
[ 1%] Building C object 3rdparty/zlib/CMakeFiles/zlib.dir/gzread.c.obj

```

Figure 37. OpenCV Making

15. Once this is complete, type the command "mingw32-make" install as shown in [Figure 38](#). This completes the installation of OpenCV.



```

C:\Windows\System32\cmd.exe
[100%] Building CXX object apps/haartesting/CMakeFiles/opencv_performance.dir/p
erformance.cpp.obj
[100%] Linking CXX executable ..\..\bin\opencv_performance.exe
[100%] Built target opencv_performance
Scanning dependencies of target opencv_traincascade
[100%] Building CXX object apps/traincascade/CMakeFiles/opencv_traincascade.dir/
traincascade.cpp.obj
[100%] Building CXX object apps/traincascade/CMakeFiles/opencv_traincascade.dir/
cascadeclassifier.cpp.obj
[100%] Building CXX object apps/traincascade/CMakeFiles/opencv_traincascade.dir/
boost.cpp.obj
[100%] Building CXX object apps/traincascade/CMakeFiles/opencv_traincascade.dir/
features.cpp.obj
[100%] Building CXX object apps/traincascade/CMakeFiles/opencv_traincascade.dir/
haarfeatures.cpp.obj
[100%] Building CXX object apps/traincascade/CMakeFiles/opencv_traincascade.dir/
lbpfeatures.cpp.obj
[100%] Building CXX object apps/traincascade/CMakeFiles/opencv_traincascade.dir/
HOGfeatures.cpp.obj
[100%] Building CXX object apps/traincascade/CMakeFiles/opencv_traincascade.dir/
imagestorage.cpp.obj
[100%] Linking CXX executable ..\..\bin\opencv_traincascade.exe
[100%] Built target opencv_traincascade
C:\OpenCV>mingw32-make install

```

Figure 38. OpenCV Make Install

7 Installing the DLP ALC SDK

This section guides the user through the installation of the DLP ALC SDK. This installer is a wrapper for the source code and copies the files onto the system in the location specified during the installation process. After running the installer, it is still required to build the source code using CMake.

1. Download the DLP ALC SDK from the link in [Section 1](#).
2. Run the executable *DLPSDK-2.0-windows-installer.exe*.
3. The installer will open and present a set-up screen. Click "Next" as shown in [Figure 39](#).

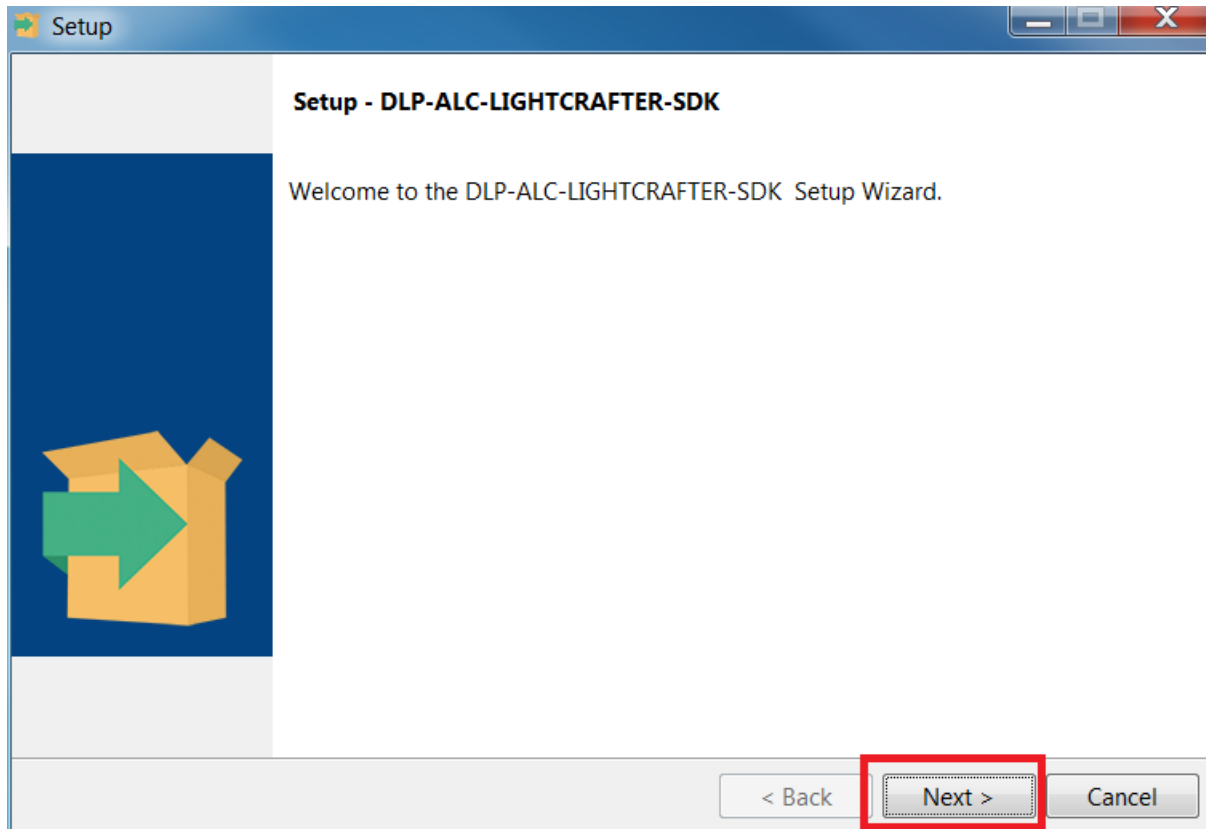


Figure 39. DLP ALC SDK Installer Setup

4. Accept the license agreement shown in [Figure 40](#) and click "Next."

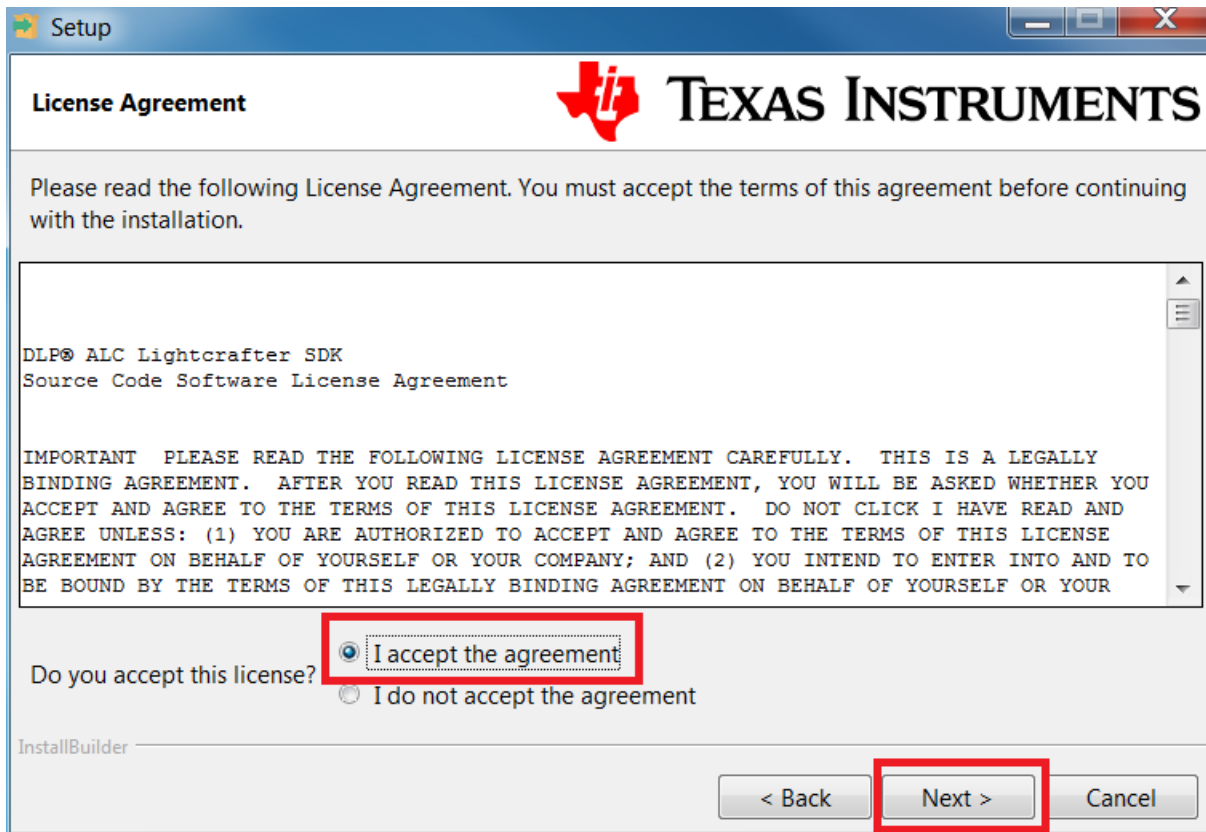


Figure 40. DLP ALC SDK Installer License

5. Select the directory where the files should be stored. The default directory is shown in [Figure 41](#).

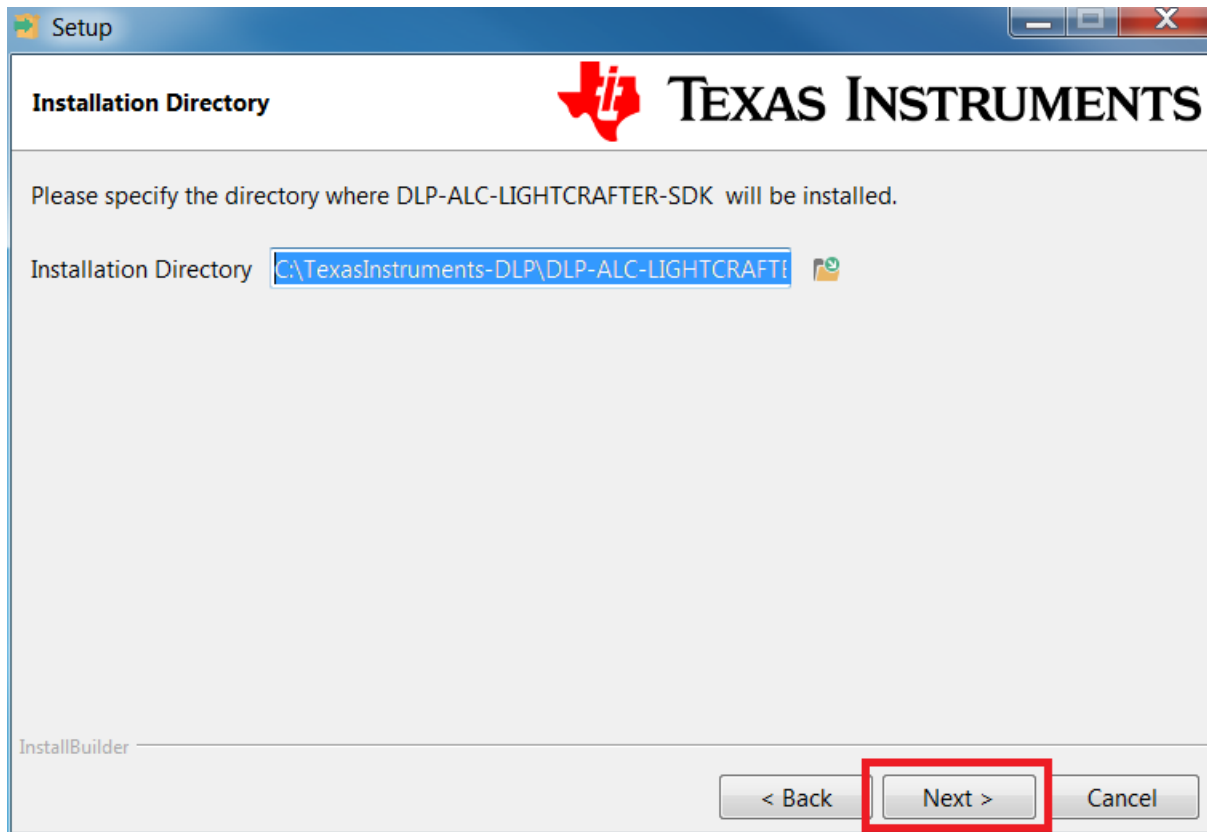


Figure 41. DLP ALC SDK Install Directory

6. The installer is now ready to copy the source files. Select "next" as shown in [Figure 42](#).

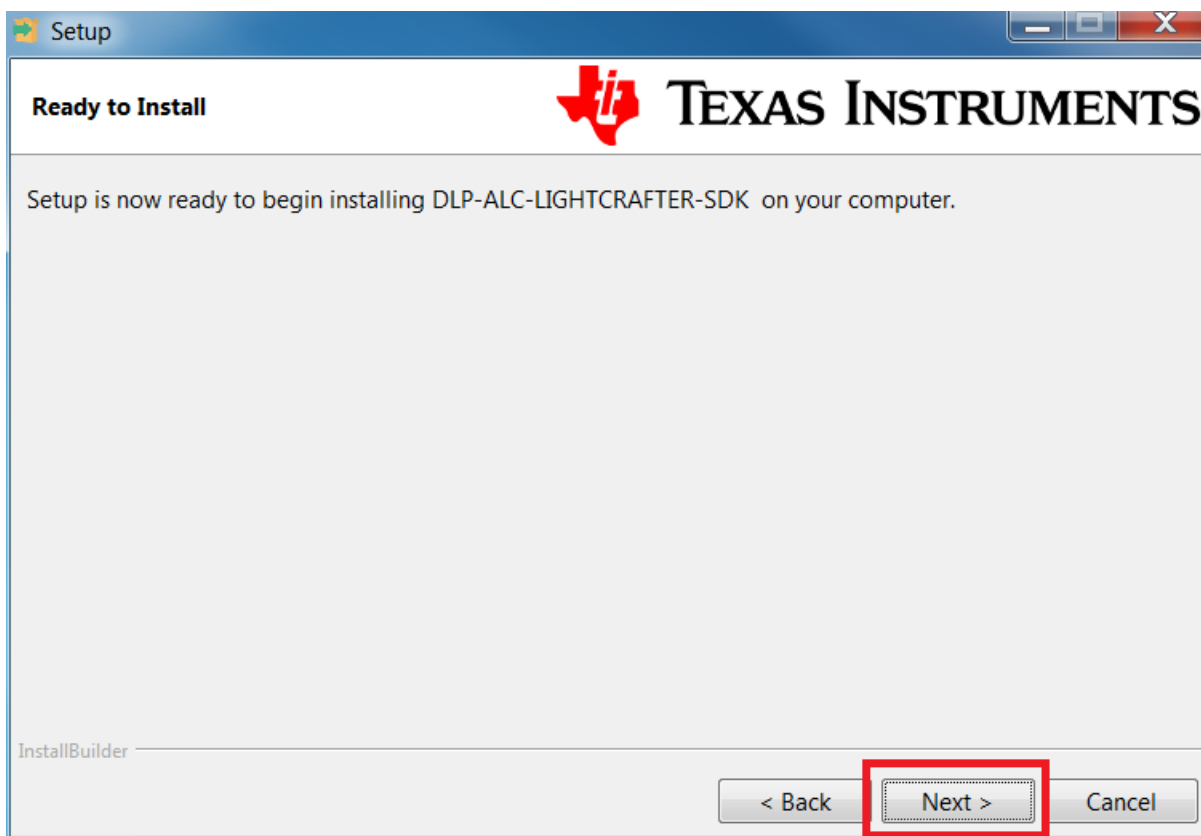


Figure 42. DLP ALC SDK Installer Ready

7. The installer will be copying the files and will display a progress bar as in Figure 43.

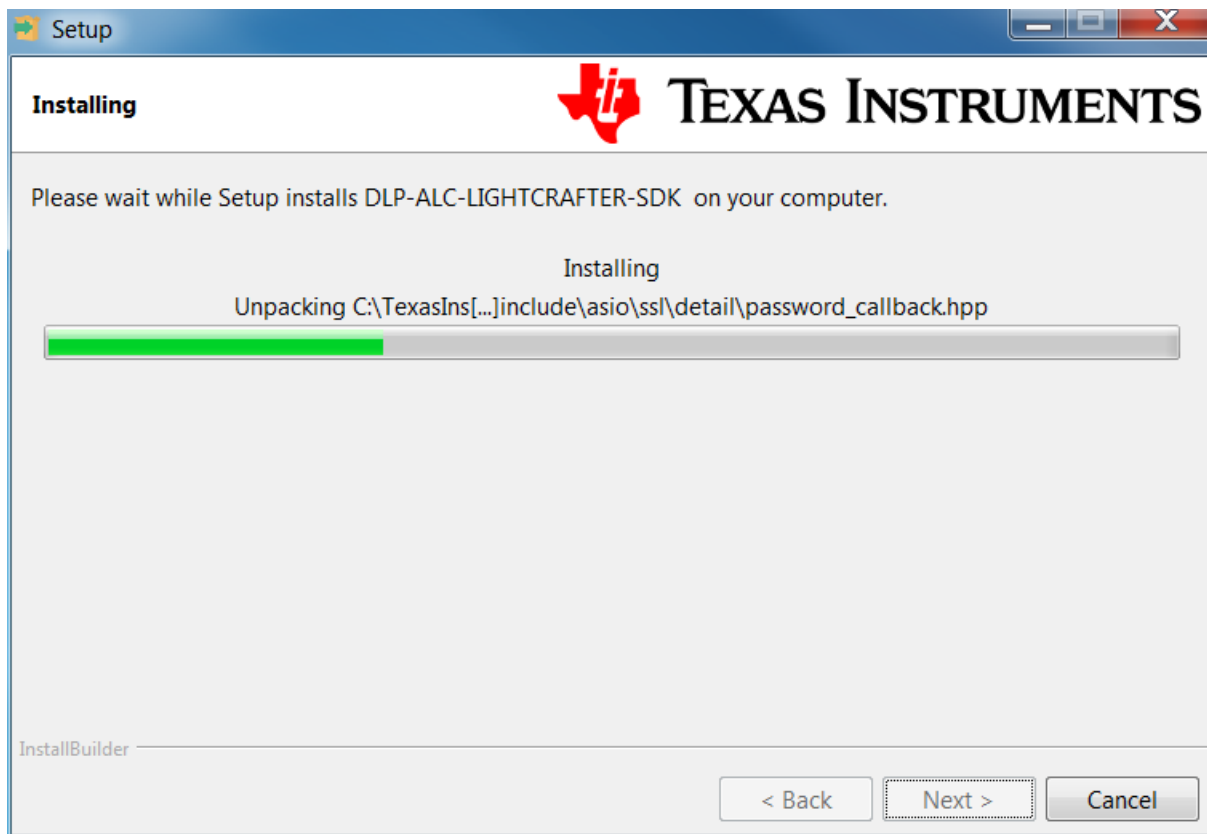


Figure 43. DLP ALC SDK Installer Progress Bar

8. The installer will finish and look like [Figure 44](#). Click "Finish" to exit the installer.

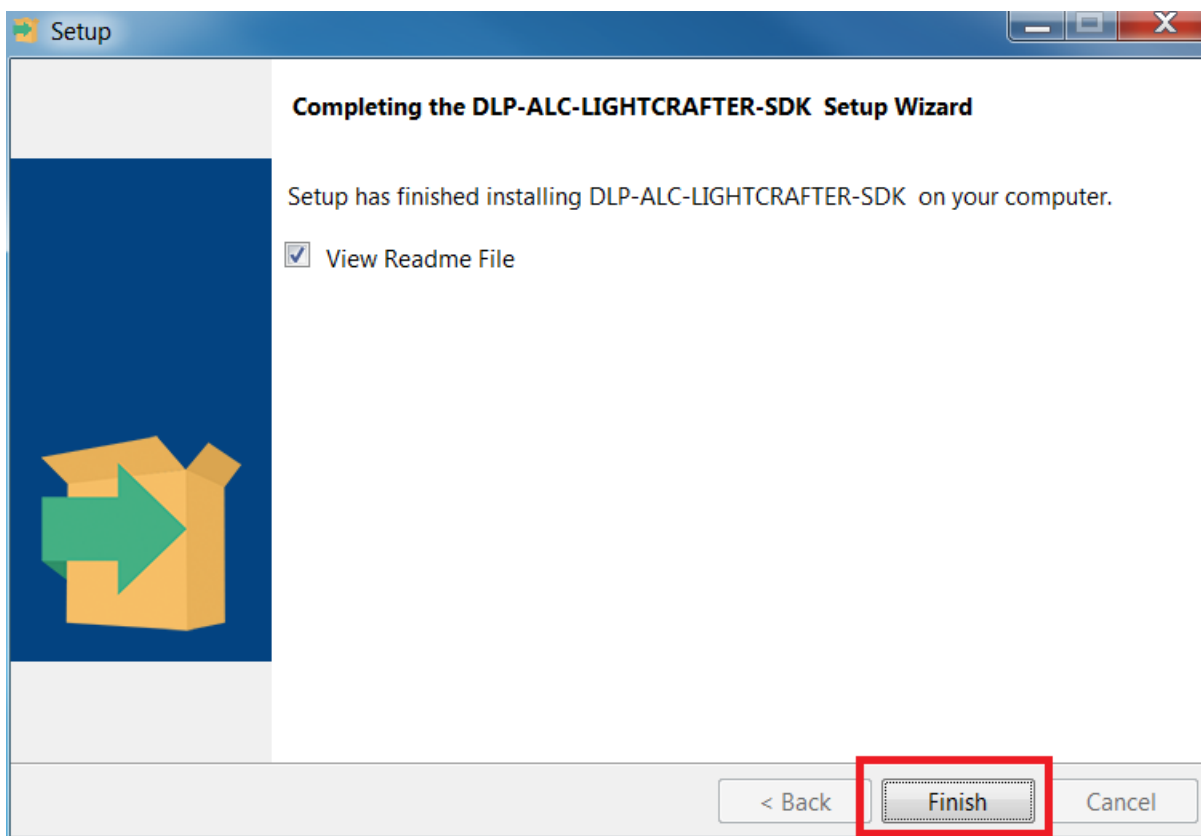


Figure 44. DLP ALC SDK Installer Finished

8 Building the DLP ALC SDK

This section guides the user on building the DLP ALC SDK for 3D scanning.

1. Once the installation from [Section 7](#) is complete, open the CMake GUI as in [Section 6](#).
2. Browse for the source files of the DLP_structured_light_SDK by clicking “Browse Source”. An example file path: C:\TexasInstruments-DLP\DLP-ALC-LIGHTCRAFTER-SDK-2.0\DLP-ALC-LIGHTCRAFTER-SDK.
3. Next, click “Browse Build.” Create a new folder for the built DLP ALC SDK. [Figure 45](#) shows what the CMake GUI should look like. An example file path with a new folder for the built files would be: C:/TexasInstruments-DLP/DLP-ALC-LIGHTCRAFTER-SDK-2.0_bld.

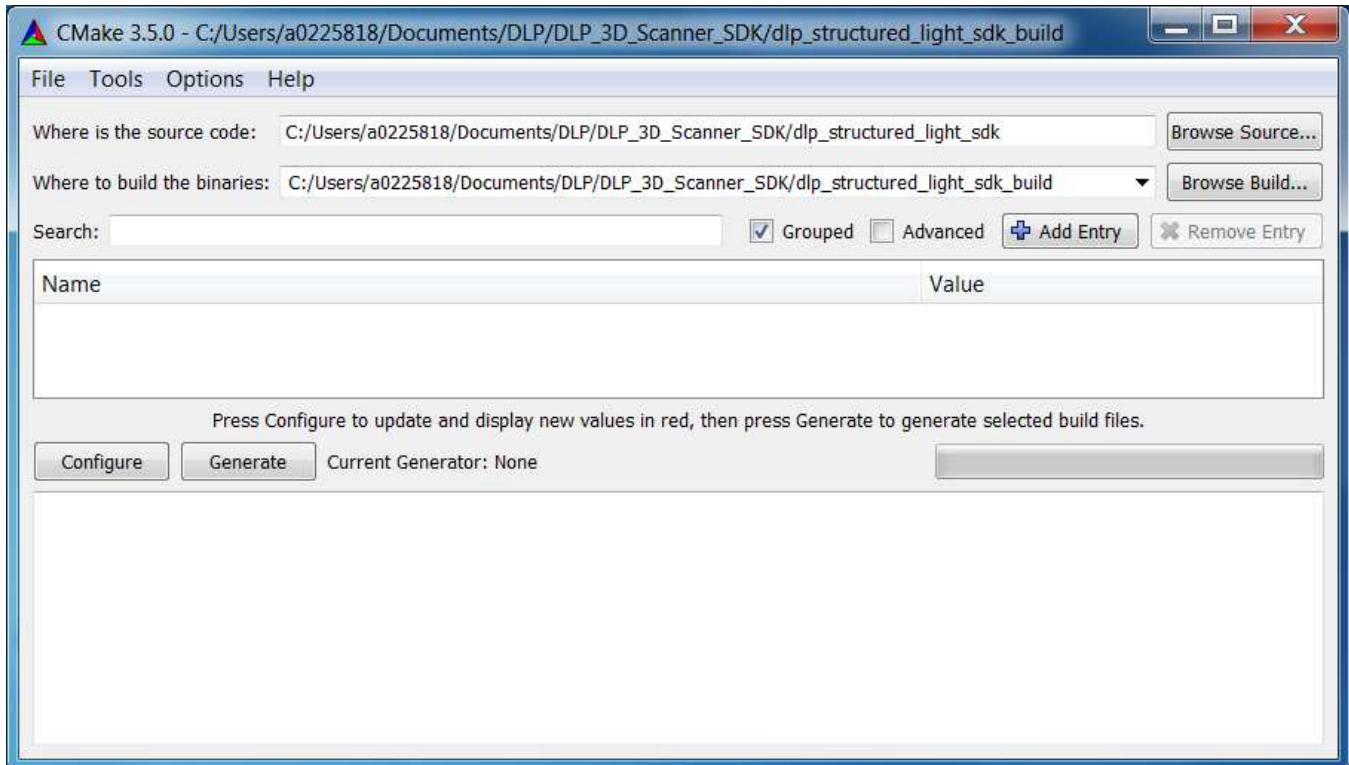


Figure 45. DLP ALC SDK File Paths

4. Click "Configure." Use MinGW Makefiles and default native compilers as in [Section 6](#). Allow CMake to run.

Note: If you get an error as shown in [Figure 46](#), you may need to point CMake to the location of OpenCV. In "Ungrouped entries" look for OpenCV_DIR and browse for the location of OpenCV on your machine. As shown in [Figure 47](#) an example is: C:/OpenCV. Click "Configure" once the path has been specified. There should be no errors displayed.

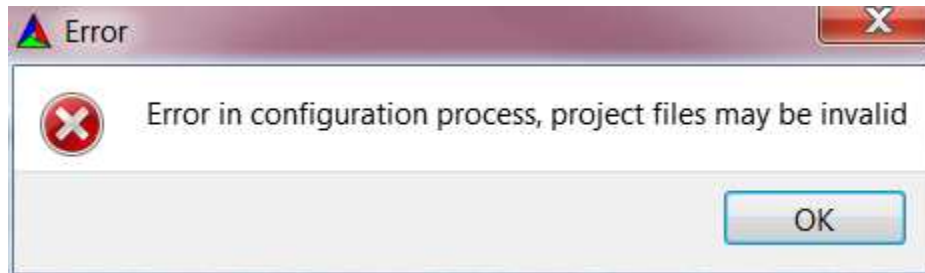


Figure 46. DLP SDK Error

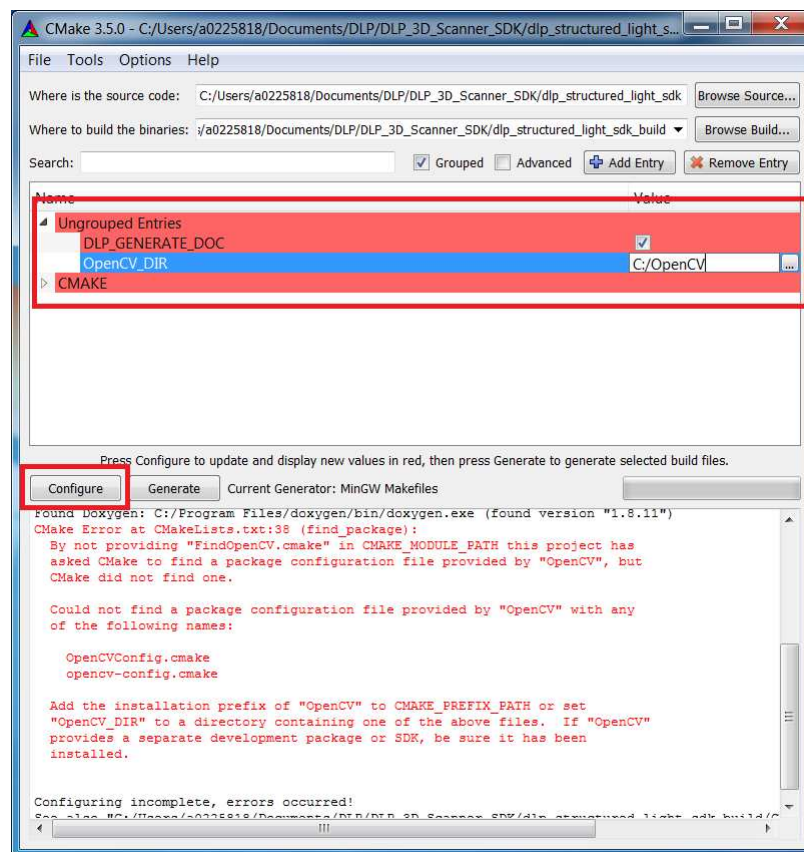


Figure 47. Browse for OpenCV

5. Once configuration is complete, click "Generate." This should be quick and have no errors.
6. As shown in [Figure 48](#) and [Figure 49](#), browse to the location of the built DLP ALC SDK files and type "cmd" in the file path and hit enter.

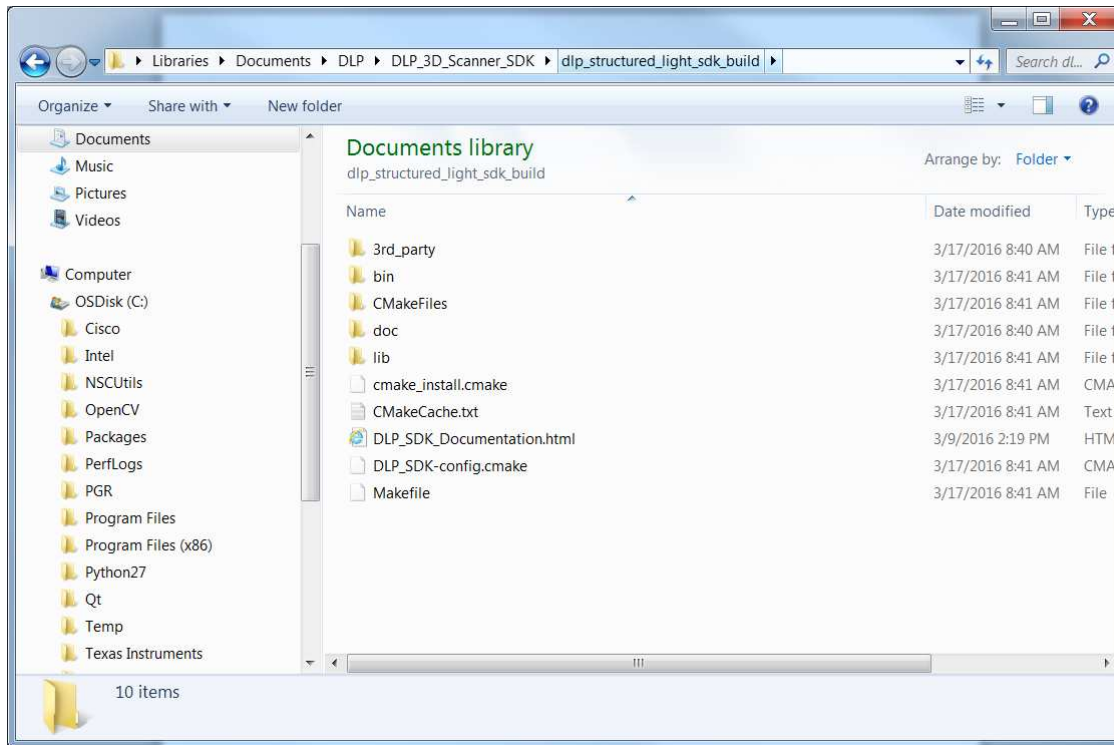


Figure 48. Browse for DLP ALC SDK Built Files

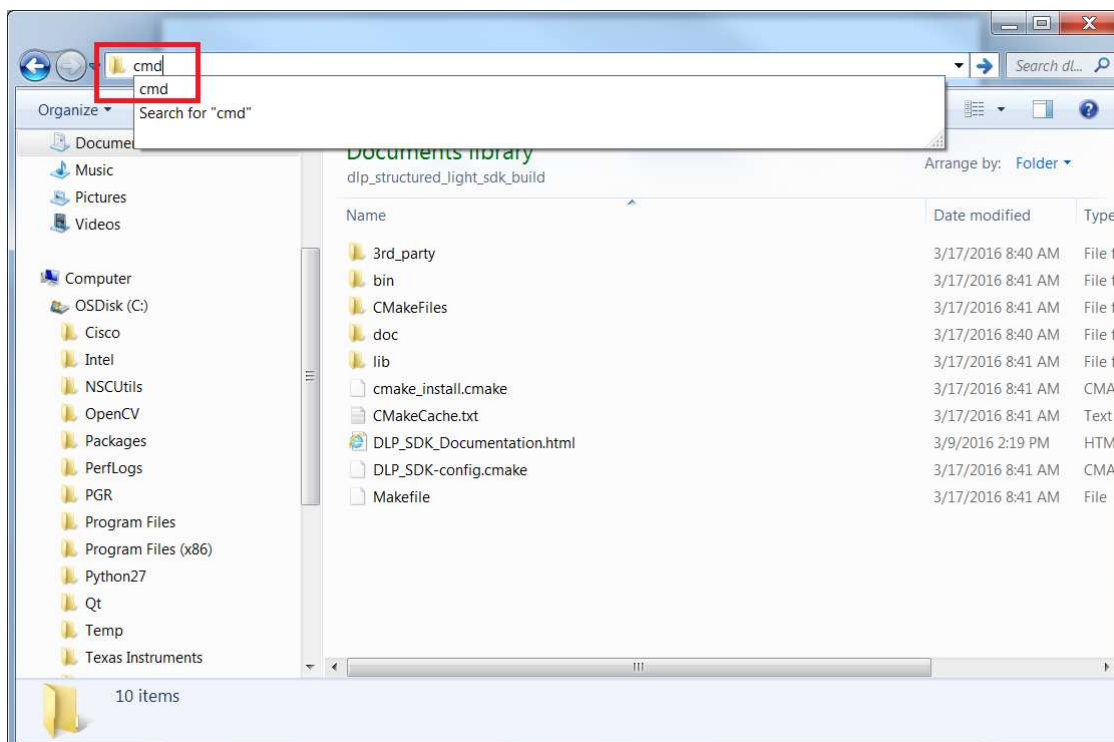


Figure 49. DLP ALC SDK CMD Line

7. Type the command "mingw32-make" as shown in [Figure 50](#).

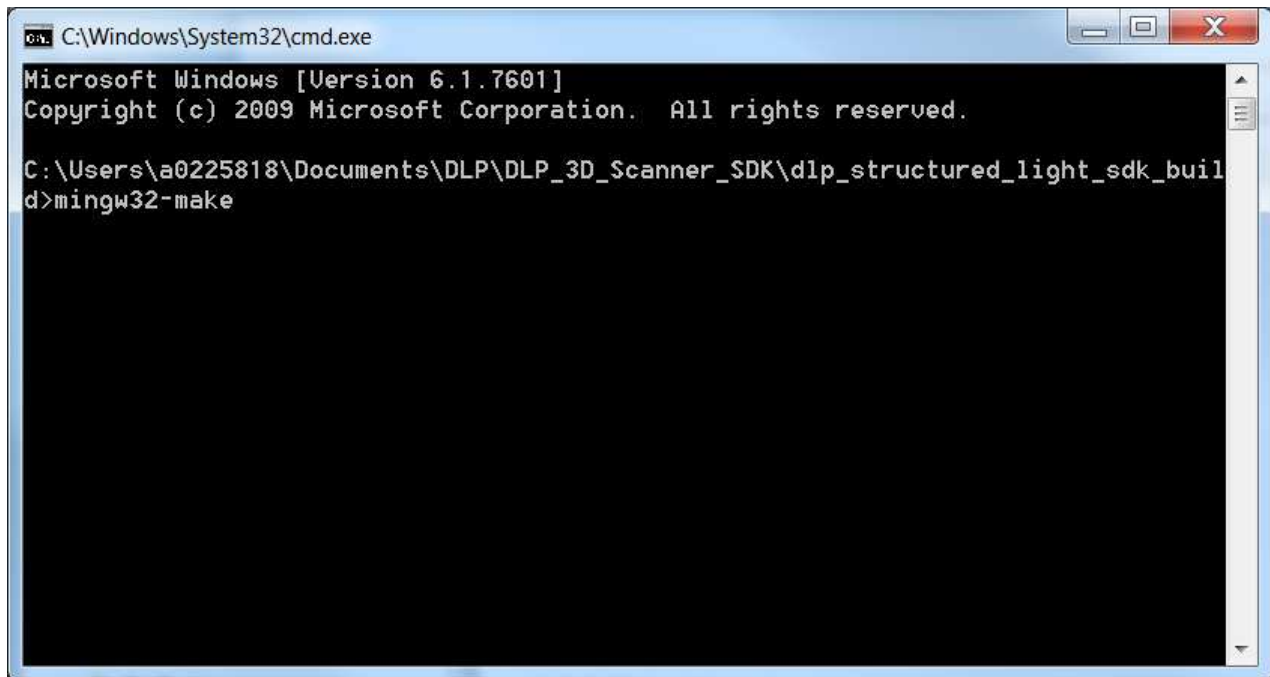


Figure 50. DLP ALC SDK Make

8. Let the command complete. The DLP ALC SDK is now built on the system.
9. To build the documentation for the DLP ALC SDK, in the same command line as [Figure 50](#), type the command "mingw32-make doc". This will utilize Doxygen to create easy to use HTML documentation. Click on the HTML file indicated in [Figure 51](#) to open the top level documentation file (an example path is C:\TexasInstruments-DLP\DLP-ALC-LIGHTCRAFTER-SDK-2.0_bld). [Figure 52](#) shows the welcome page for the compiled documentation.

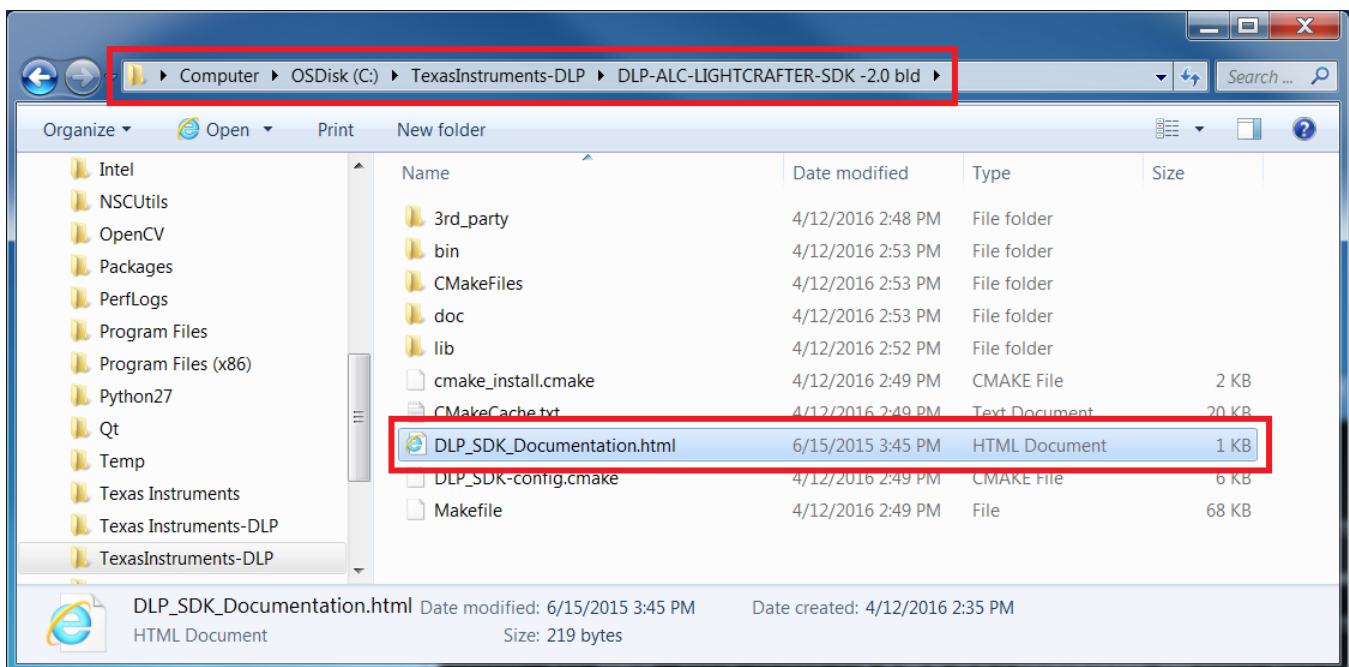


Figure 51. DLP ALC SDK Documentation Location

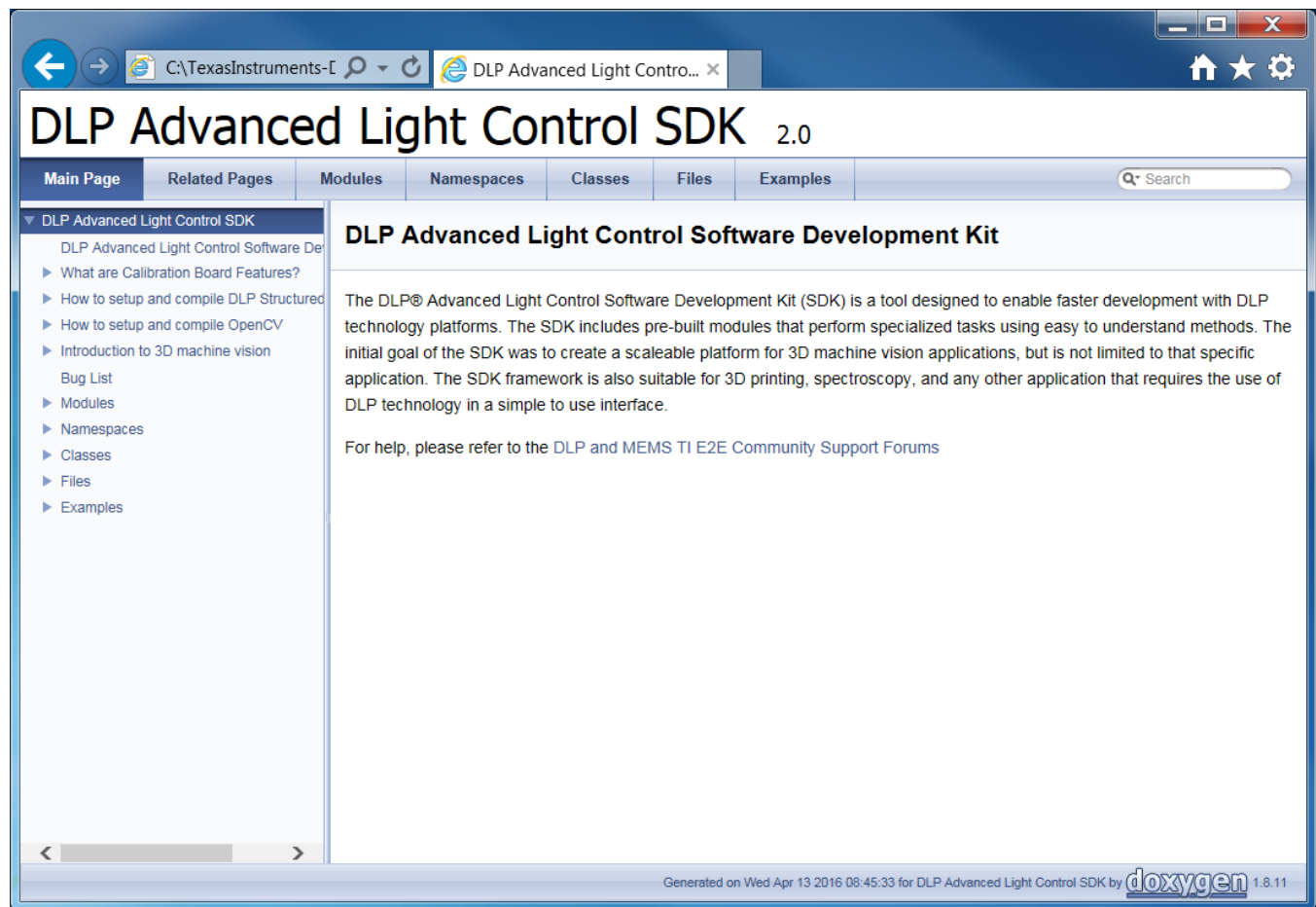


Figure 52. DLP ALC SDK Documentation Home Page

10. The installed and built DLP ALC SDK also includes example code for reference. This example code can be found in \bin folder of the built code. An example file path might be: C:\TexasInstruments-DLP\DLP-ALC-LIGHTCRAFTER-SDK-2.0_bld\bin. Once built, the examples are in the form of .exe files and include simple operations such as point cloud viewing, taking a picture, and opening the camera. To run these examples, three DLLs need to be copied into the \bin folder. These DLLs are *libgcc_s_dw2-1.dll*, *libstdc++-6.dll*, and *libwinpthread-1.dll*, and they can be found in C:\Qt\Qt5.3.2\Tools\mingw482_32\bin. Copy them from the folder shown in [Figure 53](#) into the build folder as shown in [Figure 54](#).

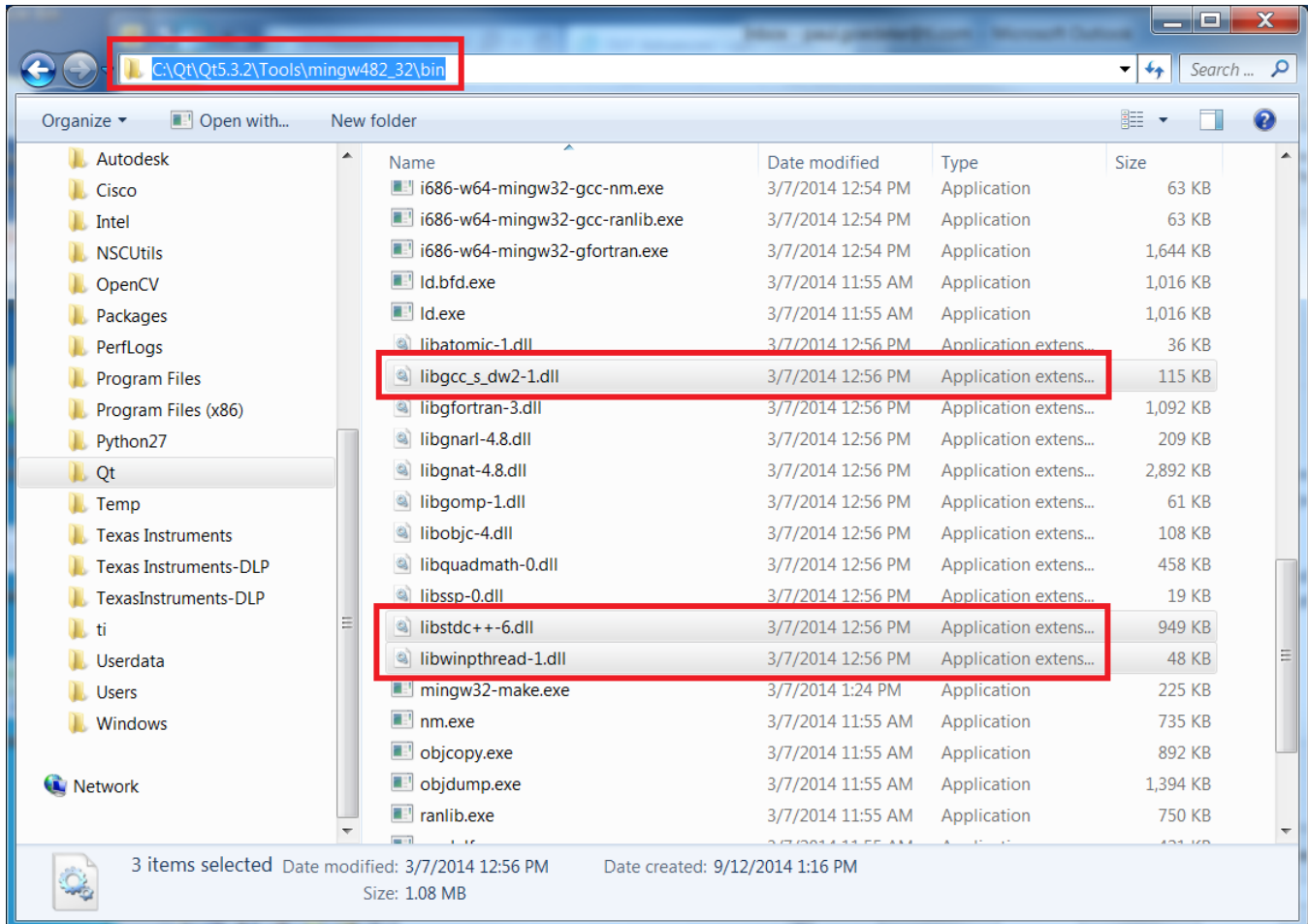


Figure 53. Example Code DLLs to Copy

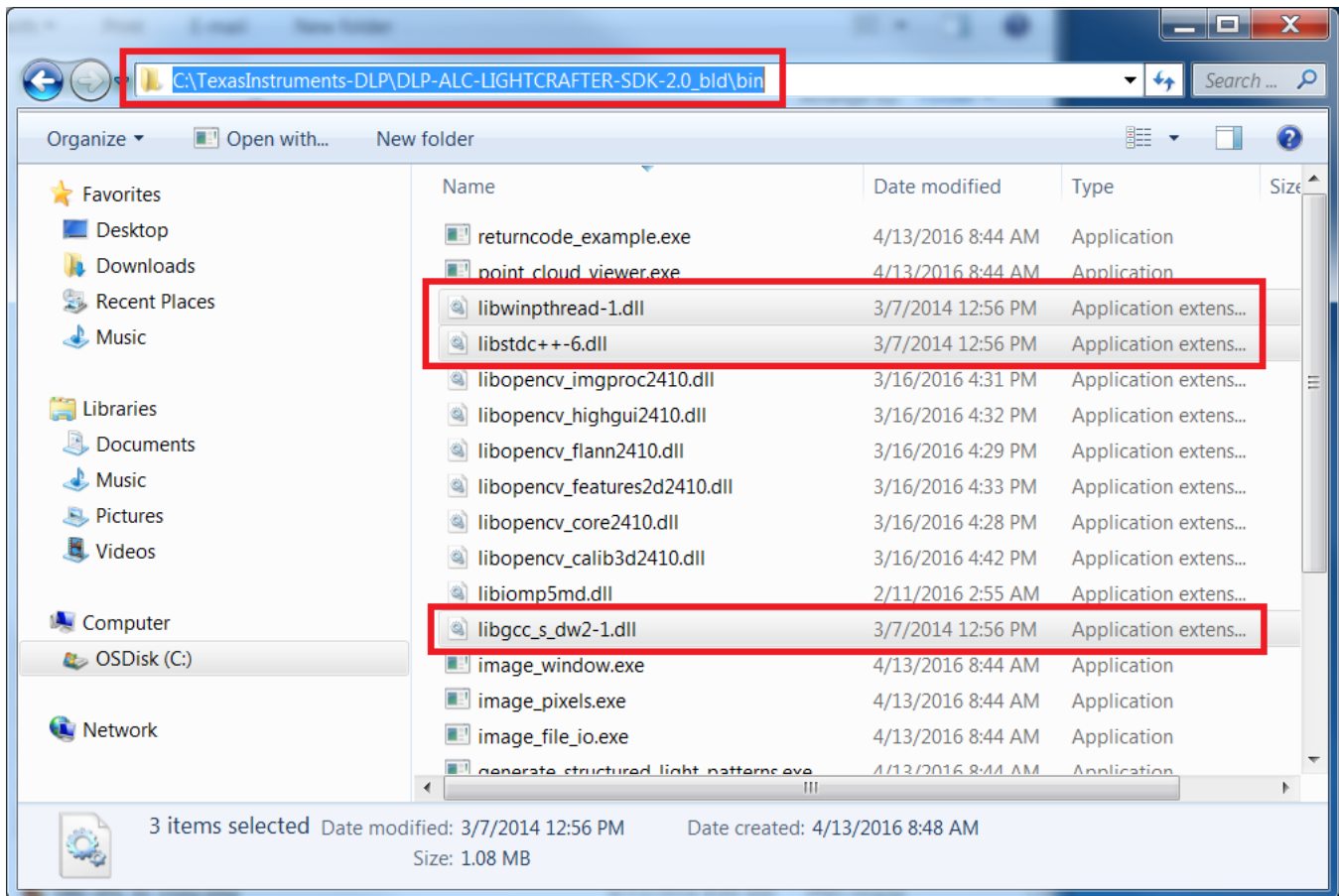


Figure 54. Example Code DLLs Copied Into Correct Folder

9 Compiling the 3D Machine Vision Reference Designs from Source

After the DLP ALC SDK has been compiled and installed, the 3D Machine Vision Reference Design source code can be built for further development activity beyond the binaries provided in the TI Designs. Users can modify the scanning applications to fit their own needs. As with the DLP ALC SDK, the design source code can be configured and compiled with CMake.

1. Open the start menu. Search for and run CMake as shown in [Figure 55](#).

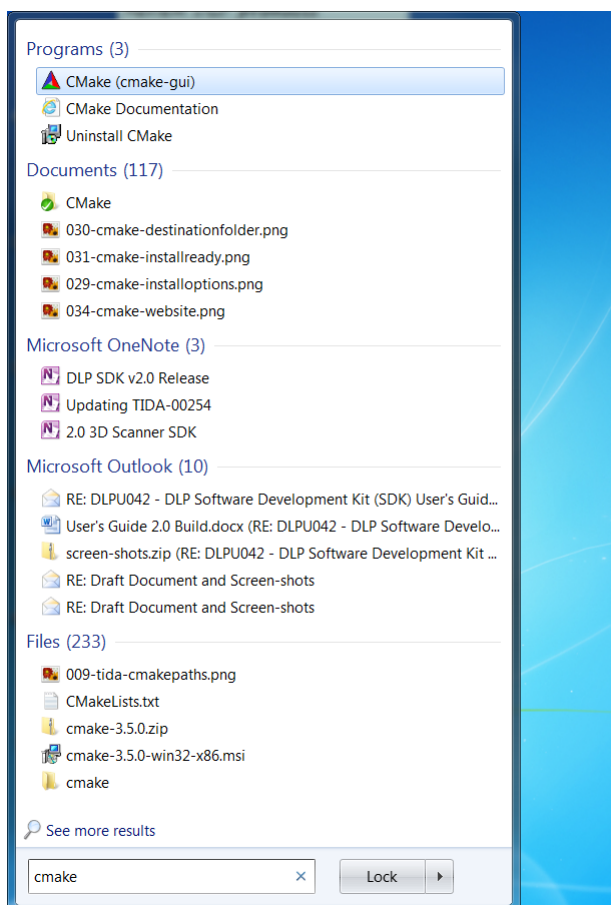


Figure 55. Opening CMake from Start Menu

2. Once CMake is open, browse for the reference design source code and create a new folder for the built source code. The CMake interface should look like [Figure 56](#).

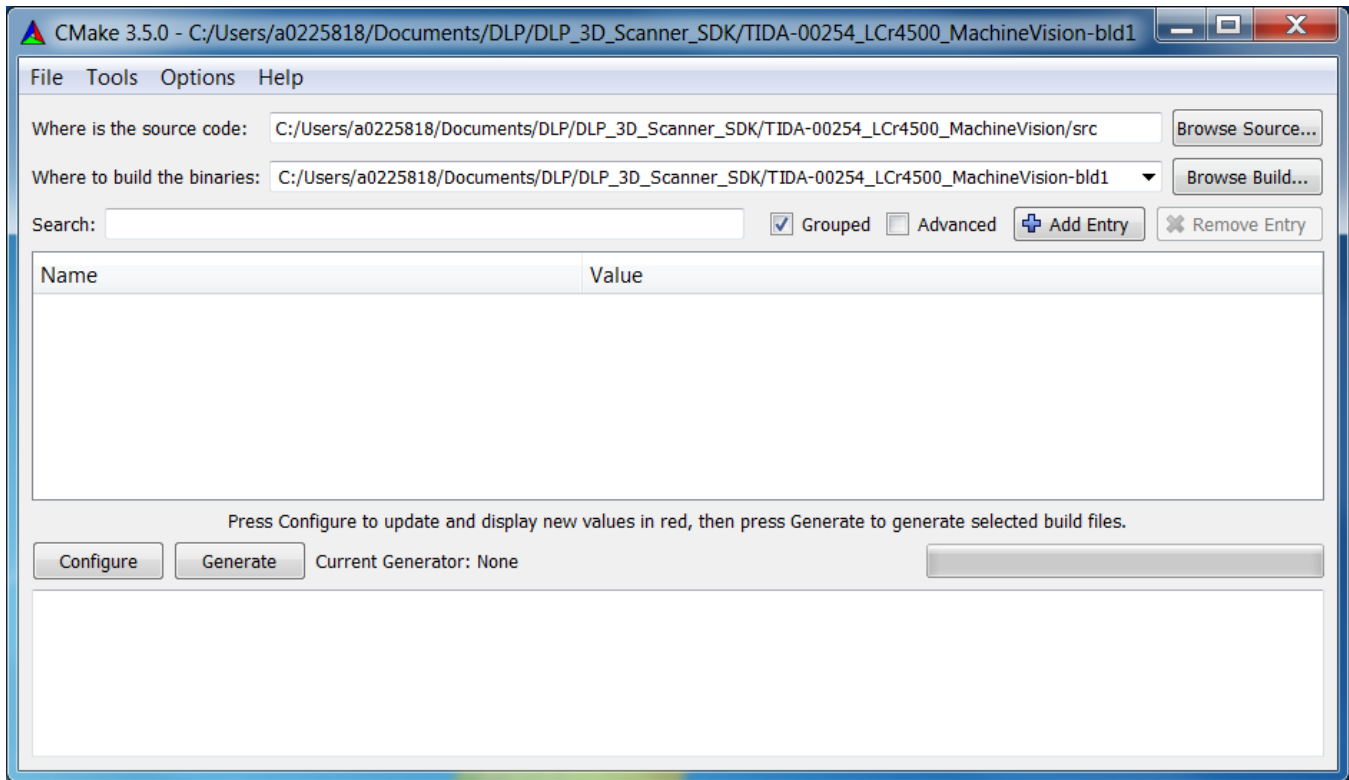


Figure 56. CMake Source and Build File Paths

- Next, click the "Configure" button and make sure that "MinGW Makefiles" is specified as the generator and that the "Use default native compilers" radio button is selected as shown in [Figure 57](#). Click "Finish" to initiate the configuration process.

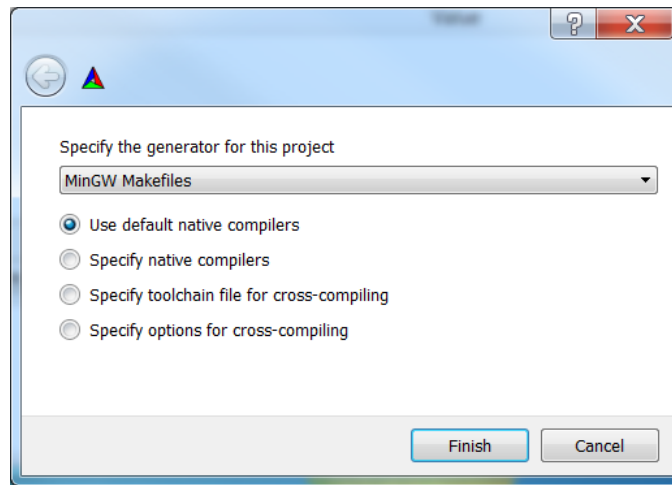


Figure 57. CMake Configuration Options

- There should be no errors upon the completion of this process.
Note: If errors occur, you may have to point CMake to the location OpenCV and the DLP SDK. [Figure 58](#) shows browsing for the path of the DLP SDK. Click "Configure" to continue. [Figure 59](#) shows browsing for OpenCV. Click "Configure" again and there should be no further errors. [Figure 60](#) indicates what CMake will look like once configuration is complete.

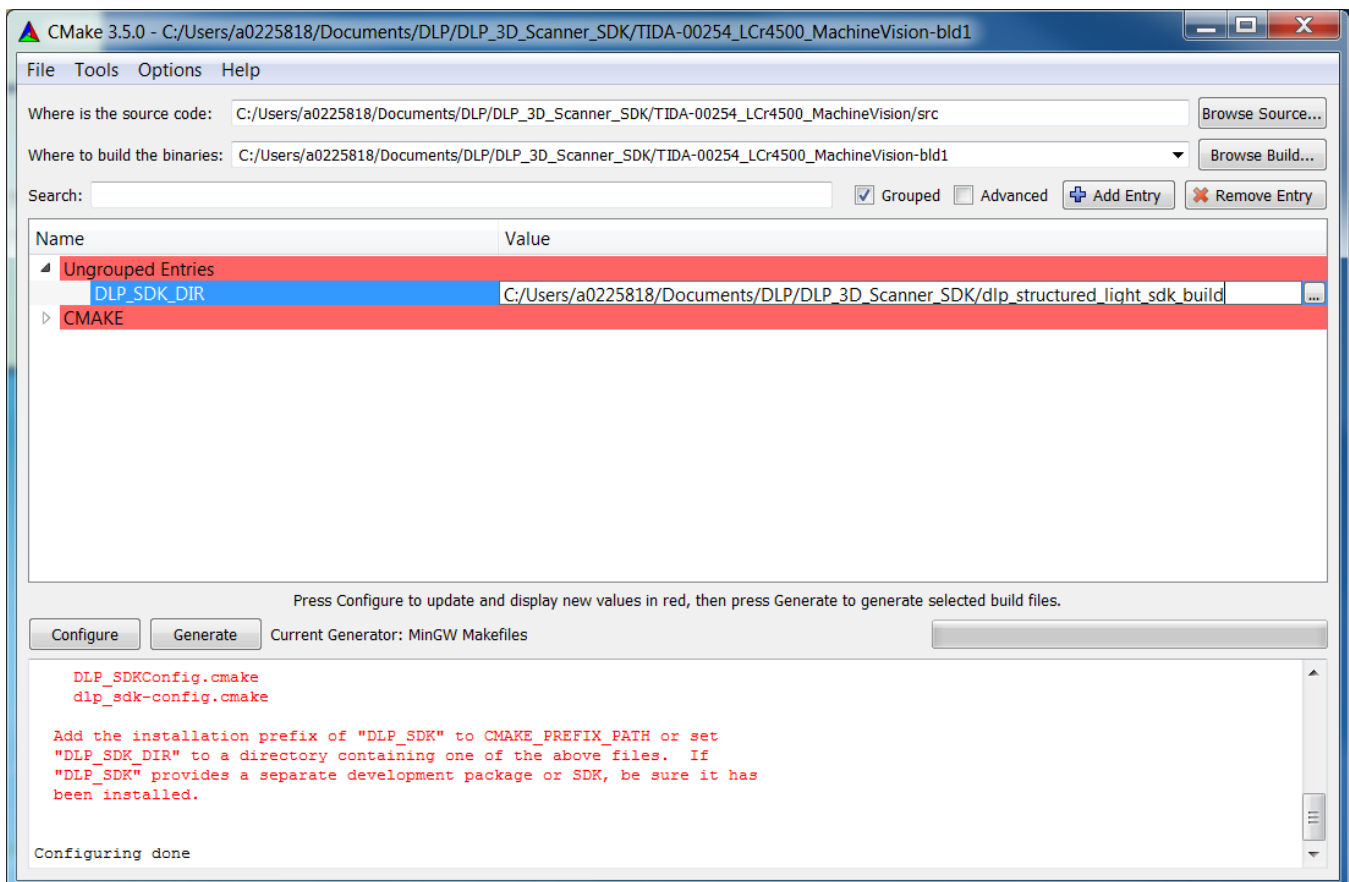


Figure 58. Browsing for DLP SDK

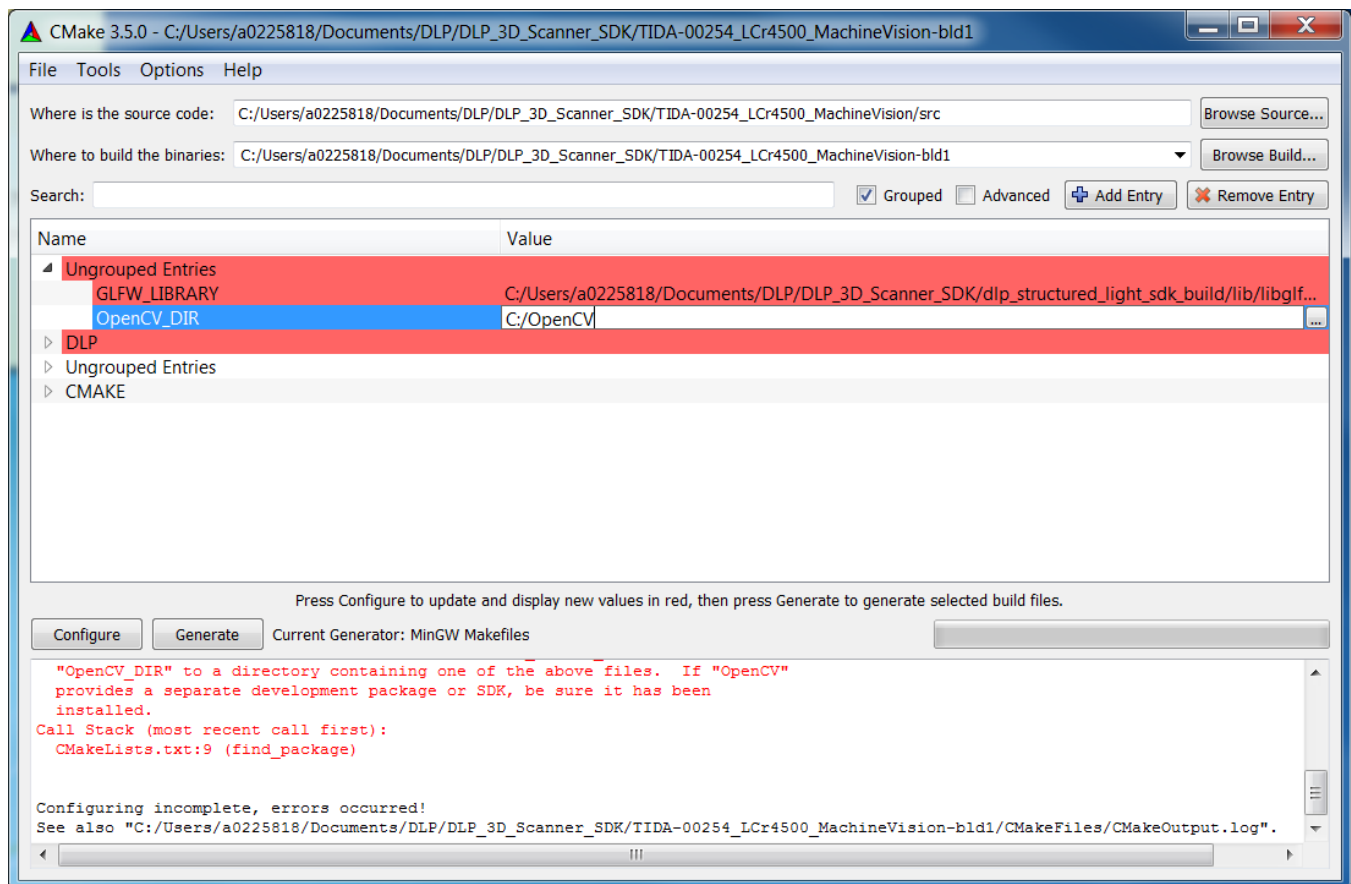


Figure 59. Browsing for OpenCV in CMake

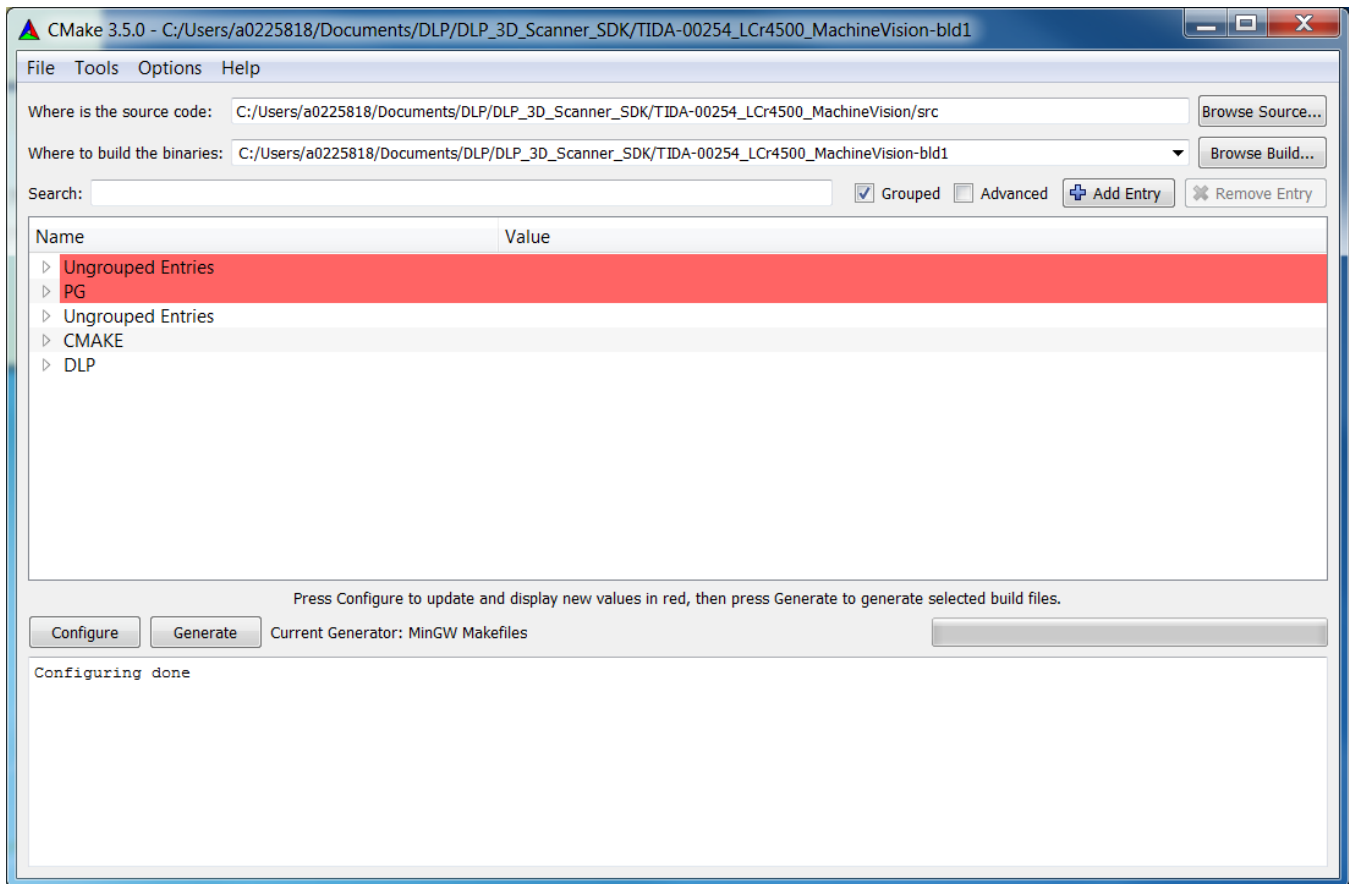


Figure 60. CMake Configure Complete

5. Once the configuration is complete, click "Generate."
6. Open a command line window in the build folder and run the command "mingw32-make" as shown in [Figure 61](#). This will create the executable which can be found in the binary folder as seen in [Figure 62](#).

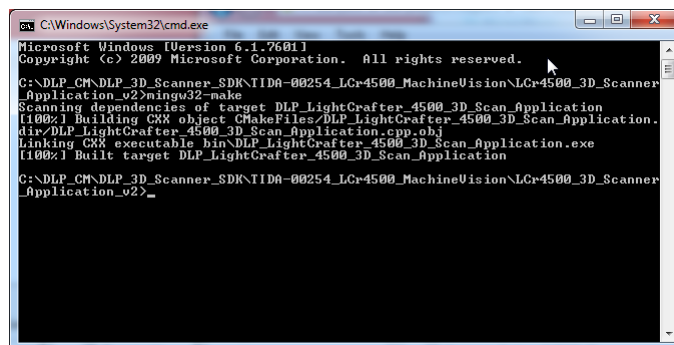


Figure 61. Creating the Executable

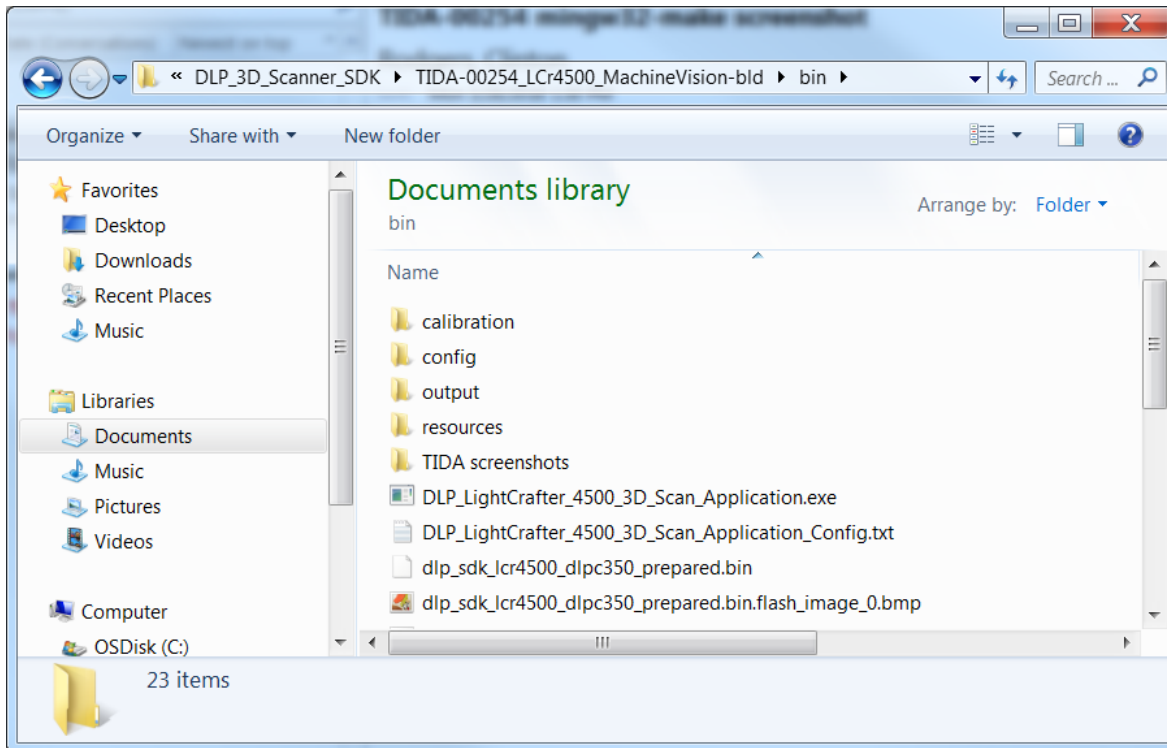


Figure 62. Executable Location

7. Before the TI Design can be run, the DLLs from [Figure 53](#) must be copied into the \bin folder inside the built code folder for the TI Design. An example path is C:\TexasInstruments-DLP\DLP-ALC-LIGHTCRAFTER-SDK-2.0\TIDA-00254_LCr4500_MachineVision_bld\bin.
8. Once the DLLs have been copied, the 3D Scanner program may now be run from the build folder.

10 Conclusion

The required software and source code is now installed and built to properly use the DLP ALC SDK. An example of building the source code for a TI Design has been demonstrated. For specifics on running [TIDA-00254](#), [TIDA-00362](#), and [TIDA-00361](#), please see the user's guides for those designs.

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES "AS IS" AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you fully indemnify TI and its representatives against any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI's products are provided subject to [TI's Terms of Sale](#), [TI's General Quality Guidelines](#), or other applicable terms available either on [ti.com](#) or provided in conjunction with such TI products. TI's provision of these resources does not expand or otherwise alter TI's applicable warranties or warranty disclaimers for TI products. Unless TI explicitly designates a product as custom or customer-specified, TI products are standard, catalog, general purpose devices.

TI objects to and rejects any additional or different terms you may propose.

Copyright © 2026, Texas Instruments Incorporated

Last updated 10/2025