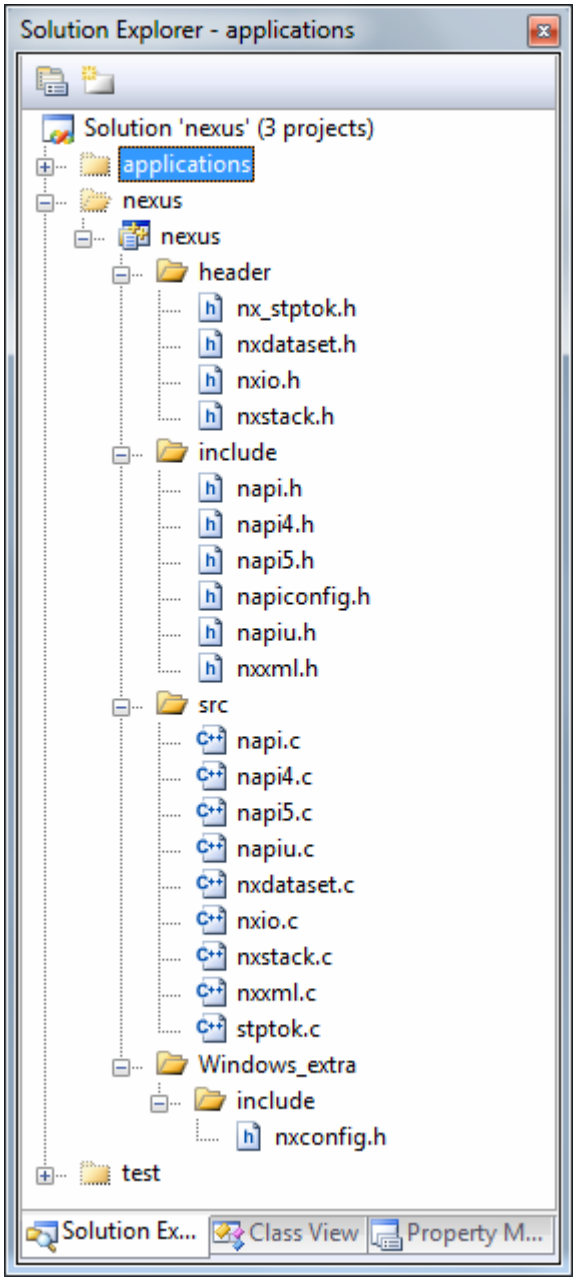
Building NeXus with Visual Studio 2008

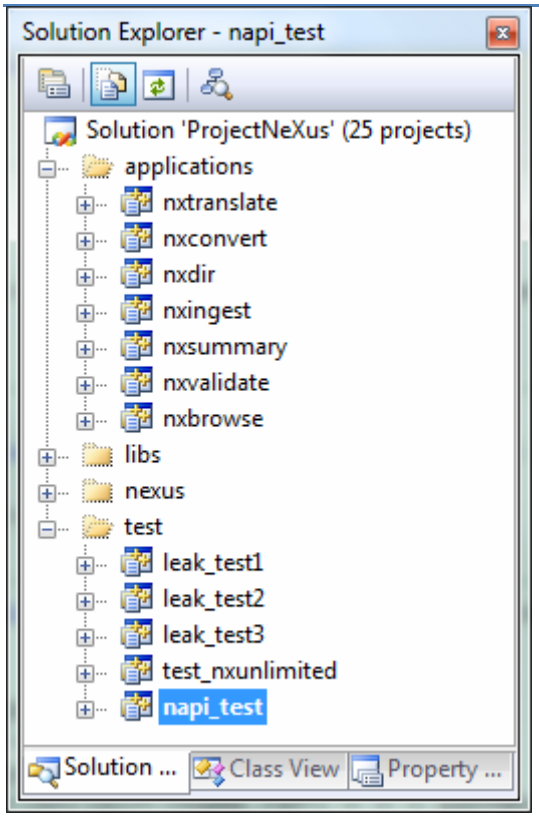


1 Introduction

Open the file /windows/nexus.sln with Visual Studio.

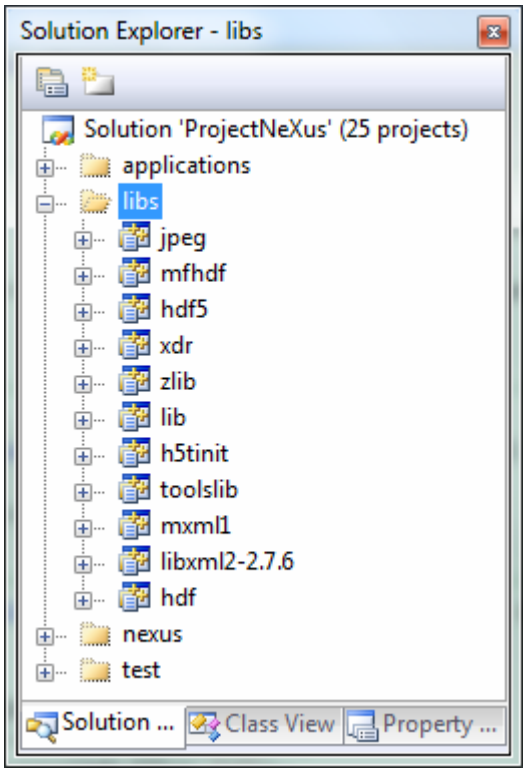
2 Project layout

The project layout is shown in the figure in the left. The applications folder contains projects for the NeXus applications and the test folder contains projects for the NeXus test programs. The folder nexus contains a project named nexus that contains the C source code for the NeXus API. The folder contains 4 subfolders named ‘header’ , the C header files currently located in /src of the NeXus distribution, ‘include’ , the C header files currently located in /include of the NeXus distribution, ‘src’ , the C source files currently located in /src of the NeXus distribution and ‘Windows_extra/include’ that mimics a folder with the same name in the NeXus distribution.



The 'Windows_extra/include' folder contains the 'nxconfig.h' file, that contains several Visual Studio system dependent macros for use in the NeXus API. Unlike the UNIX systems, this file is not generated automatically by the configure process in those systems.

The figure on the left contains the projects regarding the applications and test folders. For applications the projects are nxtranslate, nxconvert, nxdir, nxingest, nxsummary and nxbrowse. For the test folder, the projects are leak_test1, leak_test2, leak_test3, test_nxunlimited and napi_test.



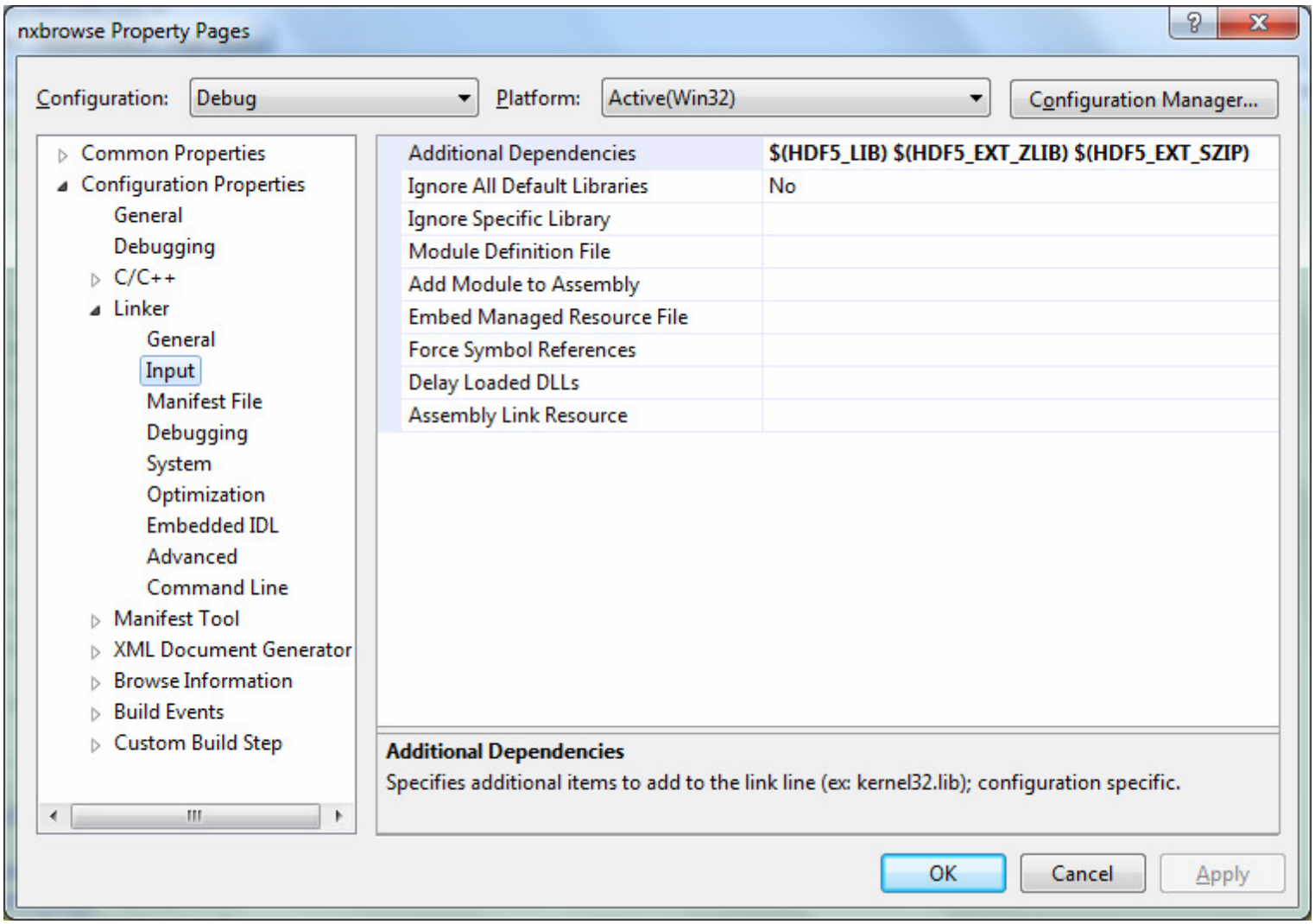
3 Add additional libraries

A feature of the Visual Studio IDE is that it allows to insert and delete projects by means of a GUI interface. Thus, it is possible to include in the solution projects for the base libraries that NeXus depends, such as HDF5 (and its dependencies SZLIB and ZLIB), HDF4 (and its dependency JPEG), and the XML libraries. This allows advanced developers to have direct access to the code of the underlying libraries, for debugging purposes, for example. Since these external libraries are not distributed with NeXus, these projects are not proposed to be included in the Visual Studio Solution, but can be made available for interested developers.

4 Applications project settings

This section shows how to modify the Visual Studio project settings to build the NeXus API and applications with the underlying libraries HDF5, HDF4 or NXXML.

Figure 4.1: Linker Input Property Page and additional libraries in Additional Dependencies



To link a NeXus application with a underlying library, two actions must be done, add the library name and path for each application:

4.1 Add the library name to be linked for each application

To library name to be linked is entered in the Linker Input Property Page (Figure 4.1), in the Additional Dependencies field. Either the literal library name or a Windows system environment variable[1] with the library name can be used. Figure 4.1 shows an example of NXBrowse linked with the HDF5 library: The following environment variables were defined, for the HDF5, ZLIB and SZIP libraries[2], HDF5_LIB, HDF5_EXT_ZLIB, HDF5_EXT_SZIP with the names of those libraries. Then, in the NeXus application Linker Input Property Page, in the Additional Dependencies field, the following must be entered

[1] To define an environment variable in Windows 7, click Start, right-click Computer, and select Properties. Select Advanced System Settings. Then, click the Environment Variables button.

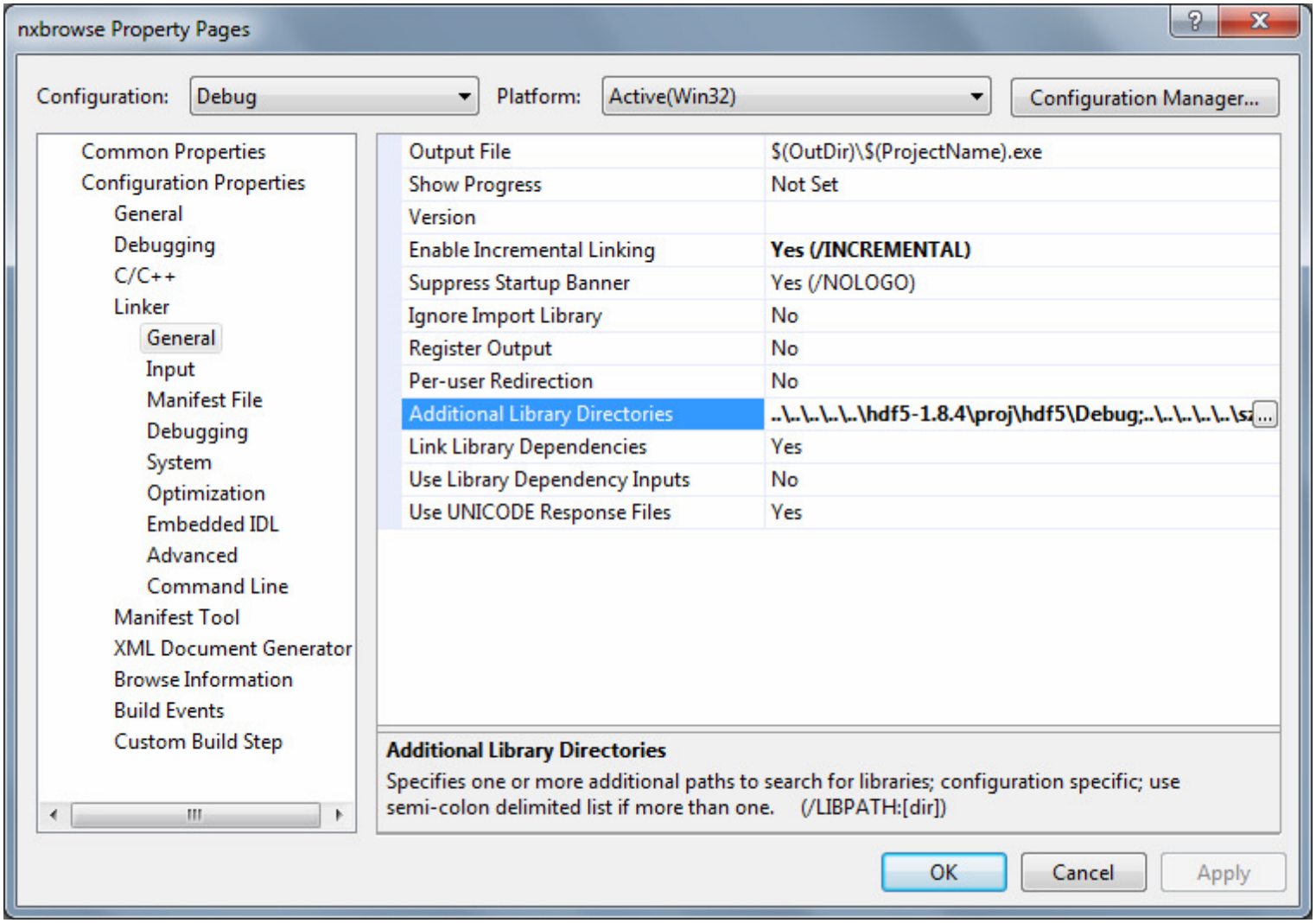
[2] The HDF5 library needs the SZIP and ZLIB libraries

$(HDF5_LIB) $(HDF5_EXT_ZLIB) $(HDF5_EXT_SZIP)

4.2 Add the library path to be linked for each application

This is done in the Linker General, Additional Library Directories field, shown in Figure 4.2

Figure 4.2: Linker General, Additional Library Directories



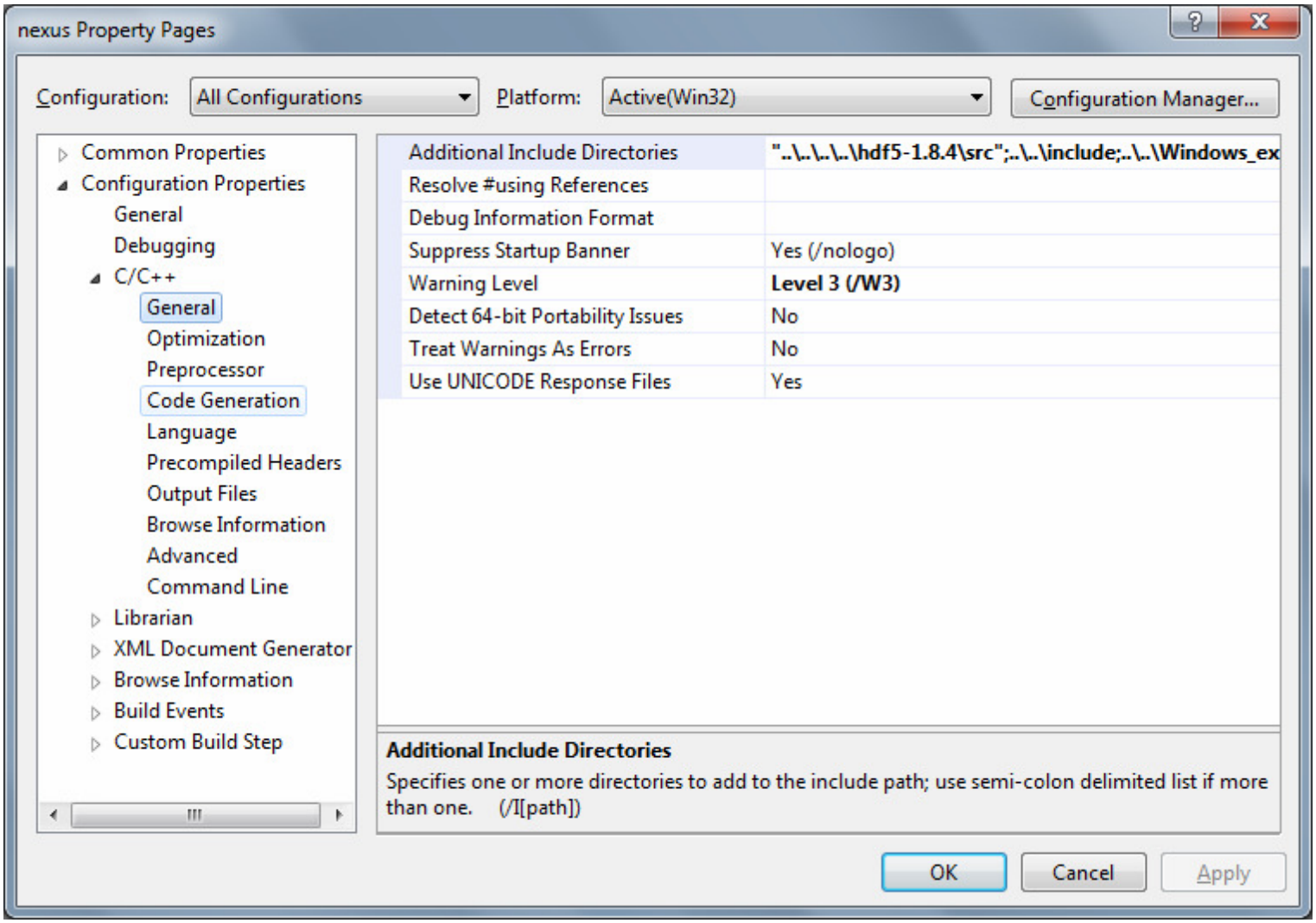
5 Nexus library project settings

To compile the NeXus library with any of the HDF5, HDF4 or NXXML underlying libraries, the following steps must be done in the Visual Studio editor.

5.1 Add the library path of the underlying library

This is done in the C/C++, General, Additional Include Directories of the **nexus** project file, Figure 5.1.

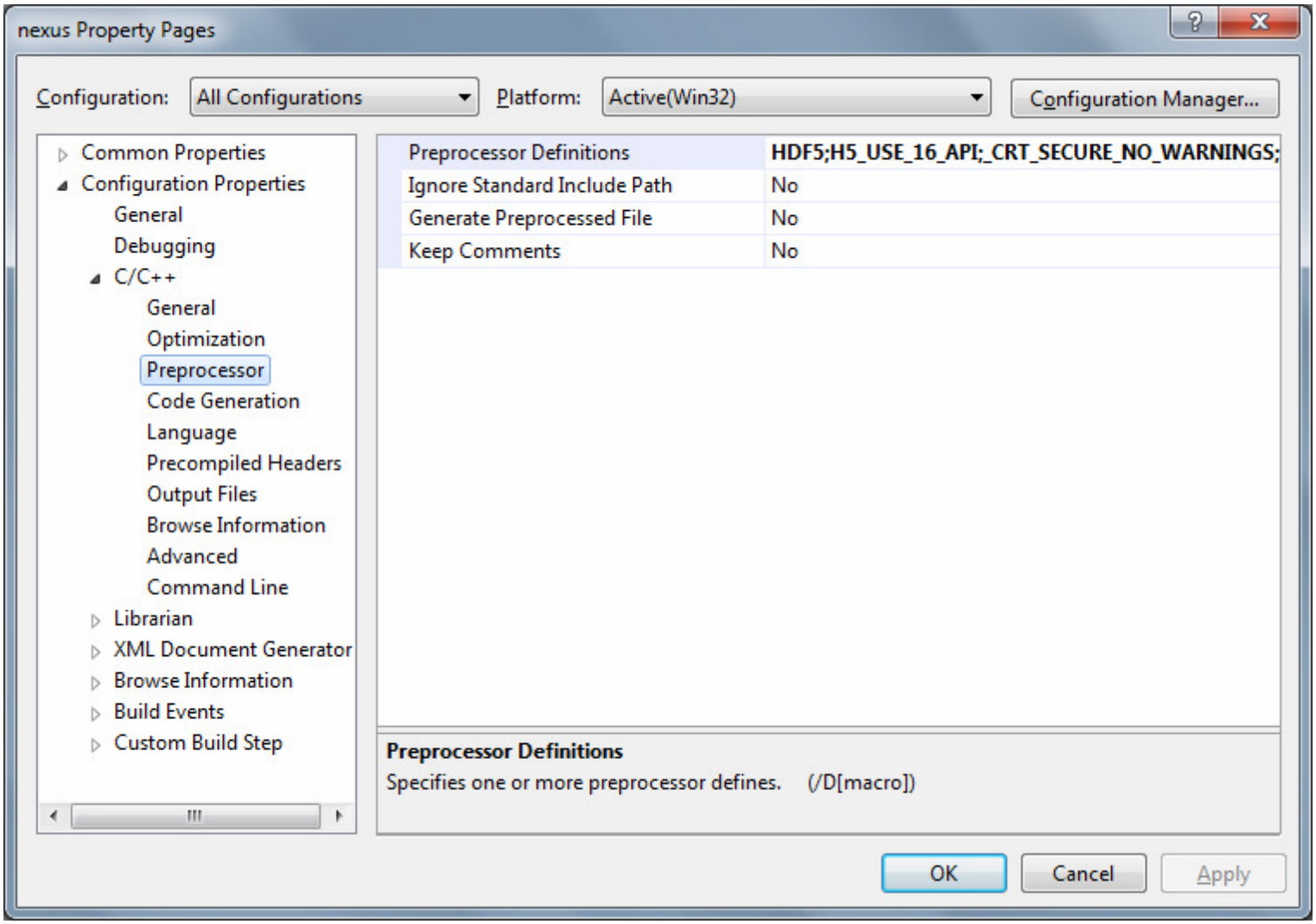
Figure 5.1: the C/C++, General, Additional Include Directories



5.2 Add a preprocessor definition for the underlying library

This is done in the C/C++, Preprocessor, Preprocessor Definitions field of the **nexus** project file, Figure 5.2.

Figure 5.1: the C/C++, Preprocessor, Preprocessor Definitions



For compiling with the HDF5 library, use the following definitions: HDF5, H5_USE_16_API. For compiling with the HDF4 library, use the following definition: HDF4. For compiling with the NXXML library, use the following definition: NXXML.