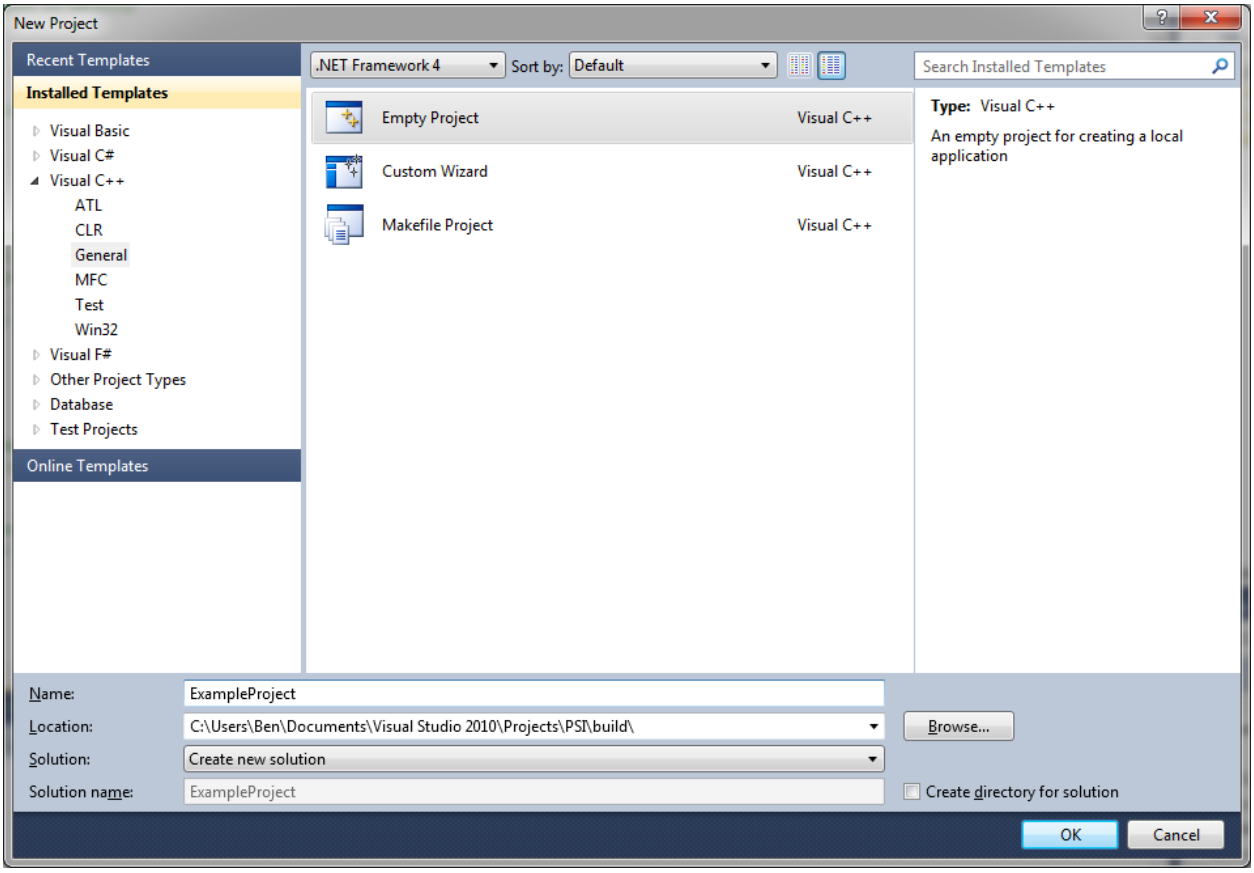
Setting up a Visual Studio 2010 project.

1 CREATING THE SOLUTION.

To keep the locations of all solution and project files clear, we will store them in the 'build' folder with their corresponding sub-folder. Choose 'Visual C++ -> Empty Project' for making a project and be sure to have the box with "Create directory for solution" unchecked. Make up a name for your project and click 'OK'.

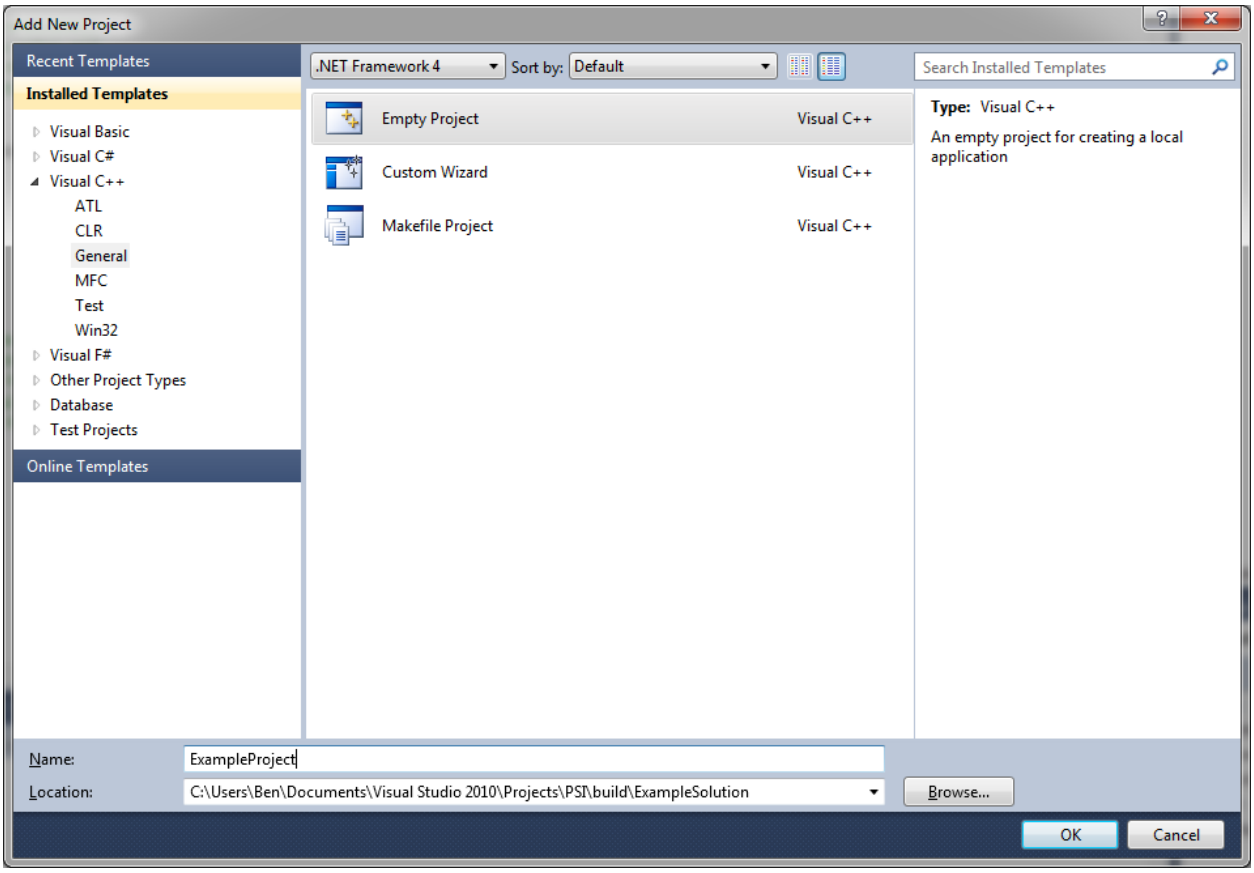


If you will use only one project in your solution then skip the following and skip to step 3.

The solution will have a project attached to it with the same name as your Solution. I would advise you to delete the project by right clicking on it and select 'remove'. Go into the 'build' folder and delete the vcxproj files.

2 ADDING A NEW PROJECT TO YOUR SOLUTION.

Right click on the solution in your solution browser and select ‘Add -> New project’. In the screen select ‘Visual C++ -> General -> Empty Project’ and make up a name. Click ‘Browse…’, navigate to the ‘build’ folder and into the corresponding sub-folder of your solution. Create a new folder inside the sub-folder bearing the same name as your project. Select the folder and submit. Then click ‘OK’ to create your project.

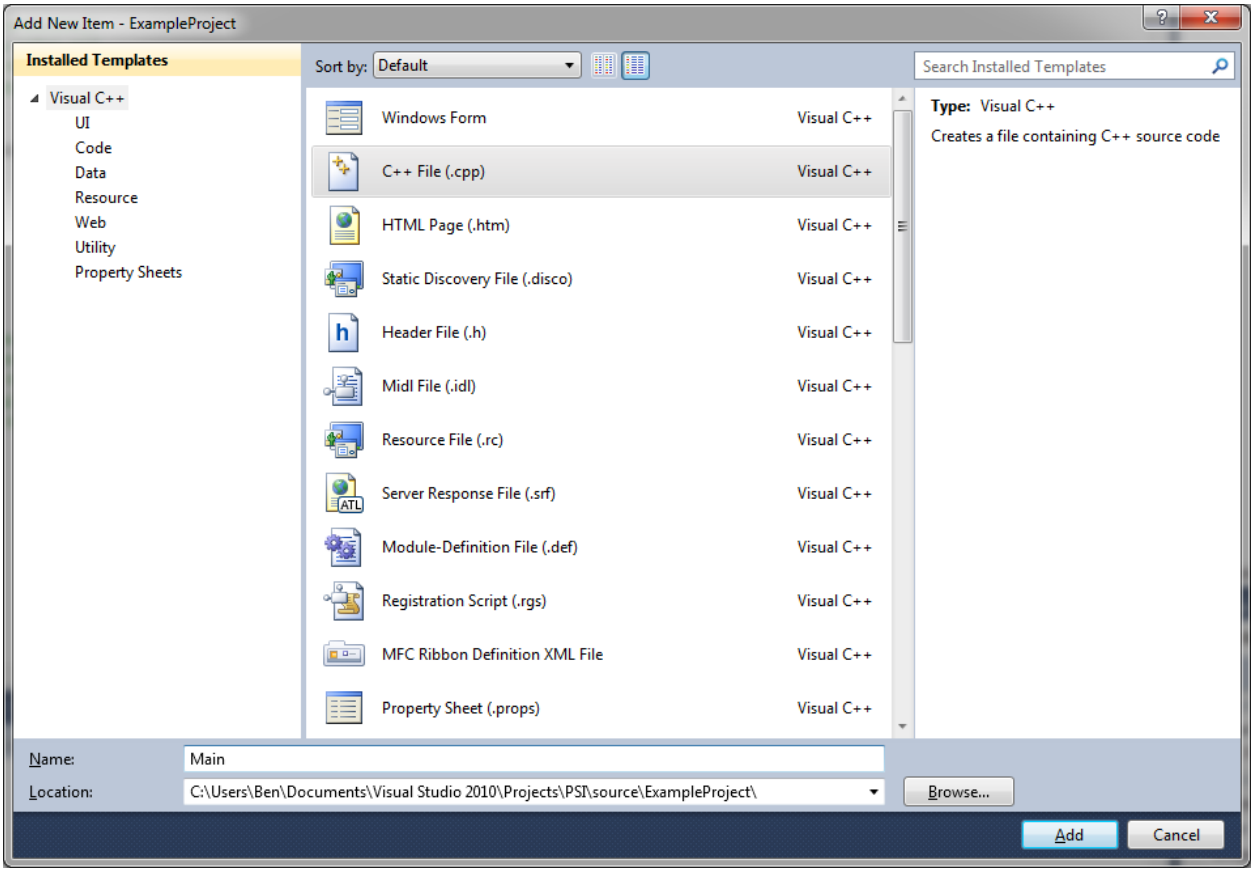


If you did everything right your folder structure will be like this:

- ***Root folder***
  - build
    - ExampleSolution (Containting the solution file)
      - ExampleProject (Containing the vcxproj file)

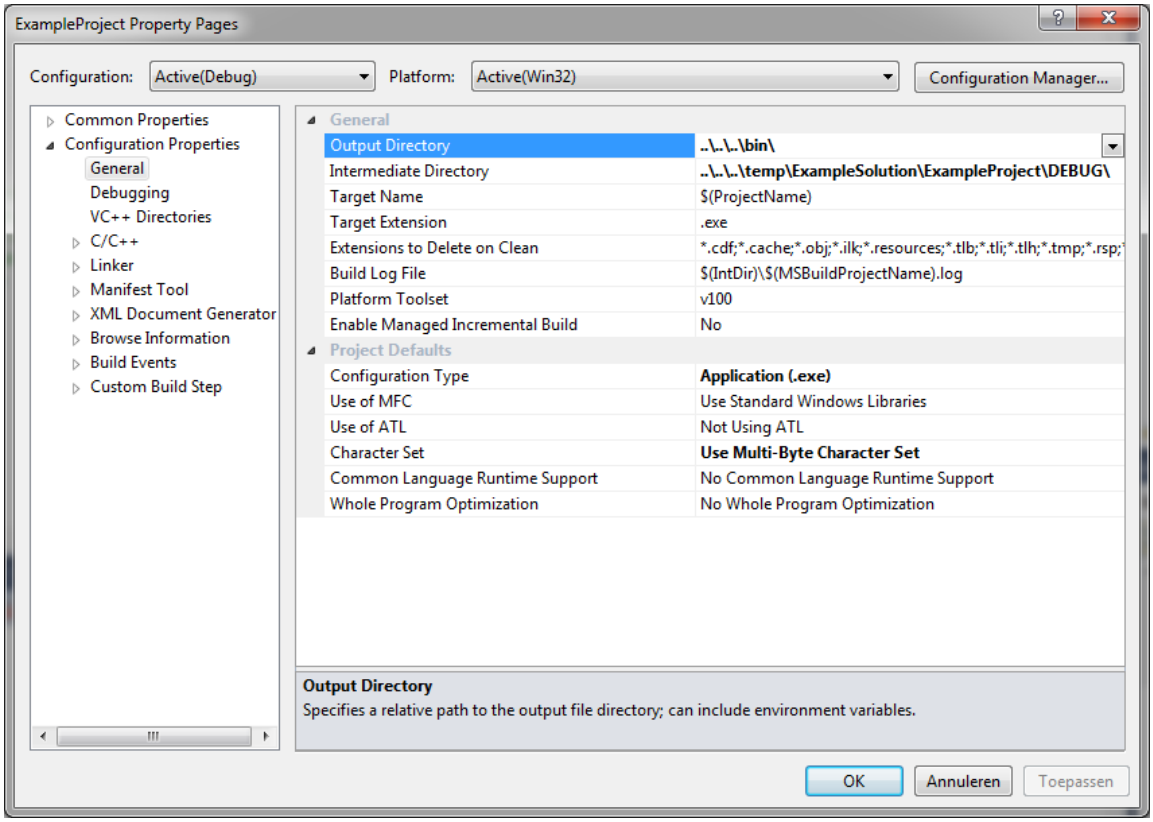
3 ADDING SOURCE FILES TO THE PROJECT.

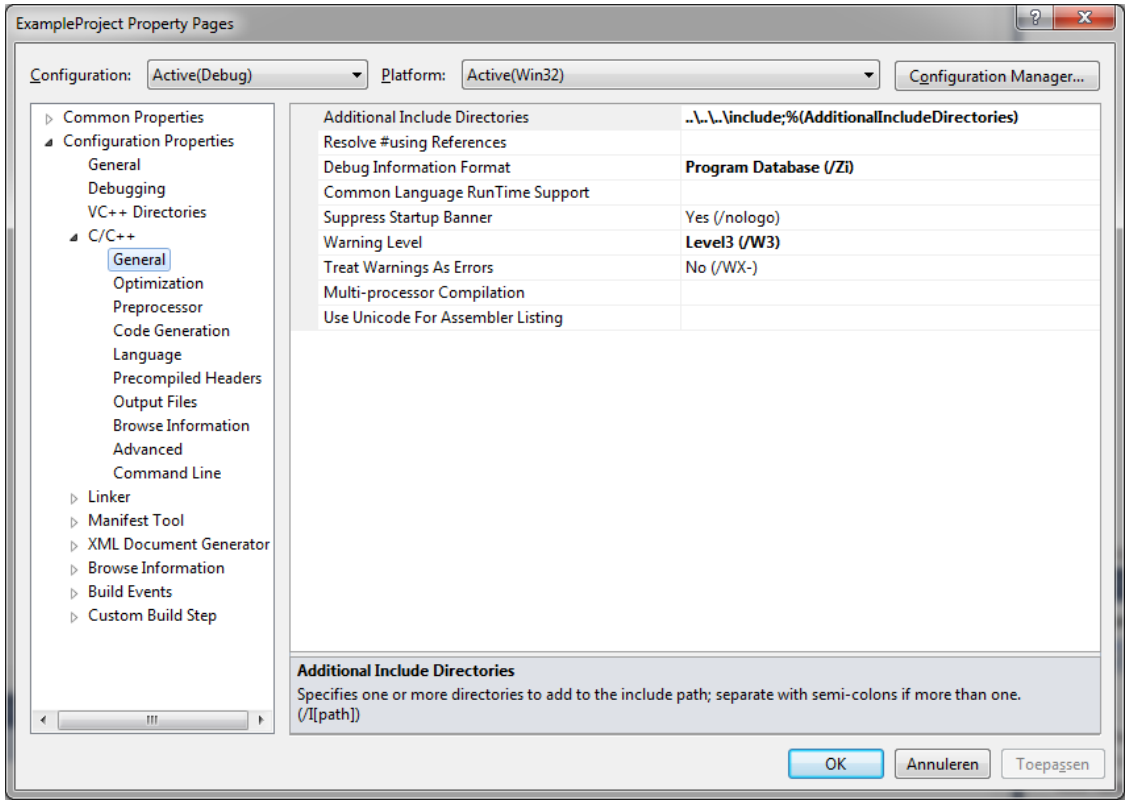
All the source files from your project will be saved in the corresponding folder in the ‘source’ folder. So for the example the ‘source’ folder would contain ‘ExampleProject’ (note: use the project name and NOT the solution name). In the solution browser: right click on the project and go to ‘Add -> New Item…’. Select C++ File (.cpp) and browse to the corresponding folder in ‘source’ by using ‘Browse…’. Make up a name and save the file. Always place your files in the corresponding ‘source’ folder! You can – of course – make your own sub-folders in the corresponding folder in ‘source’.

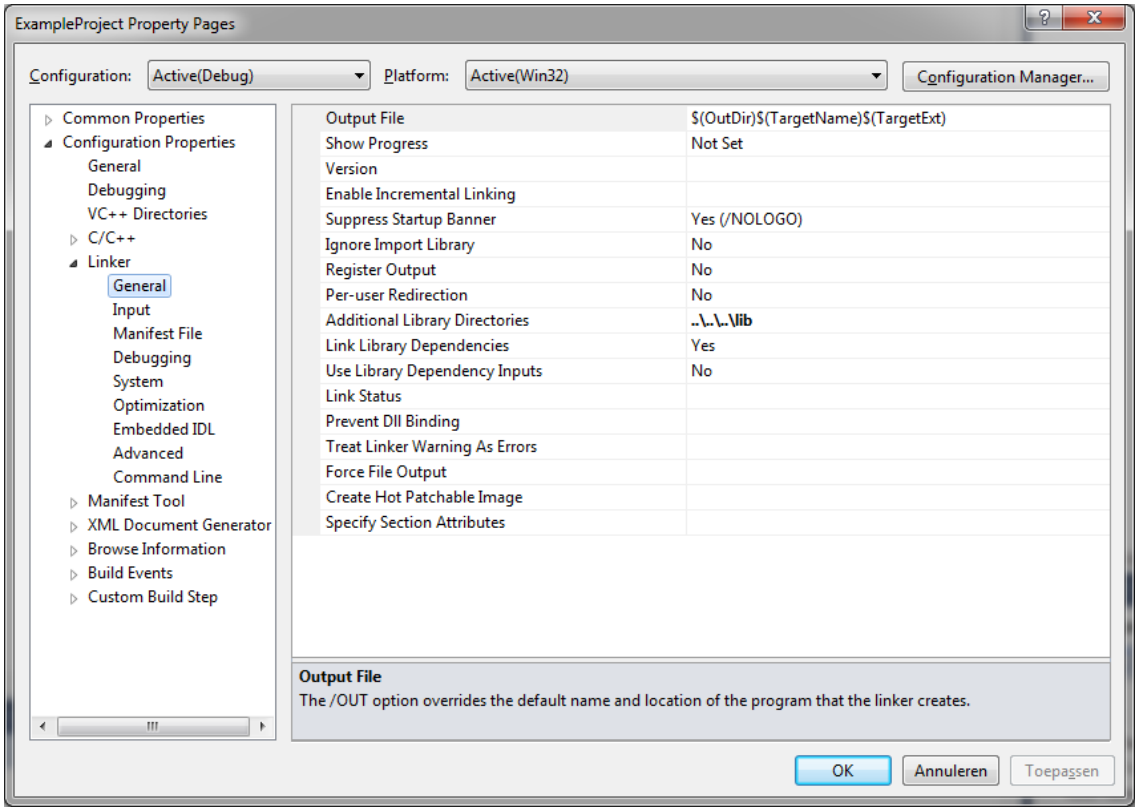


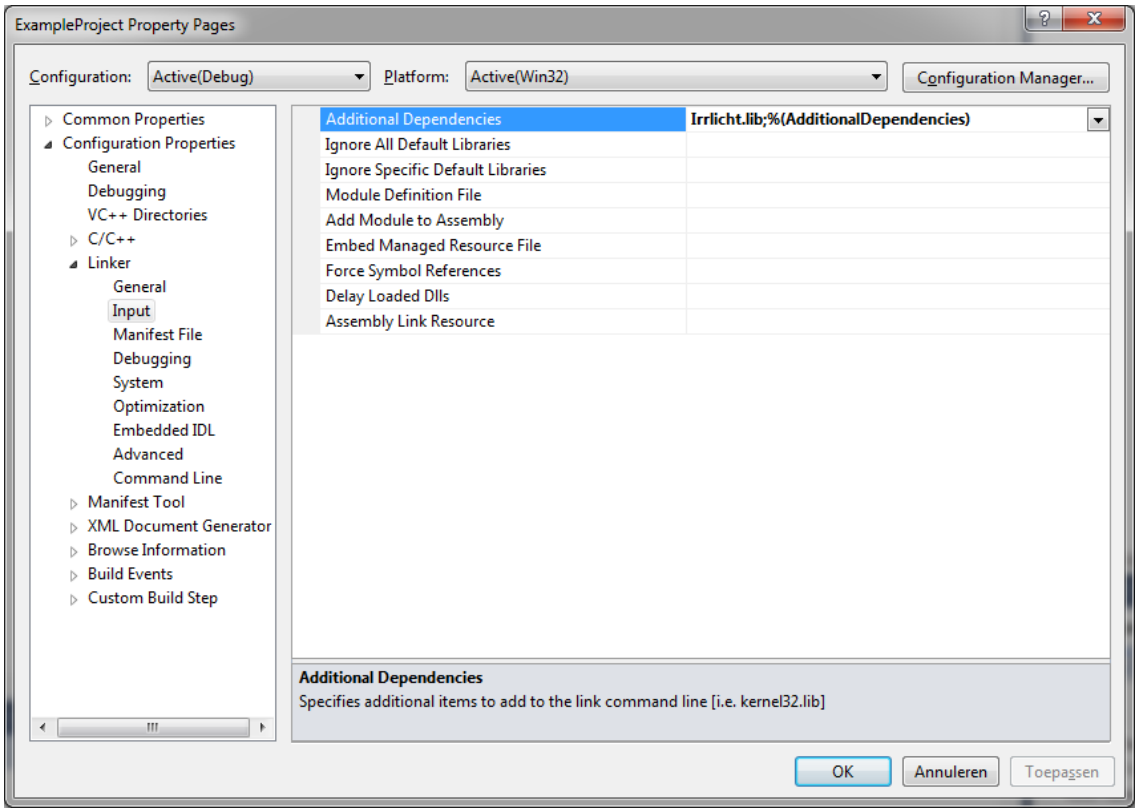
4 C++ AND LINKER OPTIONS OF THE PROJECT.

All the include files for your project will be stores in the ‘include’ folder, the library files are stored in the ‘lib’ folder, the executable of your code will be stored and compiled in the ‘bin’ folder and all the debug files are stored inside their corresponding folders in the ‘temp’ folder. The ExampleProject only uses the Irricht engine for illustration and instead of writing down lots of text down here for every option, I will just show screenshots of all the changes in the options. Note: all the paths are relative to the vcxproj files.




ExampleProject Property Pages
Configuration: Active(Debug)
Platform: Active(Win32)
Configuration Manager...
Common Properties
Configuration Properties
General
Debugging
VC++ Directories
C/C++
General
Optimization
Preprocessor
Code Generation
Language
Precompiled Headers
Output Files
Browse Information
Advanced
Command Line
Linker
Manifest Tool
XML Document Generator
Browse Information
Build Events
Custom Build Step
Additional Include Directories
..\..\..\include;%(AdditionalIncludeDirectories)
Resolve #using References
Debug Information Format
Program Database (/Zi)
Common Language RunTime Support
Suppress Startup Banner
Yes (/nologo)
Warning Level
Level3 (/W3)
Treat Warnings As Errors
No (/WX-)
Multi-processor Compilation
Use Unicode For Assembler Listing
Additional Include Directories
Specifies one or more directories to add to the include path; separate with semi-colons if more than one.
(/I[path])
OK
Annuleren
Toepassen


ExampleProject Property Pages
Configuration: Active(Debug)
Platform: Active(Win32)
Configuration Manager...
Common Properties
Configuration Properties
General
Debugging
VC++ Directories
C/C++
Linker
General
Input
Manifest File
Debugging
System
Optimization
Embedded IDL
Advanced
Command Line
Manifest Tool
XML Document Generator
Browse Information
Build Events
Custom Build Step
Output File
$(OutDir)$(TargetName)$(TargetExt)
Show Progress
Not Set
Version
Enable Incremental Linking
Suppress Startup Banner
Yes (/NOLOGO)
Ignore Import Library
No
Register Output
No
Per-user Redirection
No
Additional Library Directories
..\..\..\lib
Link Library Dependencies
Yes
Use Library Dependency Inputs
No
Link Status
Prevent Dll Binding
Treat Linker Warning As Errors
Force File Output
Create Hot Patchable Image
Specify Section Attributes
Output File
The /OUT option overrides the default name and location of the program that the linker creates.
OK
Annuleren
Toepassen



If you did everything right the project should be compatible with the folder structure in the Master branch and SHOULD be easy to merge.

Please tell me if you run into any trouble or find some problems with this approach.