Compiling DynAdjust on Windows

Roger Fraser
30 June 2018

These notes outline the steps for compiling DynAdjust on Windows using Microsoft's freely available Visual Studio 2017 Community Edition.

1. Install Microsoft Visual Studio 2017 Community Edition

Microsoft's Visual Studio 2017 Community Edition is available from
https://visualstudio.microsoft.com/thank-you-downloading-visual-studio/?sku=Community&rel=15

Once installed, the prerequisites must be installed before attempting to compile DynAdjust.

2. Install Prerequisites

There are three prerequisites that must be installed. These are:

1. Boost C++ headers and libraries
2. CodeSynthesis XSD and Apache xerces-c
3. Intel's Math Kernel Library (MKL) and Threaded Building Blocks (TBB)

2.1 Boost C++ header and library paths

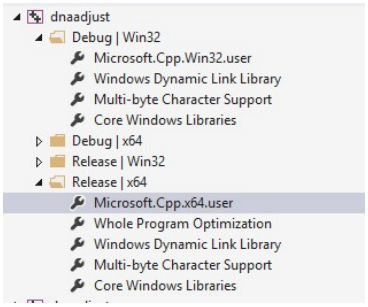
DynAdjust requires boost headers and libraries. The headers and library sources are available from
https://www.boost.org/users/download/

The boost libraries needed by DynAdjust include `filesystem`, `system`, `program_options`, `thread`, `date_time`, `math`, `timer`, `atomic` and `chrono`. These will need to be built from the boost C++ sources, and installed to a user-defined location (e.g. `C:\Data\boost\boost_1_67_0\`).

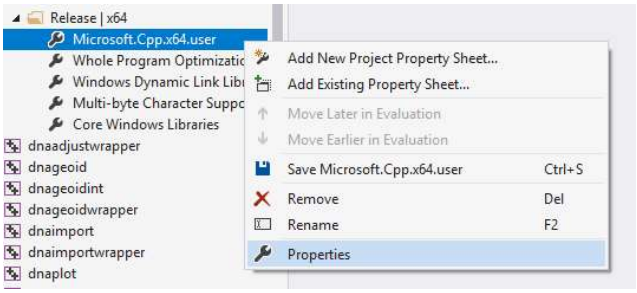
By default, Visual Studio will produce an error message stating that it was unable to find the boost header files or libraries. To resolve this issue, two User Macros containing the boost header and library folder paths need to be added to the Visual Studio IDE. These User Macros, named **`BoostIncludeDir`** and **`BoostLibDir`**, are referenced throughout the solution's project properties and provide a convenient way to reference unique (and varied) boost C++ file paths without having to change the solution property pages.

For Visual Studio 2010/10.0 and later:

1. Open an existing project, or create a new C++ project
2. Open the Property Manager view. If the Property Manager window is not shown, click **View | Property Manager** or **View | Other Windows | Property Manager** from the menu.
3. Expand the property hierarchy for any one of the projects and find the Microsoft.Cpp.x64.user property sheet



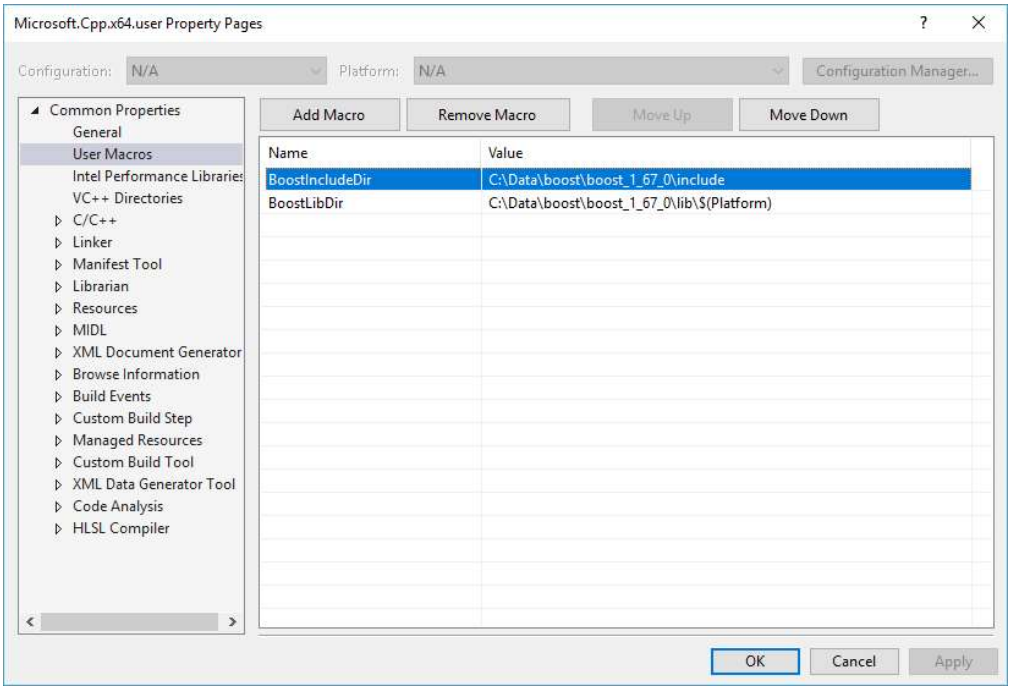
4. Right click on **Microsoft.Cpp.x64.user** and select the **Properties** menu action



5. Under **Common Properties**, select **User Macros**
6. Add two new macros named **`BoostIncludeDir`** and **`BoostLibDir`** with values set to wherever the directories are stored. For example, if the boost paths are:

`BoostIncludeDir`	`C:\Data\boost\boost_1_67_0\include\`
`BoostLibDir`	`C:\Data\boost\boost_1_67_0\lib\x64\` `C:\Data\boost\boost_1_67_0\lib\Win32\`

then the following User Macros can be created. The Macro **`Platform`** is a standard Visual Studio Macro that can be used to distinguish between the 32-bit and 64-bit build configurations. An example is given below.



7. Repat steps 4 - 6 for the **Microsoft.Cpp.Win32.user** property sheet.

2.2 CodeSynthesis XSD and Apache xerces-c header and library paths

DynAdjust requires CodeSynthesis XSD (version 4.0) headers and Apache xerces-c headers and libraries. The x86 and x64 Windows dependencies are available as a bundle via: https://www.codesynthesis.com/products/xsd/download.xhtml

If the default installation path (`C:\Program Files (x86)\CodeSynthesis XSD 4.0`) is used during setup, the XSD and xerces-c paths will be correctly referenced and DynAdjust will compile without any changes

However, if an alternative installation path is chosen, the project files will need to be amended with the alternative paths. Alternatively, the **Microsoft.Cpp.x64.user** and **Microsoft.Cpp.Win32.user** property sheets in Property Manager can be amended with the correct header and library paths. In the latter case, the following steps can be followed.

For Visual Studio 2010/10.0 and later:

1. Open an existing project, or create a new C++ project
2. Open the Property Manager view. If the Property Manager window is not shown, click **View | Property Manager** or **View | Other Windows | Property Manager** from the menu.
3. Expand the property hierarchy for any one of the projects and find the Microsoft.Cpp.x64.user property sheet (as above)
4. Right click on **Microsoft.Cpp.x64.user** and select the **Properties** menu action
5. Under **Common Properties**, select **VC++ Directories**
6. Add the CodeSynthesis header and library paths to **`Include directories`** and **`Library directories`** as follows:

   `Include directories` `C:\Program Files (x86)\CodeSynthesis XSD 4.0\include`

   `Library directories` `C:\Program Files (x86)\CodeSynthesis XSD 4.0\lib64\vc-10.0`
   `C:\Program Files (x86)\CodeSynthesis XSD 4.0\lib\vc-10.0`

7. Repat steps 4 - 6 for the **Microsoft.Cpp.Win32.user** property sheet.

2.3 Intel's Math Kernel Library (MKL) and Threaded Building Blocks (TBB)

DynAdjust requires Intel's MKL and TBB libraries. The free versions of Intel's MKL and TBB libraries are available from https://software.seek.intel.com/performance-libraries

The DynAdjust VS2017 solution references Intel's MKL and TBB libraries and headers using two Macros, named **`MKLIncludeDir`** and **`TBBIncludeDir`**. Upon installing Intel's performance libraries from the above Web link, the installer should create these Macros which will reference the folder where the performance libraries were installed.

If DynAdjust fails to compile because it cannot find Intel's MKL or TBB, check that the Macros **`MKLIncludeDir`** and **`TBBIncludeDir`** point to the folder where the libraries and headers were installed.

3. Compiling DynAdjust

3.1 Solution architecture

DynAdjust is comprised of several executables and dependent dynamic link libraries (DLL). The project name of each executable is named using the convention `dna<program-name>wrapper`, except for the main program `dynadjust`. Upon compilation, these projects will create executables named `<program-name>`.

Each executable named `dna<program-name>wrapper` is dependent on a DLL named `dna<program-name>`. These must and will be compiled first before compiling the executables.

The executable projects and their dependent DLLs are listed below:

```
+ dnaadjustwrapper
  + dnaadjust
+ dnageoidwrapper
  + dnageoid
+ dnaimportwrapper
  + dnaimport
+ dnaplotwrapper
  + dnaplot
+ dnareftranwrapper
  + dnareftran
+ dnasegmentwrapper
  + dnasegment
+ dynadjust (no project dependencies, but requires all project to be built for normal
    execution behaviour)
```

3.2 Build Configurations

Four build configurations have been created:

1. Debug Win32
2. Release Win32
3. Debug x64
4. Release x64

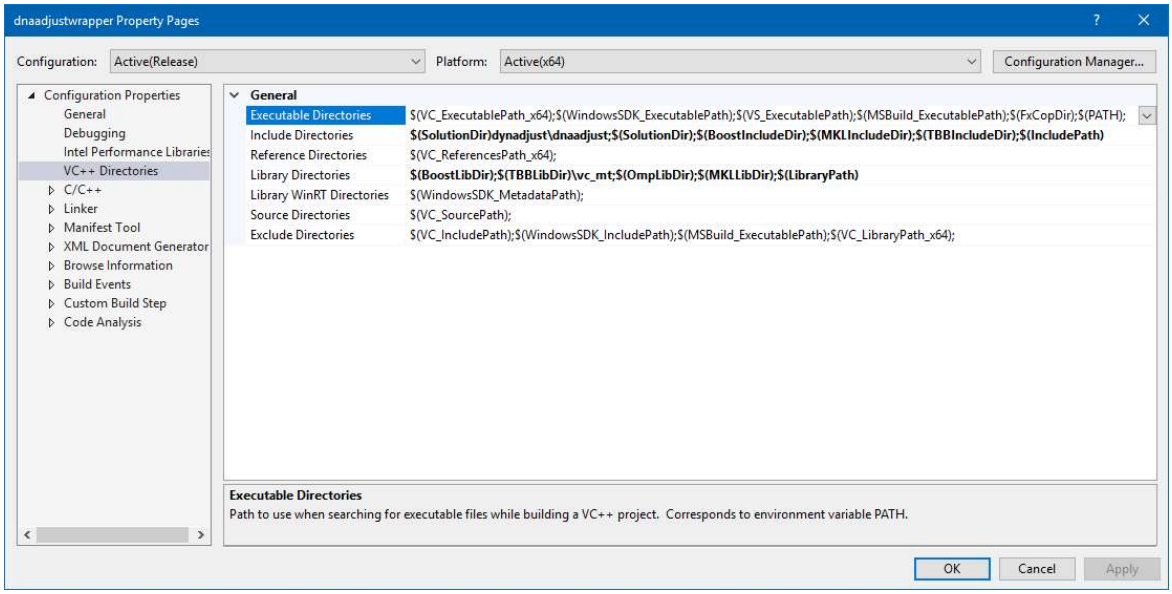
The project properties pages for each executable and DLL project make use of User Macros that simplify the creation of settings for the four configurations.

3.3 Precompiled headers

Given that many functions are shared throughout the suite of executables and DLLs, the DynAdjust solution makes extensive use of precompiled headers to simplify and speed up compile time.

3.4 Example compilation

The following image shows the C++ Directories for the `dnaadjustwrapper` project.



Successful compilation `dnaadjustwrapper`, which invokes compilation of `dnaadjust`, is shown below.

```
1>------ Rebuild All started: Project: dnaadjust, Configuration: Release x64 ------
1>precompile.cpp
1>dnastringfuncs.cpp
1>trace.cpp
1>dnaioadj.cpp
1>dnaioaml.cpp
1>dnaioasl.cpp
1>dnaiobase.cpp
1>dnaiobms.cpp
1>dnaiobst.cpp
1>dnaiomap.cpp
1>dnaioseg.cpp
1>dnaiosnxwrite.cpp
1>dnamatrix_contiguous.cpp
1>dnamsrtally.cpp
1>dnastation.cpp
1>dnastntally.cpp
1>dnafile_mapping.cpp
1>dnadatum.cpp
1>dnaellipsoid.cpp
1>dnaprojection.cpp
1>dnaadjust-stage.cpp
1>dnaadjust.cpp
1>   Creating library C:\Data\vs17\dynadjust_1_00_01\build\dnaadjust\Release\x64\dnaadjust.lib and object
C:\Data\vs17\dynadjust_1_00_01\build\dnaadjust\Release\x64\dnaadjust.exp
1>Generating code
1>All 7369 functions were compiled because no usable IPDB/IOBJ from previous compilation was found.
1>Finished generating code
1>dnaadjust.vcxproj -> C:\Data\vs17\dynadjust_1_00_01\bin\Release\x64\dnaadjust.dll
2>------ Rebuild All started: Project: dnaadjustwrapper, Configuration: Release x64 ------
2>precompile.cpp
2>dnaprojectfile.cpp
2>dnastringfuncs.cpp
2>dnaiobase.cpp
2>dnaiobms.cpp
2>dnaiobst.cpp
2>dnadatum.cpp
2>dnaellipsoid.cpp
2>dnaadjustprogress.cpp
2>dnaadjustwrapper.cpp
2>Generating code
2>All 3620 functions were compiled because no usable IPDB/IOBJ from previous compilation was found.
2>Finished generating code
2>dnaadjustwrapper.vcxproj -> C:\Data\vs17\dynadjust_1_00_01\bin\Release\x64\adjust.exe
========== Rebuild All: 2 succeeded, 0 failed, 0 skipped ==========
```