**Setting up 32-bit and 64-bit ADMB 10 for Windows with the Microsoft Visual C++ 2010 compilers contained in the SDK 7.1 installation**

Allan Hicks
Allan.Hicks@noaa.gov
Ian Taylor
March 9, 2012

This is an installation guide for setting up both 32-bit and 64-bit versions of ADMB for Windows on a single computer using the compiler from Microsoft Visual Studio 2010. To set up ADMB and the compilers, you must have administrative access for the installation as well as to change system-wide environmental variables. If ADMB and Visual Studio express are installed and you do not have administrative access, you can still set up the proper environmental variables, but in a slightly different location than identified here.

This setup differs from the setup described on the admb-project website. I have chosen to eliminate the batch scripts that are designed to set up environmental variables (such as the PATH) and to manually enter these variables such that they are persistent and automatically available when any command window is opened. This greatly increases my ability to work with ADMB, especially when opening command prompts in specific windows to quickly compile ADMB code, or when compiling ADMB code from within text editors such as textpad.

**NOTE:** *These instructions indicate the way that I set up ADMB on my computers and I do not guarantee that they will work on your computer. In addition, because of different versions of SDK and Microsoft.NET, these instructions may need to be modified slightly depending on your versions. This guide will likely be updated in the future and I appreciate any feedback.*

# 1 ADMB installation

AD Model Builder (ADMB) can be downloaded from the ADMB project website (`http://admb-project.org`) and is available for 32-bit and 64-bit Windows Visual Studio compilers. This guide assumes that you will want to install both 32-bit and 64-bit compilers, but can be used to install only the 32-bit compiler. To use the 64-bit compiler and executables compiled with the 64-bit compiler, you must have a 64-bit Windows operating system.

1. Install the 32-bit ADMB Visual Studio version (admb-10.X-vc10-32bit.exe[1])
   - Available from `http://admb-project.org/downloads` or at `http://admb-project.googlecode.com`
     - If you do not find the exact version you would like, you may want to check out `http://www.admb-project.org/buildbot/snapshots/`

[1]The 'X' refers to the minor version number

- Install it into a folder that is accessible by all users (i.e., do not install in the Program Files directory if using it with non-administrative accounts). To keep 32-bit and 64-bit installations separate, install them into their own directories. I suggest and assume that you will install 32-bit ADMB into the directory `C:\ADMB\ADMB10\Microsoft\32bit`. Be aware that the installer may add an additional folder, but I assume the exact folder mentioned previously.

2. Install the 64-bit ADMB Visual Studio version (admb-10.X-vc10-64bit.exe[1])

   - Available from `http://admb-project.org/downloads` or at `http://admb-project.googlecode.com`
     - If you do not find the exact version you would like, you may want to check out `http://www.admb-project.org/buildbot/snapshots/`
   - Install it into a folder that is accessible by all users (i.e., do not install in the Program Files directory if using it with non-administrative accounts). To keep 32-bit and 64-bit installations separate, install them into their own directories. I suggest and assume that you will install 64-bit ADMB into the directory `C:\ADMB\ADMB10\Microsoft\64bit`. Be aware that the installer may add an additional folder, but I assume the exact folder mentioned previously.

There is no need to run any admb batch files. We'll set these environmental variables up such that they are persistent.

## 2 Install Visual Studio 2010 compilers

There are three options for installation of the Visual Studio 2010 compilers onto your computer (Our recommendation is option 3).

**Option 1** Install the paid for version of Visual Studio 2010

- This will by default install 32-bit compilers and you should select the option to install 64-bit compilers.

**Option 2** Install the free version of Visual Studio 2010 (called Microsoft Visual C++ Express)

- `http://www.microsoft.com/express/vc/`
- Includes only the 32-bit compiler.
- It does install an IDE (integrated development environment) that can be used to edit code with syntax highlighting as well as compile ADMB code with the proper setup.
- I do not see a need to install Silverlight or the SQL server add-ons unless you would like them for additional projects
- You will have to also follow the next step to install 64-bit compilers

**Option 3** Install Microsoft SDK for Windows 7 with .NET Framework 3.5 SP 1 (**Recommended**).

- `http://www.microsoft.com/download/en/details.aspx?id=8279`
  or search for `Windows SDK 7.1`
- You can download the web installer (recommended) or the ISO to burn to a CD for later use.
- Is free and includes both 32-bit and 64-bit Visual Studio 2010 compilers. (Make sure to check the box next to Install Visual C++ compilers)
- Does not include an IDE, but compiles ADMB code via the command line.
- Works with the Visual Studio Express IDE if you wish to use that environment.
- To greatly reduce download and installation time, do not install documentation and examples (uncheck these options)
- There is no need to install .NET 4 if you do not already have it.

I assume that you will use option 3 above and install the SDK (software development kit). If you wish to use the Visual Studio Express IDE, install Visual Studio Express first then install the SDK for Windows 7. If you use the paid for Visual Studio 2010, you do not need to install the SDK, but do need to make sure that you select the 64-bit compilers when installing Visual Studio 2010.

You do not need to install Visual Studio 2010 or Visual Studio 2010 Express to compile ADMB code. The compilers are installed with the SDK and a Microsoft Visual Studio folder is created in the Program Files directory.

# 3 Setting up the environmental variables for ADMB and Visual Studio

When compiling ADMB code, the compiler needs to know where to look for header and library files. Environmental variables are used to indicate where these files are. To set these environmental variables for the entire system (all users) you will need administrator access. If you do not have administrator access, follow my guidelines for setting environment variables for the "user".

## 3.1 Manual setup of environment variables for ADMB 32-bit in Windows

1. Windows XP
   - Right click on "My Computer" and select Properties, or choose "System" from the Control Panel.
   - Click on the "Advanced" tab at the top
   - Choose "Environment Variables" near the bottom
2. Windows 7
   - Go to the Control Panel and find your user account, or go to
     `Control Panel\User Accounts\User Accounts`
   - On the left, click on the option "Change my environment variables"

3. There are two windows here: "User variables for ..." and "System variables". We will modify the "User Variables". If you have administrator access and you would like to make these variables available to all users (and ADMB and Visual Studio are already installed on your computer and available to all users) you can do the following steps to the "System variables." BE CAREFUL NOT TO CHANGE ANY SETTINGS ALREADY MADE, BUT TO ADD TO THEM.

4. Under "User variables for ..." at the bottom, check to see if `ADMB_HOME` is present.

   (a) If not, choose `New...`. If it is present, select `ADMB_HOME` and choose `Edit`

   (b) "Variable name:" `ADMB_HOME`

   (c) "Variable value:" type in the directory you chose during installation of the 32-bit version (Step 2 above).

   - For example, I typed in `C:\ADMB\ADMB10\Microsoft\32bit`.
   - If setting under "System variables," make sure `ADMB_HOME` isn't set in "User variables for ..." as this will mask the "System variables."

5. Under "User variables for ...," choose `Path` and click on `Edit...`, or if `Path` is not present, choose `New...`.

   (a) Confirm that the following three folders exist, and enter them at the end of all the text in "Variable Value" with semi-colons separating them. Modify them to match your folders, if necessary.
   `C:\ADMB\bin`
   `C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\bin`
   `C:\Program Files (x86)\Microsoft Visual Studio 10.0\Common7\IDE`

   (b) **Make sure to keep the current text there or serious problems could occur!**

   (c) The `C:\ADMB\bin` folder is where all of the ADMB batch scripts and the tpl2cpp executable are kept. Note that I used `C:\ADMB\bin` instead of `C:\ADMB\ADMB10\Microsoft\32bit\bin`. You can enter the path to any folder, as long as it contains the batch files to do your compiling. For example, I have copied all of the items included in the installed ADMB bin folder (i.e., `C:\ADMB\ADMB10\Microsoft\32bit\bin`) to the location `C:\ADMB\bin`. This allows me to add new scripts to this folder, as we'll see below for the 64-bit installation.

   (d) The `VC\bin` folder houses the Visual Studio 10 compiler. It may not have the (x86) if you do not have a 64-bit operating system. Check that this folder exists and separate folders in the "Variable Value" field with a semicolon.

   (e) The `Common7\IDE` folder is the Visual Studio 10 Common IDE folder. It may not have the (x86). Check that this folder exists and separate folders in the "Variable Value" field with a semicolon.

6. Under "User variables for ...", check to see if `MSSDK` is present

   (a) If it is not, choose `New`. If it is, select it and choose `Edit`

   (b) "Variable name:" `MSSDK`

   (c) "Variable Value:"
   `C:\Program Files\Microsoft SDKs\Windows\v7.1`

(d) You may want to verify that this directory exists and is where these files were installed. If not, you may have to search for the directories or install the programs

(e) Make sure `MSSDK` is not set anywhere else. If it is, I don't recommend changing it, as I am not sure what other programs reference that variable. Your installation is more advanced and I would suggest modifying the batch scripts specifically.

7. Under "System Variables" at the bottom, check to see if there is a variable called `Include`

   (a) If there is not, choose `New`. If there is, click on `Include` and choose `Edit`

   (b) "Variable name:" `Include`

   (c) "Variable Value:" add in the directory
   `C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\include`

   (d) You may want to verify that this directory exists and is where these files were installed. If not, you may have to search for the directories or install the programs

   (e) If `Include` was already set, you can add the path to the list by separating from other paths with a semicolon. However, be aware that the search goes from the first folder listed and some files may be masked from the actual files in this folder.

8. Under "System Variables" at the bottom, check to see if `Lib` is present

   (a) If it is not, choose `New`. If it is, select it and choose `Edit`

   (b) "Variable name:" type `Lib`

   (c) "Variable Value:" add in the directory
   `C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\lib`

   (d) You may want to verify that this directory exists and is where these files were installed. If not, you may have to search for the directories or install the programs

   (e) If `Lib` was already set, you can add the path to the list by separating from other paths with a semicolon. However, be aware that the search goes from the first folder listed and some files may be masked from the actual files in this folder.

This should allow you to open a command window in any directory and immediately compile an admb tpl file in 32-bit without setting up any variables or navigating to various directories. One problem that I foresee is the ADMB, Microsoft SDK and Visual Studio directories change when updates occur. The variable values will have to be modified when updates occur. However, when a new version of any of those programs is installed and desired to be used, you will have to make sure that ADMB works with them. Therefore, an update usually means updating everything.

### 3.2 Manual setup of environment variables for ADMB 64-bit in Windows

1. Windows XP

   - Right click on "My Computer" and select Properties, or choose "System" from the Control Panel.
   - Click on the "Advanced" tab at the top

- Choose "Environment Variables" near the bottom

2. Windows 7
   - Go to the Control Panel and find your user account, or go to `Control Panel\User Accounts\User Accounts`
   - On the left, click on the option "Change my environment variables"
3. There are two windows here: "User variables for ..." and "System variables". We will modify the "User Variables". If you have administrator access and you would like to make these variables available to all users (and ADMB and Visual Studio are already installed on your computer and available to all users) you can do the following steps to the "System variables." BE CAREFUL NOT TO CHANGE ANY SETTINGS ALREADY MADE, BUT TO ADD TO THEM.
4. Under "User variables for ..." at the bottom, check to see if `ADMB64_HOME` is present.
   (a) If not, choose `New...`. If it is present, select `ADMB64_HOME` and choose `Edit`
   (b) "Variable name:" `ADMB64_HOME`
   (c) "Variable value:" type in the directory you chose during installation of the 64-bit version (Step 2 above).
      - For example, I typed in `C:\ADMB\ADMB10\Microsoft\64bit`.
      - If setting under "System variables," make sure `ADMB64_HOME` isn't set in "User variables for ..." as this will mask the "System variables."
5. Under "User variables for ...," choose `Path` and click on `Edit...`, or if `Path` is not present, choose `New...`.
   (a) At the end of all the text in "Variable Value" enter in `C:\ADMB\bin` if this folder is not already in the list. You may have already entered this in the list if you installed the 32-bit version.
   (b) **Make sure to keep the current text there or serious problems could occur!**
   (c) The `C:\ADMB\bin` folder is where all of the ADMB batch scripts and the tpl2cpp executable are kept. Note that I used `C:\ADMB\bin` instead of `C:\ADMB\ADMB10\Microsoft\64bit\bin`. You can enter the path to any folder, as long as it contains the batch files to do your compiling. For example, I have copied all of the items included in the installed ADMB bin folder (i.e., `C:\ADMB\ADMB10\Microsoft\64bit\bin`) to the location `C:\ADMB\bin`. This allows me to add new scripts to this folder, as we'll see next.

The additional path and library settings will be modified within the batch files. First, in the ADMB bin folder (i.e., `C:\ADMB\bin`) copy `admb.bat`, `adcomp.bat`, and `adlink.bat` to new text files and name them `admb64.bat`, `adcomp64.bat`, and `adlink64.bat` so that we can modify them and call them when compiling with 64-bit. Keep them in the ADMB/bin folder since that is already on the PATH (i.e., `C:\ADMB\bin`). These three files are described next.

**admb64.bat**

There are only two changes necessary in the `admb.bat` file. These are modifications to call adlink64.bat and adcomp64.bat on lines 37 and 42. The file should look like this:

```
@echo off
setlocal
if [%1]==[] goto HELP
if [%1]==[-help] goto HELP
if [%1]==[--help] goto HELP

rem Pop args until model=%1
set bounds=
set d=
set dll=
set g=
set r=
set s=
set tpl2cpp=tpl2cpp
set i=0
:STARTLOOP
if [%2]==[] goto ENDLOOP
if %1==-d set d=-d & set dll=-dll & shift
if %1==-g set g=-g & shift
if %1==-r set r=-r & set tpl2cpp=tpl2rem& shift
if %1==-s set s=-s & set bounds=-bounds & shift
set /a i=%i%+1
if %i%==100 shift & set i=0 & echo.&echo Warning: illegal option %1 (discarded)
goto STARTLOOP
:ENDLOOP

set model=%~n1
if not exist %model%.tpl goto ERROR1
del %model%.cpp %model%.htp %model%.obj %model%.exe 2> NUL
```

```
set CMD=%tpl2cpp% %bounds%%dll%%model%
echo.&echo *** %CMD%
%CMD%
if not exist %model%.cpp set ext=cpp& goto ERROR2
if not exist %model%.htp set ext=htp& goto ERROR2

set CMD=adcomp64 %d%%g%%r%%s%%model%
echo.&echo *** %CMD%
call %CMD%
if not exist %model%.obj set ext=obj& goto ERROR2

set CMD=adlink64 %d%%g%%r%%s%%model%
echo.&echo *** %CMD%
call %CMD%
if defined dll (if not exist %model%.dll set ext=dll& goto ERROR2)
if not defined dll (if not exist %model%.exe set ext=exe& goto ERROR2)

echo.&echo Done
goto EOF

:HELP
echo Usage: admb [-d] [-g] [-r] [-s] model
echo.
echo Build AD Model Builder executable from TPL.
echo.
echo   -d     Create DLL
echo   -g     Insert debugging symbols
echo   -r     Create ADMB-RE
echo   -s     Enforce safe bounds
echo   model  Filename prefix, e.g. simple
echo.
goto EOF
```

```
:ERROR1
echo.&echo Error: %model%.tpl not found
goto EOF

:ERROR2
echo.&echo Error: could not create %model%.%ext%
goto EOF

:EOF

REM      [2011-11-03] hicksa  modified to call 64-bit batch scripts
REM r982 [2011-02-16] arnima  rewrite, fixed bug when user option is not
REM                           recognized, fixed spaces, improved messages
REM r927 [2010-12-24] johnoel moved to 'admb' dir
REM r917 [2010-12-24] johnoel pruned 'mingw' dir
REM r914 [2010-12-24] johnoel changed script so it deletes cpp/htp/obj/exe and
REM                           reports error or success, moved to 'g++' dir
REM r629 [2010-05-20] johnoel changed .o to .obj
REM r623 [2010-05-20] johnoel changed script so it exits if tpl/cpp/htp/o/exe do
REM                           not exist
REM r525 [2009-08-07] arnima  added support for filename extension like
REM                           simple.tpl
REM                   johnoel split -s option into separate -g and -s options
REM r244 [2009-05-28] arnima  created
```

**adcomp64.bat**

There are only two changes necessary in the `adcomp.bat` file. These are

- Add path
- change ADMB_HOME to ADMB64_HOME

Modify the "SET PATH" statement to include your appropriate folder. The file should look like this (shown without the comments at the top):

```
@echo off
setlocal
if [%1]==[] goto HELP
if [%1]==[-help] goto HELP
if [%1]==[--help] goto HELP

rem Pop args until model=%1
set g=
set dll=
set opt=-DOPT_LIB
:STARTLOOP
if [%2]==[] goto ENDLOOP
if %1==-d set dll=-DBUILDING_DLL& shift
if %1==-r shift
if %1==-s set g=-g& set opt=-DSAFE_ALL& shift
goto STARTLOOP
:ENDLOOP

SET PATH=C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\BIN\amd64;%PATH%

@echo on
cl -c /EHsc -DUSE_LAPLACE -DWIN32 %opt% /Ox -D__MSVC32__=8 -I. -I"%ADMB64_HOME%"\include -I"%MSSDK%"\include %1.cpp
```

```
@echo off

goto EOF

:HELP
echo Usage: adcomp [-d] [-r] [-s] model
echo.
echo Compile AD Model Builder C++ to object code, using the MinGW GCC 3.4.5 compiler.
echo.
echo   -d     Create object file for DLL
echo   -r     Create object file for ADMB-RE
echo   -s     Create object file with safe bounds and debugging symbols
echo   model  Filename prefix, e.g. simple
echo.

:EOF
```

**adlink64.bat**

There are only three changes necessary in the `adlink.bat` file. These are

- Add to the PATH (make sure to append ;
- Add to the LIB (make sure to append ;
- change ADMB_HOME to ADMB64_HOME
- add x64 to MSSDK lib path

Modify the "SET PATH" and "SET LIB" to the appropriate folders. The file should look like this (shown without the comments at the top):

```
@echo off
setlocal
if [%1]==[] goto HELP
if [%1]==[-help] goto HELP
if [%1]==[--help] goto HELP

rem Pop args until model=%1
set re=0
set s=
set df1b2lib=df1b2stubo.lib
set adlib=ado32.lib
:STARTLOOP
if [%2]==[] goto ENDLOOP
if %1==-r set re=1& shift
if %1==-s set adlib=ads32.lib& set s=s& shift
goto STARTLOOP
:ENDLOOP

if %adlib%==ado32.lib set df1b2lib=df1b2o.lib
```

```
if %adlib%==ads32.lib set df1b2lib=df1b2s.lib
set LIBPATH_MSSDK=/libpath:"%MSSDK%"\lib

SET PATH=C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\BIN\amd64;%PATH%
SET LIB=C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\LIB\amd64;%LIB%

@echo on
cl  %1.obj %df1b2lib% admod32%s%.lib %adlib% adt32%s%.lib /link /libpath:"%ADMB64_HOME%"\lib /libpath:"%MSSDK%"\lib\x64
@echo off

goto EOF

:HELP
echo Usage: adlink [-d] [-r] [-s] model
echo.
echo Link AD Model Builder object code to executable, using the MinGW GCC 3.4.5 compiler.
echo.
echo   -d     Create DLL
echo   -r     Create ADMB-RE
echo   -s     Use safe bounds and debugging symbols
echo   model  Filename prefix, e.g. simple
echo.

:EOF
```

I have created a small registry editor file which allows you to right-click on a folder and open a command prompt set up to work in that folder. For example, I can right-click on the "simple" folder of the examples, open a command prompt and immediately compile the simple.tpl file. You can create this by adding these lines to a text file and renaming it `CmdHere.reg`

```
Windows Registry Editor Version 5.00
[HKEY_CLASSES_ROOT\Folder\shell\Cmd Here]
@="Command &Prompt Here"
[HKEY_CLASSES_ROOT\Folder\shell\Cmd Here\command]
@="cmd.exe /k pushd %L"
```

then run this file by double clicking on it. Since this modifies the registry, you must do this in an account with administrator access.

**If problems occur**

1. make sure that the directories added to the variables (i.e., PATH, LIB, ADMB_HOME, ...) are the actual directories
2. make sure that you installed ADMB 10 and Windows SDK 7.1. Other verisons are not cross-compatible.
3. if you get an error similar to
   `'admb64' is not recognized as an internal or external command, operable program or batc`
   you do not have your ADMB bin directory on the PATH. Make sure that the proper folder containing admb64.bat is in your path. See item 5 (a) in Section 3.2.
4. with 64-bit, if you get an error similar to
   `Module machine type x64 conflicts with target machine type X86`,
   you are probably compiling with the 32-bit compiler and linking 64-bit libraries, or vice versa. Check that
   `SET PATH=C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\BIN\amd64;%PATH%`
   is in the adcomp64.bat file, and that
   `SET LIB=C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\LIB\amd64;%LIB%`
   is in the adlink64.bat file.
5. e-mail me (Allan.Hicks@noaa.gov) and I can see what I can do