Configuring Eclipse for GMAT builds

Instructions for Windows Users

Rev 0.3

Darrel J. Conway*

July 12, 2007

Abstract

This document provides instructions about how to configure the Eclipse IDE to build GMAT on Windows based PCs. The current instructions are preliminary; the Windows builds using Eclipse are currently a bit crude. These instructions are intended to give you enough information to get Eclipse setup to build wxWidgets based executables in general, and GMAT in particular.

1 Install Java

Install a recent version of Java on your machine. Eclipse is written in Java, and requires that you have a JRE (Java Runtime Environment), version 1.3 or 1.4. You can get the JRE from, for example,

"http://java.sun.com/j2se/1.4.2/"

From that page, you'd select "Java 2 Platform, Standard Edition version 1.4.2" and install the downloaded executable. Once you've installed Java, check to be sure it runs by opening a command prompt and typing

```
java -version
```

You should see some text similar to this:

```
java version "1.4.2_03" Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.2_03-b02)

Java HotSpot(TM) Client VM (build 1.4.2_03-b02, mixed mode)
```

If you reach this point, proceed to the next section.

2 Install Eclipse

1. Using your favorite browser, go to http://eclipse.org/.
2. Click on the "Download" tab on the web page.
3. Select the most recent version for download (as of this writing, that is 3.2.0, selected by pressing the link in the "Download now: Eclipse SDK 3.2, Windows" text line).
4. Select a site to start the download.
5. Once the download has finished running, decompress the archive into a convenient location (I use "c:\Eclipse"). You probably should avoid paths with spaces in them.

*Senior Scientist and CEO, Thinking Systems, Inc., 6441 N Camino Libby, Tucson, AZ 85718, e-mail djc@thinksysinc.com.

6. Open a "Windows Explorer" or "My Computer" window, browse to the folder where you installed Eclipse, and double click on the file "Eclipse.exe". When you do that, Eclipse will start up and ask for the location of your default workspace. Select a folder that you want to use to contain your Eclipse projects.

If you get this far, you're ready to start setting up the C++ environment.

3 Install a gcc Compiler

Install gcc for Windows, either cygwin or MinGW (http://www.mingw.org/). My current preference for this step is to use the MinGW installation from the wxWidgets book[1]. For MinGW, you'll need to install both the MSYS shell and the GCC compiler suite[1], including the Fortran compiler. Install MinGW before installing MSYS.

For Cygwin, you'll need to select from a pretty extensive shopping list of options. If you have trouble figuring out what to pick, contact a member of the GMAT development team for assistance.

Both MinGW and Cygwin include a version of the bash shell configured to run under Windows. When you installed the environment, an icon for the terminal interface to this shell should have been placed on your Windows desktop. Open this shell and type

```
g++ -v
```

When you do this, you should receive version information about the Gnu C++ compiler, similar to this:

```
Reading specs from c:/mingw/bin/../lib/gcc/mingw32/3.4.2/specs
Configured with:  ../gcc/configure -with-gcc -with-gnu-ld -with-gnu-as
-host=mingw32 -target=mingw32 -prefix=/mingw -enable-threads -disable-nls
-enable-languages=c,c++,f77, ada,objc,java -disable-win32-registry
-disable-shared -enable-sjlj-exceptions -enable-libgcj -disable-java-awt
-without-x -enable-java-gc=boehm -disable-libgcj-debug -enable-interpreter
-enable-hash-synchronization -enable-libstdcxx-debug
Thread model:  win32
gcc version 3.4.2 (mingw-special)
```

After you install the compiler, check to be sure it works by building a "Hello, world" program. If you use C++ library calls (e.g. std::cout), be sure you compile using g++, not gcc, so that the correct libraries get linked.

4 Install the CDT in Eclipse

This is the last step in the generic configuration.

1. Start Eclipse.
2. From the menu bar, select Help, then Software Updates -> Find and Install
3. Select "Search for New Features to Install" from the resulting menu.
4. Select "Callisto Discovery Site".[2]
5. Select the most recent version of the CDT (currently 3.1).
6. Run the installation. You'll need to accept licensing agreements (they are similar to the GPL).
7. Test it with a Hello World program.

Once you're this far, you can start on the GMAT specifics.

[1] See Appendix 2 for information about updating the gcc compiler suite in MinGW to obtain more optimized performance for GMAT.

[2] The actual selection here depends on the release of Eclipse that you installed above. Callisto is the name of the coordinated Eclipse 3.2 release package.

5 Install wxWidgets

In order to build the GMAT user interface, you'll need to set up the wxWidgets framework. wxWidgets is build from source, using the compiler you just finished configuring. The wxWidgets source is available for download from http://wxwidgets.org/. Download the latest version of wxWidgets (2.6.3 at this writing) and install it on your system. Then follow these steps:

1. Open the bash shell that you installed in step 3.
2. Change directories to the folder containing wxWidgets.
3. Setup a build directory, and change directories to that folder:
   ```
   mkdir installation
   cd installation
   ```
4. Configure the framework:[3]
   ```
   $ configure --with-opengl --disable-shared --enable-optimise
   ```
5. Read the summary information after configuring to be sure that the correct operating system and options were set.
6. Build wxWidgets:
   ```
   $ make
   ```
7. Install it:
   ```
   $ make install
   ```

Once you have completed these steps successfully, you are ready to check out the GMAT code and build it. (You can validate your wx installation by making any of the sample programs, running `make` on the files in the folders under `samples` in your configuration folder.)

6 Check Out GMAT under Eclipse

The easiest way to configure Eclipse to build GMAT is to create a GMAT project in Eclipse, and check out the GMAT build folders into this project. This is a somewhat involved procedure, but will simplify things later. Here is the procedure:

1. Start Eclipse.
2. Select `File | New > Standard Make C++ Project`. This will open a panel used to configure the project.
3. Set the project name to be GMAT. You can also set the folder to something other than the default location by unchecking the "Use Default" box on the panel and selecting a different folder if you'd like. Then click on the `Next` button.
4. The panel that opens here is a tabbed panel. We can accept most of the default settings. Change the following elements:
   (a) On the `Make Builder` tab, deselect `Use default` and set the make command to read `make -f Makefile.Eclipse`.
   (b) On the `C/C++ Indexer` tab, change the indexer setting to `CTags Indexer (declarations only)`.

[3]You may also want to add the --with-msw to specify a Microsoft Windows build; the configure script will detect the operating system you are using when building, so this option is not required unless you are building libraries for a different operating system from the one you are using.

5. Press `Finish` to create the project.
6. Select the `Project` menu item, and turn off the `Build Automatically` option. It's really pretty useless for C++ code.
7. Select `File | Import` from the menu bar. This opens the import wizard.
8. Select `Checkout Projects from CVS`, and press the `Next` button to open the CVS settings panel.
9. Set the CVS options you use (mine are shown in 1) and press `Next`.
10. Select the `Use an Existing Module` radio button to get the list of CVS modules available for use from the repository. Select `base` from the list, and then press the `Next` button (**NOT Finish!!!**).
11. Select the `Check out into an existing project` radio button and press `Next`.
12. Select the GMAT project, and press `Next`.
13. Press the `Finish` button to retrieve the base files.
14. Repeat steps 7 through 13, using the `gui` selection from the list in step 10.
15. (Do this from the Windows file manager rather than from Eclipse.) Put the DevIL libraries in place. You'll need to make a folder in your GMAT project folder named DevIL, and copy DevIL.dll, ILU.dll, and ILUT.dll into it. When you do this, the configured Makefiles will be able to find the image library files.
16. Edit the eclipseWin makefiles (`MakeBase.eclipseWin` and `MakeGui.eclipseWin`) to use your path information for the MATLAB files. (I need to add targets to the make files for non-MATLAB builds.)
17. Setup targets to build the base and gui components:

    (a) Configuring base targets

    - i. Right click in the `base` node in the Eclipse Project or Navigator tree, and select `Create Make Target...`
    - ii. Change the target name to `Base4Windows`
    - iii. Change the Build command to `make -f MakeBase.eclipseWin`
    - iv. Turn off `Run all project builders`
    - v. Press the Create button
    - vi. Right click in the `base` node in the Eclipse Project or Navigator tree, and select `Create Make Target...`
    - vii. Change the target name to `Rebuild Base4Windows`
    - viii. Set the to Make target to `clean all`
    - ix. Change the Build command to `make -f MakeBase.eclipseWin`
    - x. Turn off `Run all project builders`
    - xi. Press the Create button.

    (b) Configuring gui targets

    - i. Right click in the gui node in the Eclipse Project or Navigator tree, and select `Create Make Target...`
    - ii. Change the target name to `Gui4Windows`
    - iii. Change the Build command to `make -f MakeGui.eclipseWin`
    - iv. Turn off `Run all project builders`
    - v. Press the Create button
    - vi. Right click in the `base` node in the Eclipse Project or Navigator tree, and select `Create Make Target...`

vii. Change the target name to `Rebuild Gui4Windows`
viii. Set the to Make target to `clean all`
ix. Change the Build command to `make -f MakeGui.eclipseWin`
x. Turn off `Run all project builders`
xi. Press the Create button.

You are now ready – finally! – to build GMAT.

7 Build GMAT

GMAT is built in 2 steps – first the base library is built, then the GUI is built and linked to the base library.

1. Right click on the `base` node in the Eclipse Project or Navigator tree, and select `Build Make Target...`
2. Select `Rebuild Base4Windows`, and press `Build`.
3. Go get an ice cream cone – this will take a few minutes.
4. Right click on the `gui` node in the Eclipse Project or Navigator tree, and select `Build Make Target...`
5. Select `Rebuild Gui4Windows`, and press `Build`.
6. Eat your ice cream.

If everything is configured correctly, GMAT will build an executable and you'll be ready to start flying spacecraft! (Well, okay, you'll need to put the startup file in place, find the other support files, whine about things a while, and other trivial matters, but you get the idea.)

Note: Eclipse can be configured to simplify the build procedure. See the Appendix for a description of the project level make file and configuration file.

8 Using CVS from Eclipse

Assuming you followed all of these steps successfully, you'll be able to use CVS pretty efficiently from inside of the Eclipse UI. To get a taste of what you can do from Eclipse, open a file in the Eclipse editor. Make a change – just add a blank line or something. Now save the file, and look at its node on the Eclipse tree. It should have a '>' marker indicating that the file changed. The folder that contains that file also has a marker. as does the project.

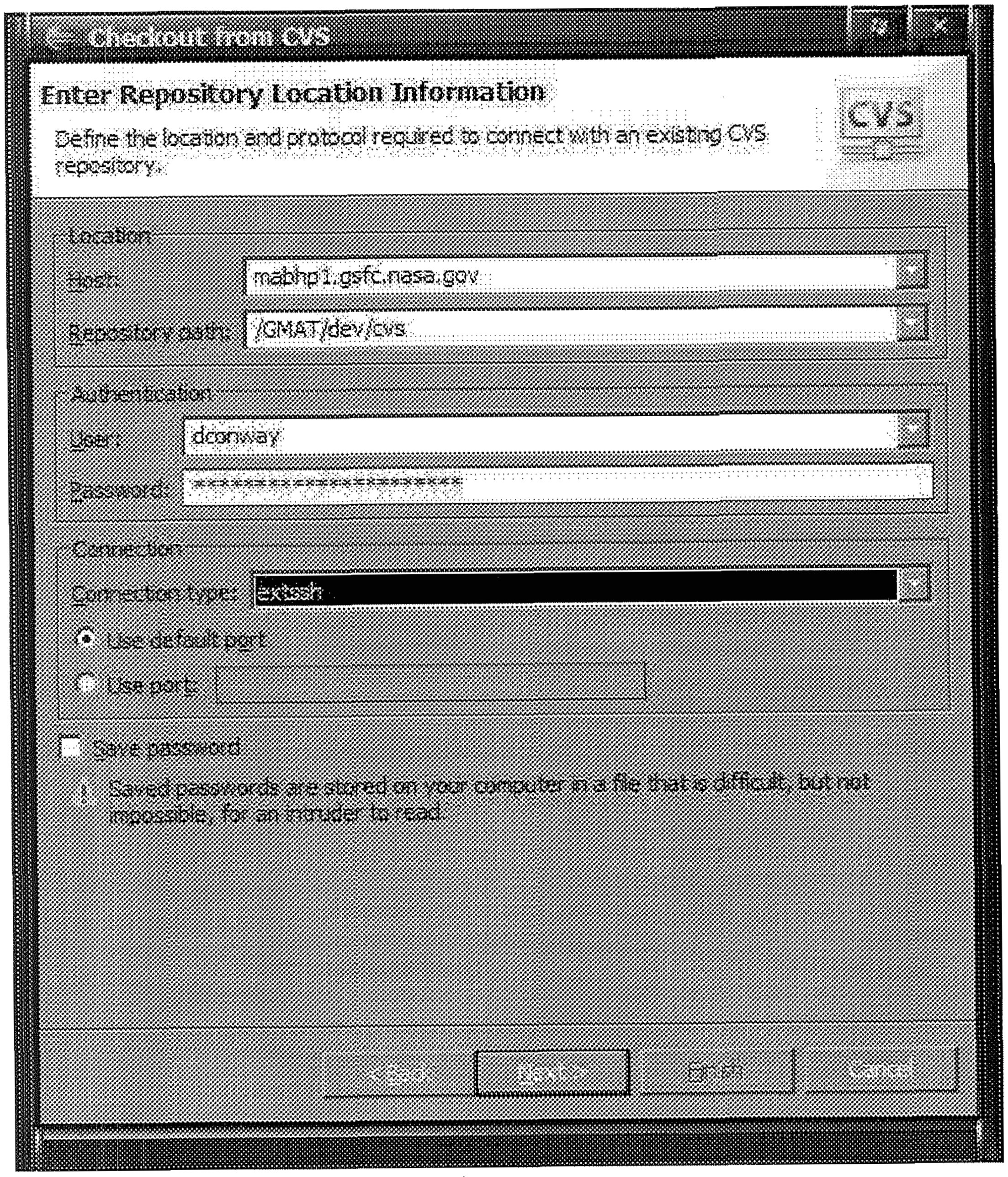
Now select the GMAT node, and right click on it. Select the `Team > Synchronize with Repository` entry. Tell Eclipse to switch the Perspective when it asks. If you expand the nodes in the Synchronize panel, the new arrangement will show you all of the changes in the project (including any that need to be downloaded to your machine from CVS). If you double click on a selected file, you'll get a side-by-side view of the files with the changes highlighted. You can right click on nodes to update or commit changes, and to do many other common CVS tasks.

You can switch back to the C++ Perspective using the '>>' marker in the upper right corner of the Eclipse UI, or using the `Window | Open Perspective >` menu entry.

Appendix 1: Project Level Settings

The Eclipse make files contained in the CVS repository require the creation of a support file, `BuildEnv.mk`, to set configuration parameters. The current version of this file is presented here:

```
# Flags used to control the build
USE_MATLAB = 1
USE_DEVIL = 1
```



Enter Repository Location Information

Define the location and protocol required to connect with an existing CVS repository.

Location

Host: mabhp1.gsfc.nasa.gov

Repository path: /GMAT/dev/cvs

Authentication

User: dconway

Password: **********************

Connection

Connection type: extssh

(●) Use default port

( ) Use port:

☐ Save password

Saved passwords are stored on your computer in a file that is difficult, but not impossible, for an intruder to read.

Figure 1: CVS Settings

```
# MATLAB specific data

MATLAB_INCLUDE = -IC:/Program\ Files/MATLAB71/extern/include

MATLAB_LIB = -LC:/Program\ Files/MATLAB71/bin/win32

MATLAB_LIBRARIES = -leng -lmx -lmat


# Compiler options
CPP = g++
C = gcc
FORTRAN = g77

PROFILE_FLAGS = -pg
OPTIMIZATIONS = -O2

# Do not edit below this line -- her we build up longer compile/link
strings

# Build specific flags
CONSOLE_FLAGS = -D__CONSOLE_APP__
MATLAB_FLAGS = -D__USE_MATLAB__=1

# Set options for debugging and profiling
DEBUG_FLAGS =

# Build the complete list of flags for the compilers

ifeq ($(USE_MATLAB),1)

CPPFLAGS = $(OPTIMIZATIONS) -Wall $(MATLAB_FLAGS) \
           '/usr/local/bin/wx-config --cppflags' \
           $(MATLAB_INCLUDE)
else

CPPFLAGS = $(OPTIMIZATIONS) -Wall \
           '/usr/local/bin/wx-config --cppflags'

endif

F77_FLAGS = $(CPPFLAGS)

# Link specific flags
ifeq ($(USE_DEVIL),1)

ifeq ($(USE_MATLAB),1)

LINK_FLAGS = '/usr/local/bin/wx-config --libs --static=yes'\
             $(MATLAB_LIB) $(MATLAB_LIBRARIES) \
             -lwx_msw_gl-2.6 -lopengl32 -lglu32 \
             -lg2c $(DEBUG_FLAGS) \
             -L../devIL -lilu -lilut -lDevIL
```

```
else

LINK_FLAGS = '/usr/local/bin/wx-config --libs --static=yes'\
             -lwx_msw_gl-2.6 -lopengl32 -lglu32 \
             -lg2c $(DEBUG_FLAGS) \
             -L../devIL -lilu -lilut -lDevIL
endif

else

ifeq ($(USE_MATLAB),1)

LINK_FLAGS = '/usr/local/bin/wx-config --libs --static=yes'\
             $(MATLAB_LIB) $(MATLAB_LIBRARIES) \
             -lwx_msw_gl-2.6 -lopengl32 -lglu32 \
             -lg2c $(DEBUG_FLAGS)

else

LINK_FLAGS = '/usr/local/bin/wx-config --libs --static=yes'\
             -lwx_msw_gl-2.6 -lopengl32 -lglu32 \
             -lg2c $(DEBUG_FLAGS)

endif

endif
```

In addition, you may want to create a project level make file so that Eclipse can clean, make, and build the project without using specific make targets. If you followed the configuration instructions presented here, you will need a file named `MakeGmat.EclipseWin` at the project level. The content of this file should be similar to this:

```
# Windows makefile for GMAT, using Eclipse

TARGET = bin/gmat.exe
LIBRARY = base/lib/libGmatGui.a

all: $(TARGET)

clean:
    cd base; make -f MakeBase.eclipseWin clean
    cd gui; make -f MakeGui.eclipseWin clean
    rm -rf $(TARGET) $(LIBRARY) gui/gmat.exe

base: $(LIBRARY)

bin/gmat.exe: base/lib/libGmatGui.a gui/gmat.exe

gui/gmat.exe:
    cd base; make -f MakeBase.eclipseWin all
    cd gui; make -f MakeGui.eclipseWin all

base/lib/libGmatGui.a:
    cd base; make -f MakeBase.eclipseWin all
```

Appendix 2: Updating GCC for MinGW

The MinGW compiler shipped with the wxWidgets book at this writing is gcc 3.2.3. When performance is an issue for GMAT builds, we recommend that you use a later compiler, because the code optimization in the later compiler releases is much more complete than in 3.2.3. Here you will find instructions for updating a MinGW/MSys installation, specifically from gcc 3.2.3 to gcc 3.4.2, though the instructions here are appropriate for other versions of gcc if you replace the version numbers with the version you are installing.

Begin by downloading the compiler updates. Using a web browser, go to the URL

```
http://www.mingw.org/download.shtml
```

Download the core, g++, and g77 components of the compiler suite for the version you want to install. (These are the only components required to build GMAT; you may also want to install or update other pieces as well.) If, for example, you are upgrading to gcc 3.4.2, these packages are named:

- `gcc-core-3.4.2-20040916-1.tar.gz`
- `gcc-g++-3.4.2-20040916-1.tar.gz`
- `gcc-g77-3.4.2-20040916-1.tar.gz`

(The latter part of the name will change if you are upgrading to a later version of the compilers.) Place these files in a convenient location in your MinGW file system; for the purposes of these instructions, I've placed the files in {MinGW}\temp, where {MinGW} is the /mingw folder in msys. (This is usually C:\MinGW, unless you installed MinGW to a different location).

Open an msys window, and type the following commands, replacing the "temp/" folder with the name of the folder containing the downloads itemized above.

```
tar -zxvf temp/gcc-core-3.4.2-20040916-1.tar.gz
tar -zxvf temp/gcc-g++-3.4.2-20040916-1.tar.gz
tar -zxvf temp/gcc-g77-3.4.2-20040916-1.tar.gz
```

Verify that the new versions of the compilers are installed by running the commands `gcc -v`, `g++ -v`, and `g77 -v`. The resulting version strings should list the compiler version just installed.

Now rebuild the wxWidgets library, using optimization flags. If you have wxWidgets 2.6.2, installed in the folder C:\wxWidgets-2.6.2, you will perform this task using the instructions listed here (tailor the path and wxWidgets version to your installation):

```
cd /c/wxWidgets-2.6.2/
mkdir optimizedBuild
cd optimizedBuild/
../configure --with-opengl --disable-shared --enable-optimise
make BUILD=release
make install
```
[4]

Now clean and rebuild GMAT. The resulting executable will be more responsive.

References

[1] J. Smart and K. Hock, **Cross-Platform GUI Programming with wxWidgets**, Prentice Hall, 2005.

[4] You may need to remove the previous install of wxWidgets to avoid multiple definitions during link. If so, from the msys window, type this command: `rm -rf /usr/local/*/*wx*` and rerun `make install`