Using Mac F2C With CodeWarrior C/C++

Before you can use the code produced by Mac F2C, you must set up and build all of the required support libraries. There are also special rules that must be followed when using code produced by Mac F2C. The process and rules are slightly different for each compiler. The instructions in this chapter are for setting up Mac F2C for use with Metrowerks CodeWarrior C/C++. Refer to other chapters for instructions on how to set things up for use with Symantec C/C++ or THINK C/C++.

Users upgrading to version 1.3.5 need to re-install all libraries, project files, stationary, and supporting files. All files, including library source code and project files, have been updated in version 1.3.5 to make Mac F2C compatible with IDE v2 the Metrowerks Standard Library version 2.1.2 (CW Pro 1).

Please note that the source code and libraries provided are set up for use with CodeWarrior IDE v2

If you have a newer version of CW, first check if there is a more recent version of Mac F2C. If not, convert all the CW project files contained in the Mac F2C package by simply opening and closing themwith the new version of CW (answer "OK" when asked if you want to update the project file). The CW project files you must convert are located in the Test Project ƒ and Mac F2C Libraries folders. Then run the installer. If this does not work (I obviously can't guarantee it will), let me know.

Setting Up Mac F2C Using the Installer

The easiest way to set up Mac F2C is to use the installer included with Mac F2C. This installer will only work correctly if you have System 7.5 (or higher) and have CodeWarrior IDE v2 using the Metrowerks Standard [ANSI] Library. If you do not meet both of these requirements, please follow the instructions for manual installation found in the following section.

The installer is stand-alone AppleScript application called Mac F2C Installer. To run the installer, simply double click on it, and answer the dialogs it presents. The installer will do the following:

- Create a folder called Mac F2C Support within CodeWarrior's MacOS Support folder.
- Build the appropriate Mac F2C libraries.
- Move the built libraries to the Mac F2C Support folder.

• Copy the other support files (found in the For 'Mac F2C Support' folder: F2Cmain.c, F2Cmain.cp, f2c.h, and F2C Cursors.rsrc) to the Mac F2C Support folder.
• Copy the Mac F2C project stationary files to CodeWarrior's (Project Stationery) folder, creating a Mac F2C folder there too.
• Translate and build the appropriate versions of the test application.

When the installer is finished, you will find completed test applications in the Test Project ƒ folder. You should run these to verify correct operation of Mac F2C and its libraries. After that, you are ready to go. To compile translated FORTRAN code, simply open a new project in CodeWarrior, select the appropriate version of Mac F2C stationary, and add the translated files.

The remaining sections provide step-by-step instructions for installing Mac F2C manually (including how to install for earlier CodeWarrior versions), a more detailed description of how to test your Mac F2C installation, and additional information on using code generated by Mac F2C with CodeWarrior.

Setting Up Mac F2C Manually

Set up is a two step process. First you must build the libraries, and then you must move several files to their proper locations.

Step 1: Build all the libraries

The libraries of the Mac F2C distribution come without binaries, so you have to build them according to the following algorithm. There is one version of each library project. Each project file has three targets: one to make a 68K library, one to make a PPC library, and one to make both.

FOR the project files in the Mac F2C Libraries folder:
(1) libI77.µ
(2) libF77.µ
REPEAT the following steps:
(a) Double-click on the project file.
(b) Select the desired target (68K, PPC, or Both)
(c) In the CW Environment's Project menu,
select the Make command.
END REPEAT

Step 2: Move things to the recommended locations

For easiest and smoothest operation, the support libraries and other support files should be installed where the CodeWarrior C compiler can find them easily. I recommend you install Mac F2C libraries and support files as follows:

• Create a folder called Mac F2C Support inside the MacOS Support folder within the folder that contains the CodeWarrior application. All of the libraries and files required to support Mac F2C code will be placed in this folder.

• Drag the libraries libI77.68K, libF77.68K, libI77.PPC, and libF77.PPC you obtained by following the instructions of Step 1 from the folder Mac F2C Libraries to the Mac F2C Support folder that you just created.

WARNING: Do not place the source code for these libraries (found in the folders libF77 Sources and libI77 Sources) in the folder containing the CodeWarrior application or in any of its sub-folders). The source code for these libraries has name conflicts with Apple's Universal Headers (e.g., a file called fp.h appears in both, but the two are not equivalent files). Otherwise any of your code that #includes any of the conflicted files may inadvertently access the wrong file.

• The For 'Mac F2C Support' folder located in the CodeWarrior Support folder contains files (f2c.h, F2Cmain.c, F2Cmain.cp, and F2Ccursors.rsrc) that are needed to compile C programs produced by Mac F2C. Drag these files to to the Mac F2C Support folder you created earlier inside the MacOS Support folder.

• The For '(Project Stationary)' located in the CodeWarrior Support folder contains project stationary files that are used for compiling C and C++ code produced by Mac F2C. Open the For 'MacOS' folder located inside the For '(Project Stationary)' folder and find a folder called Mac F2C. Drag the Mac F2C folder into the MacOS folder located inside CodeWarrior's (Project Stationary) folder.

Verifying Correct Operation of Mac F2C

The folder Test Project ƒ contains the following files relevant to CodeWarrior:

test.f -- a sample FORTRAN program.

F2Cmain.c -- the main program required to run programs produced by Mac F2C.

F2Cmain.cp -- the main program required to run programs produced by Mac F2C.

f2c.h -- an include file required to compile programs produced by Mac F2C.

test.c (C Output) -- what you should get when you translate the sample FORTRAN code files.

Test.μ -- a CodeWarrior project to create run the sample program. It has targets to build 68K and PPC versions. It serves as a model for how to compile, link, and run C code produced by Mac F2C.

test.cp (C++ Output) -- what you should get when you translate the sample FORTRAN code files and select the C++ output option.

Test++.μ -- a CodeWarrior project to run the sample program when Mac F2C is used to produce C++ output. It has targets to build 68K and PPC versions. It serves as a model for how to compile, link, and run C++ code produced by Mac F2C.

Translate the sample FORTRAN program Test.f simply by dragging it onto Mac F2C. Do not change any of the options (use Factory Defaults). Once you have done this you can compare it with Test.c (C Output) file to verify that you got the same thing. If so, double click on the CodeWarrior project Test.μ and run it to verify correct operation.

If you also plan to use Mac F2C C++ output with the Codewarrior, you can run a second test to verify correct operation with the C++ compiler. Start Mac F2C and in the C Options dialog, select C++ code. Do not change any of the other options. Translate Test.f. Compare it with Test.cp (C++ Output) to verify that you got the same thing. If so, double click on the CodeWarrior project Test++.μ and run it to verify correct operation.

Installing MrC Support

MrC is a PowerPC-only compiler (it only generates PowerPC code and does not run on 68K Macs). MrC support for Mac F2C is only available on PowerPC computers. Mac F2C v1.3.5 includes incomplete support for the MrC plug-in compiler for CodeWarrior. The MrC project files included with v1.3.5 are

compatible with the older IDE v1.7.4, not with IDE v2. Because I do not have MrC (or a PowerPC), I haven't been able to convert these to IDE v2.

If you have MrC, you should be able to update the supplied MrC project files to IDE v2 format using the Metrowerks supplied conversion tool. You can then install MrC support manually, simply by following the instructions above for manual installation, but in each step use the MrC versions of the CodeWarrior project files and stationary files. The MrC versions of the project files are clearly identified by a .MrC. suffix.

Using C Code Generated by Mac F2C

The C code produced by Mac F2C has the following compile and link requirements:

68K version:

- the header file:
  f2c.h
- the F2C libraries:

libI77.68K

libF77.68K

- the resource file:
  F2C Cursors.rsrc
- the CodeWarrior libraries:

MSL C.68K Far(4i_8d).Lib
MSL SIOUX.68K.Lib
MSL Runtime68K.Lib
MathLib68K Fa(4i_8d).Lib
MacOS.lib

- for C++ code only, the CodeWarrior libraries:
  MSL C++.68k Far(4i_8d).Lib
- 4-byte integers
- 8-byte doubles
- Far Data
- Smart code model

PPC version:

- the header file:
  f2c.h
- the F2C libraries:
  libI77.PPC
  libF77.PPC
- the resource file:
  F2C Cursors.rsrc
- the CodeWarrior libraries:
  MSL C.PPC.Lib
  MSL SIOUX.PPC.Lib
  MSL Runtime.Lib
  Interface.Lib
  MathLib
- for C++ code only, the CodeWarrior library:
  MSL C++.PPC.Lib

In addition, if you compile a stand-alone FORTRAN program (instead of only some FORTRAN subroutines) you must include F2Cmain.c in your project (or F2Cmain.cp if you use C++; the two files are identical). This is because the original main routine in the FORTRAN program becomes a function that is called by F2Cmain.c. In addition, F2Cmain.c performs a series of initializations (primarily related to error catching) prior to executing the main FORTRAN program.

The For '(Project Stationary)' and For 'Mac F2C Support' folders provided in the CodeWarrior Support folder contain everything you need to compile and run code produced by Mac F2C. You can install them following the instructions above.

To start a new project using Mac F2C code, launch the CodeWarrior application,

select New Project..., and then choose the appropriate Mac F2C project stationary in the resulting Standard File dialog. Then add your code files to the project and bring everything up-to-date. Use the "C" versions of project stationary to work with C code generated by Mac F2C and the "C++" versions of project stationary to work with C++ code generated by Mac F2C.

If you compile a FORTRAN subroutine or function that you want to call from a C program, look at the output C code to see the appropriate calling protocol. You may or may not need to include the F2C support libraries (libF77 and libI77). In rare cases, you may also need to copy some of the initialization code from F2Cmain.c to your calling program.

Special 68K Considerations

Please read the following section carefully if you intend to use Mac F2C with the CodeWarrior 68K compiler (the PPC compiler doesn't give you any of the options mentioned, so there is nothing to consider):

As noted above, code produced by Mac F2C MUST be compiled with 4-byte integers. This requirement cannot be relaxed. The other requirements (8-byte doubles, far data) can sometimes be relaxed.

IF you do not use doubles in any situation where their size relative to reals or integers matters (e.g., if you do not use doubles in equivalence and common statements), then your code probably does not require 8-byte doubles. You need to verify this on a case-by-case basis.

This requirement exists because Mac F2C follows FORTRAN sizing rules when compiling FORTRAN code: sizeof(real) == sizeof(integer) and sizeof(double) == 2*sizeof(real). FORTRAN real is translated as C float and FORTRAN double as C double, so doubles have to be 8-bytes long for equivalence and common statements to be properly aligned. There are a few other cases where the size of double variables matters; see AT&T Computing Science Technical Report No. 149 (included with Mac F2C) for a detailed discussion.

IF you compile your program with the option Local variables are automatic and you do not have large static data structures, you might not need Far Data. You need to verify this on a case-by-case basis.

Mac F2C creates large static data structures for I/O. If you create local variables

in the global area (static instead of automatic) or if you have other static data, you will almost certainly require Far Data. The I/O data structures can be large enough that you may require Far Data for that reason alone.

The 68K project files are all set to use the Smart code model. That means, the compiler generates a combination of the far and near (32 bit and 16 bit) addressing types, using near when possible. Segments are not limited to 32K of object code.

I suggest to stick with this option, since the Large code model doesn't usually give you advantages over Smart, and Small requires a jump table for inter-segment jumps which has a negative effect on code size and speed. But you can try and put all files into one segment if your program is small enough and doesn't use much of the library code. You need to verify this on a case-by-case basis. If you are working with very large FORTRAN programs, you may get a link error from Codewarrior saying something to the effect that 16-bit offsets aren't enough to reach some function or other. In this case you can simply select the Large code model to solve the problem.

If you change the 8-byte doubles, Smart, Far Data, or 68881 options, remember to also change them in all the Mac F2C libraries, specifically libI77.68K and libF77.68K, and include the appropriate ANSI library. The ANSI library normally used in the F2C project files is MSL C.68K Far(4i_8d).68K.Lib (a version of the Metrowerks Standard Library). The 'Far' stands for 'Far Model' and the arguments inside the parentheses show the compiler options (4-byte integers/8-byte doubles). For example, if you don't need 8-byte doubles, use MSL C.68K Far(4i).Lib.

I urge all users to read the enclosed AT&T Computing Science Technical Report No. 149. Consider it your compiler and language reference manual.