Using F2C for MPW

Thank you for your interest in F2C for MPW. This is the first full public release of F2C for MPW. This package includes an MPW-tool version of F2C and support for generating FORTRAN-based MPW-tools using F2C. The F2C for MPW package is integrated with Mac F2C thanks to the generous efforts of Kris L. Jorgensen.

If you have used previous versions of F2C for MPW you will need to re-install the complete F2C for MPW package. Previous libraries and tools are incompatible with the current version. F2C for MPW is a FAT tool that will run native on both the 68K Macintosh and the PowerMac. You will need MPW 3.4 or higher to run native on the PowerMac.

Setting up F2C for MPW

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

Step 1: Build and Install 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 are two versions of each library, one for the 68K compiler and one for the PowerPC compiler. They are named with the usual extensions, 68K and PPC.

After starting MPW, change directories to the `Mac F2C Libraries` directory within the F2C distribution. Execute one of the following commands from MPW to install the desired libraries. This will install the libraries in the correct location within the MPW directory tree. This will create a directory called `F2CLibraries` within the `Libraries` directory in the MPW environment.

• To install both PPC and 68K libraries type:

```
BuildProgram install
```

• To install PPC libraries only type:

```
BuildProgram installPPC
```

• To install 68K libraries only type:

```
   BuildProgram install68K
```

If you want to install the libraries in another location you will need to modify the makefile to point to the new location. You will notice that the makefile actually calls the other four makefiles with in the directory. Make sure that all four are together or this will not work.

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 MPW can find them easily. The libraries should already be in the correct location. Next there are two other files you will need to install.

• Copy the F2C tool to the Tools folder in the MPW directory.
• Copy the UserStartup.F2C to the MPW directory.
• Copy the F2C.Help to the MPW directory.
• Quit and restart MPW.

If you installed the libraries in a custom location you will need to modify the UserStartup.F2C to reflect the new location. Also, restarting is important since UserStartup.F2C sets a convenience variable that is used in the makefile for the test project and that you can also use in your projects.

Verifying Correct Operation of F2C for MPW

This is simple test tool that exercises the I/O and math functions in the libraries. The makefile will create an MPW tool called test.ppc or test.68k that you can then run.

To build one of the tools change to the Test Project ƒ directory from within MPW and then type one of the following lines:

```
BuildProgram test.ppc
BuildProgram test.68k
```

To run the tool you just created type either:

```
test.ppc
test.68k
```

Using C Code Generated by F2C for MPW

I recommend that you start out using the makefiles included with the test project as the basis for your own projects until you are familiar with F2C. You can refer to them to see which libraries must be linked in. Other requirements are summarized here.

68K version:

- the header file:
    `f2c.h`
- the F2C libraries:
    `"F2CLibraries"libI77.PPC`
    `"F2CLibraries"libF77.PPC`
- the CodeWarrior libraries:
    `"MW68KLibrariesMPW ANSI (4i/8d) C.68K.Lib"`
    `"MW68KLibrariesMacOS.Lib"`
    `"MW68KLibrariesMathLib68K (4i/8d).Lib"`
    `"MW68KLibrariesToolLibs.o.lib"`
- for C++ code only, the CodeWarrior libraries:
    `ANSI (4i/8d) C++.68K.Lib`
    `CPlusPlus.Lib`
- 4-byte integers
- 8-byte doubles
- Far Data
- Smart code model

PPC version:

- the header file:
    `f2c.h`
- the F2C libraries:
    `"F2CLibraries"libI77.PPC`
    `"F2CLibraries"libF77.PPC`
- the CodeWarrior libraries:
    `"MWPPCLibraries"MWStdCRuntime.Lib`
    `"MWPPCLibraries"StdCLib`
    `"MWPPCLibraries"InterfaceLib`
    `"MWPPCLibrariesPPCToolLibs.o"`
    `"MWPPCLibraries"MathLib`

In addition, if you compile a stand-alone FORTRAN program (instead of only some FORTRAN subroutines) you must include F2Cmain.c in your project. 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 prior to executing the main FORTRAN program.

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.

If your code uses common blocks look in the makefile of the test project for instructions on how to handle the common blocks within your FORTRAN code. It's not pretty but it solves the problem!

MPW tools created with the F2C for MPW libraries provide cooperative multitasking for your tools. This has been coupled with the I/O of the library. You will notice that while your program is performing io the cursor will be a spinning beach ball. This allows other programs to have some of the CPU and allows your tool to be put in the background. If you have some computationally intensive code that does not perform I/O you can call the MPW SpinCursor routine explicitly to free up the CPU for other programs. The documentation for this routine is provided with MPW. To call it from FORTRAN use the something like this:

```
      CALL SPINCURSOR(N)
```

where N is the number of ticks (60ths of a second) that you are willing to let your program sleep while other programs execute. You can also use the same subroutine call as for Symantec, THINK, and CodeWarrior, namely:

```
      CALL DOMULTITASK(N)
```

In the MPW case, the call to DOMULTITASK will simply call through to SPINCURSOR.

Special 68K Considerations

Please read the following section carefully if you intend to use Mac F2C with the

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 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 compiled 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.

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 MPW ANSI library. The standard ANSI library used in the F2C project files is `MPW ANSI (4i/8d) C.68K.Lib`. 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 `MPW ANSI (4i) C.68K.Lib`, if you want direct 68881 support, use `MPW ANSI (4i/F/8d) C.68K.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. You can print the report by downloading it to any PostScript printer. You can use Apple's

LaserWriter Utility application to do this or you can use any of the many equivalent utilities.