

## A.3 Using Metrowerks Codewarrior

Codewarrior is an IDE for developing C++ and Java programs on three platforms: Windows 95/98/NT, MacOS, and Linux. The Linux IDE is an environment built around the GNU/egcs g++ compiler. The discussion and screen shots in this document are based on the Windows version of Codewarrior, but should apply to some degree to the other platforms.

Creating a program in Codewarrior requires three steps, each will be discussed in some detail.

- Creating a project for the program.
- Adding source files and libraries to the project.
- Building and running the program.

I typically create one project, then add and remove source files from it each time I write a program. Since I link the same libraries with almost all the programs I write, and since I use the same settings for the IDE for each program, reusing one project makes sense. It's very simple to remove source files and add new ones as we'll see.

### A.3.1 Codewarrior Projects

If you're creating a program for the first time in Codewarrior you'll need to create a project. Choose the *File*→*New Project* option from the **File** menu, then click on the + to expand, in turn, *Win32-x86*, *C,C++*, and *C++ Console App* as shown in Fig. A.1.

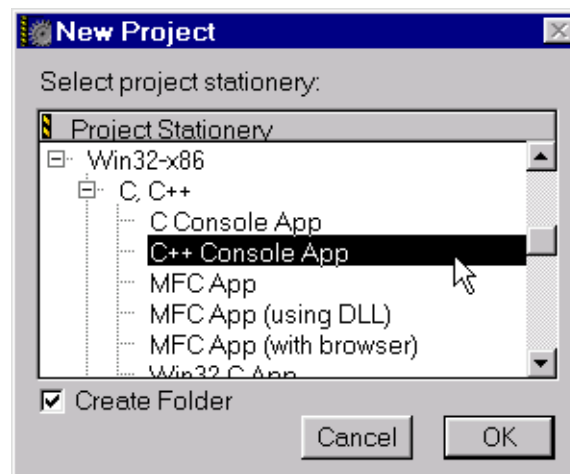


Figure A.1 Creating a new project using Metrowerks Codewarrior

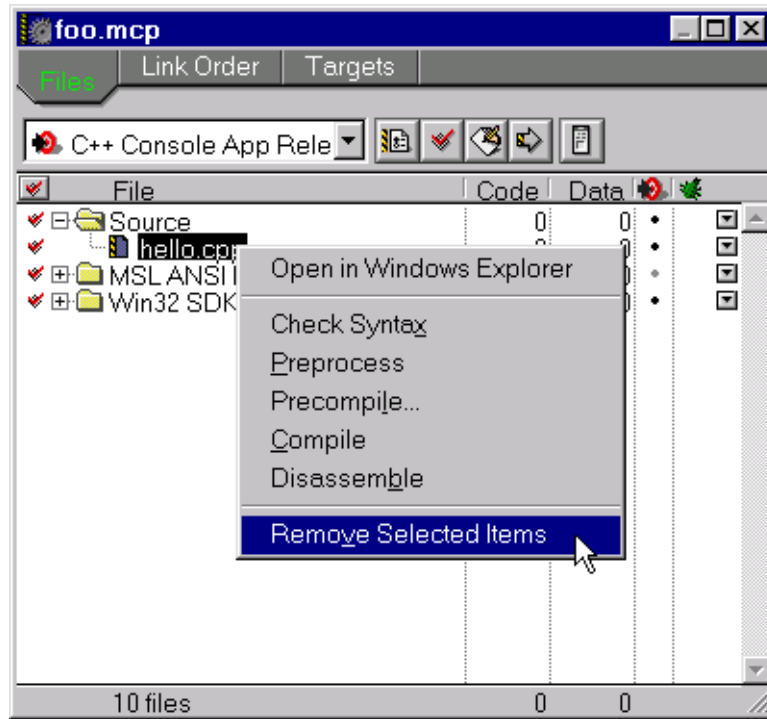


Figure A.2 Removing a file from a Codewarrior project.

You'll need to name the project and browse to create a folder for it. Codewarrior projects typically have a *.mcp* suffix (Metrowerks Codewarrior Project), which is added automatically if you don't use a suffix when providing a name. This default project contains one source file, *hello.cpp*. The first thing I typically do is remove this source file and add my own.

**Adding and Removing Files.** There are two ways to remove a file from a project.

- Select the file you want to remove by clicking with the left mouse button so the file is highlighted. Then from the **Project** menu choose *Project* → *Remove Selected Items*.
- Select the file you want to remove by clicking with the right mouse button, then from the pop-up menu choose *Remove Selected Items*. This process is shown in Fig. A.2.

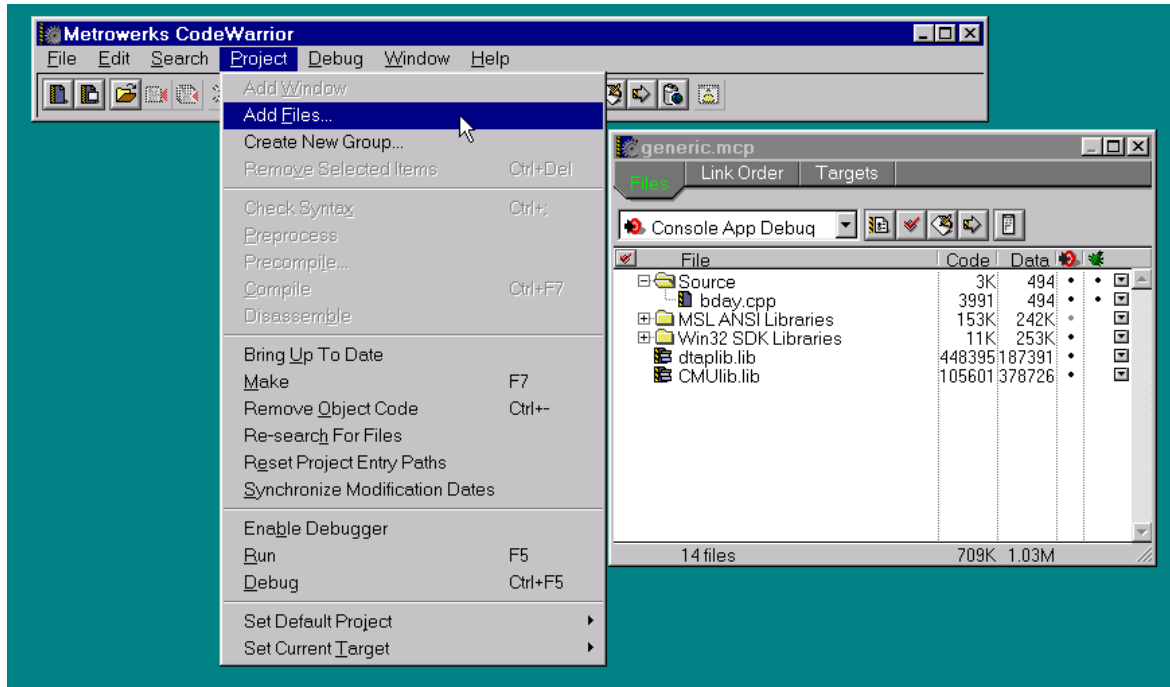


Figure A.3 Adding a file to a Codewarrior project.

You'll also need to add source files and libraries to a project. Typically, source files are added to the folder named *Source* that's part of the project, but they can be added anywhere. The first time you create a project you may need to add libraries too. Libraries are added in the same way that files are added, but typically the libraries are placed at the end of the project. Figure A.3 shows a project named *generic.mcp* with one source file named *bdlay.cpp* and two added libraries, *dtaplib.lib* and *CMUlib.lib*. The figure shows the user selecting *Project* → *Add Files* from the **Project** menu to add a new source file or library to the project.

If you're adding source, you should select the *Source* folder before selecting *Add Files* so that the source file is added to the *Source* folder. I often forget to do this and the .cpp source file is added at the end of the project with the libraries shown in Fig. A.3. You can leave the .cpp file there, but I usually click-and-drag it to the source folder within the project.

**Making a Project Debuggable.** When you first create a project, it may be in what's called *Release* mode. You can see a project's mode in the Project box's target section (there's a little target icon). You should change *Release* mode to *Debug* mode by clicking in the target section and choosing *C++ Console App Debug* as shown in Fig. A.4. At

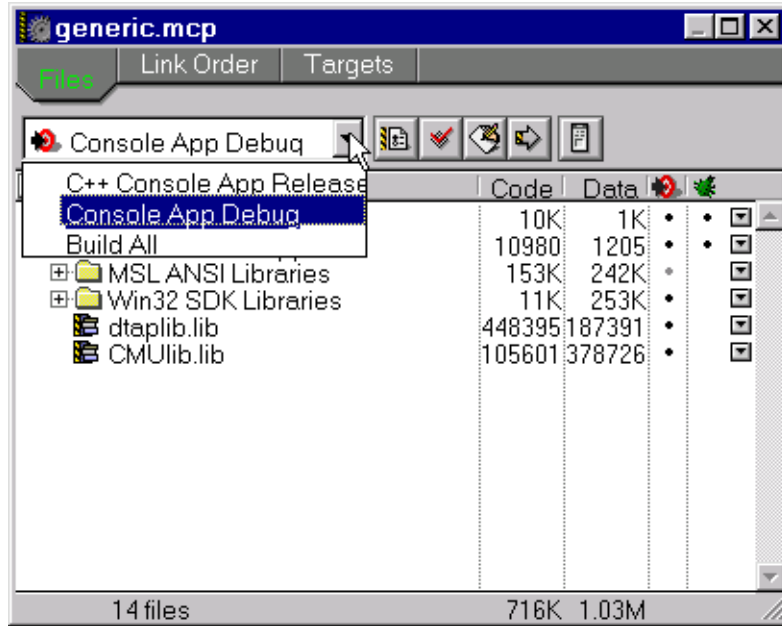


Figure A.4 Changing a Codewarrior project from Release to Debug to support symbolic debugging.

some point you'll want to use the symbolic debugger to find errors in your program; your project must be in *Debug* mode to support debugging.

### A.3.2 The Include Path in Codewarrior

Whenever you add a file to a Codewarrior project, the include path for the project is updated so that the directory in which the added file is located becomes part of the include path. This is a very useful feature and means that you don't often need to manually change the include path. However, it is sometimes necessary to explicitly add directories to the include path. To add a directory, first choose *Edit* → *Console App Settings* from the **Edit** menu. Then select *AccessPaths* from the *Target Settings* section of the dialog that pops up as shown in Fig. A.5.

Click on the *Add* button and browse to the folder you want to add. You can remove directories by selecting the directory and then clicking on the *Remove* button. To save the changes you make, you must click the *Save* button in the lower-left corner of the dialog box. To close the dialog box, click on the *x* icon in the upper right corner.

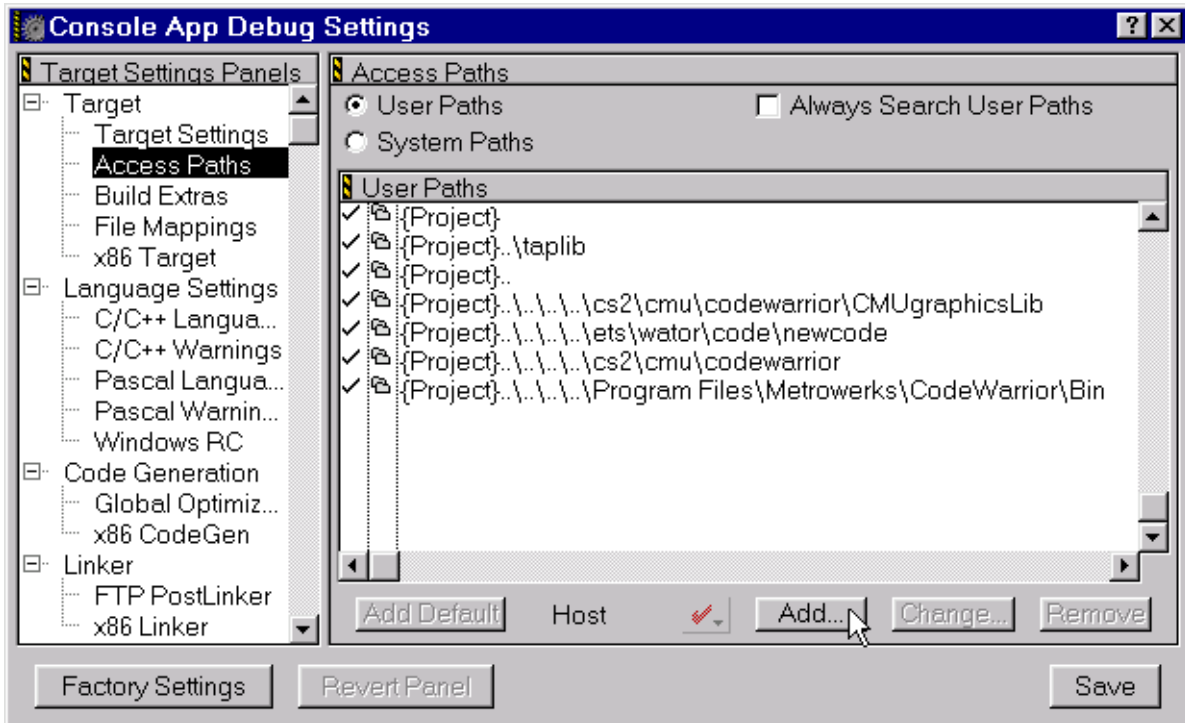


Figure A.5 Adding a directory to the include path in Codewarrior.

### A.3.3 Creating a Library in Codewarrior

Rather than storing ten or twenty source files for classes and functions you use in every project, you can create a library of compiled object files and add the library to the project. In Codewarrior it's very simple to create a library.

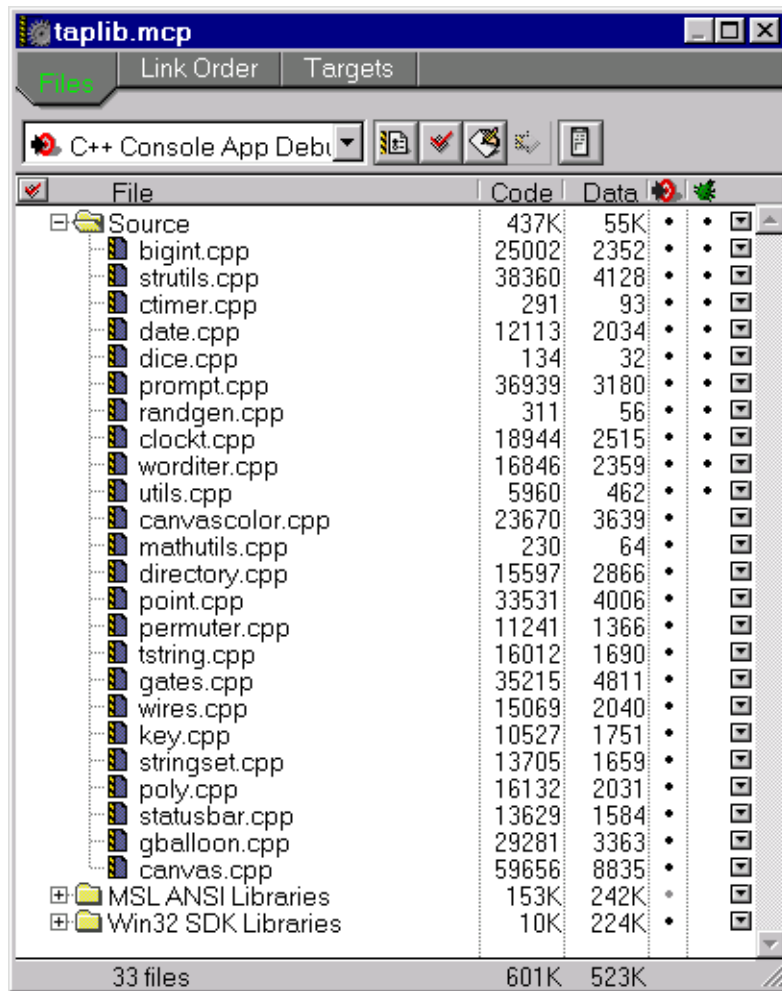
I typically create a separate project for each library I use so that I can update the library by adding a new class or recompiling changes to an old class. To create a library, start a project as described in Section A.3.1. Then add all the source files to the project that you want to be part of the library. The source files that are part of *dtaplib.lib* are shown in Fig. A.6. I call the library *dtaplib* because it's the debuggable tapestry library.

Once you've added all the source files to the project, you must change settings so that the compiler will generate a library instead of an executable program. None of the source files added to the library should have a `main` function; each source file should be the implementation of related non-templated classes or functions. There's no reason to add templated classes or functions to a library since they're not compiled, they're **instantiated**. It is possible to add an instantiated template to a library, but we don't discuss that here. To change a project so that it generates a library choose *Edit* →

**Appendix A** How to: cope with C++ environments

*Console App Settings* from the **Edit** menu. Then choose *x86 Target* from the *Target Settings* section of the dialog box that pops up. Click on the *Project Type* box and choose *Library (LIB)* as shown in Fig. A.7.

You'll need to specify a name for the library in the same dialog box, use the text box labeled *File Name* and specify a `.lib` suffix. To create, or make the library, choose *Project* → *Make* from the **Project** menu. You must choose *Make* to create the library, choosing *Build* compiles the files, but doesn't actually create the library. You'll notice that the *Run* option of the **Project** menu is grayed out since a library isn't an executable.



**Figure A.6** The source files in *dtaplib.lib*, the library of Tapestry classes and functions.

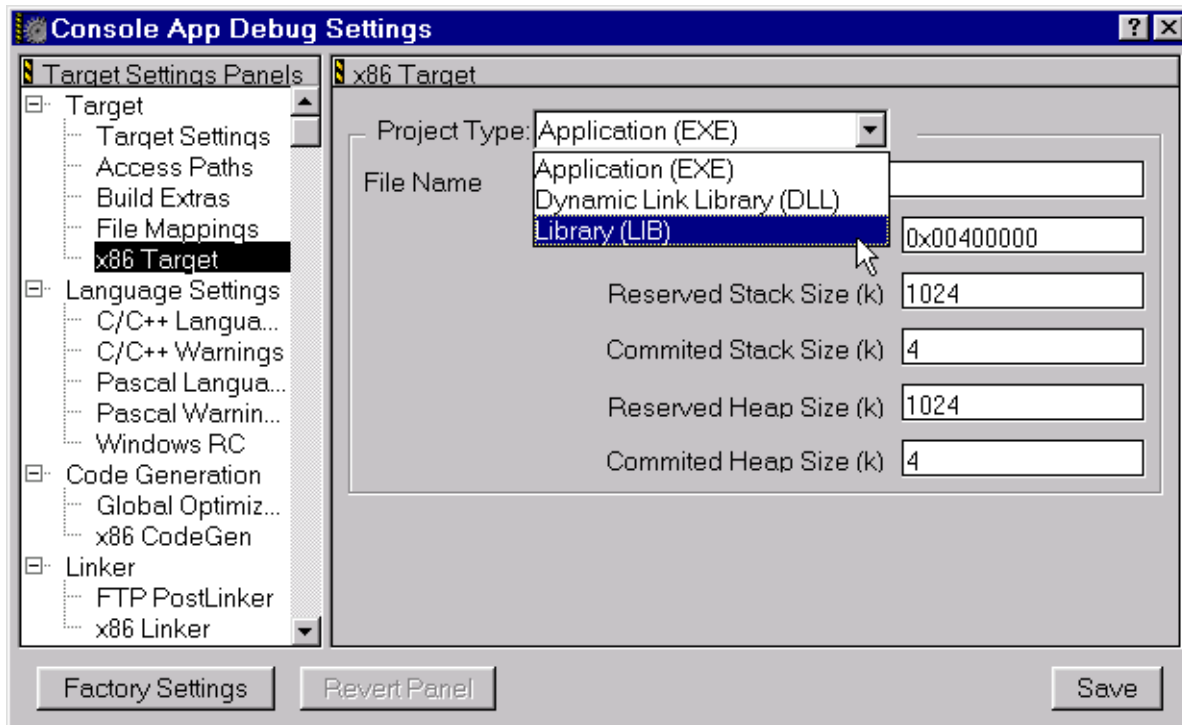


Figure A.7 Changing a project in Codewarrior so that it generates a library.

### A.3.4 Running a Program

You run a program in Codewarrior by selecting *Project* → *Run* from the **Project** menu or using the shortcut F5 key. To run a program, the IDE first determines if any source files need to be recompiled because they've changed, or header files they used have changed. After recompiling and relinking, the program is executed.

You can compile a source file without executing a program by choosing the *Compile* option from the **Project** menu. Choosing *Bring Up To Date* compiles all out-of-date source but doesn't run the linker or execute the program. You can also debug a program rather than execute it by choosing the *Debug* option. Other selections from the **Project** menu are mostly self-explanatory. Occasionally it's useful to select *Remove Object Code* which forces all source to be recompiled and removes what may be out-of-date object files. When I think a program should compile and link but it doesn't, I'll try removing all the object files and recompiling.

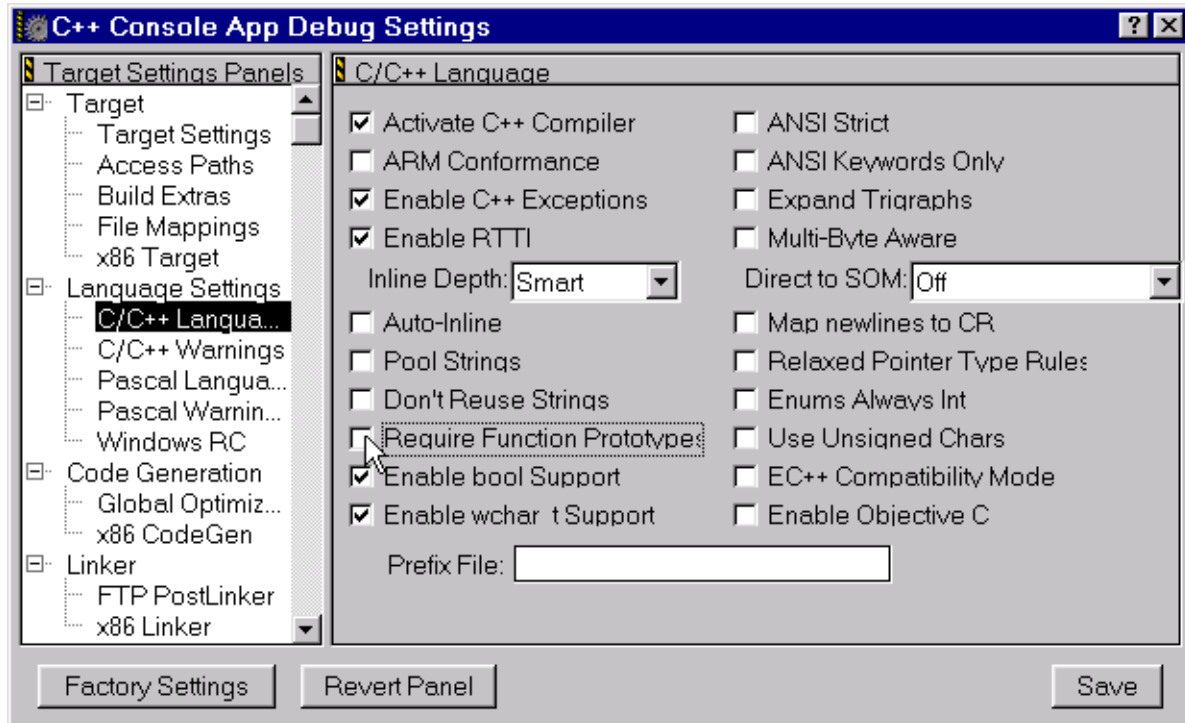


Figure A.8 Turning off *Require Function Prototypes* in a Codewarrior project.

### A.3.5 Codewarrior Warnings

Codewarrior generates a warning if you don't include a prototype for a function, even if the function appears before it's called. This annoying warning can be turned off by choosing *Edit* → *Console App Settings* from the **Edit** menu, then selecting *C/C++ Language* from the *Language Settings* section of the dialog box. Unselect the *Require Function Prototypes* as shown in Fig. A.8.

You may want to select all warnings when compiling. To change the warnings in a project use the same dialog box used for changing the *Require Function Prototypes* shown in Fig. A.8, but choose the *C/C++ Warnings* option from the *Language Settings* section. You can add all warnings, or deselect those you don't want. For example, *Illegal Pragmas* in system files are often flagged, but they're typically not part of programs you write so you may want to turn that warning off.