Paradigm C++ Quick Start Guide

Version 5.0

Paradigm Systems

Version 5.0

July 26, 2000

Paradigm Systems
3301 Country Club Road
Suite 2214
Endwell, NY 13760
USA

(607)748-5966
(607)748-5968 (FAX)
Sales information: info@devtools.com
Technical support: support@devtools.com
Web: http://www.devtools.com
FTP: ftp://ftp.devtools.com

For prompt attention to your technical questions, contact our technical support team via the Internet at support@devtools.com. Please note that our 90 days of free technical support is only available to registered users of Paradigm C++. If you haven't yet done so, take this time to register your products under the Paradigm C++ Help menu or online at http://www.devtools.com.

Paradigm's SurvivalPak maintenance agreement will give you unlimited free technical support plus automatic product updates for an additional 12 months. Call (607) 748-5966 to purchase this protection today.

Table of Contents

Chapter 1

Getting started

Command-line diehards need not despair, a complete set of command tools and make utility is included with Paradigm C++.

Welcome to Paradigm C++, a state-of-the-art integrated development environment (IDE) for creating x86 real and protected mode embedded system applications in C, C++, and assembly language. With the Paradigm C++ IDE, you can create, debug, and deploy real-time embedded system applications without resorting to the use of external tools. If you are used to running separate editor, debugger, make, and other tools to get a job done, then you are in for a real treat with Paradigm C++.

To help you get familiar with the all the powerful capabilities of the Paradigm C++ IDE, this guide offers an overview of the key technologies that work for you in Paradigm C++:

- Starting Paradigm C++
- Using SpeedMenus
- Using the Edit Window
- Working with projects
- Configuring the remote connection
- Debugging with Paradigm C++
- Customizing Paradigm C++
- Using help in Paradigm C++

At first, Paradigm C++ may take some getting used to since it breaks the old-style embedded system development metaphor of separate edit, compile, and debug tools, and instead tracks the way modern applications are generated. Paradigm C++ includes many powerful features you may not be familiar with, so it pays to explore its full potential before you jump headfirst into a new project. Take a look at the material we provide here and use it as the basis for creating and modifying your own projects.

Starting Paradigm C++

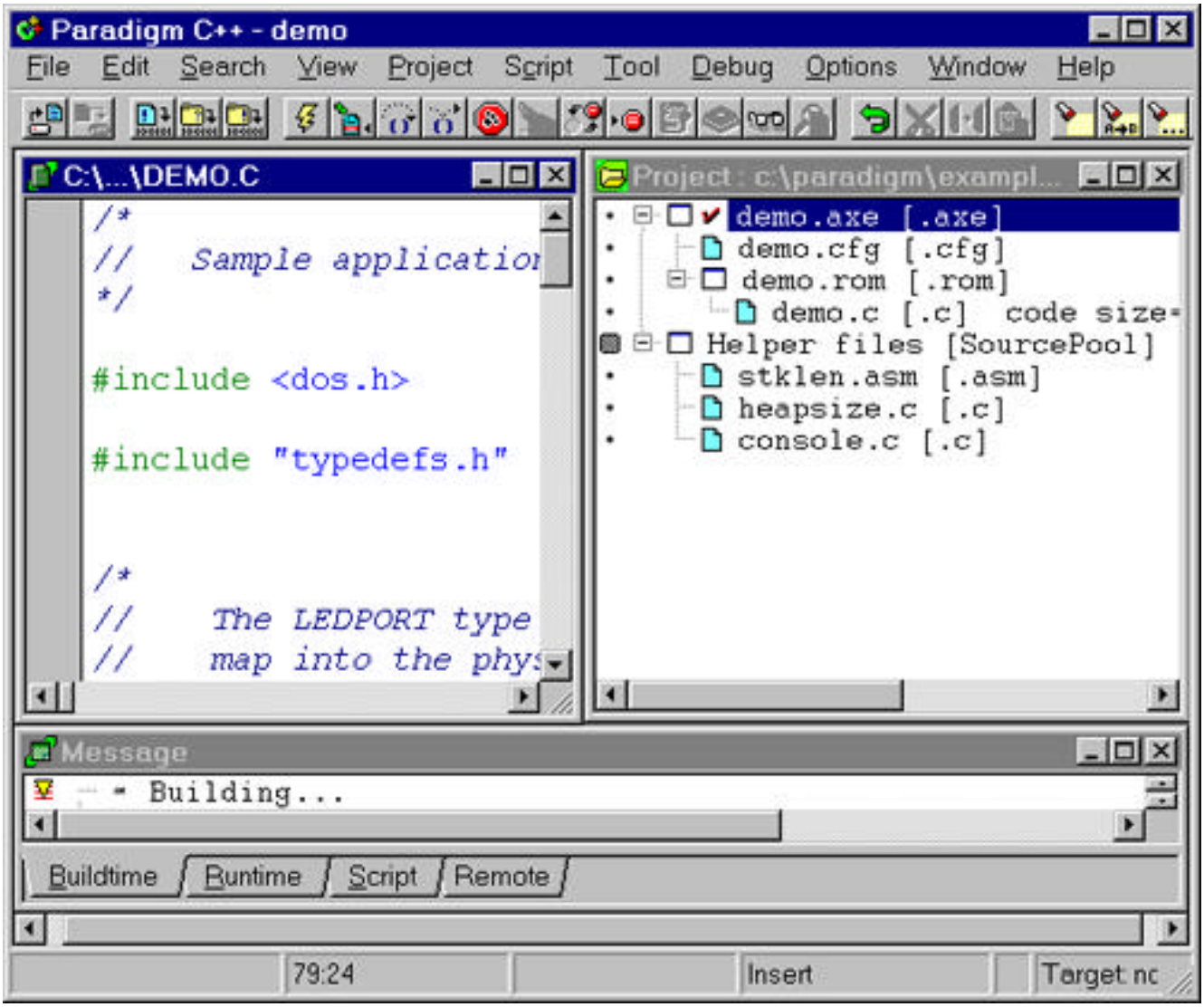
Or select INSTALL.EXE from the CD-ROM drive.

Installation instructions launch automatically from the Paradigm C++ CD. After following the instructions, exit the install screen. The Start menu will contain a program item titled Paradigm C++. Use the program item to launch Paradigm C++.

For optimum performance, Paradigm C++ requires a Pentium 120, Windows 95/NT, and 50MB hard disk space.

Figure 1-1, page 1-6 shows how Paradigm C++ looks after loading the DEMO.IDE project, opening DEMO.C, and building the application. The key features to note are the Menu Bar offering access to the various Paradigm C++ tools, the SpeedBar displaying context-sensitive shortcuts to relevant operations such as debugging and browsing, and the Status Bar at the very bottom with contains up-to-the minute information of the status of Paradigm C++. Filling the remainder of the window are the Edit, Project, Message and other views, where the real work of developing an embedded application will take place.

Figure 1-1 Paradigm C++ IDE screenshot



The Paradigm C++menu system

The following table describes the menu options on the Paradigm C++ Menu Bar.

Table 1-1 *Paradigm C++ global menus*

Menu item	Command descriptions
File	Commands to open, save, and print files. Also includes the Paradigm C++ exit command along with a list of recently accessed files.
Edit	Clipboard command and commands for undoing and redoing operations on edit buffers.
Search	Commands for searching and replacing in edit buffers, files, or the current project, browsing symbols, locating functions, and reviewing error messages generated by the programming tools.
View	Commands to open the Project Manager, Message window, and Browser. Also contains commands to open the integrated debugger views during a debugging session.
Project	Commands to open, close, and build or make a project.
Script	Provides commands to run and test scripts to automate Paradigm C++. cScript is a powerful Paradigm C++ feature that allows you to automate and integrate tools into Paradigm C++.
Tool	Commands to launch any external programming tools from Paradigm C++.

Debug	Commands to run your project under control of the Paradigm C++ integrated debugger.
SCCS	Source code control system integration commands. This is an optional menu that is present when a source code control add-in is installed.
Options	Paradigm C++ customization and project configuration commands. Here is where you can completely tailor Paradigm C++ to work as you do.
Window	Paradigm C++ window management commands give you complete control to navigate between windows and close or minimize selected windows.
Help	Commands to access the Paradigm C++ online help are included here. Paradigm C++ includes extensive online help covering all of Paradigm C++, from the IDE operation to the details of the compiler run-time libraries.

More information about using cScript is available in the online Help.

Because Paradigm C++ is fully extensible by the end-user, there may be other entries on the menu bar from version control tools, real-time operating systems, and other third-party tools. With just a single line of Paradigm Scripting Language (cScript) code, you can have your favorite commands displayed here to use whenever you need them.

The Paradigm C++ IDE SpeedBar

The SpeedBar (located under the main menu) has buttons that give quick access to menu commands that relate to the area of Paradigm C++ you're working in. For example, if you're editing code, the SpeedBar contains cut and paste commands, file save commands, and so on, as well as commands to build and debug. When the Project window has focus, the SpeedBar has buttons that pertain to projects, such as commands for adding project nodes and browsing option settings.

Figure 1-2 Paradigm C++ IDE SpeedBar example

The Status Bar at the bottom of Paradigm C++ contains "flyby" help hints; when the cursor is over a button, the Status Bar describes the button command. You can configure the flyby hints and other SpeedBar options as described in “Customizing the SpeedBars,” page 19. See Figure 1-7, page 1-16 for a description of the above Paradigm C++ SpeedButtons available during a debug session.

Using SpeedMenus

Right-clicking (clicking the right mouse button) accesses the Paradigm C++ *SpeedMenus*. SpeedMenus contain commands that are context-sensitive to the area of the program you're working in. For example, the SpeedMenu for the Edit window contains commands that are related to the editor. In the Project Manager, the SpeedMenus contain commands to help you with managing your project.

To get a feel for SpeedMenus, try the following:

If you installed Paradigm C++ in a different directory, adjust the paths used in this guide.

1. From the Paradigm C++ Menu Bar, choose Project|Open project, then select the project file DEMO.IDE in the PARADIGM\EXAMPLES\REAL\DEMO directory.
2. Double-click the DEMO.C node in the Project window to load the file in an Edit window so changes can be made.
3. Move the cursor to the **embedded.h** header file reference by clicking on the file name in the source code.

4. Right-click to open the Edit window SpeedMenu, then choose Open Source to open an Edit window that contains this header file. You can do this even quicker using the by right-clicking anywhere in the DEMO.C Edit window and selecting the Include command. Paradigm C++ will instantly parse the file and extract all include file references in the buffer. Just select the desired include file and you are instantly there to begin making changes.

☞ In addition to right-clicking, Paradigm C++ SpeedMenus can be accessed at any time by pressing *Alt-F10*.

Using the Edit window

Edit windows contain the Paradigm C++ editor, which you can use to create and edit your program code. When you're editing a file, the Paradigm C++ status bar displays the following information about the file that you are editing:

- The line number and character position of the cursor. For example, if the cursor is on the first line and first character of an Edit window, you'll see 1:1 in the Status Bar. If the cursor is on line 68 and character 23, you'll see 68:23.
- The edit mode: insert or overwrite. Press *Insert* to toggle whether your text additions overwrite existing characters or insert new ones into the file.
- The file's save status. The word Modified appears if you've made changes to the file in the active Edit window, and you have not yet saved your edits or changes.

☞ The Paradigm C++ editor contains many powerful features to help you enter and modify your program code. For example, you can undo multiple edits by choosing Edit|Undo or pressing *Alt-Backspace*. You can also open multiple Edit windows; tile the windows as you wish; subdivide the window into different Edit panes; and cut, copy, and paste text between any open files. Paradigm C++ is supplied with four editor emulations and if these don't suffice, you can create your own editor from any of the supplied editors.

Although this chapter provides a brief introduction to the editor, complete details on how to use and customize the editor can be found in the online Help. Choose Help|Contents and double-click Paradigm C++ User's Guide. The Editor is discussed within the Integrated Development Environment (IDE) book topics.

Creating a new file

To introduce you to the editor, step through the following instructions to add a new source file to a sample embedded application.

1. If not already open, File|Open the DEMO.IDE project in PARADIGM\EXAMPLES\REAL\DEMO.
2. From the Paradigm C++ Menu Bar, choose File|New|Text Edit to open a new Edit window with an empty file.

   By default, Paradigm C++ names new files NONAME*xx*.CPP, where *xx* is a number that is incremented with each new file opened. Don't worry about the filename for now, you'll be prompted to change it when you save the file.
3. In the Edit window, type the following C++ code to create a simple embedded program.

```
#include <stdio.h>

char buffer[128] ;

void main(void)
{
   unsigned passcount = 0 ;

   char* format = “%05u Welcome to ParadigmC++!\n” ;
   for (;;)   {
      sprintf( buffer, format, passcount) ;
      passcount++ ;
   }
}
```

4. Choose File|Save, and save your new file with the file name TEST.C.

Although we created the file, it is not yet part of our Paradigm C++ project. Later, in “Creating an embedded application,” page 11, we will show you how to add this file to the project where it will get built with other source files in the project.

Navigating your source files

Once you have some text in the Edit window, you can navigate around your source code. Paradigm C++ utilizes instant-parsing technology to scan the current Edit window and extract information about functions, structures and classes, enumerations, and include files. In small files, source code navigation is possible by scrolling the Edit window, in large files and multi-file projects, it really isn't possible.

To really see the parsing technology in action, try the following test. Using the file TEST.C that was just created, right-click in the window and select the Functions - only 'main()' should appear at this time.

Now add a new function to the file, such as

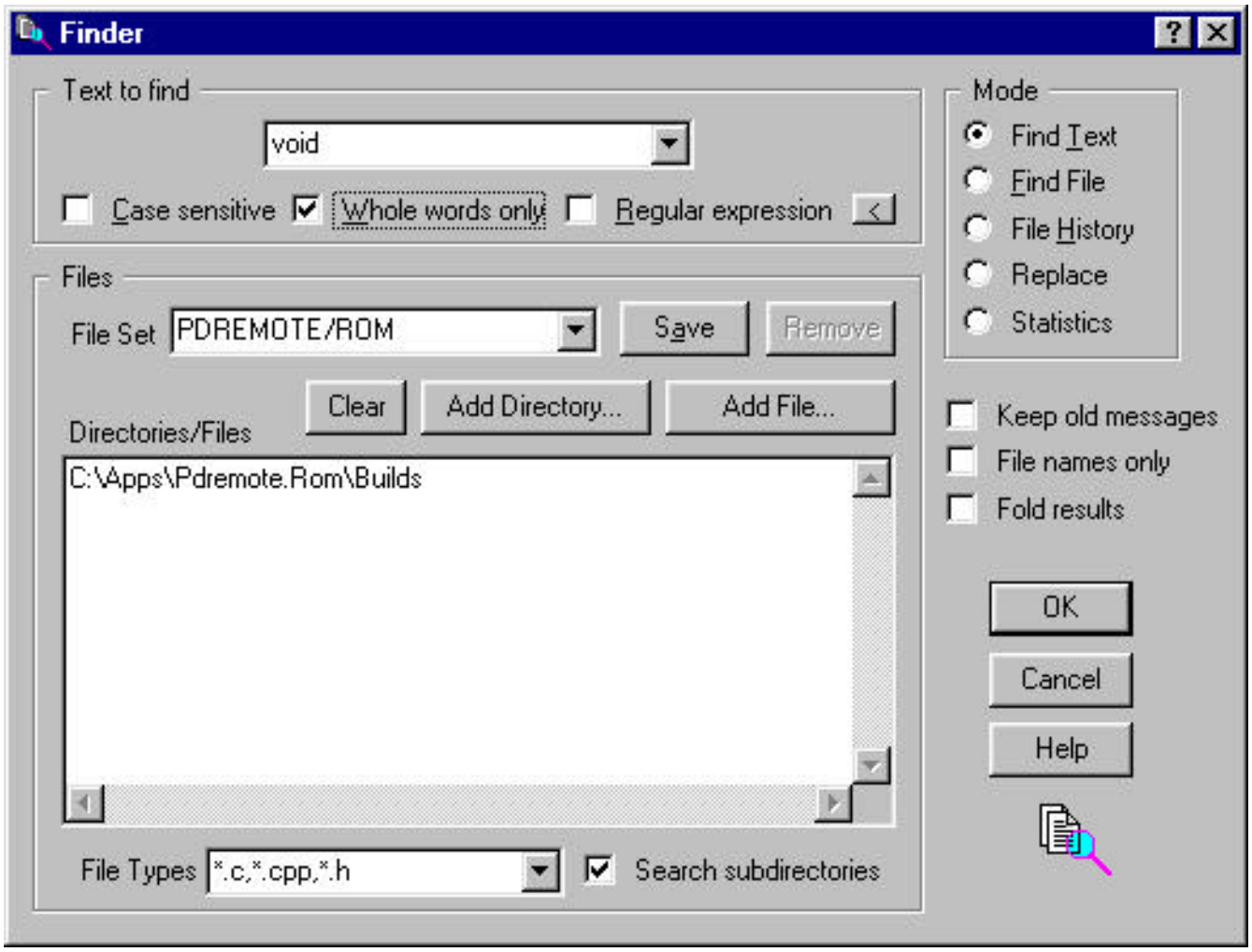
```
int test(int x, int y)
{
   return x + y ;
}
```

Now right-click in the window and select the Functions again and see that both main() and test() are in the list of functions in the file. No compiling, just instant access to your source code definitions to make it easy to navigate to any function, class or include file in the current Edit window.

Working beyond the Edit window

The Paradigm C++ Finder, found under the Search menu, can do much more to help with the software development process. The Finder provides the ability to search within a file, a project, or the entire disk drive for any regular expression. This is an incredibly powerful capability when you need to find text or make changes across one or many files in your project or on your disk.

Figure 1-3 Paradigm C++ Finder



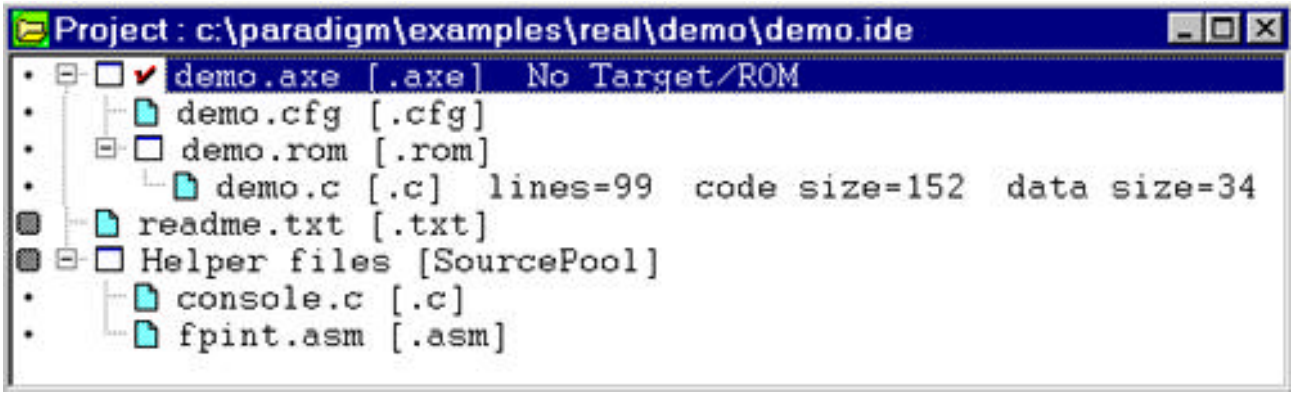
No matter what your needs, Paradigm C++ has the tools you need to manage and maintain your project files. While the Finder works on any source file, the browser adds even more power by using compiled code to create a database that can be utilized to find where a function is defined and all the instances that it is used in. More information about the browser is available in the online Help index under "browser" or see Chapter 4, “Browsing through your code,” of the Paradigm C++ Reference Manual.

Working with projects

After you install Paradigm C++, you'll want to make sure the program is correctly set up; the details of the compiler and the Paradigm C++ IDE can wait until later. The best way to test your setup is to compile, build and load the sample applications included with Paradigm C++.

Paradigm C++ uses *projects* to help manage your code and make sure any source code changes are reflected in the other files that depend on them. As an application grows in size and complexity, it becomes dependent on various intermediate files. Often, source files need to be compiled with different compilers and different sets of compiler options. Even a simple embedded application can have multiple C/C++ source files, with each file type requiring different compilers and different compiler settings.

Figure 1-4 The Project Manager Project window



As your project complexity increases, the need increases for a way to manage the different components in the project. Looking at the files that make up a project, you can see that a project combines one or more source files to produce a single target file. While target files are usually a .AXE or .HEX file, source files cover a broader range of file types, including .C, .CPP, .ASM, and other files. Additionally, many source files have autodependent files (files that are automatically included by the source), such as C header files. In larger projects, you are likely to find several targets with scores of sources.

To get the most from Paradigm C++, we need to create a project so the files and build options are saved, just as they would be in a more traditional makefile.

Creating an embedded application

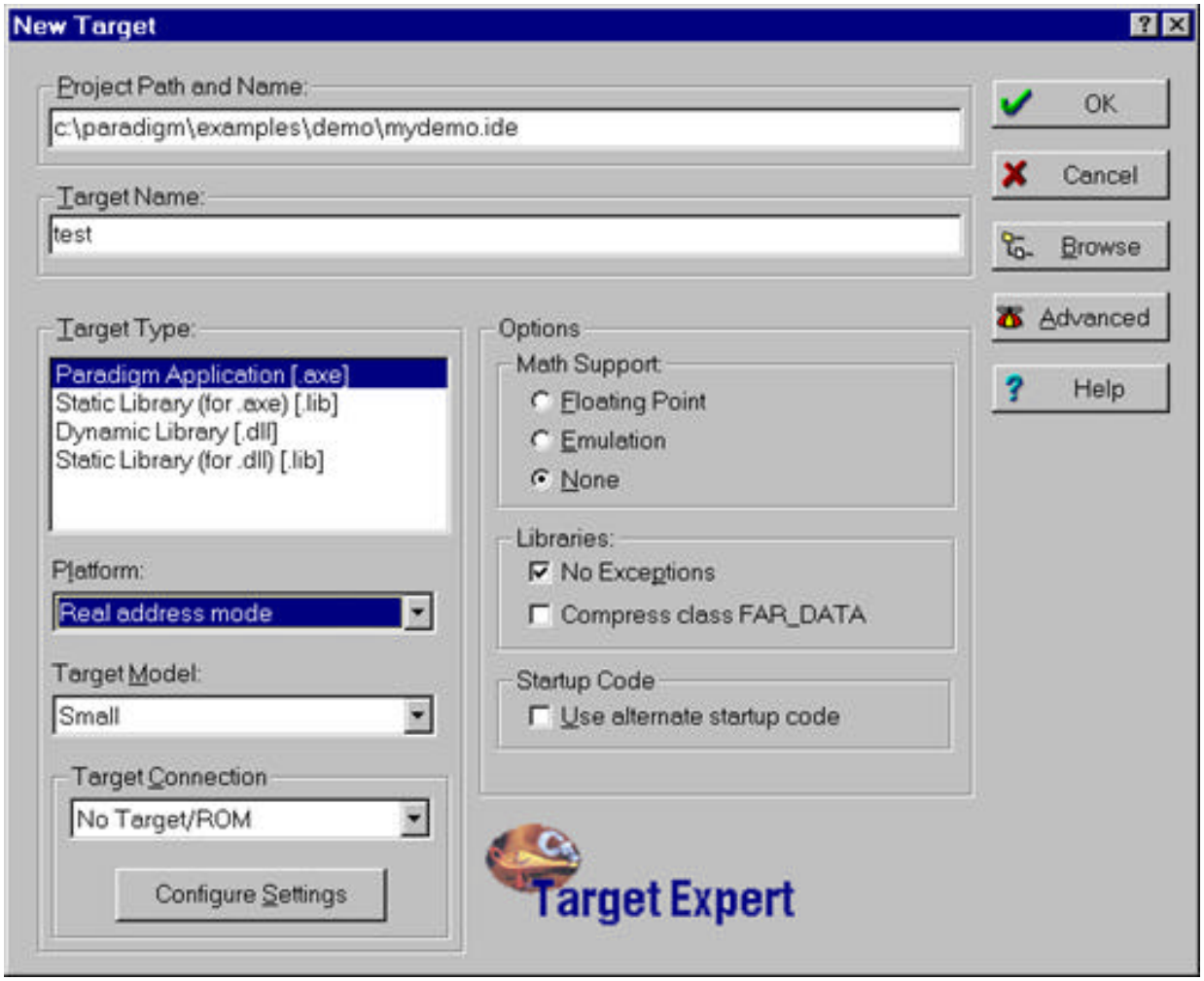
You can become familiar with the Project Manager and the C/C++ compiler by following these steps to create a simple embedded application:

If the directory doesn't exist, Paradigm C++ creates the directory for you.

1. From the Paradigm C++ Menu Bar, choose File|New|Project…, then set the following options in the New Target dialog box:
   - Type the path and name for your new project in the Project Path and Name input box. In this case, type:
     ```
     \paradigm\examples\demo\mydemo.ide
     ```
   - Type the target name you want to use. Because you can have more than one target in the project, you can have different names for targets that share files and options. In this case, type:
     ```
     test
     ```
   - In the Target Type list box, click Paradigm Application [.AXE]. This selection will create a project where the source modules are compiled, assembled, linked and located to generate .AXE files for debugging or .HEX and .BIN files for placing within flash or EPROM devices.
   - If you like, select the desired platform and memory model you want for your application. You can also enable the use of floating point arithmetic or other options that depend on the selected Target Type.
   - If you like, choose a target connection from the list of available remote connection interfaces.

The New Target dialog box should now resemble the one shown in Figure 1-5, page 1-12.

Figure 1-5 New Target dialog box



2. Choose *OK* to close the New Target dialog box.
3. The Project window opens and displays the target and dependencies of the project you just created.

   The following are definitions for the nodes within this newly created project:

   TEST.AXE This node is the final target node that is generated during the locate phase of the build and includes the absolute code for debugging or burning into flash or EPROM.

   TEST.CFG This is the Paradigm LOCATE configuration file that is used to generate the TEST.AXE output file. This file contains the description of the target system address space as well as the build instructions for placing the program code and data at addresses that you specify.

   TEST.ROM This node is generated by the Paradigm C++ linker and is also the point in the build process in which a .MAP file is generated for use in the locate phase.

TEST.C This node references the files TEST.C, the file that you created earlier in the chapter (if you haven't already done so, create this file by following the instructions listed in the section, “Creating a new file” on page 8).

4. Build the application by selecting the .AXE node and right-click to bring up the local options and select 'Build node'. Because Paradigm C++ has a built-in Project Manager, it always knows when the project is out-of date and needs to be rebuilt so there is no need to explicitly do this. We could have also selected the Project|Make all or the Project|Build all commands from the SpeedBar or from the Project menu.

If you correctly followed all the steps in this section, the application builds without errors. If the compiler reports errors or warnings during the compile, retrace the steps in this section to ensure you correctly followed the steps. When the program compiles without errors, the Project Manager creates an executable program called TEST.AXE and places it in the directory you selected when the project was created.

You can access the Master Index from within any online Help topic by right-clicking.

This is only a small fraction of the information on working with projects. See "projects" in the online Help index for complete details on managing the build process using Paradigm C++ projects or see Chapter 2 “Managing projects,” of the Paradigm C++ Reference Manual.

Configuring the remote connection

A target connection must be specified to begin debugging.

Paradigm C++ can only debug when a target such as PDREMOTE/ROM or an in-circuit emulator is connected. Configuring the remote connection gives Paradigm C++ the information it needs about your target to begin a debugging session. To configure the remote connection:

1. Select the TEST.AXE node from the project view of the Project Manager.
2. Right-click to see local menu options for the TEST.AXE node.
3. Select TargetExpert. The dialog contains a pull down menu for Target Connection.
4. Select the drop down menu to see a list of available remote connection interfaces and select the desired remote connection interface.
5. Press the Modify connection settings button to make specific changes to the remote interface settings.

Once the remote connection settings are set up, click *OK* to close the remote connection dialog. You are now ready to start debugging. Double-click the TEST.AXE node to rebuild the application (if needed) and download the application to the target.

The debugger sets a software breakpoint at the label **main** in the application. If you would rather start at the reset vector or run to a different place in the application, then select Options|Environment|Debugger|Debugger Behavior and delete **main,** or add the function name, to Run to on startup.

Stand-alone debugging

To configure the remote connection to do stand-alone debugging without the use of Project Manager,

1. Close any demo projects that may be open (Project|Close project).
2. Select Debug|Load, and *Browse* or type in the name of the .AXE (or .HEX) file for remote download, for example,

```
\paradigm\examples\demo\test.axe
```

3. Choose the desired remote connection interface.
4. Press the Modify settings button to change any specific remote connection settings for the selected interface.
5. Select *OK* to load the application file and start debugging without the use of the Project Manager.

   Again, the debugger sets a software breakpoint at the label **main** in the application. If you would rather start at the reset vector or run to a different place in the application, then select Options|Environment|Debugger|Debugger Behavior and delete **main,** or add the function name, to Run to on startup.
6. When you would like to exit stand-alone debugging mode, Select Debug|Terminate debug session or hit *Ctrl-F2*.

Debugging with Paradigm C++

If you have multiple targets, you can select the target connector from the local menu of a target node in the Project view.

For a demonstration of debugging in Paradigm C++, open the DEMO.IDE project in PARADIGM\EXAMPLES\REAL\DEMO as you did in “Creating a new file,” page 8 and double-click the DEMO.C node in the Project window. Right-click the DEMO.AXE node in the Project window and and select TargetExpert to ensure that the Target Connection is set to the desired remote connection. Then simply double-click the DEMO.AXE node in the Project window to download the application to your target. If the debugger option to execute to main(), found under Options|Environment|Debugger|Debugger Behavior, is enabled, the debugging session will look like Figure 1-6, page 1-15.

At this point the Debug menu commands and SpeedBar will come alive so you can inspect program data, view the processor registers, or access target peripherals. Right-clicking in the Edit window will bring up the debugger SpeedMenu for quick access to debugging commands.

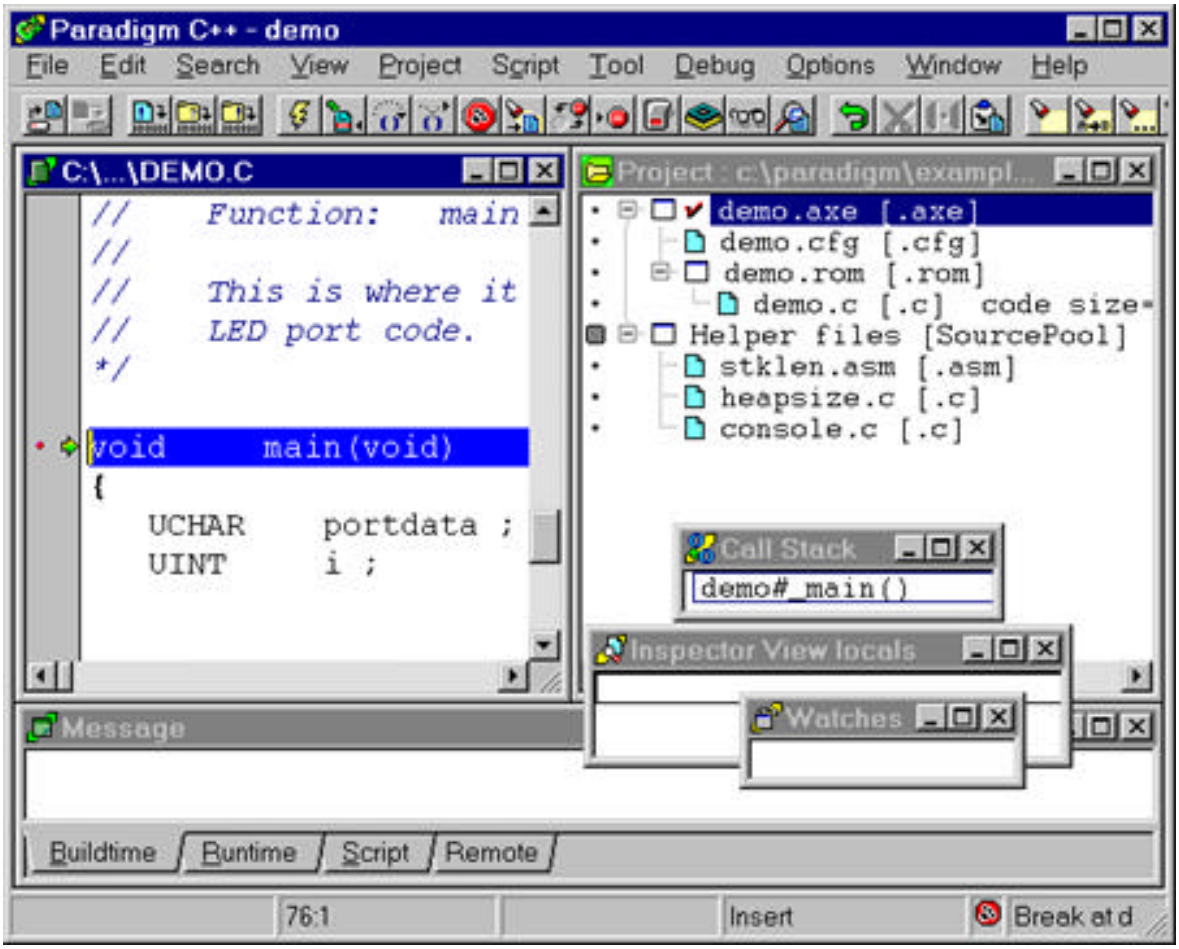
Step over/into

Run/Run to

You can step through the program and test until you find a bug that needs fixing. Use the Statement step over, step into, or step out of SpeedButtons located beneath the Menu bar to begin debugging. You could use the Run SpeedButton but the application won't stop unless you have set a breakpoint somewhere in the program. You can also use the Run to here button to execute to a particular source line that the cursor is on. See Figure 1-7, page 1-16 for a description of Paradigm C++ SpeedButtons available during a debug session.

When you find a problem, you might notice that there is no difference between editor windows and debugger windows. This is a big improvement over traditional tools since you can fix a bug right away without exiting the debugger. If you make a change, you can either continue the debugging session or you can rebuild the application and test the change - all without losing your place! This is where Paradigm C++ excels at making the most of your development time.

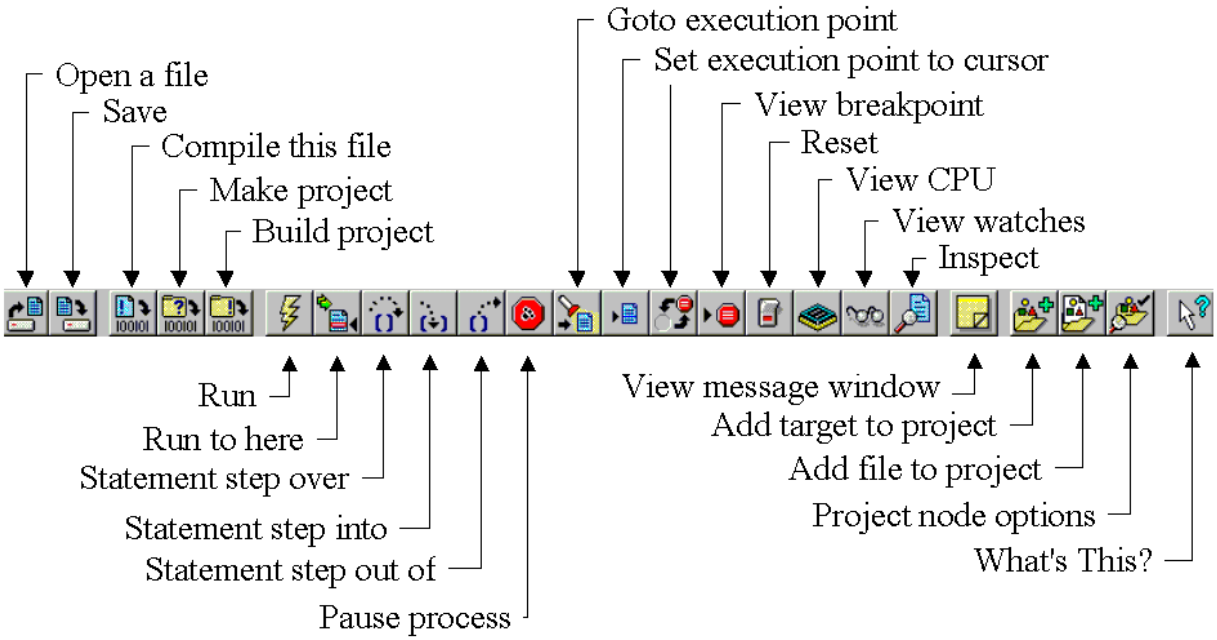
Figure 1-6 Paradigm C++ debugging session



Debugger SpeedButtons

This section will familiarize you with the Paradigm C++ SpeedButtons used in a debugging session.

Figure 1-7 Paradigm C++ SpeedButtons



In Paradigm C++, click the What's This? SpeedButton and then a SpeedButton of interest to receive a help description of that button. The Paradigm C++ debugger is covered in complete detail in Chapter 5 “Using the integrated debugger,” of the Paradigm C++ Reference Manual. We have covered just the basics here. Plan on spending some time in Chapter 5 or see "integrated debugger" in the online Help index for more assistance on the Paradigm C++ integrated debugger.

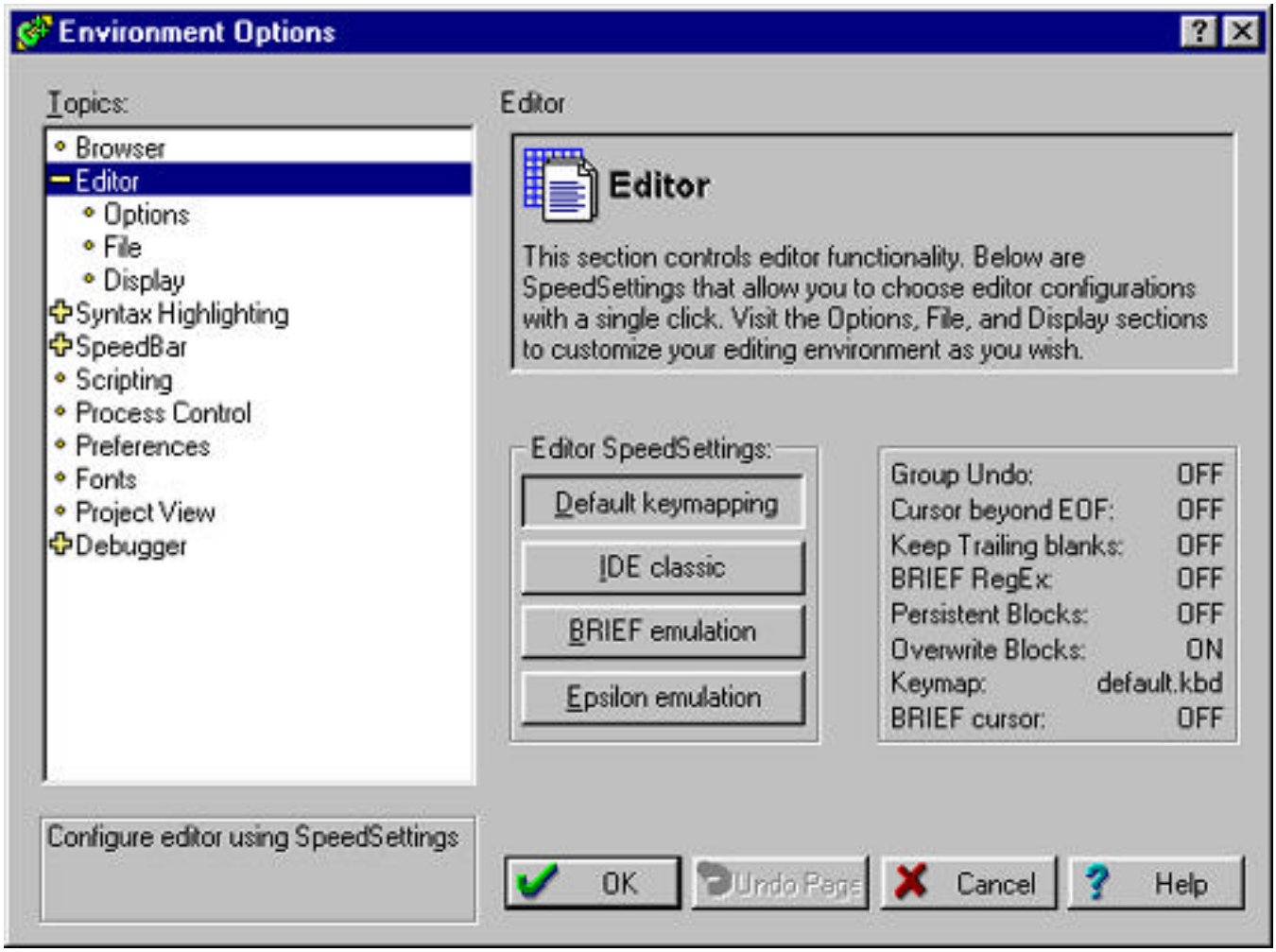
Customizing Paradigm C++

You can configure Paradigm C++ in many ways to create a customized environment that meets your programming needs. For example, you can have Paradigm C++ do tasks automatically (such as saving backups of your files in the Editor windows) or handle special events.

The Environment Options dialog box (accessed with the Options|Environment command) lets you configure the different elements and windows of Paradigm C++. Once you've customized Paradigm C++ to your liking, choose Options|Save, check the options you want to save, then choose *OK*; Paradigm C++ saves your environment settings to a file called PCCONFIG.PCW. By default, the file is saved to the BIN directory in your Paradigm C++ directory tree. This default directory is specified by the DefaultDesktopDir field of your PCW5.INI file, which is located in your Windows directory.

The Environment Options dialog box displays a list of customizable topics on the left and each topic's configurable options on the right. Some topics contain subtopics, indicated by a + next to the topic. For example, the Editor topic has subtopics called Options, File, and Display. To view a topic's subtopics, click the + sign next to the topic; its subtopics appear under it and the + turns to a - (you can then click the - to collapse the list of subtopics). Topics without subtopics appear with a dot next to their name.

Figure 1-8 Environment Options dialog box



This section discusses the following Environment options topics:

- Configuring the Paradigm C++ editor
- Selecting the Syntax highlighting options
- Customizing the SpeedBars
- Setting the Paradigm C++ preferences
- Saving your Paradigm C++ settings

☞ Although this chapter doesn't offer a complete reference to the many selections in the Environment Options dialog box, a complete reference is available by clicking the Help button.

Configuring the Paradigm C++ editor

You can configure the editor so that it looks and behaves like other editors such as Brief and Epsilon. The Paradigm C++ editor uses keyboard mapping files (.KBD files) that set the keyboard shortcuts for the editor and the other windows in Paradigm C++. You can modify this behavior using ObjectScripting. For more information, see "ObjectScripting" the online Help index.

Syntax highlighting

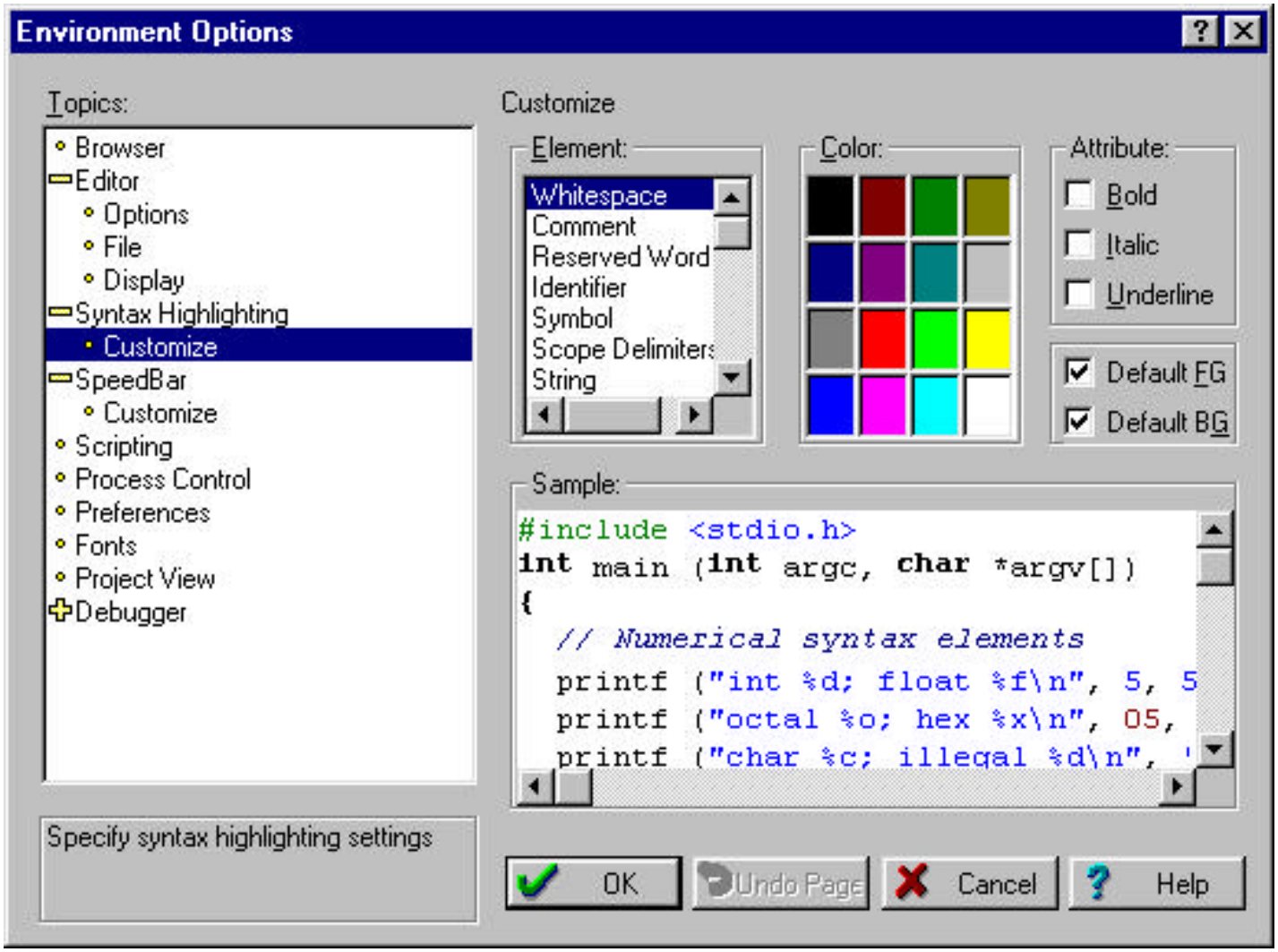
Syntax highlighting lets you define a color and font attribute (such as bold) for certain elements of code. For example, you could display comments in blue and strings in red. Syntax highlighting is on by default.

Syntax highlighting works on files whose extensions are listed in the Syntax Extensions list (by default, these files are .C, .CPP, .H, and .HPP). You can add or delete any extension from this list, but be sure to separate extensions with semicolons.

The Syntax highlighting section displays the default color scheme and four predefined color settings. To use a predefined color scheme,

1. Choose Options|Environment|Syntax highlighting.
2. Choose one of the four predefined color schemes (Defaults, Classic, Twilight, or Ocean) by choosing the Color SpeedSettings; the sample code changes to the color scheme you select.

Figure 1-9 Environment Options Syntax Highlighting dialog



To customize the syntax highlighting colors,

1. Choose Options|Environment, then select the Syntax highlighting topic.

2. Select a predefined color scheme to use as a base for your customized colors.
3. Choose the Customize topic listed under the Syntax highlighting topic. Elements and sample code appear on the right of the Environment Options dialog box.
4. Select an element you want to modify from the list of elements (for example, choose Comment), or click the element in the sample code (this selects the name in the Element list). You might need to scroll the sample code to view more elements.
5. Select a color for the element. The element color in the sample code reflects your selection. Use the left mouse button to select a foreground color for the element (FG appears in the color). Use the right mouse button to select a background color (BG appears in the color). IF FB appears in the color, the color is used as both a background and a foreground color.
6. If you want, choose an Attribute (for example, bold).
7. You can check Default FG (foreground) or BG (background) to use the Windows default colors for an element.
8. Repeat steps 2-4 for the elements you want to modify.

To *turn off* syntax highlighting, choose Options|Environment|Syntax highlighting, then uncheck Use Syntax highlighting.

Customizing the SpeedBars

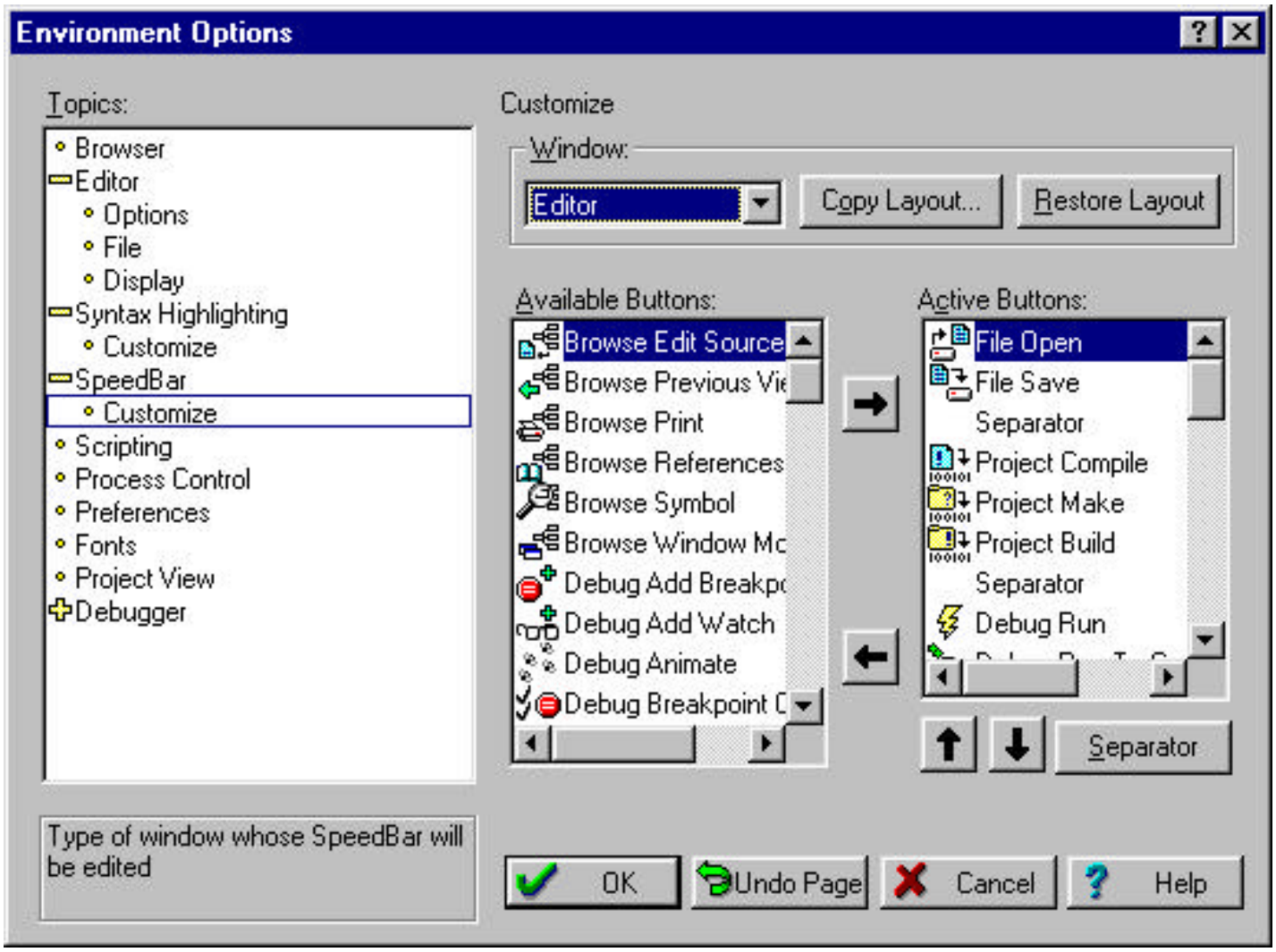
Paradigm C++ uses context-sensitive SpeedBars for all its windows, including Edit, Browser, Debugger, Project Manager, Message, and Desktop windows. When a window has focus, the corresponding SpeedBar appears just below the Menu Bar. Using the Environment Options dialog box, you can customize the SpeedBars for each window so that they include only the buttons you want.

To add or delete buttons from the SpeedBars,

1. Choose Options|Environment from the Paradigm C++ Menu Bar.
2. Choose the SpeedBar topic on the left. The right side of the dialog box displays general options for all SpeedBars.

   The options here let you specify if you want to hide or view the SpeedBar, where you want the SpeedBar to appear (on the top or bottom of the Paradigm C++ window), and if you want to use the Flyby Help Hints. If you check Use Flyby Help Hints, Paradigm C++ displays descriptions of the SpeedButtons on the status line when you pass the mouse pointer over a button. If you leave this box unchecked, the hints show on the status line only when you click a SpeedButton.
3. Choose the Customize topic listed under the SpeedBar topic to customize the SpeedBar for a particular window.

Figure 1-10 Environment Options SpeedBar customizing dialog



4. In the Window dialog box, choose the specific window (Edit, Browser, Debugger, Project, Message, or the Paradigm C++ Desktop) whose SpeedBar you want to customize.

   The Available Buttons list box displays all the unused buttons that you can add to a particular window's SpeedBar (each button has a name next to it that describes the button's function.). The Active Buttons list displays the buttons that are currently contained in the selected window's SpeedBar.

   - *To add a button* to a SpeedBar, double-click the button icon in the Available Buttons list, or select it and click the right-pointing arrow. Paradigm C++ places the button in front of the selected button in the Active Buttons list.
   - *To remove a button* from a SpeedBar, double-click the button icon in the Active Buttons list, or select it and click the left-pointing arrow. The button moves to the Available Buttons list.
   - *To reorder the button positions* for a SpeedBar, select a button in the Active Buttons list, and use the up and down arrows to move the button within the list. The top button in the list appears on the left side of the SpeedBar and the last button in the list appears on the right side of the SpeedBar.

- *To put separator spaces between buttons* on the SpeedBar, select a button from the Active Buttons list, and then click the Separator button. The separator is added *before* the selected button.

You can also make all SpeedBars identical by selecting a SpeedBar in the Window list, then pressing the Copy Layout button. A dialog box appears in which you check all the SpeedBars you want to make identical to the selected Speedbar. For example, if you first choose the Editor SpeedBar and then click Copy Layout, the dialog box appears with Editor dimmed. If you then check Project and Message, those SpeedBars will be exactly the same as the Editor SpeedBar.

You can restore any SpeedBar to its original defaults by selecting the SpeedBar in the Windows list, then clicking the Restore Layout button.

Setting Paradigm C++ preferences

The Preferences command lets you customize which of the Paradigm C++ settings you want automatically saved and how you want some Paradigm C++ windows to work.

To set preferences,

1. Choose Options|Environment|Preferences.
2. Check and uncheck the options you want, then choose *OK*. For an explanation of each option, select the option and hit *F1* to access the online Help for that option.

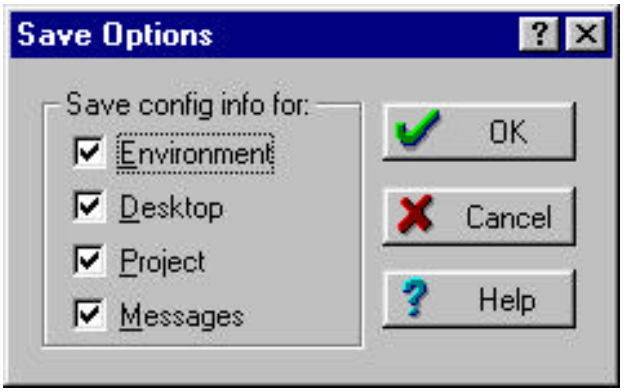
Saving your Paradigm C++ settings

Paradigm C++ automatically saves information when you exit Paradigm C++, use a transfer tool, build or make a project, run the integrated debugger, or close or open a project. You can control which areas of Paradigm C++ get saved from the Preferences topic in the Environment Options dialog box (choose Options|Environment from the main menu).

If you want to save your settings manually, you can do so as follows:

1. Choose Options|Save.

Figure 1-11 *Save options dialog*



2. Check **Environment** to save the settings from the Editor, Syntax highlighting, SpeedBar , Browser, and Preferences sections of the Environment Options dialog box. These settings are saved in a file called PCCONFIG.PCW.

3. Check **Desktop** to save information about open windows and their positions. This information is saved to a file called <*prjname*>.DSW. If you don't have a project open, the information is saved to a file called PCWDEF.DSW.
4. Check **Project** to save the changes to your project (.IDE) file, including build options and node attributes.

Using help in Paradigm C++

Paradigm C++ provides complete online documentation through the Help system. Using Help is a convenient way to get information about language features, compiler options, and any tasks you need to perform while developing applications in Paradigm C++.

Online help organization

The Help system is organized into Help files that include the following documentation:

Table 1-2 *Help files*

Help file	Description
Using Online Help	Features of Paradigm C++ Help (OPENHELP.HLP)
Paradigm C++ Class Libraries Guide	Programming and reference material (CLASSLIB.HLP)
Paradigm C++ Programmer's Guide	Programming tips and language details (PCPP.HLP)
Paradigm C++ User's Guide	Paradigm C++ tasks, projects, tools (PCW.HLP)
Error Messages and Warnings	Paradigm C++ Error message descriptions (PCERRMSG.HLP)
Tools and Utilities	Command-line tools (PCTOOLS.HLP)
ObjectScripting Guide	Customizing with scripts in Paradigm C++ (SCRIPT.HLP)
Paradigm LOCATE Reference	Reference material for Paradigm LOCATE (LOCATE.HLP)
Paradigm LOCATE Error Messages	Paradigm LOCATE Error messages (LOCERR.HLP)
PDREMOTE/ROM Help	PDREMOTE/ROM Tutorial help (PDREM.HLP)
Paradigm Assembler Help	Assembler options and operators reference (PASM.HLP)
Paradigm C++ SCCS Integration	Source code control system features (SCCS.HLP)
Run-time Library Source Code	Building and customizing tips (RUNTIME.HLP)
PDREMOTE/ROM Source Code	Building and customizing tips (PDREMSRC.HLP)
Paradigm OMFCVT Guide	Features of Paradigm OMFCVT (OMF.HLP)

Some of these files may only be available if you have optional components installed in the Paradigm C++ IDE. Additional files may be available.

Getting help in Paradigm C++

In Paradigm C++, you can get Help in the following ways:

- Context-Sensitive Help (*F1*)
- Contents Screens
- Index
- Keyword Search (*F1* or *Ctrl+F1* in the Edit Window)
- SpeedMenus (in the Help window)
- Contacting Paradigm

Getting context-sensitive help

To access context-sensitive Help for items in Paradigm C++:

1. Select the element you want help on (menu, menu command, an item in a dialog box).
2. Press *F1* or *Ctrl+F1*.

Help buttons are available on many dialog boxes and for most error messages.

Click Help to view information about:

- The entire dialog box
- An error message
- The current group of topics in an Options settings dialog box

Accessing and using contents screens

*To return to a previous topic or Help file, click the **Back** button.*

Each Help Contents offers an entry into a Help system installed with Paradigm C++. From the Contents, select the category of information that best suits your needs, then click on it.

- To display the Master Contents screen, choose **Contents** on the Help menu in Paradigm C++.
- To access the Help Contents from within a topic in the active Help file, click the **Contents** button.
- To access the Help Contents screen of a different Help file installed with Paradigm C++, right-click and select the name of the Help file you want to view.
- To access the Contents of all available Help files, click the Book Shelf button from within the topic of a Help file. Shortcuts to help files are also listed under the Start menu in Programs|Paradigm C++|Help.

You can expand books that appear on the Contents, or jump directly to a topic. To view a topic, click on it.

You can print several topics at once by clicking a book on the Contents and then clicking Print.

Using the index

In Help, click the Index tab to view a list of index entries. Either type the word you're looking for or scroll through the list.

Searching for keywords

Keyword Search gives you direct access to Help about a term in your program. To get help on a term:

1. In the Edit window, place the insertion point on the term you want help on.
2. Use one of the following methods:
   - Press *F1* or *Ctrl+F1*.
   - Choose **Keyword Search** on the Help menu.
   - Choose **Go To Help Topic** on the Edit Window SpeedMenu.
3. One of these events occurs:
   - The topic associated with the term you selected is displayed.

*To return to a previous topic or Help file, click the **Back** button.*

- If more than one topic is available on the term for which you requested Help, the Topics Found dialog box is displayed listing topics associated with the term. Double-click the topic you want to view.
- If no Help is available for the term nearest the insertion point, the index is displayed. You can then select a different searching method to locate a topic associated with that term. The term for which you requested Help appears highlighted in the top box. Click the **Display** button or double-click the term to view the list of topics associated with the term.

Help SpeedMenus

All the Paradigm C++ Help files have SpeedMenus that you access by right-clicking on the mouse. These menus provide quick access to commands for copying or printing a Help topic, or exiting Help.

The SpeedMenu also lists additional Help files containing information related to the current Help file. Right-click and select a Help file from the SpeedMenu. The Contents screen for that Help file is displayed.

Contacting Paradigm

There are several ways to contact Paradigm Systems for technical assistance on Paradigm C++.

Use the Help menu links to access the Paradigm C++ home page, newsgroups, FTP site or to register Paradigm C++.

You can contact Paradigm directly at:

Paradigm Systems
Suite 2214
3301 Country Club Road
Endwell, NY 13760
USA

Sales: 607-748-5966
Fax: 607-748-5968
Technical Support: 800-582-0864

Ninety days of free technical support is only available to registered users of Paradigm C++. If you haven't yet done so, take this time to register your products under the Paradigm C++ Help menu or online at http://www.devtools.com. Contact Paradigm to purchase a Paradigm SurvivalPak for an additional 12 months of free technical support and quarterly product upgrades.

Index