Paradigm C++ Professional Quick Start Guide

Paradigm Systems

December 9, 2005
Manual version 4

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

(607)748-5966
(607)748-5968 (FAX)
Sales information: sales@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.

You can call Paradigm at (607) 748-5966 to purchase a Paradigm SurvivalPak support agreement for an additional 12 months of technical support.

Table of Contents

Chapter

1

Getting started

Develop applications using the IDE or from the command line. A complete set of command tools are 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++, or 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 get ready for a productivity leap 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 its key technologies:

- About this manual
- Starting Paradigm C++
- Using SpeedMenus
- Using the Edit Window
- Working with projects
- Configuring the remote connection
- Debugging with Paradigm C++
- Productivity enhancing tools
- 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.

About this manual

This Quick Start Guide contains instructions for all versions of Paradigm C++.

The key shows typefaces and icon conventions used throughout this manual.

Boldface	Indicates language keywords and functions.
Italics	Indicates program variables, constants or other terms that need to be emphasized.
Key1	This typeface indicates a key on your keyboard.
`Monospace`	Represents text, such as code, as it appears on screen.
☞	This icon indicates material to which you should pay special attention.

This manual is intended to give a short overview of the functionality and features of Paradigm C++. Further details on topics are available within the online help system and in online .PDF manuals. The manuals require Adobe Reader to be viewed. If needed, Adobe Reader is a free download from http://www.adobe.com.

Starting Paradigm C++

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

The Paradigm C++ installation program will launch automatically from the Paradigm C++ CD, when inserted. Follow the instructions given by the installation program and have your Paradigm C++ serial number ready. When the installation is complete, the Start menu will contain a program item called Paradigm C++ Pro. Use the program item to launch Paradigm C++.

Licensing and Registration

Paradigm C++ is purchased with a single-user license agreement. This license entitles use of the product by one programmer at any given time. Each copy of Paradigm C++ comes with a licensing device intended to support the licensing agreement. See the enclosed literature for instructions on maintaining your license.

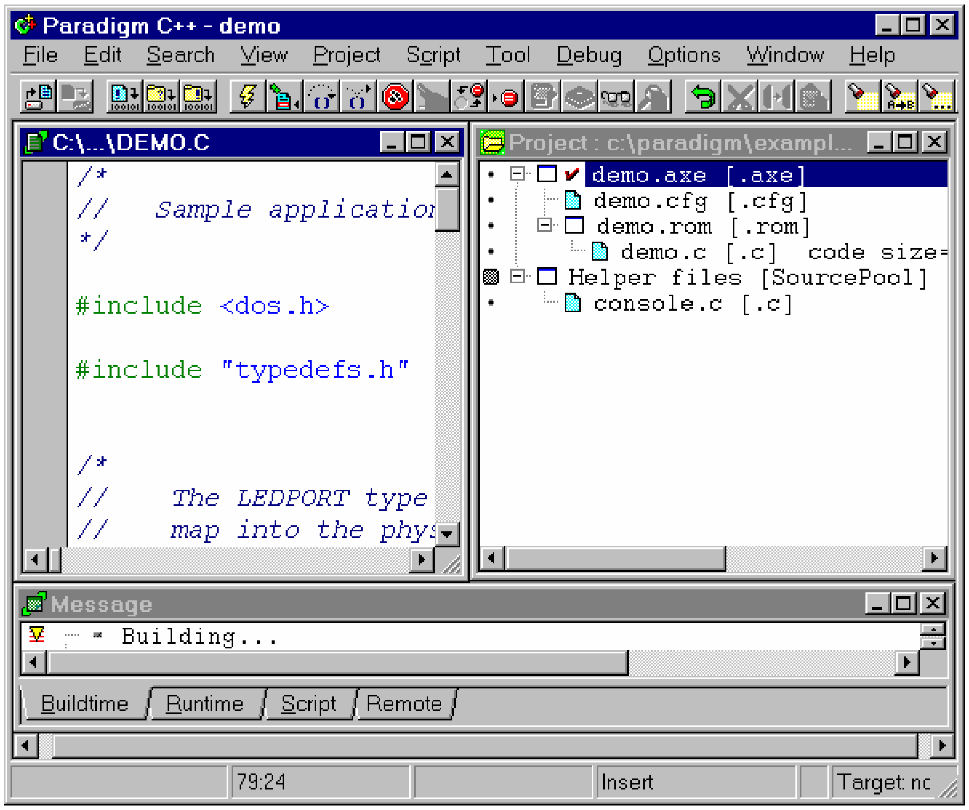
Registration of your copy of Paradigm C++ is mandatory for you to receive technical support and to be notified of changes to the software. To register your copy of Paradigm C++, visit http://www.devtools.com and select the Register button.

Starting the demo program

Example programs may vary.

Figure 1-1 shows how Paradigm C++ looks on initial startup, after it has loaded the DEMO.IDE project. 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 viewing, and the Status Bar at the very bottom which contains up-to-the minute information on the status of Paradigm C++. Filling the remainder of the window are the Edit/Debugger, Project, Message windows, where the real work of developing an embedded application will take place.

Figure 1-1 Paradigm C++ IDE screenshot


Paradigm C++ - demo
File Edit Search View Project Script Tool Debug Options Window Help
C:\...\DEMO.C
Project : c:\paradigm\exampl...
Message
Buildtime Runtime Script Remote
79:24
Insert
Target: nc

```
/*
//    Sample applicatio
*/

#include <dos.h>

#include "typedefs.h"


/*
//     The LEDPORT type
//     map into the phys
```

```
demo.axe [.axe]
  demo.cfg [.cfg]
  demo.rom [.rom]
    demo.c [.c]   code size=
Helper files [SourcePool]
  console.c [.c]
```

```
Building...
```

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 and Message window. Also contains commands to open the integrated debugger views during a debugging session.
Project	Commands to open, close, and build or make a project. Also includes source code control system commands for Paradigm C++.
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.
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.

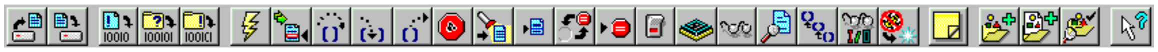
More information about using cScript is available in the online Help.

Because Paradigm C++ is fully extensible by the end-user, there may be other entries on the menu bar from 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 are working in. For example, if you are 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 22. See Figure 1-6 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 are 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've installed for extended or protected mode, your software may default to a different demo directory. Be sure to follow the real mode demo.

1. From the Paradigm C++ Menu Bar, choose Project | Open project, then select the project file DEMO.IDE in the 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 a new 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 have 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. The key presses previously described assume that you are using the Default key assignments for the editor. Paradigm C++ is supplied with five editor emulations and if these do not 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 the EXAMPLES\REAL\DEMO directory where the product was installed.
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 ;
}
```

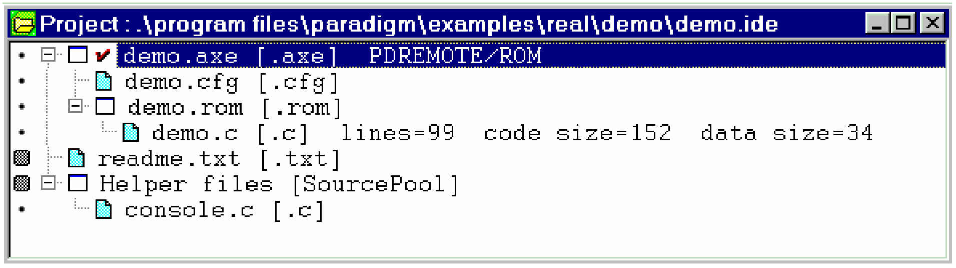
Now right-click in the window and select the Functions again and see that both main() and test() are in the list of functions in the file. No compiling, just instant access to your source code definitions to make it easy to navigate to any function, class or include file in the current Edit window.

Working 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 1-3 The Project Manager Project window


Project : .\program files\paradigm\examples\real\demo\demo.ide

```
demo.axe [.axe]  PDREMOTE/ROM
  demo.cfg [.cfg]
  demo.rom [.rom]
    demo.c [.c]  lines=99  code size=152  data size=34
readme.txt [.txt]
Helper files [SourcePool]
  console.c [.c]
```

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:
   - **a.** Type the path and name for your new project in the Project Path and Name input box. In this case, type:

     ```
     (Paradigm directory)\examples\real\demo\mydemo.ide
     ```

   - **b.** 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
     ```

   - **c.** In the Target Type list box, click Standard 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.
   - **d.** Select the desired platform and memory model you want for your application. You can also enable the use of floating point arithmetic or select other options depending on the Target Type and platform that is selected.
   - **e.** Choose a target connection from the list of available remote connection interfaces.

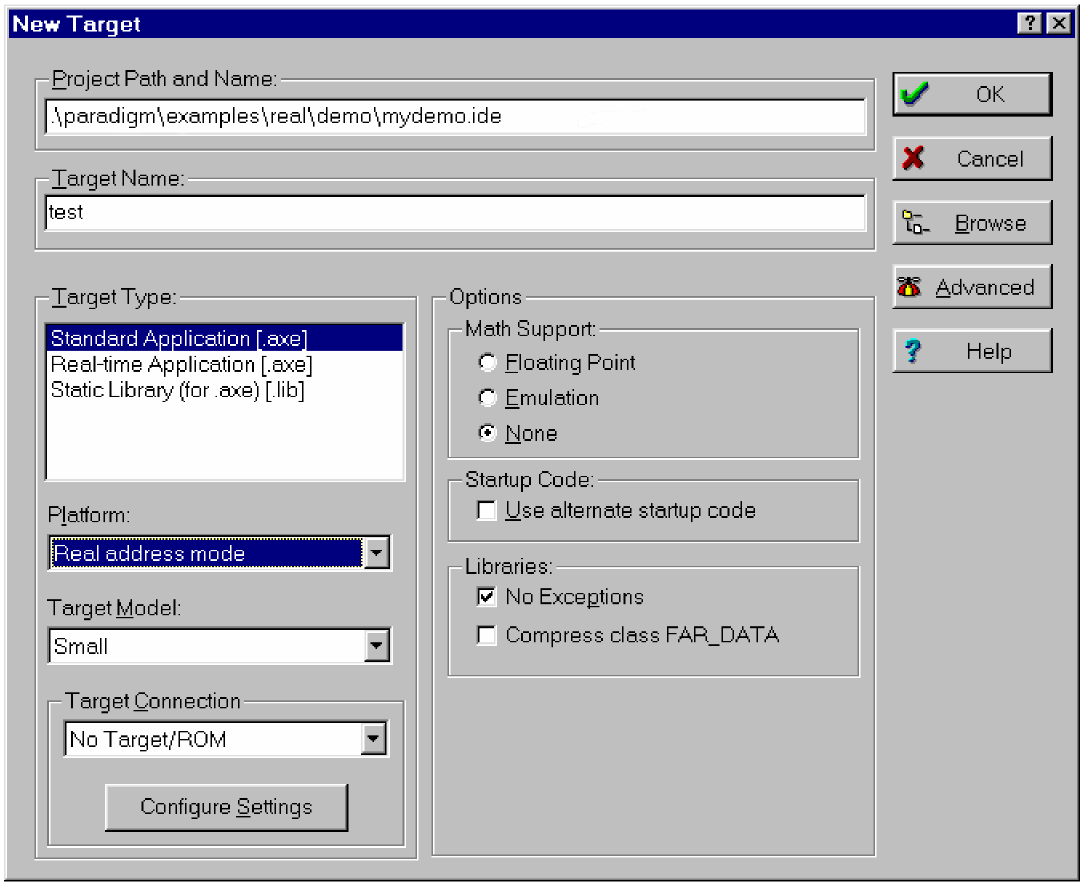
A target connection must be specified or a working PDREMOTE/ ROM kernel must be built to begin debugging.

Target connection defaults are described below. Depending on your product serial number, other hardware specific options may be available.

No Target/ROM	Select this option when you want to build and deploy your application. The Paradigm C++ IDE will generate the .HEX or binary files needed to load the application into FLASH or EPROM.
PDREMOTE/ROM	Select this option to build an application to be debugged on the target.

The New Target dialog box should resemble the one shown in Figure 1-4.

Figure 1-4 New Target dialog box



Project Path and Name: .\paradigm\examples\real\demo\mydemo.ide

Target Name: test

Target Type:
Standard Application [.axe]
Real-time Application [.axe]
Static Library (for .axe) [.lib]

Platform: Real address mode

Target Model: Small

Target Connection: No Target/ROM

Options

Math Support: ( ) Floating Point ( ) Emulation [✓ None]

Startup Code: ☐ Use alternate startup code

Libraries: ☑ No Exceptions ☐ Compress class FAR_DATA

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 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 file TEST.C (or TEST.CPP in protected mode), that you created earlier in the chapter. If you haven't already done so, create the TEST.C file by following the instructions listed in the section, “Creating a new file” on page 9).

☞ To add additional source files to a project, right-click on TEST.ROM in the Project Manager and click Add Node, then select a file name.

4. Build the application by selecting the .AXE node and right-click to bring up the local menu 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 steps 1-4 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.

5. Steps 1-4 are the basic steps to be followed when building a project in Paradigm C++. Other common actions include:
   - **a.** Double-click any .C, .CPP, .CFG, .H file in the Project Manager to pull the file into the editor.
   - **b.** Right-click on any .C or .CPP file, then select C Compile to compile the file and look for any errors or warnings in the Message window Buildtime tab.
   - **c.** To move the source files or user generated libraries around in a project, select the file in the project and press the arrow keys while holding down the ALT key.
   - **d.** Right-click on the .AXE node and select TargetExpert to change some of the more fundamental options and library support.
   - **e.** Right-click on the .AXE node and select Edit local options to change the compiler, assembler, linker, or locate options. The change will affect all of the source nodes that are used to build the .AXE node. You can edit the local options for a particular source file as well.
   - **f.** To see the specific options that have changed in the project, right-click on the .AXE node and select View options hierarchy. This can be extremely helpful in determining what options have changed.
   - **g.** To see the run-time libraries that are being linked into the project, select Options | Environment | Project View and select the Show run-time nodes checkbox.

☞ For Protected mode users, further Tutorials on developing a 32-bit application can be found in Chapter 2 of the Paradigm C++ LOCATE32 User’s Guide. See Help | Reference.

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++ User's Guide. The manual link is found in Help | Reference.

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. A dialog will open that will display the current connection, which can be found in the Target connection field.
4. Select the Target Connection drop down menu to see a list of available remote connection interfaces and select the desired remote connection interface.
5. Press the Configure 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 function **main** in the application by default. This behavior is controlled by the Run to field under Options | Environment | Debugger. To avoid running to **main**, and executing the startup code of the application automatically, clear the checkbox of the Run to field. If you wish to run to a different function, replace **main** with the appropriate function name.

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 directory)\examples\real\demo\test.axe`
3. Choose the desired remote connection interface.
4. Press the Communication options 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 function **main** in the application by default. This behavior is controlled by the Run to field under Options | Environment | Debugger. To avoid running to **main**, and executing the startup code of the application automatically, clear the checkbox of the Run to field. If you wish to run to a different function, replace **main** with the appropriate function name.
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 the EXAMPLES\REAL\DEMO directory as you did in “Creating a new file,” page 9 and double-click the DEMO.C node in the Project window. Right-click the DEMO.AXE node in the Project window and 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, is enabled, then you should something similar to Figure 1-5.

☞ Protected mode users should review the Paradigm C++ LOCATE32 User’s Guide for steps to follow in protected mode application development. A list of online manuals in .PDF format can be found on page 26 of this manual. See Help | Reference for links to the manuals.

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.

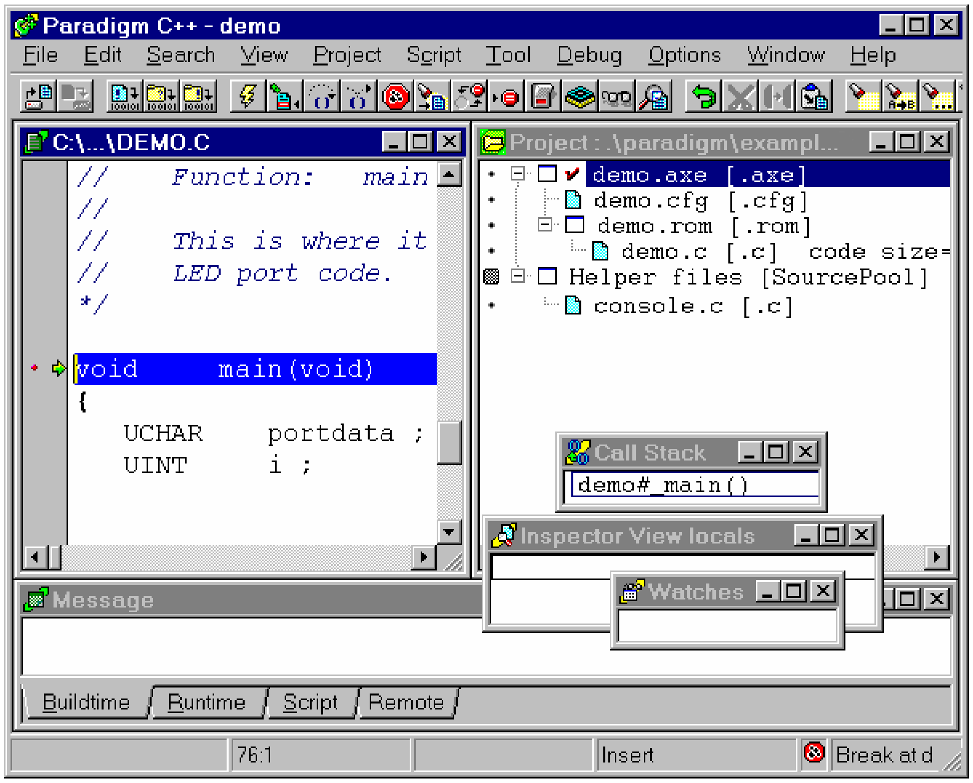
Step over/into/out of

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 1-6 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 1-5 Paradigm C++ debugging session


Paradigm C++ - demo
File Edit Search View Project Script Tool Debug Options Window Help
C:\...\DEMO.C
Project : .\paradigm\exampl...
Call Stack
Inspector View locals
Watches
Message
Buildtime Runtime Script Remote
76:1
Insert
Break at d

```
//     Function:    main
//
//     This is where it
//     LED port code.
*/

void        main(void)
{
    UCHAR     portdata ;
    UINT      i ;
```

```
demo.axe  [.axe]
  demo.cfg  [.cfg]
  demo.rom  [.rom]
    demo.c  [.c]   code size=
Helper files [SourcePool]
  console.c  [.c]
```

```
demo#_main()
```

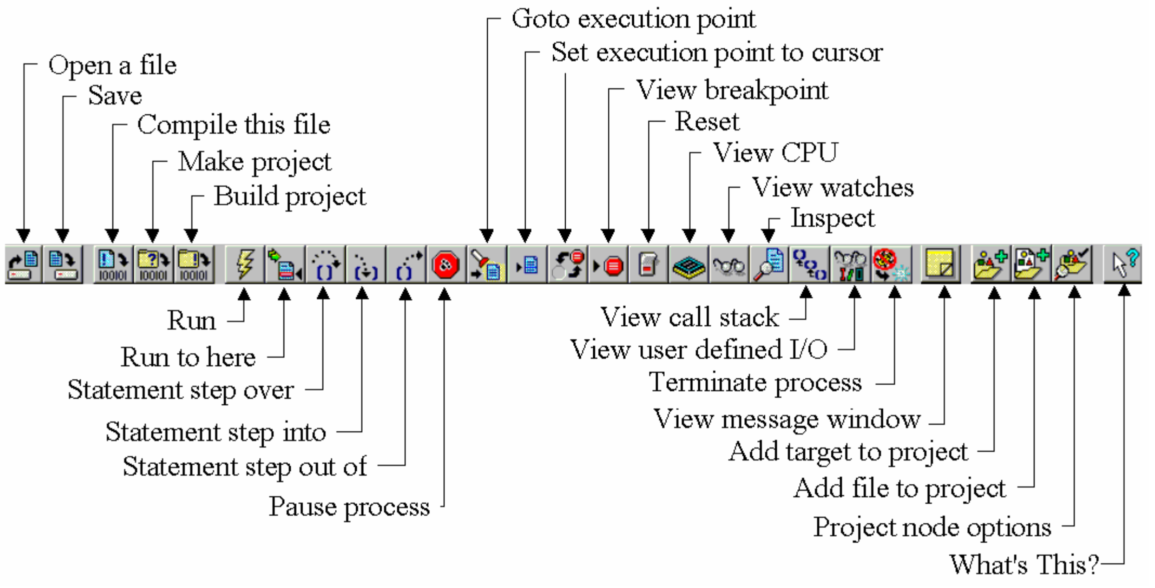
Here is an example of some common actions that can be performed in a debug session:

1. In DEMO.C, click the cursor on the variable *passcount*. Right-click to bring up the local menu and select Inspect. This will cause an Inspector window to appear for the variable *passcount*. The address of this variable is the first line of the window, followed by the name of the variable and its current value.
2. Click the mouse on the first line of the Inspector window, containing the address field. Right-click and select Dump. This will open a Dump window that will display the actual physical memory at which this variable resides at.
3. Right-click on the Dump window to view the local menu. One of the menu selections is I/O, which is needed when peeking/poking specific I/O locations on your target.
4. Another way of viewing specific I/O or memory mapped I/O locations is to select View | Target peripherals. Users can specify their own I/O locations and names associated with them by selecting the User defined I/O menu entry. The User defined I/O window can be configured by right-clicking on the window and selecting a local menu entry, such as Add to add an I/O location. These I/O entries can also be saved to .IO files for future debugging sessions.

Debugger SpeedButtons

This section will familiarize you with the Paradigm C++ SpeedButtons that can be used during a debugging session.

Figure 1-6 Paradigm C++ SpeedButtons



The Paradigm C++ debugger is covered in complete detail in Chapter 5 “Using the integrated debugger,” of the Paradigm C++ User's Guide. The manual link is found in Help | Reference or under the Start menu in Programs | Paradigm C++ Pro | Docs. We 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.

Productivity enhancing tools

There are several tools integrated into Paradigm C++ that will make writing, editing and managing code easier, thus increasing your productivity. Here are just a few.

Paradigm C++ Lint

The Paradigm C++ Lint utility can be used to perform a high level of checking on C/C++ source files. It can detect and warn of many potential problems (such as the unexpected loss of precision in a calculation) or of inefficient usage (such as including a header file that is never used). Paradigm C++ users can run the Lint utility by right-clicking on a .C or .CPP file in their project and selecting Special | Lint.

Paradigm C++ SCCS Integration

Source code control systems (SCCS) operations such as Get, Check out files, Compare differences, Show history and Check in, can be performed on files or on projects without leaving the Paradigm C++ IDE. This streamlines the software development process.

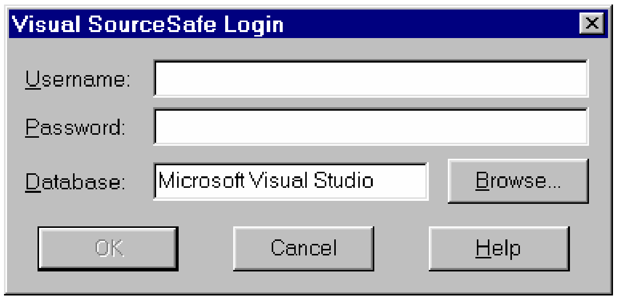
A project is a collection of related files that you store in a SCCS database. You can add, delete, edit, and share files within and among projects. Source control excels at managing team projects, and archiving and tracking old versions of source code.

To run SCCS operations, use one of the following methods:

- Select a node in the Project view and right click to select a source control action from the local menu.
- Select a source control command from the local menu of the current edit buffer in the Paradigm C++ editor.
- Choose a source control command from the Project menu.

A login dialog will appear.

Figure 1-7 Visual SourceSafe Login dialog



Visual SourceSafe Login

Username:	
Password:	
Database:	Microsoft Visual Studio

Enter your computer password to activate the source control utility.

Since there is no need to check out files to make changes, you are free to experiment with the files in your project and do the Check out only when you like the results. SCCS can maintain multiple versions of a file, including a record of all the changes to the file from one version to another. This allows you to retrieve previous versions at any time.

There are SpeedBar buttons to access most source control commands, such as Get latest version, Show history, Show differences and SCCS properties, shown in the margin. Other Source control commands can be added to the SpeedBar. See "*Customizing the SpeedBars*," page 1-22.

You can associate a Paradigm C++ project with an SCCS project using the Project | Source Control menu command.

Visual Assist

Visual Assist is aware of virtually every symbol in your project and uses this awareness to help you write your code. It will display types, functions, or defines that match a partially typed string. Simply press the *Tab* key to complete a matched string.

For example, in the Editor window, when the demo example DEMO.C, is open, typing Get followed by *Tab* will run the GetPort( ) function.

Using the *Tab* key within the Editor after a few letters of a symbol will activate a listbox of possible completions, best guesses, or member listboxes for the typed text. Try it on a macro and start saving time now.

Visual Assist can correct and highlight problems since it knows the valid symbols for every spot in your code. It's like a spell checker that switches dictionaries when you start typing in another language. Type a symbol in the wrong case and Visual Assist fixes its case immediately. Type entirely in lower case and everything is fixed. Only definitions are left untouched.

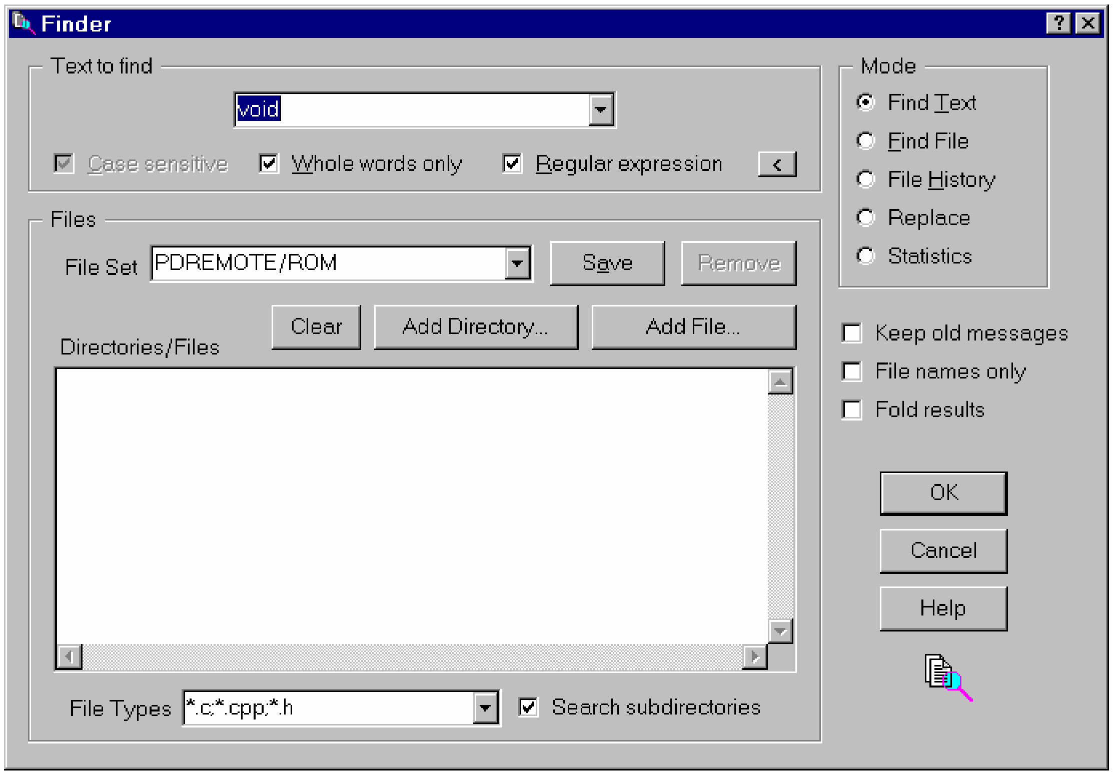
Information such as the context and definition of a symbol can be viewed immediately without losing your place in your code.

Parameter information is available for every method, function and macro in your code. Type a left parenthesis and the information will appear. Scroll when there are several parameter lists available.

Paradigm C++ Finder

The Paradigm C++ Finder utility, 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 1-8 Paradigm C++ Finder utility



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++ User's Guide. The manual link is found in Help | Reference or under the Star menu in Programs | Paradigm Pro | Docs.

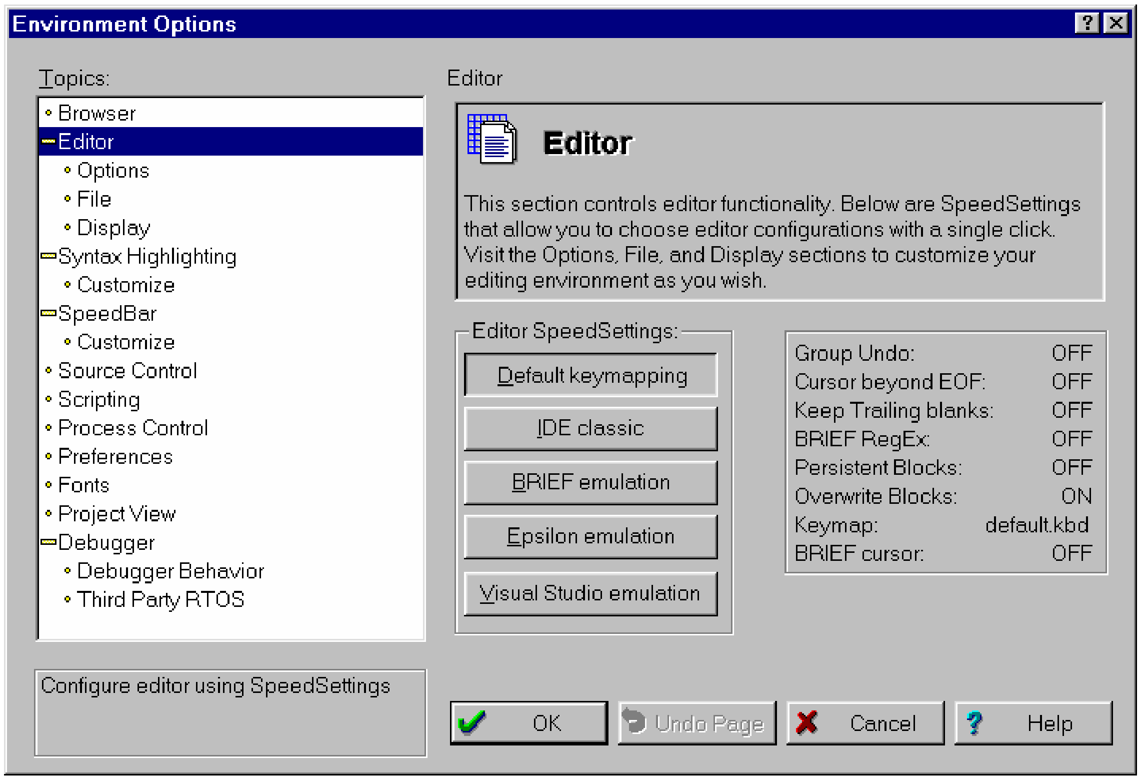
Customizing Paradigm C++

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 have 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.

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 1-9 Environment Options dialog box



This section controls editor functionality. Below are SpeedSettings that allow you to choose editor configurations with a single click. Visit the Options, File, and Display sections to customize your editing environment as you wish.

Group Undo:	OFF
Cursor beyond EOF:	OFF
Keep Trailing blanks:	OFF
BRIEF RegEx:	OFF
Persistent Blocks:	OFF
Overwrite Blocks:	ON
Keymap:	default.kbd
BRIEF cursor:	OFF

Configure editor using SpeedSettings

This section discusses the following Environment options topics:

- Configuring the Paradigm C++ editor
- Syntax highlighting

- 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

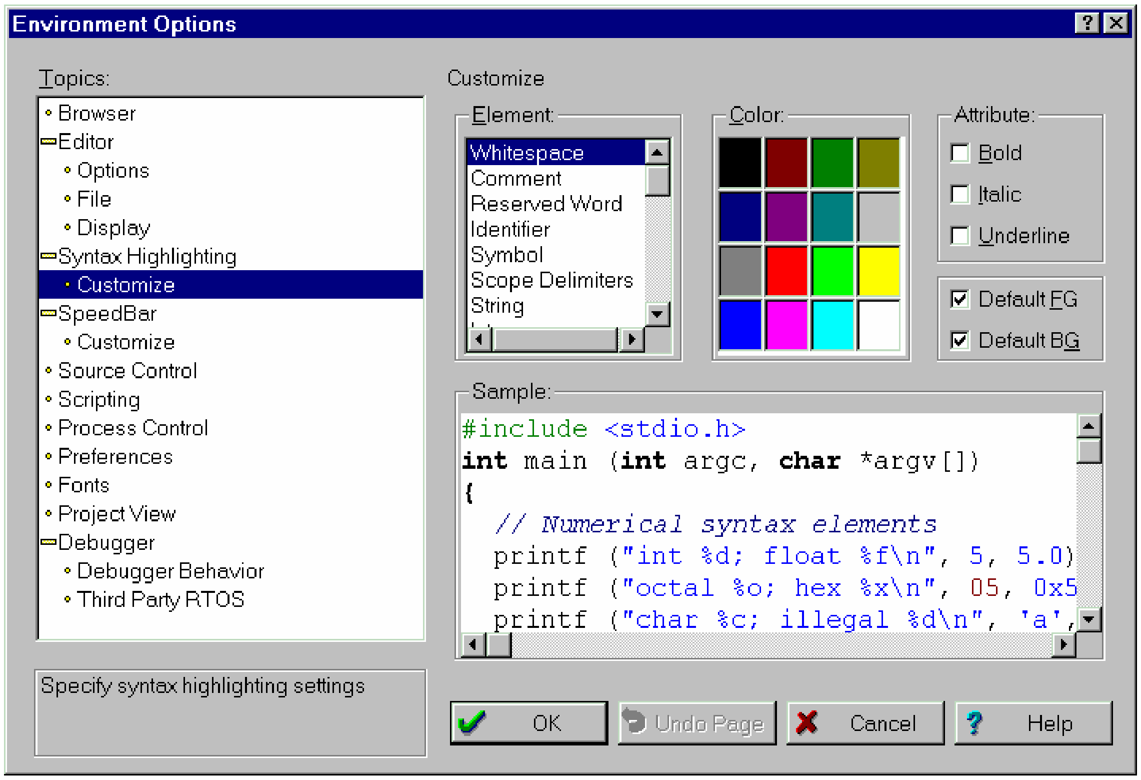
Syntax highlighting lets you define a color and font attribute (such as bold) for certain elements of code. For example, you could display comments in blue and strings in red. Syntax highlighting is on by default.

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 1-10 Environment Options Syntax Highlighting dialog



- [ ] Bold
- [ ] Italic
- [ ] Underline
- [x] Default FG
- [x] Default BG

```
#include <stdio.h>
int main (int argc, char *argv[])
{
  // Numerical syntax elements
  printf ("int %d; float %f\n", 5, 5.0)
  printf ("octal %o; hex %x\n", 05, 0x5
  printf ("char %c; illegal %d\n", 'a',
```

Specify syntax highlighting settings

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

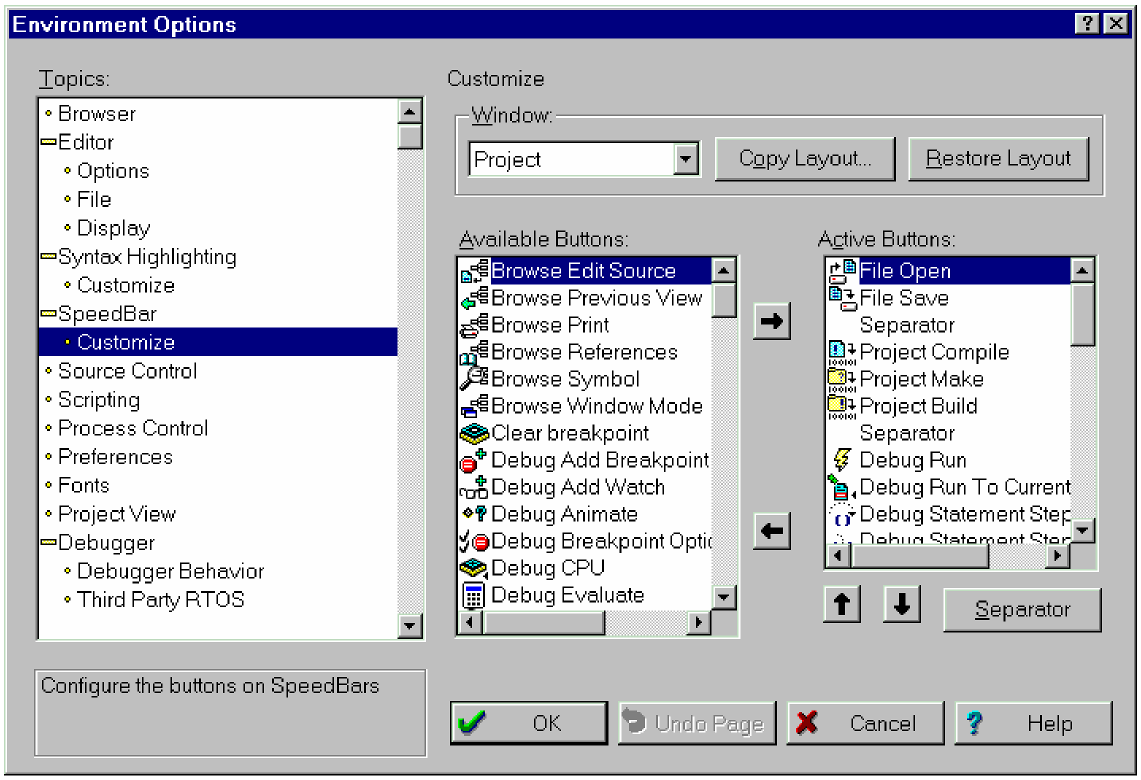
Paradigm C++ uses context-sensitive SpeedBars for all its windows, including Edit, Browser, Debugger, Project Manager, Message, and Desktop windows. When a window has focus, the corresponding SpeedBar appears just below the Menu Bar. Using the Environment Options dialog box, you can customize the SpeedBars 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 1-11 Environment Options SpeedBar customizing dialog



Window: Project

Configure the buttons on SpeedBars

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.

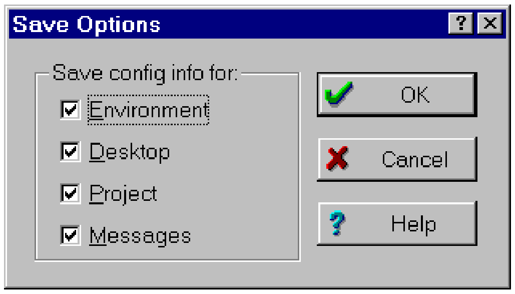
Saving your Paradigm C++ settings

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 1-12 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++. Help files can be accessed from the Help menu or from the Start menu under Programs | Paradigm C++ Pro | Help.

Online help organization

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

Table 1-2
Help files

Help file	Description
Paradigm C++ User's Guide	Guide to using projects and debugging (PCW.HLP)
Paradigm C++ Programmer's Guide	Programming tips and language details (PCPP.HLP)
Error Messages and Warnings	Paradigm C++ messages (PCERRMSG.HLP)
Paradigm C++ Tools	Command-line tools (PCTOOLS.HLP)
Paradigm C++ Locate Reference	Reference material for Paradigm C++ Locator (LOCATE.HLP)
Paradigm C++ Locate Errors	Paradigm C++ Locator Error messages (LOCERR.HLP)
Paradigm C++ Locate32 Reference	Reference for 32-bit Paradigm C++ Locator (LOCATE32.HLP)
PDREMOTE/ROM Help	PDREMOTE/ROM Tutorial help (PDREM.HLP)
Paradigm Assembler Help	Assembler options and operators reference (PASM.HLP)
ObjectScripting Guide	Customizing scripts in Paradigm C++ (SCRIPT.HLP)
Paradigm C++ Class Libraries Guide	Script class library reference (CLASSLIB.HLP)
Paradigm C++ Finder Help	Source code search utility reference (FINDER.HLP)
Version Control Integration	Source code control system reference (SCCS.HLP)
Paradigm C++ Lint Error Messages	Lint code checking utility messages (LINT.HLP)

These or other files may be available if you have optional components installed in the Paradigm C++ IDE.

Online manuals organization

A similar library of online help is available in Adobe .PDF format. The following manuals can be easily accessed in Help | Reference or from the Start menu under Programs | Paradigm C++ Pro | Docs.

Table 1-3
Online manuals

Online manual	**Description**
Quick Start Guide	Tutorial for Paradigm C++ (QKSTART.PDF)
Locate Manual	Paradigm C++ locator real mode guide (LOCATE.PDF)
PDREMOTE/ROM Manual	Real mode kernel reference (PDREMROM.PDF)
PASM Manual	Paradigm Assembler user's guide (PASMUG.PDF)
Locate32 Manual	Protected mode locator reference manual (LOCATE32.PDF)
User's Guide	Explanation of Paradigm C++ tools (PCPPUG.HLP)
Emutec PROMJet Manual	Setting up Paradigm C++ for the Emutec PROMJet (PROMJET.PDF)
GEI PromICE Manual	Setting up Paradigm C++ for the Grammar Engine PromICE (PROMICE.PDF)
Object Scripting Guide	Script customization features (CSCRIPT.PDF)
PC-Lint Reference Manual	Lint code checking reference manual (LINT.PDF)
RTKernel Manual	Paradigm RTKernel RTOS manual (RTKERNEL.PDF)
RTKernel32 Manual	Paradigm RTKernel32 RTOS manual (RTKERNEL32.PDF)

These or other online manuals may also be available if you have optional components installed in the Paradigm C++ IDE.

Help on 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++ Pro | 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 are 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.

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

3. One of these events occurs:
   - The topic associated with the term you selected is displayed.
   - 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 item to access the Paradigm C++ home page, newsgroups, and service packs. From this menu, you can also compose an email to technical support or register Paradigm C++. If you use this convenient method to contact Paradigm, your serial and version numbers will be included automatically.

You can contact Paradigm directly at:

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

Sales: 607-748-5966, sales@devtools.com
Fax: 607-748-5968
Technical Support: support@devtools.com

☞ 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 support agreement for an additional 12 months of technical support.

Index