Chapter 6.

Parallel Analyzer View Reference

This chapter describes in detail the function of each window, menu, and display in the ProDev ProMP Parallel Analyzer View's user interface. It contains the following main sections:

- Section 6.1
- Section 6.2
- Section 6.3
- Section 6.4
- Section 6.5
- Section 6.6
  - Section 6.6.1
  - Section 6.6.2
  - Section 6.6.3
  - Section 6.6.4
- Section 6.7
  - Section 6.7.1

6.1 Parallel Analyzer View Main Window

The main window is displayed when the Parallel Analyzer View begins. It consists of the following elements, shown in Figure 48:

- Main menu bar, containing these menus:
  - Admin: Discussed in Section 6.2.1.
  - Views: See Section 6.2.2.
  - Fileset: See discussion in Section 6.2.3.
  - Update: See Section 6.2.4.
  - Configuration: Find in Section 6.2.5.
  - Operations: See Section 6.2.6.
  - Help: See discussion in Section 6.2.7.
- Loop list display, which has the following members:
  - Status information: See Section 6.3.2.
  - Performance experiment information: Find in Section 6.3.2.
  - Loop list: See Section 6.3.3.
- Loop display controls, consisting of the following:
  - Search editable text field: See Section 6.4.1.

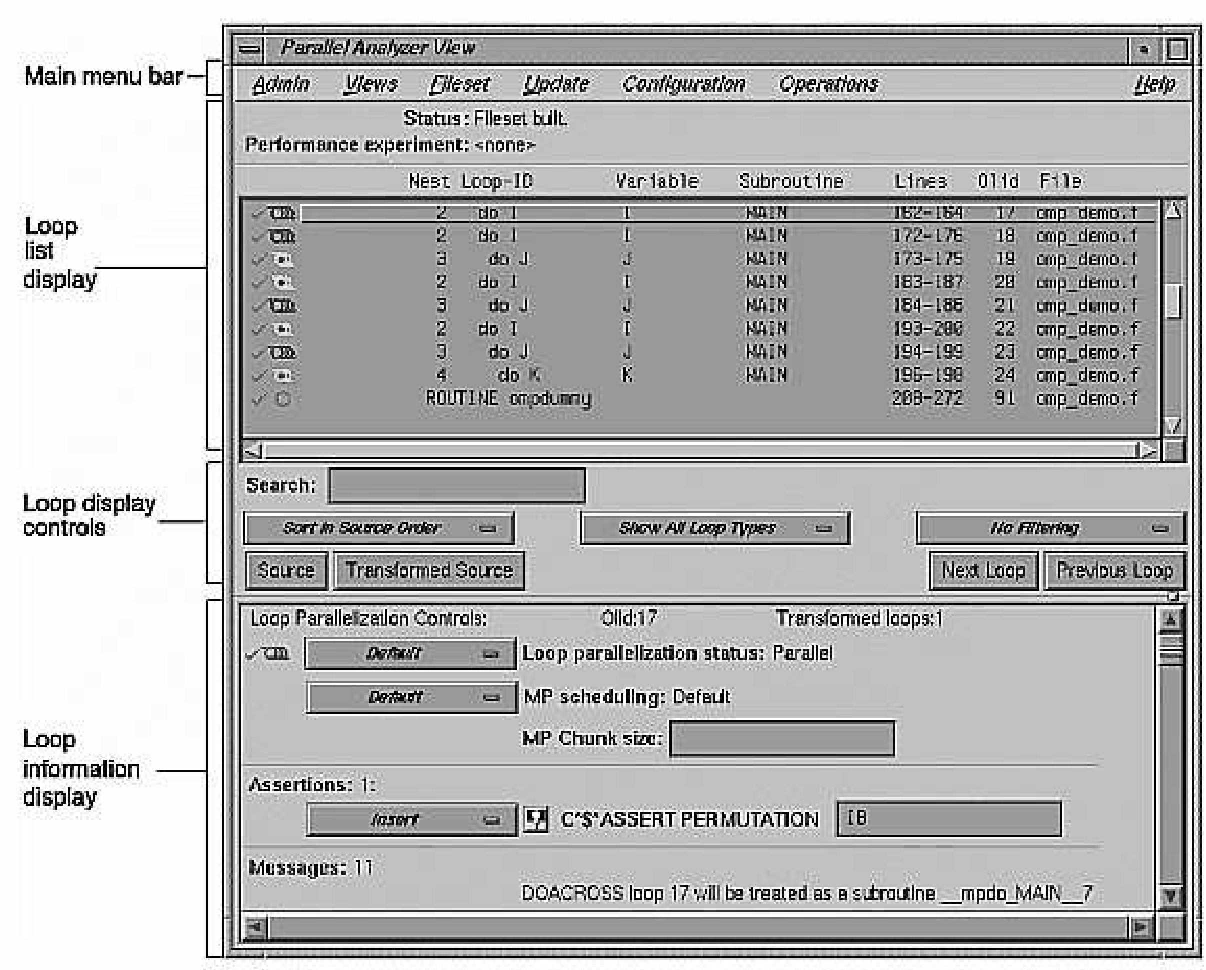
– Three option buttons displaying default values: `Sort in Source Order`, `Show All Loop Types` and `No Filtering`. These buttons are described in Section 6.4.2, Section 6.4.3, and Section 6.4.4.

– `Source` and `Transformed Source control` buttons: See Section 6.4.5.

– `Next Loop`and `Previous Loop` loop list navigation buttons: See description in Section 6.4.5.

· Loop information display: See Section 6.5.

Figure 48. Parallel Analyzer View Main Window



6.2 Parallel Analyzer View Menu Bar

This section describes the menus found in the menu bar located at the top of the Parallel Analyzer View main window as shown in Figure 49. The menus are discussed in these sections:

· Section 6.2.1

· Section 6.2.2

· Section 6.2.3

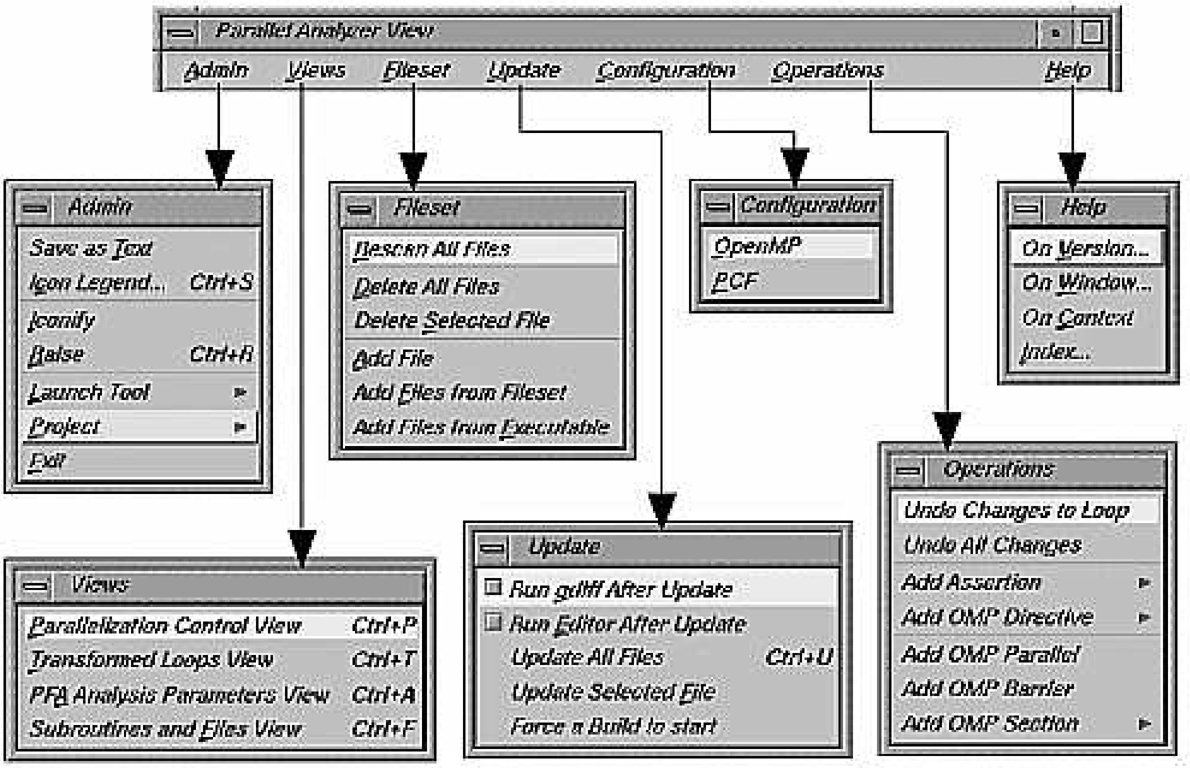
· Section 6.2.4

· Section 6.2.5

· Section 6.2.6

· Section 6.2.7

Figure 49. Parallel Analyzer View Menu Bar and Menus



Within each menu, the names of some options are followed by keyboard shortcuts, which you can use instead of the mouse for faster access to these options. For a summary, see Section 6.2.8.

You can tear off a menu from the menu bar, so that it is displayed in its own window with each menu command visible at all times, by selecting the dashed line at the top of the menu (the first item in each of the menus). Submenus can also be torn off and displayed in their own window.

6.2.1 Admin Menu

Figure 50, shows the Parallel Analyzer View Admin menu, which contains file−writing commands, other administrative commands, and commands for launching and manipulating other WorkShop application views.

Figure 50. Admin Menu

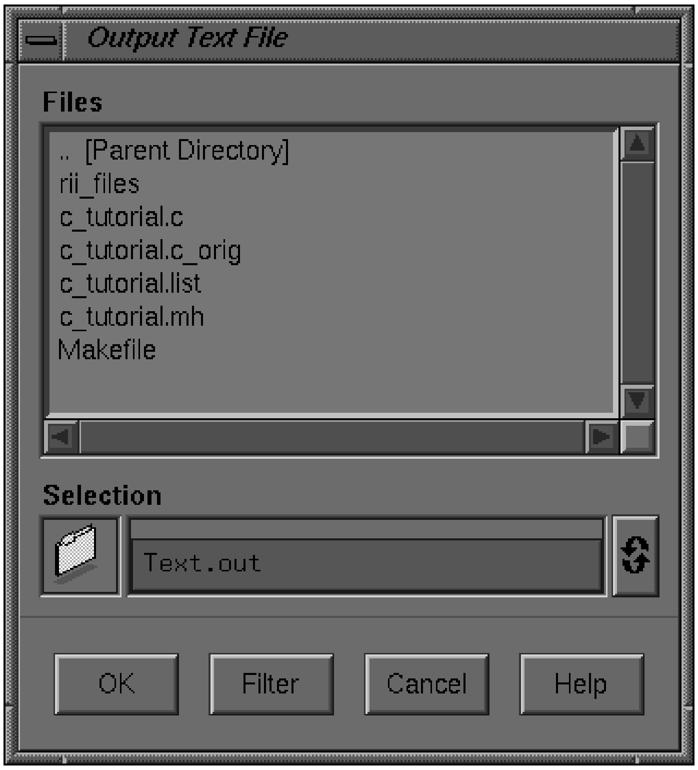
The commands in the Admin menu have the following effects:

Save as Text

Saves the complete loop information for all files and subroutines in the current session in a plain ASCII file. Choosing Admin > Save as Text brings up a File Selection dialog, which lets you choose where to save the file and what name to call it. (See Figure 51.)

The default directory is the one from which you invoked the Parallel Analyzer View; the default filename is `Text.out`. The Parallel Analyzer View asks for confirmation before overwriting an existing file.

Figure 51. Output Text File Selection Dialog



Icon Legend...

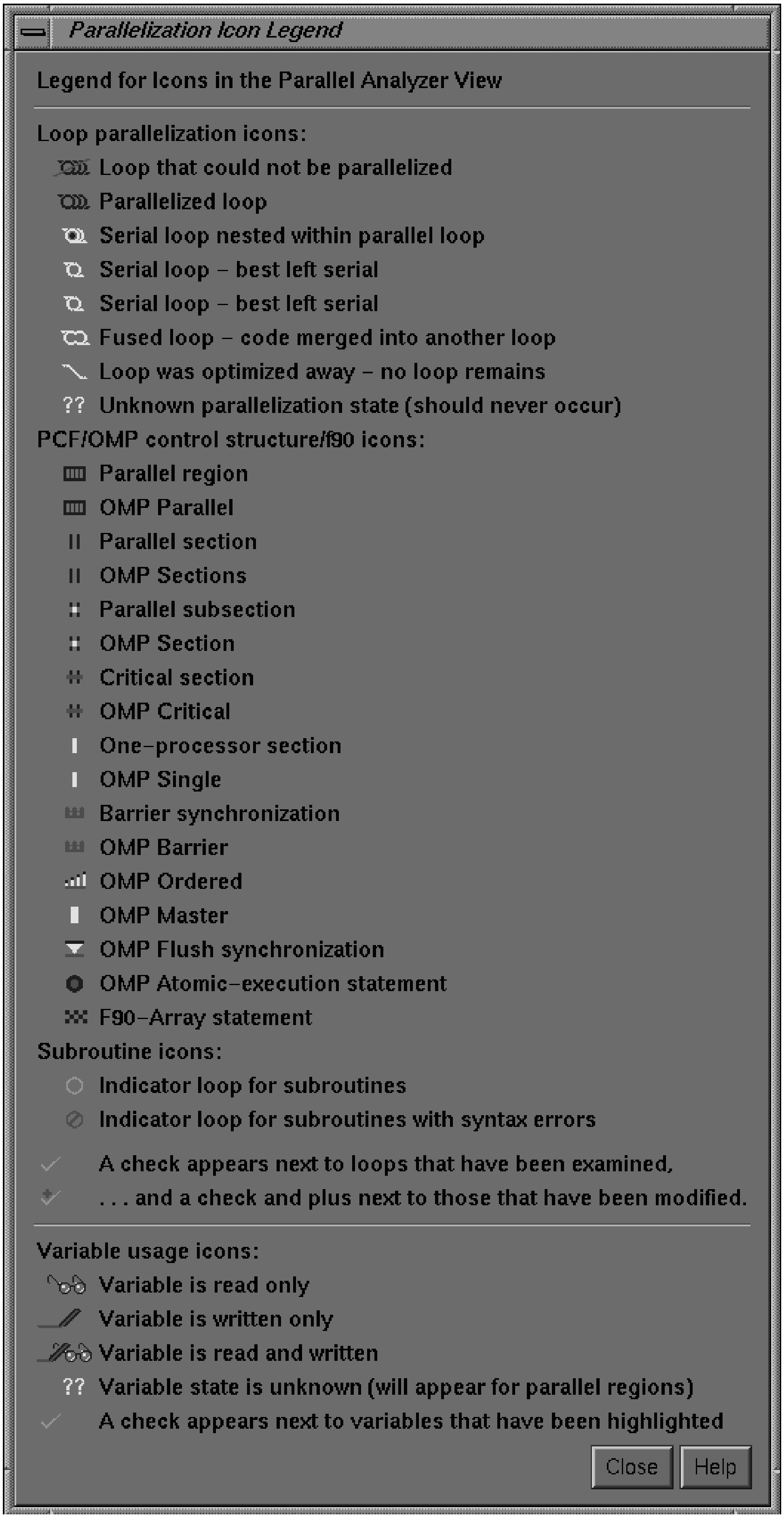
Provides an explanation of the graphical icons used in several of the views. Shortcut: **`Ctrl+S`**. See Section 6.2.1.1.

Iconify

Stows all the open windows belonging to a given invocation of the Parallel Analyzer View as icons in the style of the window

	manager you are using.
Raise	Brings all open windows in the current session to the foreground of the screen, in front of other windows. The command also opens any previously iconified windows belonging to the invocation of the Parallel Analyzer View and brings them to the foreground. Shortcut: **`Ctrl+R`**.
Launch Tool	Opens various WorkShop tools. See Section 6.2.1.2.
Project	Controls project windows. See Section 6.2.1.3.
Exit	Quits the current session of the Parallel Analyzer View, closing all windows. If you have not updated source files and have pending requests for changes, a dialog box asks if it is OK to discard the changes. Click `OK` only if you want to **discard** any changes; otherwise, click `Cancel` to update the files.

Figure 52. Parallelization Icon Legend


Parallelization Icon Legend
Legend for Icons in the Parallel Analyzer View
Loop parallelization icons:
Loop that could not be parallelized
Parallelized loop
Serial loop nested within parallel loop
Serial loop – best left serial
Serial loop – best left serial
Fused loop – code merged into another loop
Loop was optimized away – no loop remains
?? Unknown parallelization state (should never occur)
PCF/OMP control structure/f90 icons:
Parallel region
OMP Parallel
Parallel section
OMP Sections
Parallel subsection
OMP Section
Critical section
OMP Critical
One–processor section
OMP Single
Barrier synchronization
OMP Barrier
OMP Ordered
OMP Master
OMP Flush synchronization
OMP Atomic–execution statement
F90–Array statement
Subroutine icons:
Indicator loop for subroutines
Indicator loop for subroutines with syntax errors
A check appears next to loops that have been examined,
. . . and a check and plus next to those that have been modified.
Variable usage icons:
Variable is read only
Variable is written only
Variable is read and written
?? Variable state is unknown (will appear for parallel regions)
A check appears next to variables that have been highlighted
Close
Help

6.2.1.1 Icon Legend... Option

This Admin menu option opens the Parallelization Icon Legend (Figure 52) which provides the meanings of the icons that appear in various views, such as the following:

- Parallel Analyzer View, shown in Figure 1
- Transformed Loops View, shown in Figure 75
- Subroutines and Files View, shown in Figure 77
- Parallelization Control View, shown in Figure 72

6.2.1.2 Launch Tool Submenu

The Admin menu's Launch Tool submenu contains commands for launching other WorkShop tools, as well as new sessions of the Parallel Analyzer. (See Figure 53.)

To work properly with the other WorkShop tools, the files in the current fileset must have been loaded into the Parallel Analyzer from an executable. There are two ways to do this:

- Use the **–e** option on the command line. (See Section 1.2.)
- Choose the Fileset > Add File menu option. (See Section 6.2.3.)

If you launch Workshop tools from a session not based on an executable, the tools start without arguments.

Figure 53. Launch Tool Submenu

The following six options launch applications from the Launch Tool submenu:

Build Analyzer	Launches the Build Manager, a utility that lets you compile software without leaving the WorkShop environment. For more information, see Appendix B, "Using the Build Manager," in the *Developer Magic: Debugger User's Guide*.
Debugger	Launches the WorkShop Debugger, a UNIX® source–level debugging tool that provides special windows for displaying program data and execution status. For more information, see Chapter 1, "Getting Started with the WorkShop Debugger," in the *Developer Magic: Debugger User's Guide.*
Parallel Analyzer	Launches another session of the Parallel Analyzer.
Performance Analyzer	Launches the Performance Analyzer, a utility that collects performance data and allows you to analyze the results of a test run. For more information, see the *Developer Magic: Performance Analyzer User's Guide.*

Static Analyzer	Launches the Static Analyzer, a utility that allows you to analyze and display source code written in C, C++, Fortran, or Ada. For more information, see the *Developer Magic: Static Analyzer User's Guide*.
Tester	Launches the Tester, a UNIX−based software quality assurance tool set for dynamic test coverage over any set of tests. For more information, see the *Developer Magic: Performance Analyzer User's Guide*.

If any of these tools is not installed on your system, the corresponding menu item is grayed out.

If the file `/usr/lib/WorkShop/system.launch` is absent (that is, if you are running the Parallel Analyzer View without WorkShop 2.0 installed), the entire Launch Tool submenu is grayed out.

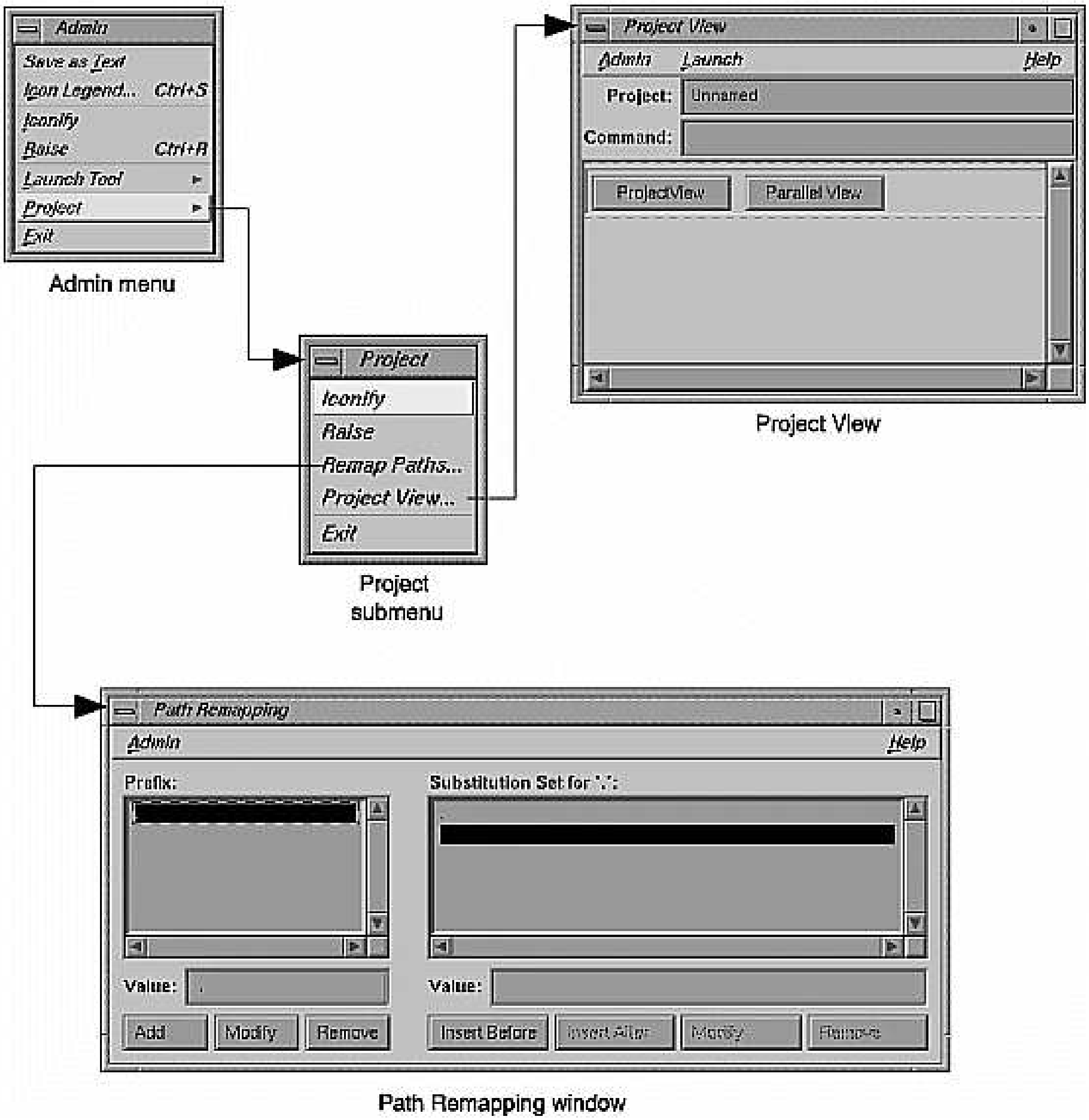
6.2.1.3 Project Submenu

The Project submenu of the Admin menu contains commands that affect all the windows containing WorkShop or ProDev ProMP applications that have been launched to manipulate a single executable. The set of windows is a WorkShop **project**. The Project submenu and windows that you can open from it are shown in Figure 54.

The Project submenu commands are as follows:

Iconify	Stows all the windows in the current project as icons, in the style of the window manager you are using.
Raise	Brings all open windows in the current project to the foreground of the screen, in front of other windows. The command also opens any previously iconified windows in the current project and brings them to the foreground.
Remap Paths...	Lets you modify the set of mappings used to redirect references to filenames located in your code to their actual locations in your file system. However, if you compile your code on one tree and mount it on another, you may need to remap the root prefix to access the named files.
Project View...	Launches the WorkShop Project View, a tool that helps you manage project windows.
Exit	Quits the current project, closing all windows, including those of related open applications. Thus the Source View closes, as well as, for example, the Parallel Analyzer. If you have not updated source files and have pending requests for changes, a dialog box asks if it is OK to discard the changes. Click `OK` only if you want to discard any changes; otherwise, click `Cancel` and update the files.

Figure 54. Project Submenu and Windows



6.2.2 Views Menu

The Views menu of the Parallel Analyzer View (Figure 55) contains commands for launching a variety of secondary windows, or **views**, that provide specific sets of information about, and tools to apply to, selected loops.

Figure 55. Views Menu

The options in the Views menu have the following effects:

Parallelization Control View

Opens a Parallelization Control View for the loop currently selected from the loop list display. Shortcut: `Ctrl+P`. For more information on this view, see Section 6.6.1.

Transformed Loops View

Opens a Transformed Loops View for the loop currently selected from the loop list display. Shortcut: `Ctrl+T`. For more information on this view, see Section 6.6.2.

PFA Analysis Parameters View

Opens the PFA Analysis Parameters View, which provides a means of modifying a variety of PFA parameters. Shortcut: `Ctrl+A`. This view is further described in Section 6.6.3.

Subroutines and Files View

Opens the Subroutines and Files View, which provides a complete list of subroutine and file names being examined within the current session of the Parallel Analyzer View. Shortcut: `Ctrl+F`. This view is further described in Section 6.6.4.

6.2.3 Fileset Menu

The Fileset menu (Figure 56) contains commands for manipulating the files displayed by the Parallel Analyzer View. A *fileset* is a list of source filenames contained in an ASCII file, each on a separate line.

Figure 56. Fileset Menu

The options in the Fileset menu have the following effects:

Rescan All Files	The Parallel Analyzer View checks and updates all the source files loaded into its current session so they match the versions of those files in the file system. The Parallel Analyzer View rereads only the files it needs to.
Delete All Files	Removes all files from the current session of the Parallel Analyzer View. You can then add new files using the Add File, Add Files from Fileset, or Add Files from Executable options,

described below.

Delete Selected File

Deletes a selected file from the current session of the Parallel Analyzer View. To select a file for deletion, open the Subroutines and Files View and double−click the desired filename.

Add File

Adds a new source file to the current session of the Parallel Analyzer View. Selecting this command brings up a File Selection dialog that lets you select a Fortran source file.

Before you can select a given source file, you must compile it to create the `.anl` file needed by the Parallel Analyzer View. (See Section 1.2.1.)

If the current session is based on an executable, you cannot add files to it until you have deleted the executable's fileset. (See the Add Files from Executable option, described below.)

Add Files from Fileset

Lets you add a list of new source files to the current session of the Parallel Analyzer View. Choosing this command brings up a File Selection dialog as it does for the Add File option. If you select a file containing a fileset list, all Fortran source files in the list are loaded into the current session (other files in the list are ignored).

If the current session is based on an executable, you cannot add files to it until you have deleted the executable's fileset.

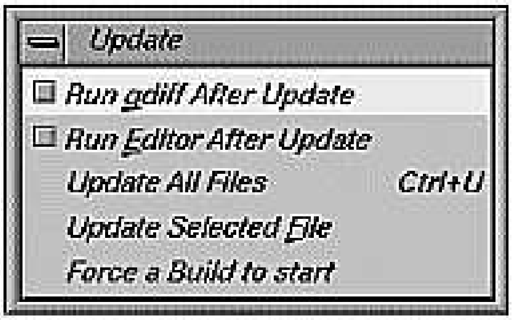
Add Files from Executable

Imports all the Fortran source files listed in the symbol table of a compiled Fortran application. This command works only if there are no files in the current session of the Parallel Analyzer View when the command is selected from the menu. Selecting this command brings up a File Selection dialog as it does for the Add File option. Other WorkShop applications can also operate on files imported from an executable.

6.2.4 Update Menu

The Parallel Analyzer View Update menu (Figure 57) contains commands for placing requested changes to directives and assertions in your Fortran source code.

Figure 57. Update Menu



The options in the Update menu have the following effects:

Run gdiff After Update

Sets a checkbox that causes a `gdiff` window to open after you have updated changes to your source file. This window illustrates in a graphical manner the differences between the unchanged source and the newly updated source.

If you always wish to see the gdiff window, you may set the resource in your .Xdefaults file:

```
cvpav*gDiff: True
```

For more information on using gdiff, see the man page for gdiff(1).

Run Editor After Update
Sets a checkbox that opens an xwsh shell window with the vi editor on the updated source file.

If you always wish to run the editor, you can set the resource in your .Xdefaults file:

```
cvpav*runUserEdit: True
```

If you prefer a different window shell or a different editor, you can modify the resource in your .Xdefaults file and change from xwsh or vi as you prefer. The following is the default command in the .Xdefaults, which you can edit for your preference:

```
cvpav*userEdit: xwsh -e vi %s +%d
```

In the above command, the **+%d** tells vi at what line to position itself in the file and is replaced with **1** by default (you can also omit the **+%d** parameter if you wish). The edited file's name either replaces any explicit **%s**, or if the **%s** is omitted, the filename is appended to the command.

Update All Files
Writes to the appropriate source files all changes to loops requested during the current session of the Parallel Analyzer View. Shortcut: **Ctrl+U**.

Update Selected File
Writes to a selected file changes to loops requested during the current session of the Parallel Analyzer View. You choose a file for updating by double–clicking in the Subroutines and Files View the line corresponding to the desired filename. (See also Section 6.6.4.)

Force a Build to start
Performs the Update All Files option and starts a build.

6.2.5 Configuration Menu

The Configuration menu (Figure 58) allows you to choose between having the Parallel Analyzer View use OpenMP or PCF directives.

Figure 58. Configuration Menu

The options are the following:

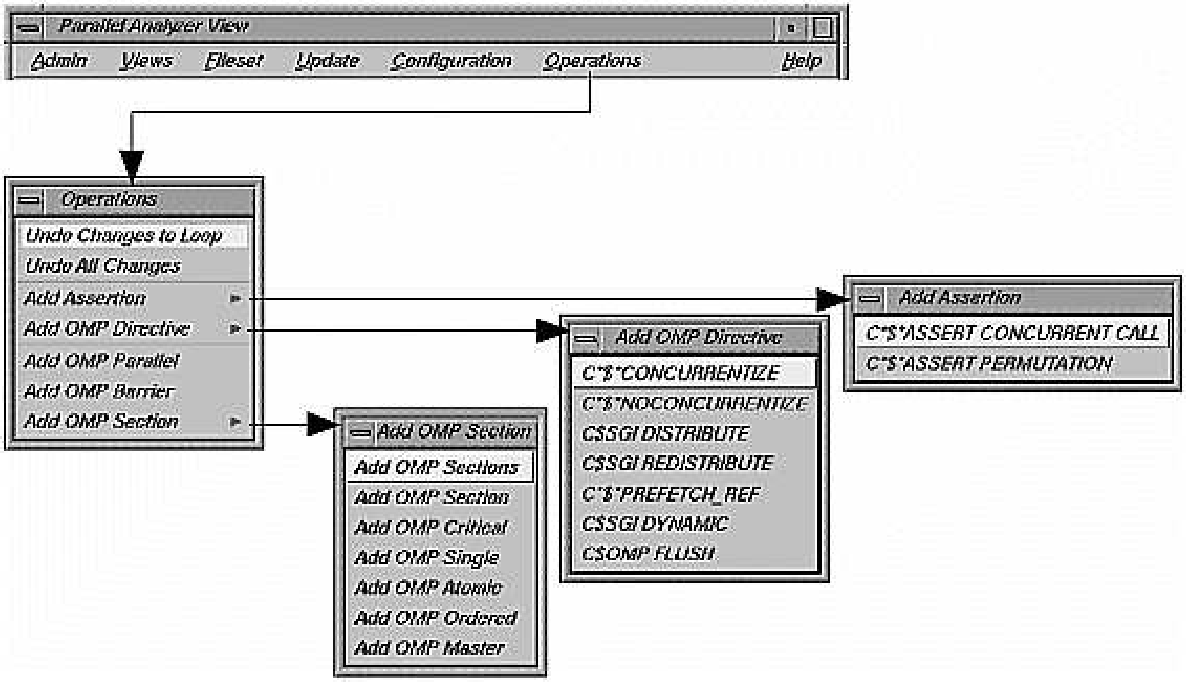
OpenMP
Causes the Parallel Analyzer View to use OpenMP directives.

PCF
Causes the Parallel Analyzer View to use PCF directives.

6.2.6 Operations Menu

The Parallel Analyzer View Operations menu contains commands for adding assertions and directives to loops, and removing pending changes to source files (Figure 59). The general effects of the Operations menu options are to prepare a set of requested changes to your source code. For information on how these changes are subsequently performed see Section 6.2.4.

Figure 59. Operations Menu and Submenus



The Operations menu is one of two points in the Parallel Analyzer View where you can add assertions and directives. The other point is discussed in Section 6.5.2. These two menus focus on different aspects of the parallelization task:

- The Operations menu focuses on automatic parallelization directives, which may be inserted in code by the MIPSpro Auto–Parallelizing Option, and memory distribution.
- The parallelization controls in the loop information display focus on manual (that is, not automatic) parallelization controls, which you can insert to further parallelize your code.

The assertions and directives you can add from the Operations menu are listed in two tables. Table 3, contains a list of directives and assertions for parallelizing code that can be added with the Add Assertion and Add OMP Directive menus. Table 4, lists directives that are available from the Add OMP Section menu and are used to synchronize access to sections of code by threads.

Table 3. Add Assertion and Add OMP Directive Menu Options

Option	Effect on Compilation	For More Information
`C*$* ASSERT CONCURRENT CALL`	Ignore dependences in subroutine calls that would inhibit parallelizing.	*MIPSpro Auto–Parallelizir Option Programmer's Guic* Chapter 3
`C*$* ASSERT PERMUTATION`	Array *array_name* is a permutation array.	*MIPSpro Auto–Parallelizir*

(*array_name*)		*Option Programmer's Guid* Chapter 3
C*$* CONCURRENTIZE	Selectively override C*$* NOCONCURRENTIZE. Typically inserted during automatic parallelization.	*MIPSpro Auto−Parallelizin Option Programmer's Guid* Chapter 3
C*$* NOCONCURRENTIZE	Do not parallelize file subroutine (depending on placement). Typically inserted during automatic parallelization.	*MIPSpro Auto−Parallelizin Option Programmer's Guid* Chapter 3
C$SGI DISTRIBUTE C$SGI REDISTRIBUTE	Distribute array storage among processors. For Origin2000 systems.	*MIPSpro Fortran 77 Programmer's Guide*, Chap
C*$* PREFETCH_REF	Load data into cache. May be used with nonconcurrent code.	*MIPSpro Compiling and Performance Tuning Guide* Chapter 4
C$SGI DYNAMIC	Allow run−time array redistribution. For Origin2000 systems.	*MIPSpro Fortran 77 Programmer's Guide*, Chap
C$OMP FLUSH	Identifies synchronization points at which the implementation is required to provide a consistent view of memory.	*OpenMP Fortran Applicat Program Interface*, see http://www.openmp.org

The options in the Operations menu have the following specific effects:

Undo Changes to Loop
: Removes pending changes to the currently selected loop. Changes that have already been written to the source file using the Update menu commands cannot be undone.

Undo All Changes
: Removes pending changes to all the loops in the current fileset. Changes that have already been written to the source file using the Update menu commands cannot be undone.

Add Assertion
: Opens the Add Assertion menu which allows you to add the following assertions, which are described in Table 3:

- C*$*ASSERT CONCURRENT CALL
- C*$*ASSERT PERMUTATION

Add OMP Directive
: Opens the Add OMP Directive menu which allows you to add these directives, described in Table 3:

- C*$* CONCURRENTIZE
- C*$* NOCONCURRENTIZE
- C$SGI DISTRIBUTE (formerly C*$* DISTRIBUTE)
- C$SGI REDISTRIBUTE (formerly C*$* REDISTRIBUTE)
- C*$* PREFETCH_REF
- C$SGI DYNAMIC (formerly C*$* DYNAMIC)
- C$OMP FLUSH

Add OMP Parallel
: Allows you to add the C$OMP PARALLEL directive. The directive defines a parallel region, that is a block of code that is to be executed by multiple threads in parallel.

Add OMP Barrier
: Allows you to add the C$OMP BARRIER synchronization directive. This directive causes each thread to wait at the designated point until all have reached it.

Add OMP Section	Opens the Add OMP Section submenu whose seven options allow you to add the OpenMP synchronization directives shown below. The directives are explained in Table 4. ·Add OMP Sections: `C$OMP SECTIONS` ·Add OMP Section: `C$OMP SECTION` ·Add OMP Critical: `C$OMP CRITICAL` ·Add OMP Single: `C$OMP SINGLE` ·Add OMP Atomic: `C$OMP ATOMIC` ·Add OMP Ordered: `C$OMP ORDERED` ·Add OMP Master: `C$OMP MASTER`

To use the Add OMP Section option do the following:

1. Bring up the Source View.
2. Using the mouse, sweep out a range of lines for the new construct.
3. Invoke the appropriate menu item to add the new construct.

When you add a new OMP Section construct, the list is redrawn with the new construct in place, and the new construct is selected. Brackets defining the new constructs are **not** added to the file loop annotations.

Table 4, lists the directives that can be added with the Add OMP Section menu. A more detailed explanation of them can be found in the document *OpenMP Fortran Application Program Interface* located at Web site of the OpenMP Architecture Review Board, http://www.openmp.org.

Table 4. Add OMP Section Menu Options

Option	**Meaning**
`C$OMP SECTIONS`	Specifies that the enclosed sections of code are to be divided a threads in a team.
`C$OMP SECTION`	Delineates a section within `C$OMP SECTIONS`.
`C$OMP CRITICAL`	Restrict access to enclosed code to one thread at a time.
`C$OMP SINGLE`	Only one thread executes the enclosed code
`C$OMP ATOMIC`	Update memory location atomically, not simultaneously.
`C$OMP ORDERED`	Execute enclosed code in same order as sequential execution.
`C$OMP MASTER`	Specify code to be executed by master thread.

Note: The Parallel Analyzer does not enforce any of the semantic restrictions on how parallel regions and or sections must be constructed. When you add nested regions or constructs, be careful that they are properly nested: they must each begin and end on distinct lines. For example, if you add a parallel region and a nested critical section that end at the same line, the terminating directives are not in the correct order.

6.2.7 Help Menu

The Help menu contains commands that allow you to access online information and documentation for the Parallel Analyzer View. (See Figure 60.)

Figure 60. Help Menu

The options in the Help menu have the following effects:

On Version...	Opens a window containing version number information for the Parallel Analyzer View.
On Window...	Invokes the Help Viewer, which displays a descriptive overview of the current window or view and its graphical user interface.
On Context	Invokes context−sensitive help. When you choose this option, the normal mouse cursor (an arrow) is replaced with a question mark. When you click on graphical features of the application with the left mouse, or position the cursor over the feature and press the **F1** key, the Help Viewer displays information on that context.
Index...	Invokes the Help Viewer and displays the list of available help topics, which you can browse alphabetically, hierarchically, or graphically.

6.2.8 Keyboard Shortcuts

Table 5, lists the keyboard shortcuts available in the Parallel Analyzer View:

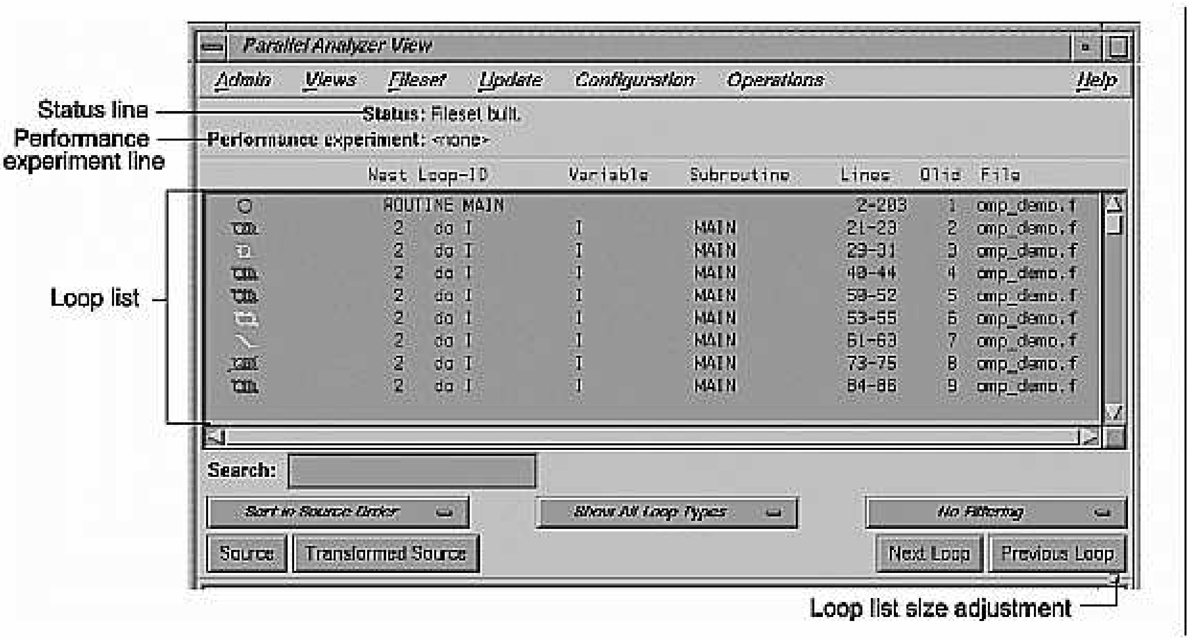
Table 5. Parallel Analyzer View Keyboard Shortcuts

Shortcut	**Menu**	**Menu Option**
Ctrl+S	Admin	Icon Legend...
Ctrl+R	Admin	Raise
Ctrl+P	Views	Parallelization Control View
Ctrl+T	Views	Transformed Loops View
Ctrl+A	Views	PFA Analysis Parameters View
Ctrl+F	Views	Subroutines and Files View
Ctrl+U	Update	Update All Files

6.3 Loop List Display

This section describes the loop list display and the various option buttons and fields that manipulate the information shown in the loop list display, shown in Figure 61.

Figure 61. Loop List Display



6.3.1 Resizing the Loop List

You can resize the loop list to change the number of loops displayed; use the adjustment button: a small square below the `Previous Loop` button.

6.3.2 Status and Performance Experiment Lines

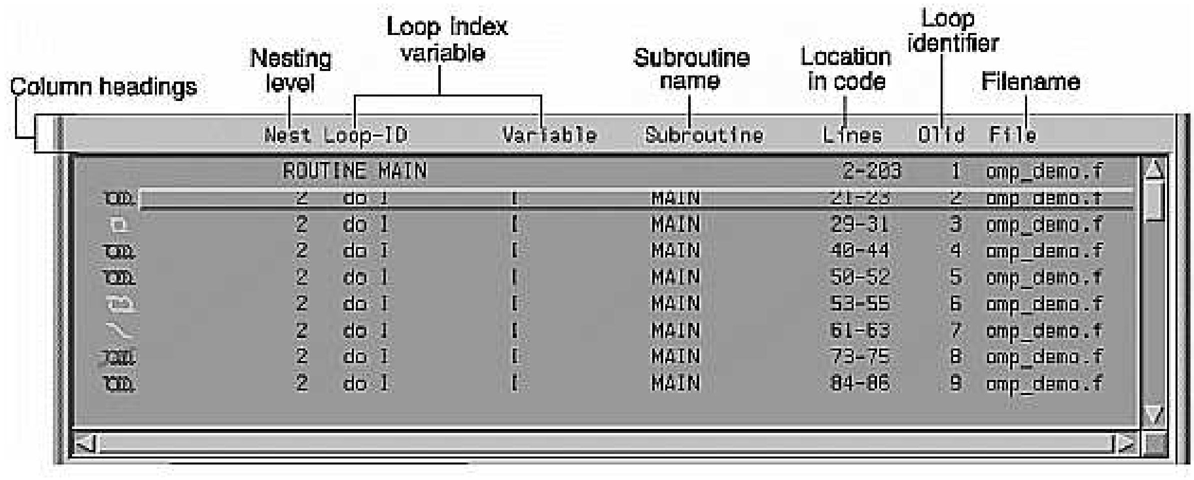
The Status line displays messages about the current status of the loop list, providing feedback on manipulations of the current fileset.

The Performance experiment line is meaningful if you run the WorkShop Performance Analyzer. The line displays the name of the current experiment directory and the type of experiment data, as well as total data for the current caliper setting in the Performance Analyzer. (See Section 6.2.1.2, for information on invoking the Performance Analyzer from the Parallel Analyzer View.) If the Performance Analyzer is not being used, the performance experiment line displays <none>.

6.3.3 Loop List

The loop list lets you select and manipulate any Fortran DO loop contained in the source files loaded into the Parallel Analyzer View. Information about the loops is displayed in columns in the list; the headings of the columns are shown at the top of Figure 62, and described below.

Figure 62. Loop List with Column Headings



The columns in the loop list contain the following information about each loop, from left to right:

- Parallelization icon: Indicates the parallelization status of each loop. The meaning of each icon is described in the Icon Legend dialog box. (See Section 6.2.1.1.) When a loop is displayed in the loop information display (by double−clicking the loop's row), a green check mark is placed to the left of the icon to indicate that it has been examined. If any changes are made from within the loop information display, a red plus sign is placed above the check mark.
- Perf. Cost (not shown in Figure 62): The performance cost is displayed when the WorkShop Performance Analyzer is launched on the current fileset. (See Section 6.2.1.2.) The loops can be sorted by Perf. Cost via the sort option button. (See Section 6.4.2.)

  When performance cost is shown, each loop's execution time is displayed as a percentage of the total execution time. This percentage includes all nested loops, subroutines, and function calls.
- Nest: The nesting level of the given loop.
- Loop−ID: An ID for each loop in the list display. The ID is displayed indented to the right to reflect the loop's nesting level when the list is sorted in source order, and unindented otherwise.
- Variable: The name of the loop index variable.
- Subroutine: The name of the Fortran subroutine in which the loop occurs.
- Lines: The lines in the source file that make up the body of the loop.
- Olid: Original loop id is a unique internal identifier for the loops generated by the compiler. Use this value when reporting bugs.
- File: The name of the Fortran source file that contains the loop.

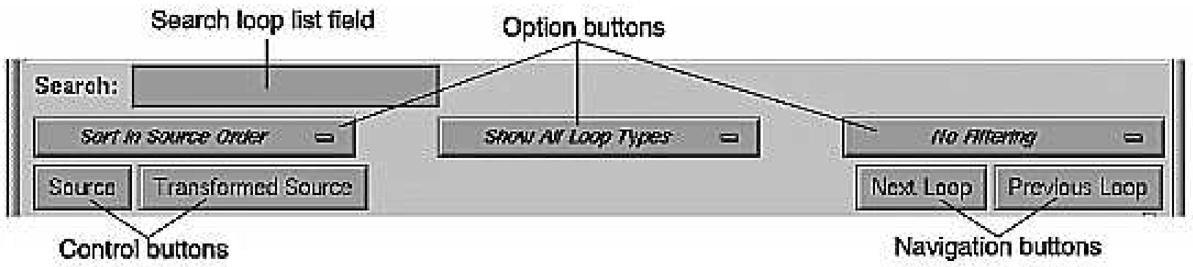
To *highlight* a loop in the list, click the left mouse anywhere in a loop's row; typing unique text from the row into the Search field does the same thing. (See Section 6.4.1.)

To *select* a loop, double−click on its row; this will bring up detailed information in the loop information display below the loop list display. (See Section 6.5.) Selecting a loop affects other displays. (See Section 2.6.3.)

6.4 Loop Display Controls

The loop display controls are shown in Figure 63, and are discussed in the next sections.

Figure 63. Loop Display Controls



6.4.1 Search Loop List Field

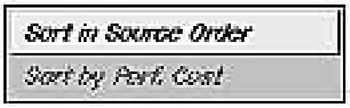
You can use the search loop list editable text field, shown at the top left of Figure 63, to find a specific loop in the loop list display. The Parallel Analyzer View matches any text typed into the field to the first instance of that text in the loop list, and highlights the row of the list in which the text occurs. The search field matches its text against the contents of each column in the loop list.

As you type into the field, the list highlights the first entry that matches what you have already typed, scrolling the list if necessary. If you press **`Enter`**, the highlight moves to the next match. If no match is found, the system beeps, and pressing **`Enter`** positions the highlight at the top of the list again.

6.4.2 Sort Option Button

The sort option button is the left–most option button under the loop list search field shown in Figure 63. It controls the order in which the loops are displayed in the loop list display.

Figure 64. Sort Option Button



The choices in the sort option button (Figure 64) have the following effects:

`Sort in Source Order`
: Orders the loops as they appear in the source file.

`Sort by Perf.Cost`
: Orders the loops by their performance cost (from greatest to least) as calculated by the Workshop Performance Analyzer. This is the default setting. You must invoke the Performance Analyzer from the current session of the Parallel Analyzer View to make use of this option. See Section 6.2.1.2, for information on how to open the Performance Analyzer from the current session of the Parallel Analyzer View.

6.4.3 Show Loop Types Option Button

The show loop types option button is the center option button under the loop list search field shown in Figure 63. It controls what kind of loops are displayed for each file and subroutine in the loop list.

Figure 65. Show Loop Types Option Button

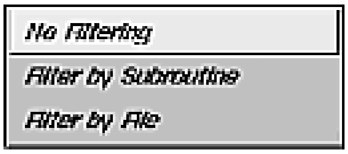
The options in the show loop types button (Figure 65) have the following effects:

`Show All Loop Types`	Default setting.
`Show Unparallelizable Loops`	Show only loops that could not be parallelized, and thereby run serially.
`Show Parallelized Loops`	Show only loops that are parallelized.
`Show Serial Loops`	Show only loops that are preferably serial.
`Show Modified Loops`	Show only loops with pending changes.
`Show OMP Directives`	Show only loops containing OMP directives.

6.4.4 Filtering Option Button

The filtering option button is the right–most option button under the loop list search field shown in Figure 63. It lets you display only those loops contained within a given subroutine or source file.

Figure 66. Filtering Option Button



The button choices have the following effects:

`No Filtering`	The default setting; lists all loops and routines.
`Filter by Subroutine`	Lets you enter a subroutine name into a filtering editable text field that appears above the option button. Only loops contained in that subroutine are displayed in the loop list.
`Filter by File`	Lets you enter a Fortran source filename into a filtering editable text field that appears above the option button. Only loops contained in that file are displayed in the loop list.

To place the name of a subroutine or file in the appropriate filter text field, you can double–click on a line in the Subroutines and Files View. If the appropriate type of filtering is currently selected, the loop list is rescanned.

6.4.5 Loop Display Buttons

The loop display controls (Figure 63) include two control buttons:

- `Source`: Opens the Source View window, with the source file containing the loop currently selected (double–clicked) in the loop list. The body of the loop is highlighted within the window. If no loop is selected, the last selected file is loaded; if no file is selected, the first file in the fileset is loaded.

For more information on the Source View window, see Section 6.7.1.

· `Transformed Source`: Opens a Parallel Analyzer View – Transformed Source window, with the compiled source file containing the loop currently selected (double–clicked) in the loop list. The body of the loop is highlighted within the window. If no loop is selected, the last selected file is loaded; if no file is selected, the first file in the fileset is loaded.

  For more information on the Transformed Source window, see Section 6.7.1.

The loop display controls also include two navigation buttons:

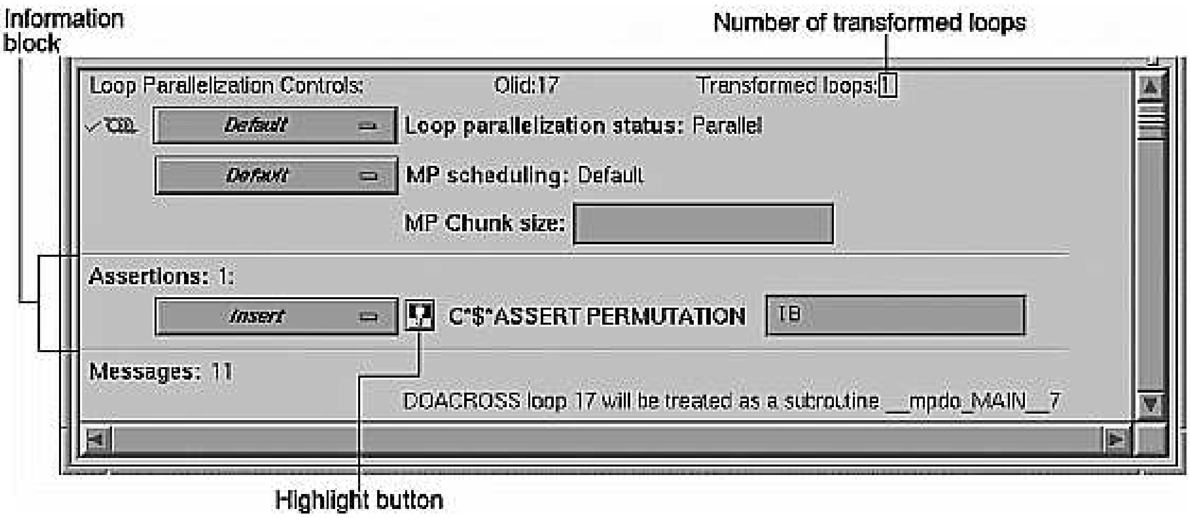
· `Next Loop`: Selects the next loop in the loop list. The information in the loop information display and all other windows is updated accordingly. If no loop is currently selected, clicking on the button selects the first loop.

· `Previous Loop`: Selects the previous loop in the loop list. The information in the loop information display and all other windows is updated accordingly. If no loop is currently selected, clicking on the button selects the first loop.

6.5 Loop Information Display

The loop information display provides detailed information on various loop parameters, and allows you to alter those parameters to incorporate the changes into the Fortran source. The display is divided into several information blocks displayed in a scrolling list as shown in Figure 67.

Figure 67. Loop Information Display



Each of these sections and the information it contains is described in detail below. The display is empty when no loop has been selected.

6.5.1 Highlight Buttons

A highlight button (light bulb, see Figure 67) appears as a shortcut to more information related to text in the display. Clicking the button does one or both of the following:

· Highlights the loop and the relevant line(s) in a Source View window. (See Section 6.7.1.)

· If a directive appears in the options menu next to it, the highlight button presents details about directive clauses in a Parallelization Control View. (See Section 6.6.1.)

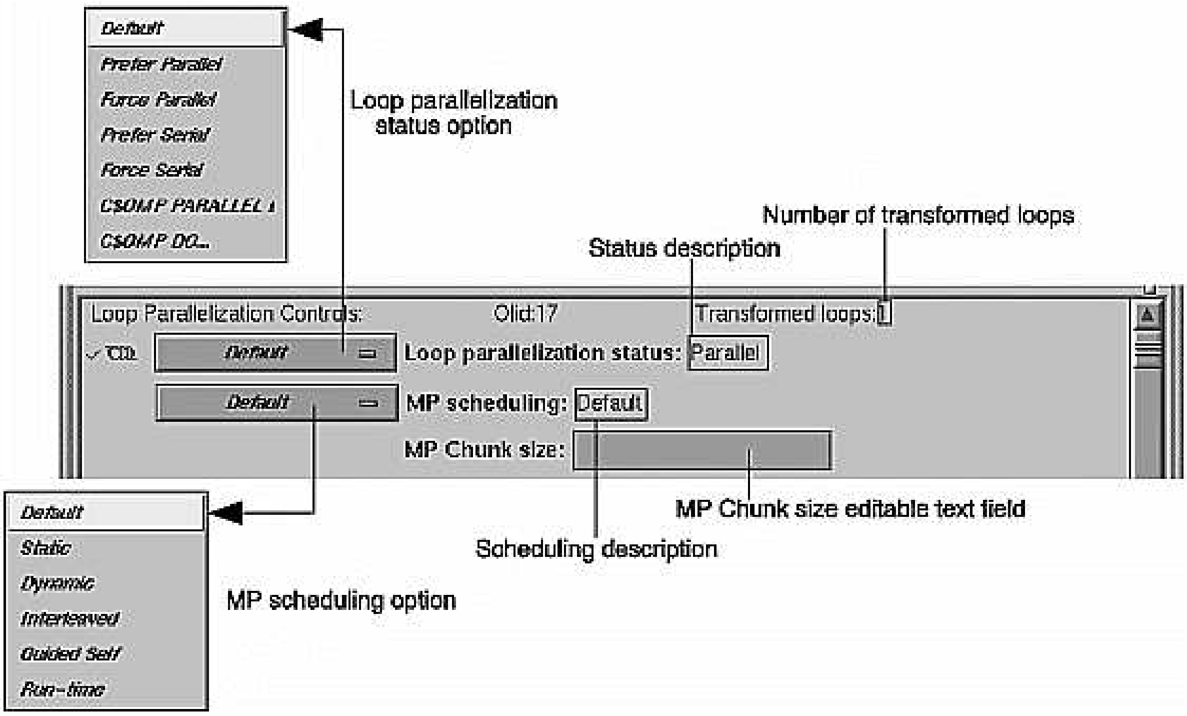
If directives or assertions with highlight buttons are also listed below the Loop Parallelization Controls, these buttons highlight the same piece of code as the corresponding button in the Loop Parallelization Controls, but they do not activate the Loop Parallelization Control View.

6.5.2 Loop Parallelization Controls in the Loop Information Display

The first line of the Loop Parallelization Controls section shows the Olid of the selected loop and, on the far right, how many transformed loops were derived from the selected loop.

Controls for altering the parallelization of the selected loop are shown in Figure 68. The controls in this section allow you to place parallelization assertions and directives in your code. Recall that you have similar controls available through the Operations menu. (See Section 6.2.6.)

Figure 68. Loop Parallelization Controls



6.5.2.1 Loop Parallelization Status Option Button

The loop parallelization status option button (shown in Figure 68) lets you alter a loop's parallelization scheme. To the right of the option button is the Loop parallelization status field, a description of the current loop status as implemented in the transformed source. A small highlight button appears to the left of this description if the status was set by a directive.

The loop parallelization status option button choices follow below. The directives and assertions mentioned in the choices are described in Table 6.

`Default`	Always selects the parallelization scheme that the compiler picked for the selected loop.
`Prefer Parallel`	Adds the assertion `C*$*ASSERT DO PREFER` (CONCURRENT).
`Force Parallel`	Adds the assertion `C*$*ASSERT DO` (CONCURRENT).
`Prefer Serial`	Adds the assertion `C*$ASSERT DO PREFER` (SERIAL).
`Force Serial`	Adds the assertion `C*$*ASSERT DO` (SERIAL).
`C$OMP PARALLEL DO...`	Adds the OpenMP directive `C$OMP PARALLEL DO`. Selecting

this item opens the Parallelization Control View. See Section 6.6.1, for more information.

`C$OMP DO...`
: Launches the Parallelization Control View, which allows you to manipulate the scheduling clauses for the OpenMP `C$OMP DO` directive and to set each of the referenced variables as either region−default or last−local.

  A `C$OMP DO` must be within a parallel region, although the tool does not enforce this restriction. If one is added outside of a region, the compiler reports an error.

A menu choice is grayed out if you are looking at a read−only file, if you invoked `cvpav` with the `-ro True` option, or if the loop comes from an included file. So in some cases you are not allowed to change the menu setting.

Table 6, lists the assertions and directives that you control from the loop parallelization status option button.

Table 6. Assertions and Directives Accessed From the Loop Parallelization Controls

Assertion or Directive	Effect on Compilation	For More Information
`C*$* ASSERT DO (CONCURRENT)`	Parallelize the loop; ignore possible data dependences.	*MIPSpro Auto−Parallelizing Option Programmer's Guide*, Chapter 3
`C*$* ASSERT DO PREFER (CONCURRENT)`	Attempt to parallelize the selected loop. If not possible, try each nested loop.	*MIPSpro Auto−Parallelizing Option Programmer's Guide*, Chapter 3
`C*$* ASSERT DO (SERIAL)`	Do not parallelize the loop.	*MIPSpro Auto−Parallelizing Option Programmer's Guide*, Chapter 3
`C*$* ASSERT DO PREFER (SERIAL)`	Do not parallelize the loop.	*MIPSpro Auto−Parallelizing Option Programmer's Guide*, Chapter 3
`C$OMP PARALLEL DO`	Parallelize the loop, ignore automatic parallelizer.	*OpenMP Fortran Application Progr Interface*, see http://www.openmp.or
`C$OMP DO`	Assign each loop iteration to a different thread, ignore automatic parallelizer.	*OpenMP Fortran Application Progr Interface*, see http://www.openmp.or

6.5.2.2 MP Scheduling Option Button: Directives for All Loops

The MP scheduling option button (Figure 68) lets you alter a loop's scheduling scheme by changing `C$MP_SCHEDTYPE` modes and values for `C$CHUNK`. For those modes that require a chunk size, there is a editable text field to enter the value. (See Section 6.5.2.3.)

These directives affect the current loop and all subsequent loops in a source file. For more information, see Chapter 5 in the *MIPSpro Fortran 77 Programmer's Guide.* For control over a single loop, use the `C$OMP PARALLEL DO` directive clause. (See Section 6.6.1.3.)

The button choices are as follows:

`Default`
: Always selects the scheduling scheme that the compiler picked for the selected loop.

`Static`
: Divides iterations of the selected loop among the processors by dividing them into contiguous pieces and assigning one to each processor.

`Dynamic`	Divides iterations of the selected loop among the processors by dividing them into pieces of size `C$CHUNK`. As each processor finishes a piece, it enters a critical section to grab the next piece. This scheme provides good load balancing at the price of higher overhead.
`Interleaved`	Divides the iterations into pieces of size `C$CHUNK` and interleaves the execution of those pieces among the processors. For example, if there are four processors and `C$CHUNK` = 2, then the first processor executes iterations 1–2, 9–10, 17–18,...; the second processor executes iterations 3–4, 11–12, 19–20,...; and so on.
`Guided Self`	Divides the iterations into pieces. The size of each piece is determined by the total number of iterations remaining. The idea is to achieve good load balancing while reducing the number of entries into the critical section by parceling out relatively large pieces at the start and relatively small pieces toward the end.
`Run-time`	Lets you specify the scheduling type at run time.

To the right of the MP scheduling option button is the MP scheduling field, a description of the current loop scheduling scheme as implemented in the transformed source. A highlight button appears to the left of this description if the scheduling scheme was set by a directive.

6.5.2.3 MP Chunk Size Field

Below the MP scheduling description is the MP Chunk size editable text field, a field that allows you to set the `C$CHUNK` size for the scheduling scheme you select.

When you change an entry in the field, the upper right corner of the field turns down, indicating the change (Figure 69). To toggle back to the original value, left–click the turned–down corner (changed–entry indicator). The corner unfolds, leaving a fold mark. If you click again on the fold mark, you can toggle back to the changed value. You can enter a new value at any time; the field remembers the original value, which is always displayed after you click on the changed–entry indicator.

Figure 69. MP Chunk Size Field Changed

Be aware of the following when you use the MP Chunk size field:

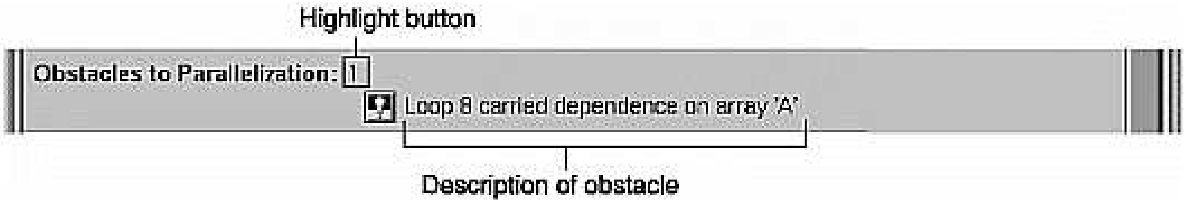
- Your entry should be syntactically correct, although it is not checked.
- Like any other editable text field, the background color changes when you cannot make edits. This can happen if you are looking at a read–only file, if you invoked `cvpav` with the **–ro True** option, if the loop comes from an included file, or in some other cases.

6.5.3 Obstacles to Parallelization Information Block

Obstacles to parallelization are listed when the compiler discovers aspects of a loop's structure that make it impossible to parallelize. They appear in the loop information display below the parallelization controls.

Figure 70, illustrates a message describing an obstacle. The message has a highlight button directly to its left to indicate the troublesome line(s) in the Source View window, and opens the window if necessary. If appropriate, the referenced variable or function call is highlighted in a contrasting color.

Figure 70. Obstacles to Parallelization Block



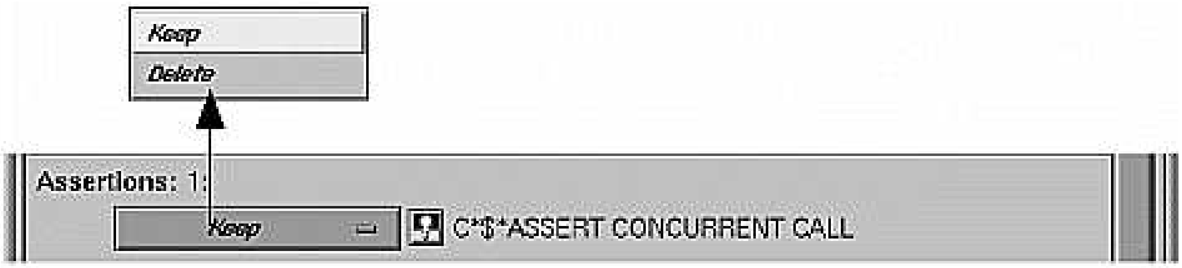
6.5.4 Assertions and Directives Information Blocks

The loop information display lists any assertions and directives for the selected loop along with highlight buttons. When you left−click the highlight button to the left of an assertion or directive, the Source View window shows the selected loop with the assertion or directive highlighted in the code.

Recall that assertions and directives are special Fortran source comments that tell the compiler how to transform Fortran code for multiprocessing. Directives enable, disable, or modify features of the compiler when it processes the source. Assertions provide the compiler with additional information about the source code that can sometimes improve optimization.

Some assertions or directives appear with an information block option button that allows you to `Keep` or `Delete it`. (If you compile `o32`, you can also `->Reverse it`.) Figure 71, shows an assertion block and its option button.

Figure 71. Assertion Information Block and Options (n32 and n64 Compilation)



Assertions and directives that govern loop parallelization or scheduling do not have associated option buttons; those functions are controlled by the loop parallelization status option button and the MP scheduling option button. (See Section 6.5.2.)

6.5.5 Compiler Messages

The Loop information display also shows any messages generated by the compiler to describe aspects of the loops created by transforming original source loops. As an example, the loop information display in Figure 67, shows there are 11 messages present although only one is shown. Some messages have associated buttons that highlight sections of the selected loop in the Source View window.

6.6 Views Menu Options

The views in this section are launched from the Views menu in the main menu bar of the Parallel Analyzer View. All of the views discussed in this section contain the following in their menu bars:

- Admin menu: This menu contains a single Close command that closes the corresponding view.
- Help menu: This menu provides access to the online help system. (See Section 6.2.7, for an explanation of the commands in this menu.)

6.6.1 Parallelization Control View

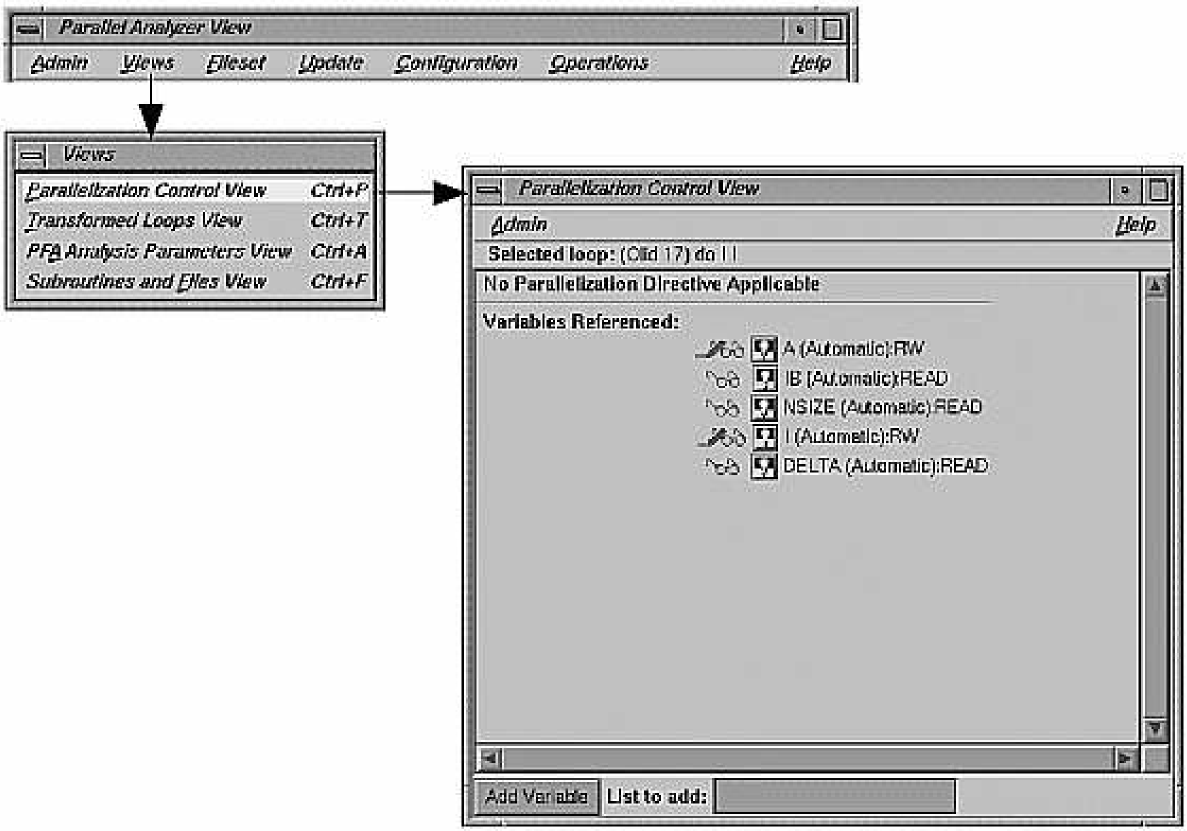
The Parallelization Control View shows parallelization controls (directives and their clauses), where applicable, and all the variables referenced in the selected loop, OpenMP construct, or subroutine. It can be opened by either of two ways.

- Selecting the Views > Parallelization Control View option. Figure 72, shows the Parallelization Control View when it is launched from the Views menu with the Default loop parallelization status option button; this is the display for loops without directives.
- Selecting `C$OMP PARALLEL DO...` or `C$OMP DO...` in the loop parallelization status option button (Figure 73, and Figure 74). This approach provides controls for clauses you can append to these directives.

Features that appear no matter which method is used to open the Parallelization Control View are discussed under Section 6.6.1.1. Features that appear only when the view is opened from the loop parallelization status option button with `C$OMP PARALLEL DO...` or `C$OMP DO...` selected are discussed in the following:

- Section 6.6.1.2
- Section 6.6.1.3
- Section 6.6.1.4

Figure 72. Parallelization Control View



6.6.1.1 Common Features of the Parallelization Control View

Independently of how you open the Parallelization Control View, these elements appear in the window (Figure 72):

- Selected loop: Contains the Olid of the loop, and the information about the loop from the Loop−ID and Variable columns of the loop list.
- Directive information section: If a directive is applicable to the loop, this section lists directive, clauses, and parameter values. (See Section 6.6.1.2.)
- Variables Referenced: The listing has two icons for each variable. They allow you to highlight the variable in the Source View and to determine the variable's read/write status; see Section 6.2.1.1, for an explanation of these icons.

  For discussion of added option buttons that appear if the view is opened from the loop parallelization status option button when `C$OMP PARALLEL DO...` or `C$OMP DO...` is selected, see Section 6.6.1.4.
- `Add Variable`: Located at the bottom of the window frame, this button allows you to add new variables to a loop.
- List to add: Located at the bottom of the window frame, this editable text field allows you to indicate the variables you wish to add to the loop. You may enter multiple variables, with each variable name separated by a space or comma.

6.6.1.2 `C$OMP PARALLEL DO` and `C$OMP DO` Directive Information

Option buttons and editable text fields in addition to those described in Section 6.6.1.1, are available if you open the Parallelization Control View from the loop parallelization status option button with either `C$OMP PARALLEL DO...` or `C$OMP DO...` selected. (See Figure 73, and Figure 74.)

There are two additional option buttons available:

- MP scheduling option button: This button allows you to alter a loop's scheduling scheme by changing the `C$MP_SCHEDTYPE` clause. See Section 6.6.1.3, for further information. This is the same button shown in Figure 68.
- Synchronization construct option button (`C$OMP DO...` only): This button allows you to set the `NOWAIT` clause at the end of the `C$OMP END DO` directive to avoid the implied `BARRIER`.

Figure 73. Parallelization Control View With `C$OMP PARALLEL DO` Directive

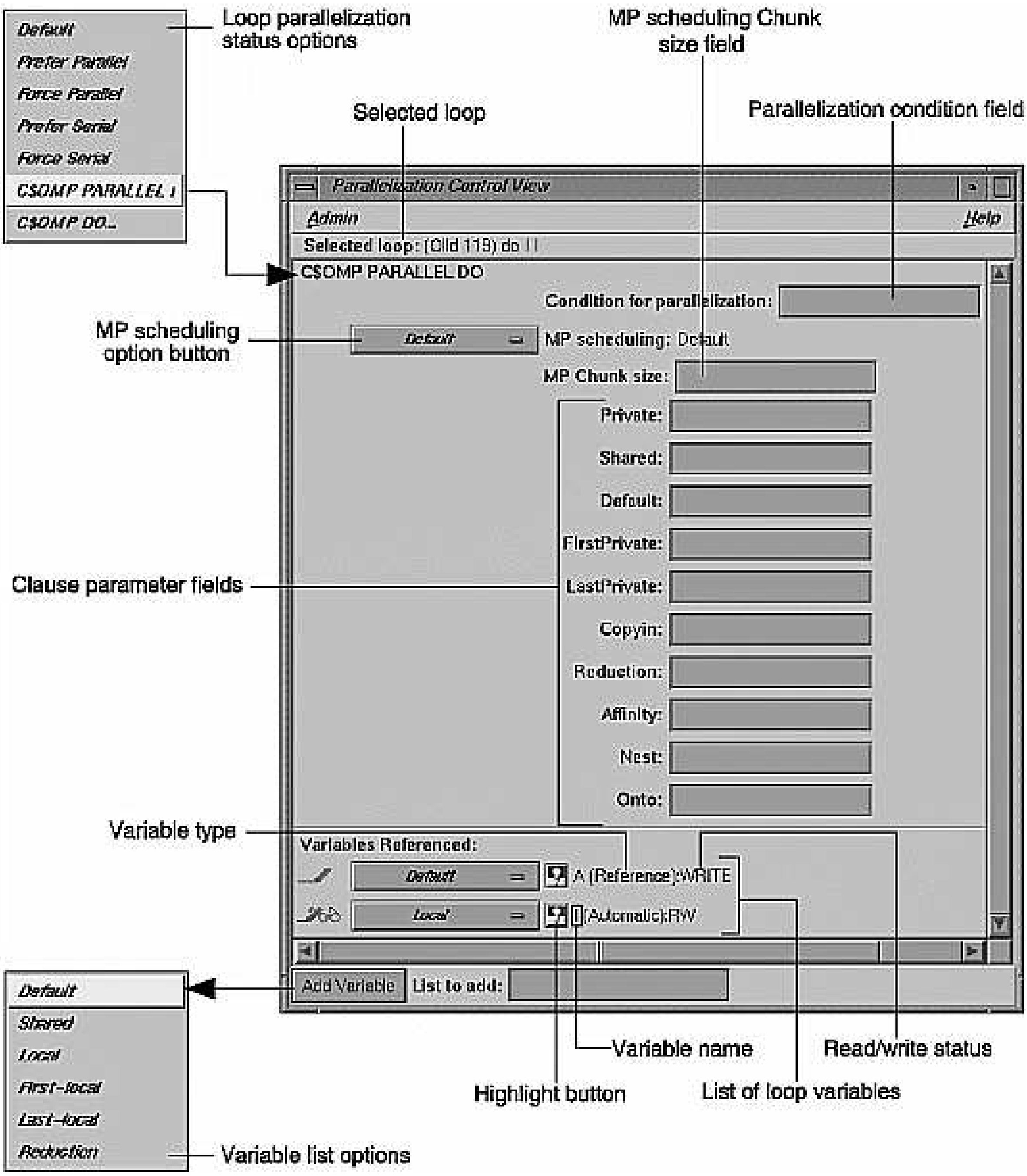
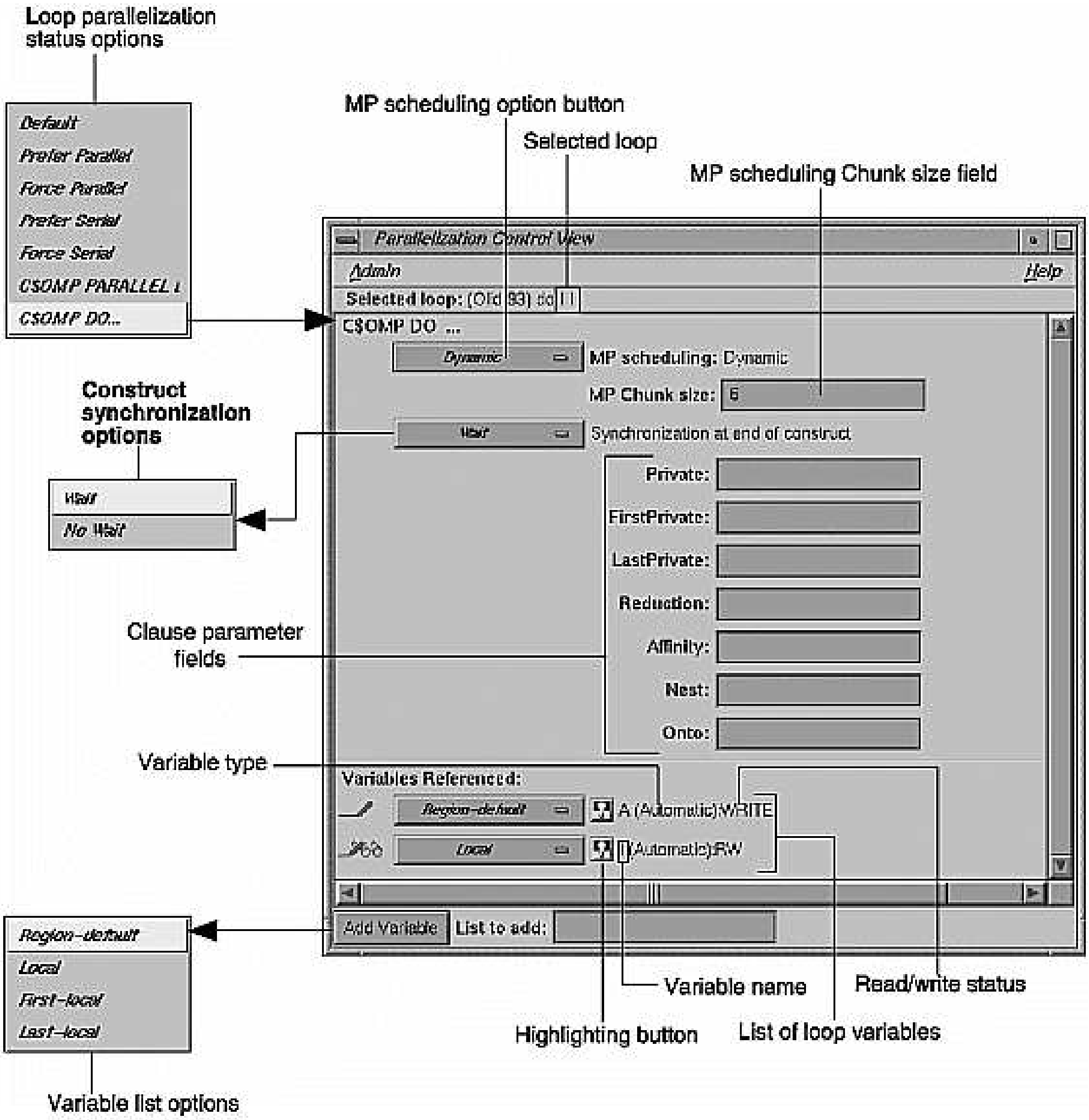


Figure 74. Parallelization Control View With `C$OMP DO` Directive



The following is a list of additional editable text fields that allow you to specify clauses for the `C$OMP PARALLEL DO` or `C$OMP DO` directives. Unless otherwise specified, the clause descriptions come from the *OpenMP Fortran Application Program Interface*.

- Condition for parallelization: Allows you to enter a Fortran conditional statement, for example, `NSIZE .GT. 83`. (`C$OMP PARALLEL DO...` only.)

  This statement determines the circumstances under which the loop will be parallelized. The upper right corner of the field changes when you type in the field. Your entry must be syntactically correct; it is not checked.

- MP Chunk size: Allows you to set the `C$CHUNK` size for the scheduling scheme you select. For further information, see Section 6.5.2.3.

- Private: Declares the variables in a list to be `PRIVATE` to each thread in a team.

- Shared: Makes variables that appear in a list shared among all the threads in a team. All threads within a team access the same storage area for `SHARED` data. (`C$OMP PARALLEL DO...` only.)

- Default: Allows you to specify a `PRIVATE`, `SHARED`, or `NONE` scope attribute for all variables in the lexical extent of any parallel region. Variables in `THREADPRIVATE` common blocks are not affected by this clause.

(C$OMP PARALLEL DO... only.)

- Firstprivate: Provides a superset of the functionality provided by the PRIVATE clause.
- Lastprivate: Provides a superset of the functionality provided by the PRIVATE clause.
- Copyin: Applies only to common blocks that are declared as THREADPRIVATE. (C$OMP PARALLEL DO... only.)

  A COPYIN clause on a parallel region specifies that the data in the master thread of the team be copied to the thread private copies of the common block at the beginning of the parallel region.
- Reduction: Performs a reduction on the variables that appear in a list with an operator (+, *, –, .AND., .OR., .EQV., or .NEQV.), or an intrinsic (MAX, MIN, IAND, IOR, or IEOR).
- Affinity: Allows you to specify the parameters for the affinity scheduling clause. The two types of affinity scheduling are described below. (For more details and syntax, see the *MIPSpro Fortran 77 Programmer's Guide.*)
  – Data affinity scheduling, which assigns loop iterations to processors according to data distribution.
  – Thread affinity scheduling, which assigns loop iterations to designated processors.
- Nest: Allows you to specify parameters in this clause for concurrent execution of nested loops. You can use the NEST clause to parallelize nested loops only when there is no code between either the opening DO statements or the closing ENDDO statements. For more details and syntax, see the *MIPSpro Fortran 77 Programmer's Guide*.
- Onto: Allows you to specify parameters for this clause to determine explicitly how processors are assigned to array variables or loop iteration variables. For more details and syntax, see the *MIPSpro Fortran 77 Programmer's Guide*.

6.6.1.3 MP Scheduling Option Button: Clauses for One Loop

The Parallelization Control View contains an MP scheduling option button if it is opened from the loop parallelization status option button with either C$OMP PARALLEL DO... or C$OMP DO... selected.

The options that appear have the same names as those for the MP scheduling option button in the loop information display, shown in Figure 68. However, the option button in the Parallelization Control View affects the C$MP_SCHEDTYPE and C$CHUNK clauses in the C$OMP PARALLEL DO directive, and so affects only the currently selected loop. Recall that the MP scheduling option button in the loop information display affects the placement of the C$MP_SCHEDTYPE and C$CHUNK directives and thus all subsequent loops.

Except for this difference in scope, the effects of both option buttons are the same; for a description, see Section 6.5.2.2. For more information, see the *MIPSpro Fortran 77 Programmer's Guide*.

6.6.1.4 Variable List Option Buttons

If the Parallelization Control View is opened from the loop parallelization status option button when either C$OMP PARALLEL DO... or C$OMP DO... is selected, each variable listed in the lower portion of the view appears with an option button. The menu allows you to append a clause to the directive, enabling you to control how the processors manage the variable. It is an addition to the highlight and read/write icons discussed in Section 6.6.1.1.

> **Note:** The highlight button may not indicate in the Source View all the occurrences relevant to a variable subject to a OpenMP directive; you may need to select the entire parallel region in which the variable occurs.

If the view is opened from the loop parallelization status option button when C$OMP PARALLEL DO... is selected, these are the variable list option button choices (Figure 73):

Default	Uses the control established by the compiler.
Shared	One copy of the variable is used by all threads of the MP process.

`Local`	Each processor has its own copy of the variable.
`Last-local`	Similar to Local, except the value of the variable after the loop is as the logically last iteration would have left it.
`Reduction`	A sum, product, minimum, or maximum computation of the variable can be done partially in each thread and then combined afterwards.

If the view is opened from the loop parallelization status option button when `C$OMP DO...` is selected, these are the variable list option button choices (Figure 74):

`Region-default`	Uses the control established by the compiler for the parallel region.
`Local`	Each processor has its own copy of the variable.
`First-local`	Similar to Local, except the value of the variable after the loop is as the logically first iteration would have left it.
`Last-local`	Similar to Local, except the value of the variable after the loop is as the logically last iteration would have left it.

6.6.1.5 Variable List Storage Labeling

In parentheses after each variable name in the list of variables is a word indicating the storage class of the variable. There are three possibilities:

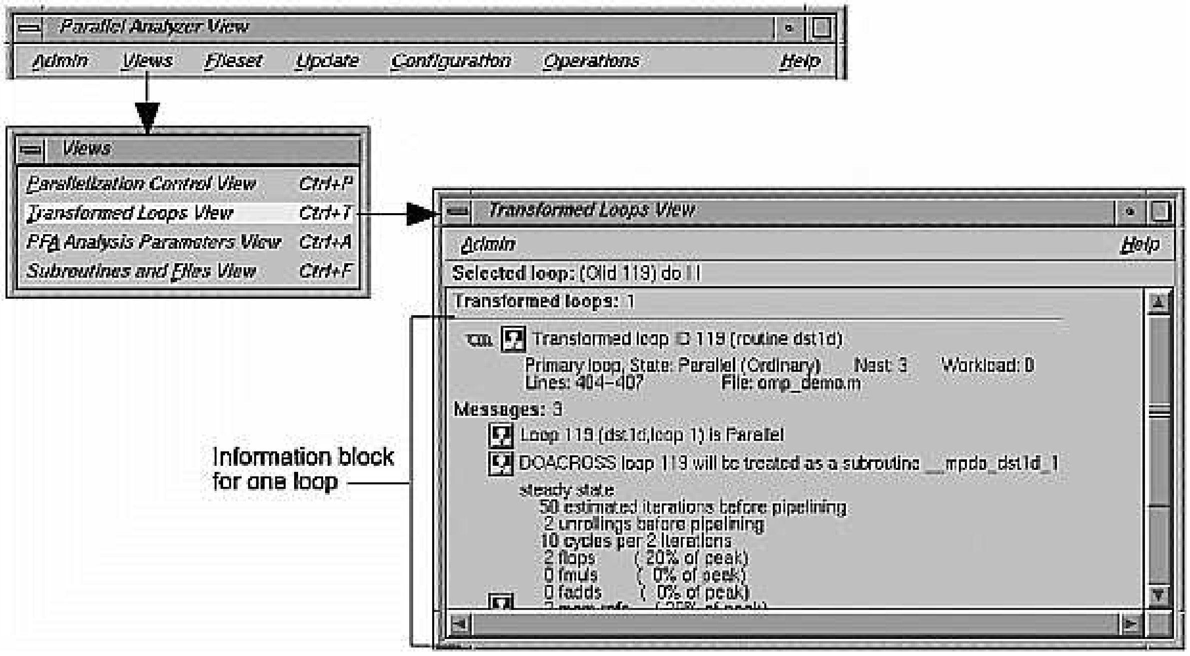
- Automatic: The variable is local to the subroutine, and is allocated on the stack.
- Common: The variable is in a common block.
- Reference: The variable is a formal argument, or dummy variable, local to the subroutine.

6.6.2 Transformed Loops View

The Transformed Loops View contains information about how a loop selected from the loop list is rewritten by the compiler into one or more *transformed loops*.

To open this view, choose Views > Transformed Loops View. (See Figure 75)

Figure 75. Transformed Loops View



Loop identifying information appears on the first line below the window menu, and below that is an indication of how many transformed loops were created.

Each transformed loop is displayed in its own section of the Transformed Loops View in an information block.

· The first line in each block for contains:

– A parallelization status icon

– A highlighting button (highlights the loop in the Transformed Source window and in the original loop in the Source View)

– The Olid number of the transformed loop

· The next line describes the transformed loop, providing information such as the following:

– Whether it is a *primary* loop or *secondary* loop (whether it is directly transformed from the selected original loop or transformed from a different original loop, but incorporates some code from the selected original loop)

– Parallelization state

– Whether it is an ordinary loop or interchanged loop

– Its nesting level

· The last line in the loop's information block displays the location of the loop in the transformed source.

Any messages generated by the compiler are below the loop information blocks. To the left of the message lines are highlight buttons. Left–clicking them highlights in the Transformed View the part of the original source that relates to the message. Often it is the first line of the original loop that is highlighted, since the message refers to the entire loop.

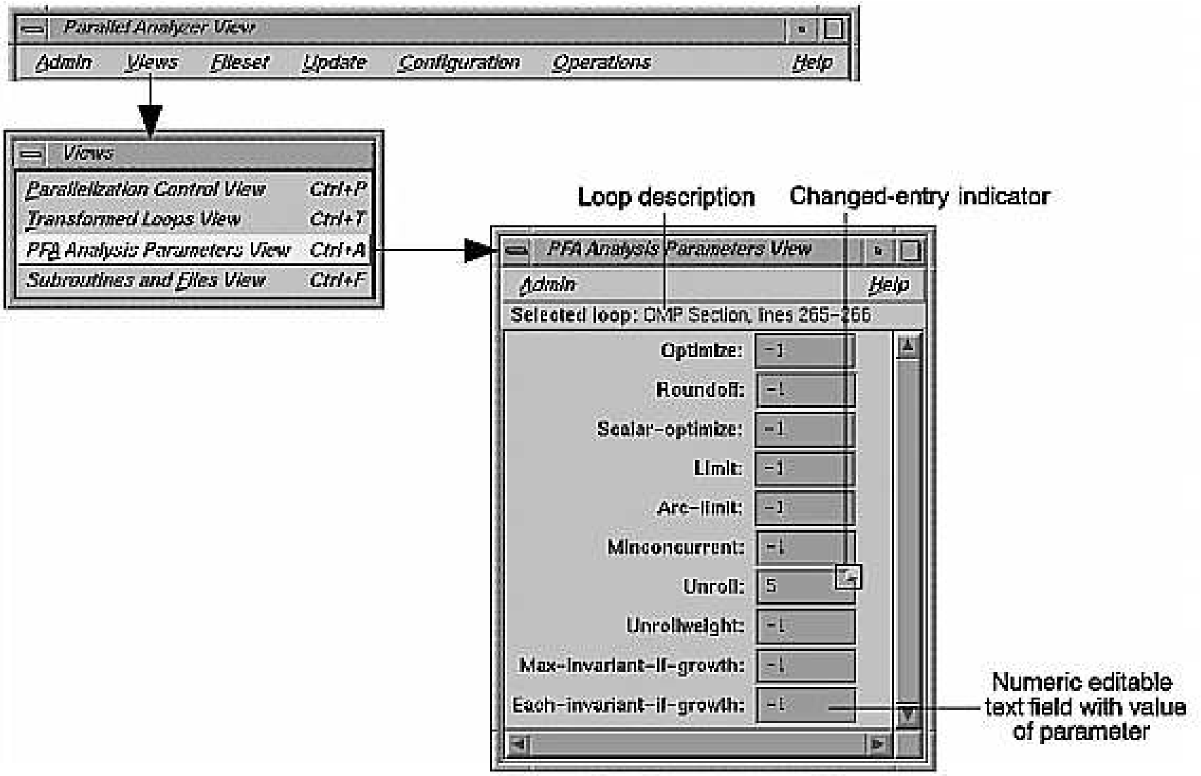
6.6.3 PFA Analysis Parameters View

If you compile with `o32`, you can use the PFA Analysis Parameters View, which contains a list of PFA execution parameters accompanied by fields into which you can enter new values. If you compile with `n32` or `n64`, these parameters have no effect and this view is not useful.

To open this view, choose Views > PFA Analysis Parameters View in the main window. (See Figure 75.)

When you update a source file, any PFA parameters you alter are changed for that file (Figure 76). When you change a parameter, the upper right corner of the field window turns down, as discussed in Section 6.5.2.3.

Figure 76. PFA Analysis Parameters View



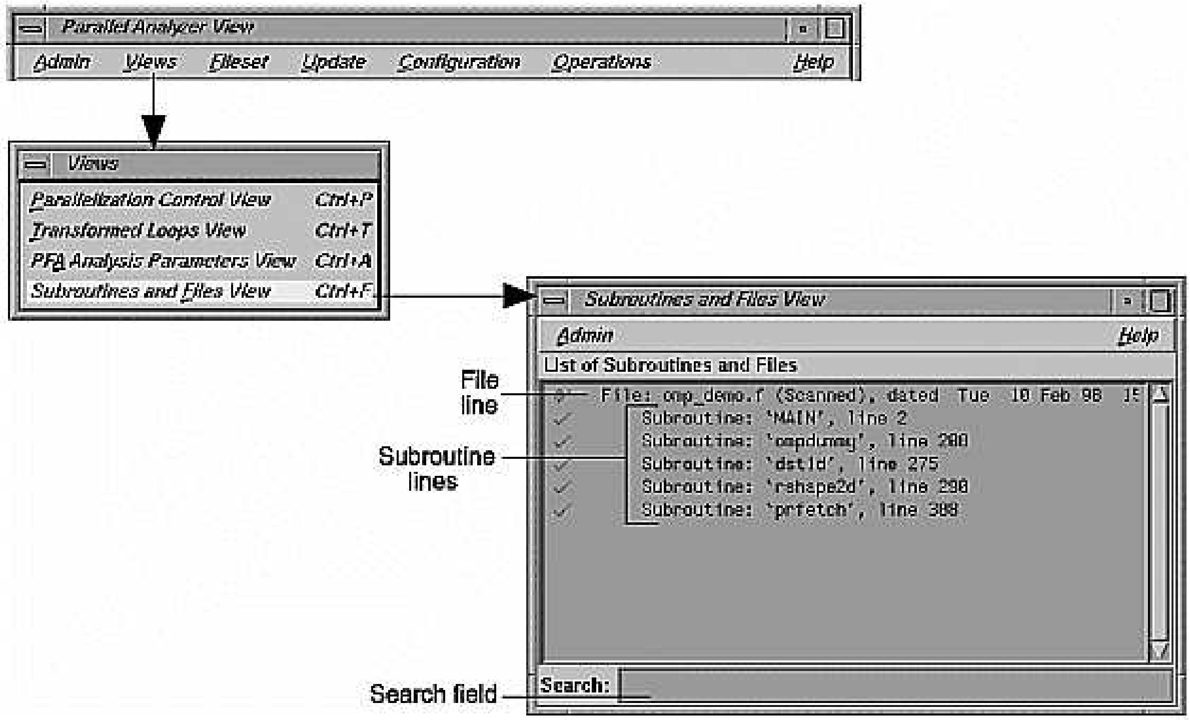
A full explanation of the PFA parameters, shown in Figure 76, can be found in Chapter 4, "Customizing PFA Execution," in the *POWER Fortran Accelerator User's Guide.*

6.6.4 Subroutines and Files View

The Subroutines and Files View contains a list from the file(s) in the current session of the Parallel Analyzer View (Figure 77). Below each filename in the list is an indented list of the Fortran subroutines it contains. Each item in the list is accompanied by icons to indicate file or subroutine status:

- A green check mark appears to the left of the file or subroutine name if the file has been scanned correctly or the subroutine has no errors.
- A red plus sign is above the green check mark shows if any changes have been made to loops in the file using the Parallel Analyzer View.
- A red international **not** symbol replaces the check mark if an error occurred because a file could not be scanned or a subroutine had errors.

Figure 77. Subroutines and Files View



The Search field in the Parallel Analyzer View uses the subroutine and file names listed in the Subroutines and Files View as a menu for search targets; see Section 6.4.1.

You can select items in the list for two purposes:

- To save changes to a selected file: click the filename and use the Update > Update Selected File option at the top of the Parallel Analyzer View main window. (See Section 6.2.4.)
- To select a file or subroutine for loop list filtering, discussed in Section 6.4.4, double−click on it. The selected name appears in the filtering text field; if the item is appropriate for the selected filtering option, the loop list is rescanned.

At the bottom of the window is a Search editable text field, which you can use to search the list of files and subroutines.

6.7 Loop Display Control Button Views

These views are summoned by clicking on the `Source` and `Transformed Source loop` display control buttons.

6.7.1 Source View and Parallel Analyzer View – Transformed Source

The Source View window and the Transformed Source window together present views of the source code before and after compiler optimization (Figure 78). The two windows use the WorkShop Source View interface.

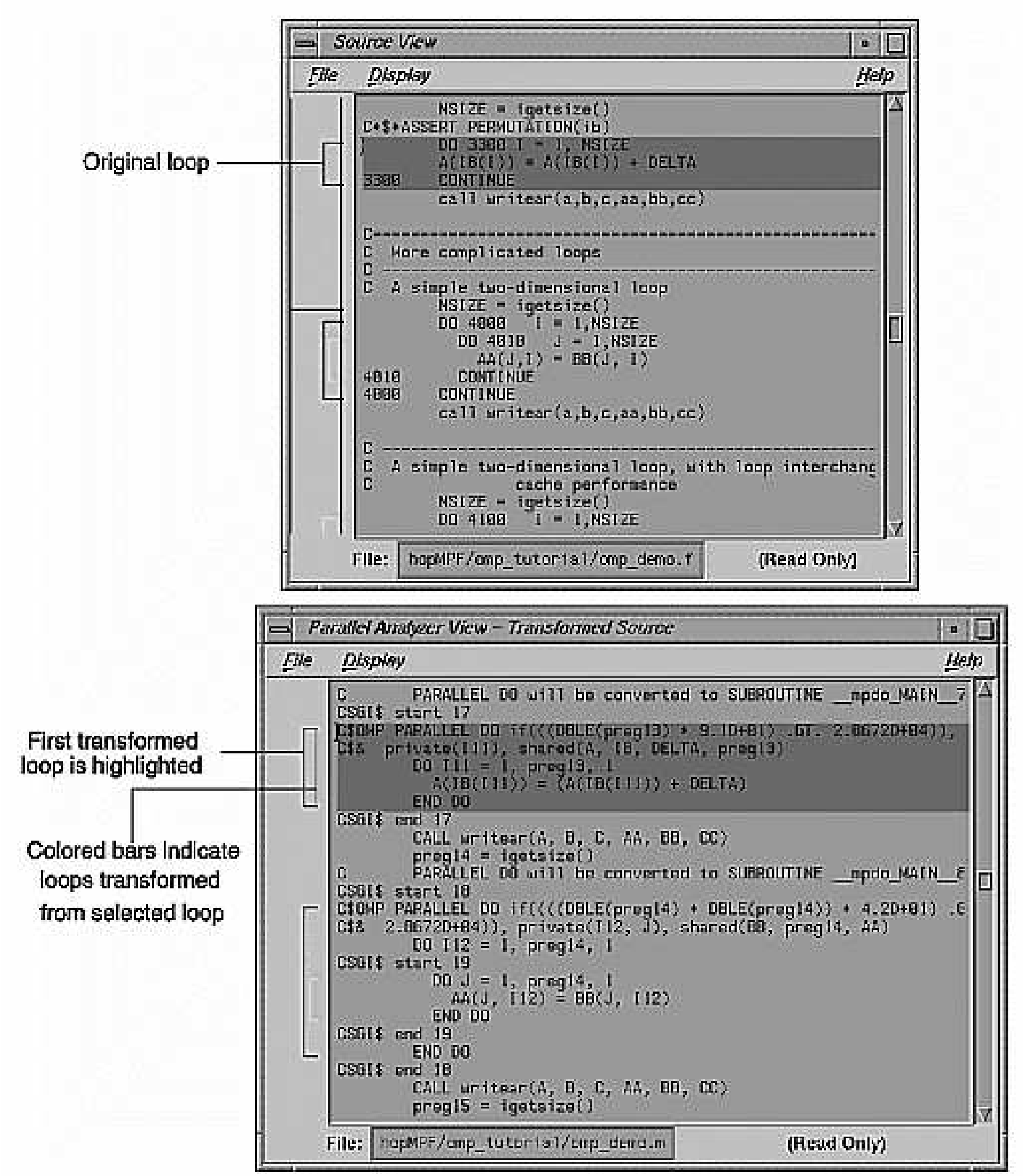
Both the Source View and Transformed Source windows contain bracket annotations in the left margin that mark the location and nesting level of each loop in the source file. Clicking on a loop bracket to the left of the code chooses and highlights the corresponding loop.

In the Transformed Source window, an indicator bar (a vertical line in a different color) indicates each loop that was transformed from the selected original loop.

If the source windows are invoked from a session linked to the WorkShop Performance Analyzer (see Section

6.2.1.2), any displayed sources files known to the Performance Analyzer are annotated with performance data.

Figure 78. Original and Transformed Source Windows



Note: The File and Display menus shown in the Source View and Transformed Source windows are standard Source View menus, and are described in the *Developer Magic: Debugger User's Guide*.