CONVEX CXpa Reference

First Edition

CONVEX COMPUTER CORPORATION

CONVEX CXpa Reference

Order No. DSW-253

First Edition
March 1993

CONVEX Press
Richardson, Texas
United States of America

CONVEX CXpa Reference

Order No. DSW-253

Printed in the United States of America

Revision information for

CONVEX CXpa Reference

Edition	Document No.	Description
First	710-004730-004	Initial release, March 1993. Released with V2.0 of CONVEX CXpa software.

Contents

Using this book

Purpose and audience

The *CONVEX CXpa Reference* describes the CXpa command set, windows, and dialog boxes that appear in the CONVEX CXpa software.

This manual is a reference for users who are already familiar with CXpa. It assumes that you have read the *CONVEX CXpa User's Guide* and that you understand the basic concepts presented there.

The reference pages contained in this manual are also available through the CXpa online help system.

Organization

This manual is organized as follows:

- **Chapter 1, "Commands"**—Contains descriptions, syntax rules and examples for all CXpa commands.
- **Chapter 2, "Windows"**—Contains descriptions and explanations for each window and dialog used in CXpa's CXwindows interface.
- **Chapter 3, "Reports"**—Contains detailed descriptions of the performance reports that CXpa creates.

Notational conventions

This document uses the following notational conventions.

Command syntax

Consider this example:

```
(CXpa) command <param1> [, ...] {a | b} [<param2>]
  ①      ②        ③        ④    ⑤      ⑥
```

1. `(CXpa)` is the CXpa command prompt.
2. **`command`** must be typed as it appears.
3. *<param1>* indicates a parameter that must be supplied.
4. The horizontal ellipsis in brackets [, ...] indicates that additional parameters may be specified.
5. Either **`a`** or **`b`** must be specified.
6. [*<param2>*] indicates an optional parameter.

General conventions

- **`Bold constant-width font`** identifies user input in examples.
- *Italics*:
  - Designate user-supplied variables in a command-line example (when enclosed in <>)
  - Indicate document titles
- `Constant-width font` designates input and output, including:
  - Command names and options
  - System calls
  - Program statements, command output, and error messages returned
- Horizontal ellipsis (...) shows repetition of the preceding item(s).
- Vertical ellipsis shows that lines have been left out of an example.
- Words and abbreviations that indicate keyboard keys you press are identified in a distinctive bold type. For example, **RETURN** refers to the carriage return key. Words separated by a hyphen indicate two keys that you press simultaneously. For

example, CTRL-X indicates that you must press and hold down the CTRL key and then press the X key.

- The shell prompt is shown as a percent sign (%).
- Unless otherwise indicated, source code examples are in FORTRAN.

Notes

NOTE: A NOTE highlights information that may be of particular interest regarding the software or your files.

Associated documents

This manual is not a complete explanation of CXpa. For more information, refer to:

- *CONVEX CXpa User's Guide* (DSW-251)—A guide that introduces you to using CXpa.
- *CONVEX CXpa Quick Reference* (DSW-252)—A quick reference card for using CXpa, containing command syntax and descriptions.
- *CONVEX Ada Optimization Guide* (DSW-144)—Describes the types of optimizations available in CONVEX Ada and shows you how to use optimization directives and options.
- *CONVEX C Optimization Guide* (DSW-089)—Describes the types of optimizations available in CONVEX C and shows you how to use optimization directives and options.
- *CONVEX FORTRAN Optimization Guide* (DSW-034)—Describes the types of optimizations available in CONVEX FORTRAN and shows you how to use optimization directives and options.

Ordering documentation

To order the current edition of this or any other CONVEX document, send requests to:

CONVEX Computer Corporation
Customer Service
P.O. Box 833851
Richardson, TX 75083-3851 USA

Include the order number or the exact title.

In some cases, you might not want the latest edition. To order a specific edition of a document, contact your local CONVEX office or call the Technical Assistance Center (TAC).

Technical assistance

If you have questions that are not answered by the documentation, contact the CONVEX Technical Assistance Center (TAC). To contact the TAC, use one of the following phone numbers:

- Within the continental U.S., call 1(800)952-0379.
- From Canada, call 1(800)345-2384.
- Outside the continental U.S., contact the local CONVEX office.

The `contact` utility

The TAC recommends using the `contact` utility to report a hardware, software, or documentation problem. The `contact` utility is an interactive program that helps the TAC track reports and route them to the CONVEX personnel most qualified to fix a problem.

After you invoke `contact`, it prompts you for information about the problem. When you finish your report, `contact` mails it to the TAC electronically. The TAC notifies you within 48 hours that your report has been received.

Using `contact` requires:

- UNIX-to-UNIX Communication Protocol (UUCP) connection to the TAC
- Full path name of the program or utility in question
- Version number of the program or utility in question

Refer to the contact(1) man page for complete details.

Acknowledgments

The author wishes to thank all the people at CONVEX and the users who contributed their ideas and time in the development of this book. In particular:

- The CXpa development team: Gary Brooks, Jeff Woods, Steve Simmons, Chuck Summers, and Tracy Webb.
- The team who created tests to help affirm the accuracy of CXpa and this book: Marianne Becker and Brent Henderson.
- The TAC representatives: Mike Gafka and Randy Torres.
- For their technical and usability comments: Ken Harward and Ray Cetrone.
- For her editorial comments that help make this book more consistent and readable: Mary Clare Bernier.

—Keith Knox

Commands

1

This chapter contains a reference page for each CXpa command. You can enter these commands at the CXpa command prompt in CRT mode (invoked with the `-nw` option) or in the text entry field of the Command Window.

Each reference page displays its command name at the top, followed by a one-line description. The rest of the page is divided into the following sections:

- **Syntax**—Lists the format rules for the command and its parameters.
- **Description**—Explains the purpose and functionality of the command.
- **Reports**—Lists the reports produced by this command and refers you to the appropriate section in Chapter 3, "Reports." If the command does not produce any reports, this section does not appear.
- **Examples**—Shows one or more examples illustrating the use of the command.
- **Related Commands**—Lists CXpa commands related to the command being described.

about cxpa

Display release information about CXpa.

Syntax

about cxpa

Description

The about cxpa command lists:

- CXpa's version number
- The release date of this version of CXpa
- The product number
- Copyright information

Examples

The output of the about cxpa command is shown in the following example.

```
(CXpa) about cxpa
Convex Performance Analyzer
Version 2.0
Released March 24, 1993
Product #710-005715-008
Copyright (c) 1990 CONVEX Computer Corporation
All rights reserved.
```

Related Commands

info cxpa

add path

Add directories to CXpa's search path for source files.

Syntax

`add path` *<directory-list>*

Parameter	Meaning
<directory-list>	Specifies one or more directories, separated by a space, to add to CXpa's search path.

Description

The `add path` command appends the specified list of directories to CXpa's search path. CXpa uses its search path to locate source files.

Use the `info path` command to list CXpa's current search path. Use the `path` command to replace CXpa's current search path.

You may find it convenient to place the `add path` command in CXpa's initialization file, .cxpainit, to automatically set CXpa's search path each time you invoke CXpa.

Examples

The following examples show typical uses of the `add path` command.

```
(CXpa) add path c_progs
(CXpa) info path
       /usr/work/f_progs
       /usr/work/c_progs
```

The preceding example of the `add path` command appends the relative path of c_progs to CXpa's search path. The `info path` command lists CXpa's complete search path.

```
1.(CXpa) list monitors sub2
             File prog.f not found in search path.
2.(CXpa) info path
             /mnt/dev_tree/wrk
3.(CXpa) add path /mnt/dev_tree/progs
4.(CXpa) list monitors sub2
       R   1 SUBROUTINE sub2(MATRIX, VAR3)
      L P  5 DO I=1,5
      L    6    DO J=1,5
```

The preceding example shows a scenario for using the `add path` command when CXpa cannot find a source file you are trying to list. The line numbers are for reference only.

1. When the `list monitors sub2` command is issued, CXpa cannot find the file that contains the routine `sub2`.
2. The `info path` command displays CXpa's current search path.
3. The `add path` command adds /mnt/dev_tree/progs to CXpa's search path.
4. Now that CXpa's search path has been altered, CXpa finds the needed source file when the `list monitors sub2` command is issued.

Related Commands

`info path`
`list monitors`
`list`
`path`

analyze

Display performance reports for all monitor point types.

Syntax

analyze [*<i/o_redirection>*]

Parameter	Meaning
<i/o_redirection>	Redirects this command's standard output or error to the specified file when you include one of the redirection operators (<, >, >>, >&, >>&).

Description

The `analyze` command calculates and displays routine, loop, block, and parallel region performance reports for programs compiled with a CXpa option (`-pa`, `-par`, `-pab`). To display a specific report, use the `analyze routine`, `analyze loop`, `analyze block`, or `analyze pregion` command.

The `analyze` command generates the reports from the data in the Performance Data File (PDF), so a PDF must exist before you can display a performance report. CXpa creates a PDF when you enable monitor points with a `monitor` command and execute the program with the `run` command. If you have not specified a PDF in this profiling session, CXpa uses the default file name cxpa.pdf.

By choosing a PDF with the `set pdf` command, you can use the `analyze` command to display a report from a PDF created in a previous CXpa session.

While you would normally use the `analyze` command after your program has completed execution, you can use the `analyze` command while a program is running by first using the `pause` command.

Reports

The `analyze` command can display the following reports under the listed conditions:

- Routine Report—Displayed for programs compiled with `-pa` or `-par` and monitored with the `monitor all` or `monitor routine` command.
- Loop Report—Displayed for programs compiled with `-pa` and monitored with the `monitor all` or `monitor loop` command.

- Parallel Region Report—Displayed for programs compiled with `-pa` and monitored with the `monitor all` or `monitor pregion` command.
- Basic Block Report—Displayed for programs compiled with `-pab` and monitored with the `monitor all` or `monitor block` command.

For a detailed description of these reports, refer to Chapter 3, "Reports."

Examples

The following examples show how to use the `analyze` command.

```
1.(CXpa) monitor all
2.(CXpa) run
   (Program runs to completion, and any output is displayed.)
3.(CXpa) analyze
   (Performance reports are displayed.)
```

The previous scenario shows how you would normally use the `analyze` command. The line numbers are for reference only.

1. The `monitor all` command tells CXpa to enable all monitor points in your program.
2. The `run` command executes the program and initiates profile data collection. Profile data collection ends when the program runs to completion.
3. The `analyze` command calculates and displays the performance reports from data collected and stored in the PDF file.

```
(CXpa) analyze > report_output
```

The previous command redirects the performance reports to the file named report_output. Any existing data in report_output is overwritten.

Related Commands

`analyze block`
`analyze pregion`
`deselect`
`rerun`
`set pdf`
`analyze loop`
`analyze routine`
`monitor`
`run`

analyze block

Display a performance report for profiled basic blocks.

Syntax

`analyze block` [*<routine-list>*] [*<i/o_redirection>*]

Parameter	Meaning
<routine-list>	Specifies one or more routines separated by a space.
<i/o_redirection>	Redirects this command's standard output or error to the specified file when you include one of the redirection operators (<, >, >>, >&, >>&).

Description

The `analyze block` command calculates and displays a Basic Block Report for programs compiled with the `-pab` option.

The `analyze block` command generates this report from the data in the Performance Data File (PDF), so a PDF must exist before you can display this report. CXpa creates a PDF when you enable monitor points with a `monitor` command and execute the program with the `run` command. If you have not specified a PDF in this profiling session, CXpa uses the default file name cxpa.pdf.

By choosing a PDF with the `set pdf` command, you can use the `analyze block` command to display a report from a PDF created in a previous CXpa session.

While you would normally use the `analyze block` command after your program has completed execution, you can use this command on a running program by first using the `pause` command.

Reports

The Basic Block Report provides performance information for monitored basic blocks. For a detailed description of the Basic Block Report, refer to Chapter 3, "Reports."

Examples

The following examples show how to use the `analyze block` command.

```
1.(CXpa) monitor block all
2.(CXpa) run
   (Program runs to completion, and any output is displayed.)
3.(CXpa) analyze block
   (The Basic Block Report is displayed.)
```

The previous scenario shows how you would normally use the `analyze block` command. The line numbers are for reference only.

1. The `monitor block all` command tells CXpa to enable all basic block monitor points in your program.
2. The `run` command executes the program and initiates profile data collection. Profile data collection ends when the program runs to completion.
3. The `analyze block` command calculates and displays the Basic Block Report from the data collected and stored in the PDF file.

```
(CXpa) analyze block
No profiling data was gathered for any basic block
```

The response to the `analyze block` command in the previous example appears when either of the following conditions occur:

- You have not compiled your program with the `-pab` compiler option.
- None of the monitored basic blocks in your program were executed.

```
(CXpa) analyze block > report_output
```

The previous command redirects the Basic Block Report to the file named report_output. Any existing data in report_output is overwritten.

Related Commands

`analyze`
`analyze pregion`
`deselect`
`rerun`
`set pdf`
`analyze loop`
`analyze routine`
`monitor`
`run`

analyze loop

Display a performance report for profiled loops.

Syntax

analyze loop [*<routine-list>*] [*<i/o_redirection>*]

Parameter	Meaning
<routine-list>	Specifies one or more routines separated by a space.
<i/o_redirection>	Redirects this command's standard output or error to the specified file when you include one of the redirection operators (<, > ,>>, >&, >>&).

Description

The `analyze loop` command calculates and displays a Loop Report for programs compiled with the `-pa` option and at optimization level `-O1` or higher.

The `analyze loop` command generates this report from the data in the Performance Data File (PDF), so a PDF must exist before you can display this report. CXpa creates a PDF when you enable monitor points with a `monitor` command and execute the program with the `run` command. If you have not specified a PDF in this profiling session, CXpa uses the default file name cxpa.pdf.

By choosing a PDF with the `set pdf` command, you can use the `analyze loop` command to display a report from a PDF created in a previous CXpa session.

While you would normally use the `analyze loop` command after your program has completed execution, you can use this command on a running program by first using the `pause` command.

Reports

The Loop Report provides performance information for monitored loops. For a detailed description of the Loop Report, refer to Chapter 3, "Reports."

Examples

The following examples show how to use the `analyze loop` command.

```
1.(CXpa) monitor loop all
2.(CXpa) run
   (Program runs to completion, and any output is displayed.)
3.(CXpa) analyze loop
   (The Loop Report is displayed.)
```

The previous scenario shows how you would normally use the `analyze loop` command. The line numbers are for reference only.

1. The `monitor loop all` command tells CXpa to enable all loop monitor points in your program.
2. The `run` command executes the program and initiates profile data collection. Profile data collection ends when the program runs to completion.
3. The `analyze loop` command calculates and displays the Loop Report from data collected and stored in the PDF file.

```
(CXpa) analyze loop
No profiling data was gathered for any loop
```

The response to the `analyze loop` command in the previous example appears when either of the following conditions occur:

- You have not compiled your program with the `-pa` compiler option.
- None of the monitored loops in your program were executed.

```
(CXpa) analyze loop > report_output
```

The previous command redirects the Loop Report to the file named report_output. Any existing data in report_output is overwritten.

Related Commands

`analyze`
`analyze block`
`analyze pregion`
`analyze routine`
`deselect`
`monitor`
`rerun`
`run`
`set pdf`

analyze pregion

Display a performance report for profiled parallel regions.

Syntax

`analyze pregion` [*<routine-list>*] [*<i/o_redirection>*]

Parameter	Meaning
<routine-list>	Specifies one or more routines separated by a space.
<i/o_redirection>	Redirects this command's standard output or error to the specified file when you include one of the redirection operators (<, >, >>, >&, >>&).

Description

The `analyze pregion` command calculates and displays a Parallel Region Report for programs compiled with the `-pa` option and at optimization level `-O3`.

The `analyze pregion` command generates this report from the data in the Performance Data File (PDF), so a PDF must exist before you can display this report. CXpa creates a PDF when you enable monitor points with a `monitor` command and execute the program with the `run` command. If you have not specified a PDF in this profiling session, CXpa uses the default file name cxpa.pdf.

By choosing a PDF with the `set pdf` command, you can use the `analyze pregion` command to display a report from a PDF created in a previous CXpa session.

While you would normally use the `analyze pregion` command after your program has completed execution, you can use this command on a running program by first using the `pause` command.

Reports

The Parallel Region Report provides performance information for monitored parallel regions. For a detailed description of the Parallel Region Report, refer to Chapter 3, "Reports."

Examples

The following examples show how to use the `analyze pregion` command.

```
1.(CXpa) monitor pregion all
2.(CXpa) run
   (Program runs to completion, and any output is displayed.)
3.(CXpa) analyze pregion
   (The Parallel Region Report is displayed.)
```

The previous scenario shows how you would normally use the `analyze pregion` command. The line numbers are for reference only.

1. The `monitor pregion all` command tells CXpa to enable all parallel region monitor points in your program.
2. The `run` command executes the program and initiates profile data collection. Profile data collection ends when the program runs to completion.
3. The `analyze pregion` command calculates and displays the Parallel Region Performance report from data collected and stored in the PDF file.

```
(CXpa) analyze pregion
No profiling data was gathered for any pregion
```

The response to the `analyze pregion` command in the previous example appears when either of the following conditions occur:

- You have not compiled your program with the `-pa` and `-O3` compiler options.
- None of the monitored parallel regions in your program were executed.

```
(CXpa) analyze pregion sub2
(The Parallel Region Report is displayed for routine sub2.)
```

The previous command calculates and displays a Parallel Region Report for the routine named `sub2` only.

```
(CXpa) analyze pregion > report_output
```

The previous command redirects the Parallel Region Report to the file named report_output. Any existing data in report_output is overwritten.

Related Commands

`analyze`	`analyze block`
`analyze loop`	`analyze routine`
`deselect`	`monitor`
`rerun`	`run`
`set pdf`	

analyze routine

Display performance reports for profiled routines.

Syntax

`analyze routine` [*<i/o_redirection>*]

Parameter	Meaning
<i/o_redirection>	Redirects this command's standard output or error to the specified file when you include one of the redirection operators (<, >, >>, >&, >>&).

Description

The `analyze routine` command calculates and displays a Routine Report for programs compiled with the `-pa` or `-par` option.

The `analyze routine` command generates this report from the data in the Performance Data File (PDF), so a PDF must exist before you can display this report. CXpa creates a PDF when you enable monitor points with a `monitor` command and execute the program with the `run` command. If you have not specified a PDF in this profiling session, CXpa uses the default file name cxpa.pdf.

By choosing a PDF with the `set pdf` command, you can use the `analyze routine` command to display a report from a PDF created in a previous CXpa session.

While you would normally use the `analyze routine` command after your program has completed execution, you can use this command on a running program by first using the `pause` command.

Reports

The Routine Report provides performance information for monitored routines. For a detailed description of the Routine Report, refer to Chapter 3, "Reports."

Examples

The following examples show how to use the `analyze` command.

```
1.(CXpa) monitor routine all
2.(CXpa) run
   (Program runs to completion, and any output is displayed.)
3.(CXpa) analyze routine
   (The Routine Report is displayed.)
```

The scenario above shows how you would normally use the `analyze routine` command. The line numbers are for reference only.

1. The `monitor routine all` command tells CXpa to enable all routine monitor points in your program.
2. The `run` command executes the program and initiates profile data collection. Profile data collection ends when the program completes execution.
3. The `analyze routine` command calculates and displays the Routine Report from data collected and stored in the PDF file.

```
(CXpa) analyze routine
No profiling data was gathered for any routine
```

The response to the `analyze routine` command in the previous example appears when either of the following conditions occur:

- You have not compiled your program with the `-pa` or `-par` compiler option.
- None of the monitored routines in your program were executed.

```
(CXpa) analyze routine > report_output
```

The previous command redirects the performance reports to the file named report_output. Any existing data in report_output is overwritten.

Related Commands

`analyze`
`analyze block`
`analyze loop`
`analyze pregion`
`deselect`
`monitor`
`rerun`
`run`
`set pdf`

continue

Continue profiling a paused program.

Syntax

```
continue
```

Description

The `continue` command resumes the profiling of a paused program.

When you continue profiling, the monitored program resumes execution, and CXpa resumes collecting profile data.

Examples

The following example shows a likely scenario in which you would use the `continue` command. The line numbers are for reference only.

```
1.(CXpa) run
2.(CXpa) pause (In CRT mode, pressing CTRL-c pauses profiling.)
3.(CXpa) analyze
   (Incomplete performance reports are displayed.)
4.(CXpa) continue
   (Program runs to completion, and data collection ends.)
5.(CXpa) analyze
   (Complete performance reports are displayed.)
```

The following lines explain the example above by number:

1. The `run` command runs the program and initiates profile data collection.
2. The `pause` command pauses profiling. If you are using CXpa in CRT mode (invoked with `cxpa -nw`), pressing **CTRL-c** pauses profiling.
3. The `analyze` command displays the performance information that has been collected to this point.
4. The `continue` command resumes the program and data collection.
5. When the program has run to completion, CXpa is finished collecting profile data. The `analyze` command is used again to display the final profile results of this run.

Related Commands `pause` `stop`

cxpa

Invoke the CONVEX performance analyzer.

Syntax

cxpa [[**-path** <*dir*>] ...] [**-x** <*cmdfile*>] [**-nx**] [[**-f**] <*executable*>] [**-pdf** <*filename*>] [**-nw**] [**-ne**] [**-ncg**] [<*X-Toolkit-options*>]

Parameter	Meaning
-path <*dir*>	Append the directory specified to the end of CXpa's search path. CXpa uses its search path when it looks for a source file. Any number of directories can be specified by repeating the `-path` option several times. The list of directories is initialized to the current working directory and the directory where the executable or PDF is located. The space between the options and the directory name is optional.
-x <*cmdfile*>	Execute the CXpa commands in the specified file. After CXpa has executed the commands in the file, the profiling session is terminated if the file contains the `quit` command; otherwise the user is prompted for a CXpa command.
-nx	Do not execute the CXpa commands in .cxpainit, the CXpa start-up command file.
-f <*executable*>	Specify fixed scheduling for the program you are profiling to reserve all processors for its use. Fixed scheduling is useful when profiling programs that have sections that execute in parallel. Without fixed scheduling, the number of threads created depends on the system load.
<*executable*>	Specify the name of the executable you wish to profile. The default is a.out.
-pdf <*filename*>	Invoke CXpa with the specified performance data file. CXpa uses the PDF to store profiling data and to generate the performance reports.
-nw	Invoke CXpa in CRT mode. Do not bring up the CXwindows version.

-ne	Add the Monitored Routines Not Called table to CXpa's Routine Report.
-ncg	Do not include the Dynamic Call Graph in CXpa's performance reports.
<*X-Toolkit-options*>	Specifies X Toolkit options. For more information about these options, refer to your X Window System's documentation.

Description

CXpa is a performance analyzer for C, FORTRAN and Ada that enables you to profile optimized programs compiled with one of the profiling options (`-pa`, `-par`, `-pab`). CXpa can run under CXwindows or with CRT terminals such as the vt100. Under CXwindows, you can use CXpa's mouse-driven interface, or you can bring up a command window to enter CXpa commands with the keyboard.

CXpa enables you to profile code at different instrumentation levels. There are four levels of instrumentation: routine, loop, parallel region (often referred to as p-regions), and basic block. You can set monitor points at any of these levels provided they exist. The monitor points you select are used by CXpa to gather performance data.

To compile a program for profiling, use one of the following profiling options:

- `-pa`—Used for profiling routines, loops, and parallel regions.
- `-par`—Used for profiling routines only.
- `-pab`—Used for profiling basic blocks only.

Using the command line version of CXpa

1. Compile your program with one of the CXpa profiling flags: `-pa`, `-par`, or `-pab`.
2. Invoke CXpa with the name of your executable. The `-nw` option invokes CXpa in CRT mode.

   ```
   % cxpa -nw a.out
   ```

3. Enable monitor points by using one of the `monitor` commands:

   ```
   (CXpa) monitor all
   ```

4. Run your program:

   (CXpa) **run**

5. Create and view a performance report by using one of the `analyze` commands:

   (CXpa) **analyze all**

Using the CXwindows version of CXpa

1. Compile your program with one of the CXpa profiling flags: `-pa`, `-par`, or `-pab`.
2. Invoke CXpa with the name of your executable. The following command invokes CXpa in CXwindows mode:

   % **cxpa a.out &**

3. Enable monitor points by selecting one of the Monitor options from the Setup menu.
4. Run your program by choosing Run from the Process menu. The Control Center dialog box appears. Press the start button to run your program.
5. Create and view a performance report by selecting one of the Analyze options from the Analysis menu.The report appears in the CXpa window.

Examples

The following examples show various ways to invoke CXpa.

```
% cxpa -nw
```

The previous command invokes CXpa with the default executable, a.out, and default PDF, cxpa.pdf, if they exist. If neither exist, CXpa starts, but only allows you to analyze existing PDFs. You can specify a PDF with the `set pdf` command.

```
% cxpa prog.out
```

The previous command invokes CXpa. The file name is treated as an executable file. This begins a profiling session using the specified executable file and the default performance data file (PDF) named cxpa.pdf.

```
% cxpa -pdf my.pdf
```

The previous command invokes CXpa. The `-pdf` option specifies the file name (my.pdf) as the performance data file (PDF). CXpa uses the PDF to store the performance data and to generate performance reports.

```
% cxpa -nw prog.out
```

The previous command invokes CXpa with the executable `prog.out`. Using the `-nw` option invokes CXpa in CRT mode.

```
% cxpa -nx
```

The previous command invokes CXpa but inhibits it from processing any initialization files.

Related Commands

`analyze`
`continue`
`monitor`
`run`
`quit`

deselect

Disable every monitor point at a line, in a routine, or in your program.

Syntax

deselect {**all** | *<routine-list>* | **at** *<line-number-list>*}

Parameter	Meaning
all	Disables all monitor points in your program.
<routine-list>	Specifies the names of one or more routines whose monitor points you want to disable. Separate routine names with a space. If two routines have the same name, prefix them with a file name followed by a colon: *<file-name>*:*<routine-name>*.
at *<line-number-list>*	Disables all monitor points at *<line-number-list>*. *<line-number-list>* specifies one or more line numbers that contain monitor points you want to disable. Separate line numbers with a space. To disable a monitor point in another source file, prefix the line number with a file name followed by a colon: *<file-name>*:*<line-number>*.

Description

The `deselect` command disables every monitor point at a line, in a routine, or in your program. The `deselect` command has no effect on monitor points that are already disabled.

When you invoke CXpa, all monitor points in your program are disabled until you enable one or more of them with a `monitor` command. CXpa ignores disabled monitor points when you profile a program.

You can use the `deselect` command in three ways:

- `deselect all`—Disables all monitor points in your program.
- `deselect` *<routine-list>*—Disables every monitor point in the specified routine.
- `deselect at` *<line-number-list>*—Disables every monitor point at the specified line number.

To disable specific monitor point types, use one of the other `deselect` commands: `deselect block`, `deselect loop`, `deselect pregion`, or `deselect routine`.

Monitor points are placed in your program's executable when you compile your program with a CXpa option. The following compiler options determine which types of monitor points are inserted into your program:

- `-pa`—Inserts routine, loop, and parallel region monitor points into your program.
- `-par`—Inserts routine monitor points only.
- `-pab`—Inserts basic block monitor points only.

For example, `deselect all` only disables basic block monitor points if you compile your program with `-pab`.

Examples

The following examples show various ways to use the `deselect` command.

```
(CXpa) deselect all
```

The previous command disables all monitor points in your program.

```
(CXpa) deselect src/routines.a:sparse
```

The previous command disables all monitor points in the Ada routine named `sparse` in the file routines.a in the src directory. In this case the file name was specified because there are two routines named `sparse` in the program. Specifying a file name enables you to distinguish between routines with identical names.

```
1.(CXpa) monitor all
2.(CXpa) deselect at 82
3.(CXpa) list monitors sub4
      R     68   SUBROUTINE SUB4
      L     76   DO I = 1,100,2
      L     79      DO WHILE (K .LT. 10)
      l p   82         DO J = 1,20,1
```

The previous example shows a scenario in which you would want to use the `deselect at` command. The line numbers are for reference only.

1. The `monitor all` command enables all monitor points in the program.
2. The `deselect at 82` command disables all monitor points at line 82.
3. The `list monitors sub4` command lists all monitor points in the routine named `sub4`. The letters to the left of the line numbers in the `list monitors` output represent monitor points. The lowercase `l` and `p` indicate that line 82 has disabled loop and parallel region monitor points.

Related Commands

`deselect block`
`deselect pregion`
`list monitors`
`run`
`deselect loop`
`deselect routine`
`monitor`

deselect block

Deselect basic block monitor points in your program.

Syntax

deselect block {**all** | **in** <*routine-list*> | **at** <*line-number-list*>}

Parameter	Meaning
all	Disables all basic block monitor points in your program.
in <*routine-list*>	Disables basic block monitor points in <*routine-list*>. <*routine-list*> specifies the names of one or more routines whose basic block monitor points you want to disable. Separate routine names with a space. If two routines have the same name, prefix them with a file name followed by a colon: <*file-name*>**:**<*routine-name*>.
at <*line-number-list*>	Disables the basic block monitor point at <*line-number-list*>. <*line-number-list*> specifies one or more line numbers that contain basic block monitor points you want to disable. Separate line numbers with a space. To disable a monitor point in another source file, prefix the line number with a file name followed by a colon: <*file-name*>**:**<*line-number*>.

Description

The `deselect block` command disables basic block monitor points at specified places in your program. The `deselect block` command has no effect on basic block monitor points that are already disabled.

When you invoke CXpa, all monitor points in your program are disabled until you enable one or more of them with a `monitor` command. CXpa ignores disabled monitor points when you profile a program.

NOTE: Basic block monitor points only exist if you have compiled your program with the `-pab` option.

You can use the `deselect block` command in three ways:

- `deselect block all`—Disables all basic block monitor points in your program.
- `deselect block in` <*routine*>—Disables all basic block monitor points in the specified routine.
- `deselect block at` <*line-number*>—Disables the basic block monitor point at the specified line.

To disable other monitor point types, use one of the other `deselect` commands: `deselect loop`, `deselect pregion`, or `deselect routine`.

Examples

The following examples show various ways to use the `deselect block` command.

```
(CXpa) deselect block all
```

The previous command disables all basic block monitor points in your program.

```
(CXpa) deselect block in src/routines.a:sparse
```

The previous command disables all basic block monitor points in the Ada routine `sparse` in the file routines.a in the src directory. In this case the file name was specified because there are two routines named `sparse` in the program. Specifying a file name enables you to distinguish between routines with identical names.

```
(CXpa)
1.(CXpa) monitor all
2.(CXpa) deselect block at 36
3.(CXpa) list monitors sub1
 b     36    PRINT *,'ENTERING SUB1'
 B     42    PRINT *,FACTOR, BOGUS, BOGUS3
```

The previous example shows a scenario in which you would want to use the `deselect at` command. The line numbers are for reference only.

1. The `monitor all` command enables all monitor points in the program.
2. The `deselect block at 36` command disables the basic block monitor point at line 36.
3. The `list monitors sub1` command lists all monitor points in the routine named `sub1`. The letters to the left of the line numbers in the `list monitors` output represent monitor points. The lowercase `b` indicates that line 36 has a disabled basic block monitor point.

```
(CXpa) deselect block in sub2
Routine sub2 has no instrumented basic block entries.
```

In the previous example, a basic block monitor point could not be disabled because the program was not compiled with the `-pab` option.

Related Commands

`deselect`
`deselect pregion`
`list monitors`
`run`
`deselect loop`
`deselect routine`
`monitor`

deselect loop

Disable loop monitor points in your program.

Syntax

deselect loop {**all** | **in** <*routine-list*> | **at** <*line-number-list*>}

Parameter	Meaning
all	Disables all loop monitor points in your program.
in <*routine-list*>	Disables loop monitor points in <*routine-list*>. <*routine-list*> specifies the names of one or more routines whose loop monitor points you want to disable. Separate routine names with a space. If two routines have the same name, prefix them with a file name followed by a colon: <*file-name*>:<*routine-name*>.
at <*line-number-list*>	Disables loop monitor points at <*line-number-list*>. <*line-number-list*> specifies one or more line numbers that contain loop monitor points you want to disable. Separate line numbers with a space. To disable a monitor point in another source file, prefix the line number with a file name followed by a colon: <*file-name*>:<*line-number*>.

Description

The `deselect loop` command disables loop monitor points at specified places in your program. The `deselect loop` command has no effect on loop monitor points that are already disabled.

When you invoke CXpa, all monitor points in your program are disabled until you enable one or more of them with a `monitor` command. CXpa ignores disabled monitor points when you profile a program.

NOTE: Loop monitor points only exist if you have compiled your program with the `-pa` option at the `-O1` optimization level or higher.

You can use the `deselect loop` command in three ways:

- `deselect loop all`—Disables all loop monitor points in your program.
- `deselect loop in` <*routine*>—Disables all loop monitor points in the specified routine.
- `deselect loop at` <*line-number*>—Disables all loop monitor points at the specified line.

To disable other monitor point types, use one of the other `deselect` commands: `deselect block`, `deselect pregion`, or `deselect routine`.

Examples

The following examples show various ways to use the `deselect loop` command.

```
(CXpa) deselect loop all
```

The previous command disables all loop monitor points in your program.

```
(CXpa) deselect loop src/routines.a:sparse
```

The previous command disables all loop monitor points in the Ada routine `sparse` in the file routines.a in the src directory. In this case the file name was specified because there are two routines named `sparse` in the program. Specifying a file name enables you to distinguish between routines with identical names.

```
(CXpa) deselect loop at 9
```

The previous command disables all loop monitor points at line 9 in the current source file.

```
1.(CXpa) monitor all
2.(CXpa) deselect loop at 6
3.(CXpa) list monitors sub2
  R       1       SUBROUTINE SUB2(MATRIX, BOGUS3)
 l P      6       DO I=1,5
 L        7          DO J=1,5
 L        9               DO K=1,30
```

The previous example shows a scenario in which you would want to use the `deselect at` command. The line numbers are for reference only.

1. The `monitor all` command enables all monitor points in the program.
2. The `deselect loop at 6` command disables the loop monitor point at line 6.
3. The `list monitors sub2` command lists all monitor points in the routine named `sub2`. The letters to the left of the line numbers in the `list monitors` output represent monitor points. The lowercase `l` indicates that line 6 has a disabled loop monitor point.

```
(CXpa) deselect loop in sub2
Routine sub2 has no loop instrumentation.
```

In the previous example, a loop monitor point cannot be disabled because the program was not compiled with the `-pa` option.

Related Commands

`deselect`
`deselect block`
`deselect pregion`
`deselect routine`
`list monitors`
`monitor`
`run`

deselect pregion

Disable parallel region monitor points in your program.

Syntax

deselect pregion {**all** | **in** <*routine-list*> | **at** <*line-number-list*>}

Parameter	Meaning
all	Disables all parallel region monitor points in your program.
in <*routine-list*>	Disables parallel region monitor points in <*routine-list*>. <*routine-list*> specifies the names of one or more routines whose parallel region monitor points you want to disable. Separate routine names with a space. If two routines have the same name, prefix them with a file name followed by a colon: <*file-name*>**:**<*routine-name*>.
at <*line-number-list*>	Disables all parallel region monitor points at <*line-number-list*>. <*line-number-list*> specifies one or more line numbers that contain parallel region monitor points you want to disable. Separate line numbers with a space. To disable a monitor point in another source file, prefix the line number with a file name followed by a colon: <*file-name*>**:**<*line-number*>.

Description

The `deselect pregion` command disables parallel region monitor points at specified places in your program. The `deselect pregion` command has no effect on parallel region monitor points that are already disabled.

When you invoke CXpa, all monitor points in your program are disabled until you enable one or more of them with a `monitor` command. CXpa ignores disabled monitor points when you profile a program.

NOTE: Parallel region monitor points only exist if you have compiled your program with the `-pa` option at the `-O3` optimization level or higher.

You can use the `deselect pregion` command in three ways:

- `deselect pregion all`—Disables all parallel region monitor points in your program.
- `deselect pregion in` *<routine>*—Disables all parallel region monitor points in the specified routine.
- `monitor pregion at` *<line-number>*—Disables all parallel region monitor points at the specified line.

To disable other monitor point types, use one of the other `deselect` commands: `deselect block`, `deselect loop`, or `deselect routine`.

Examples

The following examples show various ways to use the `deselect pregion` command.

```
(CXpa) deselect pregion all
```

The previous command disables all parallel region monitor points in your program.

```
(CXpa) deselect pregion in src/routines.a:sparse
```

The previous command disables all parallel region monitor points in the Ada routine named `sparse` in the file routines.a in the src directory. In this case the file name was specified because there are two routines named `sparse` in the program. Specifying a file name enables you to distinguish between routines with identical names.

```
(CXpa)
1.(CXpa) monitor all
2.(CXpa) deselect pregion at 82
3.(CXpa) list monitors sub4
       R       68   SUBROUTINE SUB4
      L        76   DO I = 1,100,2
      L        79      DO WHILE (K .LT. 10)
      L p      82         DO J = 1,20,1
```

The previous example shows a scenario in which you would want to use the `deselect at` command. The line numbers are for reference only.

1. The `monitor all` command enables all monitor points in the program.
2. The `deselect pregion at 82` command disables the parallel region monitor point at line 82.
3. The `list monitors sub4` command lists all monitor points in the routine named `sub4`. The letters to the left of the line numbers in the `list monitors` output represent monitor points. The lowercase `p` indicates that line 82 has a disabled parallel region monitor point.

```
(CXpa) deselect pregion in sub2
Routine sub2 has no pregions instrumentation
```

In the previous example, a parallel region monitor point cannot be disabled because the program was not compiled with the `-pa` and/or `-O3` options.

Related Commands

`deselect`
`deselect block`
`deselect loop`
`deselect routine`
`list monitors`
`monitor`
`run`

deselect routine

Disable routine monitor points in your program.

Syntax

`deselect routine` {**`all`** | *<routine-list>*}

Parameter	Meaning
`all`	Disables all routine monitor points in your program.
<routine-list>	Specifies the names of one or more routines whose routine monitor points you want to disable. Separate routine names with a space. If two routines have the same name, preface them with a file name: *<file-name>*`:`*<routine-name>*.

Description

The `deselect routine` command disables all routine monitor points in your program or all routine monitor points in a specified set of routines. The `deselect routine` command has no effect on monitor points that are already disabled.

When you invoke CXpa, all monitor points in your program are disabled until you enable one or more of them with a `monitor` command. CXpa ignores disabled monitor points when you profile a program.

NOTE: Routine monitor points only exist if you have compiled your program with the `-pa` or `-par` option.

You can use the `deselect routine` command in two ways:

- `deselect routine all`—Disables all routine monitor points in your program.
- `deselect routine` *<routine>*—Disables the routine monitor points in the specified routine.

To disable other monitor point types, use one of the other `deselect` commands: `deselect block`, `deselect loop`, or `deselect pregion`.

Examples

The following examples show various ways to use the `deselect routine` command.

```
(CXpa) deselect routine all
```

The previous command disables all routine monitor points in your program.

```
(CXpa) deselect routine src/routines.a:sparse
```

The previous command disables the routine monitor point in the Ada routine named `sparse` in the file routines.a in the src directory. In this case the file name was specified because there are two routines named `sparse` in the program. Specifying a file name enables you to distinguish between routines with identical names.

```
(CXpa) deselect routine sub1
Routine sub1 has no routine entries instrumentation
```

The output from the previous command appears because `sub1` was not compiled with `-pa` or `-par` or because `sub1` has no enabled routine monitor points to disable.

```
1.(CXpa) monitor all
2.(CXpa) deselect routine sub2
3.(CXpa) list monitors sub2
  r      1 SUBROUTINE SUB2(MATRIX, BOGUS3)
 L P     6 DO I=1,5
 L       7    DO J=1,5
 L       9       DO K=1,30
```

The previous example shows a scenario in which you would want to use the `deselect routine` command. The line numbers are for reference only.

1. The `monitor all` command enables all monitor points in the program.
2. The `deselect routine sub2` command disables the routine monitor point at the beginning of `sub2`.
3. The `list monitors sub2` command lists all monitor points in the routine named `sub2`. The letters to the left of the line numbers in

the `list monitors` output represent monitor points. The lowercase `r` indicates that the first line of `sub2` has a disabled routine monitor point. Routine monitor points only occur at the beginning of a routine.

```
(CXpa) deselect routine sub1
Routine sub1 has no instrumented routine entries.
```

The output from the previous command appears because `sub1` was not compiled with `-pa` or `-par`.

Related Commands

`deselect`
`deselect block`
`deselect loop`
`deselect pregion`
`list monitors`
`monitor`
`run`

help

Display help information.

Syntax

help [*<string>*]

Parameter	Meaning
<string>	Specifies the name or name fragment of a command that you want help on.

Description

The `help` command displays help information on the specified command. If no command is specified, CXpa displays the online version of this page.

To get help for the CXwindows version of CXpa, press the Help button on CXpa's main window.

Examples

The following example shows how to use the `help` command.

```
(CXpa) help continue
```

The previous command displays help information for the `continue` command.

```
(CXpa) help ana
```

The previous command displays help information for the `analyze` command

Related Commands

```
about cxpa
analyze
analyze loop
analyze routine
deselect
deselect loop
deselect routine
info cxpa
list
monitor
monitor loop
monitor routine
pause
rerun
set pdf
stop
add path
analyze block
analyze pregion
continue
deselect block
deselect pregion
help
info path
list monitors
monitor block
monitor pregion
path
quit
run
source
```

info cxpa

Display information about your CXpa session.

Syntax

```
info cxpa
```

Description

The `info cxpa` command lists the following information:

- **CXpa version number**—Lists the version number of CXpa you are using.
- **Current Process State**—Lists the current state of the process you are profiling. The states include: `running`, `paused`, `terminated`, `exited`, and `<no_process>`.
- **Current executable**—Lists the executable that you are profiling.
- **Created on**—Lists the date that the current executable was created.
- **Instrumentation Version**—Lists the version of the instrumented libraries linked into your program with a CXpa compiler option (`-pa`, `-par`, `-pab`).
- **Current PDF**—Lists the name of the current performance data file (PDF).
- **Created on**—Lists the date and time that the PDF was created.
- **PDF Format**—Lists the format version of the PDF.
- **Created by CXpa Version**—Lists the version number of CXpa that created the PDF.
- **Process State**—Lists the process state recorded in the PDF.
- **Executable Profiled**—Lists the name of the executable you are profiling.
- **Executable Created on**—Lists the date that the executable was created.
- **Instrumentation Version**—Lists the version of the instrumented libraries used when the PDF was created.
- **Current List File**—Lists the name of the file being used when you use the `list` or `list monitor` commands.
- **Current Working Directory**—Lists the current directory.

Examples

The following example shows the output of the `info cxpa` command.

```
(CXpa) info cxpa
             CXpa Version : 2.0
    Current Process State : paused

       Current Executable : block.out
                Created on : Thu Mar 21 15:45:31 1993
  Instrumentation Version : 2.0

              Current PDF : cxpa.pdf
               Created on : Tue Mar 2 17:54:50 1993
               PDF Format : 2.0
  Created by CXpa Version : 2.0
            Process State : paused
     Executable Profiled : /usr/smith/programs/block.out
    Executable Created on : Thu Mar 21 15:45:31 1993
  Instrumentation Version : 2.0

        Current List File : bugs.f
Current Working Directory : /usr/smith/programs/
```

Related Commands

`about cxpa` `info path`

info path

Display CXpa's search path.

Syntax

```
info path
```

Description

The `info path` command displays CXpa's search path. CXpa uses this search path to find source files needed by the `list` and `list monitor` commands.

This command is helpful when CXpa informs you that it cannot find a source file that you are attempting to list. When this occurs, you can find out what directories are currently in CXpa's search path by using the `info path` command.

To add new directories to CXpa's search path, use the `add path` command. To replace CXpa's search path, use the `path` command.

Examples

The following example shows how to use the `info path` command.

```
(CXpa) info path
/usr/frank/progs/profiles
```

The above command displays CXpa's current search path.

Related Commands

`add path`
`info cxpa`
`path`

list

List lines of text from a source file.

Syntax

`list` [<*routine*> | [[<*filename*>] [`:`] {<*first-line*> [<*last-line*>] | <*routine*>}]]

Parameter	Meaning
<*routine*>	Specifies the name of a routine to display.
<*filename*>	Specifies the name of a file to display.
<*first-line*>	Specifies a source code line number as the first line to display.
<*last-line*>	Specifies a source code line number as the last line to display.

Description

The `list` command lists lines of text from the source files that were compiled to form the current executable. Instrumented source sections are prefaced with one or more of the following letters: `r` for routine, `l` for loop, `p` for parallel region, and `b` for basic block. Lowercase letters indicate disabled monitor points, while uppercase letters indicate enabled monitor points.

With no parameters, this command lists the current source file. The current source file is either the last source file specified in a CXpa command or the source file that corresponds to the first object file given to the linker to form the current executable.

When you use the `list` command, CXpa uses the directories in its search path to find the needed source file. If CXpa cannot find a source file, use the `add path` command to add the needed directory to CXpa's search path.

The following list describes each permutation of the `list` command:

- `list`—List 10 lines of the current source file. The first time, it will list the first 10 lines. The second time, it will list the next 10 lines, and so on.
- `list` <*routine*>—List the routine specified.
- `list` <*filename*>—List 10 lines of the specified source file. This source file must be one of the files compiled to form the current executable.

- `list` *<first-line>* [*<last-line>*]—List parts of the current source file by specifying the first and possibly the last line to display. If you do not specify the last line to display, 10 lines are listed.
- `list` *<filename>*:*<first-line>* [*<last-line>*]—List parts of the specified file by specifying the first and possibly the last line to display. If you do not specify the last line to display, 10 lines are listed.
- `list` *<filename>*:*<routine>*—List the routine specified. The file name allows you to choose between routines with the same names that are in different files.

Examples

The following examples show various ways to use the `list` command. The letters beside the line numbers indicate instrumented sections of your code: `r` for routine, `l` for loop, `p` for parallel region, and `b` for basic block. Lowercase letters indicate disabled monitor points, while uppercase letters indicate enabled monitor points.

```
(CXpa) list
 r  1 PROGRAM GENERIC
    2 INTEGER VAR, VAR3, VAR4, COUNT
    3
    4 VAR = 126
    5 VAR4 = VAR * 16
    6 VAR3 = 2
    7 VAR4 = VAR4 - VAR
    8
L   9 DO 88 COUNT = 1,10,1
   10
```

The previous command lists the next 10 lines of the current source file. If this file has not been listed yet, the listing starts at one. If you use the `list` command again, the listing will start at next line, as shown in the following example.

```
(CXpa) list
 11 IF (COUNT .LT. 5) THEN
 12 VAR = VAR + 3
 13 CALL SUB1(VAR, VAR3)
 14 ELSE
 15 VAR = VAR + 32
 16 CALL SUB1(VAR, VAR3)
 17 ENDIF
 18
 19 VAR3 = VAR3 - 1
 20
```

The previous command lists the next 10 lines in the current source file.

```
(CXpa) list sub2
 r    1 SUBROUTINE SUB2(MATRIX, VAR3)
      2 INTEGER MATRIX(5,5), VAR3, VAR5
      3
      4 PRINT *,'ENTERING SUB2'
 l p  5 DO I=1,5
 l    6 DO J=1,5
      7 MATRIX(I,J) = (I + J) - VAR3
      8 ENDDO
      9 ENDDO
     10 VAR5 = 38
     11 PRINT *,'LEAVING SUB2'
     12 PRINT *
     13 VAR5 = 12
     14 END
```

The previous command lists the routine named `sub2`.

```
(CXpa) list subs.f:8 30
   8
 L 9 DO 88 COUNT = 1,10,1
  10
  11 IF (COUNT .LT. 5) THEN
  12 VAR = VAR + 3
  13 CALL SUB1(VAR, VAR3)
  14 ELSE
  15 VAR = VAR + 32
  16 CALL SUB1(VAR, VAR3)
  17 ENDIF
  18
  19 VAR3 = VAR3 - 1
  20
  21 88 CONTINUE
  22
  23 CALL SUB4
  24
  25 IF (VAR .EQ. 2000) THEN
  26 CALL SUB5
  27 END IF
  28
  29 END
  30
```

The previous command lists lines 8 through 30 in the file named subs.f.

```
(CXpa) list subs.a:funnel
```

The previous command lists the Ada routine named `funnel` in the file named subs.a. This enables you to choose between routines that have the same name but are in different source files.

```
(CXpa) list subs.a
```

The previous command lists the next 10 lines in the file named subs.a.

```
(CXpa) list 20 55
```

The previous command lists lines 20 through 55 of the current file named subs.a.

```
(CXpa) list 85
```

The previous command lists 10 lines of the current source file starting at line 85.

Related Commands

`add path`
`info path`
`list monitors`
`path`

list monitors

List monitor points in a source file.

Syntax

`list monitors` [*<routine>* | [[*<filename>*] [`:`] {*<first-line>* [*<last-line>*] | *<routine>*}]]

Parameter	Meaning
<routine>	Specifies the name of a routine to display.
<filename>	Specifies the name of a file to display.
<first-line>	Specifies a source code line number as the first line to display.
<last-line>	Specifies a source code line number as the last line to display.

Description

The `list monitors` command lists lines with monitor points from the source files that were compiled to form the current executable. Lines with monitor points are prefaced with one or more of the following letters: `r` for routine, `l` for loop, `p` for parallel region, and `b` for basic block. Lowercase letters indicate disabled monitor points, while uppercase letters indicate enabled monitor points.

With no parameters, this command lists lines with monitor points in the current source file. The current source file is either the last source file specified or the source file that corresponds to the first object file given to the linker to form the current executable.

When you use the `list` command, CXpa uses the directories in its search path to find the needed source file. If CXpa cannot find a source file, use the `add path` command to add the needed directory to CXpa's search path.

The following list describes each permutation of the `list monitors` command:

- `list monitors`—List lines with monitor points in the current source file. The first time, it lists the monitor points in the first 10 lines. The second time, it lists the monitor points in the next 10 lines, and so on.
- `list monitors` *<routine>*—List the lines with monitor points in the routine specified.

- `list monitors` *<filename>*—List lines of source code with monitor points in the file name specified. The first time, it will list the monitor points in the first 10 lines of the file. The second time, it will list the monitor points in the next 10 lines, and so on.
- `list monitors` *<first-line>* [*<last-line>*]—List the lines of source code with monitor points in the range specified.
- `list monitors` *<filename>*:*<first-line>* [*<last-line>*]—List the lines with monitor points in the specified source file in the range specified.
- `list monitors` *<filename>*:*<routine>*—List the monitor points in the routine specified. Specifying the file name allows you to choose between two routines with the same name that are in different files.

If you enter the `list monitors` command and get no output, it is because there were no lines with monitor points in the range you specified. If you do not specify a range, the implied range is the next 10 lines.

When you use the `list monitors` command, CXpa uses the directories in its search path to find the needed source file. If CXpa cannot find a source file, use the `add path` command to add the needed directory to CXpa's search path.

Examples

The following examples show various ways to use the `list monitors` command. The letters beside the line numbers indicate instrumented sections of your code: `r` for routine, `l` for loop, `p` for parallel region, and `b` for basic block. Lowercase letters indicate disabled monitor points, while uppercase letters indicate enabled monitor points.

```
(CXpa) list monitors
  r    1 PROGRAM GENERIC
 L     9 DO 88 COUNT = 1,10,1
```

The previous command lists the monitor points in the next 10 lines of the current source file.

```
(CXpa) list monitors sub2
  r   1 SUBROUTINE SUB2(MATRIX, VAR3)
 L p  6 DO I=1,5
 L    7 DO J=1,5
 L    9 DO K=1,30
```

The previous command lists the monitor points in the routine named sub2.

```
(CXpa) list monitors prog.f:68 99
  r  68 SUBROUTINE SUB4
 L   76 DO I = 1,100,2
 L   79 DO WHILE (K .LT. 10)
 L p 82 DO J = 1,20,1
  r  99 SUBROUTINE SUB5
```

The previous command lists the monitor points in lines 68 through 99 in the file named prog.f.

```
(CXpa) list monitors subs.a:funnel
```

The previous command lists the monitor points in the Ada routine named `funnel` in the file named subs.a. This enables you to choose between routines that have the same name but are in different source files.

```
(CXpa) list monitors subs.f
```

The previous command lists the monitor points in the next 10 lines of the file named subs.f.

```
(CXpa) list monitors 20 55
 R 33 SUBROUTINE SUB1(VAR, VAR3)
 R 51 SUBROUTINE SUB3(MATRIX)
```

The previous command lists the monitor points on lines 20 through 55 of the current source file.

```
(CXpa) list monitors 51
 R 51 SUBROUTINE SUB3(MATRIX)
 L 56 DO I=1,5
```

The previous command lists the monitor points in the next 10 lines of the current source file starting at line 51.

Related Commands

`add path`	`info path`
`list`	`path`

monitor

Enable every monitor point at a line, in a routine, or in your program.

Syntax

monitor {**all** | *<routine-list>* | **at** *<line-number-list>*}

Parameter	Meaning
all	Enables all monitor points in your program.
<routine-list>	Specifies the names of one or more routines whose monitor points you want to enable. Separate routine names with a space. If two routines have the same name, prefix them with a file name followed by a colon: *<file-name>*:*<routine-name>*.
at *<line-number-list>*	Enables all monitor points at *<line-number-list>*. *<line-number-list>* specifies one or more line numbers that contain monitor points you want to enable. Separate line numbers with a space. To enable a monitor point in another source file, prefix the line number with a file name followed by a colon: *<file-name>*:*<line-number>*.

Description

The `monitor` command enables every monitor point at a line number, in a routine, or in your entire program. Enabled monitor points collect performance data when you run your program.

When you invoke CXpa, all monitor points in your program are disabled until you enable one or more of them with a `monitor` command.

You can use the `monitor` command in three ways:

- `monitor all`—Enables every monitor point in your program.
- `monitor` *<routine-list>*—Enables every monitor point in the specified routine.
- `monitor at` *<line-number-list>*—Enables every monitor point at the specified line number.

To enable specific monitor point types, use one of the other monitor commands: `monitor block`, `monitor loop`, `monitor pregion`, or `monitor routine`.

Monitor points are placed in your program's executable when you compile your program with a CXpa option. The following compiler options determine which types of monitor points are inserted into your program:

- `-pa`—Inserts routine, loop, and parallel region monitor points into your program.
- `-par`—Inserts routine monitor points only.
- `-pab`—Inserts basic block monitor points only.

For example, `monitor all` only enables basic block monitor points if you compile your program with `-pab`.

Examples

The following examples show various ways to use the `monitor` command.

```
(CXpa) monitor all
```

The previous command enables all monitor points in your program.

```
(CXpa) monitor src/routines.a:sparse
```

The previous command enables all monitor points in the Ada routine named `sparse` in the file routines.a in the src directory. In this case the file name was specified because there are two routines named `sparse` in the program. Specifying a file name enables you to distinguish between routines with identical names.

```
(CXpa) monitor at 82
(CXpa) list monitors sub4
      r      68   SUBROUTINE SUB4
     l       76   DO I = 1,100,2
     l       79      DO WHILE (K .LT. 10)
     L P     82         DO J = 1,20,1
```

In the previous example, the `monitor at 82` command enables all monitor points at line 82.

The `list monitors sub4` command lists the monitor points in the routine named `sub4`. The letters to the left of the line numbers in the `list monitors` output represent monitor points. The uppercase `L` and `P` indicate that line 82 has enabled loop and parallel region monitor points.

Related Commands

`deselect`
`list monitors`
`monitor block`
`monitor loop`
`monitor pregion`
`monitor routine`
`run`
`set pdf`

monitor block

Enable basic block monitor points in your program.

Syntax

monitor block {**all** | **in** <*routine-list*> | **at** <*line-number-list*>}

Parameter	Meaning
all	Enables all basic block monitor points in your program.
in <*routine-list*>	Enables basic block monitor points in <*routine-list*>. <*routine-list*> specifies the names of one or more routines whose basic block monitor points you want to enable. Separate routine names with a space. If two routines have the same name, prefix them with a file name followed by a colon: <*file-name*>:<*routine-name*>.
at <*line-number-list*>	Enables all basic block monitor points at <*line-number-list*>. <*line-number-list*> specifies one or more line numbers that contain basic block monitor points you want to enable. Separate line numbers with a space. To enable a monitor point in another source file, prefix the line number with a file name followed by a colon: <*file-name*>:<*line-number*>.

Description

The `monitor block` command enables basic block monitor points at specified places in your program. Enabled basic block monitor points collect basic block performance data when you run your program.

When you invoke CXpa, all monitor points in your program are disabled until you enable one or more of them with a `monitor` command.

NOTE: Basic block monitor points only exist if you have compiled your program with the `-pab` option.

You can use the `monitor block` command in three ways:

- `monitor block all`—Enables all basic block monitor points in your program.
- `monitor block in` *<routine-list>*—Enables all basic block monitor points in the specified routines.
- `monitor block at` *<line-number-list>*—Enables the basic block monitor point at the specified line numbers.

To enable other monitor point types, use one of the other monitor commands: `monitor loop`, `monitor pregion`, or `monitor routine`.

Examples

The following examples show various ways to use the `monitor block` command.

```
(CXpa) monitor block all
```

The previous command enables all basic block monitor points in your program.

```
(CXpa) monitor block in sub2
Routine sub2 has no instrumented basic block entries.
```

In the previous example, a basic block monitor point could not be enabled because the program was not compiled with the `-pab` option.

```
(CXpa) monitor block in src/routines.a:sparse
```

The previous command enables all basic block monitor points in the Ada routine named `sparse` in the file routines.a in the src directory. In this case the file name was specified because there are two routines named `sparse` in the program. Specifying a file name enables you to distinguish between routines with identical names.

```
(CXpa) monitor block at 9
```

The previous command enables the basic block monitor point at line 9 in the current source file.

```
(CXpa) monitor block at src/routines.f:36
(CXpa) list monitors sub1
 B     36    PRINT *,'ENTERING SUB1'
 b     42    PRINT *,FACTOR, BOGUS, BOGUS3
```

The `monitor block at src/routines.f:36` command in the previous example enables the basic block monitor point at line 36.

The `list monitors sub1` command lists the monitor points in the routine named `sub1`. The letters to the left of the line numbers in the `list monitors` output represent monitor points. The uppercase `B` indicates that line 36 has an enabled basic block monitor point.

Related Commands

`deselect`
`list monitors`
`monitor`
`monitor loop`
`monitor pregion`
`monitor routine`
`run`
`set pdf`

monitor loop

Enable loop monitor points in your program.

Syntax

monitor loop {**all** | **in** <*routine-list*> | **at** <*line-number-list*>}

Parameter	Meaning
all	Enables all loop monitor points in your program.
in <*routine-list*>	Enables loop monitor points in <*routine-list*>. <*routine-list*> specifies the names of one or more routines whose loop monitor points you want to enable. Separate routine names with a space. If two routines have the same name, prefix them with a file name followed by a colon: <*file-name*>**:**<*routine-name*>.
at <*line-number-list*>	Enables loop monitor points at <*line-number-list*>. <*line-number-list*> specifies one or more line numbers that contain loop monitor points you want to enable. Separate line numbers with a space. To enable a monitor point in another source file, prefix the line number with a file name followed by a colon: <*file-name*>**:**<*line-number*>.

Description

The `monitor loop` command enables loop monitor points at specified places in your program. Enabled loop monitor points collect loop performance data when you run your program.

When you invoke CXpa, all monitor points in your program are disabled until you enable one or more of them with a `monitor` command.

NOTE: Loop monitor points only exist if you have compiled your program with the `-pa` option at the `-O1` optimization level or higher.

You can use the `monitor loop` command in three ways:

- `monitor loop all`—Enables all loop monitor points in your program.

- `monitor loop in` *<routine-list>*—Enables all loop monitor points in the specified routines.
- `monitor loop at` *<line-number-list>*—Enables all loop monitor points at the specified line numbers.

To enable other monitor point types, use one of the other monitor commands: `monitor block`, `monitor pregion`, or `monitor routine`.

Examples

The following examples show various ways to use the `monitor loop` command.

```
(CXpa) monitor loop all
```

The previous command enables all loop monitor points in your program.

```
(CXpa) monitor loop in sub2
Routine sub2 has no loop instrumentation
```

In the previous example, a loop monitor point cannot be enabled because the program was not compiled with the `-pa` option.

```
(CXpa) monitor loop src/routines.a:sparse
```

The previous command enables all loop monitor points in the Ada routine named `sparse` in the file routines.a in the src directory. In this case the file name was specified because there are two routines named `sparse` in the program. Specifying a file name enables you to distinguish between routines with identical names.

```
(CXpa) monitor loop at 9
```

The previous command enables all loop monitor points at line 9 in the current source file.

```
(CXpa) monitor loop at src/routines.f:6
(CXpa) list monitors sub2
  r      1      SUBROUTINE SUB2(MATRIX, BOGUS3)
 L p     6      DO I=1,5
 l       7         DO J=1,5
 l       9              DO K=1,30
```

In the previous example, the `monitor loop at src/routines.f:6` command enables the loop monitor point at line 6.

The `list monitors sub2` command lists the monitor points in the routine named `sub2`. The letters to the left of the line numbers in the `list monitors` output represent monitor points. The uppercase `L` on line 6 indicates that line 6 has an enabled loop monitor point.

Related Commands

`deselect`
`list monitors`
`monitor`
`monitor block`
`monitor pregion`
`monitor routine`
`run`
`set pdf`

monitor pregion

Enable parallel region monitor points in your program.

Syntax

monitor pregion {**all** | **in** *<routine-list>* | **at** *<line-number-list>*}

Parameter	Meaning
all	Enables all parallel region monitor points in your program.
in *<routine-list>*	Enables parallel region monitor points in *<routine-list>*. *<routine-list>* specifies the names of one or more routines whose parallel region monitor points you want to enable. Separate routine names with a space. If two routines have the same name, prefix them with a file name followed by a colon: *<file-name>*:*<routine-name>*.
at *<line-number-list>*	Enables all parallel region monitor points at *<line-number-list>*. *<line-number-list>* specifies one or more line numbers that contain parallel region monitor points you want to enable. Separate line numbers with a space. To enable a monitor point in another source file, prefix the line number with a file name followed by a colon: *<file-name>*:*<line-number>*.

Description

The `monitor pregion` command enables parallel region monitor points at specified places in your program. Enabled parallel region monitor points collect parallel region performance data when you run your program.

When you invoke CXpa, all monitor points in your program are disabled until you enable one or more of them with a `monitor` command.

NOTE: Parallel region monitor points only exist if you have compiled your program with the `-pa` option at the `-O3` optimization level or higher.

You can use the `monitor pregion` command in three ways:

- `monitor pregion all`—Enables all parallel region monitor points in your program.
- `monitor pregion in` *<routine-list>*—Enables all parallel region monitor points in the specified routines.
- `monitor pregion at` *<line-number-list>*—Enables all parallel region monitor points at the specified line numbers.

To enable other monitor point types, use one of the other monitor commands: `monitor block`, `monitor loop`, or `monitor routine`.

Examples

The following examples show various ways to use the `monitor pregion` command.

```
(CXpa) monitor pregion all
```

The previous command enables all parallel region monitor points in your program.

```
(CXpa) monitor pregion in sub2
Routine sub2 has no pregions instrumentation
```

In the previous example, a parallel region monitor point cannot be enabled because the program was not compiled with the `-pa` and/or `-O3` options.

```
(CXpa) monitor pregion in src/routines.a:sparse
```

The previous command enables all parallel region monitor points in the Ada routine named `sparse` in the file routines.a in the src directory. In this case the file name was specified because there are two routines named `sparse` in the program. Specifying a file name enables you to distinguish between routines with identical names.

```
(CXpa) monitor pregion at 9
```

The previous command enables all parallel region monitor points at line 9 in the current source file.

```
(CXpa) monitor pregion at src/routines.f:82
(CXpa) list monitors sub4
       r       68   SUBROUTINE SUB4
      1        76   DO I = 1,100,2
      1        79      DO WHILE (K .LT. 10)
      1 P      82         DO J = 1,20,1
```

In the previous example, the `monitor pregion at src/routines.f:82` command enables the parallel region monitor point at line 82.

The `list monitors sub4` command lists the monitor points in the routine named `sub4`. The letters to the left of the line numbers in the `list monitors` output represent monitor points. The uppercase `P` indicates that line 82 has an enabled parallel region monitor point.

Related Commands

`deselect`
`list monitors`
`monitor`
`monitor block`
`monitor loop`
`monitor routine`
`run`
`set pdf`

monitor routine

Enable routine monitor points in your program.

Syntax

`monitor routine {all | <routine-list>}`

Parameter	Meaning
`all`	Enables all routine monitor points in your program.
<routine-list>	Specifies the names of one or more routines whose routine monitor points you want to enable. Separate routine names by a space. If two routines have the same name, prefix them with a file name followed by a colon: *<file-name>*`:`*<routine-name>*.

Description

The `monitor routine` command enables all routine monitor points in your program or all routine monitor points in a specified set of routines. Enabled routine monitor points collect performance data for routines when you run your program.

When you invoke CXpa, all monitor points in your program are disabled until you enable one or more of them with a `monitor` command.

NOTE: Routine monitor points only exist if you have compiled your program with the `-pa` or `-par` option.

You can use the `monitor routine` command in two ways:

- `monitor routine all`—Enables all routine monitor points in your program.
- `monitor routine` *<routine-list>*—Enables the routine monitor points in the specified routines.

To enable other monitor point types, use one of the other monitor commands: `monitor block`, `monitor loop`, or `monitor pregion`.

Examples

The following examples show various ways to use the `monitor routine` command.

```
(CXpa) monitor routine all
```

The previous command enables all routine monitor points in your program.

```
(CXpa) monitor routine sub2
Routine sub2 has no routine entries instrumentation
```

In the previous example, a routine monitor point cannot be enabled because the program was not compiled with the `-pa` or `-par` option.

```
(CXpa) monitor routine src/routines.a:sparse
```

The previous command enables the routine monitor point in the Ada routine named `sparse` in the file routines.a in the src directory. In this case the file name was specified because there are two routines named `sparse` in the program. Specifying a file name enables you to distinguish between routines with identical names.

```
(CXpa) monitor routine sub1
Routine sub1 has no instrumented routine entries.
```

The output from the previous command appears because `sub1` was not compiled with `-pa` or `-par`.

```
(CXpa) monitor routine test/subs.f:sub2
(CXpa) list monitors sub2
  R      1 SUBROUTINE SUB2(MATRIX, BOGUS3)
 1 p     6 DO I=1,5
 1       7    DO J=1,5
 1       9       DO K=1,30
```

In the previous example, the `monitor routine test/subs.f:sub2` command enables the routine monitor point at the beginning of `sub2`.

The `list monitors sub2` command lists the monitor points in the routine named `sub2`. The letters to the left of the line numbers in the `list monitors` output represent monitor points. The uppercase `R` indicates that the first line of `sub2` has an enabled routine monitor point. Routine monitor points only occur at the beginning of a routine.

Related Commands

`deselect`
`list monitors`
`monitor`
`monitor block`
`monitor loop`
`monitor pregion`
`run`
`set pdf`

path

Set CXpa's search path for source files.

Syntax

path *<directory-list>*

Parameter	Meaning
<directory-list>	Specifies one or more directories, separated by a space, as CXpa's search path.

Description

The `path` command replaces CXpa's current search path with the one specified in *<directory-list>*.

By setting CXpa's search path, you tell CXpa where to look for source files.

You can append one or more directories to CXpa's current search path with the `add path` command. You can display CXpa's current search path with the `info path` command.

Examples

The following examples show how to use the `path` command.

```
(CXpa) path /usr/data /usr/data/input /usr/data/output
(CXpa) info path
        /usr/data
        /usr/data/input
        /usr/data/output
```

The above command sets CXpa's search path to the specified directories. The `info path` command lists CXpa's search path.

Related Commands

`add path`
`info path`
`list monitors`
`info cxpa`
`list`

pause

Pause profiling of a running program.

Syntax

```
pause
```

Description

The `pause` command suspends profiling of a running program and stops execution of the program. This command does not affect your profile time.

Once you pause profiling, you can use the `analyze` command to view profile data that you have collected so far. For example, when you are profiling a program that takes a long time to run, you may want to pause profiling occasionally to see the intermediate results.

NOTE: When you display performance reports for a paused program, the information in the reports is incomplete and only reflects the partial execution of your program.

If your program is paused, you can use the `continue` command to allow your program to finish execution, or you can use the `stop` command to terminate execution and data collection.

If you are using CXpa in CRT mode (invoked with `cxpa -nw`), you issue the `pause` command by pressing **CTRL-c**.

NOTE: You cannot use the `monitor` or `deselect` commands when you have paused profiling.

Examples

The following example shows a likely scenario in which you would use the `pause` command. The line numbers are for reference only.

```
1.(CXpa) run
   (Program output is displayed.)
2.(CXpa) pause (In CRT mode, pressing CTRL-c pauses profiling.)
3.(CXpa) analyze
   (Incomplete performance reports are displayed.)
4.(CXpa) continue
   (Program runs to completion, and any output is displayed.)
```

The following lines explain the previous example by number:

1. The `run` command runs the program and initiates profile data collection.
2. The `pause` command pauses profiling. If you are using CRT mode (invoked with `cxpa -nw`), pressing **CTRL-c** pauses profiling.
3. The `analyze` command displays the performance information from the data that has been collected to this point.
4. The `continue` command continues the execution of the program and resumes collecting performance data.

Related Commands

`continue`
`rerun`
`run`
`stop`

quit

Exit CXpa.

Syntax

```
quit
```

Description

The `quit` command exits you from CXpa. If you issue the `quit` command during an active profiling session, CXpa notifies you and asks if you still wish to quit. If so, the performance data file (PDF) is closed, and the program you were profiling is terminated.

Examples

This section shows examples of the `quit` command.

```
(CXpa) quit
%
```

The `quit` command exits CXpa and returns you to your shell prompt.

```
(CXpa) quit
Program is paused, quit CXpa anyway? ([y]/n)?
```

In the above example, CXpa informs you that you have paused a process and asks if you still wish to quit.

Related Commands

`pause` `stop`

rerun

Run and profile your program again using the same argument list.

Syntax

rerun [*<argument>* ...] [*<i/o_redirection>*]

Parameter	Meaning
<argument>	Specifies any number of command line arguments to the program you are profiling.
<i/o_redirection>	Redirects the program's standard input, output, or error from or to the specified file when you include one of the redirection operators (<, >, >>, >&, >>&).

Description

The `rerun` command executes the current executable with the arguments that you specified in the last `run` command. If you specify arguments with the `rerun` command, the old arguments are discarded, and `rerun` works just like the `run` command.

NOTE: The `rerun` command does not carry over file redirection from the `run` command.

Examples

The following examples show how the `rerun` command works.

```
(CXpa) run 8 5
The arguments are 8 and 5.
(CXpa) rerun
The arguments are 8 and 5.
```

The previous example shows that the `rerun` command uses the arguments specified in the last `run` command.

```
(CXpa) rerun 20 34
The arguments are 20 and 34.
```

If you specify arguments with the `rerun` command as shown in the previous example, the `rerun` command works like the `run` command.

```
(CXpa) rerun < /usr/data/input
```

The previous command runs your program, using the arguments from the last `run` command and redirects standard input from the file /usr/data/input.

```
(CXpa) rerun > /usr/data/output
```

The previous command runs your program, using the arguments from the last `run` command and redirects standard output to the file /usr/data/output.

```
(CXpa) rerun >& /usr/data/output
```

The `>&` in the previous command tells CXpa to redirect standard output and standard error to the file /usr/data/output.

Related Commands

`pause` `run` `stop`

run

Run and profile your program with the specified arguments.

Syntax

run [*<argument>* . . .] [*<i/o_redirection>*]

Parameter	Meaning
<argument>	Specifies any number of command line arguments to the program you are profiling.
<i/o_redirection>	Redirects the program's standard input, output, or error from or to the specified file when you include one of the redirection operators (<, >, >>, >&, >>&).

Description

The `run` command starts profile data collection and executes the current executable. You specify this executable when you invoke CXpa.

As your program runs, CXpa collects profile data at the monitor points that you set with the `monitor` command. CXpa writes the data it collects to the performance data file (PDF). The default PDF name is cxpa.pdf. Use the `set pdf` command to specify another PDF name.

NOTE: If you use a preexisting PDF, the performance data in the PDF will be overwritten with new data.

While your program is running, you can use the `pause` command to pause profiling. Once paused, you can use the `analyze`, `stop`, or `continue` commands.

Examples

The following examples show various ways to use the `run` command.

```
(CXpa) run
(Program runs to completion, and any output is displayed.)
```

The command in the previous example executes the current executable. No arguments are passed to the program. The program's output follows the `run` command.

```
1.(CXpa) run
Nothing is being monitored; No monitor points selected
prior to run
2.(CXpa) pause (Press CTRL-C in CRT mode.)
    (Program execution is paused.)
3.(CXpa) stop
Process 24144 terminated with signal: SIGKILL.
```

The previous scenario shows what happens when you forget to enable any monitor points. The line numbers are for reference only.

1. No monitor points were enabled before the `run` command was issued. The program runs, but no performance data is collected.
2. Execution of the program is paused with the `pause` command.
3. Execution is terminated with the `stop` command.

In CRT mode you must type **CTRL-C** to pause the program before you can enter the `stop` command.

```
(CXpa) run 18 364
```

The previous command executes the program you are profiling and supplies 18 and 364 as arguments to the program.

```
(CXpa) run < /usr/data/input
```

The previous command executes the program you are profiling and redirects standard input from the file /usr/data/input.

```
(CXpa) run > /usr/data/output
```

The previous command executes the program you are profiling and redirects standard output to the file /usr/data/output.

```
(CXpa) run >& /usr/data/output
```

The `>&` in the previous command tells CXpa to redirect standard output and standard error to the file /usr/data/output.

```
(CXpa) run > /usr/data/output >& /usr/data/errors
```

The previous example redirects standard output and standard error to different files.

Related Commands

`pause` `rerun` `stop`

set pdf

Select the name of the performance data file (PDF).

Syntax

`set pdf` *<filename>*

Parameter	Meaning
<filename>	Specifies the name of a PDF.

Description

The `set pdf` command sets the name of the performance data file (PDF). If you do not set the PDF name, CXpa uses the default PDF name of cxpa.pdf.

CXpa uses the PDF to store performance data collected from enabled monitor points when you use the `run` command. The data in a PDF is used to calculate the performance reports that appear when you use one of the `analyze` commands.

You can select a PDF created in a previous CXpa session to display a previous performance report with the `analyze` command.

Examples

The following examples show how to use the `set pdf` command.

```
(CXpa) set pdf /usr/data/my_pdf
```

The previous command sets the name of the PDF to my_pdf.

```
(CXpa) set pdf ../profiles/prog1.pdf
```

The previous command sets the name of the PDF to prog1.pdf in the ../profiles directory.

Related Commands

`analyze` `monitor` `run`

set pdf

source

Execute a CXpa command file.

Syntax

source *<filename>*

Parameter	Meaning
<filename>	Specifies the name of a CXpa command file.

Description

The `source` command executes the CXpa commands in a specified CXpa command file. Sourcing a command file is useful when you find that you are repeating a series of commands during a profiling session.

A CXpa command file is a text file containing a list of CXpa commands. Each command must be on a separate line. Comment lines are denoted with the pound sign (#).

If CXpa encounters an error in command file, CXpa stops executing the command file, returns the CXpa prompt, and lists the line with the error.

Examples

The following examples show how to use the `source` command and CXpa command files.

```
(CXpa) source my_cmd_file
```

The previous command executes the CXpa commands in the file named my_cmd_file.

```
# CXpa command file
monitor loop in sub4
run
analyze loop > sub4.report
# End of command file
```

The previous example shows the format of a CXpa command file.

Related Commands

`analyze` `monitor` `quit` `run`

stop

Stop profiling a paused program.

Syntax

```
stop
```

Description

The `stop` command stops profile data collection, saves the collected data, and terminates the process being profiled. Before you can use the `stop` command, you must pause the program first by using the `pause` command. If you are running CXpa in CRT mode (invoked with `cxpa -nw`) you press **CTRL-c** to pause the program. Then enter the `stop` command.

When you stop or pause profiling, you can use the `analyze` command to view profile data that was collected up until your program was stopped.

NOTE: When you display performance reports for a stopped program, the information in the reports is incomplete and only reflects the partial execution of your program.

Examples

The following example shows a likely scenario in which you would use the `stop` command. The line numbers are for reference only.

```
1.(CXpa) run
   (Program output is displayed.)
2.(CXpa) pause (In CRT mode, pressing CTRL-c pauses profiling.)
3.(CXpa) stop
4.(CXpa) analyze
   (Incomplete performance reports are displayed.)
```

The following lines explain the previous example by number:

1. The `run` command runs the program and initiates profile data collection.
2. The `pause` command pauses profiling. If you are using CXpa in CRT mode (invoked with `cxpa -nw`), pressing **CTRL-c** pauses profiling.

3. The `stop` command stops the profiling of your program.
4. The `analyze` command displays the collected profile data.

Related Commands

`analyze`
`continue`
`pause`
`run`

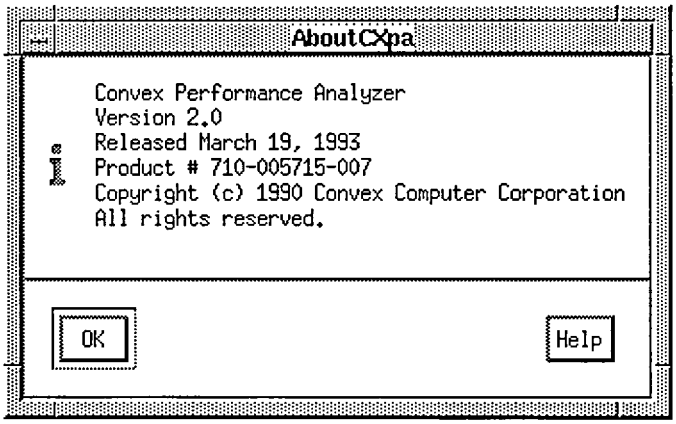
Windows

2

This chapter contains help pages for all CXpa windows and dialogs. These pages are in alphabetical order by title. Each help page is divided into the following sections:

- **Description**—Explains the purpose and functionality of the window or dialog.
- **Fields**—Describes the purpose of each field that appears in the window or dialog.
- **Buttons**—Describes the function of each button that appears in the window or dialog.
- **Context**—Describes the action that makes this window or dialog appear.
- **Reports**—Lists the reports produced by this window or dialog and refers you to the appropriate section in Chapter 3, "Reports." If the window or dialog does not produce any reports, this section does not appear.

About CXpa



Description

The About CXpa dialog box displays release information about CXpa:

- CXpa's version number
- The release date of this version of CXpa
- The product number
- Copyright information

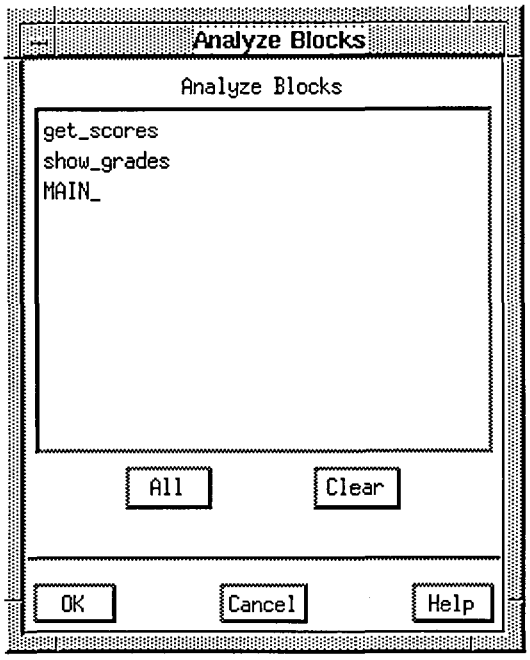
Buttons

Name	Meaning
OK	Closes this dialog box.
Help	Displays a help page for this dialog box.

Context

The About CXpa dialog box appears when you select the About CXpa... option from the Info menu on the CXpa main window.

Analyze Blocks



Description

The Analyze Blocks dialog box enables you to display Basic Block Reports by choosing routines with profiled basic blocks that you want to analyze. To select these routines:

1. Highlight the desired routines in the list.
2. Press the OK button. A performance report appears in the CXpa window.

CXpa calculates and displays a Basic Block Report from the data in the performance data file (PDF).

NOTE: CXpa can only generate this report if you use the `-pab` option to compile routines that have basic blocks you wish to monitor.

Analyze Blocks

Fields

Heading	Meaning
Analyze Blocks	Lists the routines that contain profiled basic blocks.

Buttons

Name	Meaning
All	Selects all the routines in the list by highlighting them.
Clear	Deselects all of the highlighted routines in the list.
OK	Accepts the highlighted routines in the list as the routines with basic blocks that you want to analyze and then closes this dialog box. A Basic Block Report appears in the CXpa window.
Cancel	Closes this dialog box without producing a report.
Help	Displays a help page for this dialog box.

Context

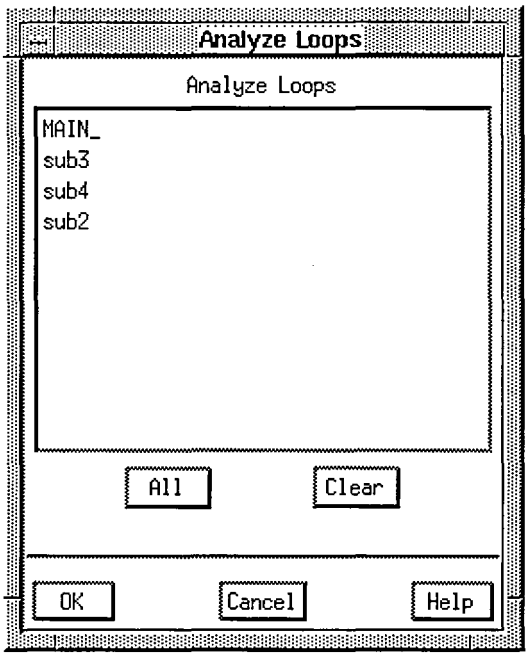
The Analyze Blocks dialog box appears when you select the Analyze Blocks... option from the Analysis menu on the CXpa window.

Use the Analyze Blocks... option when you want to view a Basic Block Report for the basic blocks in your program.

Reports

The Basic Block Report provides performance information for monitored basic blocks. For a detailed description of the Basic Block Report, refer to Chapter 3, "Reports."

Analyze Loops



Description

The Analyze Loops dialog box enables you to display Loop Reports by choosing routines with profiled loops that you want to analyze. To select these routines:

1. Highlight the desired routines in the list.
2. Press the OK button. A Loop Report appears in the CXpa window.

CXpa builds and displays a Loop Report from the information in the performance data file (PDF).

NOTE: CXpa can only generate this report if you use the `-O1` and `-pa` options to compile routines that have loops you wish to monitor.

Fields

Heading	Meaning
Analyze Loops	Lists the routines that contain profiled loops.

Analyze Loops

Buttons

Name	Meaning
All	Selects all the routines in this list by highlighting them.
Clear	Deselects all of the highlighted routines in the list.
OK	Accepts the highlighted routines in the list as the routines with loops that you want to analyze and then closes this dialog box. A Loop Report appears in the CXpa window.
Cancel	Closes this dialog box without producing a report.
Help	Displays a help page for this dialog box.

Context

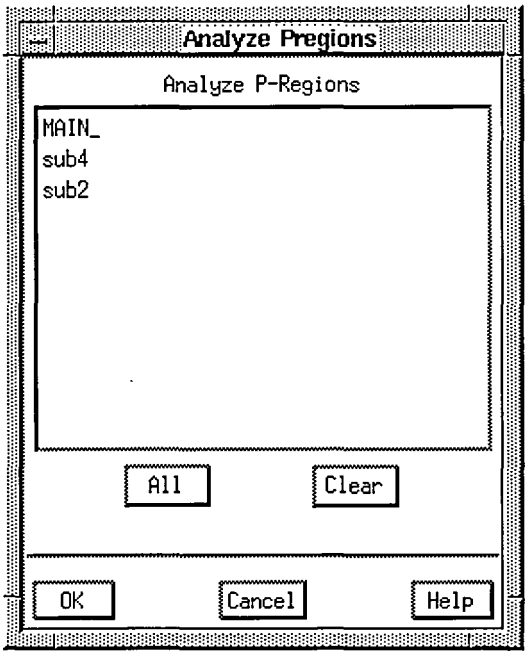
The Analyze Loops dialog box appears when you select the Analyze Loops... option from the Analysis menu on the CXpa window.

Use the Analyze Loops... option when you want to view a Loop Report for the profiled loops in your program.

Reports

The Loop Report provides performance information for monitored loops. For a detailed description of the Loop Report, refer to Chapter 3, "Reports."

Analyze P-Regions



Description

The Analyze P-Regions dialog box enables you to display Parallel Region Reports by choosing routines with profiled parallel regions that you want to analyze. To select these routines:

1. Highlight the desired routines in the Analyze P-Regions list.
2. Press the OK button. A Parallel Region Report appears in the CXpa window.

CXpa calculates and displays a Parallel Region Report from the information in the performance data file (PDF).

NOTE: CXpa can only generate this report if you use the `-O3` and `-pa` options to compile routines that have parallel regions you wish to monitor.

Analyze P-Regions

Fields

Heading	Meaning
Analyze P-Regions	Lists the profiled routines that contain profiled parallel regions.

Buttons

Name	Meaning
All	Selects all the routines in this list by highlighting them.
Clear	Deselects all of the routines highlighted in the list.
OK	Accepts the highlighted routines in the list as the routines with parallel regions that you want to analyze and then closes this dialog box. A Parallel Region Report appears in the CXpa window.
Cancel	Closes this dialog box without producing a report.
Help	Displays a help page for this dialog box.

Context

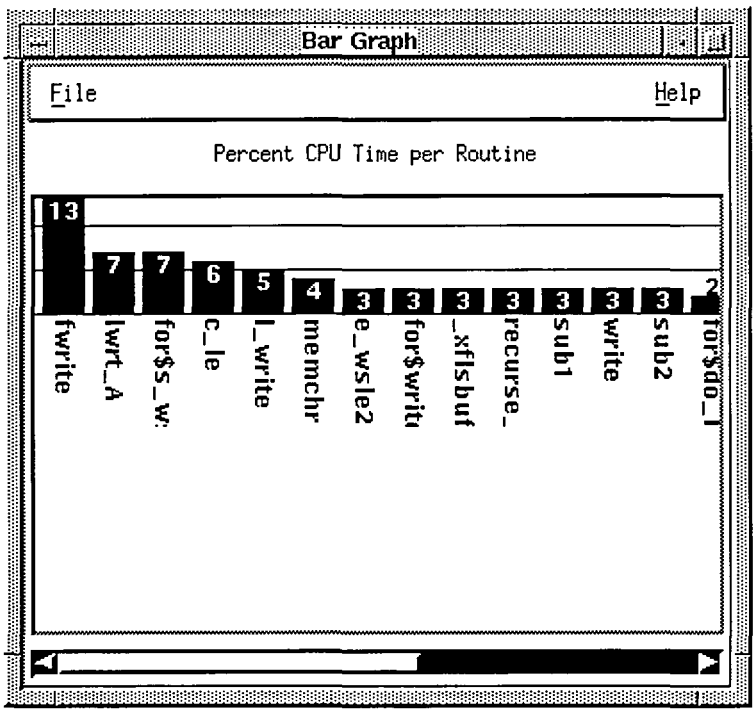
The Analyze P-Region dialog box appears when you select the Analyze P-Region... option from the Analysis menu on the CXpa window.

Use the Analyze P-Region... option when you want to view a Parallel Region Report for the profiled parallel regions in your program.

Reports

The Parallel Region Report provides performance information for monitored parallel regions. For a detailed description of the Parallel Region Report, refer to Chapter 3, "Reports."

Bar Graph



Description

The Bar Graph window enables you to view a bar graph that shows the percentage of CPU time for each monitored routine. The Bar Graph window only shows routines that consume 1% or more of the total CPU time.

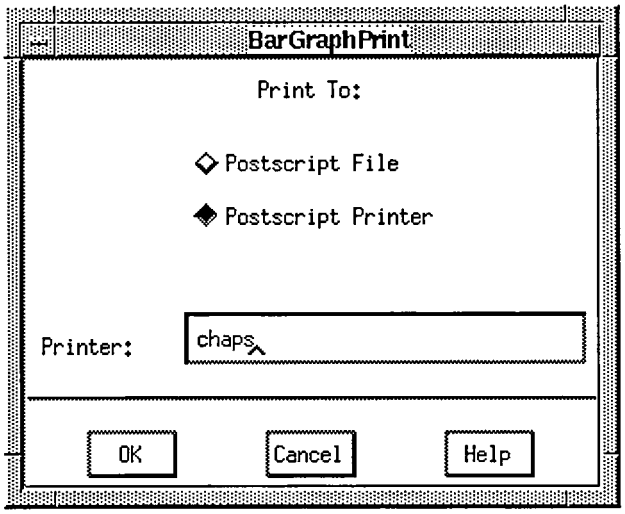
Menus

Heading	Meaning
File	Contains the Close option.
Help	Displays a help window that describes the Bar Graph window.

Context

The Bar Graph window appears when you select the Bar Graph... option from the Graphs menu on the CXpa main window.

Bar Graph Print



Description

The Bar Graph Print dialog box enables you to print the graph in the Bar Graph window to a PostScript file or a printer.

Printing to a printer

To print the bar graph to a printer:

1. Click the toggle button to the left of the PostScript Printer label. The Printer label appears to the left of the input field.
2. Enter the printer to print to in the Printer input field.
3. Press the OK button.

Printing to a PostScript file

To print the bar graph to a PostScript file:

1. Click the toggle button to the left of the PostScript File label. The Filename label appears to the left of the input field.
2. Enter the file to print to in the Filename input field.
3. Press the OK button.

Fields

Heading	Meaning
Print To:	Contains toggle buttons that give you the choice of printing to a file or a printer.
Filename:	Enables you to specify the file name to print to. The Filename label only appears if the PostScript File toggle button is on. By default, this field contains CXpa.chart.
Printer:	Enables you to specify the printer of your choice. By default, this field contains the printer named in your `$PRINTER` environment variable. The Printer label only appears if the PostScript Printer toggle button is on.

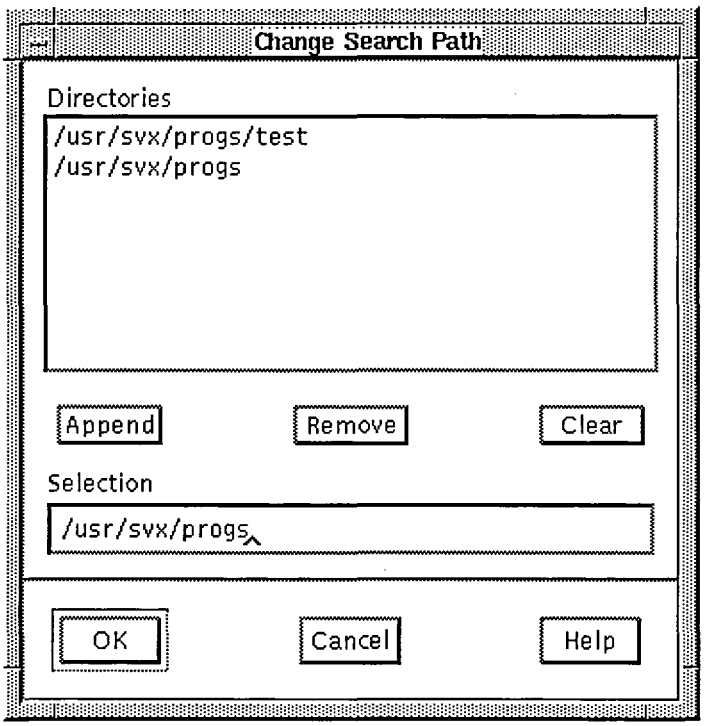
Buttons

Name	Meaning
OK	Prints the bar graph to a file or a printer and closes the Bar Graph Print dialog box.
Cancel	Closes this dialog box without printing.
Help	Displays a help page for this dialog box.

Context

The Bar Graph Print dialog box appears when you select the Print... option from the File menu of the Bar Graph window.

Change Search Path



Description

The Change Search Path dialog box enables you to change CXpa's search path. CXpa uses its search path to find source files when you list a source file or list monitor points using CXpa.

Adding a directory to CXpa's search path

To add a directory to CXpa's search path:

1. Enter a directory path name in the Selection field.
2. Press the Append button. The path name appears in the Items list.
3. Press the OK button to apply this change and close the dialog box.

Removing a directory from CXpa's search path

To remove a directory from CXpa's search path:

1. Highlight the path name to remove in the Items list.
2. Press the Remove button.
3. Press the OK button to apply this change and close the dialog box.

Closing the Change Search Path dialog box

You can close the Change Search Path dialog box by pressing the OK or Cancel button:

- If you press the Cancel button, any changes you have made are discarded and the dialog box closes.
- If you press the OK button, any changes you have made are applied and the dialog box closes.

Fields

Heading	Meaning
Directories	Lists the directories in CXpa's search path.
Selection	Enables you to enter a directory name to add to or delete from CXpa's search path.

Buttons

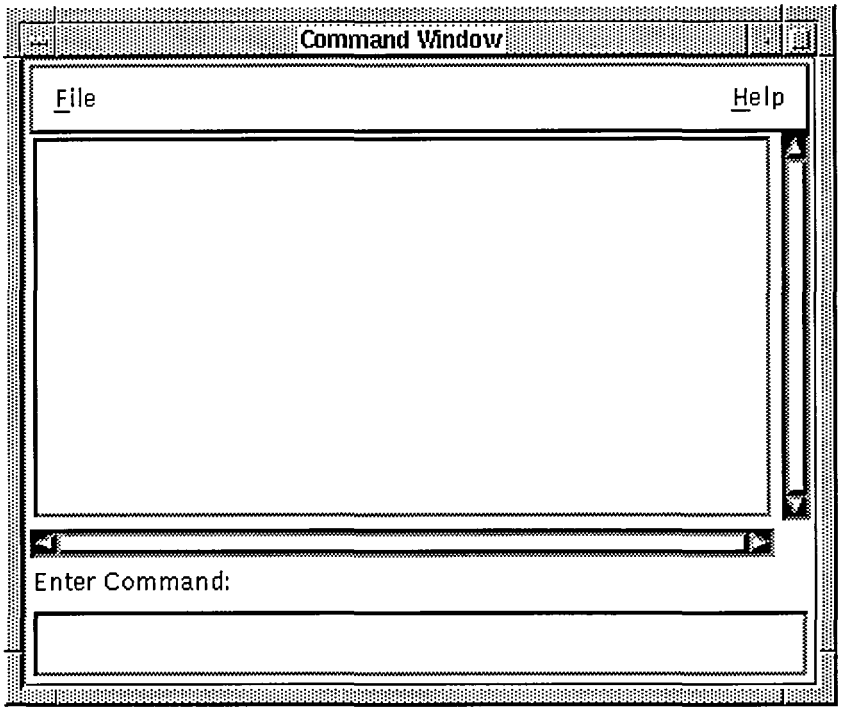
Name	Meaning
Append	Adds the directory in the Selection field to the Directories list.
Remove	Removes a highlighted directory from the Directories list.
Clear	Removes all the directories from the list.
OK	Accepts the directories in the Items list as CXpa's search path and closes this dialog box.
Cancel	Closes this dialog box without changing CXpa's search path.
Help	Displays a help page for this dialog box.

Context

The Change Search Path dialog box appears when you select the Search Path... option from the File menu on the CXpa main window.

Use this option to add directories to CXpa's search path so that it can find needed source files.

Command Window



Description

The Command Window enables you to use CXpa in command line mode.

Entering CXpa commands

To enter CXpa commands:

1. Type a CXpa command in the Enter Command field.
2. Press **RETURN**. CXpa output appears in the output area of the Command Window.

Getting help on CXpa commands

To get help on CXpa commands, type `help` *<command-name>* in the Enter Command field. If you do not specify *<command-name>*, the help page for the `help` command appears. A list of all CXpa commands appears at the bottom of the help page.

Menus

Heading	Meaning
File	Contains the Close option for the Command Window.
Help	Displays a help page that describes the Command Window.

Fields

Name	Meaning
Enter Command:	Enables you to enter any valid CXpa command.
output area	Echoes CXpa commands entered in the Enter Command field and displays CXpa output.

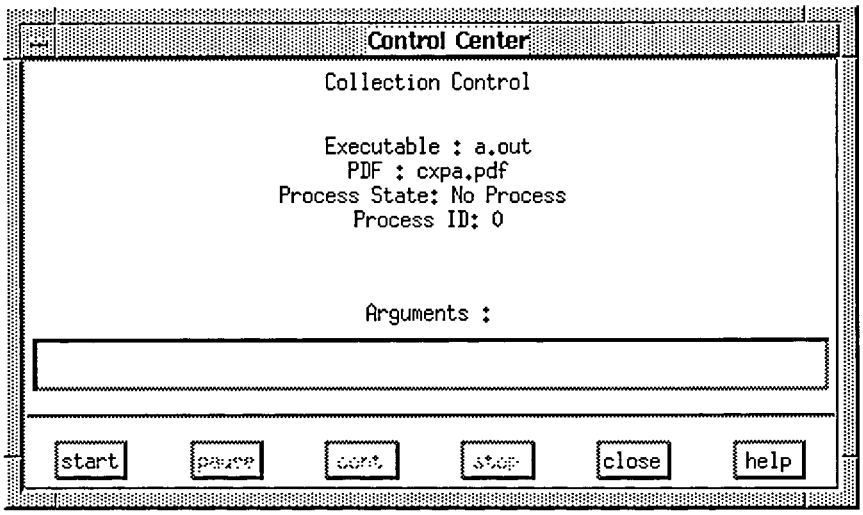
Context

The Command Window appears when you select the Command Window... option from the Windows menu on the CXpa main window.

There are three cases in which the Command Window or CRT mode (invoked with the `-nw` option) has more functionality than the mouse-oriented interface. Using the Command Window or CRT mode, you can:

- Execute command files with the `source` command
- Enable single monitor points at a specific line with one of the `monitor at` commands
- List sections of source code by line number with the `list` command

Control Center



Description

The Control Center dialog box enables you to control the execution of the program you are profiling.

Running a program with arguments

To run a program with arguments:

1. Enter the arguments to your program in the Arguments field.
2. Press the start button.

Controlling your program's execution

To control your program's execution, simply click the appropriate control buttons. Refer to the "Buttons" section of this help page for more information.

Fields

Heading	Meaning
Executable	Lists the executable that you are controlling.
PDF	Lists the PDF where performance data will be collected.

Process State	List the process state for the program you are running. These states include: Running, Paused, Finished, and Terminated.
Process ID	Lists the process ID for the program you are running.
Arguments	Enables you to specify command line arguments to your program.

Buttons

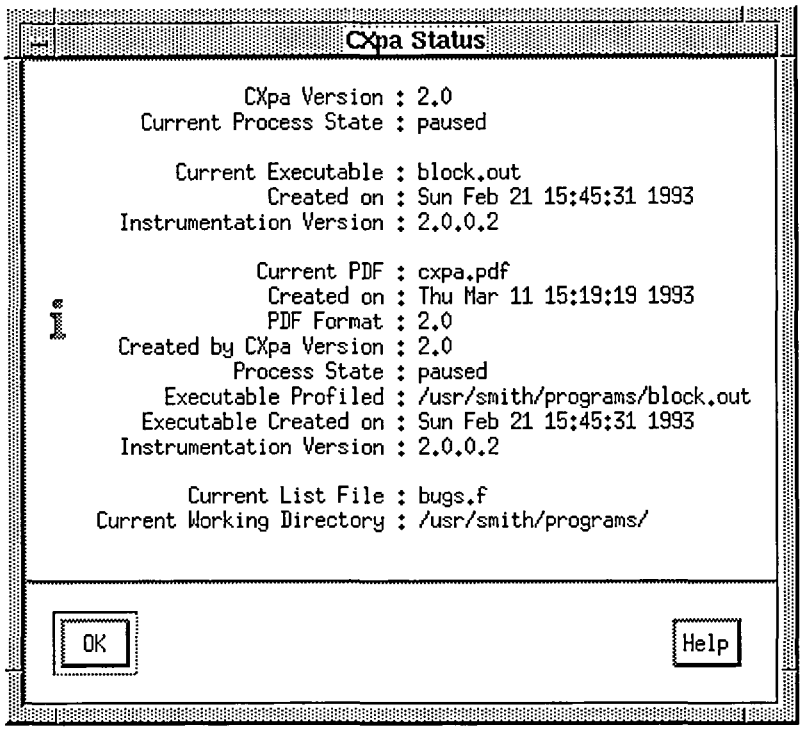
Name	Meaning
start	Runs the executable and initiates profile data collection.
pause	Pauses the executable and profile data collection.
cont	Continues a paused program and resumes profile data collection.
stop	Terminates the program and stops profile data collection.
close	Closes this dialog box.
help	Displays a help page for this dialog box.

Context

The Control Box appears when you select the Run... or Rerun... option from the Process menu on the CXpa window.

Use the Control Box to control the execution of your program and to control profile data collection.

CXpa Status



Description

The CXpa Status dialog box displays CXpa's current status.

Fields

Name	Meaning
CXpa Version:	Lists the version number of CXpa you are using.
Current Process State:	Lists the current state of the process you are profiling. The states include: `running`, `paused`, `terminated`, `exited`, and `<no_process>`.
Current Executable:	Lists the executable that you are profiling.
Created on:	Lists the date that the current executable was created.

Instrumentation Version:	Lists the version of the instrumented libraries linked into your program with a CXpa compiler option (`-pa`, `-par`, `-pab`).
Current PDF:	Lists the name of the current performance data file (PDF).
Created on:	Lists the date and time that the PDF was created.
PDF Format:	Lists the format version of the PDF.
Created by CXpa Version:	Lists the version number of CXpa that created the PDF.
Process State:	Lists the process state recorded in the PDF.
Executable Profiled:	Lists the name of the executable you are profiling.
Executable Created on:	Lists the date that the executable was created.
Instrumentation Version:	Lists the version of the instrumented libraries used when the PDF was created.
Current List File:	Lists the name of the file being used when you choose the Monitor List... option from the Info menu or when you use the `list` or `list monitor` commands.
Current Working Directory:	Lists the current directory.

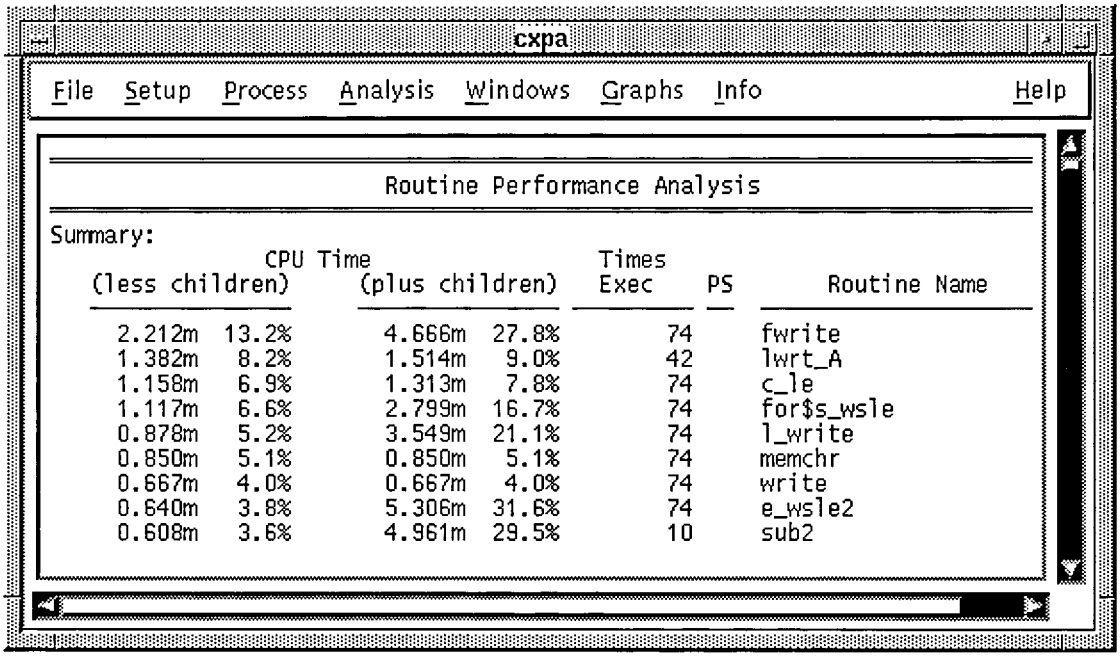
Buttons

Name	Meaning
OK	Closes this dialog box.
Help	Displays a help page for this dialog box.

Context

The CXpa Status dialog box appears when you select the Status... option from the Info menu on the CXpa main window.

CXpa Window



Description

The CXpa window is the main window to CXpa's graphical user interface. It contains various menus that enable you to profile a program that has been compiled with one of the profiling compiler options (`-pa`, `-par`, `-pab`).

After you have compiled your program with one of the profiling options, you can use CXpa's graphical user interface by following these steps:

1. Specify your program's executable name by invoking CXpa with the name of your executable:

   % **`cxpa a.out`**

2. Enable monitor points by selecting a monitor option from the Setup menu.

3. Select the Run... command from the Process menu. The Control Center dialog box appears.

4. Press the start button. A window appears that displays your program's input and output. Your program runs to completion.
5. Create and view a performance report by selecting one of the Analyze options from the Analysis menu. A performance report appears in the main CXpa window.

Menus

Heading	Meaning
File	Contains options that manipulate or specify files or directories. This menu also contains the Quit option.
Setup	Contains options to enable or disable monitor points in your program before you run your program.
Process	Contains options to run your program to collect performance data.
Analysis	Contains options that enable you to create and view performance reports.
Windows	Contains an option to bring up the CXpa Command Window. This window enables you to use CXpa in command line mode.
Graphs	Contains an option to display a bar graph of routines listed by percent CPU time.
Info	Contains options that enable you to get information about CXpa and your current profiling session.
Help	Displays CXpa's online help system.

Context

The CXpa window appears when you start CXpa.

Info Search Path

Description

The Info Search Path dialog box displays CXpa's search path.

Buttons

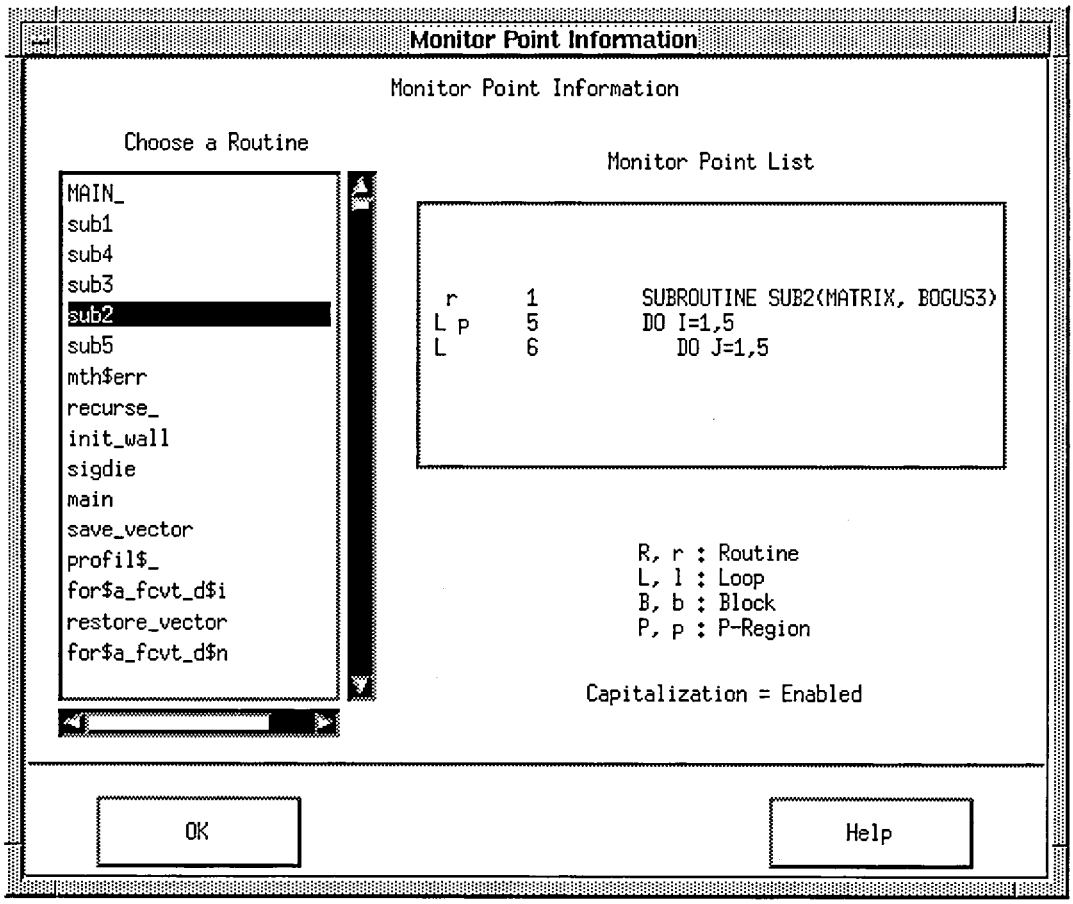
Name	Meaning
OK	Closes this dialog box.
Help	Displays a help page for this dialog box.

Context

The Info Search Path dialog box appears when you select the Search Path... option from the Info menu on the CXpa window.

This dialog box is helpful when CXpa informs you that it cannot find a source file that you are attempting to list. When this occurs, you can find out what directories are currently in CXpa's search path by using the Info Search Path dialog box. To add new directories to CXpa's search path, select the Search Path... option from the File menu.

Monitor Point Information



Description

Using the Monitor Points Information dialog box, you can list the enabled and disabled monitor points in your program.

Listing a routine's monitor points

To list a routine's monitor points:

1. Highlight the routine in the Choose a Routine column. The monitor points in that routine are displayed in the Monitor Point List column.
2. Press the OK button when you are finished. The dialog box disappears.

Viewing the Monitors list

The letters beside the line numbers indicate monitor points in your program:

- `r` or `R` for routine
- `l` or `L` for loop
- `p` or `P` for parallel region
- `b` or `B` for basic block

Lowercase letters indicate disabled monitor points, while uppercase letters indicate enabled monitor points.

Fields

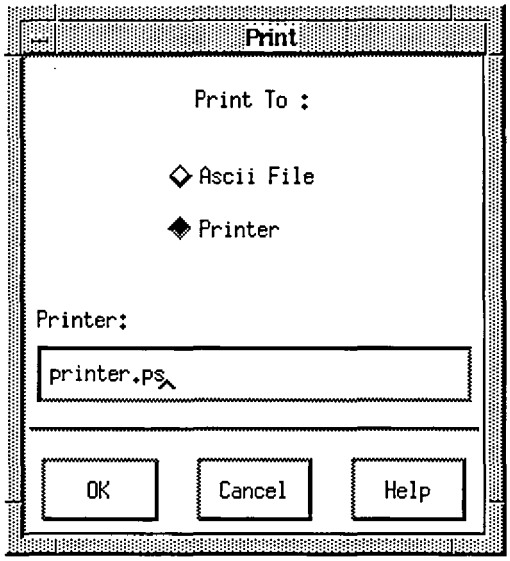
Heading	Meaning
Choose a Routine	Lists all routines with monitor points associated with the current executable.
Monitor Point List	Lists monitor points by source line numbers for the highlighted routine in the Routines list.

Buttons

Name	Meaning
OK	Closes this dialog box.
Help	Displays a help page for this dialog box.

Context

The Monitor Point Information dialog box appears when you select the Monitor List... option from the Info menu on the CXpa main window.



Description

The Print dialog box enables you to print the report displayed in the CXpa window to an ASCII file or a printer.

Printing to an ASCII file

To print the report in the CXpa window to a file:

1. Click the Ascii File toggle button. The Filename label appears above the input field.
2. Enter the file to print to in the Filename input field.
3. Press the OK button.

Printing to a printer

To print the report in the CXpa window to a printer:

1. Click the Printer toggle button. Printer appears above the input field.
2. Enter the printer to print to in the Printer input field.
3. Press the OK button.

Fields

Heading	Meaning
Print to:	Contains toggle buttons that give you the choice of printing to an ASCII file or a printer.
Filename:	Enables you to specify the file name to print to. The Filename label only appears if the Ascii File toggle button is selected. By default, this field contains CXpa.report.
Printer:	Enables you to specify the printer of your choice. By default, this field contains the printer named in your `$PRINTER` environment variable. The Printer label only appears if the Printer toggle button is selected.

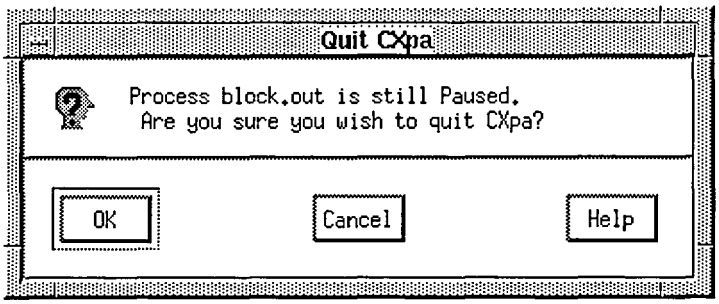
Buttons

Name	Meaning
OK	Prints the report in the CXpa window to an ASCII file or a printer and closes the Print dialog box.
Cancel	Closes this dialog box without printing.
Help	Displays a help page for this dialog box.

Context

The Print dialog box appears when you select the Print... option from the File menu.

Quit CXpa



Description

The Quit CXpa dialog box enables you to quit CXpa if a process is still running under CXpa.

To quit CXpa, press the OK button. If a process is running or paused under CXpa, it is terminated and performance data collection stops.

To cancel the quit, press the Cancel button.

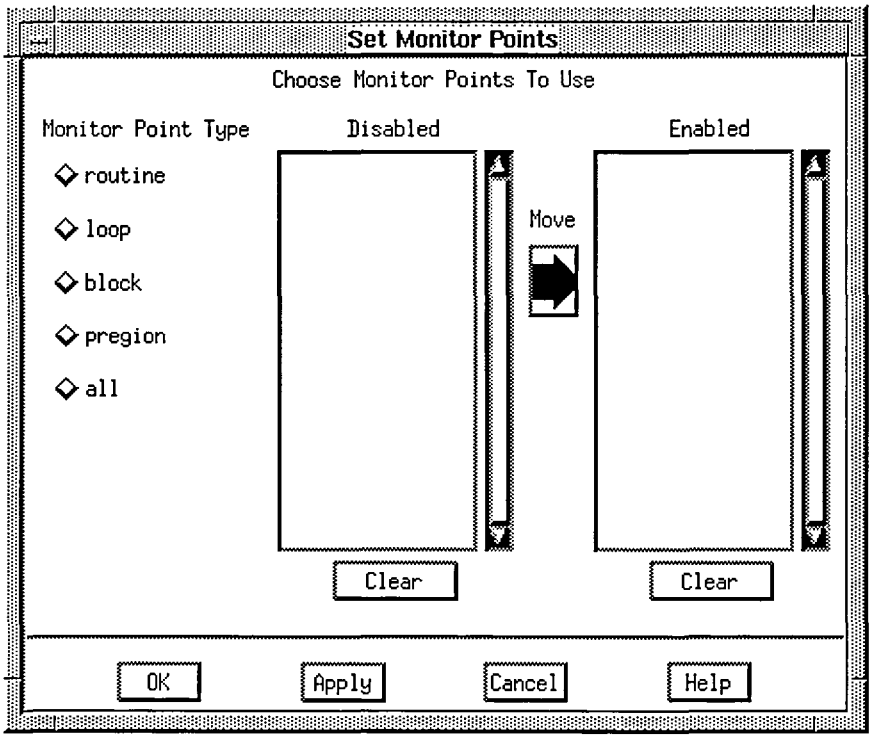
Buttons

Name	Meaning
OK	Quits CXpa. If a process is still running under CXpa, it is terminated.
Cancel	Cancels the quit and closes this dialog box.
Help	Displays a help page for this dialog box.

Context

The Quit CXpa dialog box appears when you select the Quit option from the File menu on the CXpa window if your process has not finished.

Set Monitor Points



Description

With the Set Monitor Points dialog box you can enable and disable monitor points in your program.

Enabling monitor points

To enable monitor points:

1. Choose the type of monitor point to enable by clicking one of the choices in the Monitor Point Type column. All routines with the specified type of monitor points initially appear in the Disabled list.
2. Highlight the desired routines with the mouse.
3. Click the Move arrow button to move the selected routine names to the Enabled list. You can also double click on a routine name to move it from one list to the other.

4. Click the Apply button to enable the specified monitor points.
5. Repeat steps 1 through 4 until you have enabled all the monitor points that you want enabled.
6. Press the OK button. The dialog box disappears.

Disabling monitor points

To disable an enabled monitor point:

1. Highlight the desired routine names from the Enabled list. The Move arrow button now points to the Disabled list.
2. Click the Move arrow button. The highlighted routines move to the Disabled list. You can also double click on a routine name to move it from one list to the other.
3. Click the Apply button to disable the specified monitor points.

Fields

Heading	Meaning
Monitor Point Type	Enables you to select the type of monitor points that you want to enable or disable.
Disabled	Lists the monitor points of the selected type that are not enabled.
Enabled	Lists the monitor points of the selected type that are enabled.

Buttons

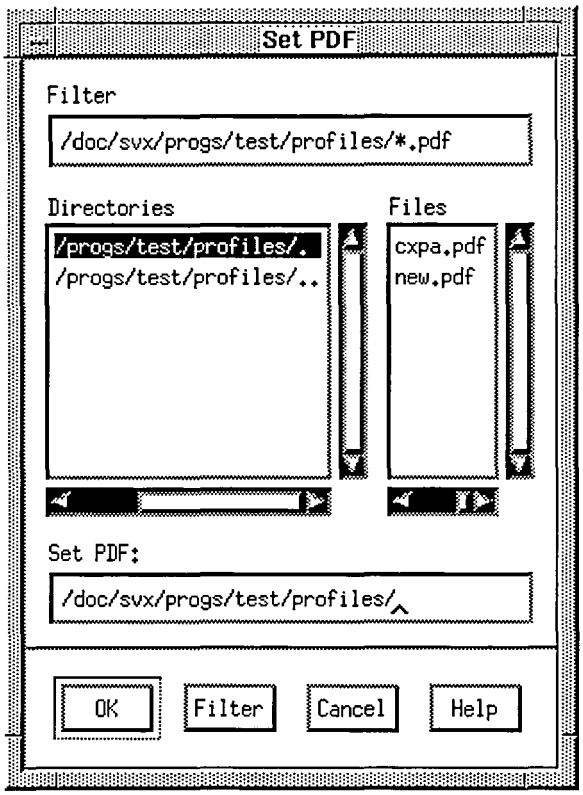
Name	Meaning
Clear	Clears the highlighting of all the highlighted monitor points in the list above.
OK	Applies any changes made and closes this dialog box.
Apply	Applies any changes made with this dialog box.
Cancel	Cancels any changes made since the last apply and closes this dialog box.
Help	Displays a help page for this dialog box.

Context

The Set Monitor Points dialog box appears when you select the Monitor... option from the Setup menu on the CXpa main window.

Use this dialog box when you want to enable a subset of the monitor points in your program.

Set PDF



Description

The Set PDF dialog box enables you to choose a performance data file (PDF). If you do not set the PDF name, CXpa uses the default PDF name of cxpa.pdf.

A PDF is a file that CXpa uses to store performance data. The data in a PDF is used to calculate the performance reports that appear when you select an Analysis option or use one of the `analyze` commands.

You can select a PDF created in a previous CXpa session to display a previous performance report with an Analysis option.

Filtering files

When the file selection box appears, the Filter field will contain the path to the current directory with *.pdf appended to it. To use the filter feature:

1. Alter the path name in the Filter field by typing in the field or by clicking on one of the directories. You can use wildcards (* or ?).
2. Press the Filter button. The Files list displays files and the Directories list displays directories that match the specification in the Filter field (usually *.pdf).

Selecting a file

You can select a PDF in the following ways:

- Double-click on a file name in the Files list.
- Highlight a file in the Files list and press the OK button.
- Type the full path name to the file in the Set PDF field and press the OK button or press **RETURN**.

Fields

Heading	Meaning
Filter	Contains a path name, usually containing wildcards, that you set to determine which files and directories appear in the Files and Directories lists.
Directories	Lists all subdirectories in the directory specified in the Filter field.
Files	Lists all files in the selected directory in the Directories list.
Set PDF:	Contains the file name to use as the PDF. Collected profiling data is placed in this file.

Buttons

Name	Meaning
OK	Accepts the file name in the Set PDF field as the new PDF.
Filter	Causes the selection box to display the files and directories of the Filter directory.
Cancel	Closes this dialog box without making any changes.
Help	Displays a help page for this dialog box.

Context

The Set PDF dialog box appears when you select the Set PDF... option from the File menu on the CXpa main window.

Use the Set PDF dialog box when you do not want to use the default PDF name of cxpa.pdf.

Reports

3

This chapter describes the reports that CXpa displays when you use one of the `analyze` commands or when you select one of the options from the Analysis menu.

CXpa can display four types of reports:

- **Basic Block Report**—Displays basic block performance data, including:
  - Number of times the block was executed
  - Starting address of the basic block
  - Percentage of total block executions in the routine
- **Loop Report**—Displays loop performance data, including:
  - Number of times each loop was executed
  - Iteration count for each loop
  - CPU time for each loop
  - Optimizations performed on each loop
  - Estimated Mflops for vectorized loops
- **Parallel Region Report**—Displays parallel region performance data, including:
  - Number of times each region was executed
  - CPU time of each region
  - Wall clock time of each region
  - Process virtual time of each region
- **Routine Report**—Displays routine performance data, including:
  - CPU time of each routine
  - Number of times each routine was executed
  - Dynamic Call Graph

Basic Block Report

Description

The Basic Block Report provides performance information for the monitored basic blocks in your program. This report consists of one Basic Block Performance Analysis table for each routine with a monitored basic block that executed.

The Basic Block Report can appear when you:

- Use the `analyze` command
- Use the `analyze block` command
- Select the Analyze All or Analyze Blocks... option from the Analysis menu of the CXpa window

NOTE: CXpa can only generate this report if you use the `-pab` option to compile routines that have basic blocks you wish to monitor.

Report

The following table is an example of the Basic Block Performance Analysis table.

```
================================================================================
                      Basic Block Performance Analysis
                                 For: sub4
================================================================================

Summary:
             100.0% of all basic block(s) executed at least once.
Details:
                                      Times       % of
 Line            PC Value             Exec     Total Exec                    PS
-------       -----------          ---------  ----------                    --
     4          0x80002164                10       12.5
     6          0x800021da                50       62.5
     6          0x800021cc                10       12.5
    10          0x80002202                10       12.5
```

The Summary section indicates the percentage of basic blocks that were executed. The Details section contains the following information:

- `Line`—Lists the starting source file line number for the basic block. The starting line number helps to trace a basic block back to the original source code.

  Due to optimizations, it is possible for many basic blocks to begin at the same line number.

- `PC Value`—Lists the starting address of the basic block. Because of optimizations, the PC Value is often more accurate than the original starting line number.

- `Times Exec`—Lists the total number of times the basic block was executed.

- `% of Total Exec`—Lists the percentage of total block executions in this routine.

- `PS`—This column is not used for basic blocks in the current version of CXpa.

Loop Report

Description

The Loop Report describes the performance of monitored loops. There is one Loop Performance Analysis table for each executed routine that has monitored loops. Each Loop Performance Analysis table contains three sections:

- **Summary**—Lists summary information such as the number of times each loop was executed, the iteration count for each loop, and the CPU time spent in each loop, including time spent in inner loops.
- **Details**—Lists detailed information such as the resulting nesting level of each loop after optimization, the compiler optimizations performed on each loop, and the CPU time spent in each loop, not including time spent in inner loops.
- **Vectorized loop**—Lists information such as vector operation spills, vector floating point instructions for each loop, and an estimate of the number of millions of floating point operations per second (Mflops). This section only appears for monitored loops that have vector floating point operations.

The Loop Report can appear when you:

- Use the `analyze` command
- Use the `analyze loop` command
- Select the Analyze All or Analyze Loops... option from the Analysis menu of the CXpa window

NOTE: CXpa can only generate this report if you use the `-O1` and `-pa` options to compile routines that have loops you wish to monitor.

Report

The Loop Report expresses CPU time in seconds unless annotated with the letter "m" for milliseconds. The following table is an example of the Loop Performance Analysis table.

```
================================================================================
                          Loop Performance Analysis
                                  For: sub4
================================================================================
Summary:
                              Times      Iteration Count          CPU Time
 Line                         Exec      Min      Max      Avg   (plus inner loops) PS
-------                     -------- ------- ------- ------- ------------------ --
   31                              1     256     256     256           0.000119
   35                              1     256     256     256           0.000042
   39                              1     254     254     254           0.000218

Details:
             Optimization  Times      Iteration Count          CPU Time
 Line    NL    Report      Exec      Min      Max      Avg   (less inner loops) PS
------- --- ----------- -------- ------- ------- ------- ------------------ --
   31    0 P/V                 1     256     256     256           0.000119

   35    0 P/V                 1     256     256     256           0.000042

   39a   0 SM,Hs               1     254     254     254           0.000031
   39b   1 S,pV                2     126     128     127           0.000187

                 Static Profile                  Estimated  Mflops
              Vector  Vector  Chime  (less inner loops)  (plus inner loops)
 Line    NL  Spills  Flops   Count      Avg       Peak      Avg       Peak    PS
------- --- ------ ------- ------- --------- --------- --------- --------- --
   31    0       0       0       0     0.000     0.000     0.000     0.000

   35    0       0       0       0     0.000     0.000     0.000     0.000

   39a   0       0       1       2     8.194    12.500     1.165    12.500
   39b   1       0       0       0     0.000     0.000     0.000     0.000
```

Summary section

The Summary section of the Loop Performance Analysis table contains the following information:

- `Line`—Lists the source line number of the loop.
- `Times Exec`—Lists the number of invocations of the loop.

- `Iteration Count`— Lists the Min, Max, and Avg iteration counts. The Iteration Count is split between Min, Max, and Avg because the iteration count can vary from one invocation of a loop to another:
  - `Min`—Lists the fewest number of iterations of this loop for a single invocation.
  - `Max`—Lists the highest number of iterations of this loop for a single invocation.
  - `Avg`—Lists the average number of iterations of this loop for a single invocation.
- `CPU Time (plus inner loops)`—Lists the total CPU time for the loop, including time spent in inner loops.
- `PS`—Lists the profiling status. If this column is blank, then the loop executed normally. Other possible profiling statuses are:
  - `e`—Indicates that the program exited at this point.
  - `p`—Indicates that the program was paused at this point and that the timing information is incomplete.
  - `t`—Indicates that the program terminated at this point.
  - `u`—Indicates that this loop was too small to gather timing data.

Details section

The following report fragment shows the Details section of the Loop Performance Analysis table.

```
Details:
                Optimization  Times       Iteration Count          CPU Time
 Line     NL      Report      Exec      Min      Max      Avg   (less inner loops) PS
------- --- ----------- -------- ------- ------- ------- ------------------ --
    31    0 P/V                1     256     256     256           0.000119

    35    0 P/V                1     256     256     256           0.000042

    39a   0 SM,Hs              1     254     254     254           0.000031
    39b   1 S,pV               2     126     128     127           0.000187
```

The Details section contains the following information:

- `Line`—Lists the starting source file line number of the loop. A letter appended to the line number indicates that the loop was split into two or more loops during optimization.
- `NL`—Lists the resulting nesting level of the loop after optimization.

- `Optimization Report`—Lists an abbreviation for the transformation that the compiler performed on the loop. The abbreviations possible are:
  - `D`—Distributed
  - `Ds`—Dynamic selection
  - `Hs`—Hoisted
  - `I`—Interchanged
  - `V`—Fully vectorized
  - `P`—Parallel
  - `PS`—Parallel strip-mined
  - `P/V`—Parallel vectorized
  - `pV`—Partially vectorized
  - `S`—Scalar
  - `SM`—Strip-mined vector
  - `UL`—Uninstrumentable
  - `V`—Fully vectorized

  For more information, refer to the *CONVEX C Optimization Guide*, the *CONVEX FORTRAN Optimization Guide*, or the *CONVEX Ada Optimization Guide*.
- `Times Exec`—Lists the number of invocations of the loop.
- `Iteration Count`—Lists the Min, Max, and Avg iteration counts. Refer to the previous description in the Summary section.
- `CPU Time (less inner loops)`—Lists the total CPU time for the loop, not including time spent in inner loops.
- `PS`—Lists the profiling status. Refer to the previous explanation of the PS column in the description of the Summary section.

Vectorized loop section

The following report fragment shows the vectorized loop section of the Loop Performance Analysis table. This section only appears for loops compiled at optimization level `-O2` that have been vectorized by the compiler.

		Static Profile			Estimated Mflops (less inner loops)		Mflops (plus inner loops)		
Line	NL	Vector Spills	Vector Flops	Chime Count	Avg	Peak	Avg	Peak	PS
31	0	0	0	0	0.000	0.000	0.000	0.000	
35	0	0	0	0	0.000	0.000	0.000	0.000	
39a	0	0	1	2	8.194	12.500	1.165	12.500	
39b	1	0	0	0	0.000	0.000	0.000	0.000	

The vectorized loop section contains the following information:

- `Line`—Lists the starting source file line number of the loop.
- `NL`—Lists the nesting level of the resulting loop after optimization.
- `Vector Spills`—Lists the number of times that a previous value in a vector register had to be restored.
- `Vector Flops`—Lists the number of arithmetic vector floating point operations in the loop. This count does not include load or store operations.
- `Chime Count`—Lists the number of chimes occurring within a given loop. Chime is an acronym for chained instruction measurement. The greater the chime count, the more resource conflicts, and therefore the less vector chaining.
- `Estimated Mflops`—Lists the estimated millions of floating point operations per second. There are two subcategories:
  - `Avg`—Lists the average number of Mflops per loop invocation. This is based on the average CPU time per invocation of the loop.
  - `Peak`—Lists the theoretical maximum number of Mflops for this loop on this architecture.
- `PS`—Lists the profiling status. Refer to the previous explanation of the PS column in the description of the Summary section.

For more information on the reporting of vectorized loops and floating point operations, refer to the *CONVEX CXpa User's Guide*.

Parallel Region Report

Description

The Parallel Region Report describes the performance of monitored parallel regions. This report consists of one Parallel Region Performance Analysis table for each executed routine with a monitored parallel region. Each Parallel Region Performance Analysis table contains:

- **Summary**—Lists summary information such as the number of times each region was executed, the CPU time of each region, the wall clock time of each region, and the process virtual time of each region.
- **Details**—Describes the performance of the threads that executed each parallel region, including CPU time, wall clock time, and chore count.

The Parallel Region Report can appear when you:

- Use the `analyze` command
- Use the `analyze pregion` command
- Select the Analyze All or Analyze P-Regions... option from the Analysis menu of the CXpa window

NOTE: CXpa can only generate this report if you use the `-O3` and `-pa` options to compile routines that have parallel regions you wish to monitor.

Report

The Parallel Region Report expresses CPU time in seconds unless annotated with the letter "m" for milliseconds. The following table is an example of the Parallel Region Performance Analysis table.

```
================================================================================
                    Parallel Region Performance Analysis
                                  For: calc
================================================================================
Summary:
          Times            All Regions          Process Virtual
 Line      Exec       CPU Time   Wall Clock Time    Time (PVT)      CPU/PVT  PS
-------  --------   ------------ --------------- ---------------  --------- --
    31      1377       0.013454        0.197947        0.102075    0.13181
    34       153       0.002840        0.070716        0.013925    0.20395
    36       153       0.001876        0.024405        0.012664    0.14814
    42         5       0.006884        0.017322        0.011449    0.60128
    42        42       0.001634        0.007869        0.004658    0.35079

Details:
                              By Region
 Line                  CPU Time   Wall Clock Time    Chore Count              PS
-------              ------------ ---------------   -------------             --
     31                  0.012366        0.040601          2573
                         0.001088        0.003545           181
     34                  0.002754        0.005857           306
                         0.000086        0.000225             0
     36                  0.001740        0.005592           440
                         0.000136        0.000369            19
     42                  0.006588        0.008465           259
                         0.000296        0.000644            11
     42                  0.001608        0.002480           105
                         0.000026        0.000064             0
```

Summary section

The Summary section of the Parallel Region Performance Analysis table contains the following information:

- `Line`—Lists the starting source file line number for the parallel region. Typically, this line number corresponds to the starting point for the parallel loop.
- `Times Exec`—Lists the number of times the parallel region was executed.
- `CPU Time`—Lists the total CPU time spent executing the parallel region.
- `Wall Clock Time`—Lists the wall clock time during execution of the parallel region. Wall clock time is time to solution and includes time when the process is idle.

- `Process Virtual Time (PVT)`—Lists the CPU time interval during which one or more threads of parallel execution were executing. This time interval does not include time spent executing kernel code or waiting on I/O.
- `CPU/PVT`—Lists the concurrency factor for the parallel region. The concurrency factor is the CPU time divided by the Process Virtual time and indicates the speed-up achieved through parallelization.
- `PS`—This column is not used in the Parallel Region Performance Analysis table for the current version of CXpa.

Details section

The Details section describes the performance of the threads that executed each parallel region. Each line represents a single thread. In the previous example report, each parallel region has two lines, indicating that two threads executed each region.

The Details section contains the following information:

- `Line`—Lists the starting source file line number for the parallel region.
- `CPU Time`—Lists the total CPU time spent by each thread in the parallel region.
- `Wall Clock Time`—Lists the total wall clock time spent by each thread in the parallel region.
- `Chore Count`—Lists the total number of chores executed by the thread. In the example above, the chore counts are unevenly distributed between the two threads for all of the parallel regions. This may indicate room for improvement in parallel optimization or machine load.

  By invoking CXpa with the `-f` option, CXpa runs in fixed scheduling mode, reserving all processors for its use. Running CXpa with fixed scheduling produces a more accurate chore count and gives a better indication of a program's parallel efficiency.

  A chore is one unit of work performed by a single thread in a parallel region. This unit of work usually corresponds to a single iteration of the body of a parallelized loop.
- `PS`—This column is not used in the Parallel Region Performance Analysis table for the current version of CXpa.

Routine Report

Description

The Routine Report describes the performance of monitored routines. This report contains:

- **Routine Performance Analysis table**—Contains an entry for each routine that was monitored and executed. The Summary section lists the total CPU time spent in each routine. The Details section lists the minimum, maximum, and average CPU time spent per call in each routine.
- **Dynamic Call Graph**—Displays an outline of the routine calls in your program.

The Routine Report can appear when you:

- Use the `analyze` command
- Use the `analyze routine` command
- Select the Analyze All or Analyze Routines... option from the Analysis menu of the CXpa window

NOTE: CXpa can only generate this report if you use the `-pa` or `-par` option to compile routines that you wish to monitor.

Report

The Routine Report express CPU time in seconds unless annotated with the letter "m" for milliseconds. The following table is an example of the Routine Performance Analysis table. The entries in the table are listed in order of CPU Time (less children).

```
================================================================================
                        Routine Performance Analysis
================================================================================
Summary:
                CPU Time                    Times
   (less children)      (plus children)    Exec    PS Routine Name
   ---------------      ---------------  --------- -- ------------------------
     5.072m  47.1%        8.435m  78.3%        10    sub1
     3.363m  31.2%        3.363m  31.2%        10    sub2
     0.629m   5.8%        0.629m   5.8%         1    sub4
     0.063m   0.6%        9.127m  84.7%         1    MAIN_

Details:
                                CPU Time
        (less children)                         (plus children)              Routine
    Min         Max         Avg         Min         Max         Avg     PS    Name
 ---------   ---------   ---------   ---------   ---------   ---------  --  --------
    0.467m      0.784m      0.507m      0.784m      1.168m      0.843m      sub1
    0.309m      0.384m      0.336m      0.309m      0.384m      0.336m      sub2
    0.629m      0.629m      0.629m      0.629m      0.629m      0.629m      sub4
    0.063m      0.063m      0.063m      9.127m      9.127m      9.127m      MAIN_
```

Summary section

The Summary section of the Routine Performance Analysis table contains the following information:

- `CPU Time (less children)`—Lists the CPU time and percent of the total CPU time for each monitored routine without including the time spent executing the routines that it called (children).
- `CPU Time (plus children)`—Lists the CPU time and percent of the total CPU time for each monitored routine including the time spent executing the routines that it called (children).
- `Times Exec`—Lists the number of times each routine was executed.
- `PS`—Lists the profiling status. If this column is blank, then the routine executed normally. The other possible profiling statuses are:
  - `e`—Indicates that the program exited at this point.
  - `m`—Indicates that this monitor point detected invalid time management due to incorrect instrumentation in your program or a library routine.

    You can work around improper instrumentation in one of your routines by disabling the monitor points in the routine.

You can work around improper instrumentation in a library routine by linking your program with uninstrumented libraries (refer to Chapter 3 of the *CONVEX CXpa User's Guide*).

- p—Indicates that the program was paused at this point and that the timing information is incomplete.
- t—Indicates that the program terminated at this point.

- Routine name—Lists the names of all monitored routines that were called, listed in order of highest CPU Time (less children).

Details section

The following report fragment shows the Details section of the Routine Performance Analysis table.

```
Details:
                                CPU Time
          (less children)                        (plus children)                  Routine
    Min          Max          Avg          Min          Max          Avg      PS    Name
 ---------    ---------    ---------    ---------    ---------    ---------  --  --------
    0.467m       0.784m       0.507m       0.784m       1.168m       0.843m      sub1
    0.309m       0.384m       0.336m       0.309m       0.384m       0.336m      sub2
    0.629m       0.629m       0.629m       0.629m       0.629m       0.629m      sub4
    0.063m       0.063m       0.063m       9.127m       9.127m       9.127m      MAIN_
```

The Details section contains the following information:

- CPU Time (less children)—Lists the minimum, maximum, and average CPU time for each routine:
  - Min—Lists the minimum time that the CPU took to execute this routine.
  - Max—Lists the maximum time that the CPU took to execute this routine.
  - Avg—Lists the average time that the CPU took to execute this routine.
- CPU Time (plus children)—Lists the CPU time for each routine including the time spent in any lower-level routines (children) called by the routine. The minimum, maximum, and average CPU times are listed.
- PS—Lists the profiling state. Refer to the previous explanation of the PS column in the description of the Routine Performance Analysis table.
- Routine name—Lists the names of all monitored routines that were called.

Dynamic Call Graph

The following table is an example of the Dynamic Call Graph.

```
                    Dynamic Call Graph
          (in topological order, cycles severed)

MAIN_:
 parents: (spontaneous call)
 children: recurse[1] sub1[10]

sub1:
 parents: MAIN_
 children: sub2[10]

sub2:
 parents: sub1

recurse_: start of cycle
 parents: recurse_ MAIN_
 children: recurse_[100]
```

The Dynamic Call Graph outlines the routine calls that took place in your program. The routines are listed in the order that they were executed (topological order). Only routines that are instrumented, monitored, and executed appear in the table.

For each routine, the table lists the calling routine (parents) and the called routines (children). A spontaneous call means that the routine could not be traced to a parent. Potential calls between routines that are not executed do not appear. Recursive calls are labeled "start of cycle" and are shown as one parent and one child in the table before its recursive cycle is severed.

The number enclosed in square brackets that follows the name of each child is the number of times the routine was called by the parent.

If you would rather not receive the Dynamic Call Graph, then invoke CXpa with the `-ncg` option. This option inhibits CXpa from displaying the Dynamic Call Graph.

Index

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

X

Order Number
DSW-253

Document Number
710-004730-004