Source Line Decorations

A, B, M, P, T	The eventpoint type that was set within this source line.
=	Execution is stopped somewhere within or at the beginning of this line. When displaying instructions, the instruction that will be executed next.
>	The line (or instruction) in the current frame where execution will resume when the called routine returns.
<	The line (or instruction) in the current frame which was executing when the called frame was created.
*	This source line corresponds to executable code.
@	The first instruction for the corresponding source line.

GUI Accelerators

NightView Menu

Ctrl+Q	Exit (Quit NightView)

Eventpoint Menu

Ctrl+B	Set Breakpoint
Ctrl+M	Set Monitorpoint
Ctrl+P	Set Patchpoint
Ctrl+T	Set Tracepoint
Ctrl+W	Set Watchpoint
Ctrl+U	Summarize/Change

View Menu

Ctrl+D	Display Group Area
Ctrl+S	Single Process Mode
Ctrl+G	Group Process Mode
Ctrl+O	Source Only
Ctrl+C	Source Preferred
Ctrl+X	Mixed Preferred
Ctrl+A	Disassembly Preferred
Ctrl+L	Show Qualifier

For More Information

See the *NightView User's Guide* (0890395).
See **nview(1)** and the on-line help.
Type **nview -help**.
See http://www.ccur.com
Call 800.245.6453 or 954.971.6248.

Publication Number 0890475-060

CONCURRENT COMPUTER CORPORATION

NightView™ Pocket Reference

Contents

NightView is a general-purpose, source-level debugger for Ada, C, C++ and Fortran. NightView can be used to debug multiple processes on the local system or on different hosts.

Invoking NightView

`nview` [`-attach` *pid*] [`-editor` *program*] [`-help`] [`-ktalk`] [`-nogui`] [`-noktalk`] [`-nolocal`] [`-nx`] [`-prompt` *string*] [`-safety` *safe-mode*] [`-simplescreen`] [`-version`] [*-Xoption ...*] [`-x` *command-file*] [`-xeditor`] [*program-name* [*corefile-name*]]

Compile with symbolic debugging information turned on (**`-g`**). Run programs by giving the program name as an argument on the NightView invocation or by using the **`run`** command within Nightview. In the graphical user interface, you can type the program invocation directly into the dialogue shell in the Dialogue I/O section of the Dialogue Window.

line_number:*unit_name* [`'specification` | `'body`]
: An Ada unit name, which may be specified as a fully expanded unit name, preceded by the line number in the source file. `'body` is the default.

**expression*
: The address given by *expression*.

Special Expression Syntax

`$`
: Refers to the last value history entry.

`$$`
: Refers to the value history entry immediately prior to `$`.

`$`*number*
: Refers to that number entry in the value history.

`${-`*number*`}`
: Refers to command history values prior to the most recent one. E.g., `${-0}` means `$`, and `${-1}` means `$$`.

`$`*identifier*
: Refers to a convenience variable.

`${`*file*:*line expression*`}`
: Evaluates *expression* in the context specified by *file* and *line* number.

`${+`*number*: *routine expression*`}`
: Goes up the stack to the *number*th previous occurrence of *routine* relative to the current frame. *expression* is then evaluated in that context. To go down the stack, use `${-`*number*: *routine expression*`}`.

`${+`*number expression*`}`
: Refers to previous stack frames, regardless of the routine name. The immediately previous frame is `+1`. To go down the stack, use `${-`*number expression*`}`.

`${=`*number expression*`}`
: Evaluates the expression in the context of the given absolute frame number, regardless of the current frame.

`${*`*frame-addr expression*`}`
: Uses *frame-addr*, a numeric constant, as an absolute frame address. It evaluates *expression* in the context of this frame address, regardless of the current frame.

Selecting Overloaded Entities

entity`#?`
: Query overloaded entity. An error message lists all choices.

entity`##`
: Turn on overloading temporarily. If there is only one choice, it is used. Otherwise, an error message lists all choices.

entity`#`*digits*
: Select a particular overloaded choice where *digits* is the number of the choice from the list printed in the error message.

Qualifier Specifiers

family-name A user-defined name that identifies a set of processes and/or dialogues, called a family.

dialogue-name A dialogue in your NightView session. In contexts where the qualifier is being used to specify a set of processes, a *dialogue-name* refers to all the processes being debugged in that dialogue.

PID The process ID of one of the processes being debugged by NightView.

dialogue-name:PID A particular process when processes in different dialogues have the same process ID.

`all` All processes or dialogues known to NightView.

`auto` Designates the one process that is currently stopped and has been stopped for the longest time.

Predefined Convenience Variables

`$_` Holds the address of the last item dumped with the **`x`** command, the address of the last eventpoint listed by an eventpoint status command, or the address of the first instruction in a line described by the **`info line`** command.

`$__` Holds the contents of the last item dumped by the **`x`** command. Depending on the dump, it holds the last word, byte, etc.

`$`*reg* Provides access to the machine registers. *reg* is any machine register name, including `$pc` (program counter), `$sp` (stack pointer), `$fp` (frame pointer), and machine-specific registers.

Location Specifiers

function_name or *unit_name* [`'specification` | `'body`] The beginning of the named function or Ada unit. `'body` is the default for Ada units.

file_name:*line_number* The first instruction generated for the given line in the given file.

file_name:*function_name* The beginning of *function_name* declared in *file_name*. (This is required for static functions that are not globally visible).

line_number The first instruction generated for *line_number* in the current file.

Command Syntax

[(*qualifier*)] *command* [*command-arg* ...]

Controlling the Debugger

Quitting NightView

Stop everything. Exit the debugger.
`quit`
Abbreviation: **`q`**

Managing Dialogues

Login to a new dialogue shell.
`login` [`/conditional`] [`/popup`] [`name=`*dialogue-name*]
[`user=`*login-name*] [*others* ...] *machine*

Specify names for programs you wish to debug.
`debug` *pattern* ...

Specify names for programs you do not wish to debug.
`nodebug` *pattern* ...

Translate object filenames for a remote dialogue.
`translate-object-file` [*from* [*to*]]
Abbreviation: **`x1`**

Terminate a dialogue.
`logout`

Specify debugger commands to be executed when a dialogue is created.
`on dialogue` [*regexp*]
`on dialogue` *regexp command*
`on dialogue` *regexp* `do`
commands
`end on dialogue`

Execute **`on dialogue`** commands for existing dialogues.
`apply on dialogue`

Dialogue Input and Output

Pass input to a dialogue.
`!` [*input line*]

Control where dialogue output goes.
`set-show` [`silent` | `notify=`*mode* | `continuous=`*mode*]
[`log`[`=`*filename*]] [`buffer=`*number*]

Control dialogue output.
`show` [*number* | `all` | `none`] [| *shell-command*]

Managing Processes

Run a program in a dialogue and wait for NightView to start debugging it.
run *input line*

Control how you are notified of events.
set-notify [silent | continuous=*mode*]

Ask about pending event notifications.
notify

Attach the debugger to a process that is already running.
attach PID

Stop debugging a list of processes.
detach

Terminate a list of processes.
kill

Establish the file containing symbolic information for a program.
symbol-file *program-name*

Create a pseudo-process for debugging an aborted program's core image file.
core-file *corefile-name* [exec-file=*program-name*]

Specify the location of the executable file corresponding to a process.
exec-file *program-name*

Specify debugger commands to be executed when a program is '**exec**'ed.
on program [*pattern*]
on program *pattern command*
on program *pattern* do
commands
end on program

Execute **on program** commands for existing processes.
apply on program

Specify debugger commands to be executed when a program is restarted.
on restart [*pattern*]
on restart *pattern command*
on restart *pattern* do
commands
end on restart

Take a restart checkpoint now.
checkpoint

Symbol Table Information

Print description of current routine arguments.
info args

Print information about local variables.
info locals [*regexp*]

Print global variable information.
info variables [*regexp*]

Determine the location of a variable.
info address *identifier*

List names of source files.
info sources [*pattern*]

List names of functions, subroutines, or Ada unit names.
info functions [*regexp*]

Print type definition information.
info types [*regexp*]

Describe the result type of an expression visible in the current context.
info whatis *expression*
Abbreviation: **whatis**

Describe the storage representation of an expression.
info representation *expression*
Abbreviation: **representation**

Print the declaration of variables or types.
info declaration *regexp*
Abbreviation: **ptype**

Print the names of the executable, symbol table and core files.
info files

Describe location of a source line.
info line [at] *location-spec*

Defining and Using Macros

Define a NightView macro.
define *macro-name* [(*arg-name* [, *arg-name*] ...)] [*text*]
define *macro-name* [(*arg-name* [, *arg-name*] ...)] as
commands
end define

Print a description of one or more NightView macros.
info macros [*regexp*]

Describe convenience variables.
info convenience

Describe expressions that are automatically displayed.
info display

Print history information.
info history [*number*]

Print information about limits on expression and location output.
info limits

Print information about registers.
info registers [*regexp*]

Print information about signals.
info signal [*signal* ...]

Describe processes being debugged.
info process

Print information about the virtual address space.
info memory [/verbose]

Print information about active dialogues.
info dialogue

Print information about an existing process family.
info family [*regexp*]

Print information about an existing eventpoint-name.
info name [*regexp*]

Print **on dialogue** commands.
info on dialogue [*name*]

Print **on program** commands.
info on program [*filename*]

Print **on restart** commands.
info on restart [output=*outname* | append=*outname*] [*program*]

Print information about Ada exception handling.
info exception *exception-name* ...
info exception *unit-name*
info exception
Abbreviation: **exception**

Describe lightweight processes, Ada tasks and C threads.
info threads

Give a name to a family of one or more processes.
family *family-name* [[-] *qualifier-spec* ...]

Control whether children should be debugged.
set-children {all[resume] | exec | none}

Control whether a process stops before exiting.
set-exit [stop] | [nostop]

Reserve a region of memory in a process.
mreserve start=*address* {length=*bytes* | end=*address*}

Setting Modes

Log session to file.
set-log *keyword filename*

Establish a default language context for variables and expressions.
set-language {ada | auto | c | c++ | fortran}

Specify the default list of processes or dialogues that will be affected by subsequent commands which accept qualifiers.
set-qualifier [*qualifier-spec* ...]

Specify the number of items to be kept in the value history list.
set-history *count*

Specify limits on the number of array elements, string characters, or program addresses printed when examining program data.
set-limits {array=*number* | string=*number* | addresses=*number*} ...

Set the string used to prompt for command input.
set-prompt *string*

Set the string used to recognize end of dialogue input mode.
set-terminator *string*

Control debugger response to dangerous commands.
set-safety [forbid | verify | unsafe]

Control whether restart information is applied.
set-restart [always | never | verify]

Define process local convenience variables.
set-local *identifier* ...

Control the size of patch areas created in your process.
set-patch-area-size {data=*data-size* | eventpoint=*eventpoint-size* | monitor=*monitor-size* | text=*text-size*} ...

Control which subprograms are interesting.
interest [*level*] [[at] [*location-spec*]]
interest inline[=*level*]
interest justlines[=*level*]
interest nodebug[=*level*]
interest threshold[=*level*]

Control the positioning of the stack when a process stops.
set-auto-frame *args ...*

Control how overloaded operators and routines are treated in expressions.
set-overload [operator={on | off}]
[routine={on | off}]

Control case sensitivity of regular expressions.
set-search [sensitive | insensitive]

Set the mode for editing commands in the simple full-screen interface.
set-editor *mode*

Control how NightView displays disassembled instructions.
set-disassembly [flavor={att|intel}]
[symbols={off|on}]

Debugger Environment Control

Set the debugger's default working directory.
cd *dirname*

Print NightView's current working directory.
pwd

Source Files

Viewing Source Files

List a source file.
list *where-spec*
list *where-spec1*, *where-spec2*
list ,*where-spec*
list *where-spec*,
list +
list -
list =
list
Abbreviation: **l**

Set the directory search path.
directory [*dirname ...*]

Miscellaneous Commands

Access the online help system.
help [*section*]

Refresh the terminal screen.
refresh

Run an arbitrary shell command.
shell [*shell-command*]

Input commands from a source file.
source *command-file*

Delay NightView command execution for a specified time.
delay [*milliseconds*]

Info Commands

Status Information

Describe any open log files.
info log

Describe current state of breakpoints, tracepoints, patchpoints, monitorpoints, and agentpoints.
info eventpoint [/verbose] [*name* | *number*] ...

Describe current state of breakpoints.
info breakpoint [/verbose] [*name* | *number*] ...
Abbreviation: **i b**

Describe current state of tracepoints.
info tracepoint [/verbose] [*name* | *number*] ...

Describe current state of patchpoints.
info patchpoint [/verbose] [*name* | *number*] ...

Describe current state of monitorpoints.
info monitorpoint [/verbose] [*name* | *number*] ...

Describe current state of agentpoints.
info agentpoint [/verbose] [*name* | *number*] ...

Describe current state of watchpoints.
info watchpoint [/verbose] [*name* | *number*] ...

Describe a stack frame.
info frame [/v] [**expression* [at *location-spec*]]

Print the search path used to locate source files.
info directories

Execute one instruction, stepping into procedures.
stepi [*repeat*]
Abbreviation: **si**

Execute one instruction, stepping over procedures.
nexti [*repeat*]
Abbreviation: **ni**

Continue execution until the current function finishes.
finish

Stop a process.
stop

Continue execution at a specific location.
jump [at] *location-spec*

Continue execution with a signal.
signal *sigid*

Specify how to handle signals and Ada exceptions in the user process.
handle [/signal] *sigid keyword ...*
handle /exception *exception-name keyword ...*
handle /exception *unit-name keyword ...*
handle /exception all *keyword ...*
handle /unhandled_exception *keyword ...*

Selecting Context

Select a new stack frame or print a description of the current stack frame.
frame [*frame-number*]
frame **expression* [at *location-spec*]
Abbreviation: **f**

Move one or more stack frames toward the caller of the current stack frame.
up [*number-of-frames*]

Move one or more stack frames toward frames called by the current stack frame.
down [*number-of-frames*]

Select the context of an Ada task, of a thread, of a lightweight process (LWP), or of a thread process.
select-context default
select-context task=*expression*
select-context thread=*expression*
select-context lwp=*lwpid*
select-context pid=*pid*

Searching

Search forward through the current source file for a specified regular expression.
forward-search [*regexp*]
Abbreviation: **fo**

Search backwards through the current source file for a specified regular expression.
reverse-search [*regexp*]

Examining and Modifying Processes

Print an ordered list of the currently active stack frames.
backtrace [*number-of-frames*]
Abbreviation: **bt**

Print the value of a language expression.
print [/*print-format-letter*] *expression*
Abbreviation: **p**

Evaluate a language expression without printing its value.
set *expression*

Print the contents of memory beginning at a given address.
x [/[*repeat-count*] [*size-letter*] [*x-format-letter*]] [*addr-expression*]

Print the value of a language expression with minimum output.
output [/*print-format-letter*] *expression*

Print arbitrary text.
echo *text*

Control items in a Data Window.
data-display [/window="*window name*"]
{/kind=*value* | *expression*}

Add to the list of expressions to be printed each time the process stops.
display [[/*print-format-letter*] *expression*]
display /[*repeat-count*] [*size-letter*] [*x-format-letter*]
addr-expression

Disable an item from the display expression list.
undisplay *item-number ...*

Enable a display item.
redisplay *item-number ...*

Print the values of language expressions using a format string.
printf *format-string*[, *expression ...*]

Dynamically load an object file, possibly replacing existing routines.
load *object*

Set the value of a vector.
vector-set *l-value* = *component, component...*
vector-set *l-value* = *repeat-count*`,`*component*

Manipulating Eventpoints

Give a name to a group of eventpoints.
name [`/add`] *name* [[-] *eventpoint-spec*] ...

Set a breakpoint.
breakpoint [`/disabled`] [`name=`*breakpoint-name*]
[[`at`] *location-spec*] [`if` *conditional-expression*]
Abbreviation: **b**

Install a small patch to a routine.
patchpoint [`/disabled`] [`name=`*patchpoint-name*]
[[`at`] *location-spec*] `eval` *expression*
patchpoint [`/disabled`] [`name=`*patchpoint-name*]
[[`at`] *location-spec*] `goto` *location-spec*

Initialize tracing.
set-trace [`eventmap=`*event-map-file*]

Set a tracepoint.
tracepoint [`/disabled`] *event-id* [`name=`*tracepoint-name*]
[[`at`] *location-spec*] [`value=`*logged-expression*]
[`if` *conditional-expression*]

Monitor the values of one or more expressions at a given location.
monitorpoint [`/disabled`] [`name=`*monitorpoint-name*]
[[`at`] *location-spec*]

Control the monitor display window.
mcontrol [`display` | `nodisplay`] [*monitorpoint-spec* ...]
mcontrol `delay` *milliseconds*
mcontrol [`off` | `on` | `stale` | `nostale` | `hold` | `release`]
Abbreviation: **hold**
Abbreviation: **release**

Insert a call to a debug agent at a given location.
agentpoint [`/disabled`] [`name=`*agentpoint-name*]
[[`at`] *location-spec*]

Set a watchpoint.
watchpoint [*eventpoint-modifier*] [`/once`] [`/read`] [`/write`]
[`name=`*watchpoint-name*] [`at`] *lvalue* [`if` *conditional-expression*]
watchpoint [*eventpoint-modifier*] [`/once`] [`/read`] [`/write`]
`/address` [`name=`*watchpoint-name*] [`at`] *address-expression*
{`size` *size-expression* | `type` *expression*}
[`if` *conditional-expression*]

Clear all eventpoints at a given location.
clear [[`at`] *location-spec*]

Attach commands to a breakpoint or monitorpoint.
commands *eventpoint-spec*
commands
`end`

Attach a condition to an eventpoint.
condition *eventpoint-spec* [*conditional-expression*]

Delete an eventpoint.
delete [*eventpoint-spec* ...]
Abbreviation: **d**

Disable an eventpoint.
disable [*eventpoint-spec* ...]

Enable an eventpoint for a specified duration.
enable [`/once` | `/delete`] [*eventpoint-spec* ...]

Attach an ignore-count to an eventpoint.
ignore *eventpoint-spec count*

Set a temporary breakpoint.
tbreak [`name=`*breakpoint-name*] [[`at`] *location-spec*]
[`if` *conditional-expression*]

Set a patchpoint that will execute only once.
tpatch [`name=`*patchpoint-name*] [[`at`] *location-spec*]
`eval` *expression*
tpatch [`name=`*patchpoint-name*] [[`at`] *location-spec*]
`goto` *location-spec*

Controlling Execution

Continue execution and wait for something to happen.
continue [*count*]
Abbreviation: **c**

Continue execution.
resume [*sigid*]

Execute one line, stepping into procedures.
step [*repeat*]
Abbreviation: **s**

Execute one line, stepping over procedures.
next [*repeat*]
Abbreviation: **n**