NightProbe User's Guide

Version 4.2

(NightStar LX)

concurrent

0897465-108
December 2010

Copyright 2010 by Concurrent Computer Corporation. All rights reserved. This publication or any part thereof is intended for use with Concurrent products by Concurrent personnel, customers, and end–users. It may not be reproduced in any form without the written permission of the publisher.

The information contained in this document is believed to be correct at the time of publication. It is subject to change without notice. Concurrent Computer Corporation makes no warranties, expressed or implied, concerning the information contained in this document.

To report an error or comment on a specific portion of the manual, photocopy the page in question and mark the correction or comment on the copy. Mail the copy (and any additional comments) to Concurrent Computer Corporation, 2881 Gateway Drive, Pompano Beach, FL 33069-4324. Mark the envelope **"Attention: Publications Department**." This publication may not be reproduced for any other reason in any form without written permission of the publisher.

Concurrent Computer Corporation and its logo are registered trademarks of Concurrent Computer Corporation. All other Concurrent product names are trademarks of Concurrent while all other product names are trademarks or registered trademarks of their respective owners.

Linux® is used pursuant to a sublicense from the Linux Mark Institute.

NightStar's integrated help system is based on Qt's Assistant from Trolltech.

NightProbe's graphing capabilities are based in part on the work of the Qwt project (http://qwt.sf.net).

Preface

Scope of Manual

This guide is designed to assist you in using NightProbe™, a real-time NightStar™ tool that provides a graphical user interface to data recording services.

Structure of Manual

This manual consists of fifteen chapters and five appendices. A brief description of the chapters and appendices is presented as follows.

- Chapter 1 introduces you to the concepts and components of NightProbe, a real-time tool that is part of the NightStar development environment.
- Chapter 2 explains how to invoke NightProbe.
- Chapter 3 explains the various selection dialogs used in NightProbe to select programs and other resources.
- Chapter 4 introduces the components of the NightProbe main window, the main control window for NightProbe.
- Chapter 5 describes the Configuration page which presents all elements of the current NightProbe session in a tree structure and allows you to manipulate them.
- Chapter 6 describes the Browse page which is the basic page used for browsing for variables in a program and viewing their values.
- Chapter 7 describes configuring the graphical user interface to suit your needs through the use of resizable and movable panels.
- Chapter 8 describes the List view which provides a scrollable list of samples organized in rows with each row containing the variables name, value, and other attributes.
- Chapter 9 describes the Table view which provides a scrollable list of values organized in columns with each column representing a variable.
- Chapter 10 describes the Spreadsheet view which provides a grid where variable labels and values are shown.
- Chapter 11 describes the Graph view which provides line graphs of values of variables over time.
- Chapter 12 describes how NightProbe can be used to record data to files for subsequent analysis or to stream recorded data to NightTrace or user applications for immediate processing.

- Chapter 13 documents the NightProbe Application Programming Interface and provides sample programs to demonstrate usage of the data structures and functions in the API.
- Appendix A discusses the NightStar License Manager (NSLM) and how to obtain and install licenses. It also discusses approaches for dealing with a firewall either on the system acting as the license server or on a system hosting the NightStar tools.
- Appendix B discusses the features and performance gains provided by various operating system kernels that provide for superior operation of the NightStar tools.
- Appendix C describes the syntax for specifying full variable names and eligibility rules.
- Appendix D provides a reference table for keyboard shortcuts to activating menu items, controlling data sampling, and traversing dialogs.
- Appendix E consists of tutorials for probing a program and other resources.
- Appendix F discusses privileged access required for some operations in NightProbe.

Syntax Notation

The following notation is used throughout this guide:

italic

Books, reference cards, and items that the user must specify appear in *italic* type. Special terms and comments in code may also appear in *italic*.

`list bold`

User input appears in **`list bold`** type and must be entered exactly as shown. Names of directories, files, commands, options and man page references also appear in **`list bold`** type.

`list`

Operating system and program output such as prompts and messages and listings of files and programs appears in `list` type. Keywords also appear in `list` type.

window

Keyboard sequences and window features such as push buttons, radio buttons, menu items, labels, and titles appear in window type.

`[ ]`

Brackets enclose command options and arguments that are optional. You do not type the brackets if you choose to specify such option or arguments.

`{ }`

Braces enclose mutually exclusive choices separated by the pipe (|) character, where one choice must be selected. You do not type the braces or the pipe character with the choice.

. . .

An ellipsis follows an item that can be repeated.

Contents

Chapter 1 Overview

Live Monitoring and Recording 1-1
Eligible Variables 1-2
Using NightProbe 1-3

Chapter 2 Invoking NightProbe

Getting Help 2-4

Chapter 3 Selection Dialogs

System Selection Dialog 3-1
Program Selection Dialog 3-4
Shared Memory Selection Dialog 3-7
Mapped Memory Selection Dialog 3-10
PCI Device Selection Dialog 3-12

Chapter 4 NightProbe Main Window

Menu Bar 4-2
File 4-2
Target 4-4
Programs 4-4
View 4-6
Record 4-8
Tools 4-12
Help 4-14
Toolbars 4-15
Pages 4-18
Preferences Dialog 4-19
General Preferences 4-20
Font Preferences 4-22
NightStar Global Fonts Dialog 4-26
Advanced Preferences 4-29

Chapter 5 Configuration Page

Target System 5-2
Programs 5-2
Views 5-4
Playback 5-5
Recording 5-7

Chapter 6 Browse Page

Filter Area 6-2
Resource and Variable Tree 6-3
Program Items 6-4
Variable Items 6-5
Variable Icon 6-5
Variable Labels 6-6
Find Bar 6-9
Variable Values 6-10
Arrays 6-10
Multi-Dimensional Arrays 6-12
Refresh Area 6-13
Refresh Discussion 6-14
Artificial Variables 6-15

Chapter 7 Panels

Chapter 8 List View

List Panel 8-1
Selection Area 8-2
Item Selection Dialog 8-3
Scrollable List Area 8-4
Variable Context Menu 8-6
Modifying a Variable 8-7
Sampling Control Area 8-8

Chapter 9 Table View

Table Panel 9-1
Selection Area 9-2
Scrollable Table Area 9-2
Variable Context Menu 9-3
Modifying a Variable 9-3
Sampling Control Area 9-4

Chapter 10 Spreadsheet View

Spreadsheet Panel 10-1
Selection Area 10-2
Scrollable Spreadsheet Area 10-2
Context Menu 10-2
Modifying a Variable 10-7
Adding Text to the Spreadsheet 10-8
Refresh Control Area 10-8

Chapter 11 Graph View

Graph Panel 11-1
Selection Area 11-2
Graph Area 11-2

Graph Context Menu . . . 11-3
Legend Item Context Menu . . . 11-4
Viewing Control Area . . . 11-7

Chapter 12 Recording

Selecting Variables for Recording . . . 12-1
Record Attribute . . . 12-2
Recording Timing Methods . . . 12-3
Clock Timing . . . 12-3
Sampling Rate Dialog . . . 12-3
Frequency Based Scheduler Timing . . . 12-4
Frequency Based Scheduler Timing Dialog . . . 12-4
Application Trigger Timing . . . 12-5
Application Trigger Dialog . . . 12-5
On Demand Timing . . . 12-6
Recording Destinations . . . 12-6
File . . . 12-6
NightTrace . . . 12-6
NightTrace Destination Dialog . . . 12-7
Program . . . 12-10
Program Destination Dialog . . . 12-11
Playback . . . 12-15

Chapter 13 NightProbe API

NightProbe Datastream API . . . 13-1
NightProbe Data Format . . . 13-1
Data Structures . . . 13-2
np_endian_type . . . 13-2
np_handle . . . 13-3
np_header . . . 13-3
np_item . . . 13-4
np_process . . . 13-4
np_type . . . 13-5
Functions . . . 13-6
np_open() . . . 13-6
np_avail() . . . 13-7
np_read() . . . 13-8
np_close() . . . 13-10
np_format() . . . 13-11
np_host_endian() . . . 13-12
np_error() . . . 13-12
Sample Programs . . . 13-13
program_output_test.c . . . 13-13
program_output_fbs_test.c . . . 13-16
NightProbe Trigger API . . . 13-20
Data Structures . . . 13-20
np_trigger_handle . . . 13-20
Functions . . . 13-21
np_trigger_open() . . . 13-21
np_trigger() . . . 13-22
np_trigger_close() . . . 13-23

np_trigger_error() 13-24
Sample Program 13-25
nprobe_trigger_test.c 13-25

Appendix A NightStar LX Licensing

License Keys A-1
License Requests A-2
License Server A-2
License Reports A-3
Firewall Configuration for Floating Licenses A-3
Serving Licenses with a Firewall A-3
Running NightStar LX Tools with a Firewall A-5
License Support A-6

Appendix B Kernel Dependencies

Advantages for NightView B-1
Advantages for NightTrace B-2
Advantages for NightProbe B-2
Advantages for NightTune B-3
Frequency Based Scheduler B-3

Appendix C Variables

Variable Name Notation C-1
Composite Types C-2
Variable Eligibility for Program Resources C-2

Appendix D Keyboard Traversal

Appendix 0 Tutorials

Probing Programs Tutorial E-1
Creating and Selecting a Program E-1
Browsing E-4
Indirecting Pointers E-6
Selecting Variables for Viewing in Other Pages E-7
Using the Graph View E-8
Exiting NightProbe E-12
C++ Sample - cpp_sample.cpp E-13
Non-Program Probing E-17
Selecting the Resource E-17
Creating a View into the Resource E-18
Viewing the Data E-18
Exiting NightProbe E-22
Conclusion E-22

Appendix 0 Privileged Access

Capabilities F-1

Index

Illustrations

Figure 1-1. Selecting Variables with the Browse page 1-2
Figure 3-1. System Selection Dialog 3-1
Figure 3-2. Program Selection Dialog 3-5
Figure 3-3. Process Selection Dialog 3-5
Figure 3-4. Shared Memory Selection Dialog 3-7
Figure 3-5. Mapped Memory Selection dialog 3-10
Figure 3-6. PCI Device Selection Dialog 3-12
Figure 3-7. PCI Scan Dialog 3-13
Figure 4-1. NightProbe main window 4-1
Figure 4-2. File menu 4-2
Figure 4-3. Target menu 4-4
Figure 4-4. Programs menu 4-4
Figure 4-5. View menu 4-6
Figure 4-6. Toolbars Menu 4-8
Figure 4-7. Record menu 4-8
Figure 4-8. Timer Menu 4-9
Figure 4-9. Record Destination Menu 4-10
Figure 4-10. Play/Record Control Menu 4-11
Figure 4-11. Tools menu 4-12
Figure 4-12. Help menu 4-14
Figure 4-13. Tab Context Menu 4-18
Figure 4-14. Rename Page Dialog 4-18
Figure 4-15. Move Page Dialog 4-19
Figure 4-16. General Preferences Page 4-20
Figure 4-17. Font Preferences Page 4-23
Figure 4-18. NightStar Global Fonts Dialog 4-26
Figure 4-19. Advanced Preferences Page 4-29
Figure 5-1. Configuration Page 5-1
Figure 5-2. Target System Context Menu 5-2
Figure 5-3. Programs tree area 5-2
Figure 5-4. Programs Context Menu 5-3
Figure 5-5. Resource Context Menu 5-3
Figure 5-6. Views tree area 5-4
Figure 5-7. Views Context Menu 5-4
Figure 5-8. View Item Context Menu 5-5
Figure 5-9. Playback tree area 5-5
Figure 5-10. Playback Context Menu 5-6
Figure 5-11. Recording tree area 5-7
Figure 5-12. Recording Timer Context Menu 5-7
Figure 5-13. Timer Item Context Menu 5-8
Figure 5-14. Recording Destinations Context Menu 5-8
Figure 5-15. Destination Item Context Menu 5-8
Figure 5-16. Recording Variables Context Menu 5-9
Figure 6-1. Browse Page 6-1
Figure 6-2. Program Items Context Menu 6-4
Figure 6-3. Variable Context Menu 6-6
Figure 6-4. Variable Info dialog 6-7
Figure 6-5. Browse Page Find Bar 6-9

Figure 6-6. Array Variable With Extension Icon 6-11
Figure 6-7. Show Subscripts Dialog 6-11
Figure 6-8. Array Expansion in Browse Tree 6-12
Figure 6-9. Show Subscripts Dialog for Multi-dimensional array 6-12
Figure 6-10. Portion of Three-Dimensional Array Shown 6-13
Figure 6-11. Artificial Variable Definition Dialog 6-16
Figure 6-12. Type Tree with Symbol File 6-17
Figure 7-1. Viewing Page with List and Graph Panels 7-1
Figure 7-2. Panel Detaches from Page 7-2
Figure 7-3. Panel Movement in Progress 7-3
Figure 7-4. Graph Panel on Top of List Panel 7-4
Figure 7-5. Table View added to Page 7-5
Figure 7-6. Panel in Motion Creating Tab 7-6
Figure 8-1. List View 8-1
Figure 8-2. Item Selection Dialog 8-3
Figure 8-3. Detached Browse Page 8-4
Figure 8-4. Tooltip for Variable Name 8-5
Figure 8-5. List View Variable Context Menu 8-6
Figure 8-6. Modifying a Variable in a List View 8-7
Figure 9-1. Table View 9-1
Figure 9-2. Table View Variable Context Menu 9-3
Figure 9-3. Modifying a Variable in a Table View 9-4
Figure 10-1. Spreadsheet View 10-1
Figure 10-2. Spreadsheet View Cell Context Menu 10-3
Figure 10-3. Font Selection dialog 10-4
Figure 10-4. Text Alignment Selection dialogs 10-4
Figure 10-5. Cell Color Selection dialog 10-5
Figure 10-6. Edit Thresholds/Gradients Dialog 10-6
Figure 10-7. Modifying a Variable in a Spreadsheet View 10-8
Figure 11-1. Graph View 11-1
Figure 11-2. Graph View Context Menu 11-3
Figure 11-3. Edit Curve Attributes Dialog 11-4
Figure 11-4. Graph View Scaling Dialog 11-6
Figure 11-5. Relationship of Source Domain to Graph Domain 11-7
Figure 12-1. Selecting Variables for Recording from the Browse Page 12-1
Figure 12-2. Scalar Item Context Menu with Record Attribute Set 12-2
Figure 12-3. List of Recorded Variables Shown in Configuration Page 12-2
Figure 12-4. Sampling Rate Dialog 12-3
Figure 12-5. Frequency Based Scheduler Timing Dialog 12-4
Figure 12-6. Application Trigger Dialog 12-5
Figure 12-7. NightTrace Destination Dialog 12-7
Figure 12-8. NightTrace Destination Dialog Advanced Tab 12-9
Figure 12-9. Program Destination Dialog 12-11
Figure 12-10. Program Destination Dialog FBS Tab 12-13
Figure 12-11. Program Destination Dialog Advanced Tab 12-14
Figure 13-1. Structure of NightProbe datastream 13-2

Tables

Table D-1. NightProbe Accelerators D-2
Table F-1. Recommended /etc/pam.d Configuration F-2

1
Overview

NightProbe is a graphical tool for viewing, recording, and modifying data within a variety of resources:

- executing programs
- shared memory segments
- memory-mapped files and devices
- PCI devices

For brevity within the remainder of this document, we refer to any of these resources as *programs*, unless explicitly stated otherwise.

Data is sampled using non-intrusive techniques to guarantee short response time and minimal impact on the target resources and the target system. The source code of target programs does not need to be modified or recompiled in order to be monitored. Executing programs can be monitored and recorded without being stopped and restarted.

NOTE

Non-intrusive monitoring depends on the capabilities of the underlying operating system kernel. If these features are available, NightProbe makes use of them. Otherwise, it falls back to the next least intrusive technique available. See "Kernel Dependencies" on page B-1 for more information.

NightProbe can be run on a different processor or system from the target resource, which minimizes NightProbe's impact upon the target system.

Furthermore, NightProbe can probe executable programs which have been stripped of debug and symbol information, such as those in deployed scenarios. In such cases, a copy of the program containing the necessary debug and symbol information must be available for reading on the host system.

Live Monitoring and Recording

Live monitoring refers to displaying the value of variables for interactive visual inspection. Variables can be modified during live monitoring as well.

Data recording refers to sampling memory locations in target programs and recording that data for subsequent analysis. Memory locations may be identified by logical address, offset, or by variable name. NightProbe allows you to record data to a file in NightProbe

Datastream API format (see "NightProbe Datastream API" on page 13-1), in NightTrace data format, or stream data to NightTrace or an executing user program.

Eligible Variables

Variables with static base addresses may be sampled. Pointer variables may be indirected as well with some stipulations. Supported languages include C, C++, and Fortran, as well as Ada programs built with the Concurrent MAXAda compiler.

You can monitor and record any valid memory location in a program's address space, whether that location corresponds to a named variable or not.

Variables are selected from program symbol files using the Browse Page as shown in the following figure:

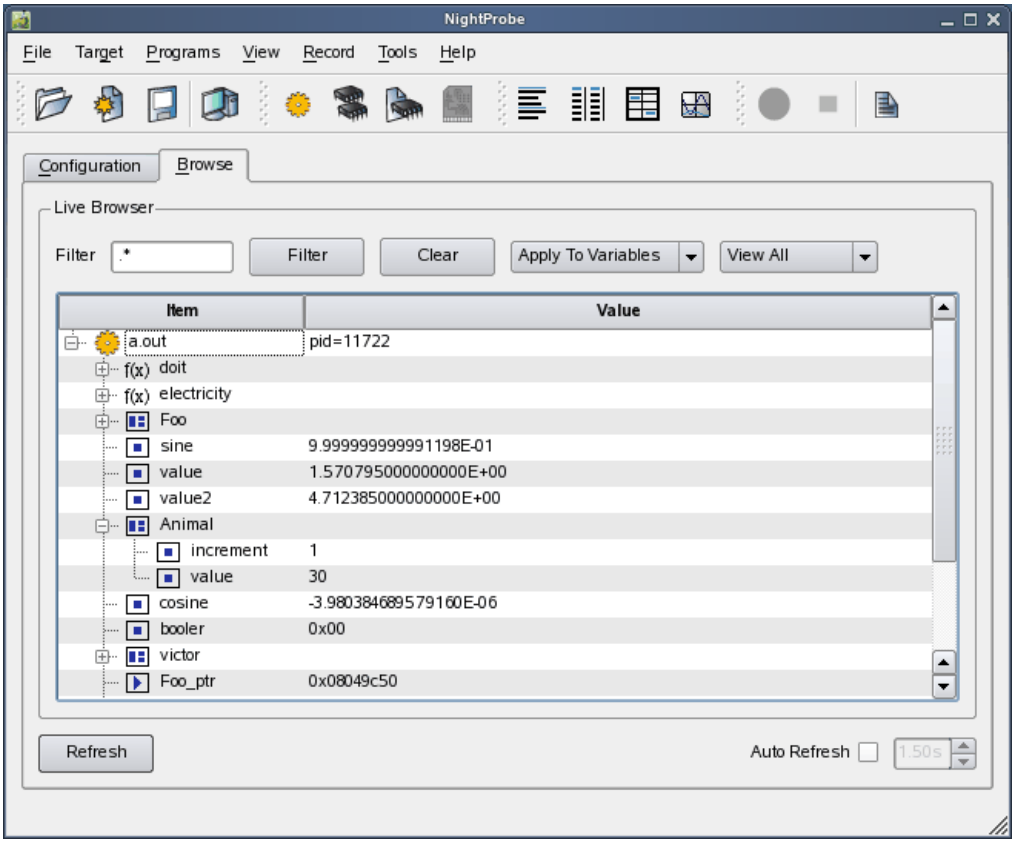


Figure 1-1. Selecting Variables with the Browse page

When probing resources other than programs, you must define artificial variables within NightProbe with user-specified names and offsets, and types selected from an optional symbol file or from a list of standard types.

Configuration files can be created and saved to retain variable selections and display lay-out, allowing for fast start-up on subsequent invocations of NightProbe.

Using NightProbe

The primary steps in setting up NightProbe for live monitoring or recording are:

1. Define the target system.
2. Select the target programs or devices you wish to probe.
3. Optionally create artificial variables to create views into a target resource (this is usually done when monitoring something other than a program resource).

Live Monitoring

During live monitoring you can browse programs for variables, view their values, and optionally modify them.

Recording

When using NightProbe to record variables, you must take the following additional steps:

a. Specify the variables that are to be recorded

a. Specify the desired recording timer mechanism.

b. Select the recording destinations.

c. Start recording.

A data sample is taken when triggered by the recording timing source:

NightProbe supports four mechanisms which control sampling:

- on-demand sampling
- iterative sampling driven by the system clock
- iterative sampling controlled by a frequency-based scheduler
- programmed sampling driven by a NightProbe Trigger Client program

See “Recording Timing Methods” on page 12-3 for a more complete discussion of these timing sources.

NOTE

You must have appropriate file access to the target program or have privileged access in order to record, monitor, or modify the program.

In order to set the CPU bias, scheduling policy, or priority of the NightProbe server or of the program specified using the Program recording destination, you must have privileged access.

See "Privileged Access" on page F-1 for more information.

2
Invoking NightProbe

The NightProbe tool is installed on your system as **/usr/bin/nprobe**. The syntax for executing **nprobe** is described below.

To get information about NightProbe invocation options and parameters:

```
nprobe [--help]
```

To use NightProbe to record or monitor variables:

```
nprobe [--listing | -l ]
       [--listing-format=fmt | -lf fmt]
       [--playback=nprobe_data_file | -p nprobe_data_file]
       [--session=session_file | -s session_file]
       [--target=target_system | -t target_system]
       [executable_file ...]
```

Options are described as follows:

--help

This option allows you to display the usage information for **nprobe** and then exit.

--listing

This option instructs NightProbe to generate a listing of the content of a previously recorded NightProbe data file to `stdout`; then exit.

By default, the listing is generated with a single line for each sample in the data file. Each line begins with the sample number followed immediately by a colon, and then the name of each variable and its value separated by spaces. For example:

```
1: x=5 y=2.5 z=18
2: x=3 y=2.5 z=-17
```

You can specify alternative formatting using the **--listing-format** option.

The data file to be listed must be specified with the **--playback** option.

--listing-format=*fmt*

This option defines the format of the playback listing.

The *fmt* argument can be either *xml* or a quoted string containing free-text and descriptors. Free-text is copied for each sample, but descriptors are replaced with information from the sample and are of the form %*letter*, where *letter* is one of the following:

s, replaced with the sample number;
n, replaced with the variables' name;
v, replaced with the variables value;
a, replaced with the variable's address;
p, replaced with the variables process ID;
P, replaced with the variables process name.;
H, replaced with the target system's hostname;
R, replaced with the sampling rate (in seconds);
T, replaced with the date and time that recording was initiated.

Text surrounded by { and } brackets is repeated for each variable in the sample. Variable-specific descriptors (e.g. %n, %v, %a, %p, %P) can only appear inside {}-bracketed text.

Text surrounded by [and] brackets defines a header to be printed before each sample is processed and printed. The %H, %R, and %T descriptors can only appear inside []-bracketed text.

For example,

-lf *"[System: %H\nStart: %T\n]sample=%s{,%n=%v}"*

produces lines like:

```
System: ihawk
Start: Thu Apr 25 14:10:00 2009
sample=1,x=5,y=3.14159,z=8
sample=2,x=-4,y=3.14159,z=11
...
```

The characters { and [, if encountered outside of otherwise-bracketed text, are interpretted as starting the associated bracket type. If you need to include a { or [character in free-form text, without starting a bracket, use {{ or [[. Similarly, if you require a % character, use %% so that it is not considered a descriptor.

For example,

-lf *"I like [[ ] brackets; { %n=%v%%}"*

produces output like:

```
I like [ ] brackets; x=5% y=3.14159% z=8%
I like [ ] brackets; x=-4% y=3.14159% z=11%
...
```

When *xml* is specified, the listing is in XML format which includes elements describing the processes being probed, as well as elements for each sample. For example:

```
<NightProbe>
 <header start_time_string="Thu Apr 25 14:10:00 2009"
system="ihawk" start_time="1240683000"/>
 <resources>
  <resource pid="123" name="a.out" label="pid=123" >
```

```
   <variable bit_size="32" bit_offset="64" count="1"
type="5" name="x" id="1" />
   <variable bit_size="64" bit_offset="128" count="1"
type="10" name="y" id="2" />
   <variable bit_size="32" bit_offset="192" count="1"
type="5" name="z" id="3" />
  </resource>
 </resources>
 <recording>
  <sample number="1">
   <item id="1" value="1" name="x" />
   <item id="2" value="3.14159265358979323846E+00"
name="y" />
   <item id="3" value="8" name="z" />
  </sample>
...
```

The **--listing-format** option automatically implies the **--listing** option, but an explicit **--playback** option is required to specify the data file to be listed.

NOTE

You can use the following **--listing-format** parameter to generate a listing essentially identical to the legacy NightProbe listing output, although it is not very suitable for parsing:

```
"\nSample %s\n\n{pid %p @ %a : %n %v\n}"
```

--playback=*playback_file*

This option loads the NightProbe data from specified *playback_file* for viewing.

--session=*session_file*

The option configures NightProbe based on the specified *session_file*, including all programs, selected variables, viewing pages, and recording settings.

--target=*target_system*

NightProbe immediately connects to the specified *target_system*.

executable_file

This argument allows you to specify the name of an executable file. It is automatically added to the list of target programs so that you may immediately begin viewing variables associated with an executing process matching the *executable_file* name or browse the file for variables.

You can specify multiple executable files on the command line.

You may invoke **nprobe** without specifying any options or arguments.

NOTE

nprobe requires that your DISPLAY environment variable is set appropriately.

Getting Help

In addition to the *NightProbe User's Guide*, there are several sources of information on the operation of NightProbe. These include:

- the Help menu on the menu bar of the NightProbe window
- the Help button on the NightProbe dialogs
- the **--help** command line option
- the **nprobe(1)** system manual page
- the On Context menu option from the Help menu
- pressing the F1 key when a widget or window area has focus

3
Selection Dialogs

System Selection Dialog

The System Selection dialog allows you to specify the target system to be probed and the runtime attributes of the NightProbe server that will execute there.

When changing the target system selection, all programs and variables associated with the current configuration are discarded.

Once a system is selected and authentication is successful, all subsequent selections of programs and target-related items are associated with the newly selected system.

The dialog can be launched by any of the following means:

1. Using the Ctrl+T keyboard shortcut.
2. Using the Select System… option from the Target menu of the main window or the context menu of the Target item in the Configuration page.
3. Double-clicking the Target item in the Configuration page.

Figure 3-1. System Selection Dialog

The dialog consists of two areas and control buttons.

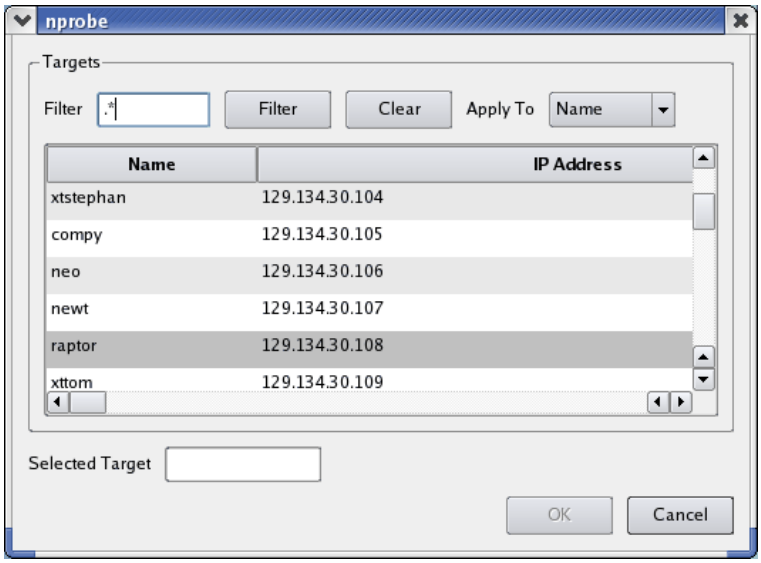
System

The system area requires that you supply the name or IP address of the target system and the user ID which you wish to connect to the target system.

Target

Specify the name of the target system or its IP address.

Pressing the Select… button launches a selection dialog which presents all systems found in the local host's **`/etc/hosts`** file. The list of systems is presented in a sortable table which can also be filtered to match user-specified patterns.



User

Specify the user name with which you wish to connect to the target system. When the Login button is pressed you will be prompted for a password. The user name and password are encrypted when passed to the target system. The password is kept in NightProbe memory for the duration of the session and is never written to disk.

Runtime Settings

This area allows you to set scheduling attributes which will be applied to the Night-Probe server which will run on the specified target system.

Scheduling Class

This option list allows you to select the POSIX scheduling class for the NightProbe server process:

- Other

  This class corresponds to the SCHED_OTHER scheduling policy which provides for general process scheduling with urgency less favorable than the other two classes. Processes in this class have their priority adjusted by the operating system based on CPU usage.

- Round Robin

  This class corresponds to the SCHED_RR scheduling policy which provides real-time process scheduling using a time-slicing algorithm to share CPU resources with other SCHED_RR processes of the same priority.

- First In First Out

  This class corresponds to the SCHED_FIFO scheduling policy which provides the strictest real-time process scheduling. Processes are not time-sliced with other SCHED_RR or SCHED_FIFO processes of the same priority.

See **`sched_setscheduler(2)`** for more information on these scheduling classes.

NOTE

Selection of the Round Robin or First In First Out scheduling class or a non-default CPU bias requires privileged access. See “Privileged Access” on page F-1 for more information.

Class Attributes

Depending on the scheduling class chosen, the following attributes can be selected:

- Nice Value

  You can use this spin-box to set the initial nice value to be associated with the NightProbe server process. A nice value provides a bias to the default priority of a process, thereby affecting the effective priority. Positive values correspond to less favorable scheduling urgency. This attribute is only available when the Other class is selected. See **`nice(1)`** for more information on the effect of nice values.

- RT Priority

  You can use this spin-box to set the real-time priority of the NightProbe server process. Priority values are constrained to be between 1..99. Selecting a priority value exceeding 90 is not recommended as it may

interfere with kernel daemon processing. A priority of one is sufficient to give the process more urgency than any process in the SCHED_OTHER class. Higher priority numbers correspond to more favorable scheduling urgency.

- Time Quantum

  You can use this spin-box to set the duration of the time-slice for processes using the SCHED_RR class. This attribute is not applicable to any other scheduling class.

CPU List

NightProbe server process execution will be constrained to the CPUs listed here.

By default, the list is `all`, which means the server process can run on any CPU on the target system which isn't shielded from process execution (consult the **`shield(1)`** man page for more information on shielding).

The list can either be the word `all`, or a comma-separated list of CPU numbers or ranges of CPU numbers; for example: `0,2-3`.

To the right of the text field a description of the resultant CPU mask is shown. Some system interfaces require CPU affinity to be specified as a mask, with each bit in the mask representing a CPU. The mask is shown to remind you that the numbers you enter into the text field here are logical CPU numbers, not hexadecimal characters in a CPU mask.

If you enter something invalid into the text field, the description to the right changes to the word `invalid`, shown in red. Ultimately, syntactically-invalid CPU lists are automatically replaced with a list indicating `all`.

Program Selection Dialog

The Program Selection dialog allows you to select a program to be probed.

The dialog can be launched by any of the following means:

1. Using the Ctrl+P keyboard shortcut.
2. Using the Program… option from the Programs menu of the main window or the context menu of the Programs item in the Configuration

page.

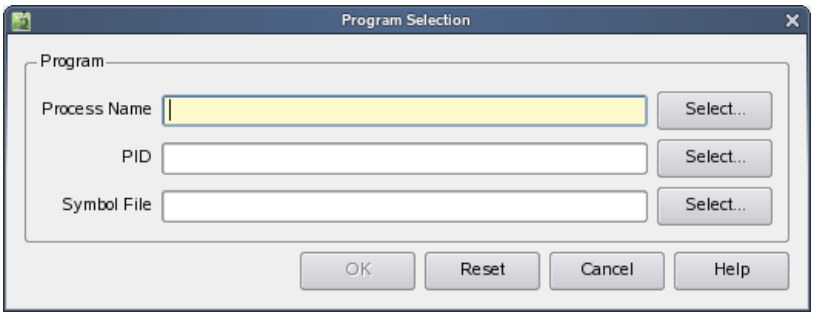


Figure 3-2. Program Selection Dialog

The dialog allows you to identify a process name and optionally the process ID and executable file name to be associated with the process within NightProbe.

Process Name

The process name is a required field. It should consist of the simple name of the process without arguments or pathnames. Often the easiest way to enter this information is to use the Select… button to the right of the Process Name field. This launches the Process Selection dialog.

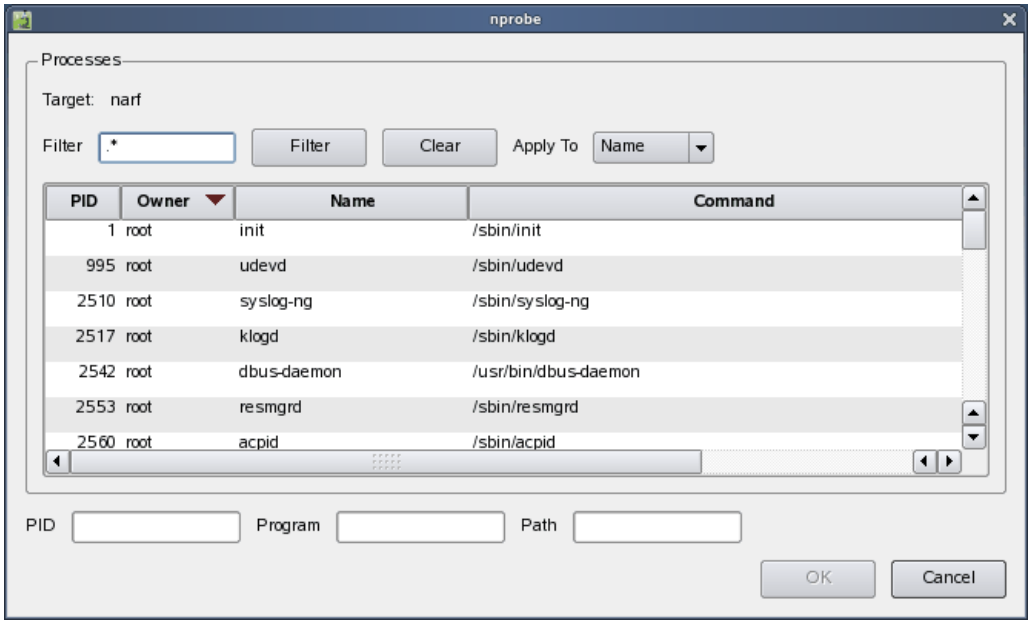


Figure 3-3. Process Selection Dialog

The Process Selection dialog presents a list of all processes currently executing on the target system.

You can click the column headers to cause the list to be sorted based on the data in the column you click on. Clicking again on the same column header reverses the order of the sort.

By selecting a process in the table, the PID, Name, and Command are automatically added to the PID, Program, and Path fields in the lower section of the dialog. When the OK button is pressed, these values will be used to fill in the Process Name, PID, and Symbol File fields of the parent Program Selection Dialog.

The dialog is designed for efficient operation. When the dialog is launched, the focus is set inside the Filter regular expression. If you type the name of the process of interest in this field and then press the Filter button, the list is pared down to any matching processes. If at least one process exists, it is automatically selected.

The keyboard shortcuts of the dialog allow you to select the process of interest, the PID, and the associated executable file with a minimum of keystrokes.

Consider the following keystrokes (shown with spaces between the keystrokes):

m y _ p r o c e s s Enter Enter

This would locate the process named my_process, select it, close the dialog, and fill in all required and optional fields in the parent Program Selection dialog.

PID

The process ID field is optional. If the process is not running you can still use NightProbe to browse for variables, but their values will not be available for live monitoring. Use of the Select… button is highly recommended as it launches the Process Selection dialog described above. Once you select a process using that dialog, NightProbe automatically fills in all three fields of the Program Selection dialog for you.

If you do not know the process ID or do not wish to specify one, leave this field blank.

Symbol File

The Symbol File field is optional, but recommended. Without a symbol file NightProbe will not be able to locate any variables within your program and all probing will be restricted to artificial variables you create within NightProbe to be associated with the process. See "Artificial Variables" on page 6-15 for more information. If you use the Select… button associated with the Process Name field, the Symbol File field is automatically filled in for you if NightProbe can determine the executable file name.

Another common use of the Symbol File field is to specify an alternative executable filename to be associated with the process within NightProbe. If the actual executable file associated with the running process has been stripped of symbols, you can specify the non-stripped version (if accessible) in this field and still be able to see variables.

Pressing the Select… button to the right of the Symbol File field launches a standard file selection dialog which allows you to specify the symbol file.

Ultimately, files specified in the Symbol File field must be accessible from the host system; they do not need to be accessible from the target system. If the program file is not accessible from the host system, NightProbe will consulte your Program File preferences, and, unless prohibited, will automatically download the file from the target to the host for processing. See "Advanced Preferences" on page -29 for more information on Program File preferences.

Shared Memory Selection Dialog

The Shared Memory Selection dialog allows you to select a shared memory segment to be probed.

The dialog can be launched by using the Shared Memory... option from the Programs menu of the main window or the context menu of the Programs item in the Configuration page.

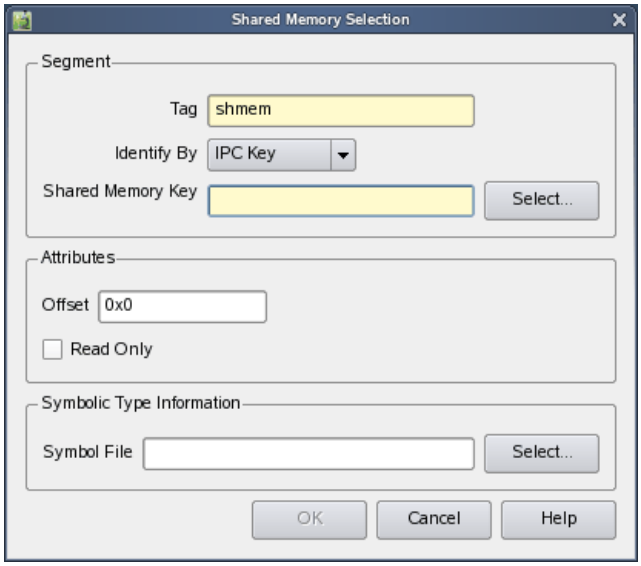


Figure 3-4. Shared Memory Selection Dialog

The dialog consists of three areas which allow you select a specific shared memory segment, set segment attributes for probing, and optionally select an executable file to be associated with the segment within NightProbe.

Segment

A shared memory segment may be identified in a variety of ways. Depending on the selection mode, different attributes are required to be specified in this dialog.

Tag

The tag is a required field. It is used merely as a handle to help identify which resource you are probing.

Identify By

The option list allows you to define the criteria by which the shared memory segment will be selected.

The information required depends on which option you choose from the option list. Pressing the Select… button will launch a selection dialog, tailored to the option you have selected, to aid you in supplying the required information.

- IPC Key

  When this option is selected, you must specify the key associated with the IPC shared memory segment. The key is the integer value passed to the **`shmget(2)`** system service from user applications; it creates or retrieves a shared memory segment and returns its ID.

  Use of the IPC Key for selection is not highly recommended because many segments are created using a private key (IPC_PRIVATE); NightProbe cannot identify such shared memory segments by their key. Use IPC ID instead.

  Pressing the Select… button launches a dialog which shows you information about all IPC shared segments that currently exist on the target system. You can select a segment in that dialog and press the OK button, causing the selected segment's key to be passed to the parent Shared Memory Selection dialog.

- IPC ID

  When this option is selected, you must specify the ID associated with the IPC shared memory segment. The ID is the value returned by the **`shmget(2)`** system call and the value passed to the **`shmat(2)`** system call from user applications which create, reference, and attach to shared memory segments.

  Pressing the Select… button launches a dialog which shows you information about all IPC shared segments that currently exist on the target system. You can select a segment in that dialog and press the OK button, causing the segment's ID to be passed to the parent Shared Memory Selection dialog.

- IPC Key File

  When this option is selected, you must specify the path to the file used to create a shared memory key using the **`ftok(3)`** service. You can also specify the value of the *`proj_id`* parameter using the supplied spin-box.

Pressing the Select… button launches a standard file selection dialog which allows you to select the key file. The files shown in that dialog are relative to the host system.

The filename specified must be accessible from the target system.

- POSIX Name

When this option is selected, you must specify the name of the shared memory segment as passed to the **`shm_open(3)`** service from user applications which create or reference named POSIX shared memory segments.

Pressing the Select… button launches a dialog which shows you information about all named POSIX shared segments that currently exist on the target system. You can select a segment in that dialog and press the OK button, causing the segment's name to be passed to the parent Shared Memory Selection dialog.

Attributes

You may specify an offset into the selected shared memory segment or mark the segment read-only to prevent subsequent user-directed modification of the segment from within NightProbe.

Offset

This field defaults to zero. When memory mappings are created to the selected shared memory segment, this offset is applied to the mapping such that offset zero within this resource within NightProbe corresponds to the offset within the actual shared memory segment as specified here. Offset values must be non-negative and cannot exceed the actual size of the shared memory segment.

This value may be specified as a hexadecimal, octal, or decimal integer value using standard C syntax.

Read Only

If the Read Only box is checked, NightProbe will create a read-only mapping to the shared memory segment. In this case, subsequent attempts to modify the contents of the shared memory segment from within NightProbe will fail.

Symbol File

The Symbol File field is optional. Without a symbol file all probing will be restricted to artificial variables with simple types you create within NightProbe to be associated with the shared memory segment. See "Artificial Variables" on page 6-15 for more information. If you have an executable file which contains types that describe the layout of the shared memory segment, you can use those types when creating artificial variables if you specify the file here.

Pressing the Select… button to the right of the Symbol File field launches a standard file selection dialog which allows you to specify the symbol file. Files shown in that dialog are accessible from the host system.

Files specified in the Symbol File field must be accessible from the host system; they do not need to be accessible from the target system.

Mapped Memory Selection Dialog

The Mapped Memory Selection dialog allows you to select files or devices which will be memory mapped within NightProbe.

The dialog can be launched by using the Mapped Memory… option from the Programs menu of the main window or the context menu of the Programs item in the Configuration page.

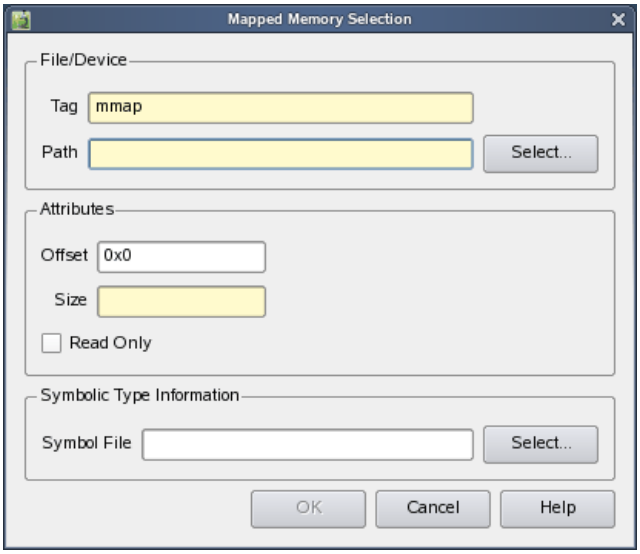


Figure 3-5. Mapped Memory Selection dialog

The dialog consists of three areas which allow you select a specific file or device, set attributes for probing, and optionally select an executable file to be associated with the mapped memory within NightProbe.

File/Device

Tag

The tag is a required field. It is used merely as a handle to help identify which resource you are probing.

Path

The path is a required field. You can specify any file path that can be accessed via the **`mmap(2)`** system service.

Pressing the Select... button will launch a standard file selection dialog which you can use to locate the file or device of interest. Files shown in this dialog are accessible to the host system.

The file specified in the Path field must be accessible from the target system; it does not need to be accessible from the host system.

Attributes

You must specify the size of the memory mapping to be made, and you may specify an offset into the selected file/device or mark the file/device read-only to prevent subsequent user-directed modification of the segment from within NightProbe.

Offset

This field defaults to zero. When memory mappings are created to the selected file/device, this offset is applied to the mapping such that offset zero within this resource within NightProbe corresponds to the offset within the actual file/device as specified here. Offset values must be non-negative and their value combined with the Size value cannot exceed the actual size of the file/device.

This value may be specified as a hexadecimal, octal, or decimal integer value using standard C syntax.

Size

The size field is required. It defines the amount of the file/device to be memory mapped.

This value may be specified as a hexadecimal, octal, or decimal integer value using standard C syntax.

Read Only

If the Read Only box is checked, NightProbe will create a read-only mapping to the file/device. In this case, subsequent attempts to modify the contents of the file/device from within NightProbe will fail.

Symbol File

The symbol file field is optional. Without a symbol file all probing will be restricted to artificial variables with simple types you create within NightProbe to be associated with the file/device. See “Artificial Variables” on page 6-15 for more information. If you have an executable file which contains types that describe the layout of the file/device, you can use those types when creating artificial variables if you specify the file here.

Pressing the Select… button to the right of the Symbol File field launches a standard file selection dialog which allows you to specify the symbol file. Files shown in that dialog are accessible from the host system.

Files specified in the Symbol File field must be accessible from the host system; they do not need to be accessible from the target system.

PCI Device Selection Dialog

This option launches the PCI device selection dialog which allows you to select a memory region from a PCI device.

PCI device probing depends on the presence of the PCI BAR File System capability in the underlying operating system (**`pci_scan_open(3)`**) which may not be available on all systems. See "Kernel Dependencies" on page B-1 for more information.

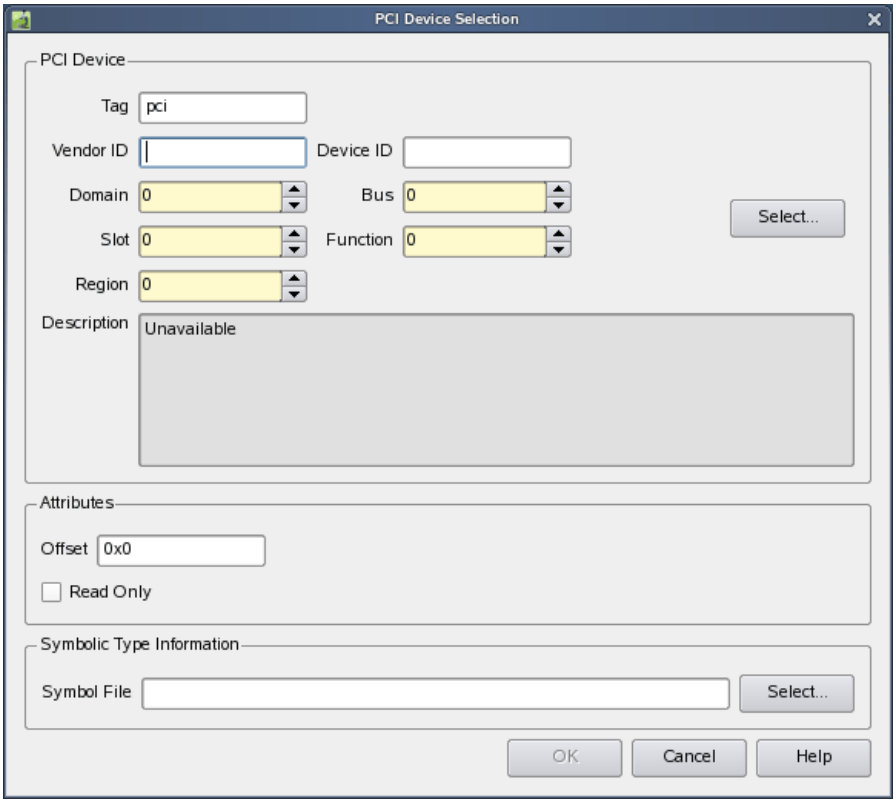


Figure 3-6. PCI Device Selection Dialog

This dialog consists of three areas which allow you to specify the PCI device of interest, set attributes for probing, and optionally select an executable file to be associated with the PCI device within NightProbe

PCI Device

Use of the Select… button is highly recommended. It launches a dialog which presents all PCI devices on the target system for easy browsing and selection.

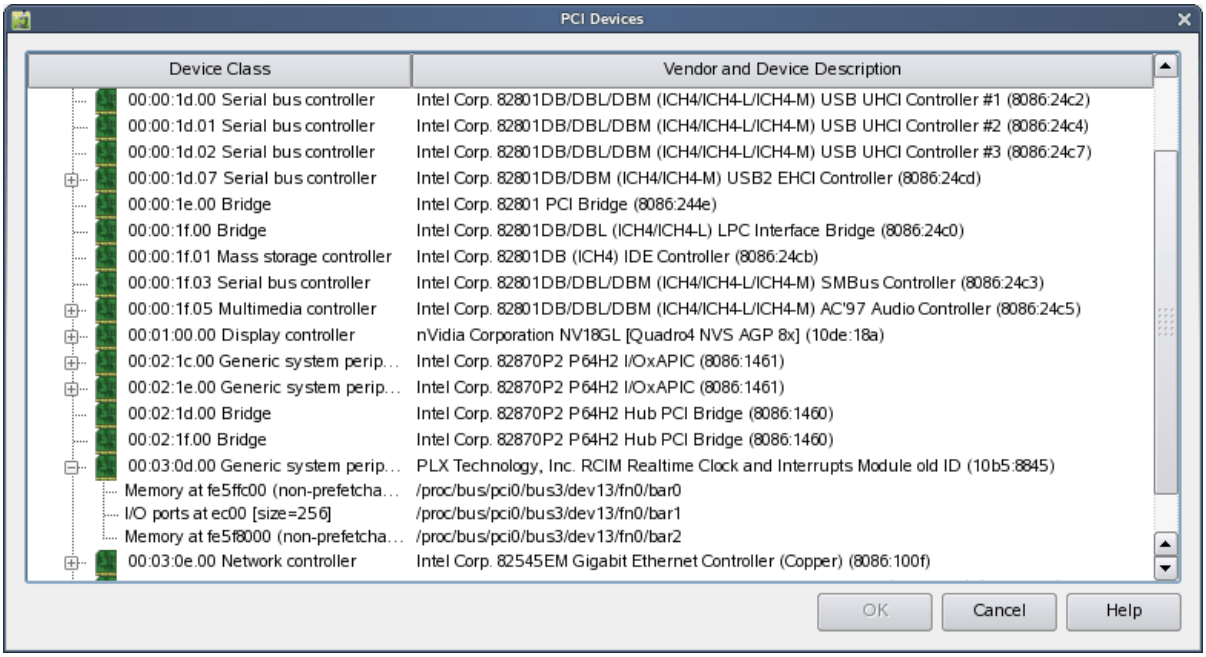


Figure 3-7. PCI Scan Dialog

If you select a memory region in one of the displayed devices and press the OK button, all the required information (as described below) is passed to the parent PCI Device Selection dialog, including the optional Vendor ID and Device ID.

Tag

The tag is a required field. It is used merely as a handle to help identify which resource you are probing.

Vendor ID

The field is optional. You can specify this field to help ensure that the remaining fields that are required accurately identify the PCI device of interest. NightProbe uses the Vendor ID only to provide the description in the Description field. If the Vendor ID you supply here does not match the Vendor ID NightProbe finds by looking up PCI device information using the required fields, the description will remain “Unavailable”.

Device ID

The field is optional. You can specify this field to help ensure that the remaining fields that are required accurately identify the PCI device of interest. NightProbe uses the Device ID only to provide the description in the

Description field. If the Device ID you supply here does not match the Device ID NightProbe finds by looking up PCI device information using the required fields, the description will remain "Unavailable".

Domain

The domain field is required. It specifies the domain number of the device. This number corresponds to the **/proc/bus/pci***DOMAIN* directory; e.g. **/proc/bus/pci0** refers to domain zero.

Bus

The bus field is required. It specifies the bus number of the device. This number corresponds to the files under the domain directory in the **/proc** file system; e.g. **/proc/bus/pci0/bus3** references devices under Domain 0, Bus 3.

Slot

The slot field is required. It specifies the slot within the bus of the device. This number corresponds to the files under the bus directory in the **/proc** file system; e.g. **/proc/bus/pci0/bus3/dev13** references the device under Domain 0, Bus 3, Slot 13.

Function

The function field is required. It specifies the function of the device. This number corresponds to the files under the slot directory in the **/proc** file system; e.g. **/proc/bus/pci0/bus3/dev13/fn0** references the device Domain 0, Bus 3, Slot 13, Function 0.

Region

The region field is required. It specifies the region within the device to be probed. Only memory regions can be probed; I/O port regions are not supported.

When the required fields are specified, NightProbe attempts to identify a PCI device region for the target system with the specified values. If it is able to locate a device with all the specified values (including the optional Device ID and Vendor ID values), a description of the device is shown in the Description area -- otherwise the area will display "Unavailable".

Attributes

You may specify an offset into the selected PCI device memory region or mark the device read-only to prevent subsequent user-directed modification of the device from within NightProbe.

Offset

This field defaults to zero. When memory mappings are created to the selected PCI device, this offset is applied to the mapping such that offset zero within this resource within NightProbe corresponds to the offset within the actual PCI device memory region as specified here. Offset values must be non-negative and their value cannot exceed the actual size of the selected PCI device memory region.

This value may be specified as a hexadecimal, octal, or decimal integer value using standard C syntax.

Read Only

If the Read Only box is checked, NightProbe will create a read-only mapping to the PCI device. In this case, subsequent attempts to modify the contents of the PCI device from within NightProbe will fail.

Symbol File

The symbol file field is optional. Without a symbol file all probing will be restricted to artificial variables with simple types you create within NightProbe to be associated with the PCI device. See “Artificial Variables” on page 6-15 for more information. If you have an executable file which contains types that describe the layout of the PCI device, you can use those types when creating artificial variables if you specify the file here.

Pressing the Select… button to the right of the Symbol File field launches a standard file selection dialog which allows you to specify the symbol file. Files shown in that dialog are accessible from the host system.

Files specified in the Symbol File field must be accessible from the host system; they do not need to be accessible from the target system.

4
NightProbe Main Window

The NightProbe main window is the primary interface to NightProbe. From this window you can perform live monitoring of variables and configure and control the data recording process.

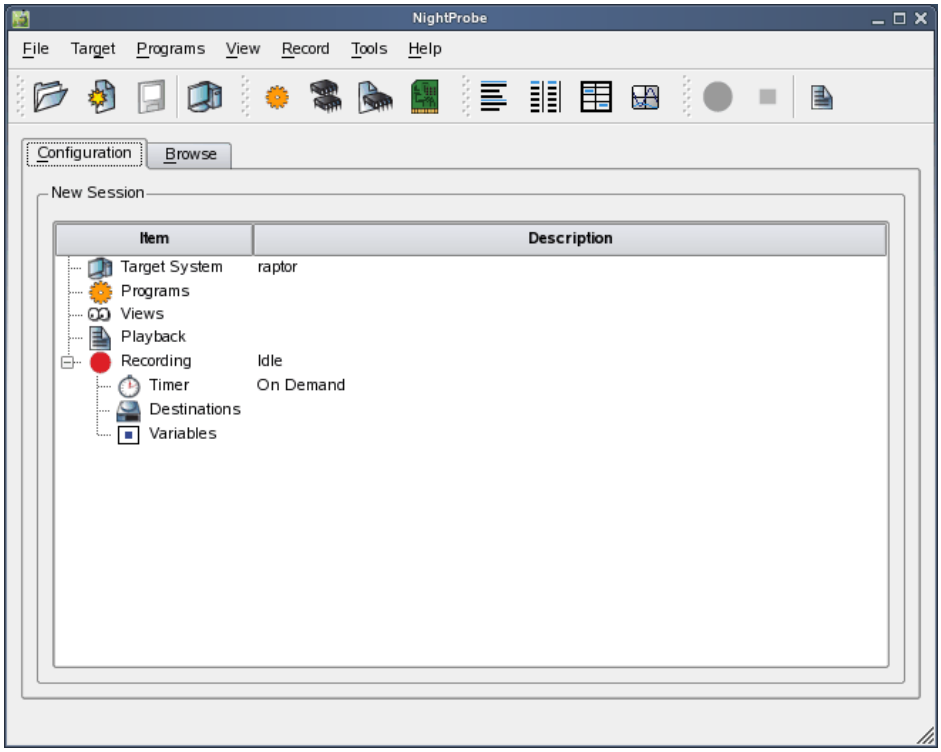


Figure 4-1. NightProbe main window

The NightProbe main window consists of the following components:

- Menu Bar (see “Menu Bar” on page 4-2)
- Toolbars (see “Toolbars” on page 4-15)
- Pages (see “Pages” on page 4-18)

Menu Bar

The menu bar provides access to session configuration services, additional tools, and help. The activities provided in the contex tmenus in the Configuration page are available from the menu bar as well. The menu bar provides the following menus:

- File (see "File" on page 4-2)
- Target (see "Target" on page 4-4)
- Programs (see "Programs" on page 4-4)
- View (see "View" on page 4-6)
- Record (see "Record" on page 4-8)
- Tools (see "Tools" on page 4-12)
- Help (see "Help" on page 4-14)

Each menu is described in the sections that follow.

File

Mnemonic: Alt+F

The File menu allows you to load a session, save the current session to a file, or create a new session. The File menu also contains the means to exit NightProbe.

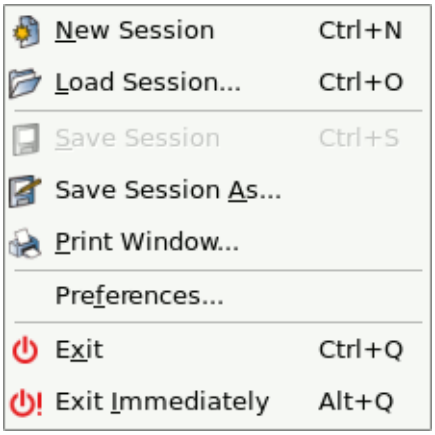


Figure 4-2. File menu

The following paragraphs describe the options on the File menu in more detail.

New Session

Mnemonic: N
Accelerator: Ctrl+N

This option allows you to clear all information from the current session and reset the various areas to blank or default values. If the window contains unsaved changes, NightProbe displays a warning dialog. You may save the changes, clear the window without saving the changes, cancel the operation, or display help related to the dialog.

Load Session...

Mnemonic: L
Accelerator: Ctrl+O

This option allows you to open a session file that you have previously saved and load all the items into the current session.

If the window contains unsaved changes, NightProbe displays a warning dialog. You may proceed to open the session, thereby discarding any unsaved changes, or cancel the operation.

Save Session

Mnemonic: S
Accelerator: Ctrl+S

This option allows you to save the configuration data from the current session in the session file that is associated with the window. If the window is not associated with a session file name, this option is the same as Save Session As.

The session file name is shown on the Configuration page as the title of the group box surrounding the configuration tree.

Save Session As...

Mnemonic: A

This option allows you to specify the name of the file in which you wish the configuration data from the current session to be saved.

Preferences...

Mnemonic: F

This option launches the Preferences Dialog which allows you to specify preferences for NightProbe, including font selection and floating-point type formatting.

Saved user preferences are applied to all NightProbe invocations for the user. Preferences are saved in the user's home directory and have a broader application than session configuration files.

See "Preferences Dialog" on page 4-19 for more information.

Exit

Mnemonic: X
Accelerator: Ctrl+Q

This option exits NightProbe. If there are unsaved changes in the current session, a dialog will ask you if you wish to save the session before exiting.

Exit Immediately

Mnemonic: I
Accelerator: Alt+Q

This option exits NightProbe immediately; if any changes to the current session are unsaved, they are discarded without warning.

Target

Mnemonic: Alt+T

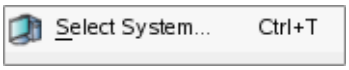


Figure 4-3. Target menu

Select System...

Mnemonic: S
Accelerator: Ctrl+T

This option launched the system selection dialog which allows you to select the target system, the user name to be used to connect to the target system, and runtime parameters for the NightProbe server process that runs on the target. See "System Selection Dialog" on page 3-1.

Programs

Mnemonic: Alt+P

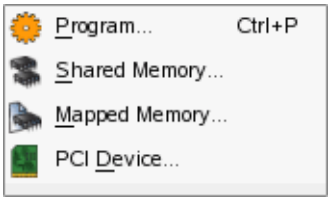


Figure 4-4. Programs menu

This menu allows you to select the various resources that you want to probe.

Program...

Mnemonic: P
Accelerator: Ctrl+P

This option launches the program selection dialog which allows you to select the process name to be probed. You can also specify the Process ID and an alternative symbolic executable file name which can be useful if the program file corresponding to the selected process has been stripped of symbolic information. See "Program Selection Dialog" on page 3-4.

Shared Memory...

Mnemonic: S

This option launches the shared memory selection dialog which allows you to select a shared memory segment to be probed. You can select an IPC shared memory segment as created by **`shmget(3p)`** or a Posix memory segment as created with **`shm_open(3)`**. You can also specify an executable file from which you can reference symbolic type information when creating artificial variables for the shared memory segment. See "Shared Memory Selection Dialog" on page 3-7.

Mapped Memory...

Mnemonic: M

This option launches the mapped memory selection dialog which allows you to select a device or file to be to be probed. You can select any file or device that the **`mmap(2)`** system service supports. You can also specify an executable file from which you can reference symbolic type information when creating artificial variables for the device or file. See "Mapped Memory Selection Dialog" on page 3-10.

PCI Device...

Mnemonic: D

This option launches the PCI device selection dialog which allows you to select a memory region from a PCI device. The memory region is selected using domain, bus, slot, function, and region attributes of the PCI device. You can also specify an executable file from which you can reference symbolic type information when creating artificial variables for the shared memory segment. See "PCI Device Selection Dialog" on page 3-12.

This menu option depends on the presence of the PCI BAR File System (**`pci_scan_open(3)`**) capability in the underlying operating system which may not be available on all systems. The menu option and corresponding toolbar icon are disabled if this feature is not available. See "Kernel Dependencies" on page B-1 for more information.

View

Mnemonic: Alt+V

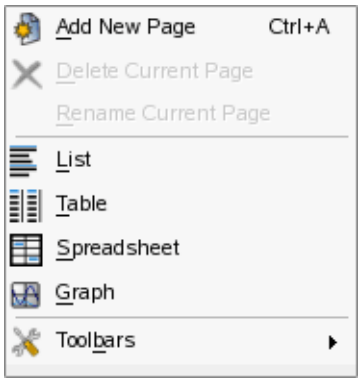


Figure 4-5. View menu

This menu allows you to add viewing panels to viewing pages and to manipulate the viewing page names and positions in the main window. It also allows you to show or hide the various toobars in the main window.

Selection of a List, Table, Spreadsheet or Graph view will cause a new panel of the selected type to be added to the current viewing page or to a new page if no viewing pages currently exist.

Add New Page

Mnemonic: A
Accelerator: Ctrl+A

This option adds a new viewing page to the right of the last page in the main window.

Delete Current Page

Mnemonic: D

This option deletes the current page and all viewing panels it contains. The current page is the page which is currently being displayed in the main window. You cannot delete the Configuration or Browse pages.

This option is also available from the context menu which appears when you right-click on a page's tab.

Rename Current Page

Mnemonic: R

This option launches a dialog that allows you to change the name of the current page. The current page is the page which is currently being displayed in the main window. You cannot rename the Configuration or Browse pages.

This option is also available from the context menu which appears when you right-click on a page's tab.

List View

Mnemonic: L

This option adds a new List panel to the current viewing page or to a new viewing page that is automatically created if no viewing pages currently exist.

A list panel shows the names and values of selected variables in a scrolling textual list providing a history of values. Each row in the list represents a variable and its value -- the variables and values are separated vertically by sample number. This panel provide live monitoring and playback monitoring of pre-recorded variables. See "List Panel" on page 8-1.

Table View

Mnemonic: T

This option adds a new Table panel to the current viewing page or to a new viewing page that is automatically created if no viewing pages currently exist.

A table panel shows the names and values of selected variables in a scrolling textual table providing a history of values. Each column in the table shows represents a single variable and each row shows the value of the variables. This panel provide live monitoring and playback monitoring of pre-recorded variables. See "Table Panel" on page 9-1.

Spreadsheet View

Mnemonic: S

This option adds a new Spreadsheet view panel to the current viewing page or to a new viewing page that is automatically created if no viewing pages currently exist.

A spreadsheet panel shows the names and values of selected variables in a grid providing a snapshot of the value of the variables. This panel provides live monitoring and monitoring of variables being recorded. See "Spreadsheet Panel" on page 10-1.

Graph View

Mnemonic: G

This option adds a new Graph panel to the current viewing page or to a new viewing page that is automatically created if no viewing pages currently exist.

A graph panel shows the values of variables over time on a two-dimensional graph. The horizontal axis represents the sample number and the vertical axis represents a variable's value. This panel provide live monitoring and playback monitoring of pre-recorded variables. See "Graph Panel" on page 11-1.

Toolbars

Mnemonic: B

Figure 4-6. Toolbars Menu

This menu allows you to hide or show individual Toolbars on the main window. You can also hide or show toolbars using the context menu that appears when you right-click a toolbar. See "Toolbars" on page 4-15.

Record

Mnemonic: Alt+R

Figure 4-7. Record menu

These menu items control playback and recording activities.

Select Playback File

Mnemonic: P

This option launches a file selection dialog which allows you to select a pre-recorded NightProbe data file for playback. Values from the playback file can be viewed with the List, Table, and Graph viewing panels.

Timer

Mnemonic: T

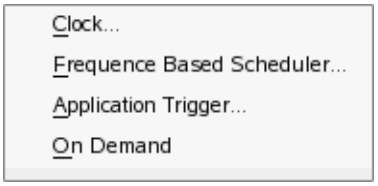


Figure 4-8. Timer Menu

This menu allows you to select the recording timing mechanism.

Clock

Mnemonic: C

This option launches the Sampling Rate dialog which allows you to define the rate at which samples will be taken. This timing mechanism provides asynchronous recording since the timing clock used is not synchronized with the application. See "Sampling Rate Dialog" on page 12-3.

Frequency Based Scheduler

Mnemonic: F

This option launches the FBS Timing dialog which allows you to identify the Frequency Based Scheduler which will control sampling. You must specify a FBS key, starting cycle, and period. This timing method provides synchronized recording if the programs being recorded are also scheduled under the same FBS. See "Frequency Based Scheduler Timing Dialog" on page 12-4.

NOTE

The Frequency Based Scheduler is not available on all operating systems. See "Kernel Dependencies" on page B-1 for more information.

Application Trigger

Mnemonic: A

This option launches the Application Trigger dialog which allows you to select the application trigger name used to control sampling. An application trigger name is a *handle* identifying a user application using the NightProbe Trigger API to control when samples are recorded. This timing method provides synchronized recording if the triggering application is the program

being recorded or is otherwise synchronized with that program. See "Application Trigger Dialog" on page 12-5.

On Demand

Mnemonic: O

This option sets the timing mechanism to On Demand. Recording samples are only taken when using the spreadsheet panel to view recorded values whenever a refresh occurs in that panel.

Recording Destinations

Mnemonic: R

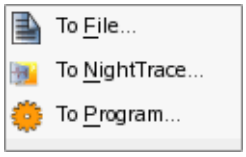


Figure 4-9. Record Destination Menu

This menu allows you to select the destinations for recorded samples.

To File

Mnemonic: F

This option launches a file selection dialog which allows you to select the name of a file to which recorded samples will be written.

The data samples can be viewed after recording using the Playback feature or using the NightProbe Datastream API. See "Playback" on page 12-15 and "NightProbe Datastream API" on page 13-1.

The specified file must be accessible or creatable from the target system. Note that the files presented in the dialog are relative to the host system.

To NightTrace

Mnemonic: N

This option launches the NightTrace Destination dialog which allows you to select variables that will be logged to NightTrace, their associated NightTrace IDs, and whether NightProbe should create a NightTrace session file including data graphs for each selected variable. This option requires that you start a NightTrace daemon to collect the recorded samples (NightProbe can optionally launch a NightTrace session to make this a trivial operation). See "NightTrace Destination Dialog" on page 12-7.

To Program

Mnemonic: P

This option launches the Program Destination dialog which allows you to select a user application to which recorded samples will be streamed. Such an application is assumed to be using the NightProbe API. The application will be executed when recording begins and its `stdin` file descriptor will be associated with the data stream. See "Program Destination Dialog" on page 12-11 and "NightProbe Datastream API" on page 13-1.

Play/Record Control

Mnemonic: C

Figure 4-10. Play/Record Control Menu

This menu allows you to start and stop recording.

Record

Mnemonic: R
Accelerator: Ctrl+Shift+R

This option starts recording. Recording is only enabled if there is at least one variable marked for recording. See "Selecting Variables for Recording" on page 12-1.

Stop

Mnemonic: S
Accelerator: Ctrl+Shift+S

This option stops recording that is already in progress. This option is only enabled if recording is active.

Tools

Mnemonic: Alt+L

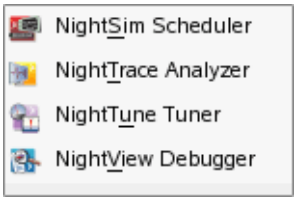


Figure 4-11. Tools menu

The following describe the options on the Tools menu:

NightSim Scheduler

Mnemonic: S

Opens the NightSim Application Scheduler. NightSim is a tool for scheduling and monitoring real-time applications which require predictable, repetitive process execution. With NightSim, application builders can control and dynamically adjust the periodic execution of multiple coordinated processes, their priorities, and their CPU assignments.

NOTE

NightSim is not available on some systems. NightSim depends on the Frequency Based Scheduler. See "Kernel Dependencies" on page B-1 for more information.

NightTrace Analysis

Mnemonic: T

Opens the NightTrace Analyzer. The NightTrace Analyzer is a graphical tool for analyzing the dynamic behavior of multi-process and/or multiprocessor user applications and operating system activity. NightTrace allows you to control user and kernel trace collection daemons and can graphically display the interplay between many real-time programs and processes across multiple processors and systems.

NightTune Tuner

Mnemonic: U

Opens the NightTune Tuner. NightTune is a graphical tool for analyzing the status of the system in terms of processes, interrupts, context switches, interrupt CPU affinity, processor shielding and hyperthreading control as well as network and disk

activity. NightTune can adjust the scheduling attributes of individual or groups of processes, including priority, policy, and CPU affinity.

For systems that support CPU shielding, NightTune provides a handy interface for controlling shielding, including downing sibling hyperthreaded CPUs to avoid interference.

NightView Debugger

Mnemonic: V

Opens the NightView Source-Level Debugger. NightView is a graphical source-level debugging and monitoring tool specifically designed for real-time applications and multithreaded applications. NightView can monitor, debug, and patch multiple real-time processes running on multiple processors with minimal intrusion.

Help

Mnemonic: Alt+H

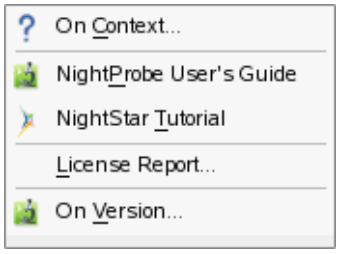


Figure 4-12. Help menu

The following describe the options on the Help menu:

On Context

Mnemonic: C

Gives context-sensitive help on the various menu options, dialogs, or other parts of the user interface.

Help for a particular item is obtained by first choosing this menu option, then clicking the mouse pointer on the object for which help is desired (the mouse pointer will become a floating question mark when the On Context menu item is selected). The cursor turns to the a circle with a backslash when the item under the cursor has no help description associated with it.

In addition, context-sensitive help may be obtained for the currently highlighted option by pressing the F1 key. NightStar's online help system, will open with the appropriate topic displayed.

NightProbe User's Guide

Mnemonic: P

Opens the online version of the *NightProbeLX User's Guide* in the online help viewer.

NightStar LX Tutorial

Mnemonic: T

Opens the online version of the *NightStar LX Tutorial* in the online help viewer.

License Report

Mnemonic: T

Opens a license dialog which indicates the current license server and the number of licenses available on the system.

On Version

Mnemonic: V

Displays a short description of the current version of NightProbe.

Check for Updates...

Mnemonic: U

Launches NUU (Network Update Utility) enabling you to update your system with the latest NightStar software. This requires network access to Concurrent's Updates web site. Updates require a login and user ID issued by Concurrent. Refer to http://redhawk.ccur.com/updates for complete information. Note that this option is not available on Ubuntu/Debian distributions, in which case synaptic should be used to check for NightStar updates.

Toolbars

NightProbe includes four toolbars which can be dragged and placed on any corner or side of the main window. These include:

- the Session Toolbar
- the Programs Toolbar
- the Views Toolbar
- the Recording Toolbar

Session Toolbar

This toolbar consists of four icons.

Open

When pressed, this icon invokes the action associated with the Load Session… option of the File menu.

New

When pressed, this icon invokes the action associated with the New Session option of the File menu.

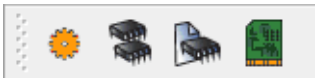
Save

When pressed, this icon invokes the action associated with the Save Session option of the File menu. This icon is disabled if no changes have been made to the current session since it was last loaded or saved.

Target

When pressed, this icon invokes the action associated with the Select System… of the Target menu.

Programs Toolbar



This toolbar consists of four icons.

Program

When pressed, this icon invokes the action associated with the Program… option of the Program menu.

Shared Memory

When pressed, this icon invokes the action associated with the Shared Memory… option of the Program menu.

Mapped Memory

When pressed, this icon invokes the action associated with the Mapped Memory… option of the Program menu.

PCI Device

When pressed, this icon invokes the action associated with the PCI Device… of the Program menu. This icon is disabled if the target system operating system kennel does not support the PCI BAR File System.

Views Toolbar

This toolbar consists of four icons.

List

When pressed, this icon invokes the action associated with the List View option of the View menu.

Table

When pressed, this icon invokes the action associated with the Table View option of the View menu.

Spreadsheet

When pressed, this icon invokes the action associated with the Spreadsheet View option of the View menu.

Graph

When pressed, this icon invokes the action associated with the Graph View of the View menu.

Recording Toolbar

This toolbar consists of three icons.

Record

When pressed, this icon invokes the action associated with the Record option of the Record menu. This icon is disabled when recording is active or when there are no items marked for recording.

Stop

When pressed, this icon invokes the action associated with the Stop option of the Record menu. This icon is disabled unless recording is active.

Playback

When pressed, this icon invokes the action associated with the Select Playback File option of the Record menu.

Pages

The remaining area of the main window is reserved for various tabbed pages which reflect and control the current configuration, provide browsing of programs, and viewing of live and recorded data.

- Configuration Page (see "Configuration Page" on page 5-1)
- Browse Page (see "Browse Page" on page 6-1)
- Viewing Panels (see "Panels" on page 7-1)

Each page has a tab which contains the page title. When clicked or right-clicked, the page is raised to the top and becomes the current page.

Each tab has a context menu which allows you to manipulate the page position and title.

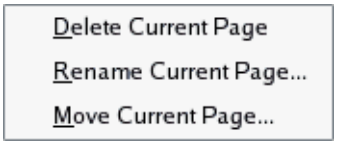


Figure 4-13. Tab Context Menu

Delete Current Page

Mnemonic: D

This option deletes the current page. The Configuration and Browse pages cannot be deleted.

Rename Current Page

Mnemonic: R

This option launches a dialog which allows you to rename the current page. The Configuration and Browse pages cannot be renamed.

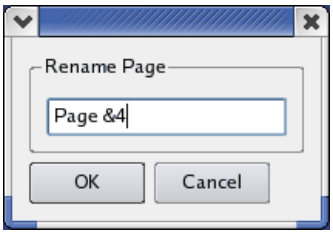


Figure 4-14. Rename Page Dialog

If the page title contains an ampersand character (&), it causes the next character to be underlined, provides a keyboard shortcut for that page, and the ampersand becomes invisible in the title that is shown for the page. In the example above, the

keyboard shortcut for this page will be Alt+4 and the displayed title will become Page 4. Activating the shortcut for a page causes it to be raised to the top and it becomes the current page. Care should be taken when choosing shortcuts for pages so they do not conflict with other shortcuts. If you desire to have an ampersand displayed in the actual page title (as opposed to defining a shortcut), use two ampersand characters, back to back in the Rename Page dialog.

Move Current Page

Mnemonic: M

This option launches a dialog which allows you to reposition the current page among other pages. This option will be disabled unless at least two viewing pages exist. The Configuration and Browse pages cannot be deleted.

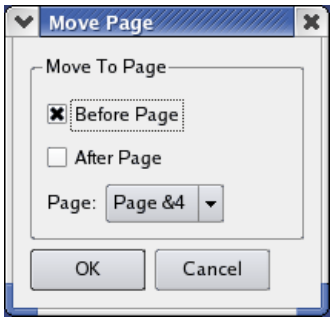


Figure 4-15. Move Page Dialog

Preferences Dialog

The Preferences Dialog is launched via the Preferences… option of the File menu.

It consists of three tabbed pages, which control General, Font and Advanced preferences.

General Preferences

This tabbed page allows you to control formatting of floating point variables and strings, as well as default refresh settings.

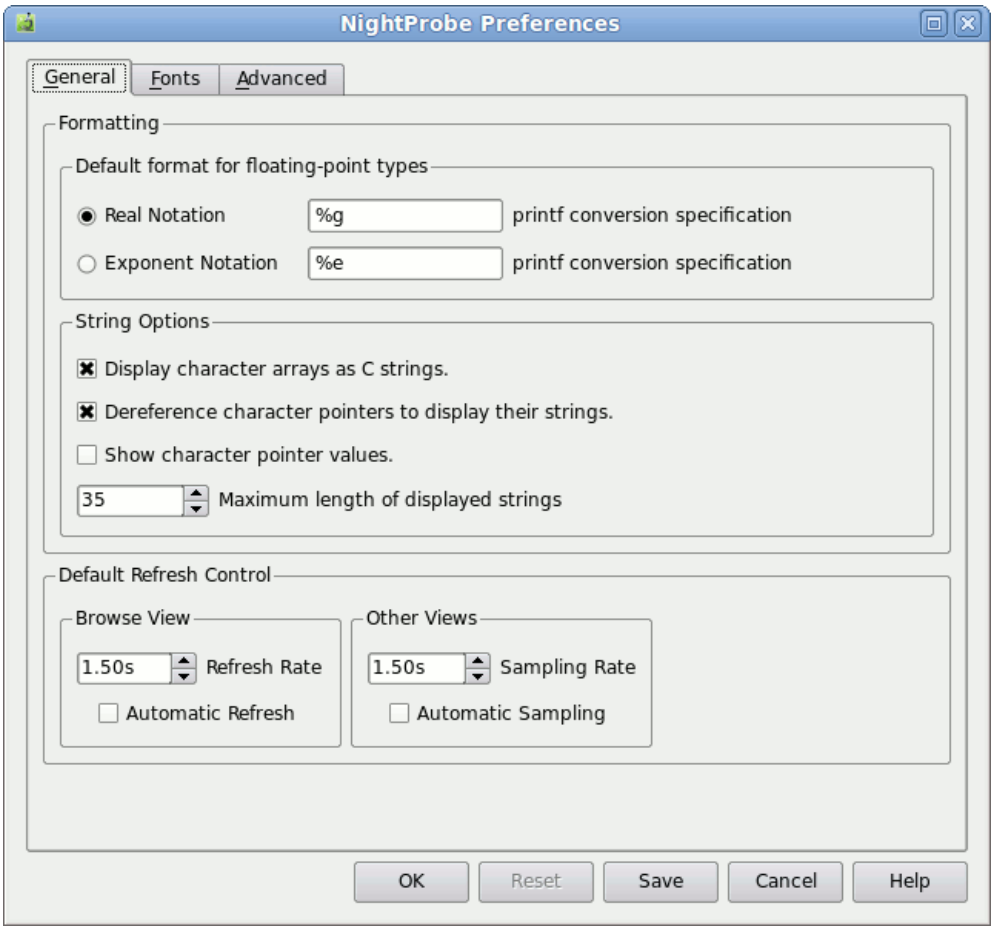


Figure 4-16. General Preferences Page

Formatting

The formatting applies to floating-point variables (and Ada fixed-point variables) displayed in the Browse panel and all other viewing panels, as well as to playback and listing operations.

Real Notation

This radio button sets the default style for floating-point variables to "real notation", which is essentially like the `printf %f` and `%g` conversion specifiers. When this button is selected, floating point values are printed without exponents. The specific format is controlled by the `printf conversion specification` shown to the right of the button.

Even if Real Notation is not selected as the default, the Real Notation `printf conversion specification` remains in effect for variables whose formatting has been specifically assigned to "real" format.

The `printf conversion specification` must be a valid specification as per **`printf(3)`**, except that the length modifier must be omitted and the conversion specifier must be one of the following: `f` or `g`.

Exponent Notation

This radio button sets the default style for floating point variables to "exponent notation", which is essentially like the `printf %e` conversion specifier. When this button is selected, floating point values are printed with exponents. The specific format is controlled by the `printf conversion specification` shown to the right of the button.

Even if Exponent Notation is not selected as the default, the Exponent Notation `printf conversion specification` remains in effect for variables whose formatting has been specifically assigned to "exponent" format.

The `printf conversion specification` must be a valid specification as per **`printf(3)`**, except that the length modifier must be omitted and the conversion specifier must be one of the following: e, E, a, or A.

String Options

These preferences control how character strings are displayed in NightProbe.

All of the preferences set here can be overridden on a per-variable basis in the Browse view, by using right-click context menu selections.

Display character arrays as C string

When selected, arrays of characters are displayed as character strings; a zero value will terminate the string even if the array has additional characters.

Regardless of whether this preference is selected, the individual characters in such arrays can be displayed in standard NightProbe tree notation in the Browse view or as individual array components (e.g. chars[2]) in other views.

The number of characters displayed in the string is curtailed by the Maximum length of displayed strings setting.

Dereference character pointers to display their strings

When selected, pointers to character types are automatically dereferenced to show the characters which they reference.

A zero character value terminates the string.

The number of characters displayed in the string is curtailed by the Maximum length of displayed strings setting.

Show character pointer values

If Dereference character pointers to display their strings is selected, then the value of the pointer itself will not be displayed unless this preference is also selected.

Maximum length of displayed strings

This limits the size of the displayed string. It is provided primarily to put an upper limit on the number of accesses to target process memory that need to be made to determine the end of the string.

Font Preferences

NightProbe uses multiple fonts to present text in the most effective manner throughout the various display areas of the tool.

Variable-width fonts are most commonly used; these fonts most closely resemble how people write or print words.

Fixed-width fonts require that all characters and numbers have the same width (visual footprint). Fixed-width fonts are of benefit when source code is being displayed or manipulated or when columns of numbers are viewed.

NightProbe further divides the use of fonts into the following categories; default and panel.

Default fonts are used for text associated with operational description and control, including: menus, buttons, selection devices, labels, tool tips, status bar messages, and generally descriptive verbiage.

Panel fonts are used in NightProbe panels, which display the data of highest importance.

Fonts are selected by querying font preferences from the following sources until a preference is found:

- Your NightProbe preference
- Your NightStar-wide preference
- The system's NightProbe preference
- The system's NightStar-wide preference
- NightProbe's ultimate default

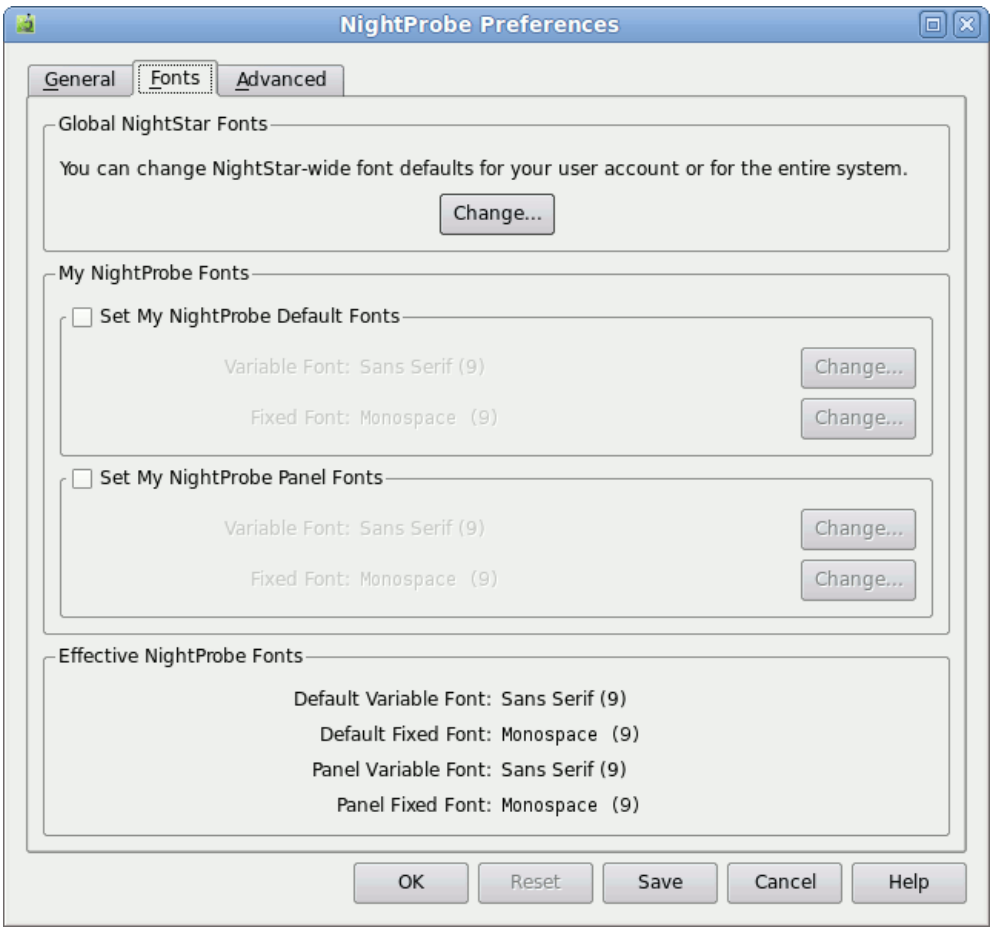


Figure 4-17. Font Preferences Page

This page is divided into three sections.

Global NightStar Fonts

The Change... button in this area launches the NightStar Global Fonts dialog which allows you to set your Nightstar-wide preferences, your preferences for another specific NightStar tool, or the system's tool or NightStar-wide preferences.

Note:

Setting a NightStar preference for the system typically requires `root` access.

Changes saved in the NightStar Global Fonts dialog are always saved to disk and apply to the current and subsequent NightProbe invocations.

My NightProbe Fonts

This area allows you to set or clear your user's preferences for NightProbe.

Selection of the checkboxes for the individual font categories control whether or not your preferences are to be consulted. Clearing a checkbox effectively removes your user preference for that category. Setting a checkbox allows you to select specific fonts within the category.

Changes to any of the settings in this area, including individual fonts or category checkboxes, are immediately reflected in the Effective NightProbe Fonts area at the bottom of the page so you can see the ultimate effect a change will have.

To change a specific font, ensure that the corresponding category's checkbox is checked and then press the Change… button. This will launch a standard font selection dialog. When you select a font from the dialog and press OK, the name of the font family is displayed to the left of the Change… button and is displayed in the selected font as well.

Effective NightProbe Fonts

This area shows you the effective fonts that will be used based on your user settings and consultation of global settings which aren't shown in the page.

The values in this area immediately change to reflect the effective font whenever any change is made within the page.

Your changes in the My NightProbe Fonts area are applied to the current invocation of NightProbe when you press the OK or Apply buttons. However, your changes are not saved to disk and will not affect subsequent invocations of NightProbe unless you press the Save button.

Separation of Apply and Save operations make it easy to experiment with fonts in the current invocation without affecting long-term usage.

Note:

Changes to font preferences in the NightStar Global Fonts dialog are always saved to disk and apply to the current and subsequent NightProbe invocations; i.e. there is no way to experiment with a global font preference without affecting subsequent NightProbe invocations.

The buttons at the bottom of the page control the application of your changes.

OK

Applies any changes made directly in the Font Preferences page to the current invocation of NightProbe and closes the dialog. The changes will not be saved to disk or affect subsequent NightProbe invocations unless you return to the Preferences… dialog and press Save.

Reset

Discards any changes you have made directly to the Font Preferences page since the dialog was launched and resets the dialog accordingly.

Note:

Changes made in the NightTrace Global Fonts dialog cannot be discarded via the Reset button.

Save

Applies the preferences from the dialog to the current invocation of NightProbe, saves the preferences to disk thereby affecting subsequent NightProbe invocations, and closes the dialog.

Cancel

Cancels any pending changes and closes the dialog.

Help

Opens the help system to display this section.

NightStar Global Fonts Dialog

The NightStar Global Fonts dialog allows you to set your Nightstar-wide preferences, your preferences for another specific NightStar tool, or the system's tool or NightStar-wide preferences.

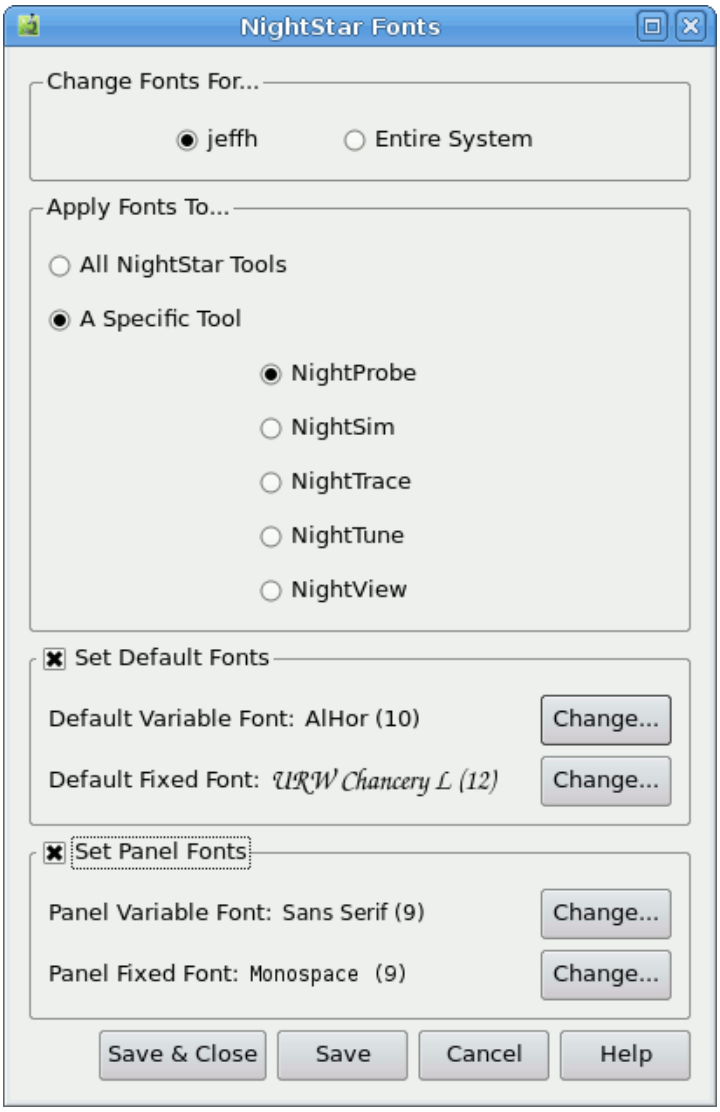


Figure 4-18. NightStar Global Fonts Dialog

Keep in mind that fonts are selected by querying font preferences from the following sources until a preference is found:

- Your NightProbe preference
- Your NightStar-wide preference
- The system's NightProbe preference
- The system's NightStar-wide preference

- NightProbe's ultimate default

This dialog has two control areas which define the scope of font preference application.

Changes Fonts For...

By default, the dialog is set up to apply font preferences to your user account. Select the Entire System button if you wish to set the system's preferences.

Note:

Changing font preference for the system typically requires `root` access.

Apply Fonts To...

This area additionally controls the scope of font preference application. You can change a preference for a specific NightStar tool or change the NightStar-wide preference.

If you wish to change the font for more than one tool from this dialog, but not change the NightStar-wide preference, select the first tool of interest, make your preference change in the areas below, and then press the Save button. Then select the second tool of interest and repeat.

Set Default Fonts
Set Panel Fonts

These areas contain the variable and fixed-width font preferences for each of the font categories, identified by the label next to each checkbox.

To remove the preferences in a category, clear its checkbox.

To change a specific font, ensure that the category's checkbox is checked and then press the Change… button. This will launch a standard font selection dialog. When you select a font from the dialog and press OK, the name of the font family is displayed to the left of the Change… button and is displayed in the selected font as well.

The buttons at the bottom of the page control the application of your changes.

Save & Close

Saves any changes made in this dialog to disk, thus affecting subsequent tool invocations, and closes the dialog.

These changes may affect the effective font preferences for the current invocation of NightProbe. When the dialog is closed, the fonts shown in the Effective NightProbe Fonts section of the Preferences dialog are updated. If you apply the changes in that dialog, they will take effect in the current invocation of NightProbe.

Save

Applies the preferences from the dialog to the current invocation of NightProbe, saves the preferences to disk thereby affecting subsequent NightProbe invocations.

These changes may affect the effective font preferences for the current invocation of NightProbe. When this dialog is subsequently closed, the fonts shown in the Effective NightProbe Fonts section of the Preferences dialog are updated. If you apply the changes in that dialog, they will take effect in the current invocation of NightProbe.

Cancel

Cancels any unsaved changes and closes the dialog.

Help

Opens the help system to display this section.

Advanced Preferences

This tabbed page provides additional preferences..

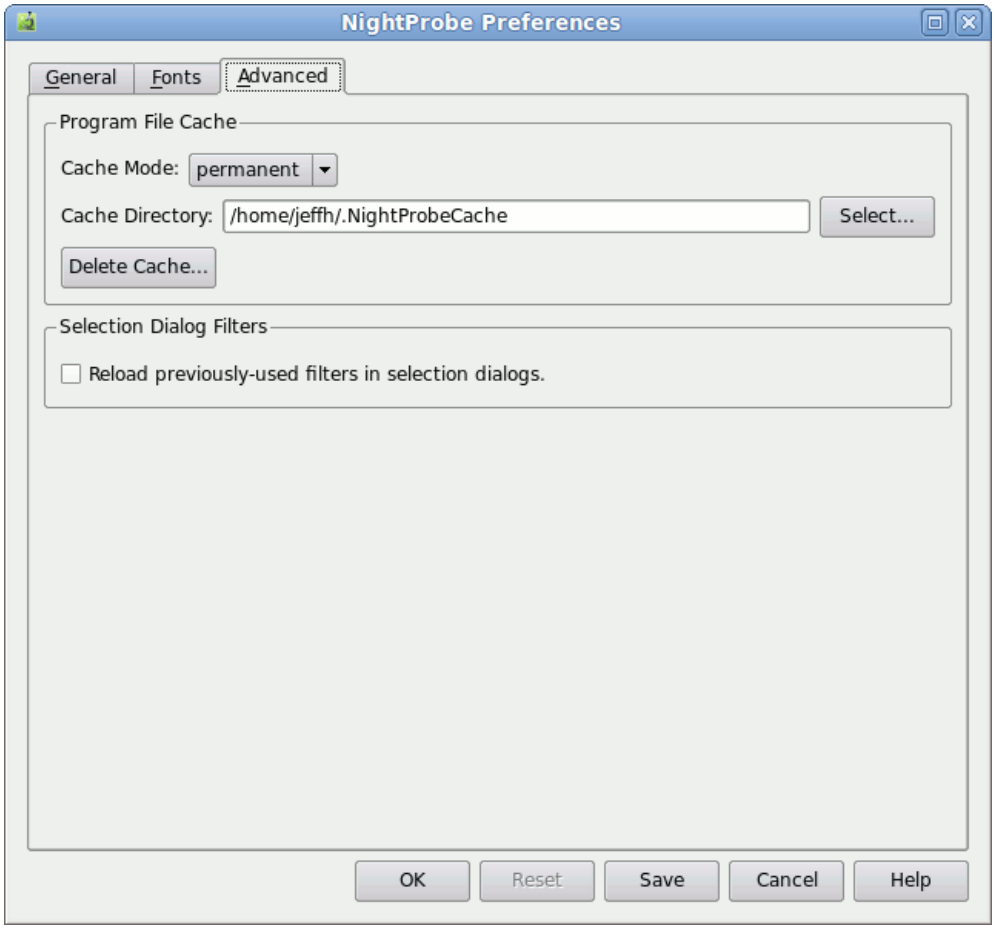


Figure 4-19. Advanced Preferences Page

Program File Cache

These preferences control NightProbe actions when probing remote target systems. NightProbe needs to open the program file corresponding to the process being probed. This needs to be done on the host system.

Cache Mode

When the mode is set to Off, NightProbe will not attempt to download the program file from the target system to the host system, nor will it consult the cache directory. You will not be able to browse the program for variables, but you can construct “artificial variables” and assign them basic formats and addresses for probing. See “Artificial Variables” on page -15 for or more information.

When the mode is set to Temporary or Permanent and the path to the program file does not exist on the host system and you are probing a remote target system, NightProbe will first consult the Cache Directory to see if the program file already exists on the host. If it exists, and the timestamp of the cached file matches the timestamp of the file on the actual target system, NightProbe will use the program file from the cache. If a cached copy does not exist or it is out of date, NightProbe will automatically download the file from the target system and place it in the Cache Directory.

File downloads are accomplished using a secure shell (**ssh**) connection which NightProbe makes for you. Depending on your **ssh** configuration, you may need to enter your user password or passphrase for the target system when this occurs. If you have appropriate **ssh** keys installed in an active **ssh** agent on the host, this will not be required.

When the mode is set to Temporary, the program file will be deleted when NightProbe exits normally; otherwise (i.e. the mode is Permanent) it will be left in the Cache Directory.

Cache Directory

Specifies a directory to store files downloaded by NightProbe from target systems.

Delete Cache...

This button removes all files from your Cache Directory.

A intermediate dialog is shown before deletion showing the exact action to be taken; the dialog requires acceptance in order to actually delete the files in the cache.

Selection Dialogs

Reload previously-used filters in selection dialogs

This option causes NightProbe to automatically reload filter settings in selection dialogs with the previous values you used earlier in the NightProbe session.

The filter text as well as the filter application option widget are reloaded, but the filter is not applied until you press the Filter button in the selection dialog.

Filters are only saved within a single NightProbe session and there is one filter for each individual selection dialog.

5
Configuration Page

The Configuration page provides a view of how NightProbe is often used and allows you to manipulate NightProbe using context menus and double-click actions.

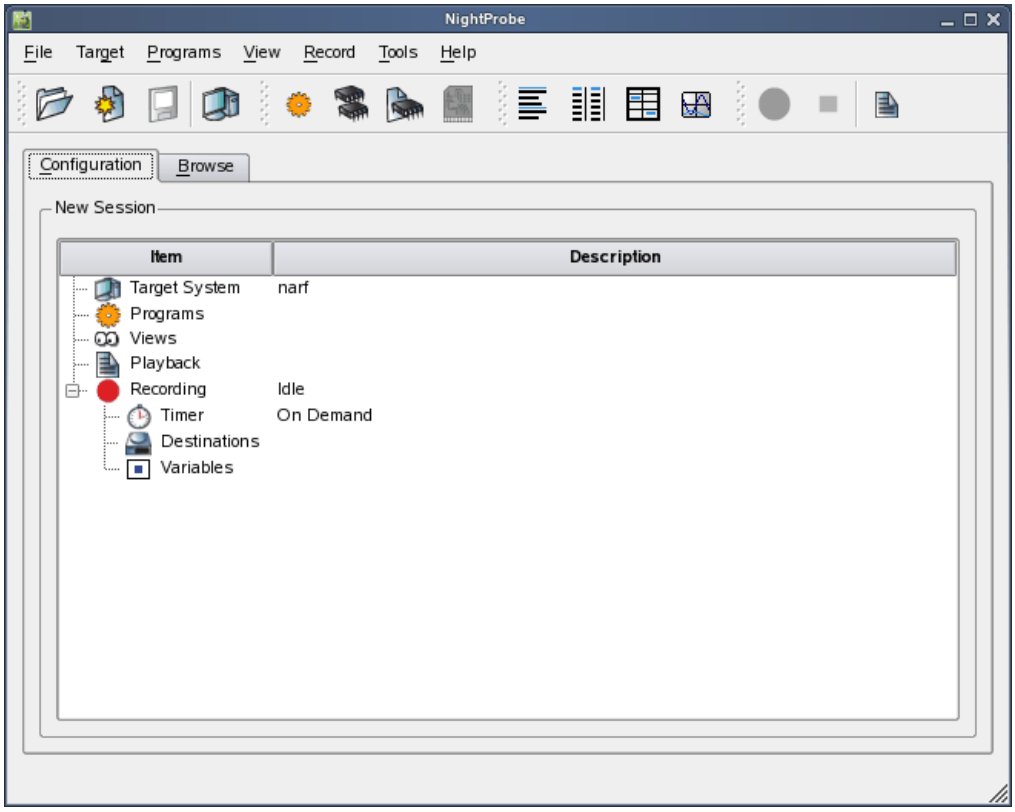


Figure 5-1. Configuration Page

The name of the current file associated with the session is displayed in the upper left hand area of the configuration page. If no named session has been loaded, the session is displayed as “New Session”.

Information is displayed in a tree with five top-level items:

- Target System
- Programs
- Views
- Playback
- Recording

Target System

The current target system is displayed in this section of the tree.

You can change the target system by double-clicking on this section of the tree or by selecting the Select System… option from the context menu.

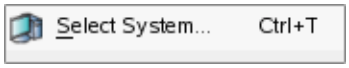


Figure 5-2. Target System Context Menu

The new target system is selected via the System Selection dialog as described in "System Selection Dialog" on page 3-1.

When changing the target system, all programs and associated variables in the current NightProbe session are discarded.

Programs

The list of all resources (programs, shared memory segments, mapped memory segments, and PCI devices) being probed is shown in this section of the tree.

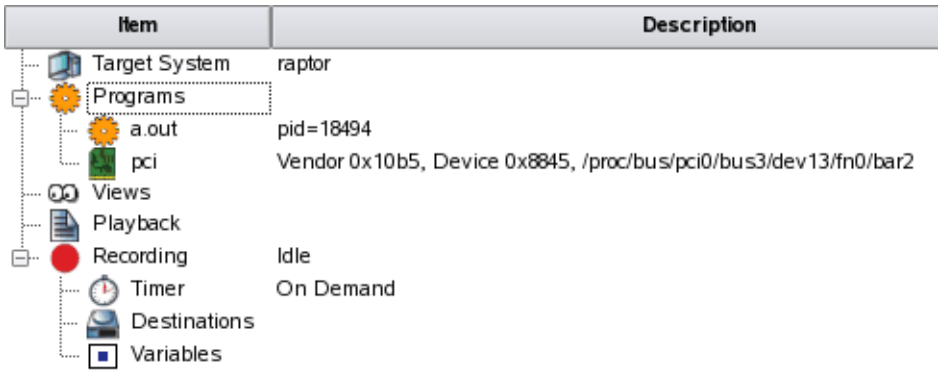


Figure 5-3. Programs tree area

You can add new resources by using the context menu launched by right-clicking the top-level Programs item in the tree:

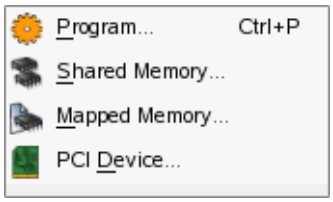


Figure 5-4. Programs Context Menu

Selecting any of these menu options causes the associated selection dialog to appear. See "Selection Dialogs" on page 3-1.

When a new resource has been added, the Browse page is immediately displayed so that you immediately browse the resource for variables and see their current values. See "Browse Page" on page 6-1 for more information.

In the Configuration page, once a resource has been added, you can change some attributes of the resource or delete it using the context menu launched when right-clicking individual resources in the tree:

Figure 5-5. Resource Context Menu

Edit

When selecting this option, or when double-clicking a specific resource item in the tree, the appropriate selection dialog will appear which will allow you to change some attributes of the resource. See "Selection Dialogs" on page 3-1.

If you change the symbol file associated with a resource, all variables associated with the resource are cleared of their Record and Mark attributes. See "Selecting Variables for Recording" on page 12-1.

Remove

If you remove a resource, all variables associated with the resource are also removed from the current NightProbe configuration.

Resources cannot be removed or edited when recording is active.

Views

The list of all views is shown in this section of the tree.

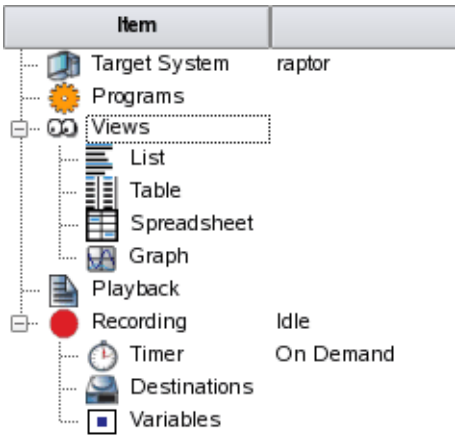


Figure 5-6. Views tree area

You can add new views by using the context menu launched by right-clicking the top-level Views item in the tree:

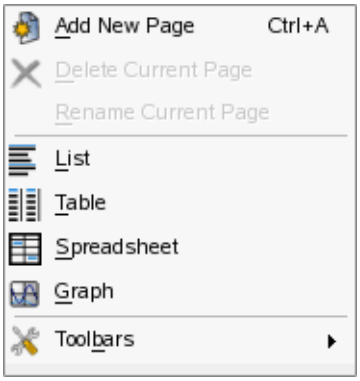


Figure 5-7. Views Context Menu

This menu option allows you to add a new page or add a view to the last viewing page if one already exists.

This menu and its options are fully explained in the description of the main window's menu bar in "View" on page 4-6.

Once a view has been added, you can show its associated page or delete it using the context menu launched by right-clicking an individual view item in the tree:

Figure 5-8. View Item Context Menu

Show

When selecting this option, or when double-clicking a specific resource item in the tree, the page containing the view is raised to the top and becomes the current page.

Delete

This option deletes the view from its page. If other views are present on the page, they will increase in size to fill the newly vacated space. If no other views are present, the page remains empty.

Playback

The current playback file, if present, is displayed in this section of the tree.

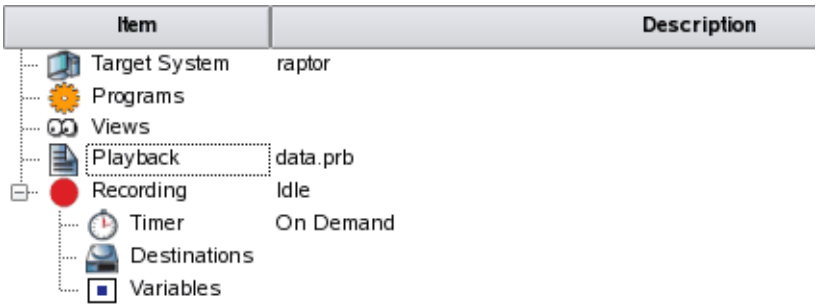


Figure 5-9. Playback tree area

A playback file is a previously recorded NightProbe data file. There can only be one playback file open at a time in a NightProbe session.

Data from playback files can be viewed in a List, Table, or Graph view.

You can change the playback file by double-clicking on this section of the tree or by selecting the Select Playback File… option from the context menu launched when

right-clicking the top-level playback item or the current playback file.

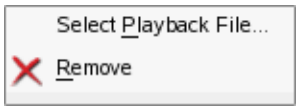


Figure 5-10. Playback Context Menu

Select Playback File

This option launches a standard file selection dialog which allows you to specify the playback file. Files shown in this dialog are accessible from the host system.

The specified file must be accessible from the host system; it does not need to be accessible from the target system.

If a playback file already exists as part of the NightProbe configuration, the newly selected file replaces it. This causes all data in any views that are displaying playback data to be flushed.

When a playback file is selected, all views displaying playback data are reloaded with the data from the selected file.

Remove

If you remove the playback file from the NightView configuration, all data in any views that are displaying playback data are emptied of data.

Selecting this option does not remove the playback from disk, merely from the current NightProbe session.

Recording

This area displays the state of recording as well as the timing source, destinations, and variables being recorded. It allows you to control recording, add items, edit, and delete them.

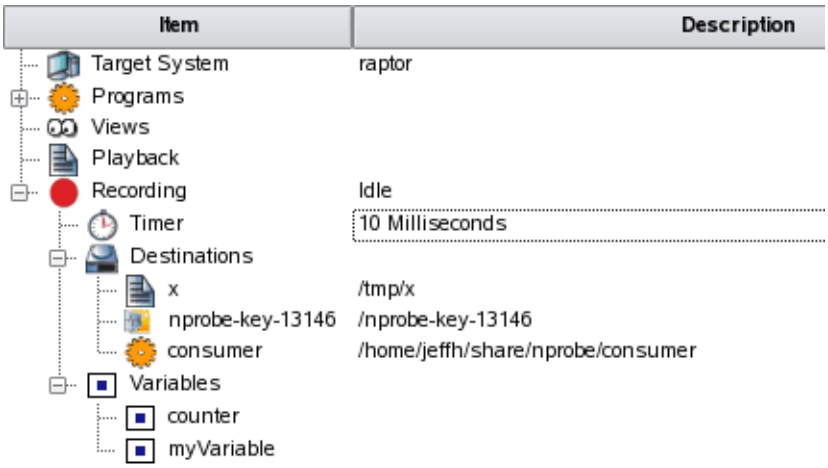


Figure 5-11. Recording tree area

The top-level item includes a description of the state of recording, either Idle or the number of samples recorded.

Recording can be started and stopped using the Record and Stop options from the Play/Record option of the Record menu in the main window or via the Record and Stop icons in the Recording toolbar.

The remaining items underneath the top-level Recording item are described below.

Timer

This item displays the recording timer which controls when samples are taken and written to the recording destinations.

The timer selection can be made using the context menu launched by right-clicking the Timer item:

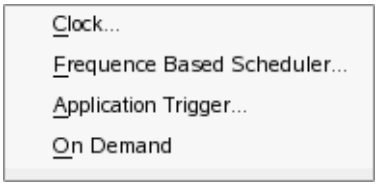


Figure 5-12. Recording Timer Context Menu

This menu and its options are discussed in detail in "Timer" on page 4-9.

Once a timer is selected, you can edit its attributes by double-clicking on the timer item itself or selecting the Edit... option of its context menu.

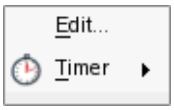


Figure 5-13. Timer Item Context Menu

You cannot change the timer selection or the attributes of the selected timer when recording is active.

Destinations

The destinations item shows all recording destinations that have been selected.

You can add additional destinations using the context menu launched by right-clicking the top-level Destinations item in the tree.

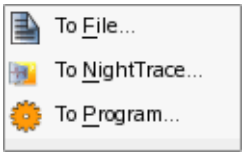


Figure 5-14. Recording Destinations Context Menu

This menu and its actions are discussed in detail in "Recording Destinations" on page 4-10.

Once a recording destination has been added, you can edit the attributes of the destination by double-clicking the specific item in the tree or by using its context menu:

Figure 5-15. Destination Item Context Menu

Edit

This option launches a dialog which allows you to change attributes of the selected destination. In the case of a File destination, a standard file selection dialog is launched which allows you to select a different filename. For the NightTrace destination, the NightTrace Destination dialog is launched as described in "NightTrace Destination Dialog" on page 12-7. For the Pro-

gram destination, the Program Destination dialog is launched as described in "Program Destination Dialog" on page 12-11.

Delete

This option causes the recording destination to be deleted.

Open file for Playback

This option is only enabled for File destinations. It changes the Playback file for the current session to this file. This option should not be used until after recording is complete, otherwise the file may not yet exist or may contain old data.

Recording destinations may not be edited or deleted while recording is active.

Variables

This item displays all variables from all programs that have been selected for recoring. See "Selecting Variables for Recording" on page 12-1.

You can browse your programs for variables in order to select them for recording by selecting the Browse... option of the Variables context menu:

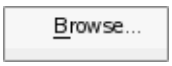


Figure 5-16. Recording Variables Context Menu

Browse

Selecting this option raises the Browse page.

You can edit artificial variables by double-clicking their item in the list or by using their context menu launched by right-clicking an item in the list:

Edit

This option is only enabled for artificial variables. It allows you to change some attributes of such variables. See "Artificial Variables" on page 6-15 for more information.

Delete

This option deselects the variable for recording. It does not remove artificial variables from their respective resources (nor does it remove real variables from their resources variable list).

Variables cannot be edited or deleted from the list of variables to record when recording is active.

6
Browse Page

The Browse page presents the list of programs being probed and the variables within them.

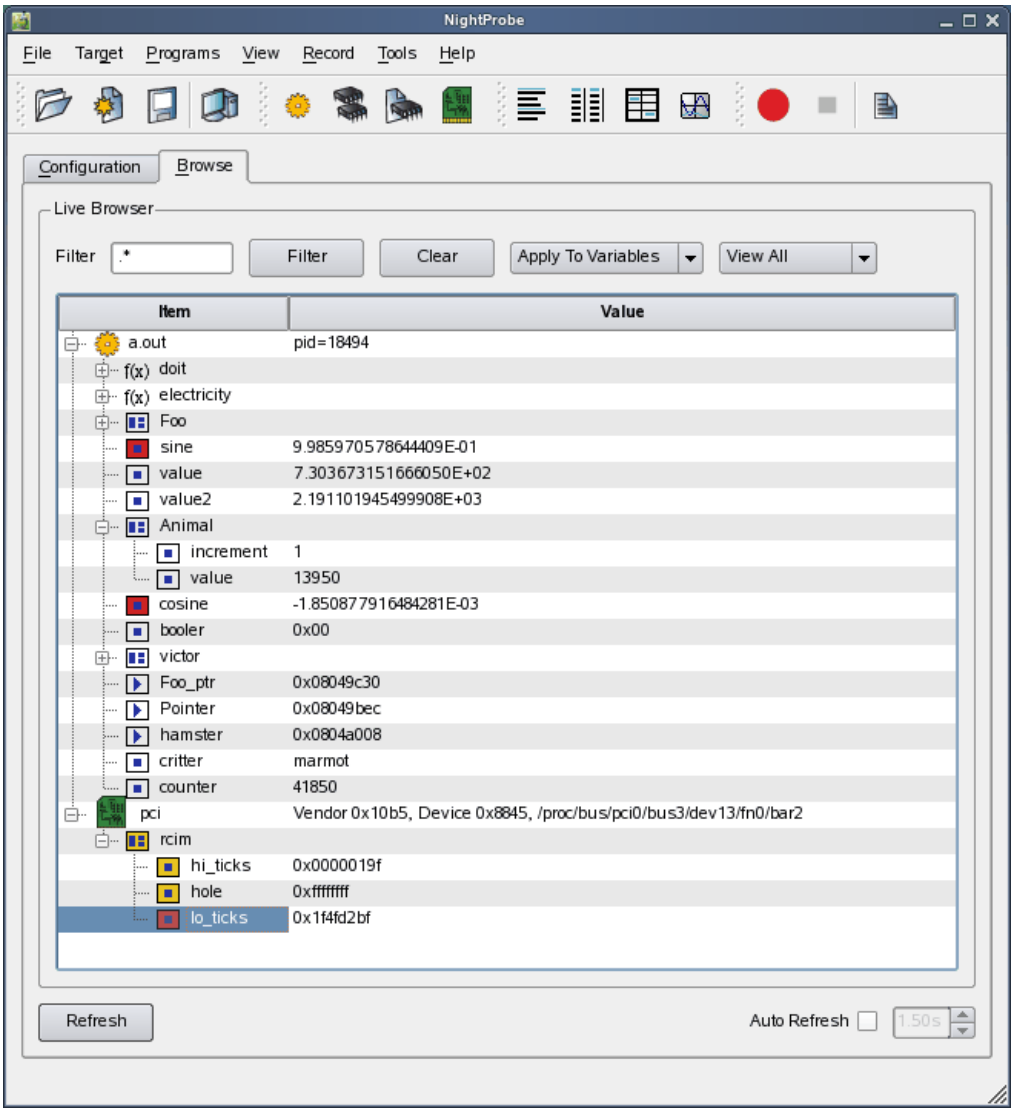


Figure 6-1. Browse Page

The top-level items in the tree are the resources being probed. Underneath each resource is a list of global variables and additional tree items which represent scopes which contain variables that can be probed.

Each variable is displayed with its current value, unless the value cannot be accessed because the program does not exist on the target system or some other error condition is present.

The page consists of three areas:

- Filter and Search Area
- Resource and Variable Tree
- Refresh Area

Filter Area

The filter area allows you to pare down the list of variables or to search for a variable based on a regular expression.

Filter Expression

The filter expression is a regular expression which will be applied to the root items or variables, as per the setting of the Apply To list.

The regular expression is not anchored by default, thus if you want to search for variables beginning with "rt_" your regular expression might be entered as "^rt_", without the quotes.

Typing Enter when inside this field activates the filter immediately.

Filter Button

Pressing this button activates the filter.

Clear Button

Pressing this button clears the filter by changing the regular expression to a string which matches everything.

Apply To List

This option list controls how the filter is applied.

- Apply To Root Items

  Selecting this option causes the filter to be applied only to the root items, which are the names of the resources begin probed.

- Apply To Scope Items

  Selecting this option causes the filter to be applied only to the scope items.

Scope items include functions containing static variables, Ada packages, or Fortran common blocks (Note: at this time, only Concurrent Fortran compilers properly describe common blocks as scope entities).

- Apply to Variables

  Selecting this option causes the filter to be applied only to variables; it does not affect which resources are shown or which components of variables are shown.

View List

This option list is orthogonal to the filtrations setting. It acts as an additional filter applied after the criteria set by the Filter regular expression and Apply To option list.

- View All

  This option shows all variables, regardless of their Mark and Record attributes.

- View Marked

  This option shows only variables with the Mark attribute set -- such variables can easily be distinguished by the yellow background of their icon.

- View Recorded

  This option shows only variables with the Record attribute set -- such variables can easily be distinguished by the red background of their icon.

To simply search for a specific variable name, leave the option lists with their default values of Apply To Variables and View All and type the name of the variable in the Filter expression then press the Enter key.

The find bar offers more functionality than the Filter Area for searching, as it expands items in the tree as necessary to locate items. See “Find Bar” on page 6-9 for more information.

Resource and Variable Tree

The top-level items of the tree represent programs and other resources that are being probed.

Expanding those items reveals the list of global variables, functions containing eligible variables, Ada packages, and common blocks.

NOTE

Ada packages and common blocks are only visible when using Concurrent's MAXAda or Fortran 77 compilers. Items within common blocks can be seen regardless of the compiler in use.

Program Items

For each resource (program, shared memory segment, mapped memory segment, or PCI device), a top-level item exists in the tree.

Expanding the item reveals scopes and variables associated with the item.

A program item's context menu provides for complete expansion or collapse of the tree as well as the ability to add artificial variables to the program or locate variables within the tree.

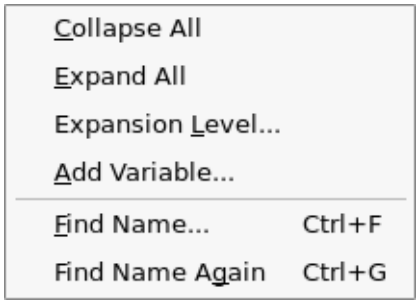


Figure 6-2. Program Items Context Menu

Collapse All

When selected, all items within the tree rooted at the program item you clicked will be collapsed. If you subsequently expand the program item by clicking its box icon, only the first level will be shown.

Expand All

When selected, the entire tree rooted at the program item you clicked will be expanded, recursively, down to the leaves of the tree.

WARNING

For complex programs this can be a time consuming operation.

Expansion Level...

When selected, you are prompted for an expansion level. If the level exceeds the current expansion, the tree is then expanded appropriately to satisfy your request.

WARNING

For complex programs this can be a time consuming operation; typically for expansion levels greater than three or four.

Add Variable...

This option causes the Artificial Variable dialog to be launched. This allows you to define a view into your resource at a specific offset with a specific type. See "Artificial Variables" on page 6-15.

Find Name...
Find Name Again

These options control the operation of the Browse page's find bar as described in "Find Bar" on page 6-9.

Selecting Find Name… activates the find bar if it is not already activated and puts the focus in the text field for editing. Selecting Find Name Again executes the previous search or simply activates the find bar if it hasn't yet been activated in the current NightProbe session.

Variable Items

For each eligible variable, you will see an entry in the tree with the variable's icon, label, and value.

NOTE

For composite variables (arrays, structures, records, etc.), their value as a whole is not displayed in that row. Expanding the variable will show the contents of the composite variable as individual components are shown.

Variable Icon

Each variable has an icon which identifies its basic type:

Scalar Variable

Composite Variable

Pointer Variable

The background color of the icon indicates whether its Mark or Record attributes have been set. Yellow indicates that the Mark attribute is set. Red indicates that the Record attribute has been set.

Mark

This attribute implies that the variable is of special interest. Such variables are available for viewing using the various View panels.

Record

This attribute causes the variable to be included in the list of variables to be recorded when recording becomes active. This attribute cannot be changed when recording is active.

To set the Mark or Record attribute, use the variable's context menu which is launched when right-clicking the variable's icon or label.

Variable Labels

Variable labels show the simple name of the variable. For components and array subscripts, they show only the component name or subscript value.

Each variable has a context menu which is launched by right-clicking the variable icon or variable label:

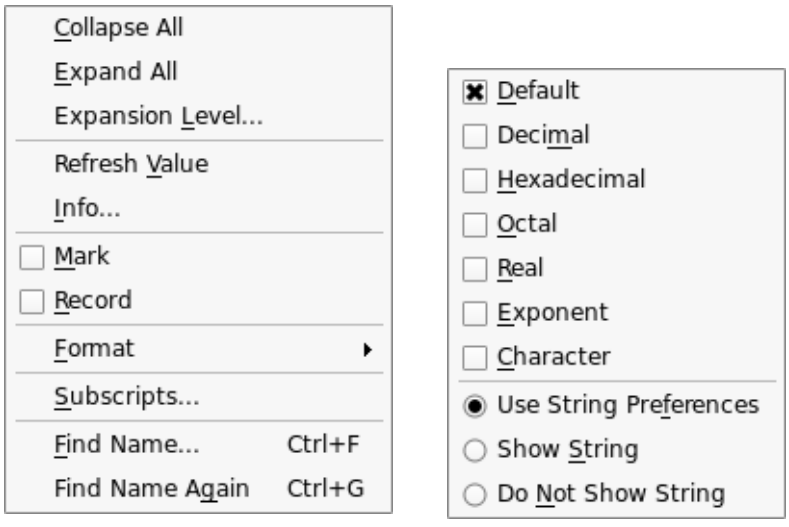


Figure 6-3. Variable Context Menu

Collapse All
Expand All
Expansion Level...

These options are only shown for composite variables or for pointers treated as arrays.

They cause the composite tree to collapse, expand, or expand to the specified expansion level.

Refresh Value

This option causes the value of the variable you clicked on to be probed from the target resource (ususually a program) and refreshed on the screen. Any children of this variable (e.g. record components) that are visible are also probed and refreshed.

See "Refresh Discussion" on page 6-14 for information on when variable values are probed and displayed.

Info...

This option launches a dialog which describes the variable in more detail.

Figure 6-4. Variable Info dialog

This information is also displayed as a tool-tip when you hover the mouse cursor over a variable item in the Browse page. This can be especially useful in order to differentiate between instances of multiply-defined variables or global variables with ambiguous names, since the Full Name in the tool-tip or Info dialog will include the source file in such circumstances.

Mark

This checkbox determines if the variable's Mark attribute is set. When set, the variable is considered of special interest and can be viewed in any of the View panels.

Marking variables also allows you to filter for such variables so that the Browse page only shows these variables. See "Filter Area" on page 6-2 for information on filtering marked variables.

Record

This checkbox determines if the variables Record attribute is set. When set, the variable is included in the list of variables to be recorded when recording is active.

Double-clicking on a variable label causes the Mark and Record attributes to be set.

This attribute may not be changed when recording is active.

Format

This option allows you to change the default format in which the variable's value is displayed.

The default format is recommended especially for enumeration items as that allows NightProbe to show you the symbolic value of each enumeration constant as opposed to its underlying integer value.

Note that the Default, Real, and Exponent formats for floating-point variables (and Ada fixed-point variables) is also affected by your user's floating-point formatting preference, which can be adjusted with the "Preferences Dialog" on page 4-19.

Formats assigned via this context menu apply persistently to the variable, throughout all viewing panels within NightProbe and to listing operations during playback. If the NightProbe session is saved, the assigned format will be persistent in future uses of the session.

NOTE

Changing the format of a variable does not apply retroactively to existing instances of the variable in the List and Table views.

Treat Pointer As Array

This option is only shown for variables which are pointers. It instructs NightProbe to treat the pointer as if it pointed to a series of consecutive elements. This is essentially the same as indexing a pointer in the C language.

When treating a pointer as an array, the value of the pointer is frozen at the time this occurs. Subsequent display of dereferenced items uses the value of the pointer when it was frozen. This is indicated by red text for the value of the pointer.

Indirect Pointer

This option is only shown for variables which are pointers. It instructs NightProbe to indirect the pointer value and show the value of what the pointer points to.

When a pointer is indirected, the value of the pointer is frozen at the time of indirection. All subsequent values associated with the indirection use the frozen pointer value. A frozen pointer value is indicated in red text in its corresponding value field.

Refreeze Pointer

This option is only shown for pointers which have been indirected or treated as arrays. When selected, the pointer value is refreshed and re-frozen, thus all indirected items may change value, shape, or size.

Subscripts...

This option is only shown for arrays, or pointers treated as arrays. It launches the Show Subscripts... dialog as described in “Show Subscripts Dialog” on page -11 and “Show Subscripts Dialog for Multi-dimensional array” on page -12.

Find Name...
Find Name Again

These options control the operation of the Browse page’s find bar as described in “Find Bar” on page 6-9.

Selecting Find Name… activates the find bar if it is not already activated and puts the focus in the text field for editing. Selecting Find Name Again executes the previous search or simply activates the find bar if it hasn’t yet been activated in the current NightProbe session.

Find Bar

A find bar, shown in Figure 6-5, appears at the bottom of the Browse page when you ask to search for a variable name. You can activate the find bar by using the Find Name… option of the Browse page’s context menu or by using the shortcut key Ctrl+F when the Browse page has focus.

Figure 6-5. Browse Page Find Bar

Enter a regular expression in the text entry field to match the name of a variable or variable component. As you type more characters, the search progresses. If no match is found, the background color of the text entry field changes to a salmon color. A label at the end of the find bar shows the status of the search.

To go to the next match, click on Next or type Ctrl+G. Click on Previous to go to the previous match. The Match case check box indicates whether the search should be case-sensitive.

Press Esc or click on the close button to hide the find bar.

The search initially progresses from the current item in the Browse page, or the root of the tree if no item has been selected. As you type characters in the text field, matched items are selected, and the page is scrolled to make them visible if necessary. If you remove characters from the text field, the display may "back up" to a previous item in the tree which now matches the inserted text. Until you press the Enter key, or the Next or Previous icons, the original item is maintained as the origin of the search.

You can control the starting point for a search by simply clicking an item in the tree.

In the Browse page, the find bar searches only for variable and variable component names, not values of variables. It will expand items in the tree as necessary in order to locate them. However, it only expands the minimal amount needed to display a located item.

In all cases, the entire tree in the Browse page is searched as required, not just the tree rooted at the current item.

Variable Values

The current value of non-composite variables is shown in this column.

The format of the value can be controlled using the context-menu of the variable's label.

To change the value of a variable, double-click the cell containing its value.

When double-clicked, the cell becomes frozen and auto-refresh of the Browse page is paused.

You can type a new value into the cell. Hitting the Enter key commits the new value to the variable.

The value should be expressed in the native format for the variable.

For character variables, character values should be entered with surrounding single-quote marks. If an integer value is entered without single-quote marks, the character variable will then be given that integer value.

For enumerated variables, you can enter either an integer or a valid enumeration constant identifier or select one of the enumeration constants from the editor's drop-down list.

See "Refresh Discussion" on page 6-14 for information on when variable values are probed and displayed.

Arrays

Array variables are composite variables. Expanding an array variable in the tree makes the individual array components visible. By default, a single array subscript is shown

along with an array extension icon.

Figure 6-6. Array Variable With Extension Icon

If the extension icon is clicked (the bluish downward pointing arrow), another subscript is added to the list of components, unless the bounds of the array prevent the extension. In this case, the Show Subscripts dialog is launched.

Figure 6-7. Show Subscripts Dialog

The dialog indicates the name of the array and the declared index constraints of the array. It allows you to select a single component, or a set of array slices -- which are ranges of components. In the figure above, we've extended the initial slice which originally had only element 0. Another single-component slice has been added as well.

If you wish to show components past either end of the declared bounds, you can clear the Prevent boundary violations checkbox.

When Prevent boundary violations is not in effect, clicking the array extension icon will add another component to the array even if doing so would exceed the declared bounds of the array.

The slice settings in the figure above would result in the following display in the Browse tree.

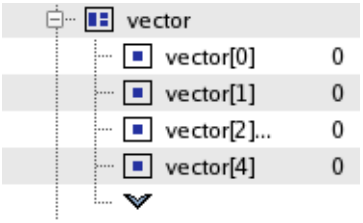


Figure 6-8. Array Expansion in Browse Tree

Notice the ellipsis that follows the index expression in the Browse tree for component `vector[2]`. The presence of the ellipsis is used to indicate that there are array components that have been omitted between the component and the next one shown in the tree

The Show Subscripts dialog is also available from the Show Subscripts... option of the variable's context menu for array variables or extension icons (not shown in figure above).

Multi-Dimensional Arrays

For multi-dimensional arrays, each slice must include information for each dimension.

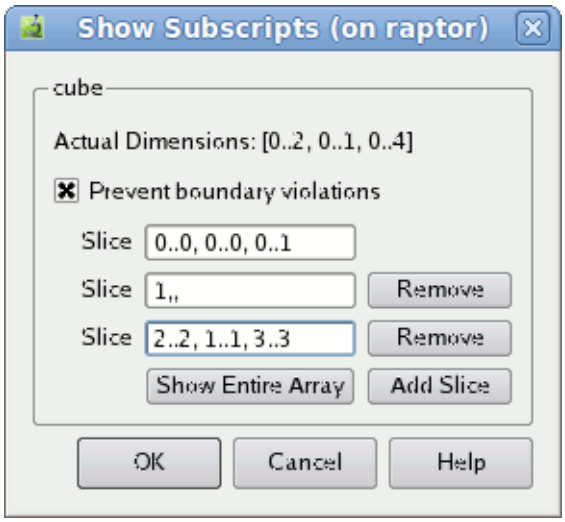


Figure 6-9. Show Subscripts Dialog for Multi-dimensional array

In the figure above, the dialog specifies three slices of a three-dimensional array.

Slices must be specified as a comma-separated list of index ranges -- one for each dimension of the array.

An index range takes one of the following forms:

```
lower_bound .. upper_bound
lower_bound
<empty>
```

When the lower bound is specified but the upper bound is omitted, the upper bound is implied to be the lower bound. Thus `1`, is equivalent to `1..1`.

If the index range is omitted, it implies the entire range of that dimension. If you omit an index range, you must still separate the omitted ranges with commas.

NOTE

A completely empty slice (one without commas or any notation) is simply ignored.

The figure below shows the array components that would be shown based on the Subscripts Dialog shown in Figure 6-9 on page -12.

Figure 6-10. Portion of Three-Dimensional Array Shown

The first slice in the Show Subscripts dialog, `[0..0,0..1,0..1],` results in the first two components of the array being shown.

The second slice in the Show Subscripts dialog, `[1,,]`, is short-hand notation for the following: `[1..1, 0..1, 0..4]`.

The third slice in the Show Subscripts dialog, `[2..2,1..1,3..3],` selects a single component; it could have been specified as simply `2,1,3` in the dialog.

For multi-dimensional arrays, you may only specify an index range that exceeds the bounds of the array if the Prevent boundary violations checkbox is cleared and the dimension you are extending is the major dimension of the array. For Fortran, column-major ordering dictates that the major dimension is the last dimension (as read from left to right). For most other languages, row-major ordering dictates that the major dimension is the first dimension.

Refresh Area

The refresh area consists of the following items as described below.

Refresh Button

Pressing the Refresh button causes the value of all variables currently viewable in the tree to be probed and refreshed with their current value.

Automatic Refresh Checkbox

If this box is checked, the values of all variables currently visible in the tree will be probed and refreshed with their current value at the rate defined by the Refresh Rate spinbox.

Refresh Rate Spinbox

This spinbox defines the rate at which automatic refreshes will occur. It is only enabled if the Automatic Refresh checkbox is checked.

Refresh Discussion

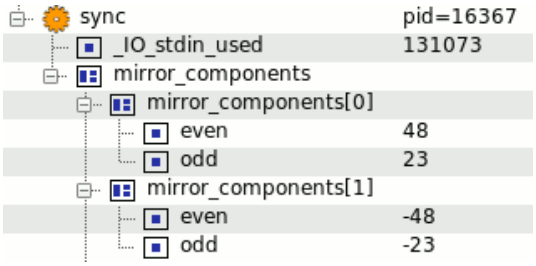
Probing of variables in the Browse page is inherently asynchronous, but the triggering of refresh requests is controlled and well-defined.

By default, the Automatic Refresh checkbox is not checked. Thus variables are only probed and displayed when you request to see them in the viewable area of the tree, or when you scroll (or search) through the tree and more become visible, or when you press the Refresh button.

One a variable's value has been displayed, it will not be probed again until a refresh request occurs (either an explicit press of the Refresh button or an expiration of the activated Automatic Refresh timer) AND either of the following occur:

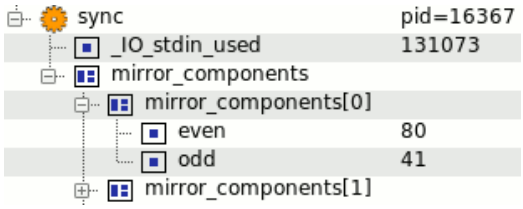
1. The variable is currently visible in the tree
2. The variable becomes visible in the tree subsequent to the refresh request (the probe and refresh for that specific variable is delayed until it becomes visible)

Keep the latter case in mind. For example, assume you are viewing two components of an array of structures as shown here:

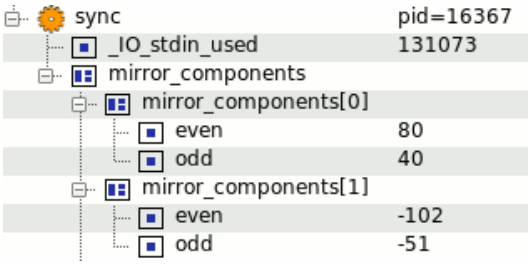


The **sync** program being probed above keeps sub-components of mirror_components[1] mirrored with the negated values from mirror_components[0], even as the program changes their values over time.

Now assume you collapsed mirror_components[1] and then pressed the Refresh button. The tree would show updated values for mirror_components[0], as shown here:



If you then wanted to view the sub-components of mirror_components[1] and do so by expanding it, those items are probed at that time (because their previous values were out-of-date due to the Refresh operation):



To get refreshed values of both at the same time, simply press the Refresh button while they are both visible.

Consider a scenario where you want to watch several variables within the program in the browser and probe their values at relatively the same time. If these variables are not closely co-located in the tree, this would be difficult. A solution here is to set the Mark attribute on each of the variables of interest, and then filter the browser to show only marked variables. Thus just the variables of interest are shown in the tree and a refresh request will probe and refresh all of them (assuming they are all visible on the screen). See “Mark” on page 6-6 and “Filter Area” on page 6-2 for more information.

This mechanism was designed for performance and minimal intrusion (otherwise, when pressing Refresh might we not need to read the entire data address space of the target process?). It affords you a reasonable level of control over when variables are probed and refreshed within the browser. For more fine-grained control over when variables are probed, or to obtained synchronized samples, use the various Views (List, Table, Spreadsheet, or Graph) independently, or in conjunction with NightProbe’s recording capability.

Artificial Variables

Artificial variables are a means of creating a view into your resource to be able to view values within it and modify the resource as well.

Typically, program resources already have variable definitions which are of interest so you don’t add artificial variables to them. However, they can still be useful if you don’t have symbolic information in your program file or you want to look at memory which isn’t associated specifically with a variable.

The Artificial Variable Definition dialog is launched from the Add Variable… option of the Program Item context menu. It is also automatically launched when a non-program resource has been added (because such resources don’t have any variables within them until you add an artificial variable).

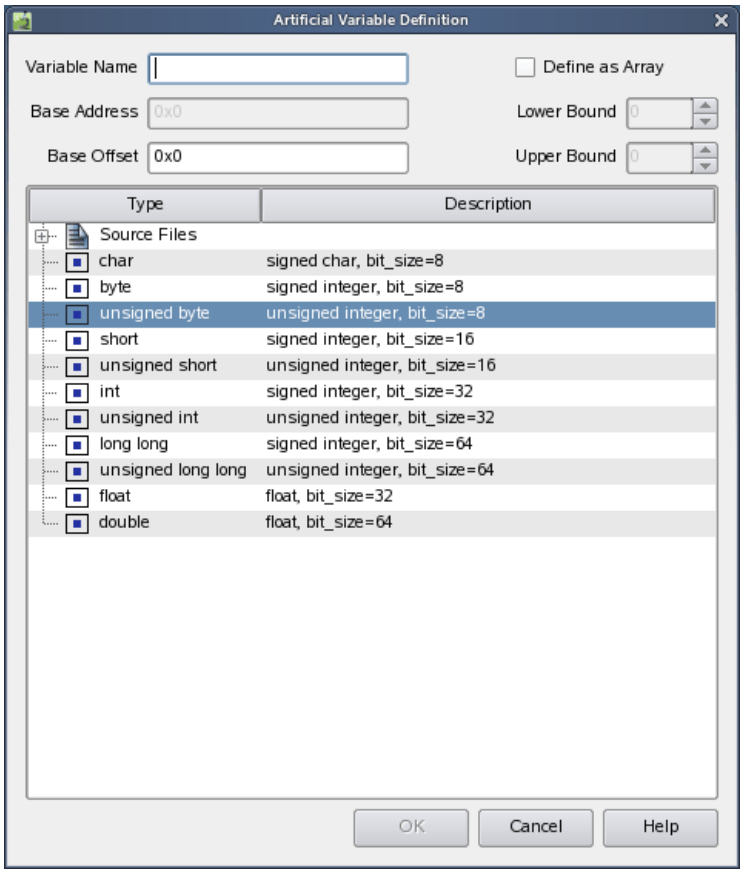


Figure 6-11. Artificial Variable Definition Dialog

The dialog consists of two areas which define the name and address/offset of the variable and the type of the variable.

Variable Name

This field is required. You must specify a name for the artificial variable.

Base Address

For program resources, this field is required. It should contain the base address of the variable which can be any valid address in the program.

Base Offset

This field is required for non-program resources. It defines the offset within the resource where the variable will be located.

Define As Array

Setting this checkbox defines the variable as an array. The Lower Bound and Upper Bound spinboxes define the bounds of the array. The type as selected in the type tree defines the type of a single component of the array.

Lower Bound

When the variable is defined as an array, this spinbox contains the lower bound of the array. Normally this would be left as zero. Whatever value is chosen here is the first index value of the array and will correspond to the location defined as the Base Offset or Base Address.

For example, if the base offset is 0x1000 and the lower bound is set to 5, the address of variable[5] is 0x1000.

Upper Bound

When the variable is defined as an array, this spinbox contains the highest bound of the array.

Type Tree

The type tree allows you to select the type for the variable.

The basic atomic types as shown in the figure above will always be available in the type tree.

Additionally, if a symbol file was associated with the resource, then any type associated with an eligible variable in that program file can be used as the type for this artificial variable.

Consider the following excerpt from a type tree:

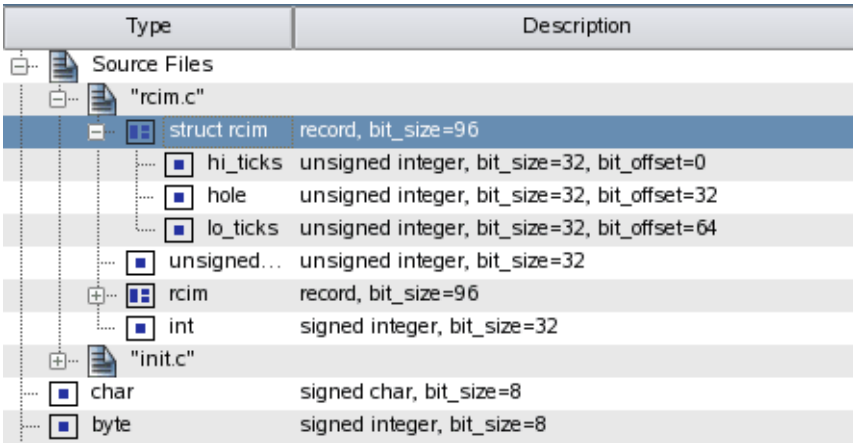


Figure 6-12. Type Tree with Symbol File

As shown above, in addition to the basic atomic types, types from the symbol file associated with the resource are included as well.

In the figure above, we have selected a structure type as the type for our artificial variable.

NOTE

Due to compiler limitations, types defined in programs may not be available in the type tree unless a variable is present in the program of that type.

7
Panels

NightProbe provides flexibility in configuring the graphical user interface to suit your needs through the use of resizable and movable panels.

Consider the following page which contains a List view and a Graph view each in their own panel:

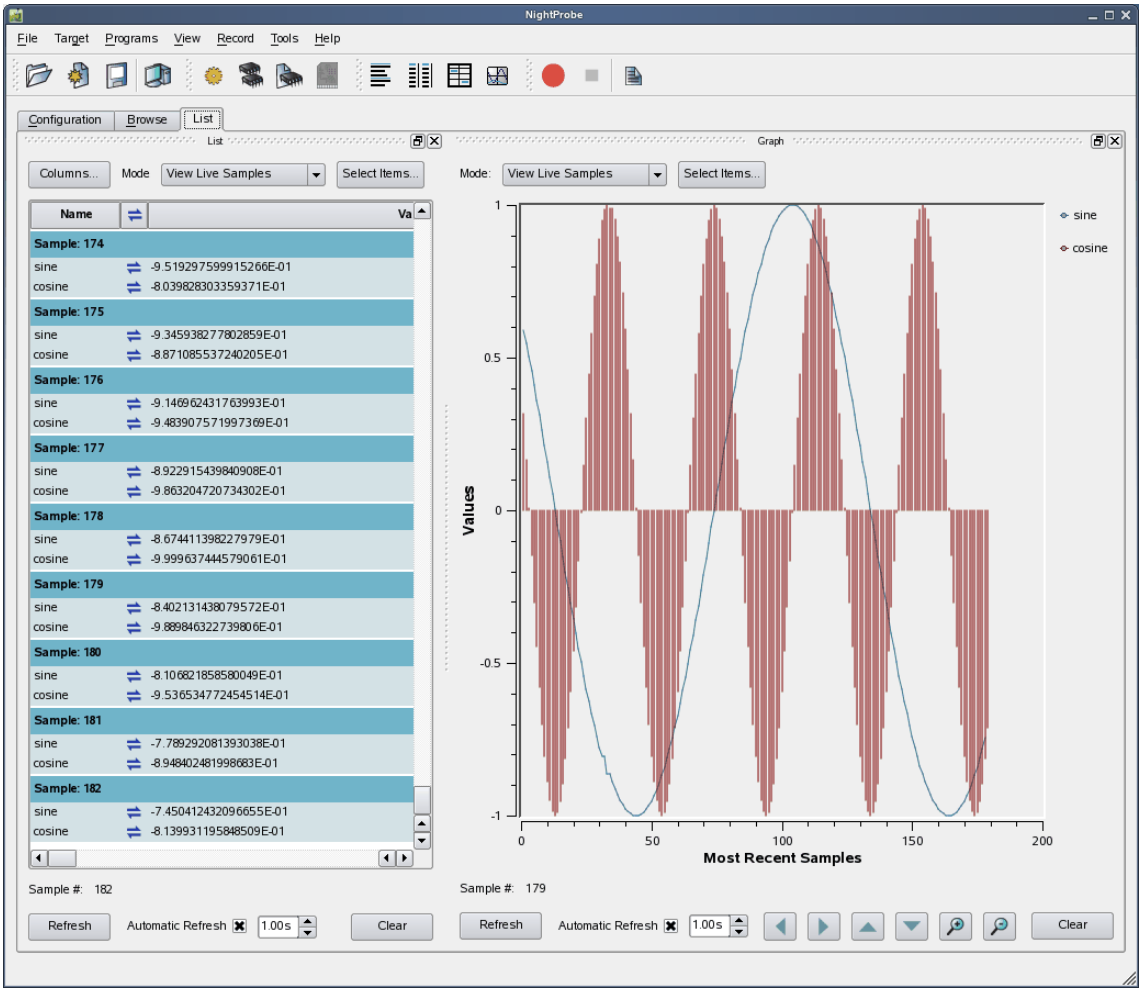


Figure 7-1. Viewing Page with List and Graph Panels

Panels are moved by left-clicking the title bar, dragging them to a new location, and then releasing the mouse button. Depending on the location of the panel when the mouse button is released, the panel will either remain detached or will be inserted into the page again.

To detach the panel from the page without inserting it, click the left-most control box in the upper right-hand corner of the panel.

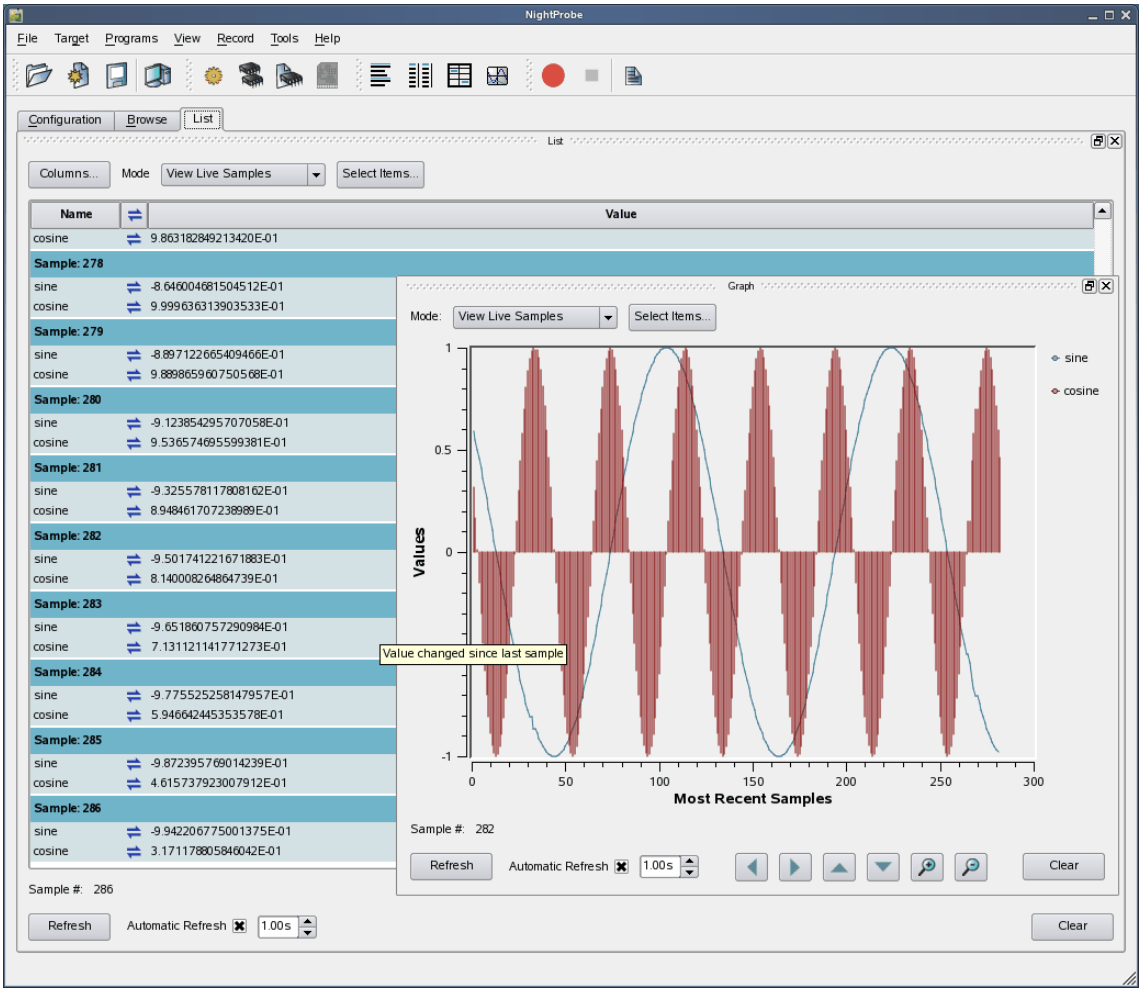


Figure 7-2. Panel Detaches from Page

The Graph panel detaches from the page and becomes free floating. If moved outside the boundaries of the main window and released, the panel will remain detached from the main window. However, even in detached mode, if the main window is iconified, the detached panel will be iconified with it.

To insert a panel into the page at a new location, drag the panel using the left mouse button on its title bar and move it until it approaches a boundary of the page. NightProbe will respond by creating space indicating where the panel will be inserted.

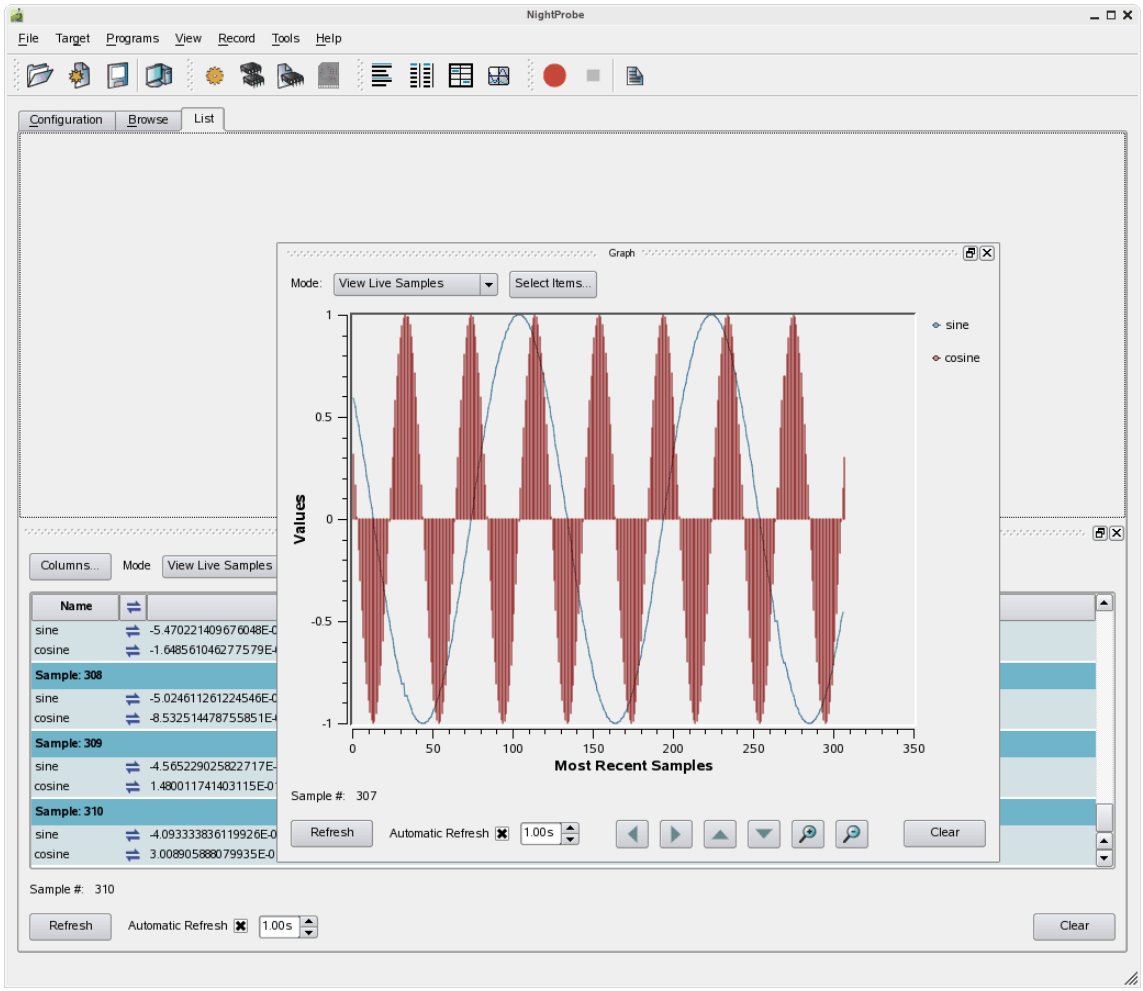


Figure 7-3. Panel Movement in Progress

The figure above shows space being created above the List panel as the Graph panel is dragged towards the upper horizontal boundary of the page.

At this point, releasing the mouse button will cause the Graph panel to be inserted into the page, consuming the recently created space.

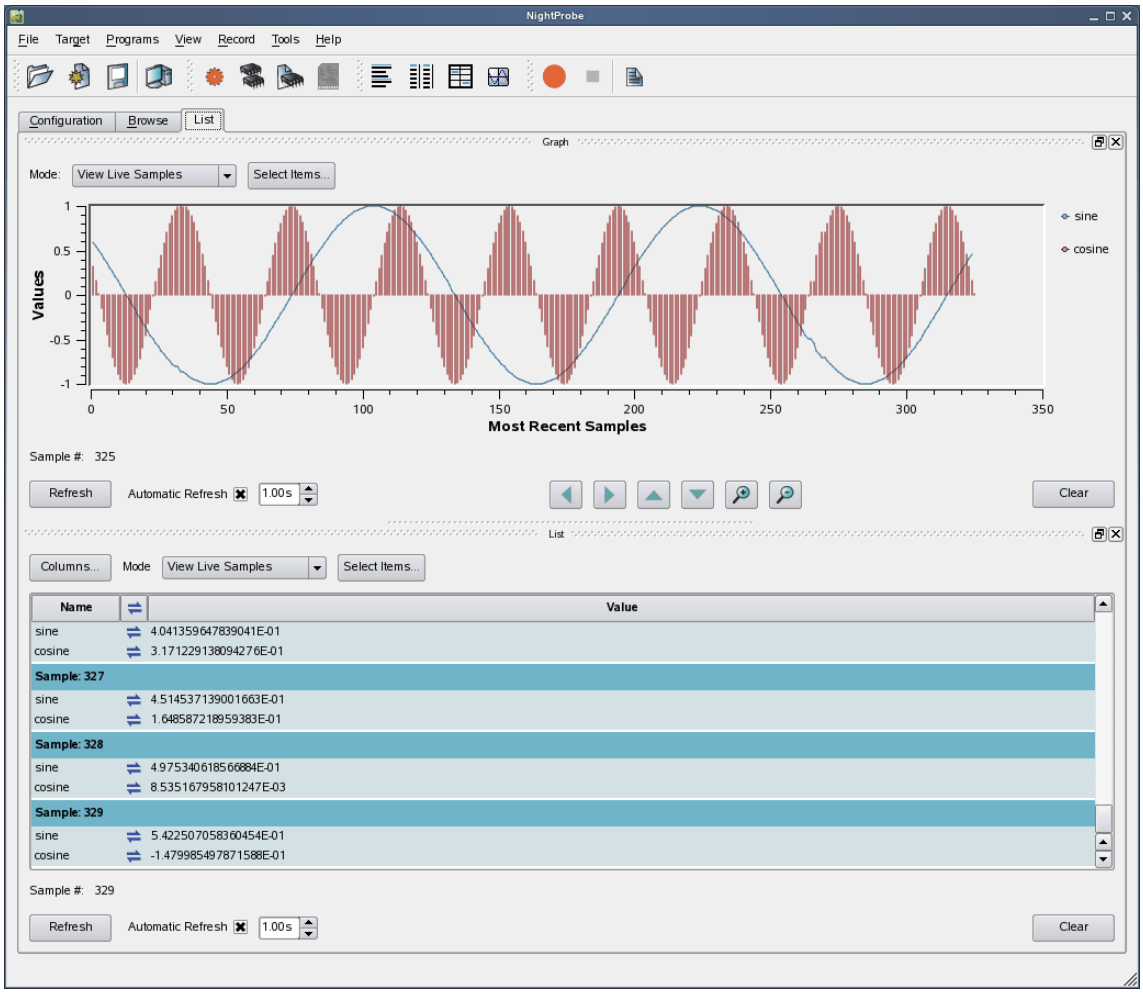


Figure 7-4. Graph Panel on Top of List Panel

IMPORTANT

When attempting to move panels inside of a page, if an empty space does not appear where you desire it, try increasing the size of the main window, decreasing the size of the undocked panel, and moving an alternative edge of the undocked panel near where you want to place it.

By default, NightProbe adds panels to the right-hand side of the page when a new view is created using the View menu.

In the following figure, a Table panel has been added to the right-hand side of the Graph and List panels.

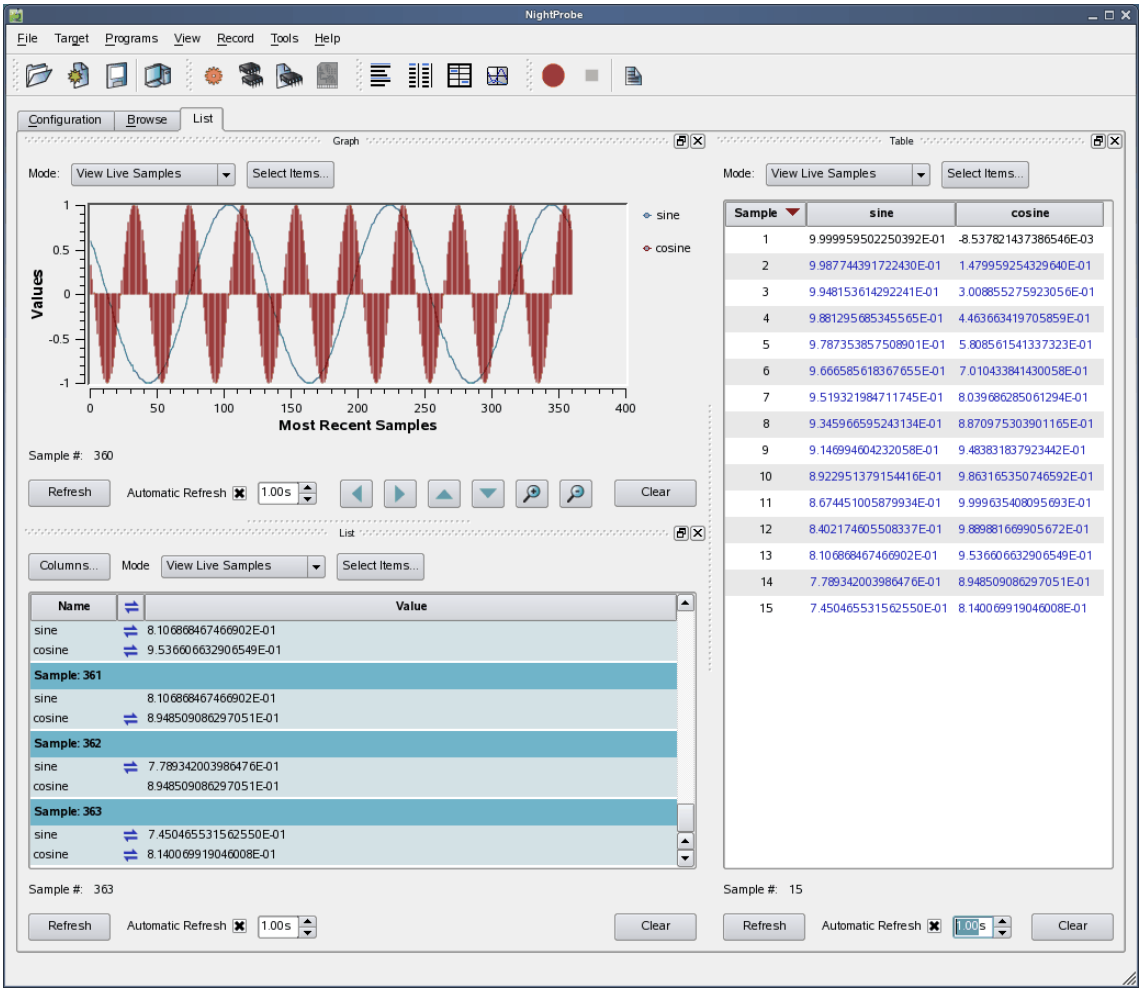


Figure 7-5. Table View added to Page

Panels can be resized by left-clicking on the separator between the panels and dragging it to the desired size.

Another feature of the graphical user interface is the use of tabbed panels. Tabbed panels allow you to maximize your GUI real estate by placing two or more panels in the same location by stacking them on top of each other. You can then raise a panel to the top by clicking on its tab.

To create a tabbed panel, move a panel to the lower horizontal edge of another panel until a tab appears at the bottom of the panel still connected to the page.

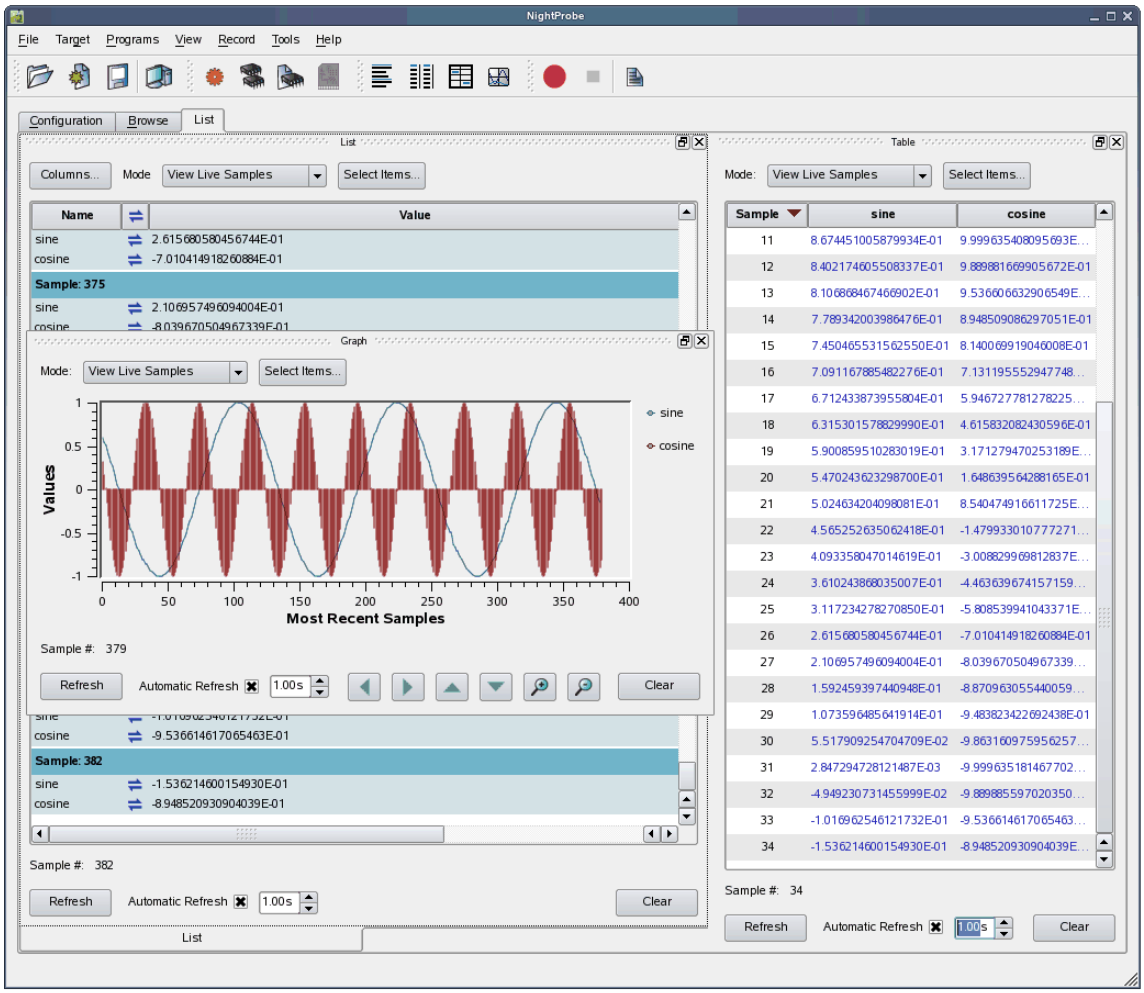
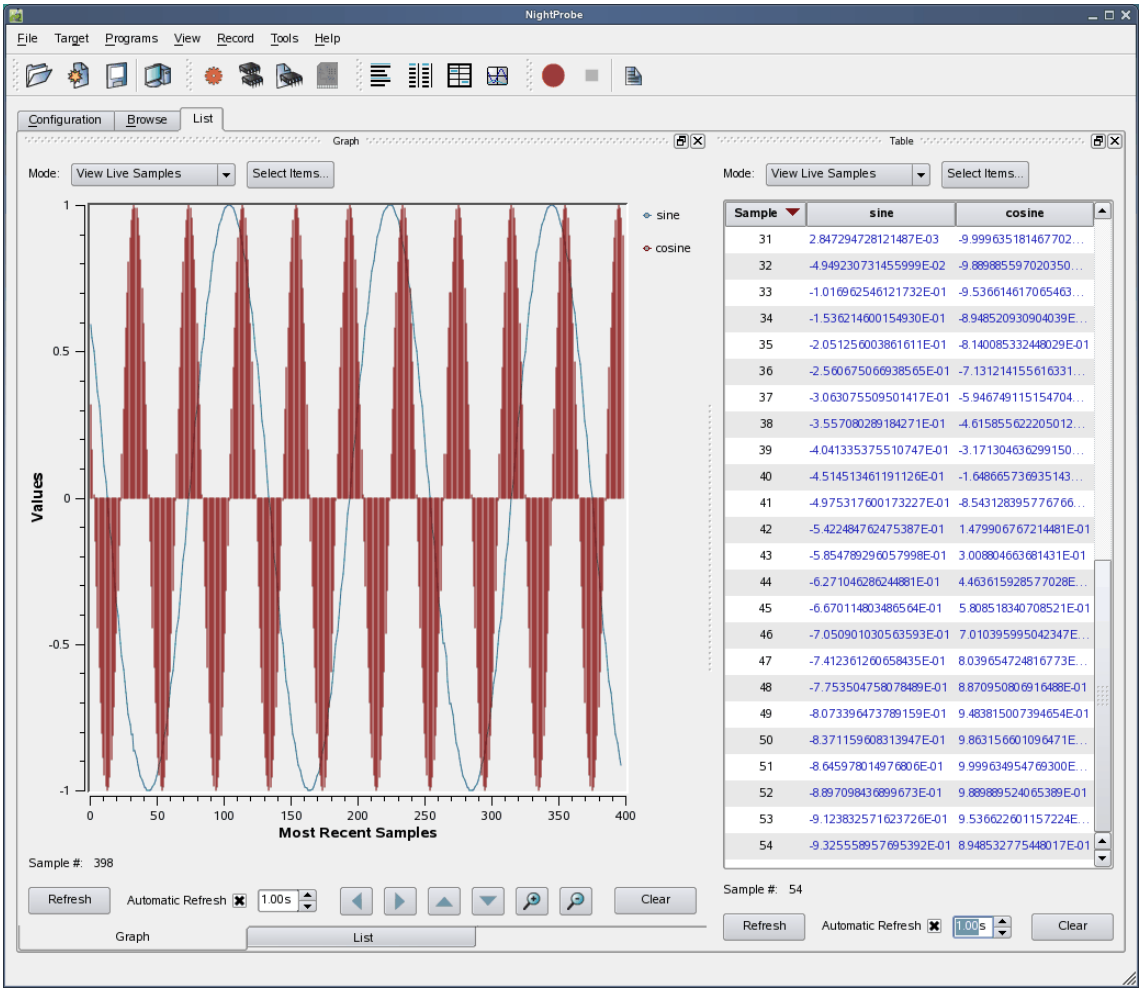


Figure 7-6. Panel in Motion Creating Tab

In the figure above, the Graph panel is being dragged from its original position on top of the List panel towards the bottom of the List panel. A tab appears on the List panel indicating that if the mouse button is released, the Graph and List panels will be tabbed and therefore consume the same area of the page.



IMPORTANT

To move a panel above another panel, move the desired panel to the top boundary of the other panel. If you move a panel to the bottom boundary of another panel, it will become a tabbed panel instead.

The orientation and size of panels within pages is saved as part of a NightProbe session.

8
List View

The List view provides a scrollable list of variable values in a moveable and resizeable panel.

List Panel

The List view allows you to view live samples as well as previously recorded samples from playback files.

It presents each sample as a section in a scrollable list. Within each sample, each variable is listed in its own row.

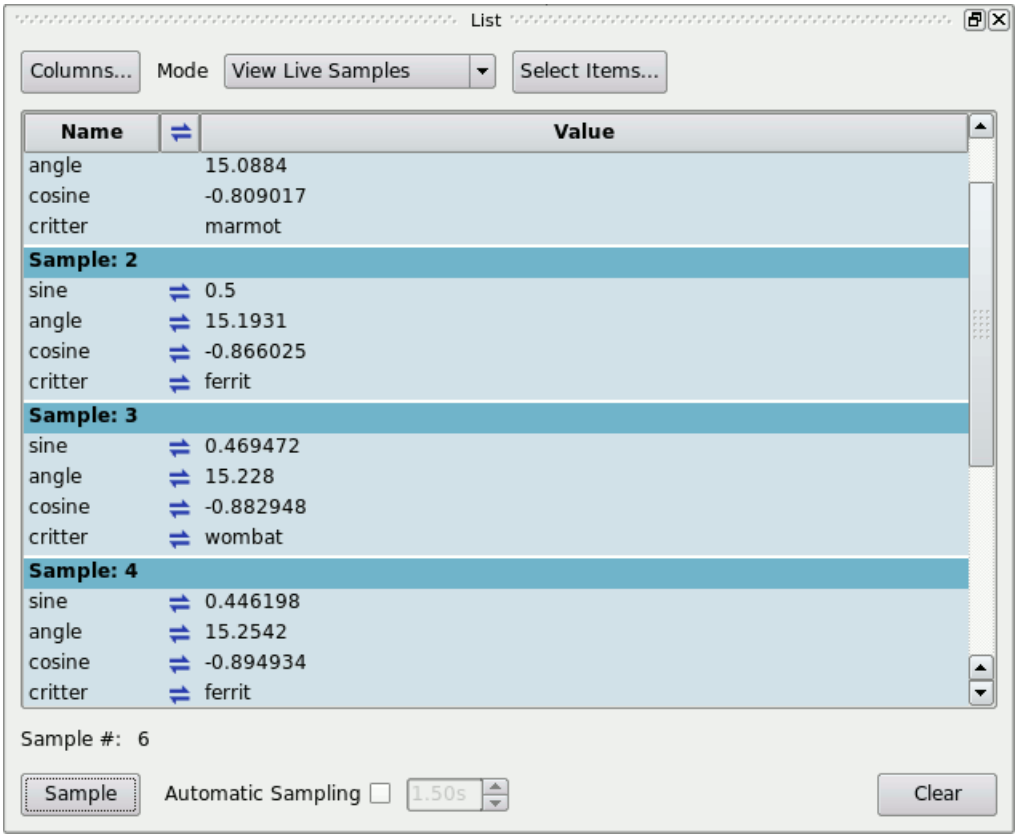


Figure 8-1. List View

The List view consists of three areas:

- the "Selection Area" on page 8-2

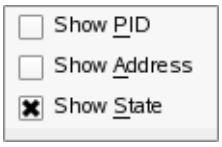
- the "Scrollable List Area" on page 8-4
- the "Sampling Control Area" on page 8-8

Selection Area

The selection area consists of three controls.

Columns

Clicking on the Columns… button presents the following menu:



Checking or unchecking each item controls whether that column is shown in the list.

View Selection

This option list controls which samples are being viewed.

View Live Samples

In this mode, the value of all variables selected for display in the List view are retrieved whenever you press the Sample button at the bottom of the page or whenever an automatic sample occurs. See "Sampling Control Area" on page 8-8.

View Playback Samples

In this mode, the entire contents of the current playback file are loaded into the scrollable list area. Thus all samples recorded to that file are available for viewing in the list. This mode is not available if no playback file has been selected.

Item Selection

By default, when a List view is added, no variables are selected for display.

Pressing the Select… button launches the Item Selection dialog as shown in the next section.

Item Selection Dialog

This dialog controls which variables will be shown in a view.

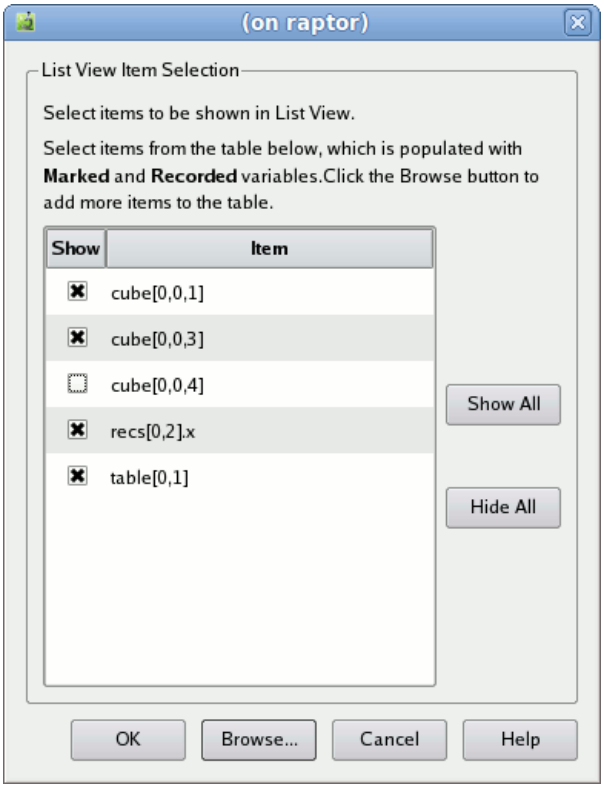


Figure 8-2. Item Selection Dialog

This dialog is populated with all variables whose Mark or Record attributes have been set.

For convenience, you can press the Browse… button to launch a detached Browse page which will allow you to set the Mark and Record attributes using each variable's context menu.

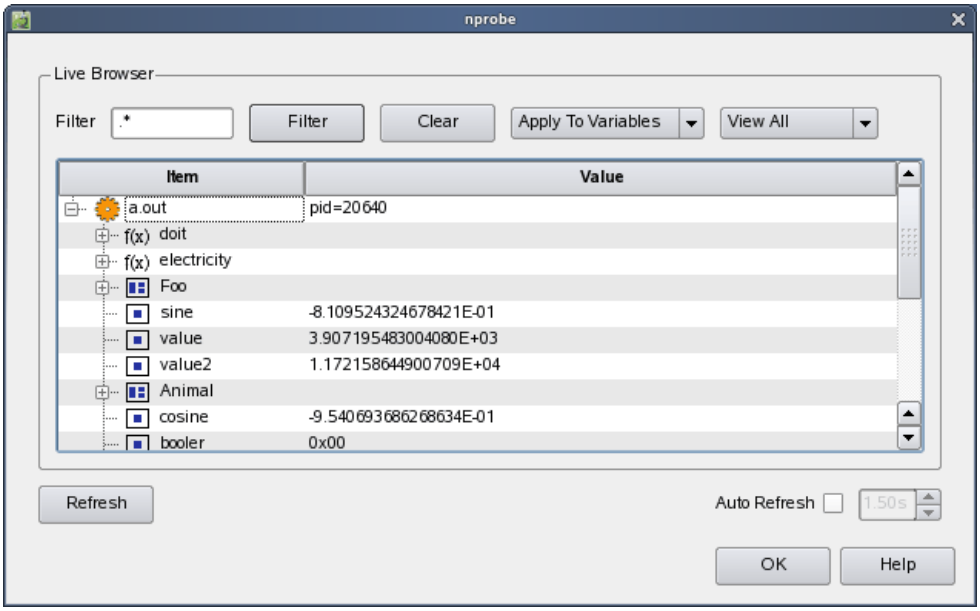


Figure 8-3. Detached Browse Page

NOTE

Double-clicking a variable in the Browse page sets its Mark and Record attributes.

Once items are visible in the Item Selection Dialog, you can select the items you want to include in your view by clicking the Show checkbox for each desired item. The Show All and Hide All buttons are provided for convenience; they set and clear all the checkboxes in the table, respectively.

Additionally, you can use the context menu to change the state of the checkboxes of currently-selected items.

The dialog is designed such that the first time it is launched from a new view, all Marked and Recorded items are set to be shown. Thus pressing the Enter key or the OK button immediately after launching the dialog will cause all such items to be included in the view.

Scrollable List Area

Each sample contains values of all variables which have been selected for viewing. The sample number is shown in the dark blue row.

By default, three columns are shown, but additional columns are available, as controlled by the Column menu described above. The Name of the variable and the Value of the variable are always shown.

The columns available for display include:

Name

This column is always displayed. It shows the simple name of the variable. For additional information on the variable, including its expanded name (if any), move the mouse over the name of a variable and let it hover there. A tooltip will appear with additional information:

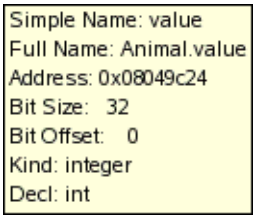


Figure 8-4. Tooltip for Variable Name

PID

If this column is shown, the process ID of the program associated with the variable is shown. If the variable is associated with a non-program resource, the resource ID is shown. Resource IDs are internal numbers used within NightProbe for identifying resources.

Address

If this column is shown, the base address of the variable is shown in hexadecimal.

State

If this column is shown, an icon will appear in the column if the value of the variable has changed since the last sample.

Value

The value of the variable is shown in this column. By default, NightProbe picks an appropriate format for the variable. However, you can specify a particular format for the variable using the Format option of the variable's context menu as described in Formnat in the Browse page.

Variable Context Menu

Each variable's context menu is launched by right-clicking anywhere in a row corresponding to that variable.

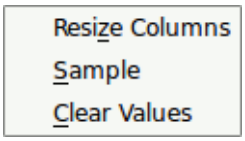


Figure 8-5. List View Variable Context Menu

Resize Columns

This option is not specific to a single variable, it operates on the entire list.

Selecting this option causes the columns to be resized to their optimal width so they can contain the entire contents of all cells in that column. The last column may not resize appropriately if insufficient horizontal space is available. In such a case, increase the horizontal size of the panel, and page, if required.

Sample

This option is not specific to a single variable, it operates on the entire list.

Selecting this option causes a new sample to be taken. This option is not available when viewing playback samples.

Clear Values

This option is not specific to a single variable, it operates on the entire list.

Selecting this option removes all samples from the list. This option is not available when viewing playback samples.

Modifying a Variable

You can modify the value of a variable by double-clicking its value cell from *any* sample in the list. When you double-click the cell, automatic sampling becomes temporarily disabled and the cell becomes an editable field:

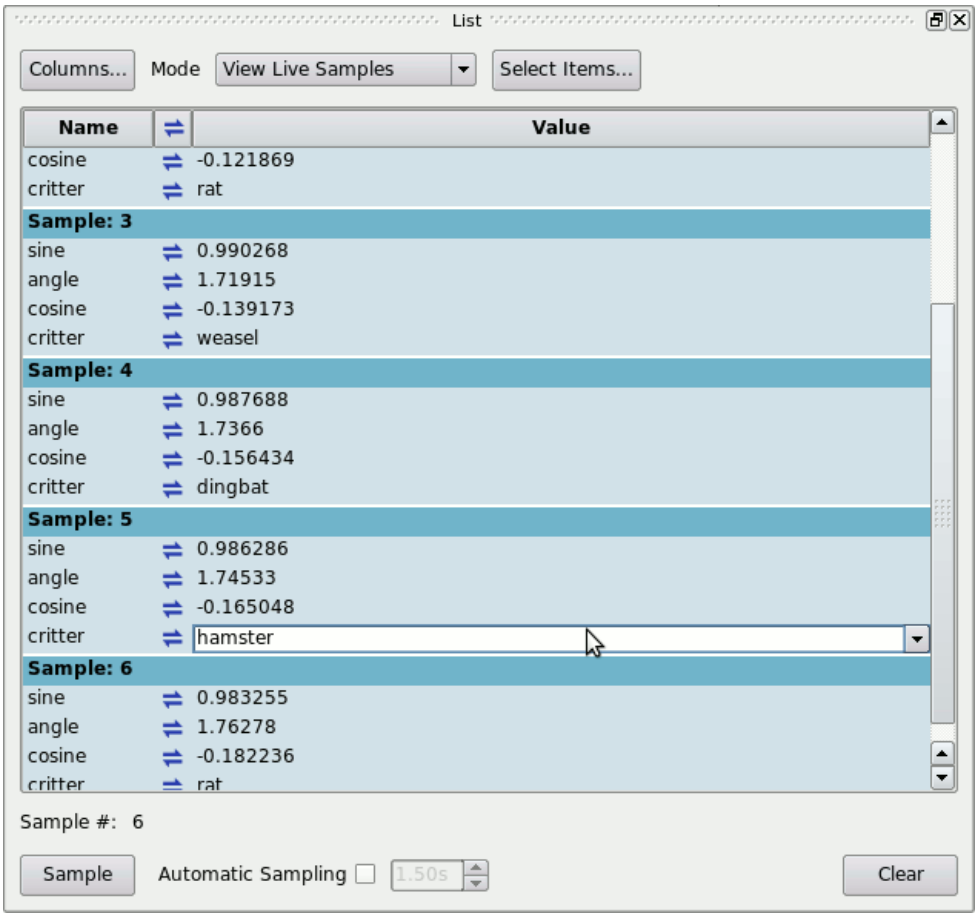


Figure 8-6. Modifying a Variable in a List View

You can then type in a new value for the variable using a format suitable for the variable. For variables of enumerated types, you can type in an enumeration identifier, select an enumeration identifier from the combo list (small arrow at right of editing arrow), or type its underlying integer value. Similarly, for character variables, you can type in a character surrounded in single quotes or provide an integer value.

To commit the new value to the variable, press the Enter key. To cancel modifying the variable (before pressing the Enter key), press the Esc key.

If automatic sampling had previously been selected, it is automatically resumed.

Sampling Control Area

This area controls how samples are added to the list.

These controls are disabled when viewing playback samples.

Sample

Pressing the Sample button causes a new sample to be taken and added to the List.

Automatic Sampling

Checking this checkbox causes samples to be automatically taken at the rate specified in the associated spinbox.

If checked, automatic sampling is temporarily paused when modifying a variable through the List view. See "Modifying a Variable" on page 8-7.

Clear

Pressing the Clear button removes all samples from the list.

9
Table View

The Table view provides a scrollable table of variable values in a moveable and resizeable panel.

Table Panel

The Table view allows you to view live samples as well as previously recorded samples from playback files.

It presents each variable as a column in a scrollable table. Each row represents a sample containing each variable's value in its appropriate column.

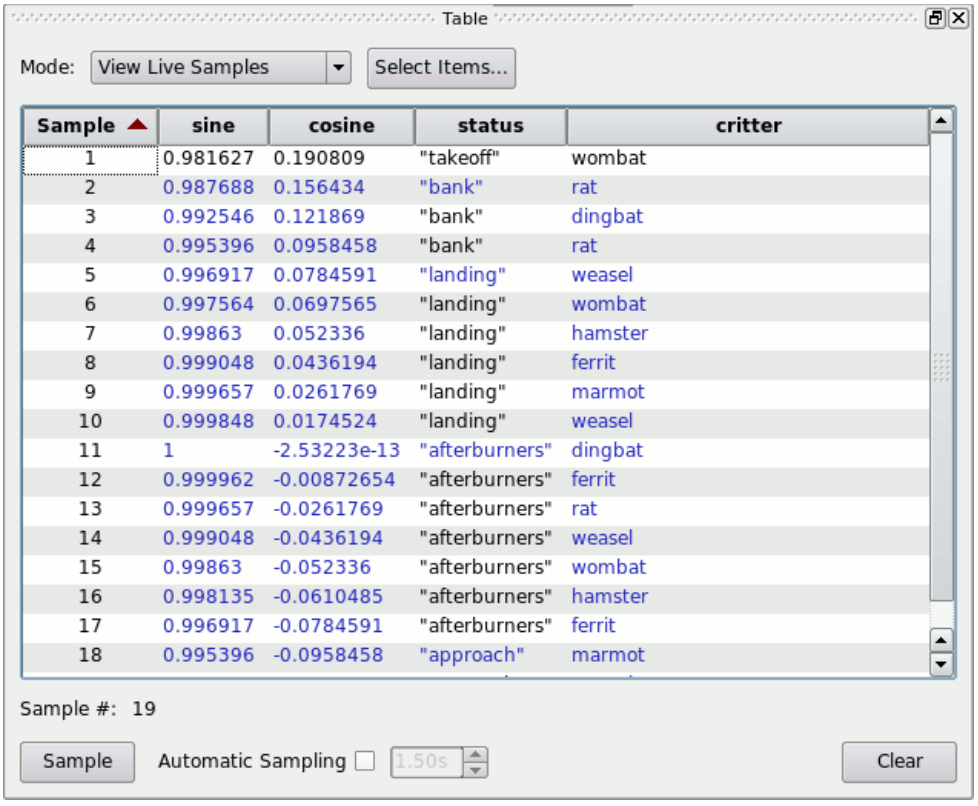


Figure 9-1. Table View

The Table view consists of three areas:

- the "Selection Area" on page 9-2

- the "Scrollable Table Area" on page 9-2
- the "Sampling Control Area" on page 9-4

Selection Area

The selection area consists of two controls.

View Selection

This option list controls which samples are being viewed.

View Live Samples

In this mode, the value of all variables selected for display in the Table view are retrieved whenever you press the Sample button at the bottom of the page or whenever an automatic sample occurs. See "Sampling Control Area" on page 9-4.

View Playback Samples

In this mode, the entire contents of the current playback file are loaded into the scrollable table area. Thus all samples recorded to that file are available for viewing in the table. This mode is not available if no playback file has been selected.

Item Selection

By default, when a Table view is added, no variables are selected for display.

Pressing the Select… dialog allows you to select which variables will be shown.

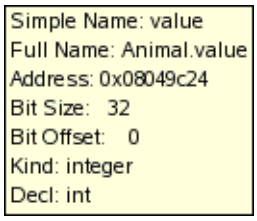
See "Selection Area" on page 8-2 for more information.

Scrollable Table Area

Each sample contains values of all variables which have been selected for viewing. The sample number is shown in the Sample column in the table.

The table can be sorted using any column as the sort key. Clicking on the column header of interest selects that column as the sort key. Subsequent clicks on the same column header reverses the sort order. Thus seeing the maximum or minimum value of a variable at all times is as simple as clicking on a variable's column.

If you place the cursor on a column header for a variable and let it hover, a tooltip will appear with additional information about the variable:



Variable Context Menu

Each variable's context menu is launched by right-clicking anywhere in a column corresponding to that variable.

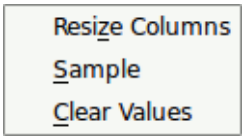


Figure 9-2. Table View Variable Context Menu

Resize Columns

This option is not specific to a single variable, it operates on the entire table.

Selecting this option causes the columns to be resized to their optimal width so they can contain the entire contents of all cells in that column. The last column may not resize appropriately if insufficient horizontal space is available. In such a case, increase the horizontal size of the panel, and page, if required.

Sample

This option is not specific to a single variable, it operates on the entire table.

Selecting this option causes a new sample to be taken. This option is not available when viewing playback samples.

Clear Values

This option is not specific to a single variable, it operates on the entire table.

Selecting this option removes all samples from the table. This option is not available when viewing playback samples.

Modifying a Variable

You can modify the value of a variable by double-clicking its value cell from *any* sample in the table. When you double-click the cell, automatic sampling becomes temporarily

disabled and the cell becomes an editable field:

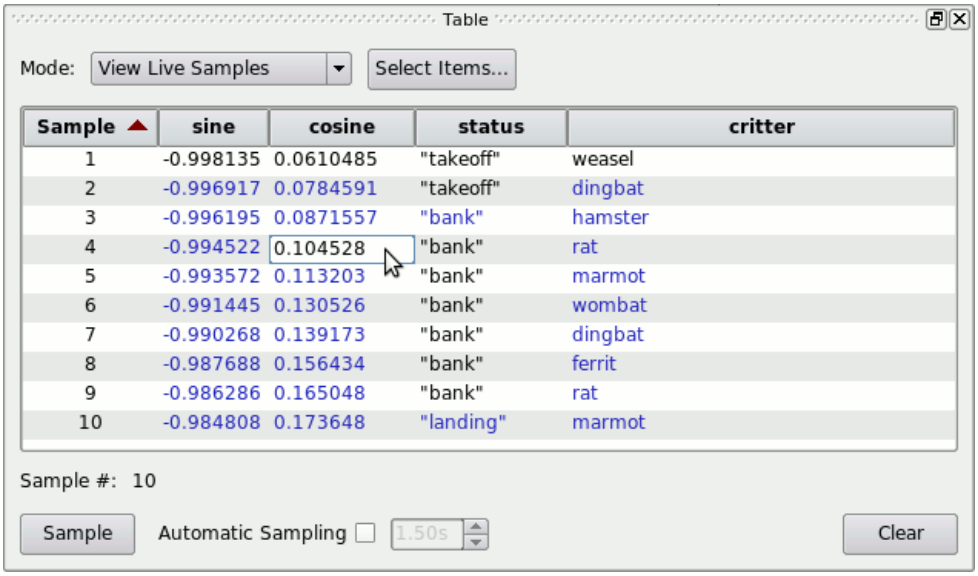


Figure 9-3. Modifying a Variable in a Table View

You can then type in a new value for the variable using a format suitable for the variable. For variables of enumerated types, you can type in an enumeration identifier, select an enumeration identifer from the combo box, or type its underlying integer value. Similarly, for character variables, you can type in a character surrounded in single quotes or provide an integer value.

To commit the new value to the variable, press the Enter key. To cancel modifying the variable (before pressing the Enter key), press the Esc key.

If automatic sampling had previously been selected, it is automatically resumed.

Sampling Control Area

This area controls how samples are added to the table.

These controls are disabled when viewing playback samples.

Sample

Pressing the Sample button causes a new sample to be taken and added to the Table.

Automatic Sampling

Checking this checkbox causes samples to be automatically taken at the rate specified in the associated spinbox.

If checked, automatic sampling is temporarily paused when modifying a variable through the Table view. See “Modifying a Variable” on page 9-3.

Clear

Pressing the Clear button removes all samples from the table.

10
Spreadsheet View

The Spreadsheet view provides a snapshot of selected variable in a moveable and resizeable panel.

Spreadsheet Panel

The Spreadsheet view allows you to view live samples as well as samples being currently recorded. Unlike the List, Table, and Graph views, the spreadsheet only provides a single set of values at any given time -- no history is available with this view.

Variable names and their values may be placed anywhere on a grid.

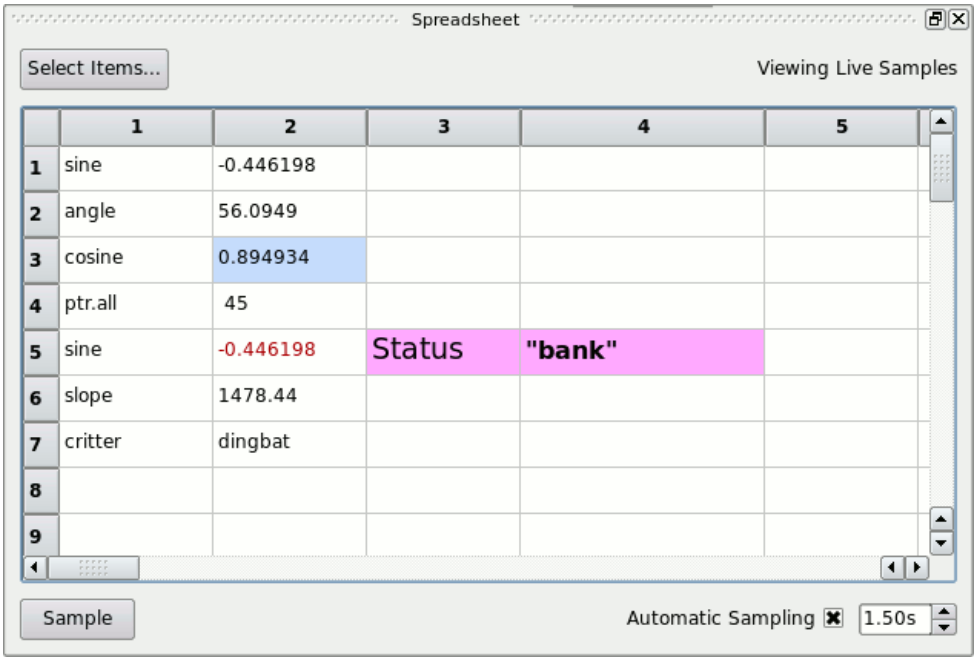


Figure 10-1. Spreadsheet View

You can control the formatting of the cells by controlling their font, foreground color, background color, and other attributes.

The Spreadsheet view consists of three areas:

- the “Selection Area” on page 10-2
- the “Scrollable Spreadsheet Area” on page 10-2

- the "Refresh Control Area" on page 10-8

Selection Area

The selection area consists of two controls.

View Selection

This option list controls which samples are being viewed.

View Live Samples

In this mode, the value of all variables selected for display in the Spreadsheet view are retrieved whenever you press the Refresh button at the bottom of the page or whenever an automatic refresh occurs. See "Refresh Control Area" on page 10-8.

View Recorded Samples

In this mode, the value of all varaibles selected for display in the Spreadsheet view are retrieved from the last recorded sample whenever you press the Refresh button at the bottom of the page or whenever an automatic refresh occurs. See "Refresh Control Area" on page 10-8.

Item Selection

By default, when a Spreadsheet view is added, no variables are selected for display.

Pressing the Select… button allows you to select which variables will be shown.

See the Item Selection dialog in "Selection Area" on page 8-2 for more information.

Scrollable Spreadsheet Area

Once variables have been inserted into the grid, you can move the variable names or their value cells anywhere in the grid. To move cells, select a single cell or multiple cells and drag the selection to a new location and then release the mouse button.

Cell attributes are controlled using a context menu.

Context Menu

A context menu is launched when you right-click in a cell.

You can change the attributes of several cells at once by selecting all of the cells of interest and then right-clicking.

Figure 10-2. Spreadsheet View Cell Context Menu

Select Items...

This option launches the Item Selection dialog which allows you to select which variables will be shown.

See the Item Selection dialog in "Selection Area" on page 8-2 for more information.

Font...

This item launches a font selection dialog which allows you to select a font for the text in the selected cells.

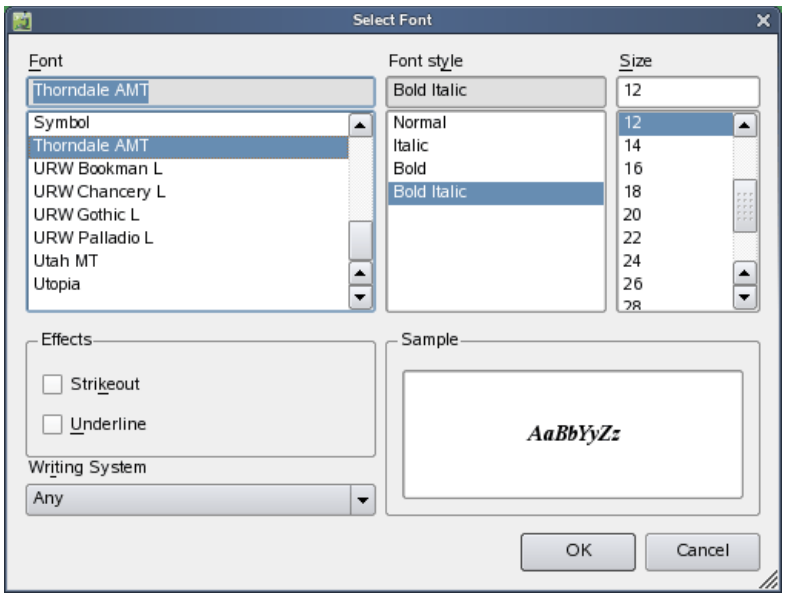


Figure 10-3. Font Selection dialog

Horizontal Alignment
Vertical Alignment

These options allow you to select the alignment of text in the selected cells.

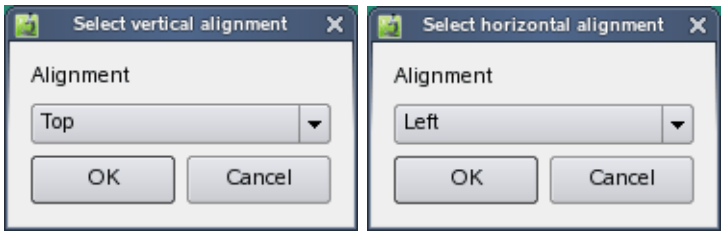


Figure 10-4. Text Alignment Selection dialogs

Background
Foreground

These options launch a color selection dialog which allows you to define the color for the background of the cell or the text displayed in the cell.

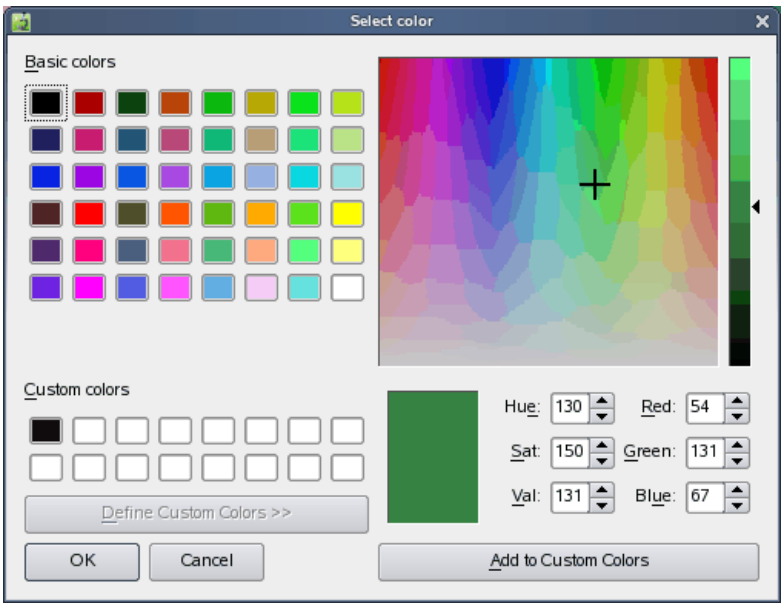


Figure 10-5. Cell Color Selection dialog

Set Cell Color Thresholds

This option launches a color threshold selection dialog which allows you to select color thresholds and gradients affecting the background color of the cell with which they are associated.

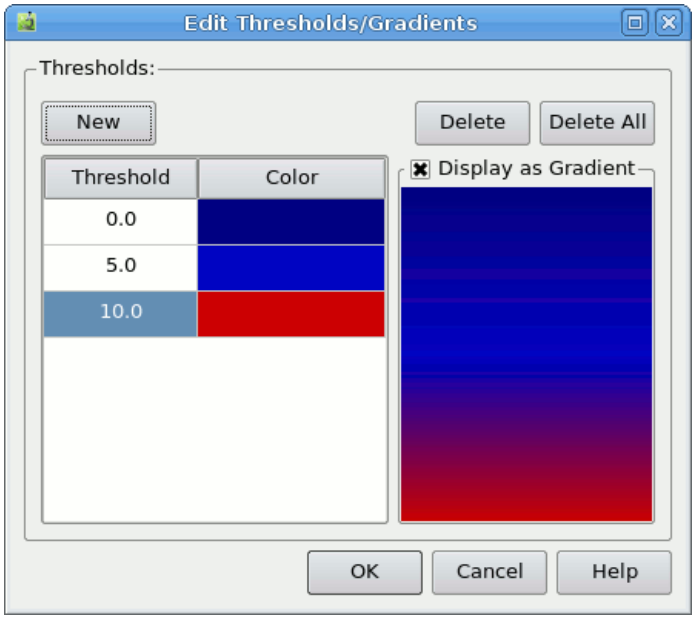


Figure 10-6. Edit Thresholds/Gradients Dialog

To define a new color threshold, click on the New button. A new row is created in the threshold table of the dialog.

The Threshold column of a row is the threshold value. Click on this column and input any valid floating point or integer number (rows with invalid numbers will be ignored).

The Color column is the color to display at the numeric threshold specified in the Threshold column of that row. Clicking on the Color column displays the Cell Color Selection dialog. From that dialog, select a desired color and click OK to set the cell color in the Color column. A valid row (i.e., a color threshold) will have a number in the Threshold column and a corresponding color in the Color column.

Checking the Display as Gradient checkbox causes the thresholds to be displayed as a linear color gradient. If the color threshold table is empty when the gradient group box is checked, two new rows with default values and colors will be added to the threshold table automatically. In its simplest form, a table with two consecutive rows of values and colors (the table is automatically sorted internally for purposes of gradient calculation) will create a linear continuous color gradient with each of the specified colors as endpoints. If the cell's value is outside the range of these endpoints, the cell's color will be that of the endpoint. A sample of the gradient displays within the gradient box.

More than two rows can be used. In the example shown in the Edit Thresholds/Gradients dialog, three thresholds are set. Values between 0.0 and 5.0 remain a variant of dark blue, but the values between 5.0 and 10.0 are spread over blue to red.

Click the OK button to apply the thresholds created to the spreadsheet cell that was selected.

Clicking on the Delete button deletes the currently selected threshold table row. Clicking on Delete All deletes all thresholds in the threshold table. Clicking on the Cancel button cancels the color threshold selection and closes the dialog. Clicking on the Help button opens the help system to display this section.

Show Grid
Show Column Headers
Show Row Headers

These checkboxes control whether the respective items are shown in the spreadsheet.

Insert Row
Insert Column
Insert Rows...
Insert Columns...

These options insert single rows or columns, respectively, above or to the left of the selected cell.

The plural options first prompt you for the number of rows or columns you wish to insert. The singular forms insert a single row or column.

Delete Rows
Delete Columns

These options delete the rows or columns, respectively, corresponding to the selected cells.

Select All

This option selects all cells in the spreadsheet.

Clear SpreadSheet

This option removes the contents of all cells and all formatting in the spreadsheet.

Modifying a Variable

You can modify the value of a variable by double-clicking its value cell. When you double-click the cell, automatic refresh becomes temporarily disabled and the cell becomes an

editable field:

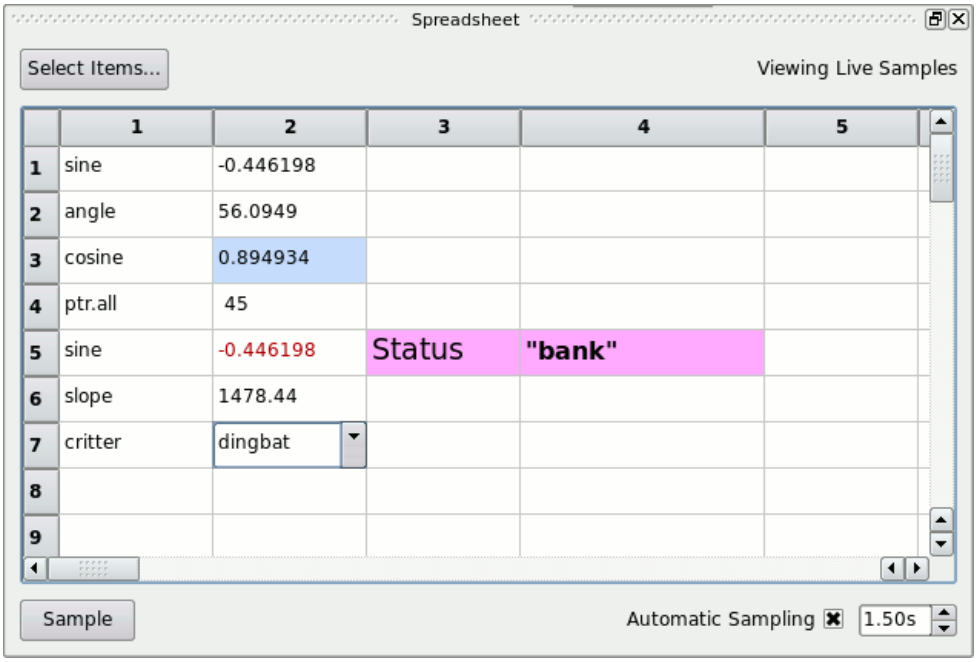


Figure 10-7. Modifying a Variable in a Spreadsheet View

You can then type in a new value for the variable using a format suitable for the variable. For variables of enumerated types, you can type in an enumeration identifier, select an enumeration identifier from the combo list (small arrow at right of editing arrow), or type its underlying integer value. Similarly, for character variables, you can type in a character surrounded in single quotes or provide an integer value.

To commit the new value to the variable, press the Enter key. To cancel modifying the variable (before pressing the Enter key), press the Esc key.

If automatic refresh had previously been selected, it is automatically resumed.

Adding Text to the Spreadsheet

You may type text directly into any cell in the spreadsheet using the procedure described above in "Modifying a Variable" on page 10-7.

If the cell is not associated with a variable's value, it simply changes the text in the cell and has no other effect.

Refresh Control Area

This area controls how samples are shown in the spreadsheet.

Sample

Pressing the Sample button causes a new sample to be taken when viewing live values. When viewing variables currently being recorded, it takes the values from the last recorded sample.

Automatic Sampling

Checking this checkbox causes samples to be automatically taken at the rate specified in the associated spinbox.

If checked, automatic refresh is temporarily paused when modifying a variable through the Spreadsheet view. See “Modifying a Variable” on page 10-7.

11
Graph View

The Graph view provides lines graphs in a moveable and resizeable panel.

Graph Panel

The Graph view allows you to view live samples as well as previously recorded samples from playback files.

It presents each variable as a separate line graph.

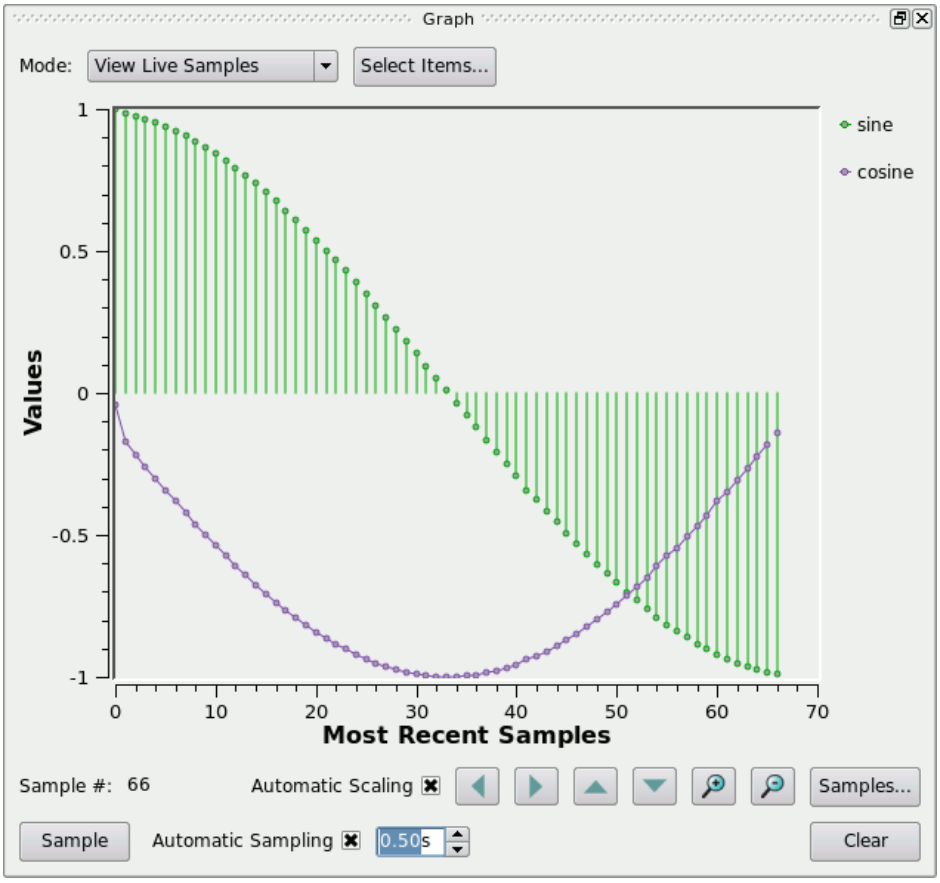


Figure 11-1. Graph View

The Graph view consists of three areas:

- the “Selection Area” on page 11-2

- the "Graph Area" on page 11-2
- the "Viewing Control Area" on page 11-7

Selection Area

The selection area consists of two controls.

View Selection

This option list controls which samples are being viewed.

View Live Samples

In this mode, the value of all variables selected for display in the Graph view are retrieved whenever you press the Refresh button at the bottom of the page or whenever an automatic refresh occurs. See "Viewing Control Area" on page 11-7.

View Playback Samples

In this mode, the entire contents of the current playback file are loaded into the scrollable graph area. Thus all samples recorded to that file area available for viewing in the graph. This mode is not available if no playback file has been selected.

Item Selection

By default, when a Graph view is added, no variables are selected for display.

Pressing the Select Items… button allows you to select which variables will be shown.

See the Item Selection dialog in "Selection Area" on page 8-2 for more information.

NOTE

Unlike other views, when invoking the Item Selection dialog from a Graph view, the default "show" settings are set to "hide". Most often, one or very few entities are desired to be displayed on the same graph.

Graph Area

As each sample occurs, a new data point is added to the graph for each line.

Graph Context Menu

The graph's context menu is launched by right-clicking anywhere in the graph.

Figure 11-2. Graph View Context Menu

Pan Right
Pan Left
Pan Down
Pan Up

These options and their keyboard shortcuts move the viewport in the specified direction. Thus panning to the left will show you older data and panning up will show you values of greater magnitude.

Zoom In
Zoom Out

These options change the zoom factor both vertically and horizontally.

Vertical Zooming
Horizontal Zooming

These options allow the mouse wheel to control zooming in the selected directions.

Title
Title Font
Title Color

These options present dialogs that allow you to change attributes of the graph's title.

Clear Graph

This option clears the graph. The curves will begin to be replotted starting with the next sample.

Automatic Scaling

When checked, this causes the graph to automatically scale in both the X and Y dimensions so all data points from all samples in the graph are visible. Such scaling is applied after any per-variable scaling (see "Scaling..." on page 11-6).

Background Color

This option launches a color selection dialog that allows you to set the background color of the graphing area.

Legend Item Context Menu

Each item in the legend has a context menu that is launched when you right-click on the item in the legend. The menu has a single item which launches the Edit Curve Attributes dialog.

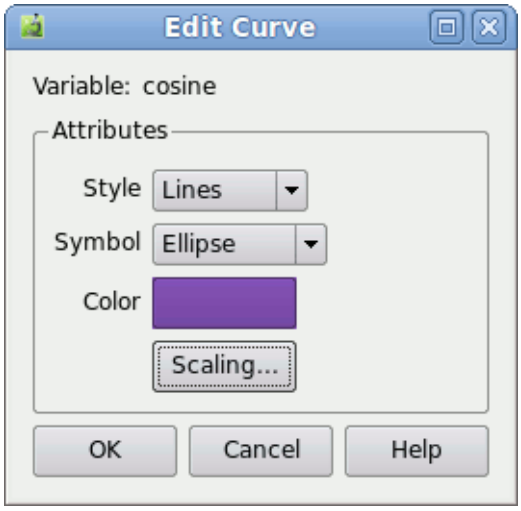


Figure 11-3. Edit Curve Attributes Dialog

Style

This option list allows you to select a line style from the following choices:

No Curve

This option removes the item from the visible graph, although values are still be plotted internally and the item remains in the Legend. If you subsequently change the line style, the previously invisible line will appears.

Lines

This is the default style. It creates a line between each point in the graph.

Steps

This option provides a stepped line to be drawn between points.

Sticks

This option creates a vertical line to be drawn from the 0.0 point to the value of each item.

Dots

This option prevents lines from being drawn between values, only the item's symbol is drawn.

Symbol

This option list allows you to select a symbol for the item.

Color

This option launches a color selection dialog which defines the color of the line and symbol drawn for each item.

Scaling...

This option launches a Graph Scaling dialog, as shown here.

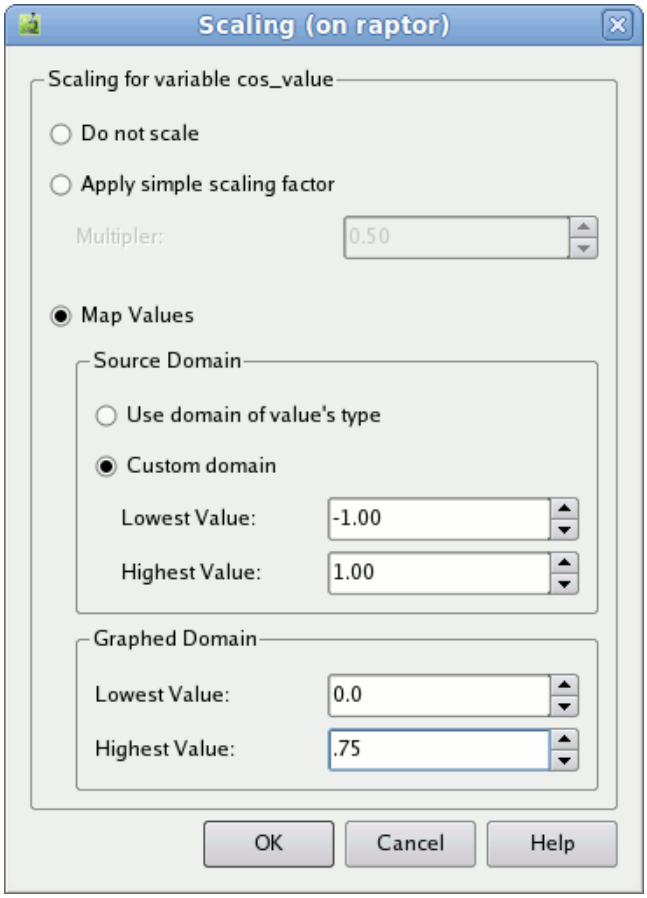


Figure 11-4. Graph View Scaling Dialog

The dialog allows you to set scaling properties on the domain of values of a specific variable item plotted in the graph.

This is especially useful when you have multiple items in the graph whose value domains differ significantly.

Do Not Scale
Apply simple scaling factor
Map Values

These mutually-exclusive options control whether scaling is applied and how it is applied.

When Apply simple scaling factor is selected, the Multiplier field is enabled and allows you to define the multiplier that will be used for all values of the variable in question.

When Map Values is selected, the scaling is more complex. A Source Domain defines the values to be mapped. If a Custom domain is selected, then values outside the endpoints of the specified range are set to the respective endpoint before scaling; otherwise, the Source Domain includes all possible values of the variable's type. The Graphed Domain is the range of values that can appear on the graph.

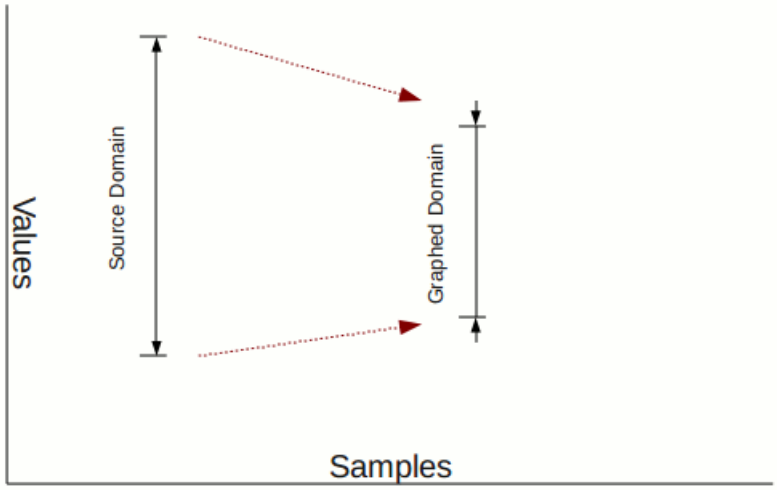


Figure 11-5. Relationship of Source Domain to Graph Domain

Viewing Control Area

This area controls the how samples are added to the graph and provides zooming and panning control.

Sample

Pressing the Sample button causes a new sample to be taken and added to the Graph.

Automatic Sampling

Checking this checkbox causes samples to be automatically taken at the rate specified in the associated spinbox.

Automatic Scaling

When checked, this causes the graph to automatically scale in both the X and Y dimensions so all data points from all samples in the graph are visible. Such scaling is applied after any per-variable scaling (see "Scaling..." on page 11-6).

Panning Controls

These four buttons provide left, right, up, and down panning of the viewport over the data, respectively.

Zooming Controls

These two zoom icons provide for zooming in and zooming out, respectively.

Samples...

This button allows you to define the number of samples that are graphed. Only the most recent samples are graphed.

Clear

Pressing the Clear button clears the graph -- the lines will begin to be replotted starting with the next sample.

12
Recording

NightProbe's recording feature provides for capturing the value of specific variables and saving the values for subsequent analysis. Additionally, it allows you to stream the recorded data to NightTrace or user applications for live analysis, alternative display, or subsequent processing.

Selecting Variables for Recording

Variables are selected for recording using the Browse page.

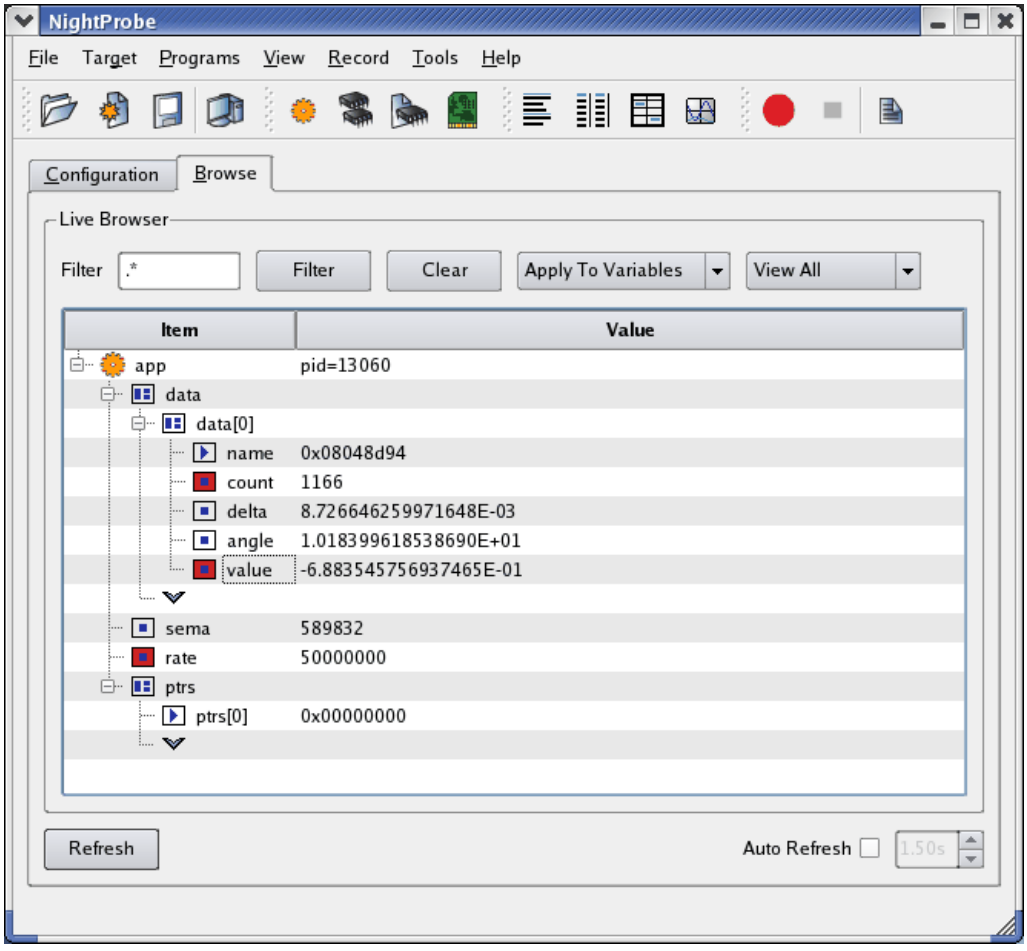


Figure 12-1. Selecting Variables for Recording from the Browse Page

In the figure above, three items have been selected for recording, as shown by the red background in their icons.

Record Attribute

Each item has a Record attribute. When set, the item is selected for recording. To set an item's Record attribute, check the Record checkbox in the context menu for the item, which is launched by right-clicking on the item in the browse page.

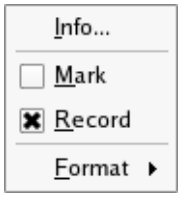


Figure 12-2. Scalar Item Context Menu with Record Attribute Set

The configuration page shows all variables that have their Record attribute set under the Variables item in the Recording tree.

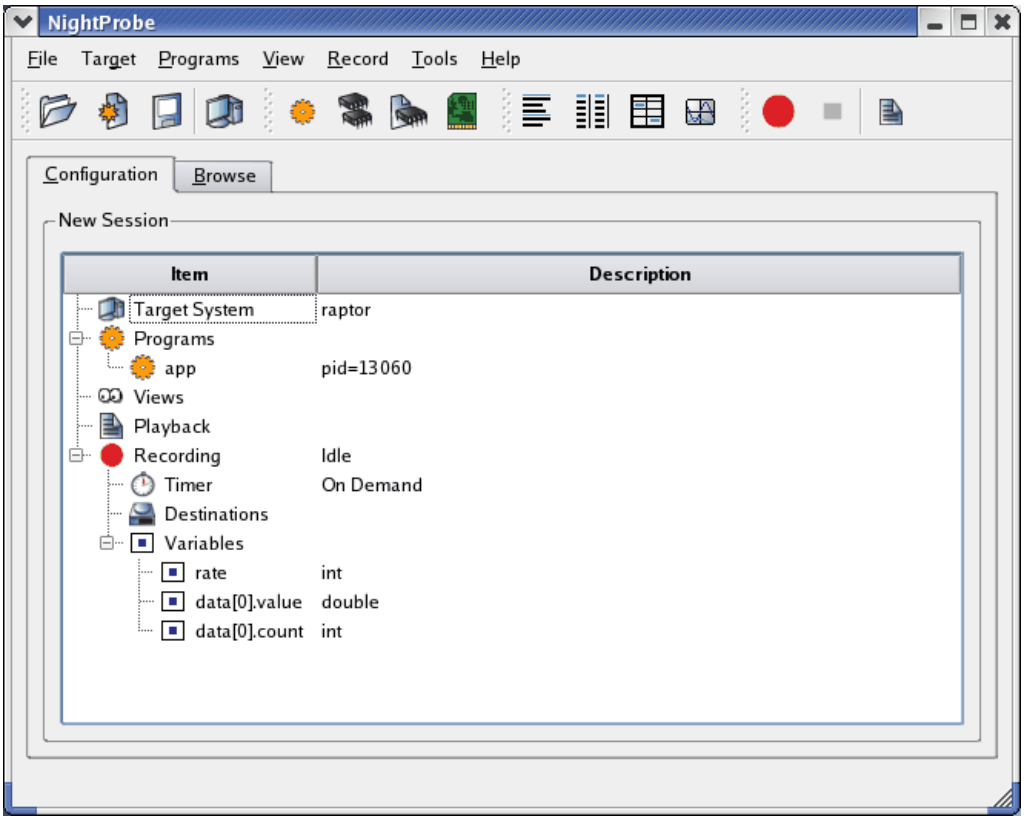


Figure 12-3. List of Recorded Variables Shown in Configuration Page

Recording Timing Methods

NightProbe provides the following timing methods that control when samples are recorded:

- "Clock Timing" on page 12-3
- "Frequency Based Scheduler Timing" on page 12-4
- "Application Trigger Timing" on page 12-5
- "On Demand Timing" on page 12-6

Selection of any of the first three methods causes an appropriate dialog to be launched which allows you to refine the selection. Selection of On Demand timing requires no refinement.

Clock Timing

Clock timing provides asynchronous data recording. Samples will be recorded based on the system clock at a user-selectable rate.

The Sampling Rate Dialog allows you to define the rate; it is launched when the Clock method is selected from a Record Timing menu, or when you double-click the description Timer item in the Recording tree of the Configuration page.

The system clock begins to count once recording is started.

Sampling Rate Dialog

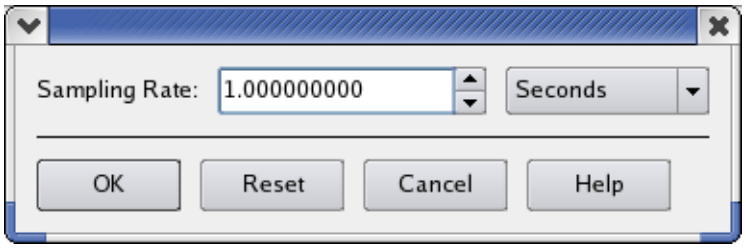


Figure 12-4. Sampling Rate Dialog

This dialog allows you to specify the rate by typing a number in the text field or using the spinbox arrows to increase or decrease the number shown.

The option list to the right of the Sampling Rate field allows you to select the units that apply to the sampling rate value.

Frequency Based Scheduler Timing

Frequency Based Scheduler timing provides synchronized recording if you application is also scheduled on the FBS you choose.

The Frequency Based Scheduler Timing dialog allows you to select an FBS and the period and cycle at which recording should occur. It is launched when the Frequency Based Scheduler method is selected from a Record Timing menu, or when you double-click the description of the Timer item in the Recording tree of the Configuration page.

Frequency Based Scheduler Timing Dialog

Figure 12-5. Frequency Based Scheduler Timing Dialog

Key

The Key is a required field. It identifies the FBS of interest. The FBS specified by the value in the Key field does not have to exist when you close the dialog, but it must exist when recording is started.

Pressing the Select... button to the right of the Key field launches a standard NightProbe selection dialog which allows you to choose a key from the list of Frequency Based Scheduler that are currently active on the target system.

Starting Cycle

The Starting Cycle field defaults to zero. This controls the first cycle within a frame when a sample will be recorded.

Period

The Period field defaults to one. This field controls the number of period of sampling after the cycle defined in the Starting Cycle field. A value of one indicates that sampling will occur for every cycle in the frame once the Starting Cycle occurs, inclusive. A value of two implies every other cycle, etc.

Application Trigger Timing

Application Trigger timing provides synchronized recording between user applications using the NightProbe Trigger API and NightProbe itself.

The NightProbe Trigger API allows user applications to cause NightProbe to sample the data by making a simple API call. See "NightProbe Trigger API" on page 13-20 for more information on the API.

The Application Trigger dialog allows you to select the triggering program's handle, which is created at runtime by the application through the API. The dialog is launched when the Application Trigger method is selected from a Record Timing menu, or when you double-click the description of the Timer item in the Recording tree of the Configuration page.

Application Trigger Dialog

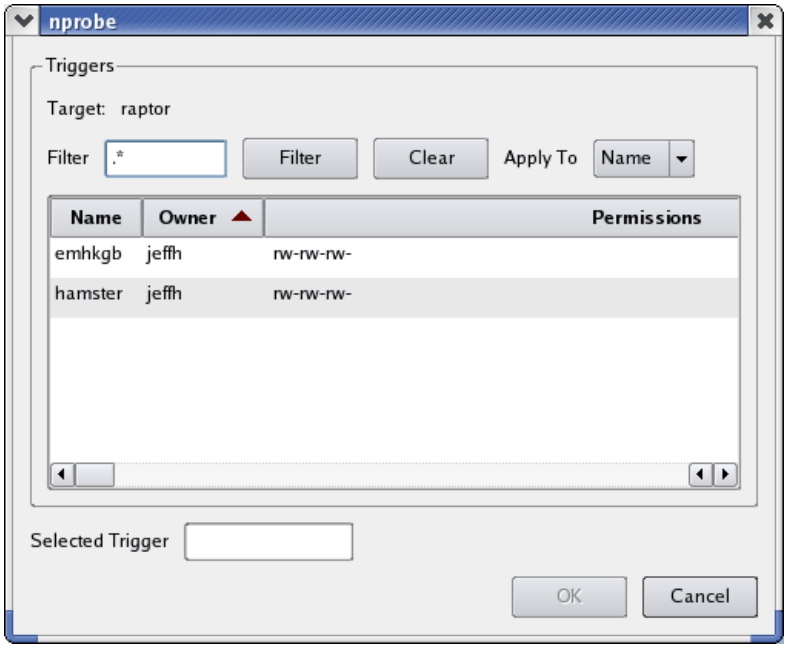


Figure 12-6. Application Trigger Dialog

The dialog is a standard NightProbe selection dialog which displays all active NightTrace Trigger API programs on the target system.

You can select an item from the list or type in the name of the trigger handle which will be created subsequently, as long as it is created before recording is started.

On Demand Timing

On Demand timing is indicates that samples will only be taken when the spreadsheet view is tracking recorded samples and a Refresh operation occurs. See "Spreadsheet Panel" on page 10-1for more information.

Recording Destinations

Recording destinations include the following:

- "File" on page 12-6
- "NightTrace" on page 12-6
- "Program" on page 12-10

File

Recording to a file allows you to subsequently replay the data within NightProbe using the Playback feature, or from user applications using NightProbe's DataStream API, as described in "NightProbe Datastream API" on page 13-1.

A standard file selection dialog is presented when this recording destination is chosen. It allows you to browse and select an existing file, or to supply the name of the file to be created when recording starts.

The files displayed in the selection dialog are relative to the host system. However, the filename selected or specified will be referenced relative to the target system when recording begins.

NightTrace

Recording to NightTrace allows you to store the recorded samples in NightTrace format for subsequent analysis or for live analysis with NightView as the samples are streamed to the NightTrace GUI.

NightProbe builds a NightTrace session files customized for the variables that are to be recorded. It includes event descriptions and formatting associated with the events holding the variables' values, as well as a NightTrace user daemon definition which can be used within the NightTrace GUI to collect the events.

The NightTrace Destination dialog is launched when the To NightTrace... method is selected from a Record Destinations menu, or when you double-click the description of the NightTrace item in the Recording Destinations tree of the Configuration page.

NightTrace Destination Dialog

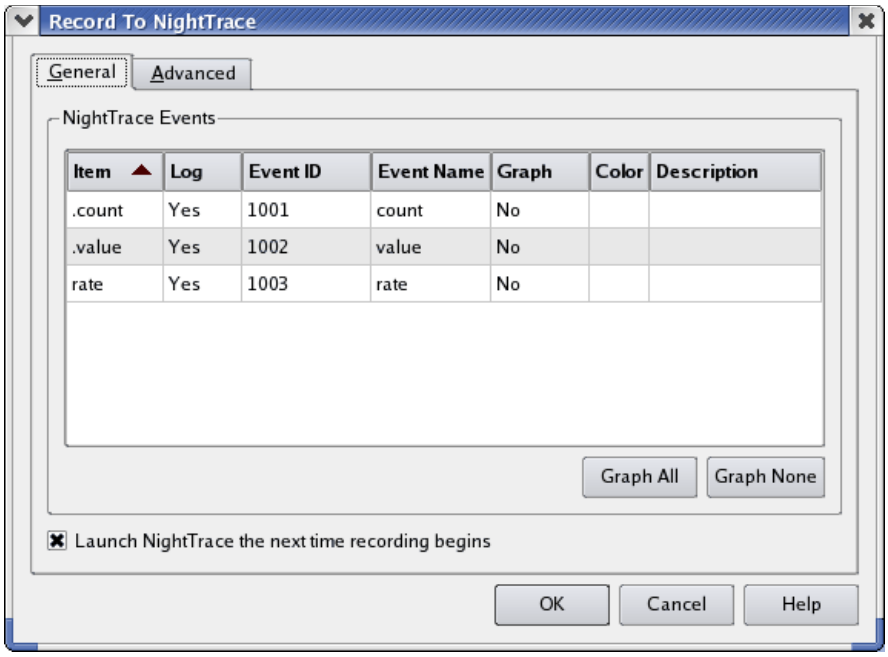


Figure 12-7. NightTrace Destination Dialog

The dialog allows you to change the attributes associated with the NightTrace events used to record the data, select specific variables for graphing within NightTrace, and for advanced users, control the key file, session file, and timing source to be used by NightTrace.

General Tab

The General tab displays all items marked for recording. It contains a context menu which allows you to set or clear the Graph and Log attributes for all items in the table.

NightProbe uses the information from this tab to create a NightTrace session file which is automatically supplied to NightTrace if the Launch NightTrace checkbox is checked on the tab. Regardless, a NightTrace session file is created and can be used subsequently with NightTrace. The name of the session file is automatically generated and can be controlled using the Advanced tab.

Item

The Item column simply shows the simple name of the variable or variable component selected for recording.

Log

The Log column indicates whether the item will be included in the samples sent to NightTrace. The default is Yes, which means it will be included. You can click on a cell in this column and select either No or Yes.

Event ID

The Event ID is the number that identifies a trace event as being associated with this item within NightTrace. By default, NightProbe assigns unique numbers to each item, but you can change the number by clicking in the cell and edit the value shown. If you change the number, NightProbe will not ensure that it is unique (since it is possible you may want more than one item to have the same Event ID number, although this is not recommended, so as to avoid confusion).

The Event ID value corresponds to the first argument to the `trace_event_arg` call (and friends) which is made by the NightProbe server. See **`trace_event(1)`** or the NightTrace User's Guide for more information.

Event Name

The Event Name is automatically generated by NightProbe based on the name of the item being recorded. You can change the name by clicking in the cell and editing the displayed text. You should only choose text which forms an identifier consisting of letters, numbers, and underscores, with a letter being the first character.

The value of Event Name is used to symbolically display the Event ID in NightTrace.

Graph

The Graph column indicates whether NightProbe should include a data graph for the item in the NightTrace display. You can adjust this setting by clicking the cell and selecting Yes or No from the option list.

Color

The Color column allows you to select the color of the data graph for items whose Graph setting is Yes. NightProbe automatically generates a pseudo-random color for such items. To change the color, click the appropriate cell. A standard color selection dialog is presented.

Description

The Description column allows you to customize how NightTrace will describe events associated with item values. By default, NightTrace will show the name of the event and the arguments which represent the value of the item. You can override this default description by clicking in the cell and providing a new description.

Descriptions usually take the form of NightTrace format() statements, e.g.

```
format("The value of pi is %f", arg_dbl())
```

Consult the NightTrace User's Guide for more information on event descriptions.

Graph All / Graph None

These buttons change the Graph settings for all items in the table.

Launch NightTrace Checkbox

This checkbox controls whether NightTrace will be automatically launched when recording is next started. The launch only occurs once, unless this checkbox is again manually rechecked between recording sessions.

Advanced Tab

The advanced tab affords you control over the choice of the NightTrace key file and the generation of the NightTrace session file, holding the data defined by the General tab.

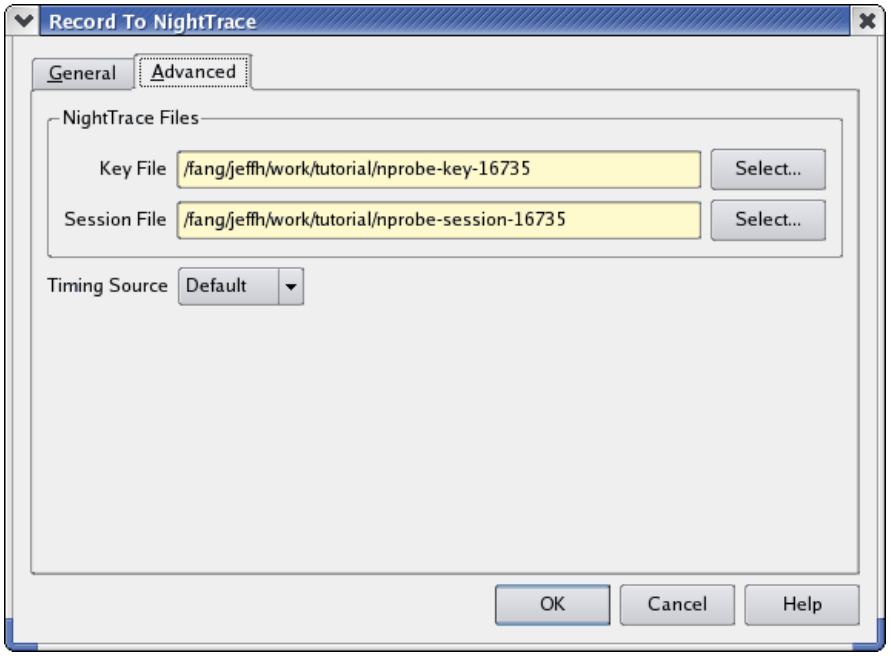


Figure 12-8. NightTrace Destination Dialog Advanced Tab

Key File

The Key File is the file name that is used to define the shared memory segment which is used between the NightProbe server and the NightTrace user daemon. It is same thing as the first parameter that you would pass to the

trace_begin NightTrace API function, if you were writing an application that wanted to log NightTrace data. See trace_begin(1) or the NightTrace User's Guide for more information.

The Key File name is automatically generated by NightProbe. If you wish to change the file name, specify or select a file which is visible or can be created relative to the target system.

Session File

The Session File is a NightTrace configuration file which contains all the data from the General tab. It is created when recording is started.

If the Launch NightTrace checkbox is checked, NightTrace is automatically launched when recording is started and the session file is included as a parameter as part of that launch.

If you wish to change the file name, specify or select a file which is visible or can be created relative to the host system.

Timing Source

This option list allows you to select the timing source use for NightTrace events It is unrelated to the Timing method selected for NightProbe recording, as described in "Recording Timing Methods" on page 12-3.

Target systems with a Real-time Clock and Interrupt Module (RCIM) may wish to select the RCIM as the timing source for NightTrace events.

Program

The Program recording destination allows you to stream recorded samples to a user application. The specified user application is assumed to be using the NightProbe DataStream API to decode the stream of recorded data.

When recording is started, the specified user application is launched with its `stdin` file description is associated with a pipe or socket, whose initiating end is attached to the NightProbe server.

The Program Destination dialog is launched when the To Program… method is selected from a Record Destinations menu, or when you double-click the description of the NightTrace item in the Recording Destinations tree of the Configuration page.

Program Destination Dialog

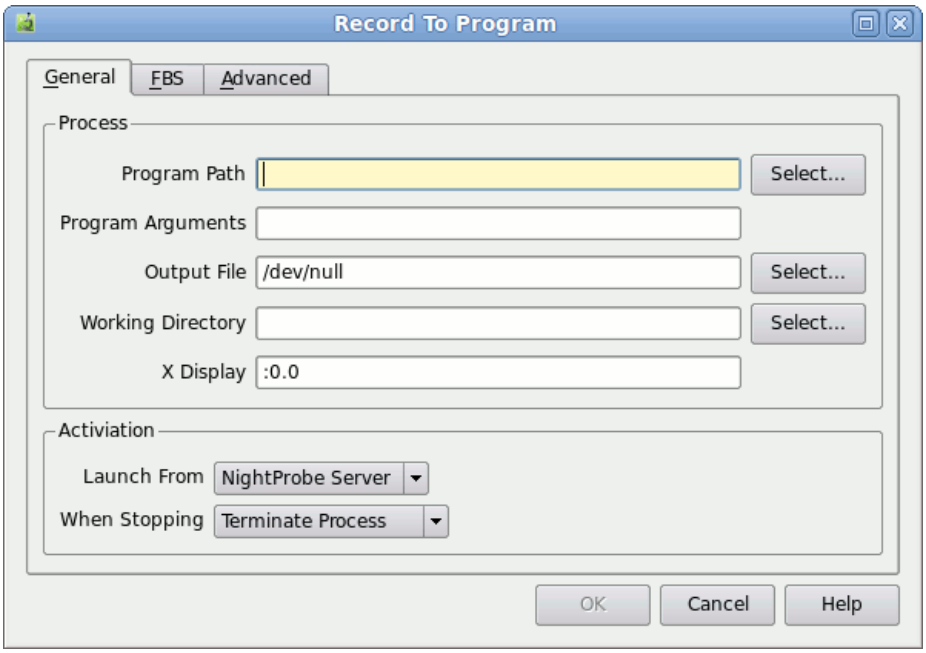


Figure 12-9. Program Destination Dialog

The dialog provides for program selection with three tabs.

General Tab

The General tab allows you to specify the path and invocation of your user application and whether it should be launched from the host system or the target system.

Program Path

The Program Path is a required field. It must specify the path to the application executable file.

Pressing the Select... button to the right of the Program Path field launches a standard file selection dialog. You can browse and select files from this dialog. Files shown in this dialog are accessible from the host system.

The Program Path ultimately specified must be an executable file which is accessible from the host or target system, depending on the setting of the Launch From option list below.

Program Arguments

This optional field can be used to pass arguments to the program when it is launched.

Output File

This field defaults to /dev/null, which means that no output from the program on the stdout file descriptor will be visible.

You can change this field to specify a file which will be accessed or created as necessary during program launch. The file specified will be accessed or created relative to the host or target system, depending on the setting of the Launch From option list below.

Working Directory

If left unspecified, the working directory of the program will be either the current working direction, if the Launch From option list is set to NightProbe GUI, or from the target system if the setting is NightProbe Server.

Alternatively, you can set the working directory, taking into account the access rules as defined in the previous paragraph.

X Display

By default, the value of the current DISPLAY variable is shown in the field. You may change this to a different display device.

Launch From

This option list controls the system from which the program will be launched.

The NightProbe Server selection will cause the program to be launched from the target system, whereas the NightProbe GUI selection will cause the program to be launched from the host system.

When Stopping

This option list controls the action to be taken when recording is stopped.

If Terminate Process is selected, NightProbe will forcibly terminate the process when recording is stopped. Otherwise, NightProbe will allow the process to continue execution.

FBS Tab

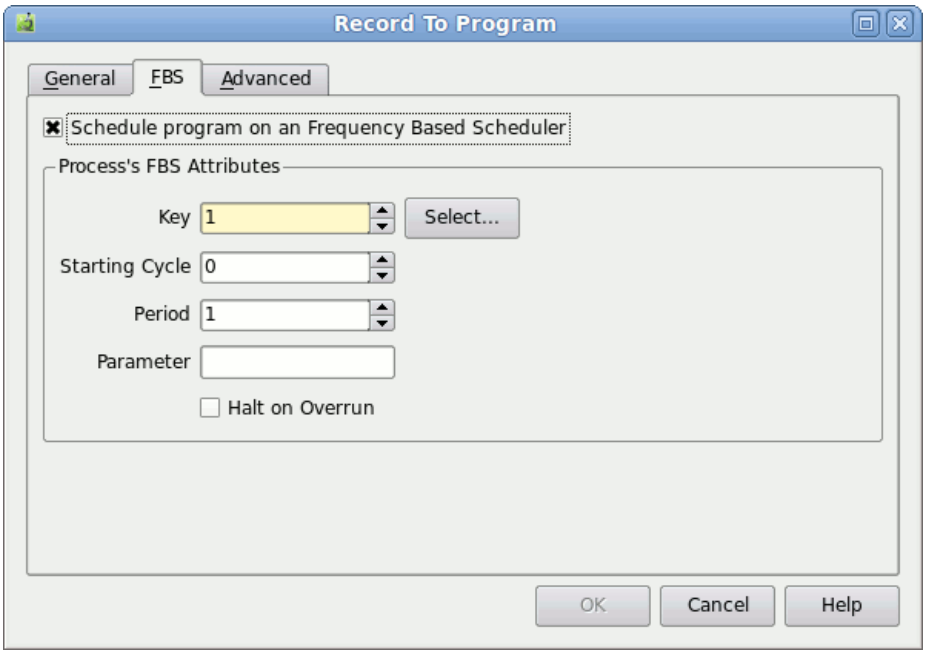


Figure 12-10. Program Destination Dialog FBS Tab

This tab allows you to schedule your user application on an FBS.

This is especially useful if you are using the FBS as the recording timing method, as described in “Frequency Based Scheduler Timing” on page 12-4.

To use FBS Scheduling effectively, your program must call `fbswait()` in a loop. See **`fbswait(3)`** for more information.

Check the Schedule program on a Frequency Based Scheduler box to activate the attribute area of the tab.

Key

This required field is used to identify the FBS on which your program will be scheduled.

The FBS specified by the value in the Key field does not have to exist when you close the dialog, but it must exist when recording is started.

Pressing the Select… button to the right of the Key field launches a standard NightProbe selection dialog which allows you to choose a key from the list of Frequency Based Scheduler that are currently active on the host or target system, depending on the setting of the Launch From option list described above.

Starting Cycle

The Starting Cycle field defaults to zero. This controls the first cycle within a frame when your program will execute.

Period

The Period field defaults to one. This field controls the number of period of execution after the cycle defined in the Starting Cycle field. A value of one indicates that execution will occur for every cycle in the frame once the Starting Cycle occurs, inclusive. A value of two implies every other cycle, etc.

Parameter

This option field can be changed to specify an integer value which will be passed to the program when it is launched. The value is not passed as an argument (i.e. argv), but through the FBS auxiliary data. You can retrieve this value from your application by using **`sched_pgmqry(3)`**.

Halt on Overrun

When checked, this causes the entire FBS to stop if your program overruns its prescribed scheduling deadlines as defined by the Starting Cycle and Period above.

Advanced Tab

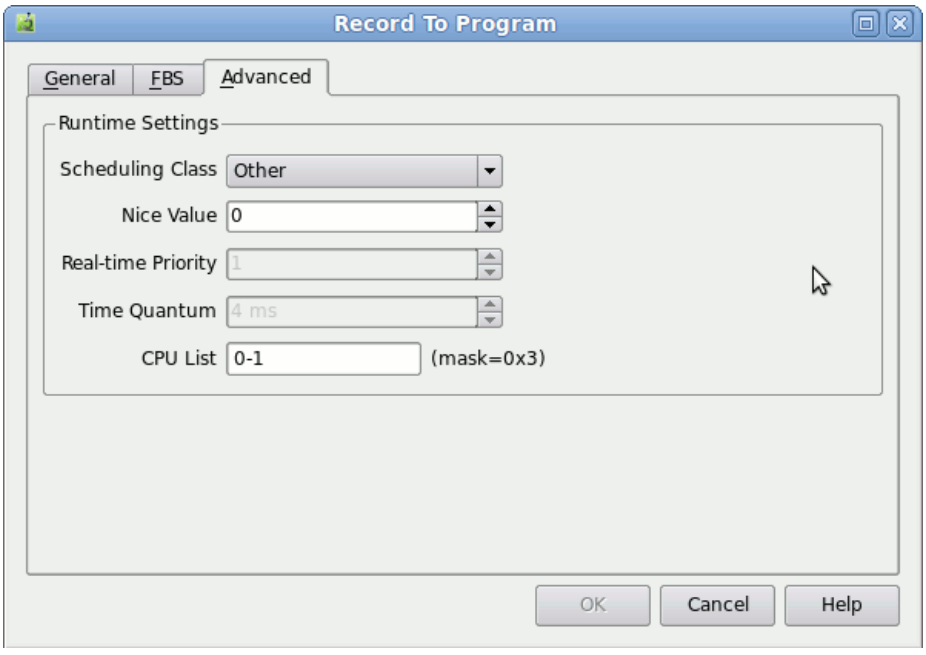


Figure 12-11. Program Destination Dialog Advanced Tab

The Advanced tab allows you to control scheduling the scheduling attributes and CPU affinity of your program. See "Runtime Settings" on page 3-2 for a description of these attributes and the selection mechanisms.

Playback

Playback is the activity of loading a previously-recorded NightProbe data file for viewing within the NightProbe GUI.

Selecting the Select Playback File… option from the Record menu or double-clicking the Playback item in the Configuration page launches a standard file selection dialog.

You can browse and select such a data file or type in an explicit filename path. Files shown in the dialog are accessible from the host system.

The ultimate file specified for the playback file must be accessible from the host system.

Once a playback file is selected, you can view the data items associated with them using the List, Table, or Graph panels by selecting View Playback Samples from the Mode option list in those panels.

13
NightProbe API

NightProbe provides two sets of APIs for use in applications. The NightProbe Datastream API (see “NightProbe Datastream API” on page 13-1) provides a basic interface to the data produced by NightProbe. These data structures and functions allow the user to process the data sampled by NightProbe either in real-time or via a previously recorded file.

The NightProbe Trigger API (see “NightProbe Trigger API” on page 13-20) provides an interface to the NightProbe Trigger Server Queue allowing an application to control the sampling of data by NightProbe in a synchronized manner.

NightProbe Datastream API

The NightProbe Datastream Application Programming Interface provides a basic interface to the data produced by NightProbe.

This API can be used with data recording output generated by NightProbe using the File and Program recording destinations (see “File” on page 12-6 and “Program” on page 12-10).

The following sections describe the general format of the data generated by NightProbe sampling (see “NightProbe Data Format” on page 13-1) as well as the data structures and functions (see “Data Structures” on page 13-2 and “Functions” on page 13-6) that comprise the NightProbe Datastream API.

Sample programs using the NightProbe Datastream API are also provided (see “Sample Programs” on page 13-13).

NightProbe Data Format

This section describes the general format of data generated by NightProbe sampling. This format is used when you select either the File or Program recording destinations (see “File” on page 12-6 and “Program” on page 12-10).

The NightProbe Datastream API allows you to open a previously recorded file and decode the individual data items, or to consume the data as it is being generated by NightProbe. In either case, the incoming data is referred to as a *datastream*.

When the File recording destination is selected, NightProbe writes the data to a file, and a user program opens that file and passes the file descriptor to the NightProbe Datastream API calls to decode the data.

When the Program recording destination is selected, a user program is launched from NightProbe and its **`stdin`** file descriptor is set to the read end of a socket or pipe. The

user program then passes the **`stdin`** file descriptor to the NightProbe Datastream API calls to decode the data.

The following diagram describes the general layout of a datastream:

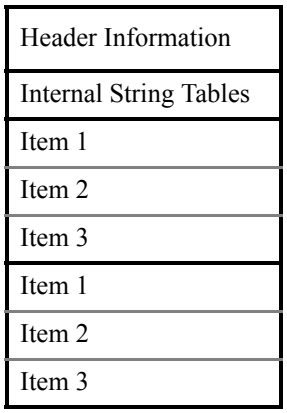


Figure 13-1. Structure of NightProbe datastream

The NightProbe Datastream API functions provide a simple interface for obtaining information about the programs from which the data was obtained, information about the variables within those programs, and individual data samples. See "Functions" on page 13-6 for more information about these functions.

Data Structures

The following data structures are part of the NightProbe Datastream Application Programming Interface:

- `np_endian_type` (see "np_endian_type" on page 13-2)
- `np_handle` (see "np_handle" on page 13-3)
- `np_header` (see "np_header" on page 13-3)
- `np_item` (see "np_item" on page 13-4)
- `np_process` (see "np_process" on page 13-4)
- `np_type` (see "np_type" on page 13-5)

See "Functions" on page 13-6 for information about the functions available in the NightProbe Datastream API.

np_endian_type

`np_endian_type` is used to represent the endian order of the NightProbe data and of the host system.

```
typedef enum np_endian_type_code {
   NP_LITTLE_ENDIAN,      /* Addresses designate the Least
                             Significant Byte of a value.  */
   NP_BIG_ENDIAN,         /* Addresses designate the Most
                             Significant Byte of a value.  */
} np_endian_type;
```

See "Data Structures" on page 13-2 for other data structures included in the NightProbe Datastream API.

np_handle

`np_handle` is a unique integer value denoting a single NightProbe datastream.

```
typedef int np_handle;
```

See "Data Structures" on page 13-2 for other data structures included in the NightProbe Datastream API.

np_header

`np_header` is a structure which is used to describe the processes and items from which data in the NightProbe datastream originates. This information is needed to interpret the sample data returned by `np_read()`.

```
typedef struct {
   int                num_items;
   int                num_processes;
   int                sample_size;
   np_endian_type     sample_endian;
   np_process       * processes;
   np_item          * items;
   const char       * system_name;
   time_t             start_time;
   unsigned long long sample_rate_nsec;
} np_header;
```

Note:

The `sample_rate_nsec` field will be zero if the recording timer selected was On Demand, Frequency Based Scheduler, or Application Trigger. See "Recording" on page 5-7 for more information on recording.

See "Data Structures" on page 13-2 for other data structures included in the NightProbe Datastream API.

SEE ALSO

- "np_endian_type" on page 13-2
- "np_process" on page 13-4

- "np_item" on page 13-4
- "np_read()" on page 13-8

np_item

`np_item` is a structure that describes a single data item present in the NightProbe datastream.

```
typedef struct np_item np_item;
struct np_item {
   char        * name;       // name of item
   unsigned      bit_offset; // bit offset within each sample
   unsigned      bit_size;   // atomic size in bits
   unsigned      count;      // number of atoms
   np_type       type;       // data type
   unsigned      event_id;   // NightTrace event ID for item
   np_process  * process;    // process info
   np_item     * link;       // next item pointer
};
```

The item occupies `count` instances of `bit_size` bits beginning at `bit_offset` within the sample.

See "Data Structures" on page 13-2 for other data structures included in the NightProbe Datastream API.

SEE ALSO

- "np_process" on page 13-4
- "np_type" on page 13-5

np_process

`np_process` is a structure which contains information about a particular process from which real-time data originates.

```
typedef struct np_process np_process;
struct np_process {
   int          pid;
   char       * name;
   np_process * link;
};
```

See "Data Structures" on page 13-2 for other data structures included in the NightProbe Datastream API.

np_type

The `np_type` enumeration in the `np_item` structure may be used (along with `size`) in order to determine an appropriate format for displaying a value from the sample buffer.

```
typedef enum np_type_code {
   NP_VOID_TYPE,               /* void                          */
   NP_CHAR_TYPE,               /* signed byte character         */
   NP_UNSIGNED_CHAR_TYPE,     /* unsigned byte character      */
   NP_SHORT_INT_TYPE,          /* signed short int              */
   NP_UNSIGNED_SHORT_INT_TYPE,/* unsigned short int           */
   NP_INT_TYPE,                /* signed int                    */
   NP_UNSIGNED_INT_TYPE,      /* unsigned int                  */
   NP_LONG_INT_TYPE,           /* signed long int               */
   NP_UNSIGNED_LONG_INT_TYPE, /* unsigned long int            */
   NP_FLOAT_TYPE,              /* single precision float        */
   NP_DOUBLE_TYPE,             /* double precision float        */
   NP_LONG_DOUBLE_TYPE,       /* long double precision float  */
   NP_SHORT_LOGICAL_TYPE,     /* short logical (boolean)      */
   NP_LOGICAL_TYPE,            /* logical (boolean)             */
   NP_COMPLEX_TYPE,            /* Fortran complex type          */
   NP_DOUBLE_COMPLEX_TYPE,    /* Fortran double complex        */
   NP_POINTER_TYPE,            /* Pointer to unspecified type  */
   NP_FIXED_POINT_TYPE,       /* fixed point                   */
   NP_EXCEPTION_TYPE,          /* exception                     */
   NP_STRUCTURE_BYTES          /* structure bytes               */
}  np_type;
```

See “Data Structures” on page 13-2 for other data structures included in the NightProbe Datastream API.

SEE ALSO

- “np_item” on page 13-4

Functions

The following functions are part of the NightProbe Datastream API:

- `np_open` (see "np_open()" on page 13-6)
- `np_avail` (see "np_avail()" on page 13-7)
- `np_read` (see "np_read()" on page 13-8)
- `np_close` (see "np_close()" on page 13-10)
- `np_format` (see "np_format()" on page 13-11)
- `np_error` (see "np_error()" on page 13-12)

np_open()

`np_open()` is used to open and initialize an input NightProbe datastream on an open file descriptor.

SYNTAX

```
int np_open (int fd, np_header *header, np_handle *handle);
```

PARAMETERS

fd

file descriptor associated with the file created using the File recording destination (see "File" on page 12-6) which contains the data recording output

If data recording output is streamed directly from NightProbe using the Program recording destination (see "Program" on page 12-10), *fd* should be set to the **`stdin`** file descriptor, `0`.

header

structure to contain information describing the processes from which the NightProbe data originates, as well as the number, names and types of the items appearing in the NightProbe datastream

handle

a unique value denoting the open NightProbe datastream

RETURN VALUES

0

indicates successful completion

-1

indicates a failure

handle contains a value which may be passed to `np_error()` to obtain a diagnostic message describing the failure

IMPORTANT

If you call `np_open()` on a data file that was produced on a target architecture having a different endian order from the host (e.g. big-endian data file/little-endian host, or vice versa), all data returned to you by `np_read()` will be of the data file's orientation. In order to obtain meaningful information about the probe samples in the file in such situations, you must first convert the format of the data in the sample buffer to the proper endian format for the host before calling `np_format()`.

See "Functions" on page 13-6 for other functions included in the NightProbe Datastream API.

SEE ALSO

- "np_header" on page 13-3
- "np_handle" on page 13-3
- "np_read()" on page 13-8
- "np_error()" on page 13-12

np_avail()

`np_avail()` is used to check a NightProbe datastream for available data items.

SYNTAX

```
int np_avail (np_handle handle);
```

PARAMETERS

handle

value (obtained from `np_open()`) which identifies the NightProbe datastream of interest

RETURN VALUES

0

if data is not currently available on the NightProbe datastream and `np_read()` would block

> 0

if data is currently available for `np_read()`

-1

indicates a failure

If *handle* is non-zero, `np_error()` may be called to obtain a diagnostic message describing the failure.

See "Functions" on page 13-6 for other functions included in the NightProbe Datastream API.

SEE ALSO

- "np_open()" on page 13-6
- "np_read()" on page 13-8
- "np_error()" on page 13-12

np_read()

Read a single data sample from the NightProbe datastream.

SYNTAX

```
int np_read (np_handle handle, void *sample);
```

PARAMETERS

handle

value (obtained from `np_open()`) which identifies the NightProbe datastream of interest

sample

upon successful completion, *sample* contains the NightProbe entire sample data

To get at individual data items, use the information from the `np_header` structure returned from `np_open()`. For each item, retrieve the appropriate number of bytes (as specified by `size` in the `np_item` structure associated

with that item) offset from the beginning of the sample buffer (as specified by `offset` in the `np_item` structure associated with that item)

See "Sample Programs" on page 13-13 for examples.

RETURN VALUES

> 0

value represents the number of bytes in the sample obtained

0

if end-of-file (EOF) was encountered on the NightProbe datastream

-1

indicates a failure

If *handle* is non-zero, `np_error()` may be called to obtain a diagnostic message describing the failure.

IMPORTANT

If you call `np_open()` on a data file that was produced on a target architecture having a different endian order from the host (e.g. big-endian data file/little-endian host, or vice versa), all data returned to you by `np_read()` will be of the data file's orientation. In order to obtain meaningful information about the probe samples in the file in such situations, you must first convert the format of the data in the sample buffer to the proper endian format for the host before calling `np_format()`.

NOTE

`np_read()` will block waiting for data to become available on the datastream if data is not immediately available. If time is critical and a blocking read is not desired, use `np_avail()` to first check if data is available prior to reading.

See "Functions" on page 13-6 for other functions included in the NightProbe Datastream API.

SEE ALSO

- "np_header" on page 13-3
- "np_item" on page 13-4
- "np_open()" on page 13-6

- "np_avail()" on page 13-7
- "np_error()" on page 13-12
- "np_format()" on page 13-11

np_close()

Close a NightProbe datastream.

SYNTAX

```
void np_close (np_handle handle);
```

PARAMETERS

handle

value (obtained from `np_open()`) which identifies the NightProbe datastream of interest

Upon completion, *handle* no longer refers to an open NightProbe datastream.

NOTE

No further diagnostic messages are available from `np_error()` after calling `np_close()`.

Furthermore, the file descriptor passed to `np_open()` remains open after the `np_close()` call. The NightProbe datastream is logically closed, but the associated file descriptor remains open. **`close(2)`** must be called to close the file descriptor as well, if desired.

See "Functions" on page 13-6 for other functions included in the NightProbe Datastream API.

SEE ALSO

- "np_open()" on page 13-6
- "np_error()" on page 13-12

np_format()

Return an allocated string representation of the specified `np_item` value from the given sample. The caller is responsible for freeing the memory associated with the returned string once it is no longer needed.

SYNTAX

```
char * np_format (
   np_handle  handle,
   np_item  * i,
   void     * sample,
   int        which);
```

PARAMETERS

handle

value (obtained from `np_open()`) which identifies the NightProbe datastream of interest

i

a pointer to an `np_item` descriptor denoting a single item within a data sample. The `np_item` is part of the `np_header` obtained from the previous call to `np_open()`.

sample

a pointer to the contents of a single sample obtained from a call to `np_read()`

which

for items with multiple atoms (i.e. *i->count* > 1), *which* determines the atom to be formatted. A *which* value of 1 indicates the first atom for the item.

RETURN VALUES

non-NULL

value represents a textual representation of the specified data in a format based on the `np_type` of the item

NULL

a parameter was invalid, or the NightProbe Datastream API was unable to allocate memory for the result. np_error may be called to obtain a diagnostic message describing the failure.

See “Functions” on page 13-6 for other functions included in the NightProbe Datastream API.

SEE ALSO

- "np_header" on page 13-3
- "np_item" on page 13-4
- "np_error()" on page 13-12

np_host_endian()

Returns the `np_endian_type` value denoting the endian order of the host system.

SYNTAX

```
np_endian_type np_host_endian (void);
```

See "Functions" on page 13-6 for other functions included in the NightProbe Datastream API.

SEE ALSO

- "np_endian_type" on page 13-2

np_error()

Return a diagnostic message describing the most recent failure encountered by a prior call to `np_open()`, `np_avail()`, or `np_read()`.

SYNTAX

```
char * np_error (np_handle handle);
```

PARAMETERS

handle

value (obtained from `np_open()`) which identifies the NightProbe datastream of interest

See "Functions" on page 13-6 for other functions included in the NightProbe Datastream API.

SEE ALSO

- "np_open()" on page 13-6
- "np_avail()" on page 13-7

- "np_read()" on page 13-8

Sample Programs

The following programs are given as examples of how to use the NightProbe Datastream API (see "NightProbe Datastream API" on page 13-1).

program_output_test.c

This program uses the NightProbe Datastream API to process a NightProbe data sample.

program_output_fbs_test.c

This program uses the NightProbe Datastream API to process a NightProbe data sample but uses a frequency-based scheduler in order to coordinate data recording activity so as to minimize interference with the probed application.

program_output_test.c

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>
#include <string.h>
#include <nprobe.h>



int    cycles   = 0;
int    overruns = 0;
char * sample;


// Perform the work of consuming a single Data Recording sample from NightProbe.
//
int
work (FILE * ofile, np_handle h, np_header * hdr) {
   np_item * i;
   int status;
   int which;

   // Read one sample, which may contain data for multiple processes
   // and variables.
   //
   status = np_read (h, sample);
   if (status <= 0)  {
      return status;
   }

   cycles++;
```

```
   fprintf (ofile, "Sample %d\n", cycles);
   for (i = hdr->items; i; i = i->link) {
      char buffer [1024];
      sprintf (buffer, "item: %s:", i->name);
      fprintf (ofile, "%-30s", buffer);        // Nice formatting :-)

      // Display the value of each item.
      // For arrays, format each individual item.
      //
      for (which = 1; which <= i->count; ++which) {
         char *   image = np_format (h, i, sample, which);

         if (image != NULL) {
            fprintf (ofile, " %s", image);
         } else {
            fprintf (ofile, "\n<error: %s>\n", np_error (h));
            return -1;
         }

         free (image);
      }
      fprintf (ofile, "\n");
   }
   fflush (ofile);

   return 1;
}




int
main (int argc, char *argv[])
{
   np_handle h;
   np_header hdr;
   np_process * p;
   np_item * i;
   int fd;
   int status;
   FILE *   ofile = stdout;


   fd = 0; // stdin

   status = np_open (fd, &hdr, &h);
   if (status) {
      fprintf (stderr, "%s\n", np_error(h));
      exit(1);
   }

   sample = (char *) malloc(hdr.sample_size);
   if (sample == NULL) {
      fprintf (stderr, "insufficient memory to allocate sample buffer\n");
      exit(1);
   }

   for (p = hdr.processes; p; p = p->link) {
```

```
      if (p->pid >= 0) {
         fprintf (ofile, "process: %s (%d)\n", p->name, p->pid);
      } else {
         fprintf (ofile, "resource: %s (%s)\n", p->name, p->label);
      }
   }
   fprintf (ofile, "\n");

   for (i = hdr.items; i; i = i->link) {
      fprintf (ofile, "item: %s (%s), size=%d bits, count=%d, type=%d\n",
              i->name, i->process->name, i->bit_size, i->count, i->type);
   }
   fprintf (ofile, "\n");

   for (;;) {
      status = work (ofile, h, &hdr);
      if (status <= 0) break;
   }

   fprintf (ofile, "Data Recording done: %d cycles fired, %d overruns\n",
            cycles, overruns);

   if (ofile != stdout) {
      fclose (ofile);
   }

   if (status < 0) {
      fprintf (stderr, "%s\n", np_error(h));
   }

   np_close (h);

   // At this point, file descriptor 0 remains open, but is no
   // longer a NightProbe Data File/Stream.
}
```

program_output_fbs_test.c

NOTE

This program requires the Frequence Based Scheduler module which is not available on all systems. See "Kernel Dependencies" on page B-1 for more information.

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>
#include <string.h>
#include <nprobe.h>
#ifdef linux
#include <fbsched.h>
#else
#include <fbslib.h>
#endif



int    cycles   = 0;
int    overruns = 0;
char * sample;


// Perform the work of consuming a single Data Recording sample from NightProbe.
//
// This function is called once every time the fbswait() system call returns
// successfully.
//
int
work (FILE * ofile, np_handle h, np_header * hdr) {
   np_item * i;
   int status;
   int n;
   char * ptr;


   // 0, 1, or >1 trigger events may have occurred since we last work()ed.
   //
   // Check whether data is available, and process it as long as new
   // data is already available within this work cycle.
   //
   // A more sophisticated program would limit the number of np_read() calls
   // per work cycle based upon how much time is left in the current cycle.
   //
   while (np_avail (h)) {

      // Read one sample, which may contain data for multiple processes
      // and variables.
      //
      status = np_read (h, sample);
      if (status <= 0)  {
```

```
            return status;
         }

         cycles++;

         fprintf (ofile, "\n");
         for (i = hdr->items; i; i = i->link) {
            fprintf (ofile, "item: %25s :", i->name);

            // Calculate the address of the item within the sample buffer.
            // This formula calculates the address of the first byte of
            // data corresponding to the item.
            //
            // The first bit is at i->bit_offset % 8 within that byte.
            //
            ptr = sample + (i->bit_offset/8);

            for (n = 0; n < i->count; ++n) {

               // Note that this simple example assumes type/format from
               // the size of the data item.  The 'i->type' field should
               // be taken into account for a more accurate means of
               // determining the data format.
               //
               if (i->bit_offset % 8) {
                  fprintf (ofile, " <size=%d bits, offset=%d bits>",
                                  i->bit_size, i->bit_offset % 8);
               } else {
                  switch (i->bit_size) {
                  case 8:
                     fprintf (ofile, " 0x%1x", ((char*)ptr)[n]);
                     break;
                  case 16:
                     fprintf (ofile, " 0x%1x", ((unsigned short*)ptr)[n]);
                     break;
                  case 32:
                     fprintf (ofile, " 0x%1x", ((unsigned*)ptr)[n]);
                     break;
                  case 64:
                     fprintf (ofile, " %lf", ((double*)ptr)[n]);
                     break;
                  default:
                     fprintf (ofile, " <size=%d bits>", i->bit_size);
                  }
               }
            }
            fprintf (ofile, "\n");
         }
         fflush (ofile);
      }

      return 1;
   }



int
main (int argc, char *argv[])
```

```
{
   np_handle h;
   np_header hdr;
   np_process * p;
   np_item * i;
   int fd;
   int status;
   FILE *   ofile = stdout;


#ifdef linux
   if (!fbsavail()) {
      fprintf (ofile, "fbsavail() reports No FBS on this target\n");
      fclose (ofile);
      exit (1);
   }
#endif



   fd = 0; // stdin

   status = np_open (fd, &hdr, &h);
   if (status) {
      fprintf (stderr, "%s\n", np_error(h));
      exit(1);
   }

   sample = (char *) malloc(hdr.sample_size);
   if (sample == NULL) {
      fprintf (stderr, "insufficient memory to allocate sample buffer\n");
      exit(1);
   }

   for (p = hdr.processes; p; p = p->link) {
      if (p->pid >= 0) {
         fprintf (ofile, "process: %s (%d)\n", p->name, p->pid);
      } else {
         fprintf (ofile, "resource: %s (%s)\n", p->name, p->label);
      }
   }
   fprintf (ofile, "\n");

   for (i = hdr.items; i; i = i->link) {
      fprintf (ofile, "item: %s (%s), size=%d bits, count=%d, type=%d\n",
               i->name, i->process->name, i->bit_size, i->count, i->type);
   }
   fprintf (ofile, "\n");

   for (;;) {

      // We wait till the Concurrent FBS wakes us up at the time which is
      // appropriate for performing data recording.  This program must be
      // scheduled on the FBS, but doing so allows the scheduling of data
      // recording activity at a time that won't disturb other critical
      // application cycles.
      //
      int stat = fbswait();
```

```
      // Diagnose the return value from fbswait()
      if (stat < 0) {
         switch (stat) {
         case -1:
            if (errno == ENOENT) {
               fprintf (ofile,
                        "%s has been removed from the scheduler\n", argv[0]);
            } else {
               fprintf (ofile, "fbs_wait(3) failed on cycle %d: "
                            "errno is %d (%s)\n",
                        cycles, errno, strerror (errno));
            }
            break;
         default:
            fprintf (ofile, "fbs_wait(3) returned unexpected %d on cycle %d\n",
                     stat, cycles);
            break;
         }

         break;
      }

      switch (stat) {
      case 0:
         break;
      case 1:
         fprintf (ofile, "fbstrig(2) caused sim to fire: cycle %d\n", cycles);
         break;
      case 2:
         fprintf (ofile, "soft overrun %d detected on cycle %d\n",
                  ++overruns, cycles);
         break;
      }

      status = work (ofile, h, &hdr);
      if (status <= 0) {
         break;
      }
   }

   fprintf (ofile, "Data Recording done: %d cycles fired, %d overruns\n",
            cycles, overruns);

   if (ofile != stdout) {
      fclose (ofile);
   }

   if (status < 0) {
      fprintf (stderr, "%s\n", np_error(h));
   }

   np_close (h);

   // At this point, file descriptor 0 remains open, but is no
   // longer a NightProbe Data File/Stream.
}
```

NightProbe Trigger API

The NightProbe Trigger API provides an interface to the NightProbe Trigger Server Queue allowing an application (the NightProbe Trigger Client) to cause the NightProbe Server to sample data in a synchronized manner.

The name of the Trigger Server Queue can be specified to NightProbe using the Set Trigger Timer dialog (see "Application Trigger Dialog" on page 12-5).

The following sections describe the data structures and functions (see "Data Structures" on page 13-20 and "Functions" on page 13-6) that comprise the NightProbe Trigger API.

A sample program using the NightProbe Trigger API is also provided (see "Sample Program" on page 13-25).

Data Structures

The following data structure is part of the NightProbe Trigger Application Programming Interface:

- `np_trigger_handle` (see "np_trigger_handle" on page 13-20)

See "Functions" on page 13-21 for a list of functions included in the NightProbe Trigger API.

np_trigger_handle

`np_trigger_handle` is a unique integer value denoting a connection to a NightProbe Trigger Server Queue. The `np_trigger_handle` may be used to request sampling events and/or obtain further information about a failure.

```
typedef int np_trigger_handle;
```

Functions

The following functions are part of the NightProbe Trigger API:

- `np_trigger_open` (see “np_trigger_open()” on page 13-21)
- `np_trigger` (see “np_trigger()” on page 13-22)
- `np_trigger_close` (see “np_trigger_close()” on page 13-23)
- `np_trigger_error` (see “np_trigger_error()” on page 13-24)

np_trigger_open()

`np_trigger_open()` is used to open a connection to a NightProbe Trigger Server Queue, allowing the caller to control when NightProbe samples are captured.

SYNTAX

```
int np_trigger_open (char *name, np_trigger_handle *sampler);
```

PARAMETERS

name

a unique identifier for the NightProbe Trigger Server Queue on the target system. This is the same name assigned to the trigger in the Set Trigger Timer dialog (see “Application Trigger Dialog” on page 12-5)

name must be a legal filename containing no '/' characters. The length of name is restricted to at most (`MAXNAMELEN`-15).

sampler

returns an `np_trigger_handle` designating the active NightProbe Trigger Sampler Queue connection.

RETURN VALUES

0

indicates successful completion

-1

indicates a failure

`np_trigger_error()` may be used to obtain further information about the reason for the failure.

See “Functions” on page 13-21 for other functions included in the NightProbe Trigger API.

SEE ALSO

- "np_trigger_handle" on page 13-20
- "np_trigger_error()" on page 13-24

np_trigger()

`np_trigger()` requests a sample be taken by the NightProbe sampler. If there is an active NightProbe Trigger Server connected to the queue, `np_trigger()` sends a trigger request to the NightProbe Trigger Server Queue and returns.

SYNTAX

```
int np_trigger (np_trigger_handle sampler);
```

PARAMETERS

sampler

an `np_trigger_handle` specifying the active NightProbe Trigger Sampler Queue connection

RETURN VALUES

0

indicates successful completion

1

no NightProbe Trigger Server was connected.

This is not necessarily an error, but the result is provided so that the NightProbe Trigger API Client can determine when a server disconnects.

Note that the server may later re-connect, and subsequent `np_trigger()` calls will again return 0.

-1

indicates a failure

`np_trigger_error()` may be used to obtain further information about the reason for the failure.

See "Functions" on page 13-21 for other functions included in the NightProbe Trigger API.

SEE ALSO

- "np_trigger_handle" on page 13-20

- “np_trigger_open()” on page 13-21
- “np_trigger_error()” on page 13-24

np_trigger_close()

`np_trigger_close()` is used to disconnect from the NightProbe Trigger Server Queue.

SYNTAX

```
int np_trigger_close (np_trigger_handle sampler);
```

PARAMETERS

sampler

an `np_trigger_handle` specifying the active NightProbe Trigger Sampler Queue connection to be closed

sampler is no longer valid after this call.

RETURN VALUES

0

indicates successful completion

-1

indicates a failure

`np_trigger_error()` may be used to obtain further information about the reason for the failure.

See “Functions” on page 13-21 for other functions included in the NightProbe Trigger API.

SEE ALSO

- “np_trigger_handle” on page 13-20
- “np_trigger_open()” on page 13-21
- “np_trigger_error()” on page 13-24

np_trigger_error()

`np_trigger_error()` returns an error message describing the most recent failure detected by the NightProbe Trigger API functions.

SYNTAX

```
char *np_trigger_error (np_trigger_handle sampler);
```

PARAMETERS

sampler

an `np_trigger_handle` specifying the active NightProbe Trigger Sampler Queue connection

RETURN VALUES

Returns the error message describing the most recent failure detected by the NightProbe Trigger API functions.

Returns '`No error`' if no errors have occurred.

See "Functions" on page 13-21 for other functions included in the NightProbe Trigger API.

SEE ALSO

- "np_trigger_handle" on page 13-20
- "np_trigger_open()" on page 13-21

Sample Program

The following program is given as an example of how to use the NightProbe Trigger API (see "NightProbe Trigger API" on page 13-20).

nprobe_trigger_test.c

This program uses the NightProbe Trigger API.

nprobe_trigger_test.c

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <fcntl.h>

#include "nprobe_trigger.h"

int
main (int argc, char * argv[])
{
   np_trigger_handle h;
   int status;
   int i;

   if (argc != 2) {
      fprintf (stderr, "Usage: nprobe_trigger_test trigger_name\n");
      exit(1);
   }

   printf ("Trigger %s: connecting...\n", argv[1]);
   status = np_trigger_open (argv[1], &h);
   if (status) {
      fprintf (stderr, "%s\n", np_trigger_error(h));
      exit(1);
   }
   printf ("Trigger %s: conected to trigger server queue\n", argv[1]);

   for (i = 0; i >= 0; i++) {
      sleep (1);
      if ((status = np_trigger (h)) < 0) {
         fprintf (stderr, "%s\n", np_trigger_error(h));
         exit(1);
      } else if (status == 0) {
         printf ("Trigger %s: triggered %d\n", argv[1], i);
      } else {
         printf ("Trigger %s: dropped %d\n", argv[1], i);
      }
   }

   np_trigger_close (h);
   printf ("Trigger %s: closed.\n", argv[1]);

   exit (0);
}
```

A
NightStar LX Licensing

NightStar LX uses the NightStar License Manager (NSLM) to control access to the NightStar LX tools.

License installation requires a licence key provided by Concurrent (see "License Keys" on page A-1).The NightStar LX tools request a licence (see "License Requests" on page A-2) from a license server (see "License Server" on page A-2).

Two license modes are available, fixed and floating, depending on which product option you purchased. Fixed licenses can only be served to NightStar LX users from the local system. Floating licenses may be served to any NightStar LX user on any system on a network.

Tools are licensed per system, per concurrent user. A single license is shared among any or all of the NightStar LX tools for a particular user on a particular system. The intent is to allow *n* developers to fully utilize all the tools at the same time while only requiring *n* licenses. When operating the tools in remote mode, where a tool is launched on a local system but is interacting with a remote system, licenses are required only from the host system.

You can obtain a license report which lists all licenses installed on the local system, current usage, and expiration date for demo licenses (see "License Reports" on page A-3).

The default configuration includes a strict firewall which interferes with floating licenses. See "Firewall Configuration for Floating Licenses" on page A-3 for information on handling such configurations.

See "License Support" on page A-6 for information on contacting Concurrent for additional assistance with licensing issues.

License Keys

Licenses are granted to specific systems to be served to either local or remote clients, depending on the license model, fixed or floating.

License installation requires a license key provided by Concurrent. To obtain a license key, you must provide your system identification code. The system identification code is generated by the **`nslm_admin`** utility:

```
nslm_admin --code
```

System identification codes are dependent on system configurations. Reinstalling Linux on a system or replacing network devices may require you to obtain new license keys.

To obtain a license key, use the following URL and click on the *Licenses* link:

http://www.ccur.com/NightStarKeys

Provide the requested information, including the system identification code. Your license key will be immediately emailed to you.

Install the license key using the following command:

```
nslm_admin --install=xxxx-xxxx-xxxx-xxxx-xxxx
```

where *xxxx-xxxx-xxxx-xxxx-xxxx* is the key included in the license acknowledgment email.

License Requests

By default, the NightStar LX tools request a license from the local system. If no licenses are available, they broadcast a license request on the local subnet associated with the system's hostname.

You can control the license requests for an entire system using the **/etc/nslm.config** configuration file.

By default, the **/etc/nslm.config** file contains a line similar to the following:

```
:server @default
```

The argument @default may be changed to a colon-separated list of system names, system IP addresses, or broadcast IP addresses. Licenses will be requested from each of the entities found in the list, until a license is granted or all entries in the list are exhausted.

For example, the following setting prevents broadcast requests for licenses, by only specifying the local system:

```
:server localhost
```

The following setting requests a license from **server1**, then **server2**, and then a broadcast request if those fail to serve a license:

```
:server server1:server2:192.168.1.0
```

Similarly, you can control the license requests for individual invocations of the tools using the **NSLM_SERVER** environment variable. If set, it must contain a colon-separated list of system names, system IP addresses, or broadcast IP addresses as described above. Use of the **NSLM_SERVER** environment variable takes precedence over settings defined in **/etc/nslm.config**.

License Server

The NSLM license server is automatically installed and configured to run when you install NightStar LX.

The **nslm** service is automatically activated for run levels 2, 3, 4, and 5. You can check on these settings by issuing the following command:

```
/sbin/chkconfig --list nslm
```

In rare instances, you may need to restart the license server via the following command:

```
/sbin/service nslm restart
```

See **nslm(1)** for more information.

License Reports

A license report can be obtained using the **nslm_admin** utility.

```
nslm_admin --list
```

lists all licenses installed on the local system, current usage, and expiration date (for demo licenses). Use of the **--verbose** option also lists individual clients to which licenses are currently granted.

Adding the **--broadcast** option will list this information for all servers that respond to a broadcast request on the local subnet associated with the system's hostname.

See **nslm_admin(1)** for more options and information.

Firewall Configuration for Floating Licenses

The default operating system configuration includes a strict firewall which interferes with floating licenses.

If such a system is used to serve licenses, then at least one port must be opened in its firewall to allow server requests to pass. See "Serving Licenses with a Firewall" on page A-3 for more information.

Similarly, if such a system is host to the NightStar LX tools, then at least one port must be opened in its firewall so that it can receive licenses from the license server. If this is not done, a tool requesting a floating license will not receive it and will not function properly. See "Running NightStar LX Tools with a Firewall" on page A-5 for more information.

Serving Licenses with a Firewall

Following are a few approaches for allowing the NSLM license server to serve floating licences when the system on which it is running is configured with a firewall:

- disable the firewall on the system entirely

- allow NSLM license requests from a specific system (or one of several)
- allow NSLM license requests from any system on a particular subnet (or one of several)
- allow NSLM license requests from any system

NOTE

You must be root in order to modify the firewall configuration.

To disable the firewall entirely, execute:

```
service iptables stop
```

and then remove the **/etc/sysconfig/iptables** file:

```
rm -f /etc/sysconfig/iptables
```

This option may not be as dangerous as it seems. Often, whole networks are protected with a firewall so it is not necessary for individual systems on the network to be protected further. If unsure, check with your network administrator.

For the remaining cases, a simple modification should be made to the **/etc/sysconfig/iptables** file to allow license requests on UDP port 25517 and TCP port 25517. By default, that file should contain a line like the following:

```
-A RH-Firewall-1-INPUT -j REJECT --reject-with icmp-host-prohibited
```

To allow NSLM license requests from a specific system, insert the following lines before the `REJECT` line:

```
-A RH-Firewall-1-INPUT -p udp -m udp -s system --dport 25517 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m tcp -s system --dport 25517 -j ACCEPT
```

Those lines can be repeated for multiple systems.

To allow NSLM license requests from any system on a particular subnet, insert the following lines before the `REJECT` line:

```
-A RH-Firewall-1-INPUT -p udp -m udp -s subnet/mask --dport 25517 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m tcp -s subnet/mask --dport 25517 -j ACCEPT
```

The subnet might be of a form like `192.168.1.0` and the mask could be a traditional network mask like `255.255.255.0` or a single number like `24`, which indicates the number of bits from the left that are part of the mask. For example, `192.168.1.0/255.255.255.0` and `192.168.1.0/24` are equivalent.

Those lines can be repeated for multiple subnets.

To allow NSLM license requests from any system, insert the following lines before the `REJECT` line:

```
-A RH-Firewall-1-INPUT -p udp -m udp --dport 25517 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 25517 -j ACCEPT
```

After modifying **/etc/sysconfig/iptables**, execute:

```
service iptables restart
```

Running NightStar LX Tools with a Firewall

Following are a few approaches for allowing a NightStar LX tool to receive floating licenses from a license server, when the system running the NightStar LX tool is configured with a firewall:

- disable the firewall on the requesting system entirely
- allow NSLM licenses from a specific license server (or one of several)
- allow NSLM licenses from any system on a particular subnet (or one of several)
- allow NSLM licenses from any system

NOTE

You must be root in order to modify the firewall configuration.

To disable the firewall entirely, execute:

```
service iptables stop
```

and then remove the **/etc/sysconfig/iptables** file:

```
rm -f /etc/sysconfig/iptables
```

This option may not be as dangerous as it seems. Often, whole networks are protected with a firewall so it is not necessary for individual systems on the network to be protected further. If unsure, check with your network administrator.

For the remaining cases, a simple modification should be made to the **/etc/sysconfig/iptables** file to allow license responses from UDP port 25517. By default, that file should contain a line like the following:

```
-A RH-Firewall-1-INPUT -j REJECT --reject-with icmp-host-prohibited
```

To allow NSLM licenses from a specific system running a license server, insert the following line before the `REJECT` line:

```
-A RH-Firewall-1-INPUT -p udp -m udp -s server --sport 25517 -j ACCEPT
```

That line can be repeated for multiple servers.

To allow NSLM licenses from any system running a license server on a particular subnet, insert the following before the `REJECT` line:

```
-A RH-Firewall-1-INPUT -p udp -m udp -s subnet/mask --sport 25517 -j ACCEPT
```

The subnet might be of a form like `192.168.1.0` and the mask could be a traditional network mask like `255.255.255.0` or a single number like `24`, which indicates the number of bits from the left that are part of the mask. For example, `192.168.1.0/255.255.255.0` and `192.168.1.0/24` are equivalent.

That line can be repeated for multiple subnets.

To allow NSLM licenses from any system running a license server, insert the following line before the `REJECT` line:

```
-A RH-Firewall-1-INPUT -p udp -m udp --sport 25517 -j ACCEPT
```

After modifying **`/etc/sysconfig/iptables`**, execute:

```
service iptables restart
```

License Support

For additional aid with licensing issues, contact the Concurrent Software Support Center at our toll free number 1-800-245-6453. For calls outside the continental United States, the number is 1-954-283-1822. The Software Support Center operates Monday through Friday from 8 a.m. to 5 p.m., Eastern Standard Time.

You may also submit a request for assistance at any time by using the Concurrent Computer Corporation web site at **http://www.ccur.com/isd_support_contact.asp** or by sending an email to **support@ccur.com**.

B
Kernel Dependencies

Additional features and benefits are granted the NightStar tools when operating with real-time kernels from Concurrent Computer Corporation and Novell, Inc.

RedHawk Linux

The RedHawk Linux kernel from Concurrent Computer Corporation (http://www.ccur.com) provides real-time capabilities over and above the Red Hat kernel. RedHawk Linux operates on top of a Red Hat distribution.

SUSE Linux Enterprise Real-Time (SLERT)

The SLERT kernel from Novell, Inc. (http://www.novell.com) provides real-time capabilities over and above other SUSE Linux Enterprise kernels. SLERT is also available from Concurrent Computer Corporation (http://www.ccur.com).

The following sections describe the additional functionality and capabilities of the NightStar tools when running these real-time kernels.

Advantages for NightView

The following advantages are afforded NightView when a RedHawk or SLERT kernel is running:

- Application speed conditions

  Provides “execution-speed” patches, conditions, and ignore counts.

- Signal handling

  Allows NightView to pass signals directly to a particular process, avoiding context switching.

NOTE

NightView may not operate at all on older versions of SUSE or Red Hat without a real-time kernel.

Advantages for NightTrace

The following advantage is afforded NightTrace when a RedHawk or SLERT tracing kernel is running:

- Kernel tracing

  Users of NightTrace gain the ability to obtain kernel trace data and combine that with user trace data. Kernel tracing is an incredibly powerful feature that not only provides insight into the operating system kernel but also provides useful information relating to the execution of user applications.

  The real-time kernel is provided in three flavors:

  - Tracing
  - Debug
  - Plain

  The Tracing and Debug flavors provide the features required for NightTrace kernel tracing. These kernels can be selected at boot-time from the boot-loader menu.

Advantages for NightProbe

The following advantages are afforded NightProbe when a RedHawk or SLERT kernel is running:

- Minimal intrusion

  Allows NightProbe to read and write variables without stopping the process for each sample or write operation.

- Sampling performance

  Allows NightProbe to use direct memory fetches for data sampling (as opposed to programmed I/O) which is important for high-rate data acquisition.

- Concurrent debugging/probing

  Allows NightProbe to probe programs already under the control of a debugger or another NightProbe session.

- PCI Device probing

  Allows NightProbe to probe PCI device memory via the Base Address Register (BAR) file system.

  The PCI BAR File System is only available with the RedHawk kernel from Concurrent Computer Corporation and SLERT versions 1.0-1.6 kernel from Novell. On other systems, PCI Device probing will be disabled within NightProbe.

Advantages for NightTune

The following advantage is afforded NightTune when a RedHawk or SLERT kernel is running:

- Context switch rate

  Allows NightTune user to display the context switch counts per CPU instead of for the overall system.

- CPU shielding

  Individual CPUs can be shielded from interrupts and processes allowing CPUs to be dedicated solely to specific interrupts and processes that are bound to the CPU.

- CPU sibling interference

  Individual CPUs can be marked down to avoid interfering with hyperthreaded sibling CPUs and dual-core sibling CPUs. Hyperthreaded CPUs share all the resources of their sibling CPU. Dual-core CPUs share the CPU cache and a path to memory with their sibling CPU.

- Detailed memory information

  Detailed process memory descriptions include the residency and lock state of any page in a process, and their association with physical memory pools for NUMA systems.

Frequency Based Scheduler

The Frequency Based Scheduler is only available on RedHawk systems from Concurrent Computer Corporation. It is required for all NightSim usage.

FBS Process Deadlines are only available for use on RedHawk 5.2.1 and later systems.

On systems without FBS Process Deadline support, the “Apply Deadline” group box will appear shaded and disabled.

NightSim is only included in NightStar distributions intended for use on RedHawk systems.

C
Variables

This section describes the notation used to reference variables in program resources (see Variable Name Notation), and describes the criteria used to determine eligibility of a variable for probing (see "Variable Eligibility for Program Resources" on page C-2).

Variable Name Notation

Variable names may be used to identify memory addresses in C, C++, and Fortran, as well as Ada (using the MAXAda compiler) programs. NightProbe accepts and displays variables with the following syntax.

NOTE

NightProbe only supports symbolic information in Ada program compiled with Concurrent's MAXAda compiler.

Syntax

`[`*scope*`.]` `...` *name*`[(`*array_subscript*`)]`

Parameters

scope

The name of the scope. Includes the names of enclosing functions, packages, or composite variables. Each one is separated from the next by a dot (`.`). (See "Composite Types" on page C-2 for information about composite types.)

name

The name of the variable. The variable may be either a scalar, an array, a structure or record, or a component of a variable of a composite type.

array_subscript

An index representing a single array element. *Array subscripts* must be enclosed in either parentheses `()` or square brackets `[]`.

Composite Types

To NightProbe, arrays, C and C++ structures and unions as well as C++ classes and Ada records are *composite types*. Composite objects may be recorded as a whole or individual components within the object may be recorded.

Variable Eligibility for Program Resources

Any process on any processor can be a target program for data recording and monitoring.

As stated before, variable names may be used to identify memory addresses in programs. If you wish to identify memory locations by variable name, the target program file must contain symbol table and debug information. Use the **`-g`** compiler option to generate debug information, and do not use the **`-s`** linker option that strips symbol table information from the executable program file.

Any fixed (static) base address in a program can be monitored and recorded. Pointers may be indirected at which time their value is frozen with respect to NightProbe.

The following text lists eligible variables by language.

C

- Variables typed `static`
- Global variables declared outside all functions

Fortran

- Variables typed `static` or `save`
- Variables initialized in a `data` statement
- Variables placed in a `common` block

Ada

The following criteria are used to determine if an Ada data object is eligible for data monitoring/recording:

- The compilation unit containing the object must be a library-level package specification or body. Objects declared in nested packages inside a library-level package are also eligible.
- The object must not be declared in a generic or in the instantiation of a generic.
- The object must have a size and representation which is statically determined at compile time.
- The object may be declared in a library-level package marked with pragma `SHARED_PACKAGE`.

The following Ada data types are eligible for data monitoring/recording:

- Any integer, fixed-point or floating-point type or subtype.
- Any character, Boolean or enumeration type or subtype.
- Access types.
- Array and record types (for records with variant parts, only components that have a statically determined component offset are eligible).

NOTE

Task types and variables declared in Ada procedures or tasks, or objects in an access type's collection, are allocated dynamically, and are, therefore, ineligible for data monitoring/recording.

NOTE

Symbolic variable probing of Ada programs is only supported when the Concurrent MAXAda compiler is used.

D
Keyboard Traversal

NightProbe uses certain key combinations as shortcuts for displaying menus and selecting menu items. These key combinations are called *accelerators* and *mnemonics*. Each dialog has its own set of accelerators and mnemonics that are active only while the keyboard focus is in that dialog. However, the keyboard focus does not have to be in any particular field of the dialog to use accelerators and mnemonics.

- Menus can be displayed with mnemonics.

  Menus can be displayed from the keyboard by typing <Alt>+*mnemonic*. Each of the main windows has a menu bar near the top of the window. The different menus are labeled. For example, the main window has a Programs menu. If you look at the Programs menu, you can see that the P is underlined. P is the mnemonic for the Program menu. That means that, in addition to displaying the Program menu by clicking on it with mouse button 1, you can also display it with <Alt>+p (hold down <Alt> and press p).

  If you decide you don't want to select any of the menu items, you can make the menu go away by typing <Esc> or by clicking somewhere else.

- Menu items can be selected with mnemonics.

  Once a menu is displayed, you can select a menu item by typing only the mnemonic for that item. The mnemonics for the menu items are underlined, just as the mnemonics for the menus are underlined. To select a menu item by using its mnemonic, just press the key.

- Menu functions can be invoked with accelerators.

  Some commonly used menu items have accelerator keys. The functions associated with these menu items can be invoked directly, without displaying the menu, by pressing the accelerator keys. The accelerator keys for a particular menu item are listed next to the item in the menu.

  The accelerator keys are often a combination of a control key plus a letter, such as Ctrl+P. To type Ctrl+P, hold down the control key and press p.

In addition to mnemonics and accelerators, there are also special keys used for navigation within and among windows and fields. These keys include Tab, Shift Tab, Home, End, Page Up, Page Down and the arrow keys.

There are many special keys used to edit text input areas.

The following table contains a list of some of NightProbe's accelerators and the resulting actions; where applicable, it indicates the menu items for which the accelerators provide shortcuts.

Table D-1. NightProbe Accelerators

Accelerator	Menu Item	Action
Ctrl+A	Add New Page	Adds a new page to the main window.
Alt+B	n/a	Raises the Browse page
Alt+C	n/a	Raises the Configuration page
Ctrl+N	New Session	Creates a new session
Ctrl+O	Load Session	Loads a previously saved session
Ctrl+P	Program	Launches the Program Selection dialog so you can add a new program to be probed
Ctrl+Q	Exit	Exits NightProbe but first checks to see if the current session has been modified
Alt+Q	Exit Immediately	Exits NightProbe even if the current session has been modified
Ctrl+Shift+R	Record	Starts recording
Ctrl+Shift+S	Stop	Stops recording
Ctrl+S	Save Session	Saves the current session
Ctrl+T	Select System	Launches the System Selection dialog
Left	n/a	In a tree view, positions the cursor to the parent item and collapses the children
Right	n/a	In a tree view, expands the current item and positions the cursor on the first child
Up	n/a	In a tree view, positions the cursor to the preceding sibling item or to the parent item if no previous sibling exists
Down	n/a	In a tree view, positions the cursor the next sibling item or to the child of an item if the item is expanded.
F1	n/a	Launches the integrated help system and positions the topic to the widget or dialog which currently has the focus.

E
Tutorials

This section contains tutorials which provide a brief introduction to NightProbe using step-by-step instructions:

- "Probing Programs Tutorial" on page E-1 demonstrates probing a program written in C++.
- "Non-Program Probing" on page E-17 demonstrates probing a device

Probing Programs Tutorial

This tutorial demonstrates some of the commonly used features of NightProbe including:

- Creating and selecting a program
- Browsing
- Using the Graph View

The supplied tutorial programs declare and initialize static and dynamic variables. Some of the variables are scalars, some are arrays, and some are structures.

The tutorial files are in the **`/usr/lib/NightProbe/tutorial`** directory. Source listings of these files are in:

- "C++ Sample - cpp_sample.cpp" on page E-13

Creating and Selecting a Program

1. The source code for the sample program used in this tutorial can be found in the **`/usr/lib/NightProbe/tutorial`** directory and are included at the end of this chapter for reference.

2. Copy the source files from **`/usr/lib/NightProbe/tutorial`** and compile the program:

   For example:

   ```
   cp /usr/lib/NightProbe/tutorial/cpp_sample.cpp .
   g++ -g -o cpp_sample cpp_sample.cpp
   ```

3. Invoke NightProbe with the following command:

   ```
   /usr/bin/nprobe &
   ```

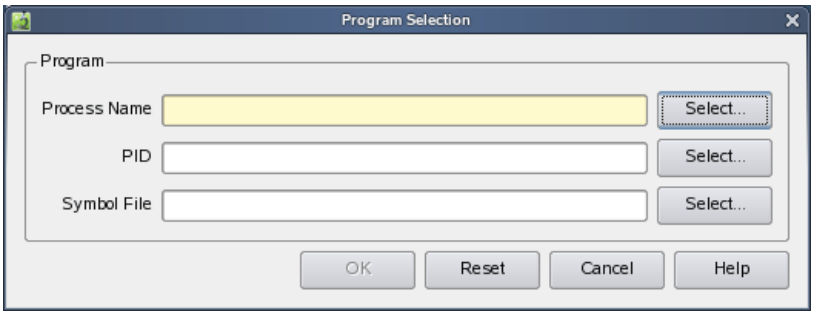
NightProbe displays the NightProbe main window.

4. Invoke the sample program with the following command:

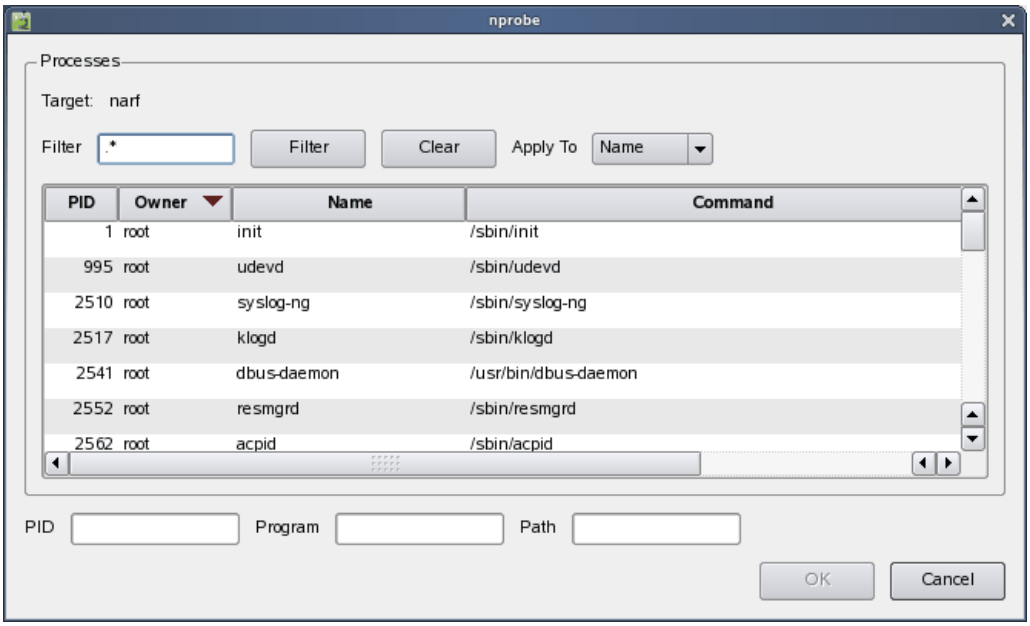
   `./cpp_sample`

5. In the NightProbe main window select the Program option from the Programs menu item.

   NightProbe displays the Program Selection dialog.

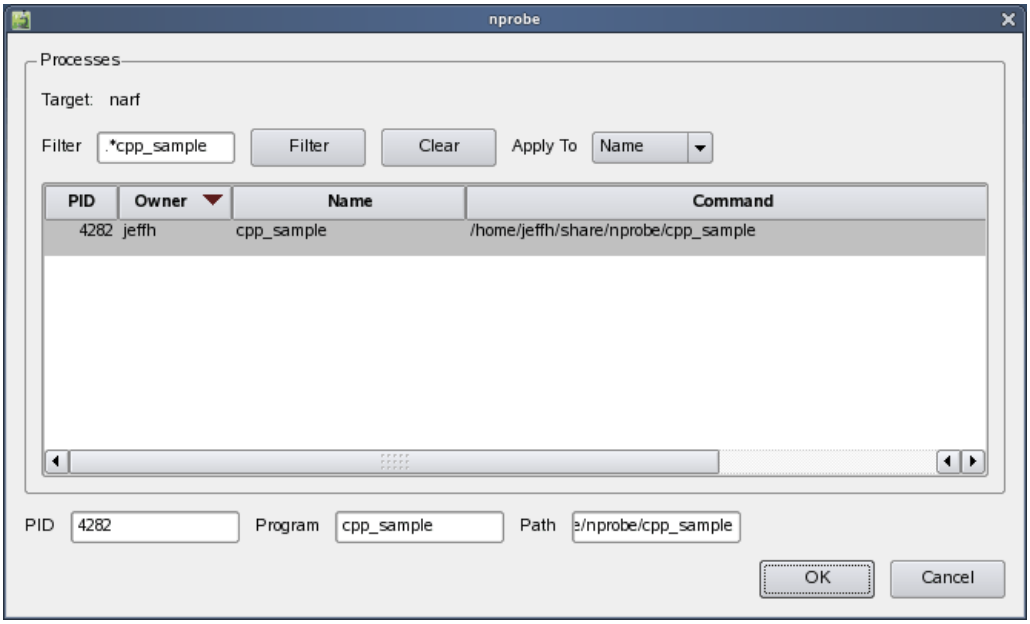


Press the Select… button to the right of the Process Name field. The Process Selection dialog is shown.



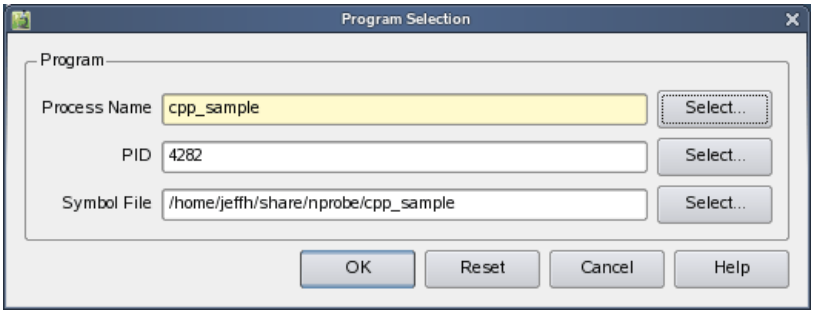
Type cpp_sample into the Filter expression and press the Enter key.

The cpp_sample program is located, its row is selected in the table, and the program name, PID, and executable file name are filled in the fields below.



6. Press the OK button or Enter key.

The parent dialog now has all the information required to add the program.

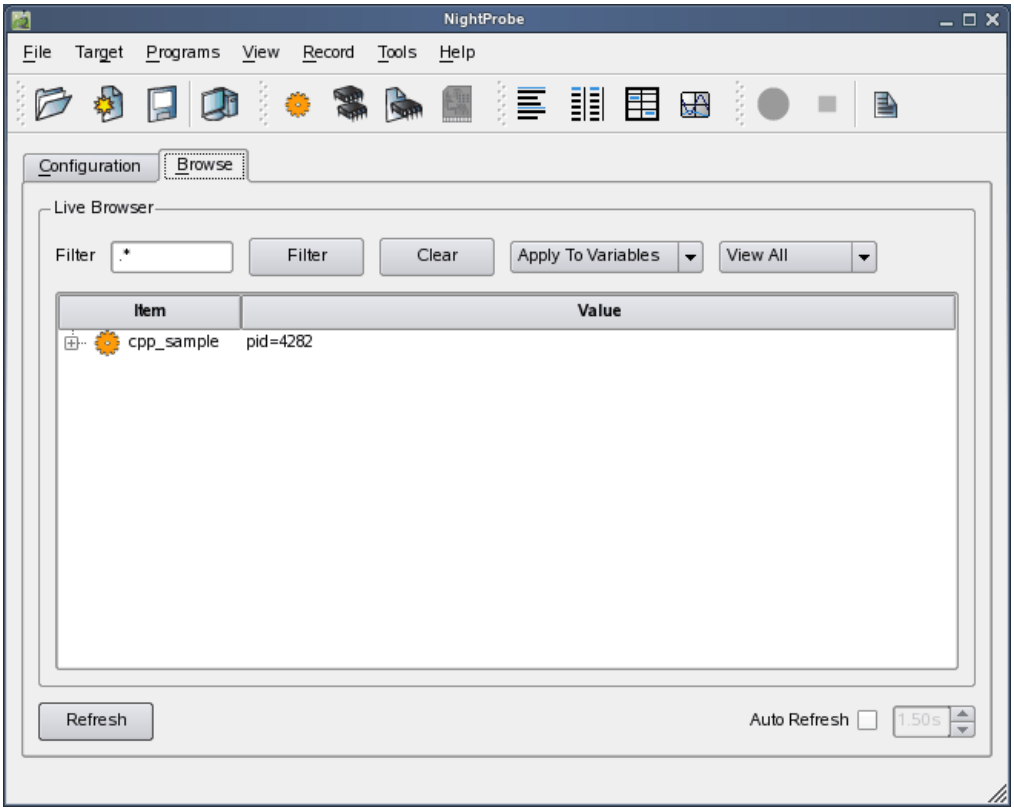


Press the OK button of the Program Selection dialog or the Enter key.

The program is now added to the configuration and the Browse page is raised.

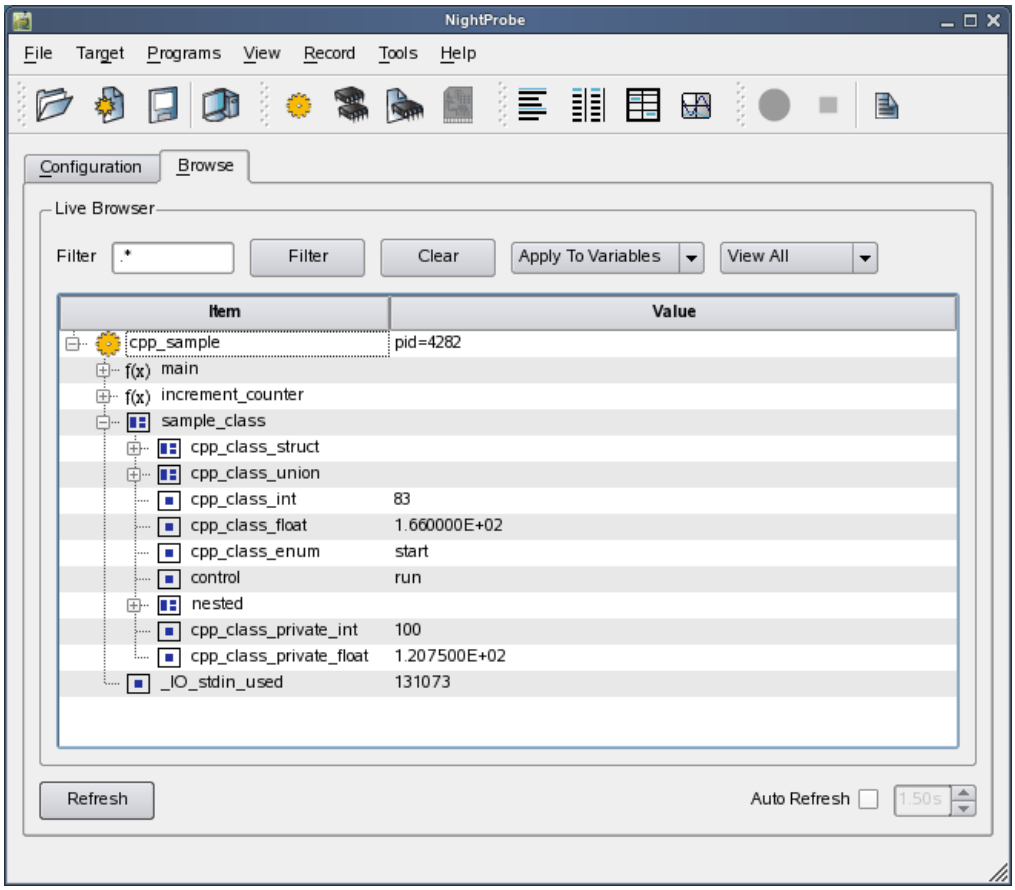
Browsing

The following sections provide an example of the use of the Browse page. For more information see "Browse Page" on page 6-1.



1. Expand the top-level item which represents your cpp_sample process.

2. Expand the sample_class variable.



The values of all the variables are shown.

3. Check the Auto Refresh button near the bottom right-hand portion of the page.

   The values are automatically refreshed at the refresh rate specified in the spinbox to the right of the Auto Refresh checkbox.

   Values are also refreshed whenever the Browse page receives or loses focus.

   Notice the value of the cpp_class_int component of the sample_class variable. It is incremented by the program once per second while the program is in `run` mode.

4. Double-click the *value* of the `cpp_class_int` variable.

   The automatic refresh is paused and the cell in the tree containing the value of cpp_class_int becomes an editable field.

   Type in the value 0 and press the Enter key.

   The value has now been changed to zero and will start incrementing from that value as the program continues to execute.

5. Double-click the *value* associated with the `control` component of `sample_class`.

The automatic refresh is paused once again and the cell in the tree containing the value of control becomes an editable field.

The program uses this enumerated variable to control its execution.

Type in the value hold and press the Enter key.

The program now stops incrementing the cpp_class_int variable. Values of enumerated variables may be typed in using their underlying integer representation or using the textual enumeration value as we have done here.

6. Change the value of the control component back to run.

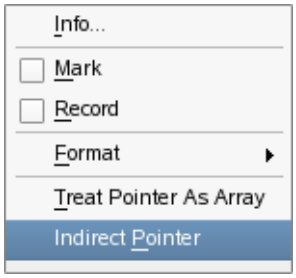
Indirecting Pointers

Pointer variables may be viewed within NightProbe and they can be indirected to see the value of memory that the variable points at.

1. Collapse the `sample_class` variable.

2. Expand the `increment_counter` function in the tree.

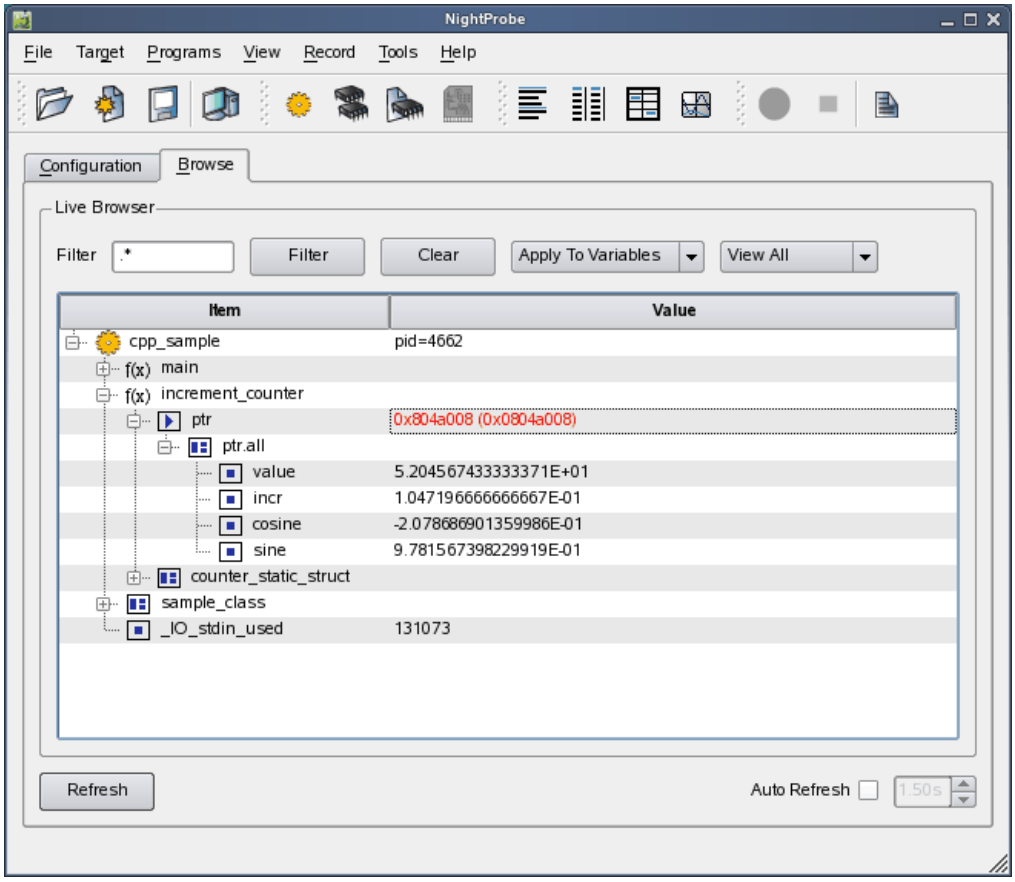
   Right-click on the `ptr` variable



   Click on the Indirect Pointer option.

   The pointer is dereferenced and the item it refers to is shown as a child of the pointer.

3. Expand the `ptr.all` variable.



Notice the value of the `ptr` variable is now shown in red. This indicates that the value has been frozen within NightProbe. The actual value of the pointer at any given time is shown in parenthesis, but the child items shown are access via the frozen value. You can refreeze the value to see what the current pointer variable points to using the context menu.

Selecting Variables for Viewing in Other Pages

The Browse page provides an immediate view of all variables that can be probed in a program. Often, however, only a subset of the variables are interesting at any given time.

We will now use the Browse page to select variables of particular interest so we can concentrate on viewing only them and use additional viewing techniques within NightProbe.

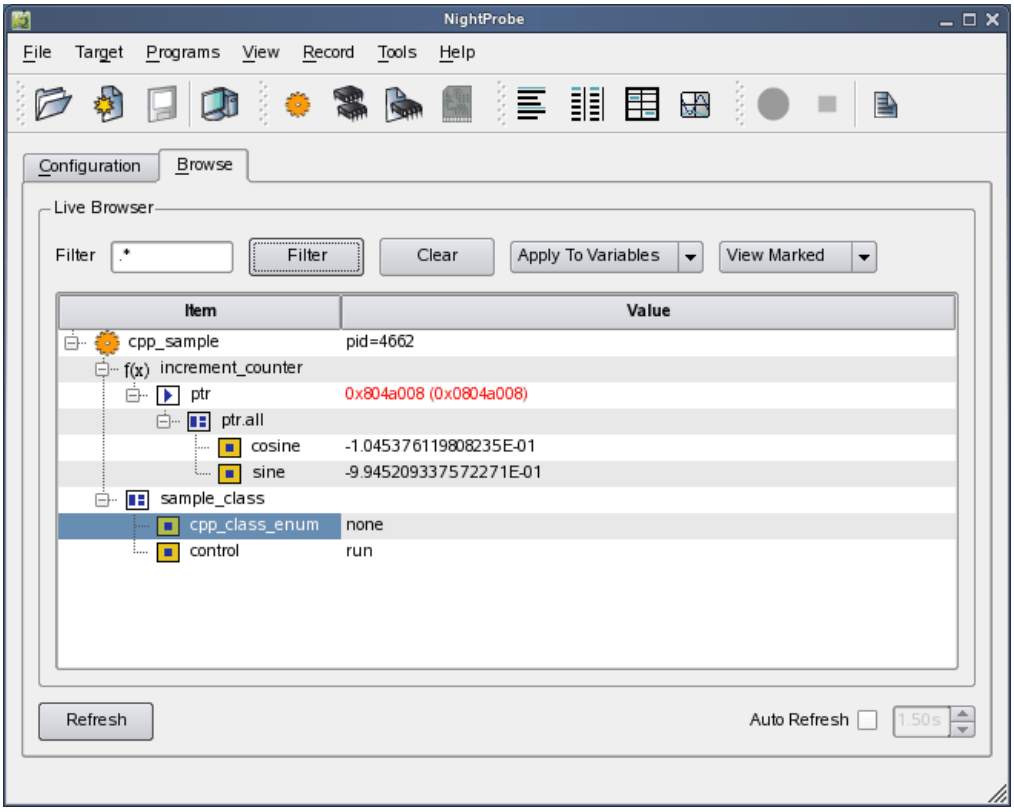
1. Right-click on the `cosine` component of the expanded `ptr` variable and check the Mark checkbox inside that menu.

This causes the `cosine` component's icon to change to a yellow background indicating it is a variable of marked interest.

2. Do the same to the `sine` component of the expanded `ptr` variable as well as to the `control` and `cpp_class_enum` components of the `sample_class` variable (you will have to re-expand the `sample_class` variable first).

3. In the upper right-hand area of the Browse page, select the View Marked item from the right-most option list.

4. Press the Filter button.



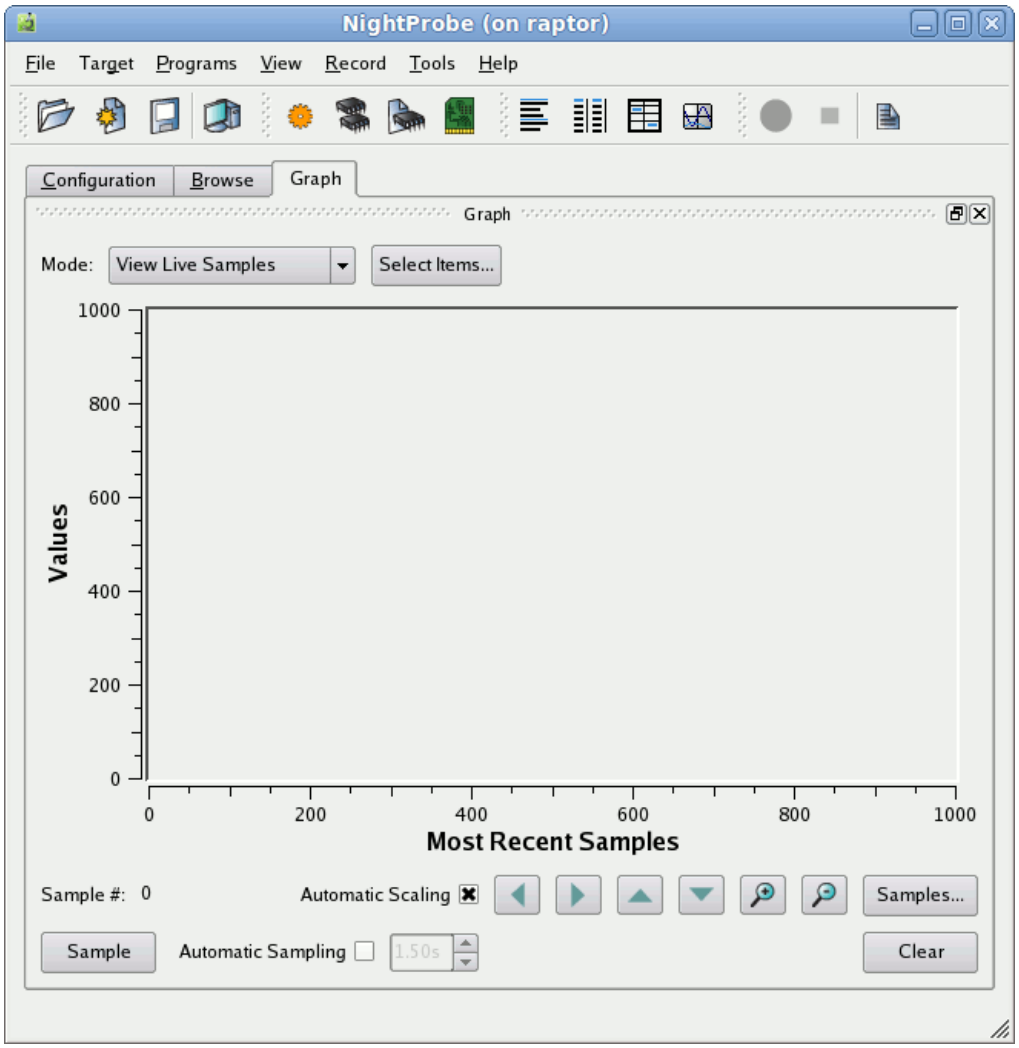
This causes the view to filter variables such that only those whose Mark attribute is set are shown, and their parent items.

5. Change the view back to View All and press the Filter button.

Using the Graph View

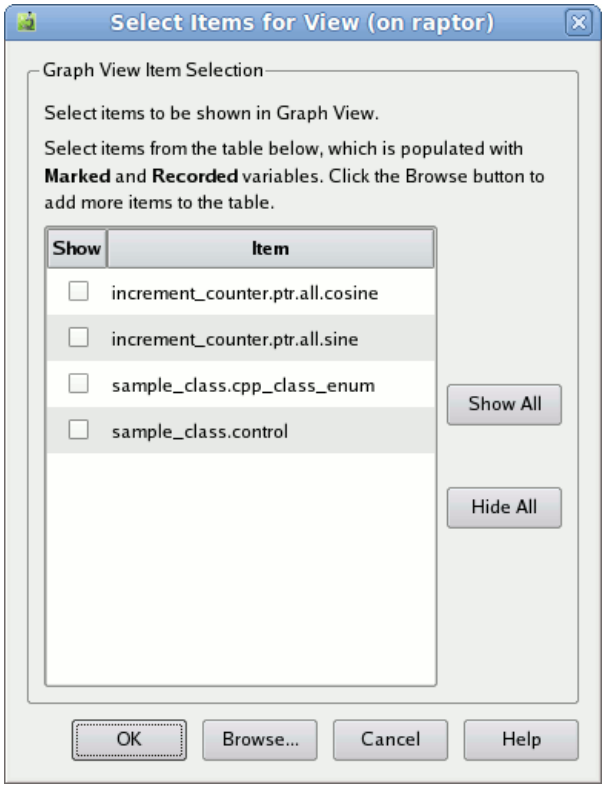
NightProbe provides four additional viewing mechanisms. We will use the Graph view.

1. Select the Graph option from the View menu on the main window.



A new viewing page is added to the main window and a graph view is added to it.

2. Press the Select Items... button.



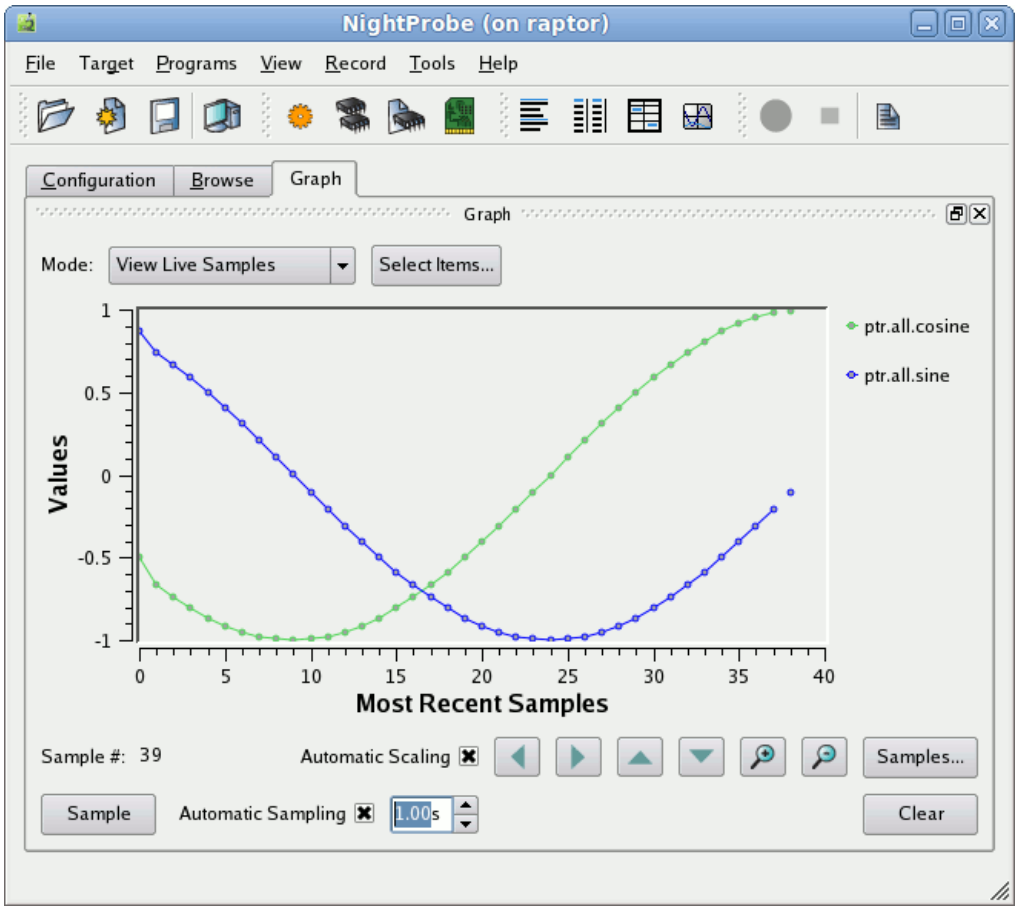
The Item Selection dialog is shown. This dialog controls which items will be shown in the graph panel.

3. Click the cells in the Show checkboxes for the rows for the `sine` and `cosine` components.

   Press the OK button to close the dialog.

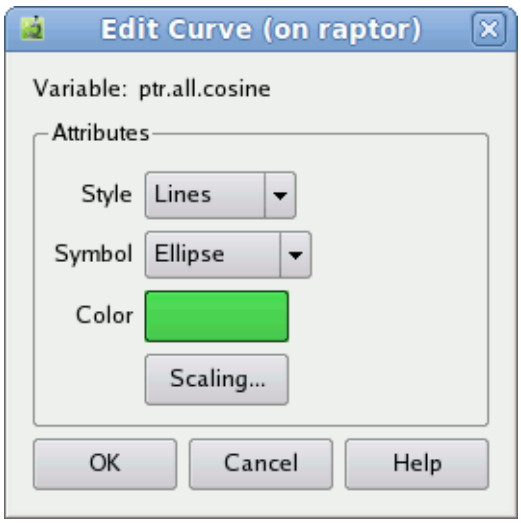
4. Check the Automatic Sampling checkbox in the Graph panel.

5. Change the refresh rate to 1.0 seconds in the spinbox to the right of the Automatic Sampling checkbox.



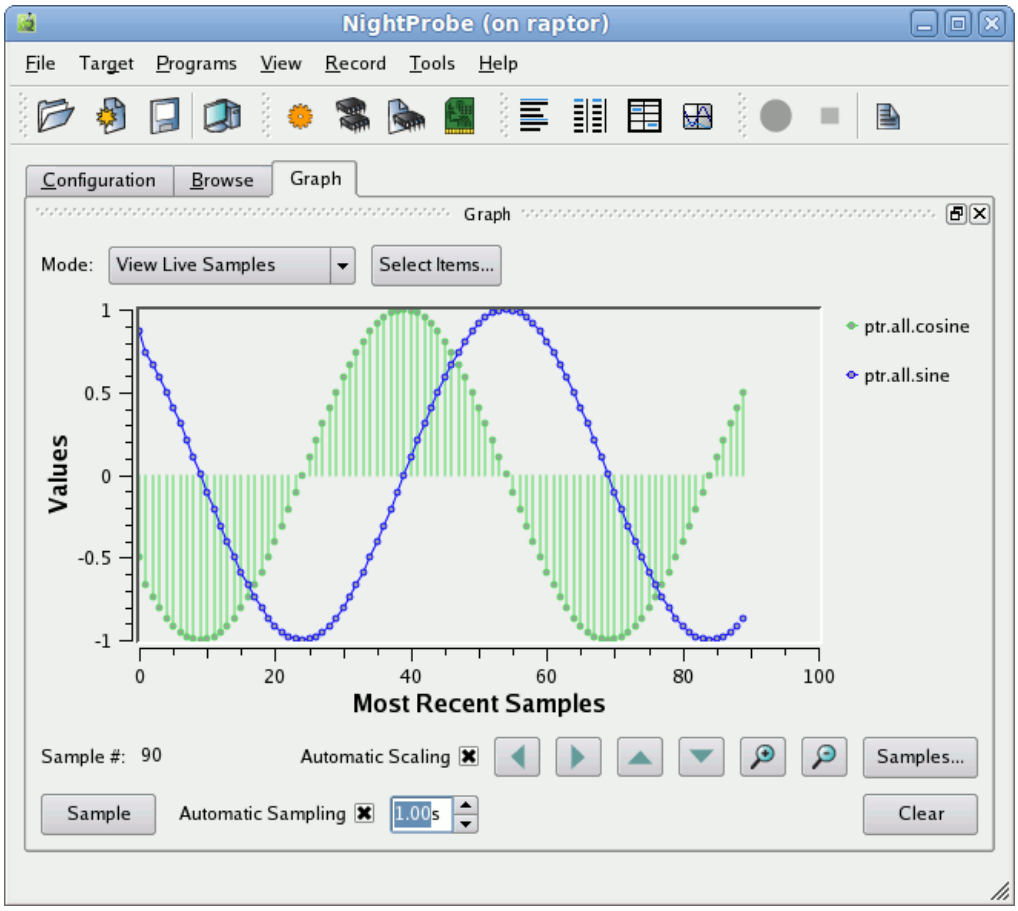
Samples begin to be collected and the sine and cosine waves start to appear in the graph.

6. Right-click the cosine label in the legend area to the right of the graphed lines.



The Edit Curve Attributes dialog appears.

Change the Style to Sticks and press the OK button.



The graph changes based on the style selected for the cosine item.

Exiting NightProbe

1. Select the Exit Immediately option from the File menu of the main window to exit NightProbe immediately, even though the current session has been modified.

C++ Sample - cpp_sample.cpp

```
#include <stdlib.h>
#include <unistd.h>
#include <math.h>

#define ARRAY_SIZE 4

class cpp_class {
public:
   enum states {
      none, init, freeze, start, stop, in_flight, approach, land
   };

   enum controls {
      halt, hold, run
   };

   struct struct_type {
      int      struct_int;
      float    struct_float;
      float    struct_float_array [ARRAY_SIZE];
      states   struct_enum;
   } cpp_class_struct;

   union union_type {
     int   union_int;
     float union_float;
   } cpp_class_union;

   int            cpp_class_int;
   float          cpp_class_float;
   states         cpp_class_enum;

   float get_private_float(void);
   void  set_private_float(float new_value);

   controls       control;

   cpp_class (void);

   class nested {
   public:
      struct nested_struct {
         struct_type nested_struct_struct;
         int         nested_struct_int_array [ARRAY_SIZE];
      } cpp_nested_class_struct;

      nested (void);
   } nested;



   void cpp_procedure(void);

private:

   int   cpp_class_private_int;
   float cpp_class_private_float;

};
```

```
cpp_class::nested::nested(void){

   // Initialize variables in the nested cpp_class::nested class.

   cpp_nested_class_struct.nested_struct_struct.struct_int = 0;
   cpp_nested_class_struct.nested_struct_struct.struct_float = 0;
   cpp_nested_class_struct.nested_struct_struct.struct_enum = none;

   for (int i = 0; i < ARRAY_SIZE; i++) {
      cpp_nested_class_struct.nested_struct_int_array[i] = i * 100;
   }
}


cpp_class::cpp_class (void) {

   // Initialize variable cpp_class

   cpp_class_int     = 0;
   cpp_class_float   = 0.0;
   cpp_class_enum    = none;

   cpp_class_struct.struct_int   = 0;
   cpp_class_struct.struct_float = 0.0;
   cpp_class_struct.struct_enum  = none;

   for (int i = 0; i < ARRAY_SIZE; i++) {
      cpp_class_struct.struct_float_array[i] = i * 100.0;
   }

   control = run;

   cpp_class_private_int    = 100;
   cpp_class_private_float  = 100.0;
}

float
cpp_class::get_private_float(void)
{
   return cpp_class_private_float;
}

void
cpp_class::set_private_float(float new_value)
{
   cpp_class_private_float = new_value;
}

void
cpp_class::cpp_procedure()
{

   // Increment variables in cpp_class

   cpp_class_int++;
   cpp_class_float += 2.0 ;

   switch (cpp_class_enum) {
   case land:
      cpp_class_enum = none;
      break ;

   default:
      cpp_class_enum = (states) (cpp_class_enum + 1);
      break;
```

```
   }

   cpp_class_struct.struct_int   += 1;
   cpp_class_struct.struct_float += 2.0;

   switch (cpp_class_enum) {
   case land:
      cpp_class_struct.struct_enum = none;
      break;
   default:
      cpp_class_struct.struct_enum =
         (states)(cpp_class_struct.struct_enum + 1);
      break;
   }

   for (int i = 0; i < ARRAY_SIZE; i++) {
      cpp_class_struct.struct_float_array[i] *= .995;
   }


   // Increment variables in nested class in cpp_class.
   for (int i = 0 ; i < ARRAY_SIZE ; i++ ) {
      nested.cpp_nested_class_struct.nested_struct_struct.struct_int += 1;
      nested.cpp_nested_class_struct.nested_struct_struct.struct_float += 2.0;

      switch (nested.cpp_nested_class_struct.nested_struct_struct.struct_enum) {
      case land:
         nested.cpp_nested_class_struct.nested_struct_struct.struct_enum = none;
         break ;

      default:
         nested.cpp_nested_class_struct.nested_struct_struct.struct_enum =
            (states) (nested.cpp_nested_class_struct.
                              nested_struct_struct.struct_enum + 1);
         break ;
      }

      nested.cpp_nested_class_struct.nested_struct_int_array[i] += 1;
   }
}


void
increment_counter(void)
{

   enum state {even, odd};

   struct counter_struct_type {
      int   counter_struct_int;
      state counter_struct_state;
   };

   static struct counter_struct_type counter_static_struct = {0, even};

   if (++counter_static_struct.counter_struct_int % 2) {
      counter_static_struct.counter_struct_state = odd;
   }
   else {
      counter_static_struct.counter_struct_state = even;
   }

typedef struct {
   double value;
   double incr;
```

```
   double cosine;
   double sine;
} calc_t;

static calc_t * ptr = NULL;

if (!ptr) {
   ptr = new calc_t;
   ptr->value = 0.0;
   ptr->incr = 3.14159/30.0;
}
ptr->value += ptr->incr;
ptr->cosine = cos(ptr->value);
ptr->sine   = sin(ptr->value);
}



// Driver program.  Continuously loop, calling the modules
// for each language.

cpp_class sample_class;


int
main() {

   static int main_static_int = 0;
   float      local_float     = 0.0;

   sample_class.control = cpp_class::run;

   while (sample_class.control != cpp_class::halt) {
      switch (sample_class.control) {

      case cpp_class::run:
         sample_class.cpp_procedure();
         local_float = sample_class.get_private_float() + 0.25;
         sample_class.set_private_float(local_float);
         break;

      case cpp_class::hold:
      case cpp_class::halt:
         break;
      }

      increment_counter();
      sleep(1);
   }

   return main_static_int;
}
```

Non-Program Probing

This tutorial demonstrates NightProbe's ability to probe resources other than programs.

For simplicity, we will probe a simple text file on disk, whereas more common and useful cases might be probing a shared memory segment set up by some applications or the memory of a device accessible through **`/dev/mem`**.

Selecting the Resource

1. Execute the following command from a shell session:

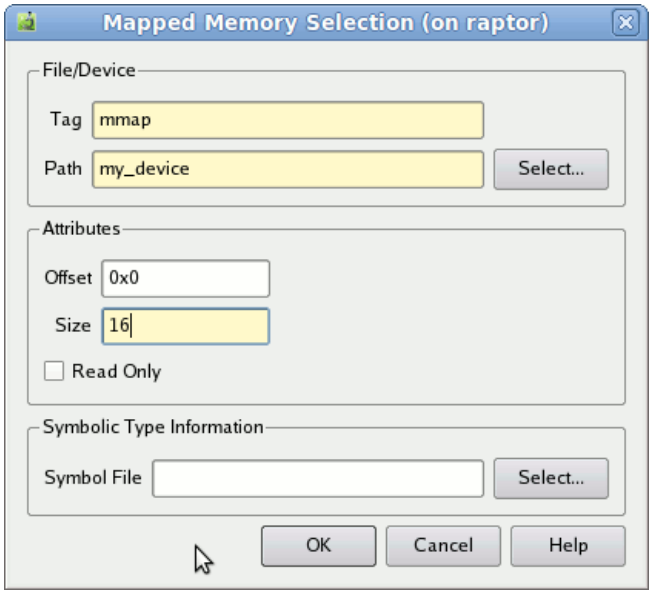
   ```
   echo "The Answer is 43" > my_device
   ```

2. Invoke NightProbe:

   ```
   /usr/bin/nprobe &
   ```

3. In the NightProbe main window select the Mapped Memory Device option from the Programs menu item.

   NightProbe displays the Mapped Memory Selection dialog.



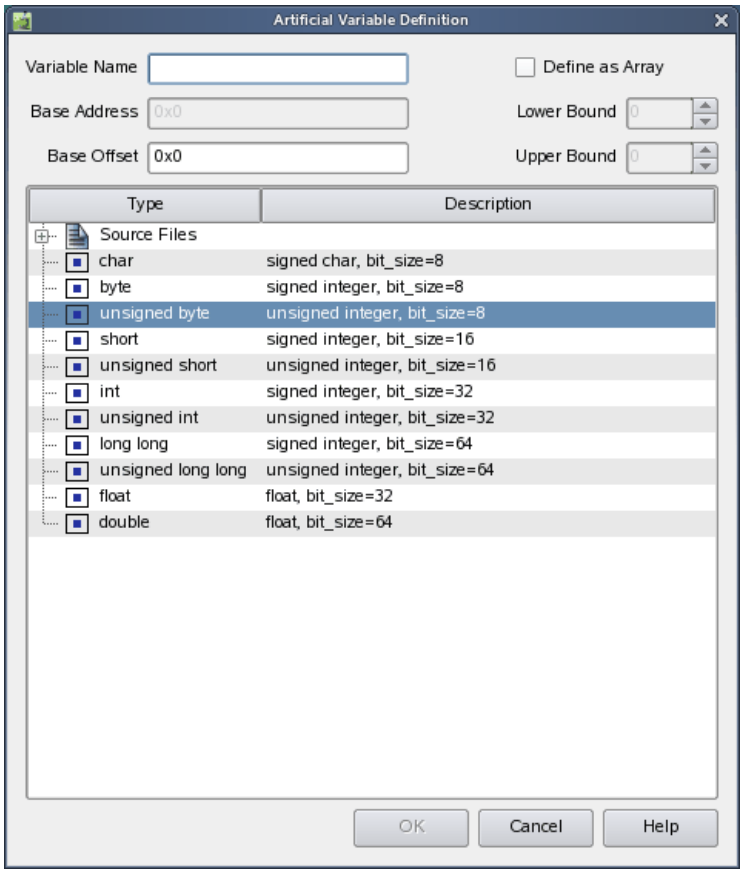
   Enter my_device in the Path field of the File/Device section or use the Select… button to locate the file that we created in step #1 above.

4. Enter 16 in the Size field of the Attributes area.

5. Press the OK button.

   The Artificial Variable Definition dialog appears as shown in the next section.

Creating a View into the Resource



This dialog is presented so that you can define an artificial variable to be associated with the resource so we can probe it.

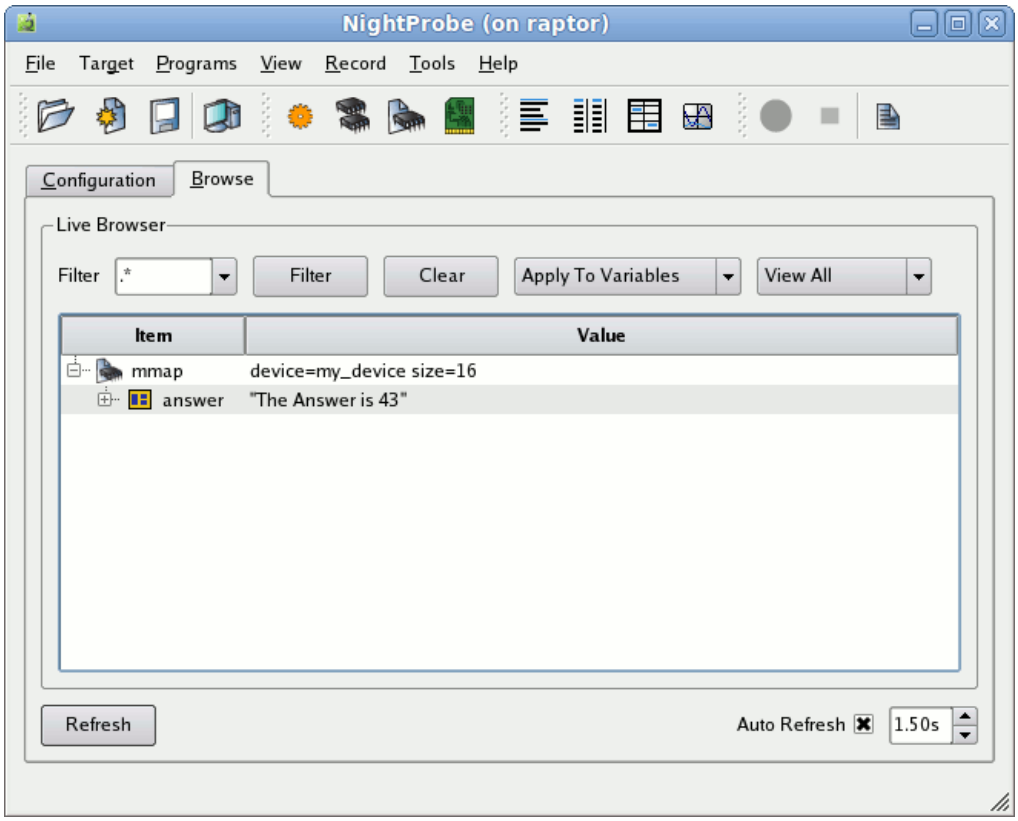
1. Type a variable name into the Variable Name field, e.g. `answer`.
2. Select the `char` item in the Type tree.
3. Check the Define as Array checkbox in the upper right area of the dialog.
4. Set the Lower Bound to zero and the Upper Bound to 15.
5. Press the OK button.

   We have now created an artificial variable which is associated with the file that we mapped in steps # 3-6 above.

Viewing the Data

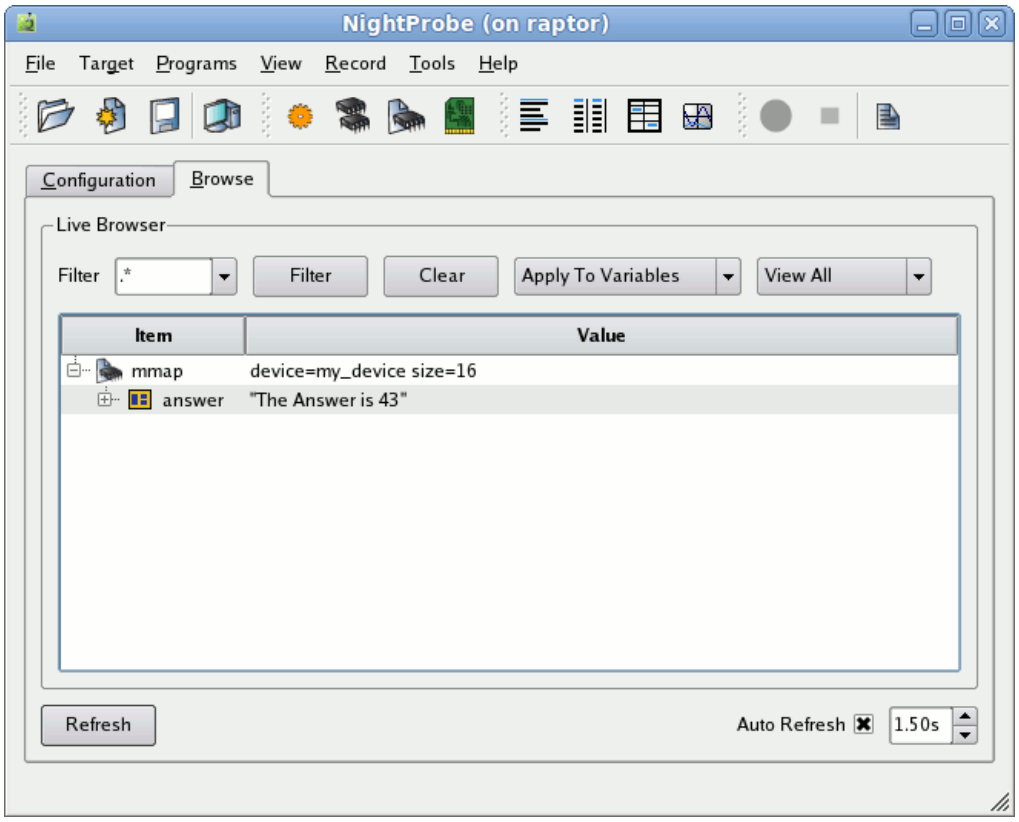
We have now added the resource to our configuration and defined a variable within it.

1. The Browse tab in the main window is automatically raised,

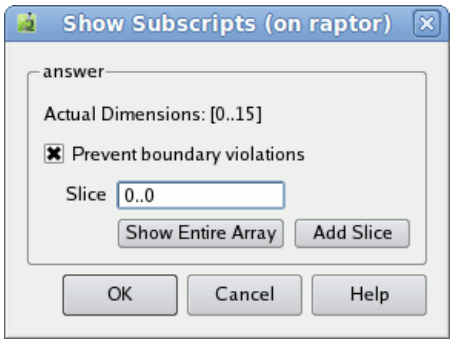


and the Mapped file is shown in the tree.

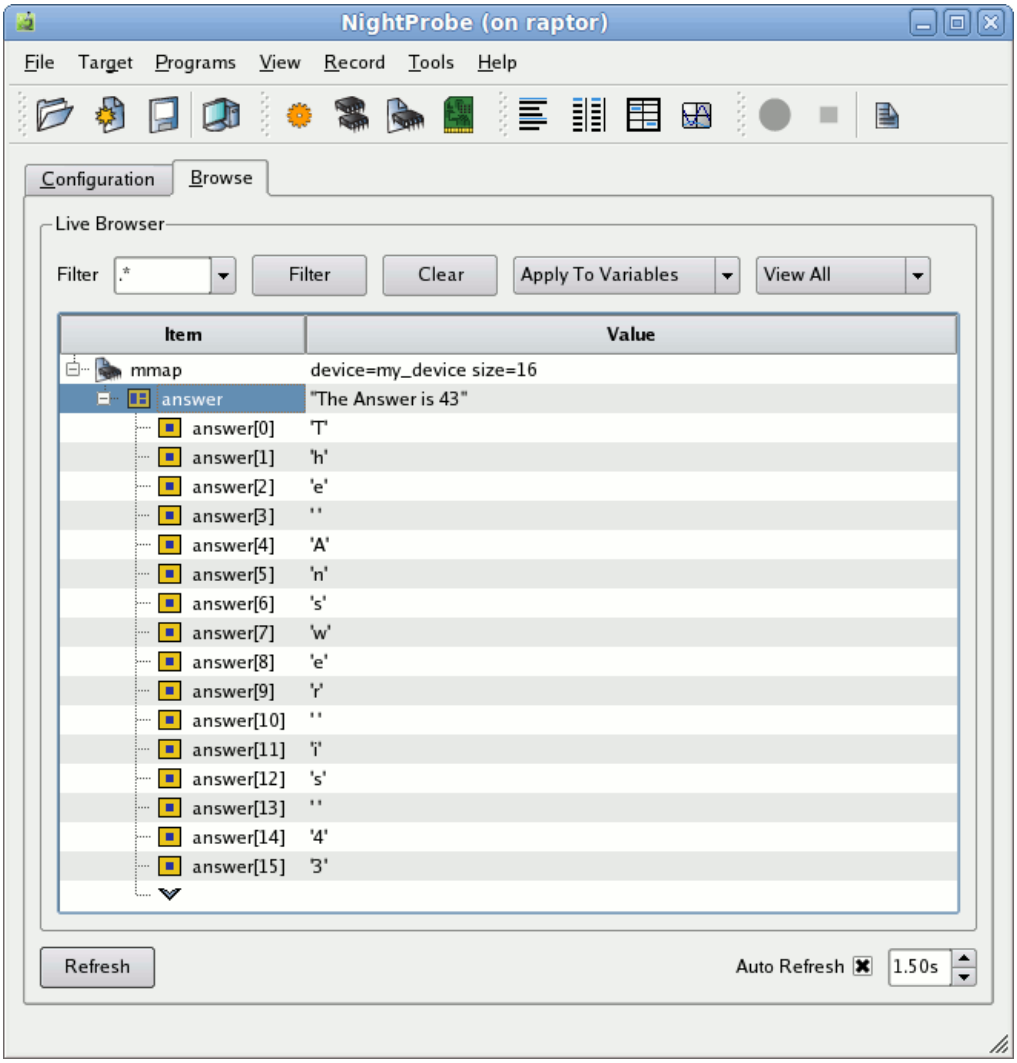
2. Expand the Mapped file item.



The string is shown. Right-click the answer item and select the Show Subscripts... option from the context menu.



Presss the Show Entire Array button and then press the OK button.



Now the array item has all 16 components visible.

4. Double-click the *value* of the last component.

   The cell becomes selected and changes to an editable field.

5. Backspace over the current value and type in '2' (including the surrounding single quotes) and press the Enter key.

   The value of that component has now change to the ASCII character 2.

6. In a shell outside of NightProbe, look at the contents of the my_device file:

```
cat my_device
```

and see that the answer is now 42.

Exiting NightProbe

1. Select the Exit Immediately option from the File menu of the main window to exit NightProbe immediately, even though the current session has been modified.

Conclusion

This concludes the tutorial chapter.

Remember to make use of context sensitive help within NightProbe itself.

You can press F1 at any time for help relating to an item or dialog that has focus, or you can select various options from the Help menu.

F
Privileged Access

Some features of NightProbe require either root access or privileged access as described below.

This chapter provides an overview of the capabilities mechanism support by some operating systems.

The following operating system kernels support the capabilities mechanism:

- RedHawk Linux (all versions)
- SUSE Linux Enterprise Real Time (versions 1.0-1.6 only)

Capabilities

The following capabilities may be required when using NightProbe:

- `CAP_SYS_RAWIO`

  If you wish to probe PCI devices or other target resources to which you do not have appropriate file access (e.g. **`/dev/mem`** or a shared memory segment or process owned by a different user), you must have this capability.

- `CAP_SYS_NICE`

  In order to set the CPU bias or scheduling policy and priority of the NightProbe target or of the program specified using the Program Output method, you must have this capability.

Linux provides a means to grant otherwise unprivileged users the authority to perform certain privileged operations. The Pluggable Authentication Module (see **`pam_capability(8)`**) is used to manage sets of capabilities, called *roles*, required for various activities.

Linux systems should be configured with an `nprobeuser` role which provides the `CAP_SYS_RAWIO` and `CAP_SYS_NICE` capabilities.

Edit **`/etc/security/capability.conf`** and define the `nprobeuser` role (if it is not already defined) in the "ROLES" section:

```
role nprobeuser CAP_SYS_RAWIO CAP_SYS_NICE
```

Additionally, for each NightProbe user on the target system, add the following line at the end of the file:

```
user   username     nprobeuser
```

where *username* is the login name of the user.

If the user requires capabilities not defined in the `nprobeuser` role, add a new role which contains `nprobeuser` and the additional capabilities needed, and substitute the new role name for `nprobeuser` in the text above.

In addition to registering your login name in **`/etc/security/capability.conf`**, certain files under the **`/etc/pam.d`** directory must also be configured to allow capabilities to be activated.

To activate capabilities, add the following line to the end of selected files in **`/etc/pam.d`** if it is not already present:

```
session required pam_capability.so
```

The list of files to modify is dependent on the list of methods that will be used to access the system. The following table presents a recommended configuration that will grant capabilities to users of the services most commonly employed in accessing a system.

Table F-1. Recommended /etc/pam.d Configuration

/etc/pam.d File	Affected Services	Comment
`remote`	telnet rlogin rsh (when used w/o a command)	Depending on your system, the **`remote`** file may not exist. Do not create the **`remote`** file, but edit it if it is present.
`login`	local login (e.g. console) telnet* rlogin* rsh* (when used w/o a command)	*On some versions of Linux, the presence of the **`remote`** file limits the scope of the **`login`** file to local logins. In such cases, the other services listed here with **`login`** are then affected solely by the **`remote`** configuration file.
`rsh`	rsh (when used with a command)	e.g. **`rsh system_name a.out`**
`sshd`	ssh	You must also edit **`/etc/ssh/sshd_config`** and ensure that the following line is present: **`UsePrivilegeSeparation no`**
`gdm`	gnome sessions	
`kde`	kde sessions	

If you modify **`/etc/pam.d/sshd`** or **`/etc/ssh/sshd_config`**, you must resetart the **`sshd`** service for the changes to take effect:

```
service sshd restart
bash /etc/init.d/sshd restart
```

In order for the above changes to take effect, the user must log off and log back onto the target system.

NOTE

To verify that you have been granted capabilities, issue the following command:

```
/usr/sbin/getpcaps $$
/sbin/getpcaps $$
```

The output from that command will list the roles currently assigned to you.

Index

Symbols

/usr/bin/nprobe 2-1

A

Accelerator D-1
Access type C-3
Address
    memory C-1
Alt key D-1
Array index C-1
Array type C-3

B

Boolean type C-3
Browse Page 6-1
    Find bar 6-9
Button
    Help 2-4

C

C program C-1
`CAP_SYS_NICE` capability F-1
`CAP_SYS_RAWIO` capability F-1
Capabilities
    `CAP_SYS_NICE` F-1
    `CAP_SYS_RAWIO` F-1
Character type C-3
Common block
    Fortran C-2
Compilation C-2
Composite C-2
Configuration
    data sampling 2-3
Configuration file 1-2
Configuration Page 5-1
Context-sensitive help 4-14
Control key D-1

D

Data monitoring 1-1
Data recording 1-1, 4-1, 12-1
Data sampling configuration 2-3
data statement
    Fortran C-2
Data structures
    `np_endian_type` 13-2
    `np_handle` 13-3
    `np_header` 13-3
    `np_item` 13-4
    `np_process` 13-4
    `np_trigger_handle` 13-20
    `np_type` 13-5
Debugging information C-2

E

End key D-1
Enumeration C-3
Environment variable
    `NSLM_SERVER` A-2
Esc key D-1
Exit menu option 4-4

F

File
    /usr/bin/nprobe 2-1
    configuration 1-2
File menu 4-2
Find bar 6-9
Fixed licenses A-1
Fixed-point type C-3

Floating licenses A-1
Floating-point type C-3
Fortran program C-1
Functions
`np_avail()` 13-7
`np_close()` 13-10
`np_error()` 13-12
`np_format()` 13-11
`np_host_endian()` 13-12
`np_open()` 13-6
`np_read()` 13-8
`np_trigger()` 13-22
`np_trigger_close()` 13-23
`np_trigger_error()` 13-24
`np_trigger_open()` 13-21

G

Generic C-2
Global variable C-2
Graph View 11-1

H

Help
On Context 4-14
Help button 2-4
Help menu 2-4, 4-14
Home key D-1

I

Index
array C-1
Input area
editing D-1
Integer type C-3
invoking nprobe 2-1

K

Key
Alt D-1
Control D-1
End D-1
Esc D-1
Home D-1
Page Down D-1
Page Up D-1
Shift Tab D-1
Tab D-1

L

License A-1
firewall configurations A-3, A-5
fixed A-1
floating A-3
installation A-1
keys A-1
modes A-1
`nslm_admin` A-1, A-3
report A-3
requests A-2
server A-2, A-3
support A-6
List View 8-1

M

Main Window 4-1
Mapped Memory Selection Dialog 3-10
Memory address C-1
Menmonic
menu D-1
menu item D-1
Menu
File 4-2
Help 2-4
Timer 4-4
Menu Help 4-14
Menu option
Exit 4-4
On Context 4-14
On Help 4-14, 4-15
Open Config File 4-3
Save Config File 4-3
Save Config File As 4-3, 4-4
Mnemonic D-1
Monitoring (see Data monitoring and Viewer)
Mouse button 1 D-1

N

NightProbe Datastream API
    np_avail() 13-7
    np_close() 13-10
    np_endian_type 13-2
    np_error() 13-12
    np_format() 13-11
    np_handle 13-3
    np_header 13-3
    np_host_endian() 13-12
    np_item 13-4
    np_open() 13-6
    np_process 13-4
    np_read() 13-8
    np_type 13-5
NightProbe Trigger API
    np_handle 13-20
    np_trigger() 13-22
    np_trigger_close() 13-23
    np_trigger_error() 13-24
    np_trigger_open() 13-21
NightTrace 1-2
np_avail() 13-7
np_close() 13-10
np_endian_type 13-2
np_error() 13-12
np_format() 13-11
np_handle 13-3
np_header 13-3
np_host_endian() 13-12
np_item 13-4
np_open() 13-6
np_process 13-4
np_read() 13-8
np_trigger() 13-22
np_trigger_close() 13-23
np_trigger_error() 13-24
np_trigger_handle 13-20
np_trigger_open() 13-21
np_type 13-5
nprobe 2-1
nslm_admin A-1, A-3
NSLM_SERVER A-2

O

On Context menu option 4-14
On Help menu option 4-14, 4-15
Open Config File menu option 4-3

P

Package
    Ada C-2
Page Down key D-1
Page Up key D-1
Panels 7-1
PCI Device Selection Dialog 3-12
Pragma SHARED_PACKAGE C-2
Program
    C C-1
    compilation C-2
    Fortran C-1
Program Selection dialog 3-4

R

Record C-2
Record type C-3
Recording (see Data recording)

S

Save Config File As menu option 4-3, 4-4
Save Config File menu option 4-3
save variable C-2
Selection Dialogs 3-1
Shared Memory Selection Dialog 3-7
SHARED_PACKAGE pragma C-2
Shift Tab key D-1
Spreadsheet View 10-1
Static variable C-2
Structure C-2
Symbol table C-2
System Selection Dialog 3-1

T

Tab key D-1
Table View 9-1
Task type C-3
Text input area
    editing D-1
Timer menu 4-4
Type
    access C-3
    array C-3

Boolean C-3
character C-3
enumeration C-3
fixed-point C-3
floating-point C-3
integer C-3
record C-3

U

Union C-2

V

Variable
global C-2
static C-2
Variable name C-1, C-2