
User Guide to

Tuner for PlayStation®2

SN Systems



**SN Systems Ltd
Version 1.3a
June 2003**

Copyright © SN Systems Ltd, 2000-2003. All rights reserved.

"ProDG" and the SN logo are registered trademarks of SN Systems Ltd. "PlayStation" is a registered trademark of Sony Computer Entertainment Inc. "Microsoft", "Windows" and "Windows NT" are registered trademarks of Microsoft Corporation. Other product and company names mentioned herein may be the trademarks of their respective owners.

TunerPS2_UG-E.doc / User Guide to Tuner for PlayStation 2 / June 2003

Document change control

Ver.	Date	Changes
1.0	June 2002	First release
1.1	9th July 2002	Release v1.1 Expanded user markers. New ToolTips. Acquire options dialog amended. Frame navigation enhanced.
1.1.5	January 2003	Corrected "0-32" to "0-31 (0 is inclusive)" in section on user markers. Revised 'Updates and technical support'. Amended Acquire options dialog, added Quick Acquire option to Acquire menu. Added Quick Acquire section to Starting the ProDG Tuner. Added new Function Display options dialog to Appendix. Added Quick Acquire option to Starting ProDG Tuner. Added new sections PC sampling and Viewing source information for a function. Amended Source view. Added extra information to Viewing ToolTips. Added Function List Issues section to Ch. 2 and Function Display Options dialog to Reference.
1.1.6	22nd Jan. 2003	Release v1.1.6.
1.1.7	April 2003	Release v1.1.7.
1.2	April 2003	Release v1.2.
1.3	May 2003	Complete restructure and rewrite.
1.3a	Jun. 2003	Minor corrections.

Contents

Chapter 1: Introduction	1
Overview	1
Installation	2
Obtain your SN Systems run-time license.....	2
System requirements.....	2
Supported development tools	2
Supported debug formats	2
Technical support and upgrades	3
Chapter 2: How it works	5
Overview	5
Frame detection	5
Game ELF.....	5
Implications for your game	5
Detecting if Tuner is running	6
Function list issues	7
Chapter 3: Quick start guide	9
Capturing data	9
Chapter 4: User interface	13
Application toolbar.....	13
File menu	13
View menu	13
Capture menu	14
Window menu	14
Help menu.....	14
Set source search path.....	14
Select target.....	15
Console View	16
Tool Tips	17
Chapter 5: Downloading and capturing data	19
Overview	19
Breakdown of capture steps	21
Select ELF	21
Select sync functions	21
Specify capture options	23

Chapter 6: Viewing and analyzing captured data	29
Overview	29
Game View	30
Frame View.....	32
Source View	35
Launching a source view	37
Moving around the source file	38
Disassembly tab	38
Mixed View	38
Context menu	39
Status bar.....	40
Default settings	40
Open related file	41
Change syntax coloring.....	41
Chapter 7: DMA activity	45
Overview	45
Capturing DMA activity	45
Viewing DMA activity data	46
Launching a source view for a DMA transfer	46
Chapter 8: Interrupts	47
Overview	47
Capturing interrupts	47
Viewing interrupt data	48
Chapter 9: VIF/GIF statistics	49
Overview	49
Capturing VIF/GIF statistics	49
Viewing VIF/GIF statistical data.....	49
Chapter 10: Performance counters	51
Overview	51
Performance Counter 0	51
Performance Counter 1	52
Capturing performance counters	52
Viewing performance counter data	53
Chapter 11: User markers	55
Overview	55
Adding user markers to your code	55
Viewing user marker data	56
Launching a Source View for a user marker	57

Chapter 12: Function instrumenting	59
Overview	59
Selecting functions to instrument.....	59
Viewing instrumented function data	60
Launching a Source View for an instrumented function.....	61
Refining the list of functions to instrument	62
Adding and removing functions.....	62
Adding parent and child functions	63
Chapter 13: PC sampling	65
Overview	65
Activating PC sampling	65
Viewing PC sampling data.....	65
Function display options	67
Filtering function names	68
Launching a Source View for a sampled function	68
Index	71

[THIS PAGE IS LEFT BLANK INTENTIONALLY]

Chapter 1: Introduction

Overview

Tuner lets you capture and visualise program behaviour so that you can eliminate conflicts and bottlenecks in your code. Data is captured while you play the game and this can then be analyzed frame by frame and saved for later comparison with your optimized code.

Tuner captures the following information:

- DMA utilization (channel 0-9)
- Frame synchronization time (via sync function)
- GIF/VIF activity
- Performance counter metrics (e.g. cache misses)
- Interrupts generated
- PC location sampling
- Absolute function timing (via instrumented functions)
- User events

Three main data views are available: *Game*, *Frame* and *Source*:

In the *Game View*, all captured frames are visible and conflicts/bottlenecks can be determined.

In the *Frame View*, detailed information is provided so that you can view the relationship between the captured events.

The *Source View* shows PC Sampling data for each line of source code and disassembly.

Multiple views of different data captures can be viewed at the same time allowing progressive comparison of different versions of your game.

Installation

This section describes how to install (or update) Tuner.

- Tuner is delivered as a ZIP file. Unzip this file to a directory, e.g. C:\Program Files\Tuner for PlayStation2.

If you have purchased this product as part of ProDG Plus then you must install the files to the same folder as ProDG.

Obtain your SN Systems run-time license

All SN Systems products must be licensed. This section explains how you obtain and install your license key file. See also the SN Systems web page at www.snsys.com/Sales/licefaq.htm for product license FAQs.

When you ordered Tuner you would have been provided with a User ID. Go to www.snsys.com/Sales/licensing.htm, click on **Online Activation Centre**, enter your Product User ID in the form provided and click **Submit**. Confirm your company, name and product details, then click the **Issue License** link.

You will be sent an e-mail containing the key file and its contents will be displayed on screen. Place it in the same directory as your Tuner software and the software will be licensed.

System requirements

- Windows 2000 or Windows XP Pro
- Sony Computer Entertainment Inc. PlayStation 2 Development Tool DTL-T10000
- Network card

Supported development tools

- ProDG for PlayStation 2

Supported debug formats

- ECOFF/STABS
- DWARF 1

Technical support and upgrades

First line support for all SN Systems products is provided by the Support areas of our website. To view these pages you must be a registered user with an SN Systems User Name and Password.

- If you have forgotten your User Name and Password, please go to www.snsys.com/Errors/Password/searcher.asp and enter the e-mail address to which your license is registered. The web server will automatically e-mail this address with the relevant account details.

Once you have a valid User Name and Password you can visit our website Support areas at these URLs:

www.snsys.com/support (English)

www.snsys.jp/support (Japanese)

If the answer to your problem cannot be found on the Support areas of our website, you can also e-mail our support team at:

support@snsys.com (English)

i-support@snsys.com (Japanese)

Please make sure that you explain your problem clearly and include details of your software version and hardware setup. If you have been given an SN Systems support log number (LN number) then this should be quoted in all correspondence about the problem.

SN Systems Limited

4th Floor - Redcliff Quay
120 Redcliff Street
Bristol BS1 6HU
United Kingdom

Tel.: +44 (0)117 929 9733

Fax: +44 (0)117 929 9251

WWW: www.snsys.com (English)

www.snsys.jp (Japanese)

[THIS PAGE IS LEFT BLANK INTENTIONALLY]

Chapter 2: How it works

Overview

This section briefly describes how Tuner works and also mentions some factors that you should be aware of to ensure compatibility of your game with Tuner's data collection process.

Frame detection

Frame detection is vital to the operation of Tuner. It uses the principle of frames to format the data on screen and also to synchronise the capture stage. Data is collected on the target until the end of a frame is detected, whereupon it is transferred to the host PC.

The end of a frame is signalled by the occurrence of a *sync function*. This must be a function that is called once per frame. This could be a Sony library call (e.g. `sceGsSyncV`, `sceGsSwapDBuff`) or a user-defined function. Multiple sync functions can be specified to accommodate code that uses a different synchronization method according to the stage of the game that is being run.

Warning: Do not specify multiple functions if they both occur in a single frame. This will cause erratic results such as very short frames.

Game ELF

The ELF file to be analyzed is automatically patched when either the **Download & Capture** or **Download** options are selected. You cannot acquire data from an ELF downloaded directly via ProDG Target Manager or similar software.

During patching, the sync functions are modified to act as frame triggers and the kernel is modified to allow very low overhead data acquisition.

Implications for your game

Several points should be noted:

1. The top 1MB of the available 128MB of memory is used to buffer the captured data for the current frame. Part of the bottom 1MB is also used. These areas should not be used by your code.
2. The Performance Counters should not be used by your code.

3. The COPO Count register can be read, but should not be written to.
4. ELF files should not be stripped of their symbol table. If extended information (such as source level PC sample hit counts) is required, then full debug information should be included.
5. The timing information can be misleading with large sync functions.

```
void Sync(void)
{
    DoTimeConsumingTask();
    DoAnotherTask();
    while (Vblank==false);
}
```

It would be better to have...

```
void Sync(void)
{
    while (Vblank==false);
}

void PreSync(void)
{
    DoTimeConsumingTask();
    DoAnotherTask();
    Sync();
}
```

This would allow you to time exactly how long you were waiting for the Vblank.

6. Problems will occur if your program reboots the IOP while you are trying to capture data. It is best to remove any calls to `sceSifRebootIOP()` when using Tuner.

Detecting if Tuner is running

Sometimes it may be useful to automatically detect whether Tuner is running from within your game code, e.g. to disable certain sections of code that contravene the implication notes above. This can be done by calling:

```
int IsTunerRunning();
```

which will return 1 if Tuner is running, and 0 otherwise.

To use this function you must include the header file `snTuner.h` which contains the definition, and link to `snTuner.lib`. These files are available in the `\TunerDev` subdirectory of your Tuner install directory.

Function list issues

When Tuner loads an ELF file it creates a list of all the functions by gathering details from the debug information and the ELF symbol table. This information is used to identify and patch the 'sync' function. It is also used by the PC Sample 'trace' and the Source View.

Some programming techniques and build tool options will affect the list as follows:

- **Multiple Identical Function Names** – In order to make all function names uniquely identifiable, multiple functions with identical names will have $\{n\}$ appended to the name, where n is an incrementing count.
- **Inline Functions** - In a non-optimized build, functions declared as inline will behave like a standard function. Any PC sample hits will be attributed to it and you will be able to view total hit counts for each line of inlined source. If you are using explicit inline methods in include files, multiple copies of the function name at different addresses will be shown in the list. Only one of these will have been linked-in and all PC sample hits will be attributed to it. Enabling stripping of unused functions in the linker (-strip-unused) will remove all of the additional list entries.

In an optimized build, PC sample hits will be attributed to the parent function that the inline function was called from. Hit counts for each line of source within the inline function will be totalled and will appear on the source line from where it was called, i.e., the code from the inline function will behave in all respects as if it was part of the parent function.

Note that if a global inline function is not defined in an include file, it becomes accessible as a standard non-inline function from outside its module. In this case the function name will still appear in the list and PC sample hits resulting from any such calls will be handled as described above for non-optimized builds. Hits resulting via calls from the same module will be handled as described for optimized builds.

- **Stripped Functions** - If the linker is used to strip unused functions, these will not be added to the list and will therefore be unavailable for features such as **Go To Function**, etc.

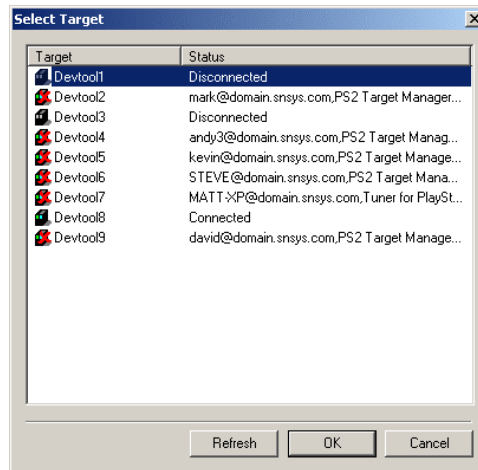
[THIS PAGE IS LEFT BLANK INTENTIONALLY]

Chapter 3: Quick start guide

Capturing data

This section will get you started. It uses the example program blow.elf (`\usr\local\sce\ee\sample\vu1\blow\blow.elf`) from the Sony samples, but any sample program will do.

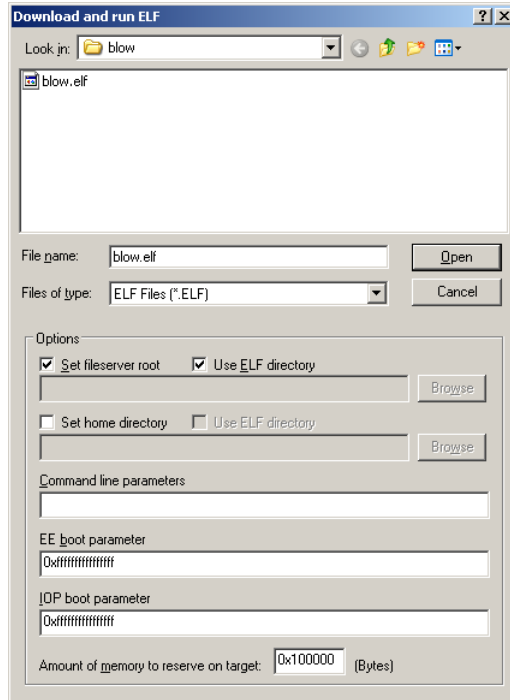
1. Build your ELF (switch debug information on if you wish to view extended function information).
2. Run `ps2tuner.exe`.
3. Select a target using the menu option **Capture > Select Target**.



Note: This should automatically load Target Manager. The target you wish to connect to must be listed in Target Manager.

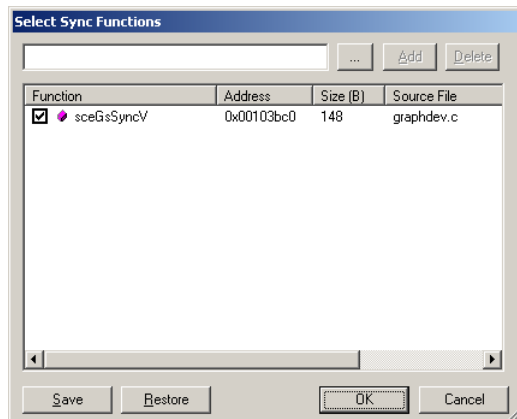
4. Choose **Capture > Download & Capture**.

You will first be prompted for the location of the ELF to load. This can be on the local machine, on a shared network drive, or across a Samba connection (for Linux users).

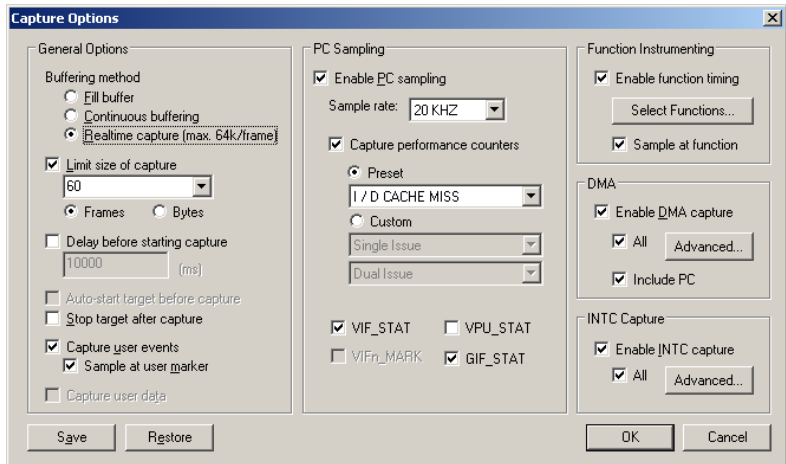


5. Browse to the location of the ELF, ensure that the filename is selected and press the **Open** button.

At this point, the target will be reset and the ELF downloaded. While this is happening, the Select Sync Functions dialog will be shown. You should either manually enter the names of the function(s) you wish to use, or select them from the list of available functions accessed from the [...] button (you can use sceGsSyncV for the program blow.elf):

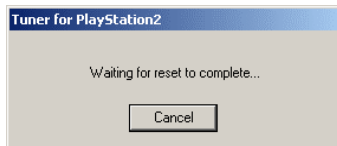


You will then be shown the Capture Options dialog, enabling you to change the sample rate, DMA channels to capture, etc.:

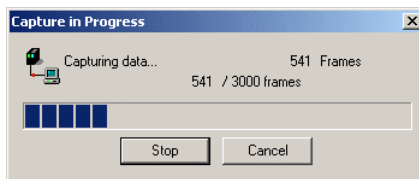


6. Leave the default options selected and click **OK**.

If the ELF has not finished downloading, the following dialog may be displayed:



The ELF will then run and data will start to be collected.



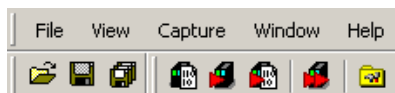
Once your specified capture size limit has been reached or the **Stop** button is pressed, the captured data will be processed and the results displayed.

7. You can save the dataset results by selecting **File > Save**.

[THIS PAGE IS LEFT BLANK INTENTIONALLY]




Chapter 4: User interface

Application toolbar



The main Tuner application toolbar has five pull-down menus (**File**, **View**, **Capture**, **Window** and **Help**), and two moveable toolbars (**File** and **Capture**) that provide quick access to common items.





File menu

 Open	Open a saved dataset file that contains performance data generated from a previous capture (saved files can also be opened by dragging them onto the Tuner application window).
Close	Close the current dataset.
Close All	Close all currently open datasets.
 Save	Save the current dataset.
Save As...	Save the current dataset with a new name.
 Save All	Save all currently open datasets.
Recent Files	Open a recent dataset.
Set Source Search Path...	Add directories to the search path for your game source files.
Exit	Close the Tuner application.

View menu

Workspace View	[This option is not yet available].
Console View	Toggle the visibility of the Console View.
Toolbar	Toggle the visibility of the application toolbar.
Status Bar	Toggle the visibility of the application status bar.

Capture menu

	Download...	Download an ELF with no data capture.
	Capture...	Capture data from an ELF that has already been downloaded via Tuner.
	Download & Capture...	Download the ELF and immediately start capturing data.
	Quick Capture	Capture data according to the values last specified via the Capture Options dialog.
	Select Target...	Specify the Sony PlayStation 2 Development Tool DTL-T10000 that will be used.

Window menu

Cascade	Cascade the Tuner windows so that you can see the title bar of every window.
Tile	Display the windows as non-overlapping tiles.
Arrange Icons	Arrange any minimized windows.

Help menu

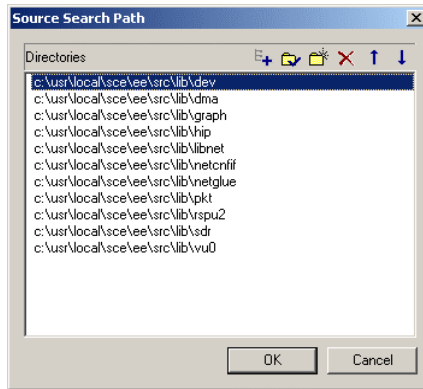
Contents	Display the Tuner help (.chm) file.
About Tuner...	Display information about the application, including the version number.







Set source search path

To be able to display extended information about PC samples, DMA transfers etc., Tuner needs to know the location of the original source files used to create the ELF.

In order to find these, Tuner uses the paths from the debug information contained in the ELF. So, if the source files have been moved to a different location since compile time or are located on a different machine, then a source search path must be set. This allows you to specify a list of alternate paths to be searched.

Selecting **Set Source Search Path...** from the File menu will display the following dialog (the paths of the Sony library source have been added as an example):



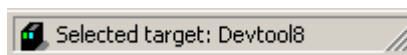
-  **Add Subdirectories** Add all the subdirectories of the currently selected path to the list, (note that this has a depth limit of 5 levels and 50 items per level).
-  **Check Paths** Check that all paths in the list are valid, an option to remove any invalid ones is given.
-  **Add Path** Add a new path to the list (<Insert>).
-  **Delete Path** Delete the currently selected path (<Delete>).
-  **Move Up** Move the currently selected path up in the list (<Alt+up arrow>).
-  **Move Down** Move the currently selected path down in the list (<Alt+down arrow>).

Selecting **Add Path** will add a blank entry into the list. To enter the path either type into the entry and use the auto-complete feature, or click the [...] button to the right of the entry to browse to the required folder. To modify a path double-click on the entry and then follow the steps above.

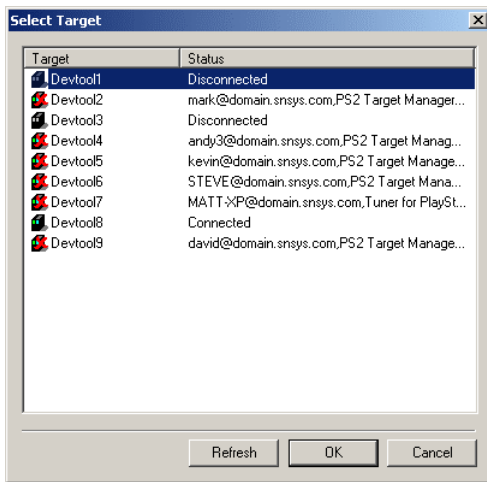
The paths will be searched in the order that they appear in the list. For example, if main.c exists in two of the folders, the first one found will be used. To alter the priority of the paths use the **Move Up** and **Move Down** buttons.




Select target

The currently selected Sony PlayStation 2 Development Tool DTL-T10000 (or *target*) that Tuner will use to download to and capture data from, is shown on the right-hand side of the application status bar. Choosing **Select Target...** from the Capture menu will allow you to change this.



The Select Target dialog lists the current connection status of each target currently listed in ProDG Target Manager.



-  **Unavailable** A red cross indicates that the target is connected to another user and is therefore unavailable for data capture.
-  **Unconnected** A black target icon indicates that it is unconnected.
-  **Connected** A green target icon indicates that it is connected to you.

Select either an unconnected or connected target and click **OK**.

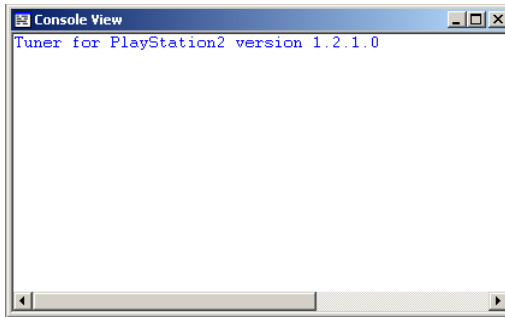
The selected target is remembered in-between sessions. Tuner will connect to it the first time it is used (if available) and will remain connected until either a different target is selected or Tuner is shutdown, where it will restore the previous connection state.

The dialog will be displayed automatically if no target has been specified when Tuner needs to perform a download or capture.

Console View

The Console View displays diagnostic information during Tuner's operation. It provides a history of performed actions and shows further information about errors and warnings.

You can toggle its visibility via the **Console View** option from the **View** menu.



Right-clicking on the Console View will display a context menu containing the following options:

Copy	Copy the currently selected text to the clipboard (<Ctrl+C>).
Select All	Select all text (<Ctrl+A>).
Clear	Clear the text (<Ctrl+R>).
Go To Line...	Move the cursor to a specific line (<Ctrl+G>).
Font...	Set the font (<Ctrl+F>).
Set Log Filename	Set the filename of the log file (<Ctrl+S>).
Log To File	When selected, all output in the Console View will be appended to a log file. Each time Tuner is run and shutdown a timestamp will also be added. If no log filename has been specified then you will be prompted for one (<Ctrl+L>).

Tool Tips

In many of the views and windows, hovering the mouse over an item will display a tool tip containing more detailed information about the subject.

Where this technique can be applied and examples of the kind of data you can expect to see, will be fully described in the appropriate sections of the manual.

[THIS PAGE IS LEFT BLANK INTENTIONALLY]

Chapter 5: Downloading and capturing data

Overview

There are a number of steps that make up the process of downloading a game ELF and capturing performance data from it. Tuner provides several different ways of navigating through these and they are explained in this chapter.

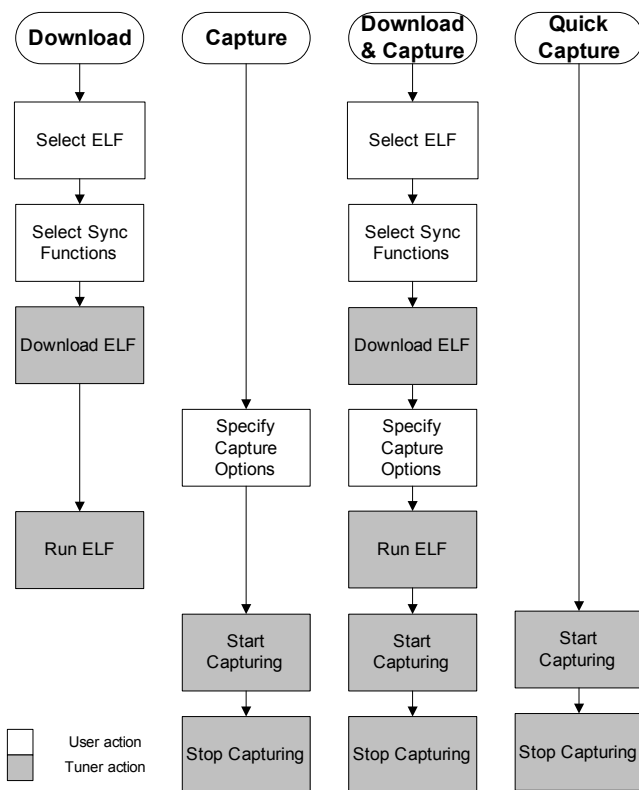
The steps are:

- You select an ELF and specify the 'sync' function(s)
- Tuner patches the ELF, downloads it to the target and runs it
- You choose capture options to specify what performance data is required and start the capture
- Tuner captures data until either some user-defined condition is met or a manual stop takes place

The four navigation routes are:

- Download
- Capture
- Download & Capture
- Quick Capture

Each action takes place on the currently selected target. See "Select target" [on page 15](#) for details on how to change the current target.



The two major concepts here are of *sessions* and *datasets*.

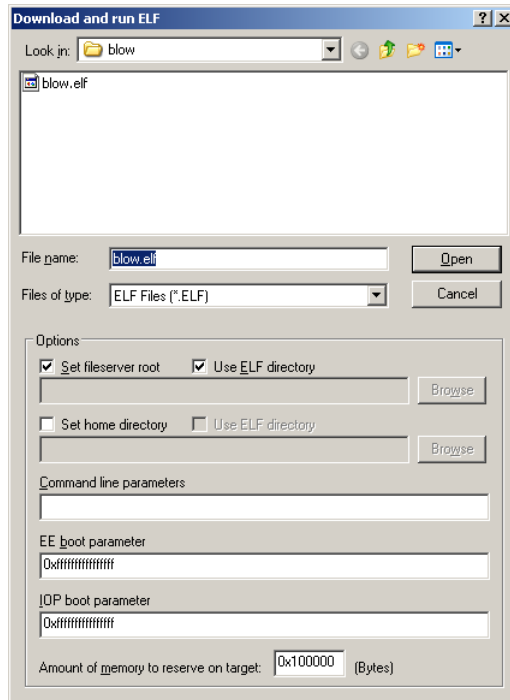
A new *session* is created whenever an ELF is downloaded onto the target (using either **Download** or **Download & Capture**). When creating a new session with **Download & Capture** the first capture is started automatically as soon as the game starts running. Once a session is active, the game can be played to a point of interest and a capture started (using either **Capture** or **Quick Capture**). Multiple captures can be performed on a single session. Each capture produces a set of performance information known as a *dataset*.

If a capture is performed when the session is unknown (e.g. Tuner was shutdown after the download) then you will be prompted for the original ELF *after* the capture has finished. This will allow Tuner to load the debug information required for certain features. If the original ELF is not available, then cancelling the dialog will allow you to continue in a limited mode.

Breakdown of capture steps

Select ELF

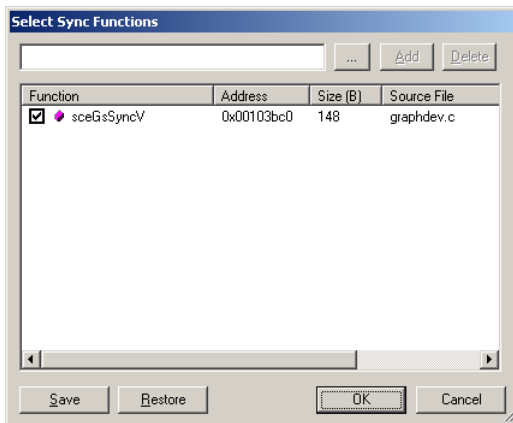
The Download and Run ELF dialog is displayed. Browse to and select the ELF file that you wish to capture performance data for and, if necessary, specify the fileserver root, home directory, command-line parameters, and target reset parameters.



You must also specify how much memory to reserve on the target for storing data when using one of the buffered capture methods. See "Specify capture options" on page 23 for details. The minimum you can reserve is 0.5MB (0x80000) and the maximum is 96MB (0x6000000). The default value is 1MB (0x100000).

Select sync functions

Specify the function(s) that will be used to detect the end of each frame in your game. See "Chapter 12: Function instrumenting" on page 59 for information on what makes a suitable sync function.



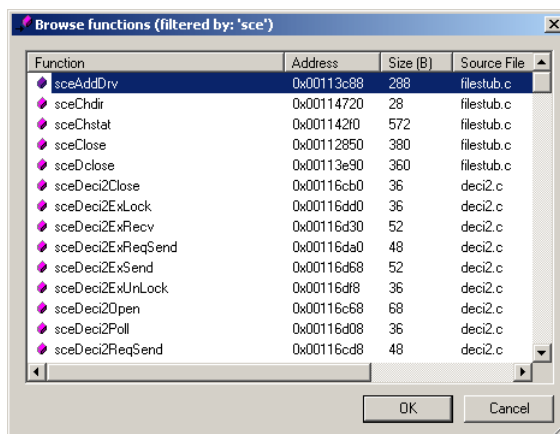
- ... **Filter** Display the function browser dialog filtered on the currently entered text.
- Add** Add the function in the edit box to the list.
- Delete** Delete the currently selected function(s) from the list.
- Save** The contents of the lists are automatically saved as default when **OK** is clicked, however the **Save** button will perform an additional one-off save that can be restored later.
- Restore** Restore the last save.

The edit box allows you to type in a function name. As you type, Tuner will display the best match in a floating tool tip. Pressing <Enter> whilst the tool top is visible will move the name into the list.

The checkbox next to the function name will be shown as checked by default when the function is added to the list.

Note: Only functions that are checked will be patched.

Clicking the **Filter** button or pressing <Space> whilst typing will bring up a function browser dialog filtered on the text already entered in the edit box, allowing you to select a function with the mouse.



Tip: Selecting the filter button when the edit box is empty will display a list of all functions available from the ELF.

Extra information about a function, including its size, address and source file, is displayed if debug information is available.

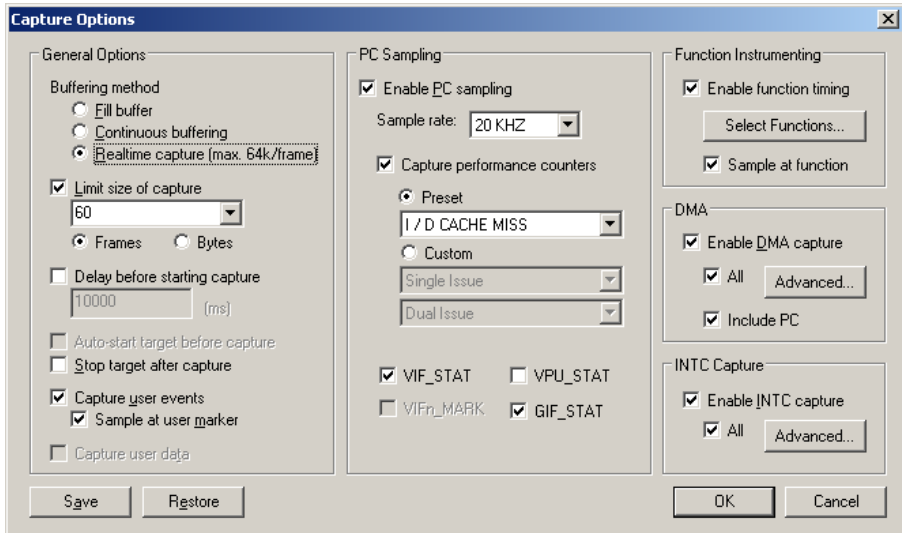
To remove a function from the list, select it and press <Delete> or click the **Delete** button.

Specify capture options

The Capture Options dialog enables you to specify the values that will be used to determine the capture of performance data.

- You can enable DMA capture and then specify that the activity of some or all DMA channels will be captured.
- In the same way, you can also enable INTC capture and PC sampling and specify the sample rate. A more frequent sampling rate will produce more data but may impact the performance of the game.
- You can also include program counter information in the data capture and limit the size of the capture to a specified value.
- The collection of timing information for functions can also be enabled here.
- You can use the **Delay before capture** option to delay capture until the specified time period has expired. This may be useful in cases where you wish to skip capturing data during loading and initialization phases.
- You can also specify that before data capture begins, the target will be started automatically if it is not already running. This is useful when you wish to capture data from the target when it is in a known state, i.e. you want to start data capture as soon as the

target starts. You can stop the target in the debugger, set the required target code state and then start data capture from Tuner. You can also pause the game after data capture has finished by selecting the **Stop target after capture** option.



Buffering method

Specify which capture method you wish to use:

Fill Buffer – Data is captured into memory until the buffer is full when the capture ends and the data is transferred back to the host.

Continuous buffering – Data is captured continuously into memory, wrapping to the beginning of the buffer when the buffer is full. The capture is stopped and the data transferred back to the host when the user presses **Stop** on the capture progress dialog.

Note: The amount of memory reserved for the **Fill buffer** and **Continuous buffering** methods is specified when the ELF is downloaded. See "Select ELF" on page 21 for details.

Realtime capture – Data is transferred back to the host at the end of each frame. The size of each frame is limited to 64KB using this method.

Limit size of capture

Specify the maximum size of the data capture. Uncheck this option for unlimited data capture. Select **Frames** and data will be captured until the specified number of whole frames has been captured. Select **Bytes** and data will be captured until it reaches the specified size.

Delay before starting capture

Note: This option is only available when the **Realtime capture** buffering method is selected.

Specify a delay (in milliseconds) before data will be captured. If you check this option you must specify the number of milliseconds in the box below.

Auto-start target before capture

Note: This option is only available when either the **Fill buffer** or **Realtime capture** buffering methods are selected.

The target will be re-started (when it is not already running), before data capture begins.

Stop target after capture

Stop the target after data capture has completed.

Capture user events

Capture user events. See "Chapter 11: User markers" [on page 55](#).

Sample at user marker

A PC sample and performance counter information will be captured when a user marker is started or stopped.

Capture user data

[This option is not yet available].

Enable PC sampling

Capture profiling data and performance counter data. See "Chapter 13: PC sampling" [on page 65](#).

Sample rate

Specify the rate at which the performance data will be sampled. Tuner sets an interrupt in the running program at that frequency and samples the categories at that moment. There is a trade-off between the program behaving as normal and gathering as much performance data as possible. More frequent sampling may reduce the performance of the running game.

Capture performance counters

Capture performance counters.

You can choose one of the predefined pairs of performance counters, or alternatively you can select any combination of **Custom** performance counters to capture. Some of the predefined options allow you to derive more data than is possible using the **Custom** option. For example, if you select **Load/Store Stalls**, you will see three traces in the Frame View – single issue, dual issue and stalls.

See "Chapter 10: Performance counters" [on page 51](#).

VIF_STAT

Capture VU/VIF activity. The following information will be displayed in the second column of the Frame View for relevant VIF 0, VIF 1 and VIF_STAT activity:

Idle – the VU is idle

Running – the VU is running

VIFCode – the VIF is stalled waiting for data after the VIFCode

Decode – the VIF is decoding the VIFCode

Transferring – the VIF is decompressing / transferring data.

The following special data is also available for VIF 1:

Buffer – the VU is using buffer 1

GIF Wait – the VU has stopped with FLUSH/FLUSHA, DIRECT/DIRECTHL or MSCALF.

See "Chapter 9: VIF/GIF statistics" [on page 49](#).

VPU_STAT

Capture the current state of the Vector Units. The information displayed in the trace is the operation status and is either *busy* (i.e. executing microcode) or *idle*. See "Chapter 9: VIF/GIF statistics" [on page 49](#).

VIFn_MARK

[This option is not yet available]

GIF_STAT

Capture data that shows which paths are active or if the GIF is idle. See "Chapter 9: VIF/GIF statistics" [on page 49](#).

Enable function timing

Collect timing information for functions. Click the **Select Functions** button to select the functions you want to instrument. See "Chapter 12: Function instrumenting" [on page 59](#).

Sample at function

A PC sample and performance counter information will be captured on entry and exit to an instrumented function.

Enable DMA capture

Capture DMA channel performance data. Click **All** to capture data from every available DMA channel or click **Advanced** to specify individual channels via the Select DMA Channels dialog. See "Chapter 7: DMA activity" [on page 45](#).

Include PC

Include program counter details when capturing DMA channel performance data.

Enable INTC capture

Capture INTC performance data. Click **All** to capture every available interrupt option or click **Advanced** to specify individual interrupt options via the Select Interrupts dialog. See "Chapter 8:

Interrupts" [on page 47](#).

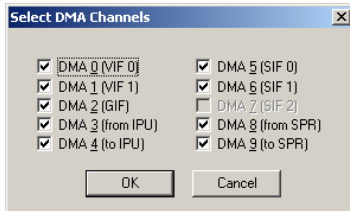
Save

The capture settings are automatically saved as default when **OK** is clicked. However the **Save** button will perform an additional one-off save that can be restored later.

Restore

Restore the last save.

Select DMA channels

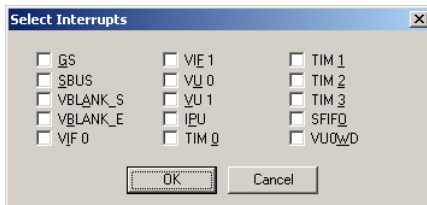


This dialog enables you to specify the DMA channels for which you wish to capture performance data.

It is displayed when you click **Advanced** in the DMA section of the Capture Options dialog. Note that if you wish to specify all the options, select **All** in the same section.

Note: The dialog is only available when **Enable DMA Capture** has been checked on the Capture Options dialog.

Select Interrupts



This dialog enables you to specify the INTC Mask Enable options for which you wish to capture performance data.

It is displayed when you click **Advanced** in the INTC Capture section of the Capture Options dialog. Note that if you wish to specify all the options, check **All** in the same section.

Note: The dialog is only available when **Enable INTC Capture** has been checked on the Capture Options dialog.

[THIS PAGE IS LEFT BLANK INTENTIONALLY]

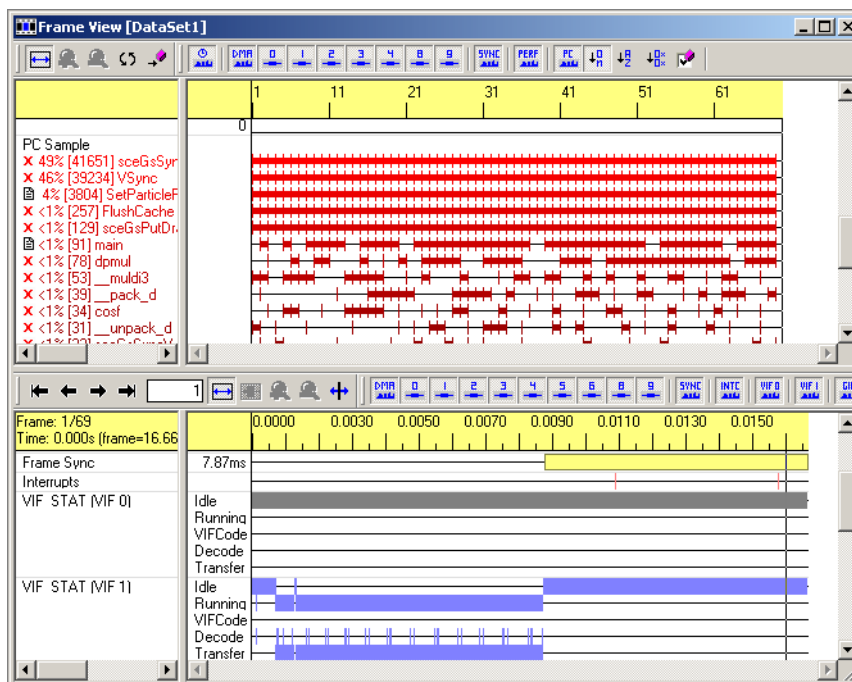
Chapter 6: Viewing and analyzing captured data

Overview

The main tools used for viewing and analyzing captured performance data are the *Game View* and *Frame View*. They both reside in a split pane window and represent a single dataset.

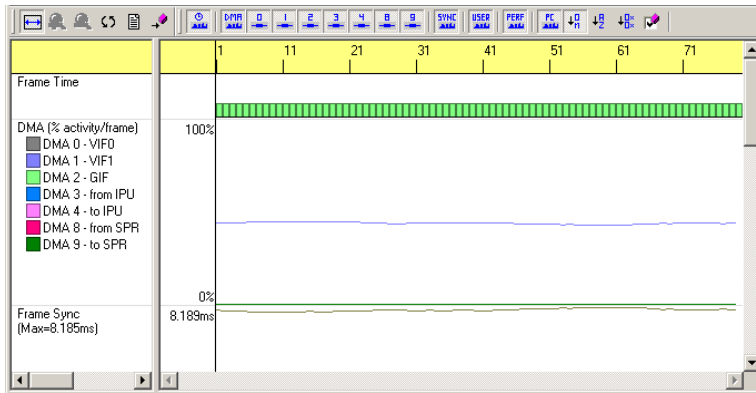
The Game View is the top pane and provides a view of the entire capture giving you a chance to spot general trends and interesting frames that can be examined further in the Frame View.

The Frame View is the bottom pane and shows performance data for a single frame (initially the first frame) against a time scale.



Game View

The Game View is divided into two sub-panes, the left-hand pane displays the category of performance data captured and the right-hand pane shows a graphical display of the actual data against a frame number scale.



Fit To Window

Stretch the scale to fill the size of the window. Single-clicking this button makes the Game View adjust its contents to fit the size of the window. Double-clicking puts the view into auto-fit mode where it automatically fits the range to the window when the size of the window is changed. Single-clicking returns it to non auto-fit mode.



Zoom In

This option is not yet available.



Zoom Out

This option is not yet available.



Rotate Views

Toggle between horizontal/vertical split of the Game View and Frame View.



Launch A Source View

Launch a new Source View for this dataset (see "Source View" [on page 35](#)).



Go To Function

Displays the function browser dialog. Selecting a function will launch a Source View showing the function definition (if available).

Each category along with its associated graphical display is called a *trace*. Each trace adds a moveable toolbar to the frame which allows features such as toggling it on and off, etc.

There is a vertical bar cursor that can be moved by dragging the mouse across the right sub-window. This can be used to mark areas of interest and also to change the current frame shown by the Frame View (see "Frame View" [on page 32](#) for full details).

The following is a list of all Game View traces. Only traces which have appropriate data to display are visible, so depending on the dataset capture options some of these may not always be available.

Frame Time

Indicates the time each frame takes to run one video frame.

Green = 1/50th or 1/60th of a second

Orange = 1/30th or 1/20th of a second

Red = slower than the above.



Toggle the Frame Time trace on/off.

DMA

Displays DMA activity throughout the period of data capture.

Each channel (0-9) is displayed in a different color and is shown by a line in the graph. SIF channels 0 and 1 are not displayed as it is not possible to determine how active they are. SIF channel 2 is not displayed either as it is only used by the Sony kernel.

See "Chapter 7: DMA activity" [on page 45](#).



Toggle the DMA trace on/off.



Toggle the DMA Channel 0 line on/off.



Toggle the DMA Channel 1 line on/off.



Toggle the DMA Channel 2 line on/off.



Toggle the DMA Channel 3 line on/off.



Toggle the DMA Channel 4 line on/off.



Toggle the DMA Channel 8 line on/off.



Toggle the DMA Channel 9 line on/off.

Frame Sync

Shows how much of each frame was spent waiting in the sync function for the frame to end.



Toggle the Frame Sync trace on/off.

Performance Counters

Shows the total count per frame for each performance counter captured. The actual counters are defined in the Capture Options settings dialog. See "Chapter 10: Performance counters" [on page 51](#).



Toggle the Performance Counters trace on/off.

User Markers

Displays a summary of how active each user marker was across the entire capture. See "Chapter 11: User markers" [on page 55](#).








Toggle the User Events trace on/off.

PC Sample

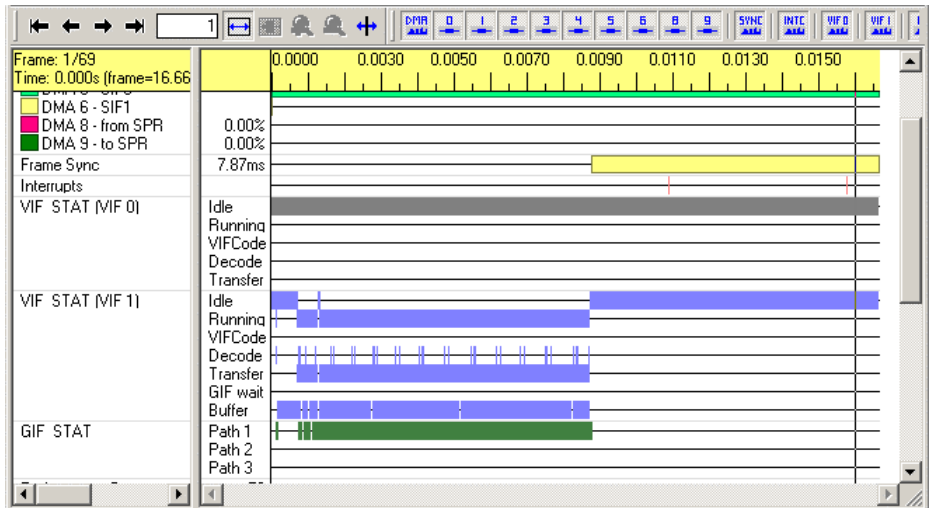
The left sub-pane shows a list of all functions sampled during the capture and contains hit count details. The right sub-pane shows which frames the functions were sampled in. See "Chapter 13:






PC sampling" on page 65.





-  Toggle the PC Sample trace on/off.
-  Sort the function list by hit count.
-  Sort the function list alphabetically.
-  Sort the function list by address.
-  Launch the Function Display Options dialog.

Frame View

The Frame View is also divided into two sub-panes. Like the Game View, the left-hand pane displays the category of performance data captured, but the right-hand pane shows a graphical display of data against a scale of time taken for the frame to complete.



-  **First Frame** Show the first frame (<).
-  **Previous Frame** Show the previous frame (,).
-  **Next Frame** Show the next frame (.)
-  **Last Frame** Show the last frame (>).
-  **Fit To Window** Stretch the scale to fill the size of the window. Single-clicking this button makes the Frame View adjust its contents to fit the size of the window. Double-clicking puts the view into auto-fit mode where it automatically fits the range to the window when the size of the window is changed. Single-clicking returns it to non auto-fit mode.

	Fit To Range	Zoom in on the data displayed between the two Frame View bar cursors.
	Zoom In	Zoom in.
	Zoom Out	Zoom out.
	Synchronise To Game View	Display the frame currently highlighted by the bar cursor in the Game View. Double-clicking this button makes the Frame View continually synchronize with the Game View and vice versa. Whenever the Game View's bar cursor position changes, the Frame View will automatically display the corresponding frame. Conversely, the Game View's cursor will track the currently displayed frame in the Frame View. A further single-click will take the Frame View out of this mode.

To view a different frame number use the navigation arrows on the toolbar or their keyboard equivalents. Alternatively frame numbers may be typed directly into the edit box located to their right. The current frame number along with its length and time location in the capture can be seen at the top of the left sub-window.

The Frame View has two vertical bar cursors that can be used to mark out a range. This is done by clicking and dragging the mouse across the right sub-window. The position of the left bar cursor is determined by the initial click and the position of the right bar cursor by the end of the drag.

Once a range has been defined the view can be zoomed to show this area in more detail by using the **Fit To Range** button.

The following is a list of all Frame View traces. Only traces which have appropriate data to display are visible so depending on the dataset capture options some of these may not always be available.

DMA	Displays DMA activity throughout the period of data capture.
	Each channel (0-9) is displayed in a different color and is shown by a time bar.
	See "Chapter 7: DMA activity" on page 45 .



Toggle the DMA trace on/off.



Toggle the DMA Channel 0 bar on/off.



Toggle the DMA Channel 1 bar on/off.



Toggle the DMA Channel 2 bar on/off.
















Toggle the DMA Channel 3 bar on/off.



Toggle the DMA Channel 4 bar on/off.



Toggle the DMA Channel 5 bar on/off.

		Toggle the DMA Channel 6 bar on/off.
		Toggle the DMA Channel 8 bar on/off.
		Toggle the DMA Channel 9 bar on/off.
Frame Sync		Shows the amount of time spent waiting in the sync function for the frame to end.
		Toggle the Frame Sync trace on/off.
Interrupts		Shows when interrupts occurred. See "Chapter 8: Interrupts" on page 47 .
		Toggle the Interrupts trace on/off.
VIF_STAT (VIF 0)		Shows the state of the VIF_STAT register for VIF 0 every time the PC was sampled. See "Chapter 9: VIF/GIF statistics" on page 49 .
		Toggles VIF_STAT (VIF 0) trace on/off.
VIF_STAT (VIF 1)		Shows the state of the VIF_STAT register for VIF 1 every time the PC was sampled. See "Chapter 9: VIF/GIF statistics" on page 49 .
		Toggle the VIF_STAT (VIF 1) trace on/off.
VPU_STAT		Displays the state of the VPU_STAT register every time the PC was sampled See "Chapter 9: VIF/GIF statistics" on page 49 .
		Toggle the VIF_STAT trace on/off
GIF_STAT		Displays the state of the GIF_STAT register every time the PC was sampled. See "Chapter 9: VIF/GIF statistics" on page 49 .
		Toggle the GIF_STAT trace on/off.
Performance Counters		Shows the value of each performance counter captured over time. The actual counters are defined in the Capture Options dialog. See "Chapter 10: Performance counters" on page 51 .
		Toggle the Performance Counters trace on/off.
User Markers		Time bars for the start and stop of each user marker are shown in the right sub-window. They may be displayed in collapsed or expanded mode. See "Chapter 11: User markers" on page 55 .
		Toggle the User Markers trace on/off.
		Expand the User Marker time bars.
		Collapse the User Marker time bars.
Instrumented		Shows time bars indicating the active time of each

Functions

function that was instrumented in the Capture Options dialog. They may be displayed in collapsed or expanded mode. See "Chapter 12: Function instrumenting" [on page 59](#).



Toggle the Instrumented Functions trace on/off.



Expand the Instrumented Function time bars.



Collapse the Instrumented Function time bars.

PC Sample

The left sub-pane shows a list of all functions sampled during the capture and contains hit count details. The right sub-pane shows where the PC was each time it was sampled over time. See "Chapter 13: PC sampling" [on page 65](#).



Toggle the PC Sample trace on/off.



Sort the function list by hit count.



Sort the function list alphabetically.



Sort the function list by address.

Source View

This view is designed to relate some of the performance data that Tuner captures back to the original code.

It is primarily used by the PC Sample trace to display hit counts for each line of source and disassembly. However, it is also used by several other traces to show, for example, where a particular DMA transfer was initiated or where a user marker was started and stopped.

It can also be used to simply view a text-based file without having to leave the Tuner environment.

```

-- 748
-- 749
-- 750 // start to exit or not.
-- 751 part_id += 3 * NUM_PART + i;
-- 752 if(part_id > frame * PART_BY_FRAME)
-- 753     return;
-- 754
-- 755 // --- calc particle position ---
-- 756 // write blur particle shadow
-- 757 for(n = 0; n < NUM_EXPLODE; n++) {
27 758
-- 759 // --- set position of each blur particle ---
27 760 src_vel[1] += accel[1];
8 761
13 762 pos1[0] += src_vel[0];
-- 763 pos1[1] += src_vel[1];
2 764 pos1[2] += src_vel[2];
-- 765 spos1[0] = pos1[0];
-- 766 spos1[2] = pos1[2];
8 767
6 768 // --- for 2nd pole ---
12 769 pos2[0] = pos1[0] + 15.0f;
-- 770 pos2[1] = pos1[1];
14 771 pos2[2] = pos1[2] + 15.0f;
-- 772 spos2[0] = pos2[0];
-- 773 spos2[2] = pos2[2];
15 774
2 775 // --- for 3rd pole ---
14 776 pos3[0] = pos1[0] + 15.0f;
-- 777 pos3[1] = pos1[1];
2 778 pos3[2] = pos1[2] - 15.0f;
-- 779 spos3[0] = pos3[0];
18 780 spos3[2] = pos3[2];
3 781
3 782 // --- velocity ---
2 783 vel1[1] = src_vel[1] * BLUR_LENGTH;
-- 784 vel2[1] = src_vel[1] * BLUR_LENGTH;
-- 785 vel3[1] = src_vel[1] * BLUR_LENGTH;
3 786
10 787 // --- set scale of shadow ---
3 788 sscale1[0] = 6.0f - pos1[1];
6 789 if(sscale1[0] < 0.0f) sscale1[0] = 0.0f;
4 790 sscale2[0] = 6.0f - pos2[1];
7 791 if(sscale2[0] < 0.0f) sscale2[0] = 0.0f;
-- 792 sscale3[0] = 6.0f - pos3[1];
-- 793 if(sscale3[0] < 0.0f) sscale3[0] = 0.0f;
-- 794
21 795 // for x clip
-- 796 if(pos1[0] > 40.0f || pos1[0] < -40.0f) {
-- 797
Frame 9/60 21.22KB Line 528/896 Col 0/86

```



Open

Open a text-based file (<Ctrl+O>) (files can also be opened by dragging them onto a Source View).



Open Related File

Launch a browser window allowing you to open a file from the list of all the available source files used in the current ELF (<Ctrl+Shift+O>).



Copy

Copy the currently selected text to the clipboard (<Ctrl+C>).



First Frame

Show hit count information for the first frame (<<>).



Previous Frame

Show hit count information for the previous frame (<, >).



Next Frame

Show hit count information for the next frame (<., >).



Last Frame

Show hit count information for the last frame (<>>).



Toggle Location

Toggle between fonts for western and eastern character sets (<Ctrl+L>).







Change Font

Change font for current location.



Change Syntax Colouring

Change the source syntax coloring settings.

	Disassembly Tab	Toggle Disassembly tab on/off.
	Game Hits Border	Toggle Game Hits border on/off.
	Frame Hits Border	Toggle Frame Hits border on/off.
	Line Numbers Border	Toggle Line Numbers border on/off.

A single Source View can show multiple files with each file shown on a separate tab:



You can select a tab either by clicking on it with the mouse or cycle through all tabs by pressing <Shift+tab>.

There are three borders on the left side of the view. These show the number of times the PC was sampled: (a) in the whole capture, (b) in the current frame, and (c) the associated line number (a '-' indicates no hits). These are called the *Game Hits*, *Frame Hits*, and *Line Numbers* borders respectively. The borders can be toggled on and off using their toolbar buttons but the two *hits borders* are only available when the source file is from the current ELF.

You can change the frame for which the Frame Hits border displays hit count information, by using the navigation arrows or their keyboard equivalents. The current frame is shown in the view status bar.

The view is capable of storing two display fonts. These default to western and eastern character sets and they can be toggled via the **Toggle Location** button. To change either of them, select the relevant location and then choose the **Change Font** button. There is also support for files that use the EUC-JP encoding. This feature can be toggled on and off from the view context menu.

Launching a source view

Each Tuner feature that can launch a Source View is described in the relevant section of this manual. The quickest way, however, is to use either the **Go To Function** or the **Launch a Source View** button on the Game View toolbar (See "Game View" [on page 30](#)).

Holding down the <Ctrl> key whilst performing an operation to view source or disassembly will cause a new view to be created. Otherwise the last used Source View is activated and the source file is added. If the source file already exists in that view then the tab containing the file is activated.

If a source file is not available then the appropriate place in the Disassembly tab will be shown.

Moving around the source file

As well as using the mouse to scroll around the source file and select areas of text, there are a number of keyboard controls that offer a finer degree of control:

Up, Down, Left, Right	Move caret.
Ctrl+(Up/Down)	Scroll window up/down.
Ctrl+(Left/Right)	Move caret to start/end of word.
Shift+Mouse Wheel	Scroll one line at a time.
Ctrl+Mouse Wheel	Scroll horizontally.
Shift	Enter 'select' mode.
Page Up	Move up one page.
Page Down	Move down one page.
Home	Move to the start of the file.
End	Move to the end of the file.

Disassembly tab

Each Source View has a single Disassembly tab. Its visibility can be toggled by using the **Disassembly Pane** button.

This shows the address, code bytes and disassembled machine instructions for the entire code. It also shows hit count information for each address location.

Pressing <Space> whilst in the Disassembly tab will switch to the matching place in a source file tab if possible. Likewise, pressing <Space> in a source file tab will switch to the Disassembly tab.

Mixed View

The Disassembly tab can be switched to a Mixed View in which the original source file is interleaved with the disassembly produced for each line, along with the associated hit counts. This lets you see how a total hit count for a line is calculated from hits to the actual machine instructions produced. To toggle Mixed View, select **Show Source Code** (<Ctrl+S>) from the shortcut menu.

Note that in optimized builds the hit count for a line might not equal the sum of the hit counts for its disassembly. This is because parts of the code may have been moved to more optimal positions. When this happens a source line might appear twice in the Mixed View. This is indicated by showing all repeated line numbers and hit counts in a different color.

```

Source View [DataSet1] [Disassembly]
Disassembly | blow.c
-- 376
-- 377
-- 378
-- 379 void CreateViewingMatrix(sceVu0FVECTOR view,
-- 380                          sceVu0FVECTOR interest,
-- 381                          sceVu0FMATRIX persMat) {
1  -- 381
-- 00100AB0 27BDFDE0 addiu sp,sp,0xFDE0
1  -- 00100AB4 7FBE0170 sq fp,0x0170(sp)
-- 00100AB8 7FB70180 sq s7,0x0180(sp)
-- 00100ABC 00C0F02D dmove fp,a2
-- 382
-- 383
-- 384 float viewLength, upLength, rightLength;
-- 385 float fFOV;
-- 386 sceVu0FMATRIX viewMat, transMat, projMat, screenMat;
-- 387 sceVu0FVECTOR viewVector, upVector, rightVector, tapVector1;
-- 388 sceVu0FVECTOR tapVector2;
-- 389
12 -- 390 sceVu0SubVector(viewVector, interest, view);
5  -- 00100AC0 27B70100 addiu s7,sp,0x100
-- 00100AC4 0080302D dmove a2,s0
-- 00100AC8 E7E70218 swcl f23,0x0218(sp)
-- 00100ACC 02E0202D dmove a0,s7
-- 00100AD0 7FB001F0 sq s0,0x01F0(sp)
1  -- 00100AD4 7FB101E0 sq s1,0x01E0(sp)
-- 00100AD8 7FB201D0 sq s2,0x01D0(sp)
-- 00100ADC 7FB301C0 sq s3,0x01C0(sp)
-- 00100AE0 7FB401B0 sq s4,0x01B0(sp)
-- 00100AE4 7FB501A0 sq s5,0x01A0(sp)
-- 00100AE8 7FB60190 sq s6,0x0190(sp)
1  -- 00100AEC FFBF0160 sd ra,0x0160(sp)
-- 00100AF0 E7B60210 swcl f22,0x0210(sp)
-- 00100AF4 E7B50208 swcl f21,0x0208(sp)
1  -- 00100AF8 E7B40200 swcl f20,0x0200(sp)
4  -- 00100AFC 0C041A2C jal sceVu0SubVector
-- 00100B00 AFA50150 sw a1,0x0150(sp)
4 - 391 viewLength = sqrtf(viewVector[0] * viewVector[0] +
-- 00100B04 C7A10100 lwc1 f1,0x0100(sp)
-- 00100B08 C7A20104 lwc1 f2,0x0104(sp)
-- 00100B0C 46010842 mul.s f1,f1,f1
Frame 4/60

```

Context menu

Right-clicking the mouse over the view will launch a shortcut menu with the following options (some options are dependent on the currently selected tab):

- Copy** Copy the currently selected text to the clipboard (<Ctrl+C>).
- Select All** Select all text in the file (<Ctrl+A>).
- Find...** [This option is not yet available (<Ctrl+F>)].
- Go To Global Function...** Launch a function browser dialog showing all functions from the ELF (<Ctrl+B>).
- Go To Local Function...** Launch a function browser dialog showing all functions in the current source file, select one to centre the view on it (<Ctrl+U>).
- Go To Source Line...** Go to a line in the current source file (<Ctrl+G>). This option is only available in Source View and Mixed View modes.
- Go To Frame...** Change the current frame number (<Ctrl+R>).
- Go To Disassembly / Source** Switch to appropriate location in either the Disassembly or a source file tab. (<Space>).
- Go To** Go to a line of disassembly at a given address

Address...	(<Ctrl+G>).
Show Source Code	Toggle Mixed View on/off. Mixed View shows source code interleaved with disassembly (<Ctrl+S>).
Set Tab Size...	Change the number of characters a tab character represents (<Ctrl+T>).
Clone	[This option is not yet available (<Ctrl+N>)].
Properties	[This option is not yet available (<Ctrl+P>)].
EUC-JP Mode	Change the character encoding of the source file to use EUC-JP mode (<Ctrl+E>).
Show Toolbar	Toggle the view toolbar on/off.
Show Status Bar	Toggle the view status bar on/off.

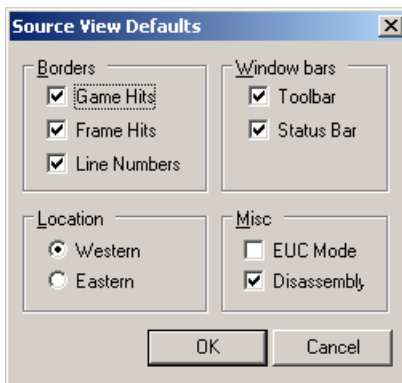
Status bar



The view status bar shows the current frame number, the size of the source file, the location of the caret (line and column) and whether select mode is on.

Default settings

Changes to many settings, including fonts, colors, tab size etc. affect all open and newly created Source Views. Other settings such as borders, location, etc. can be changed on a per-view basis. You can modify the default settings for new views from the Source View Defaults dialog button on the main application toolbar.



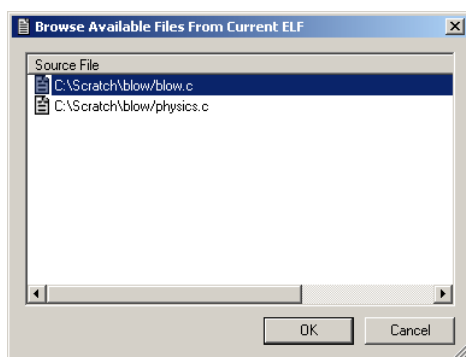
Borders Set whether the following borders are displayed:
Game Hits, Frame Hits, Line Numbers.

Window bars Set whether the toolbar or status bar are displayed.

- Location** Set the default location. Select **Western** if you use the ASCII character set and **Eastern** if you use multibyte character sets.
- EUC Mode** Set whether EUC character translation is on by default.
- Disassembly** Set whether the Disassembly tab is initially visible.

Open related file

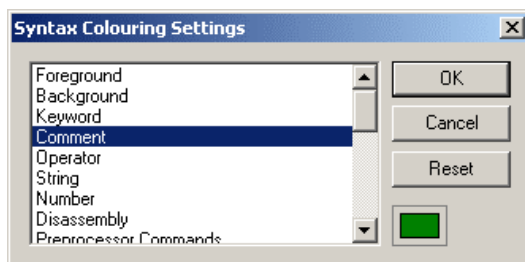
Clicking the **Open Related File** button or using the <Ctrl+Shift+O> keyboard shortcut will allow you to open any available source file from the current ELF.



Change syntax coloring

The view defaults to minimal syntax coloring of source files, highlighting only things such as keywords and comments. However, it is capable of a further degree of granularity with support for coloring things such as numbers, operators, strings, etc.

Clicking the **Change Syntax Colouring** button will launch a dialog so you can fully customise the view's color scheme.



To change a category, select it from the list and then click on the colored square in the bottom right of the dialog. This will launch a picker so you can select the required color. Selecting the **Reset** button will revert all colors back to their original values when the dialog was first opened.

The complete list of categories is shown below:

Foreground	General text.
Background	Background of the main file view.
Keyword	C/C++ keywords.
Comment	C/C++ style comments.
Operator	C/C++ operators.
String	String and character literals.
Number	Numeric characters.
Disassembly	Disassembled machine code.
Disassembly File Names	File name headings in the Disassembly tab.
Preprocessor Commands	C/C++ preprocessor commands.
Selection Foreground	Foreground of the currently selected text.
Selection Background	Background of the currently selected text.
Selection No Focus Foreground	Foreground of the last selected text when the view does not have focus.
Selection No Focus Background	Background of the last selected text when the view does not have focus.
Line Numbers Border Background	Background of the line numbers border.
Line Numbers Border Foreground	Source file line numbers.
Line Numbers Border Repeats	Repeated lines in Mixed View mode.
Frame Hits Border Background	Background of the frame hits border.
Frame Hits Border Foreground	Hit counts in the current frame for source lines.
Frame Hits Border Disassembly	Hit counts in the current frame for disassembled machine code.
Frame Hits Border Repeats	Repeated lines in Mixed View mode.
Game Hits Border Background	Background of the game hits border.
Game Hits Border	Hit counts in the entire capture for source

Foreground	lines.
Frame Hits Border Disassembly	Hit counts in the entire capture for disassembled machine code.
Game Hits Border Repeats	Repeated lines in Mixed View mode.

[THIS PAGE IS LEFT BLANK INTENTIONALLY]

Chapter 7: DMA activity

Overview

Tuner can capture activity on the following DMA Channels:

- **DMA 0 & 1** - VIF 0 (to VU0) and VIF 1 (to VU1)
- **DMA 2 (GIF)** – DMA to the GS processor
- **DMA 3 (From IPU) & DMA 4 (to IPU)** – DMA from and to the Image Processing Unit
- **DMA 5 (SIF 0) & DMA 6 (SIF 1)** – DMA Stop events are recorded
- **DMA 8 (From SPR) & DMA 9 (To SPR)** – DMA between main memory/VU1 memory and Scratch Pad RAM

For each channel, except SIF 0 and SIF 1, an absolute time is recorded for the start and end event for each DMA transfer. DMA Channels SIF 0 and SIF 1 are continuously running, therefore it is not possible to detect when DMA transfers start. Stop events are recorded whenever a transfer completes.

Note: Capture of DMA activity does not use a timer-based sampling technique and so will capture all activity for selected DMA channels.

Capturing DMA activity

This feature is activated by selecting **Enable DMA capture** from the Capture Options dialog (see "Set source search path" [on page 14](#)).

Clicking on the **Advanced** button will display a dialog for selecting which channels you want to capture performance data for.

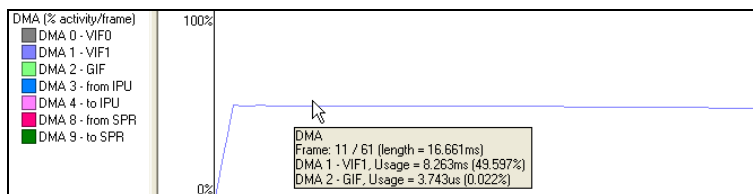
It is also possible to capture the program counter (PC) at the start of each DMA transfer by selecting **Include PC** to determine where it was started from.

Viewing DMA activity data

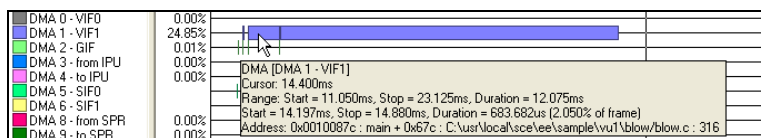
There are both Game View and Frame View traces for DMA activity.

The Game View trace shows a summary of activity for each DMA channel in the form of a line graph (except for SIF 0 and SIF 1). The left sub-window contains the color legend.

Hovering the mouse over the right sub-window shows a breakdown of activity for each channel active in the corresponding frame.



The Frame View trace shows activity for all channels as time bars (except SIF0 and SIF1 where stop events are shown as 1 pixel wide vertical lines). Each bar represents the period in-between the start and end of a DMA transfer. Hovering the mouse over a time bar shows extended timing information for that particular transfer.



Individual channels can be toggled on and off from the traces by using the appropriate trace toolbar buttons.

Launching a source view for a DMA transfer

A Source View showing the line of source code that initiated a DMA transfer can be launched if **Include PC** was selected in the Capture Options dialog. To do this double-click on the time bar you are interested in.

If double-clicking shows a Disassembly tab then the source file cannot be found. Its location should be added to the Source Search Path (see "Set source search path" [on page 14](#)).

Chapter 8: Interrupts

Overview

You can capture absolute timing information whenever an interrupt occurs. The table below gives a summary of the interrupts that can be captured:

GS	Interrupt from GS
SBUS	Interrupt from a peripheral on the SBUS
VBANK_S	Start of vertical blank (V-blank)
VBANK_E	End of vertical blank (V-blank)
VIF 0 and VIF 1	Occurs when VIF(n) detects VIF code with an interrupt bit or exception VIF(n) stalls with occurrence of an interrupt.
VU 0 and VU 1	Occurs when VU(n) executes a microinstruction with an interrupt bit VU(n) stalls with occurrence of an interrupt.
IPU	Occurs when IPU detects end of data or an exception occurs
TIM 0 – TIM 3	Occurs when predefined settings are met for timers 0 – 3
SFIFO	Caused by error detection using SFIFO transfer
VUOWD	Occurs when VU0 is in a run condition continuously for a long time and a ForceBreak has been sent to VU0

Note: Capture of interrupts does not use a timer-based sampling technique and will capture all occurrences of the selected interrupts.

Capturing interrupts

Interrupt capture is activated by selecting **Enable INTC capture** from the Capture Options dialog (see "Specify capture options" [on page 23](#)).

Clicking on the **Advanced** button will display a dialog so that you can select the interrupts for which you want to capture performance data.

Viewing interrupt data

There is only a Frame View interrupt trace. Interrupts are shown as 1-pixel wide vertical lines. Hovering the mouse over a line will detail the type of interrupt(s) that occurred and exact timing information.

Interrupts		
VIF_STAT (VIF 0)	Idle	
	Running	
	VIFCode	Interrupts
	Decode	Cursor: 1.538ms
	Transfer	Interrupt @ 1.616ms: SBUS

Chapter 9: VIF/GIF statistics

Overview

Using timer based sampling, Tuner can sample the status of various VU and GS registers.

The following registers can be sampled:

VIFn_STAT	Provides detailed information about the state of the VUs.
VPU_STAT	Used to determine whether VUs are running or stopped.
GIF_STAT	Provides the status of the GIF (currently this is limited to showing transfers on paths 1, 2 and 3).

Capturing VIF/GIF statistics

VIF/GIF statistic data capture is activated by selecting any of the following from the Capture Options dialog (see "Specify capture options" [on page 23](#)). **VIFn_STAT** and **VPU_STAT** cannot be selected at the same time.

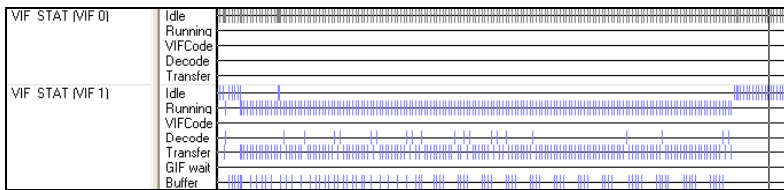
- **VIFn_STAT**
- **VPU_STAT**
- **GIF_STAT**

Note: To capture VIF/GIF data, **Enable PC Sampling** must be selected, however it is also possible to collect VIF/GIF data if either **Sample at user marker** or **Sample at function** are selected.

Viewing VIF/GIF statistical data

VIF/GIF data only has a trace in the Frame View.

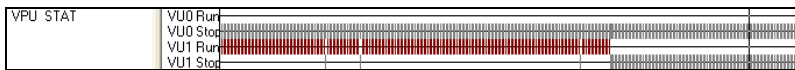
If you have the **VIF_STAT** option enabled two traces are shown for VIF0 and VIF1 respectively.



Selecting the **GIF_STAT** option shows the following trace. At the time of sample it shows which paths to the GIF are active.



Selecting the **VPU_STAT** option shows the following trace. At the time of sample it shows the state of both VU0 and VU1. Hovering the mouse over each sample gives the reason why the VU is stopped.



For more details about these registers and interpreting the data, see the PlayStation 2 technical references.

Chapter 10: Performance counters

Overview

The PlayStation 2 has two performance counters which Tuner can capture, using a fixed interval timer interrupt. Each counter can be independently configured to capture any single event.

The table below gives a brief description of each possible setting for each performance counter. Refer to the PlayStation 2 technical reference for detailed information regarding these settings.

Performance Counter 0

CPU Cycle	Processor cycle (occurs every CPU cycle).
Single Issue	Single instruction issue (occurs when an instruction is issued in only one of the EE CPU logical pipelines).
Branch Issue	Occurs whenever a branch is issued.
BTAC Miss	Occurs when lookup in the Branch Table Address Cache fails.
ITLB Miss	Occurs when a lookup in the ITLB (Instruction Address Translation Lookaside Buffer) fails.
I-Cache Miss	Occurs when a lookup in the I-Cache fails.
DTLB Accessed	Occurs whenever the DTLB (Data address Translation Lookaside Buffer) is accessed.
Nonblock Load	Occurs when a lookup to the non-blocking cache fails due to a load instruction.
WBB Single Req	Occurs when Single Request issued to the WBB (Write Back Buffer).
WBB Burst Req	Occurs when Burst request issued to the WBB.
Address Bus Busy	Occurs when the CPU address bus is unavailable.
Inst Comp	Occurs when an instruction is completed, i.e. it reaches the end of the pipeline.
Non BDS Comp	Occurs when non-branch delay slot instructions complete.
COP2 Comp	Occurs when a COP2 instruction completes.

Load Comp	Occurs when a load instruction completes
------------------	--

Performance Counter 1

Low Branch Issue

CPU Cycle	Processor cycle (occurs every CPU cycle).
------------------	---

Dual Issue	Dual instruction issue (occurs when instructions are issued in both of the EE CPU logical pipelines).
-------------------	---

Branch Miss Predict	Occurs when a branch address prediction is incorrect in conditional branches.
----------------------------	---

TLB Miss	Occurs when a lookup in the TLB (Translation Lookaside Buffer) fails.
-----------------	---

DTLB Miss	Occurs when a miss is recorded for the DTLB.
------------------	--

D-Cache Miss	Occurs when a lookup in the D-Cache fails.
---------------------	--

WBB Single Unavail	Single request to WBB fails due to insufficient free entries.
---------------------------	---

WBB Burst Unavail	Burst request to WBB fails due to insufficient free entries (≥ 5 entries used).
--------------------------	---

WBB Burst Almost	Burst request to WBB fails due to insufficient free entries (5-7 entries used).
-------------------------	---

WBB Burst Full	Burst request to WBB fails due to no free entries.
-----------------------	--

Data Bus Busy	Occurs when the CPU data bus is unavailable.
----------------------	--

Inst Comp	Occurs when an instruction is completed, i.e. it reaches the end of the pipeline.
------------------	---

Non BDS Comp	Occurs when non-branch delay slot instructions complete.
---------------------	--

COP1 Comp	Occurs when a COP1 instruction completes.
------------------	---

Store Comp	Occurs when a store instruction completes.
-------------------	--

Capturing performance counters

Performance counter capture is activated by selecting **Capture performance counters** from the Capture Options dialog (see "Specify capture options" [on page 23](#)).

It is possible to choose either from a list of predefined settings or set up each counter independently by choosing the **Custom** setting.

The most common settings are listed in the predefined dropdown list:

I/D CACHE MISS	Records I-Cache and D-Cache misses.
-----------------------	-------------------------------------

STALLS	Records Single Issue and Dual Issue counts and calculates derived stalls as follows:
---------------	--

$$\text{stalls} = t - (2d + s)$$

where:

t = CPU clock ticks

d = dual issue count

s = single issue count

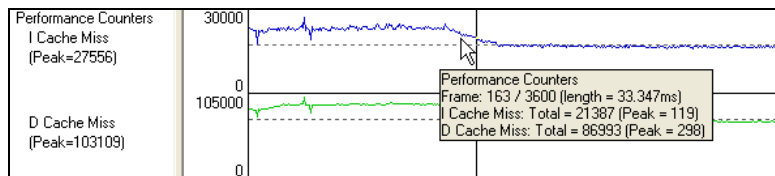
Note: To capture performance counters, **Enable PC Sampling** must be selected, however it is also possible to collect performance counter data if either **Sample at user marker** or **Sample at function** are selected.

Viewing performance counter data

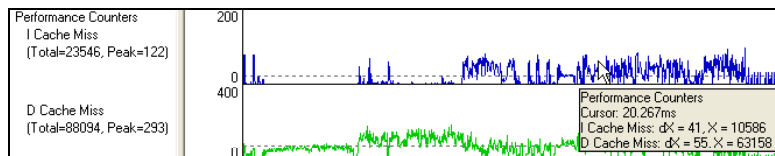
There are both Game View and Frame View traces for performance counter data.

The Game View trace shows the total count for each frame for each performance counter.

The following image shows the Game View trace where the I-Cache and D-Cache miss count were captured. The I-Cache and D-Cache miss rate is higher for the first half of the dataset. The dotted line shows the *mean value* for the entire dataset. Hovering the mouse over the right sub-window shows the actual values for the relevant frame.

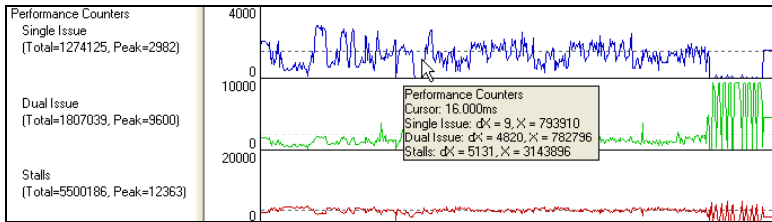


The Frame View trace plots the delta values for each sample (i.e. for each sample $y = \text{current count} - \text{last count}$). This enables peak detection or areas where usage is higher than normal. Hovering the mouse over the trace shows more detailed information about a specific point.



Hovering the mouse over the left sub-window in either view will show information regarding the peaks and mean values.

When selecting **STALLS** from the Defaults list, the derived stalls are shown as a third trace. The Stalls trace shows how many cycles the CPU spends idle/stalling.



Chapter 11: User markers

Overview

User markers are designed to be placed around areas of your code to allow custom profiling/timing of logical units.

Adding user markers to your code

Each marker has an ID within the range 0-31 inclusive and can be given a text string to identify them further. They can be started and stopped using the following function calls:

```
void snStartMarker(unsigned int uID, const char *pText);
```

```
void snStopMarker(unsigned int uID);
```

For example:

```
while(1)
{
    snStartMarker(0, "Render...");

    snStartMarker(1, "Draw Road");
    DrawRoad();
    snStopMarker(1);

    snStartMarker(1, "Draw Cars");
    DrawCars();
    snStopMarker(1);

    snStopMarker(0);

    //wait for VSync...
    sceGsSyncV(0);
}
```

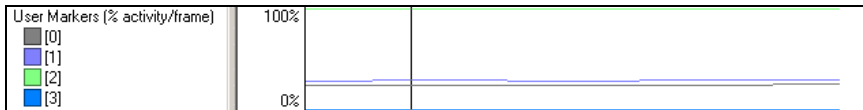
To use the functions you must include the header file `snTuner.h`, which contains the definitions, and link to `snTuner.lib`. These files are available in the `\TunerDev` subdirectory of your Tuner install directory.

Finally, to include these markers in the captured performance data, **Capture user events** must be selected in the Capture Options dialog (see "Specify capture options" [on page 23](#)). For extended information such as performance counter values and PC values, **Sample at user marker** should also be checked.

Viewing user marker data

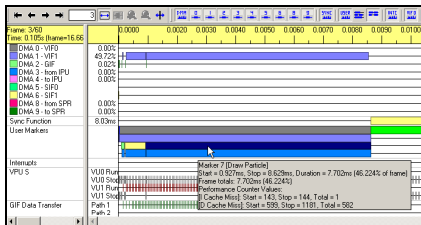
User markers have a trace in the Game View and Frame View.

The Game View trace shows the active ID numbers in the left sub-window and a line graph showing their frame activity percentage across the entire capture. Hovering over an active ID number will show minimum, maximum and average statistics, while hovering over a line will show the exact activity values for that location.

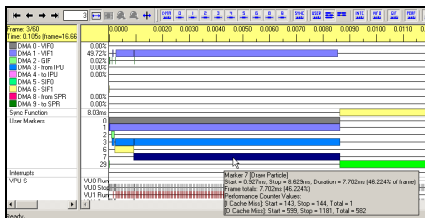


The Frame View trace shows each ID start/stop sequence as a separate time bar.

These bars can be displayed in *collapsed* or *expanded* mode. The following image shows *collapsed* mode, in which markers with different IDs are drawn side by side in relation to time. When a marker start overlaps with another, it moves to the next available line down.



The following image shows *expanded* mode. In this mode each marker with a distinct ID is drawn on its own separate line. Markers with the same ID cannot overlay. IDs that do not contain information are not drawn.



Hovering over a time bar will display further information such as its ID, the identifying text string passed into `snStartMarker()`, timing information, if the **Sample at user marker** capture option was selected, performance counter values and the locations of the start and stop function calls.

User Markers are preprocessed to remove unwanted, multiple start/stop events. These are drawn over the top of the trace as single red or blue vertical lines, representing unmatched start and stop events respectively.

Launching a Source View for a user marker

If the **Sample at user marker** capture option was selected, a Source View showing where the call to `snStartMarker()` or `snStopMarker()` was made can be launched by double-clicking on a time bar. If a time bar has both a start and end within the same frame then a double-click will take you to `snStartMarker()` and a **Shift** and double-click to `snStopMarker()`.

If double-clicking shows a Disassembly tab then the source file cannot be found. Its location should be added to the Source Search Path (see "Set source search path" [on page 14](#)).

[THIS PAGE IS LEFT BLANK INTENTIONALLY]

Chapter 12: Function instrumenting

Overview

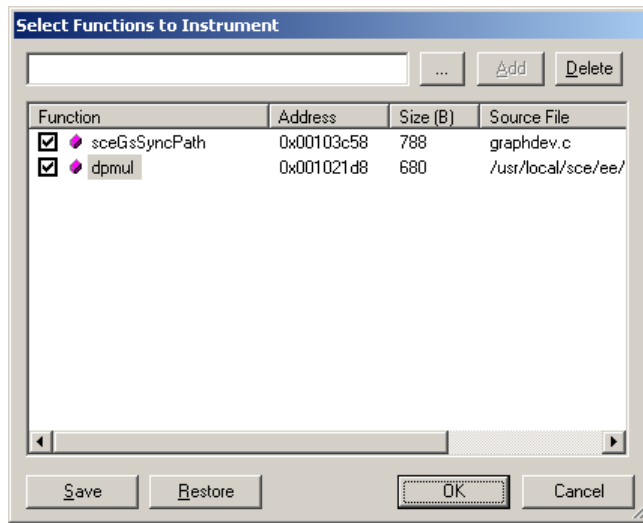
Function instrumenting is a facility that allows you to collect absolute timing information for selected functions during an capture. This feature differs from traditional hierarchical profiling in that only the selected functions are instrumented, thus minimizing the impact upon the normal running of the game.

Additionally, you can choose to take a snapshot of the performance counters at the start and end of a function that is being instrumented. The information returned is dependent on the configuration of the performance counter options. You may, for example, want to determine the total I-CACHE or D-CACHE misses occurring between the start and end of a function.

You can change the functions to be instrumented before each capture from a downloaded ELF. This allows you to determine where problems might lie by refining which functions are being timed in successive captures.

Selecting functions to instrument

1. Select either **Capture** or **Download & Capture** and proceed until you reach the Capture Options dialog (see "Specify capture options" [on page 23](#)).
2. Select the **Enable function timing** checkbox.
3. Optionally, select the **Sample at function** checkbox to enable performance counter information to be returned for the start and end of the instrumented functions.
4. Click the **Select Functions...** button to select the functions you want to instrument. A dialog similar to the one used to select the sync function(s) will be displayed (See "Select sync functions" [on page 21](#)).

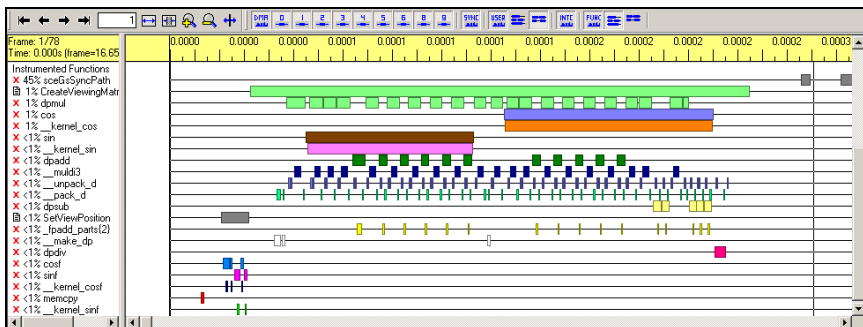


Note: Only functions that are checked will be instrumented.

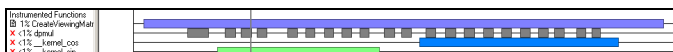
5. Perform the capture as normal.

Viewing instrumented function data

Absolute timing details are shown in the Frame View Instrumented Functions trace as time bars indicating the active period(s) of each function in the current frame. The left sub-window contains the function names and the percentage of total frame time they were active for.



Like the **User Markers** trace, the time bars can be viewed in expanded or collapsed mode. The following image shows expanded mode where each function's time bars appear on a separate line. The functions are sorted in order of consumed time with the most active at the top of the list.

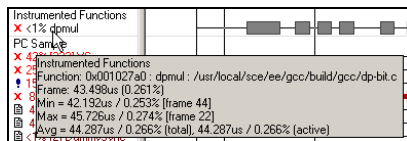


In collapsed mode, the trace shows the hierarchical call order of the instrumented functions. Several functions may appear on the same line, but the color coding matches that of the expanded mode.

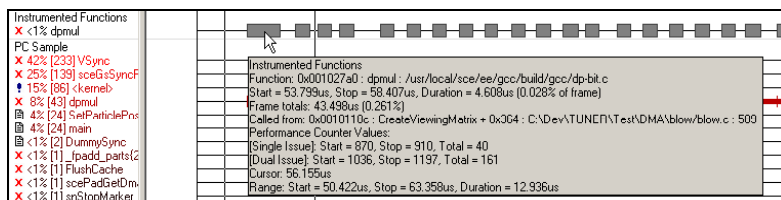


More detailed information can be shown by hovering the mouse pointer over either a function name or time bar.

Hovering over a function name in the left sub-window will show general statistics relating to the total amount of time the function was active for in the current frame, and which frames the function was least and most active.



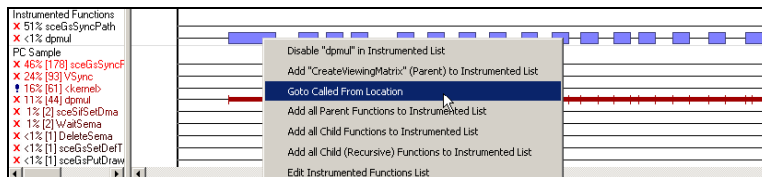
Hovering over a single time bar in the right sub-window will bring up detailed information relating to that period of activity. As well as displaying absolute timing information, if the **Sample at function** capture option was selected, performance counter values will be included.



Launching a Source View for an instrumented function




A Source View may be launched showing the source code of an instrumented function or the calling location.

This can be done by right-clicking the mouse over the function of interest and selecting the appropriate option from the context menu.



Alternatively, you can double-click a time bar or function name to view the source code or hold down **Shift** and double-click to view the calling location.

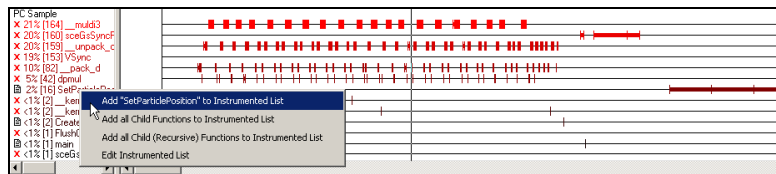
The icon to the left of each function name in the trace shows the availability status of the function's source file.

	Available	The source file is available and may be opened.
	No Line Info	The source file is available but no debug line information is present. It may be opened but the Source View will not be able to go automatically to the function in question.
	Unavailable	The source file cannot be found or is unknown. This location may only be viewed in the Disassembly tab. Setting the Source Search Path to include its location will remedy this (see "Set source search path" on page 14).

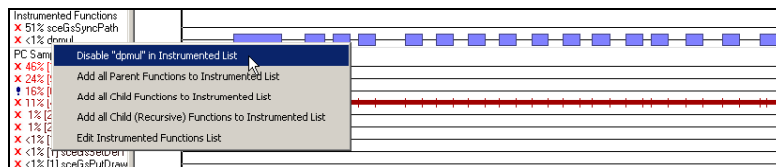
Refining the list of functions to instrument

Once you have performed a capture, you may want to change which functions are instrumented for the next capture. You can do this by adding or deleting them via the Capture Options dialog (see "Specify capture options" on page 23) or by right-clicking on either the function name or a function bar.

Adding and removing functions



If you decide you want to add/enable or disable a function from the list, you can right-click the function in either the Instrumented Functions or the PC Sample trace and choose the appropriate option.



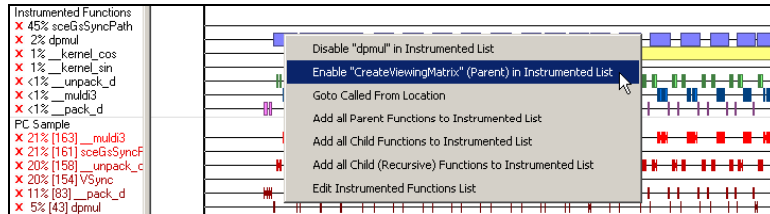
Note: Disabling a function does not remove the function from the list but prevents it from being instrumented in the next capture. Choose **Edit Instrumented Functions List** and select **Delete** to remove functions from the list.

Adding parent and child functions

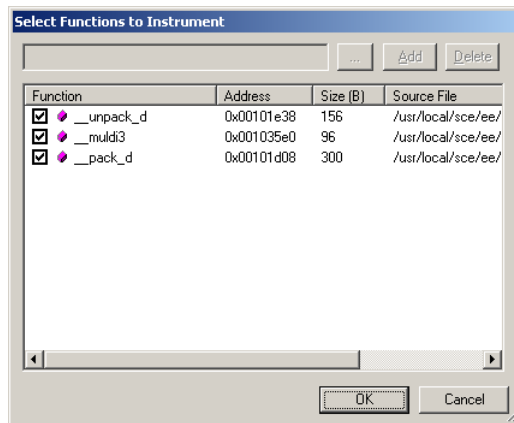
Once you have instrumented a function, you may want to determine which functions called it and how much time *they* took.

To add/enable the parent for a particular instance of the function, move the mouse over the bar and select **Add/Enable <function> (Parent) in Instrumented List**, where <function> is the parent function name.

To quickly add all the parent functions right-click in the Instrumented Functions trace and select **Add all Parent Functions to Instrumented List** from the context menu.



You may also instrument every function that is called from a currently instrumented function by choosing **Add all Child Functions to Instrumented List**. The list of child functions that can be instrumented will be displayed in a dialog so that you can selectively chose which functions will be instrumented.



You can recursively instrument all child functions of a currently instrumented function by choosing **Add all Child (Recursive) Functions to Instrumented List**.

Note: It is not currently possible to instrument a function that is an ascendant of a sync function. This can currently not be detected automatically and may result in the target code crashing.

Chapter 13: PC sampling

Overview

Tuner is capable of performing low-overhead sample based profiling by recording the value of the program counter (PC) at a specified rate. This gives a general overview of which functions are being used the most during a capture period. For more accurate timing information (but with a larger performance impact), the Function Instrumenting feature should be used.

Activating PC sampling

PC Sampling is activated by selecting **Enable PC Sampling** from the Capture Options dialog (see "Specify capture options" [on page 23](#)). The **Sample rate** can be set at 2, 4, 8, 16, 20, 30, 40, 60, or 80 kHz. The higher the sample rate the more accurate the data is, but the performance impact becomes larger.

Viewing PC sampling data

The results of PC Sampling are displayed in traces in both the Game View and the Frame View.

The way the data is displayed is highly customisable through the Function Display Options dialog (see "Function display options" [on page 67](#)). The following description assumes default options.

In both views the left sub-window shows an ordered list of function names that were sampled in the given range (either the entire capture or the current frame). The data that is shown alongside the name is the function's hit count, percentage of the total hit count and the status of the defining source file.

The right sub-window in the Game View shows the frames that each function was sampled at least once in. This is indicated by a vertical bar.

The right sub-window in the Frame View shows every PC sample for each function against a time scale, again indicated by vertical bars. This gives you an approximate picture of how the frame time was spent.

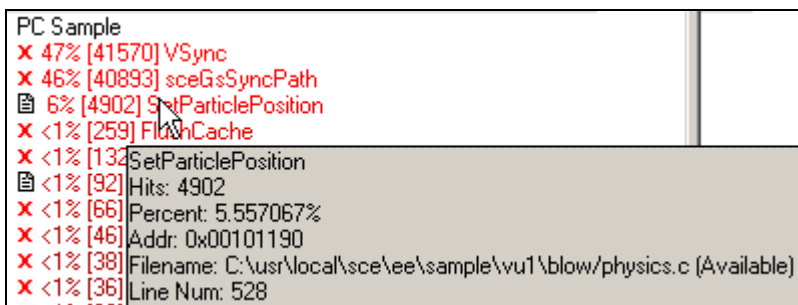
In both views sequential bars are joined together with horizontal bars. This makes trends easier to spot and also improves the usefulness of the trace when the view is highly zoomed in.

The default order that the function names are displayed, is by the number of times they were sampled. This can be changed to alphabetical or address order by using the trace toolbar buttons. Each function and its data are colored using a value on a red-black gradient that represents its percentage of the total hit count (red being higher). This helps to highlight the most important functions when alternate orders are used.

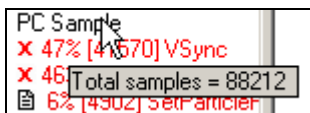
The Frame View trace has an additional feature that allows function names to be 'floated' to the top of the list. To do this select a time range using the two Frame View bar cursors (see "Frame View" on page 32) and all functions that have a hit inside this period will be listed at the top of the trace. A horizontal black line indicates the end of this 'priority' list and the start of the standard list. A single-click in the right sub-window will reset the lists back to a normal state. This technique is useful for marking areas of interest or for keeping function data visible when zooming.

Further information about a function or a sample can be shown by hovering the mouse pointer over certain areas of the trace.

To show detailed information about a function, hover over its name in the left sub-window of either the Game View or Frame View. This will show the function name, number of hits, exact percentage of the total hits, function address, file name and the line number.



If you hover over the PC Sample trace heading in the Game View or Frame View, the total number of samples for the respective amount of time will be shown.

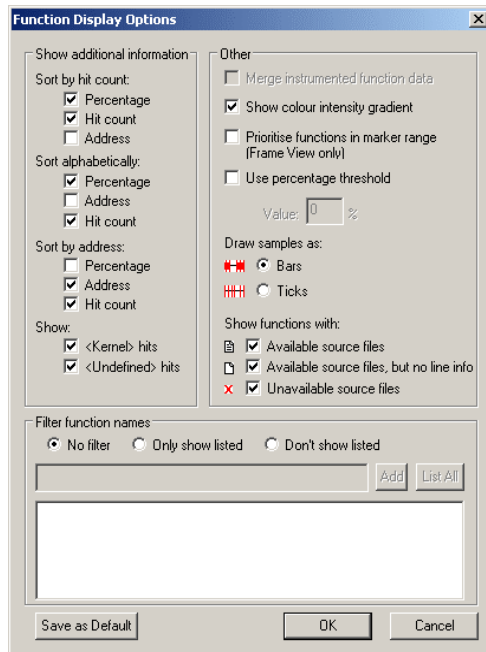


Hovering over a vertical bar in the Game View will show the frame number and the number of hits represented by the line.

Hovering over a vertical bar in the Frame View will display the address of the PC sample and the corresponding file name and line number (if debug line information is available).

Function display options

Selecting the **Function Display Options** button from the Game View trace toolbar (see "Game View" [on page 30](#)) will display a dialog allowing full customisation of how the PC Sample Game View and Frame View traces are displayed.



Sort by hit count

Specify the information that will be shown in the left sub-window when functions are sorted by hit count. Choose from **Percentage**, **Hit count**, and **Address**.

Sort alphabetically

Specify the information that will be shown in the left sub-window when functions are sorted alphabetically. Choose from **Percentage**, **Hit count**, and **Address**.

Sort by address

Specify the information that will be shown in the left sub-window when functions are sorted by address. Choose from **Percentage**, **Hit count**, and **Address**.

Show

Choose whether to show **<Kernel>** and/or **<Undefined>** hits.

Merge instrumented

[This option is not yet available.]

function data

Show colour intensity gradient

Show functions and their data using a red-black gradient representing their percentages of the total hit count.

Prioritise functions in marker range

For the Frame View only, move the functions with hits inside the marker time range into a priority list at the top of the trace.

Use percentage threshold

Specify a threshold so functions with percentage values below this value will not be displayed.



Draw samples as

Choose whether consecutive data is joined together or whether just the individual bars are drawn.

Show functions with

Choose whether to show certain functions based on their source file status. Select from **Available source files**, **Available source files but no line info** and **Unavailable source files**.

Filter function names

Filter the display of functions to only display/not display specified functions. See "Filtering function names" [on page 68](#).

Save as default

Save the options so they will be used automatically the next time you launch a Game View or Frame View that contains the PC Sample trace. Note that the filtered function list is saved with each dataset and will not be made default.

Filtering function names

The **Filter function names** section of the Function Display Options dialog can be used to choose which functions are shown in the PC Sample trace.

To create a list of functions, type their names one at a time into the edit box and click **Add** or press **Enter**. Note that function names will complete automatically as you type. In addition, if you press **Tab** or **Space**, a function browser dialog will be displayed containing all functions starting with the currently entered text. Alternatively, click **List All** to display a function browser populated with all the functions from the ELF.

Check **Only show listed** if you only want to display PC sampling information for the specified functions or **Don't show listed** to exclude the specified functions from the display. **No filter** will ignore the list.

Launching a Source View for a sampled function





If full debug information was present in the downloaded ELF then launching a Source View for a function in the PC Sample trace will show

hit counts for each line, and optionally each line of disassembled machine code. See "Source View" [on page 35](#) for more information about this view.

This can be done in a number of ways:

- Double-clicking on a function name will launch a Source View and take you to the definition line. Doing this from a Frame View will set the Source View to show hit counts for the current frame number, doing the same from the Game View will make it default to the first frame.
- Double-clicking on a vertical tick in the Game View right sub-window will launch a Source View showing the function and hit count information for the frame it represents.
- Double-clicking on a vertical tick in the Frame View right sub-window will launch a Source View showing the exact line that the address of the PC sample corresponds to and hit counts for the current frame.

The icons to the left of the function names in the left sub-windows of both views indicate the source file status.

	Available	The source file is available and may be opened.
	No Line Info	The source file is available but no debug line information is present. It may be opened but the Source View will not be able to go the function in question automatically.
	Unavailable	The source file cannot be found or is unknown. This location may only be viewed in the Disassembly tab. Setting the Source Search Path to include its location will remedy this (see "Set source search path" on page 14).
	Special	Any sample hits that can't be attributed to a function or are in the kernel area of memory will have this icon. No source file can be viewed.

[THIS PAGE IS LEFT BLANK INTENTIONALLY]

Index

- Acquire Options dialog 23
- adding user markers to your code 55
- analyzing data 29
- capturing data 21
- capturing DMA activity 45
- capturing interrupts 47
- capturing performance counters 52
- capturing VIF and GIF statistics 49
- changing instrumented functions 59
- displaying fonts 37
- displaying function names 66
- DMA activity 45
- DMA activity data 46
- Download and Run ELF dialog 21
- downloading and capturing data 19
- ELF files 5
- File menu 13
- function names 68
- filtering display of function names 68
- frame detection 5
- Frame view 32
- Frame view traces 33
- Function Display Options dialog 67
- function list issues 7
- function names
 - displaying 66
- Game view 30
- Game view traces 31
- installing license key file 2
- installing the Tuner 2
 - overview 2
- instrumented functions trace 60
- interrupt data 48
- interrupts 47

- launching a source view for a DMA transfer 46
- launching a source view for a sampled function 68
- launching a source view for a user marker 57
- launching a source view for an instrumented function 61
- license key file 2
- obtaining license key file 2
- optimized builds 38
- overview 2
- parent functions
 - adding 63
- patching ELF files 5
- patching functions 10
- PC sample trace
 - additional information 66
- PC sampling 65
- performance counter data 53
- performance counters 51
- Reference
 - Acquire options 23
 - Download and Run ELF 21
 - File menu 13
 - Function Display Options 67
 - Select DMA Channels 27
 - Select Interrupts 27
 - Source view 35
- refining list of instrumented functions 62
- run-time licenses 2
- Select DMA Channels dialog 27
- Select Interrupts dialog 27
- selecting a target 15
- selecting functions to instrument 59
- setting source search path 14
- Sony samples 9
- source files

- navigation 38
 - related files 41
- source view
 - changing syntax coloring 41
 - context menu 39
 - status bar 40
- Source view 35
- starting Tuner 9
- supported development tools 2
- sync function 5
- syntax coloring 41
- system requirements 2
- target selection 15
- ToolTips 17
- Tuner for PlayStation 2
 - analyzing data 29
 - displaying fonts 37
 - DMA activity 45
 - downloading and capturing data 19
 - filtering display of function names 68
- Frame view 32
- function list issues 7
- game implications 5
- Game view 30
- how it works 5
- instrumented functions 59
- interrupts 47
- PC sampling 65
- performance counters 51
- starting Tuner 9
- supported development tools 2
- system requirements 2
- user markers 55
- viewing console output 16
- VIF and GIF statistics 49
- user marker data 56
- user markers 55
- viewing
 - console output 16
 - DMA activity data 46
 - instrumented function data 60
 - instrumented functions 59
 - interrupt data 48
 - PC sampling data 65
 - performance counter data 53
 - user marker data 56
 - VIF and GIF statistical data 49
 - viewing an area in more detail 33
 - VIF and GIF statistics 49