RadiSys.

Using HawkEye

Version 2.1

Copyright and publication information

This manual reflects version 2.1 of HawkEye. Reproduction of this document, in part or whole, by any means, electrical, mechanical, magnetic, optical, chemical, manual, or otherwise is prohibited, without written permission from RadiSys Corporation.

Disclaimer

The information contained herein is believed to be accurate as of the date of publication. However, RadiSys Corporation will not be liable for any damages including indirect or consequential, from use of the OS-9 operating system, Microware-provided software, or reliance on the accuracy of this documentation. The information contained herein is subject to change without notice.

Reproduction notice

The software described in this document is intended to be used on a single computer system. RadiSys Corporation expressly prohibits any reproduction of the software on tape, disk, or any other medium except for backup purposes. Distribution of this software, in part or whole, to any other party or on any other system may constitute copyright infringements and misappropriation of trade secrets and confidential processes which are the property of RadiSys Corporation and/or other parties. Unauthorized distribution of software may cause damages far in excess of the value of the copies involved.

Contents

1 Getting Started

This chapter provides and overview of HawkEye and how it works, as well as steps to begin running HawkEye. The following sections are included:

- Introduction
- Running HawkEye

Introduction

The HawkEye Graphic User Interface (GUI) provides an execution and visualization tool for the OS-9® operating system. HawkEye allows you to capture and analyze logs of events during the execution of a program on an OS-9 target machine. Possible events may include context switches, interrupts, and system calls (such as process forks, signals, and exits).

HawkEye also enables you to view the interactions among processes in single or multiple applications running on the OS-9 system. To do this, HawkEye evaluates the cause and effect relationships that occur among processes, such as the manner in which one process can signal another to awaken.

Through a TCP/IP connection (as shown in Figure 1-1), HawkEye communicates a specific set of information to a target machine that is running OS-9. The target machine contains the following subsystems, which work in coordination wtih HawkEye to provide the resulting data:

- a system logging module (`slm`)
- a command module (`cmdd`)
- two communications modules (`router` and `loggerd`)

Figure 1-1. HawkEye Software Architecture
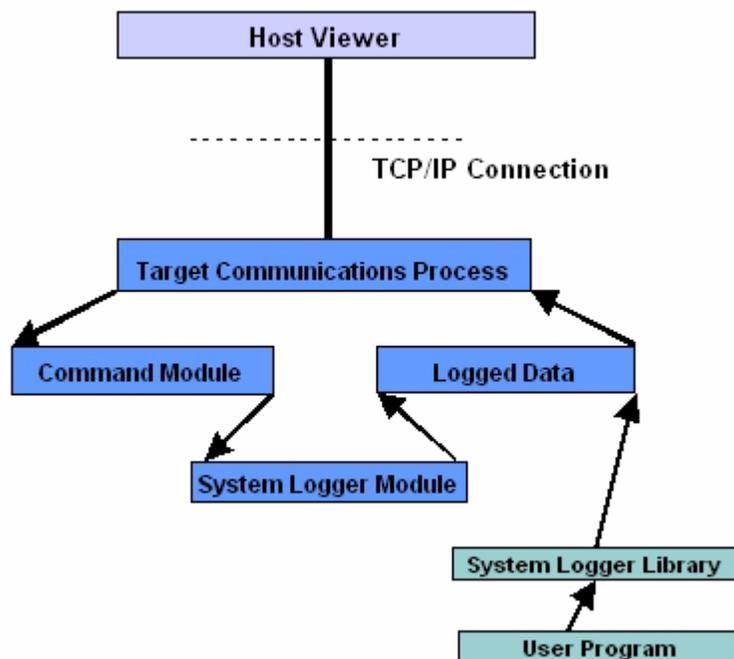
Running HawkEye

This section provides an example session to help familiarize you with HawkEye.

Assumptions

The sections below assume the following information:

- You have OS-9 configured on your reference platform.
- You understand how to use Hawk with application development.
- You have Hawk set up for remote debugging with your target.

Requirements

Host Machine

- an OS-9 development system
- a TCP/IP connection

Target Machine

- a TCP/IP connection configured with OS-9
- the debugger daemons `spfndpd` and `spfndpdc` loaded into memory (with `spfndpd` running)

Setting Up the Target Machine

To set up the target machine, complete the following steps:

Step 1. Open RadiSys Hawk™ from your Windows desktop by selecting `Start -> Programs -> Microware OS-9 for <product> -> Hawk IDE`.

> Refer to the *Getting Started with Hawk* manual (included with your OS-9 product CD) for information on setting up Hawk for development.

Step 2. From the Hawk menu, select `Target -> Load`.

Step 3.   Load the required HawkEye modules onto the OS-9 target machine. These required modules and their pathnames are listed below.

```
<MWOS>/OS9000/<processor directory>/CMDS/router
<MWOS>/OS9000/<processor directory>/CMDS/cmdd
<MWOS>/OS9000/<processor directory>/CMDS/loggerd
<MWOS>/OS9000/<processor directory>/CMDS/slm
```

For x86 Users:

The x86 processor requires different `slm` modules for both MMX and non-MMX systems. The location of the appropriate `slm` module for each system is displayed below.

- MMX processors:
  ```
  <MWOS>/OS9000/80386/CMDS/slmmmx
  ```
- non-MMX processors:
  ```
  <MWOS>/OS9000/80386/CMDS/slm80386
  ```

Once you have loaded the required modules, you have two ways in which to proceed:

- If you are using **ARM**, **MIPS 3000**, or **custom hardware**, proceed to Step 4 for information on loading an additional module.
- If you are using other standard architecture, proceed to Step 5 to start HawkEye.

Step 4.   If you are using ARM, MIPS 3000, or custom hardware, you need to load the high-resolution clock subroutine module, `hcsub`, into memory before HawkEye can run successfully. The process for loading this module varies among these pieces of hardware:

The `hcsub` module can be found in the following location:

```
<MWOS>/OS9000/<processor>/PORTS/<port dir>/CMDS/BOOTOBJS/hcsub
```

If you are porting HawkEye to custom hardware, refer to Appendix A for information on loading the `hcsub` module.

Step 5.   Start HawkEye by typing the following on the command line:

```
p2init slm
router <>>>/nil &
```

Setting Up the Host Machine

To set up the host machine for running HawkEye, complete the following steps:

Step 1.   Start HawkEye by selecting `Target -> Hawkeye` from the Hawk user interface.

Step 2.   From the HawkEye **File** menu, select `Preferences`; the **Preferences** dialog appears. From here, enter the TCP/IP address of your target in the **Host Connection** field.

Step 3.   Click on the **Target** tab and select the processor type of your OS-9 target. When you are done, click `OK`.

Step 4.   From the **Control** menu, select `Trigger` to open the **Trigger Criteria** dialog box. Select the `After Time` radio button, and type `200` in the corresponding field.

Step 5.   Click the `Add` button in the **Current Trigger Criteria** field. `AFTER 200` should now be added to the trigger criteria.

Step 6.   Click on the **Immediately** radio button. This specifies that information should be uploaded immediately after the 200 ticks (two seconds) elapses on the target. Click `OK` to save the selected settings and close the dialog box.

The target and host machines are now configured for HawkEye. Select the `Start Capture` button to record 200 ticks worth of data.

Verifying Registration of the ActiveX Component

As soon as you have started HawkEye, verify that the CharFX ActiveX component, `cfx32.ocx`, (located in `MWOS\DOS\BIN`) is registered with Windows. Registering the component allows you to view detailed charts of your HawkEye captures.

To verify that the ActiveX component is registered with Windows, select the **Tools** menu of the HawkEye interface. There should be three options available in this menu. If none is accessible, `cfx32.ocx` has not been registered with Windows.

If the component is not registered with Windows, complete the following steps to register it manually:

Step 1. Place `cfx32.ocx` in the `windows/system` folder (**Windows 98**)

-OR-

Place `cfx32.ocx` in the `winnt/system32` folder (**Windows NT**)

Step 2. Run `regsvr32 cfx32.ocx` to register it. `regsvr32` should be included with your Windows system.

2 The HawkEye Interface

The HawkEye application window consists of three main parts: the Menu Bar, Toolbar, and Log File Window. This chapter describes these three items of the HawkEye graphic user interface (GUI). It includes the following sections:

- The Menu Bar
- The Toolbar
- Log File Window

The Menu Bar

The HawkEye menus enable you to set filter and trigger criteria when capturing log files, select display options for log files, and set preferences for the application. In addition, keyboard mnemonics provide you with easy access to menu commands. The HawkEye application window displays the following menu bar options:

- **File**: Open and save log files within HawkEye.

    You can also use this menu to configure the system, using the **Preferences** option.

    For information on the **Preferences** option, refer to Chapter 3.

- **Edit**: Perform various clipboard functions.

    Certain rules apply to some Edit menu options. For example, once you copy an event, you can only paste the clipboard selection on a new process line (timeline). In addition, you can only cut and clear events located on a newly-created timeline.

- **Control**: Set trigger, filter, and capture configuration options

    For information on trigger and filter criteria, refer to Chapter 3.

- **View**: Control the way you view the log file window.
- **Tools**: View various charting formats of log files.
- **Window**: Navigate through open log file windows in HawkEye.
- **Help**: Access the on-line help facility and information about HawkEye.

Keyboard Mnemonics

The HawkEye software application employs many keyboard mnemonics. These mnemonics can be found by clicking on any menu item. (The mnemonics appear next to the topics in the drop-down menu list.)

To use the keyboard mnemonic for a command, press the keyboard keys in combinations listed below.

Table 2-1. HawkEye Keyboard Mnemonics

Function	Mnemonic
New	`<Ctrl> + <N>`
Open	`<Ctrl> + <O>`
Close	`<Ctrl> + <W>`
Save	`<Ctrl> + <S>`
Undo	`<Ctrl> + <Z>`
Cut	`<Ctrl> + <X>`
Copy	`<Ctrl> + <C>`
Paste	`<Ctrl> + <V>`
Clear	`<Del>`
Select All	`<Ctrl> + <A>`
Annotate	`<Ctrl> + <E>`
Trigger	`<Ctrl> + <T>`
Filter	`<Ctrl> + <F>`
Start Capture	`<Ctrl> + <R>`
Zoom In	`<Ctrl> + <Keypad Add>`
Zoom Out	`<Ctrl> + <Keypad Sub>`
Expand	`<Ctrl> + <Keypad Mul>`
Collapse	`<Ctrl> + <Keypad Div>`

The Toolbar

The HawkEye toolbar (shown in Figure 2-1) offers quick access to several menu options:

**Figure 2-1. HawkEye Toolbar**

- **New**: Open a new log file document.
- **Open**: Open an existing log file document.
- **Save**: Save the current log file document.
- **Cut**: Cut the current selection from the pasted timeline.
- **Copy**: Copy the current selection.
- **Paste**: Paste the clipboard item to the cursor placement area.

- **Annotate**: View and create annotations for an event.
- **Start Capture**: Start a log capture.
- **Trigger**: Set trigger criteria.
- **Filter**: Set filter criteria.
- **Zoom in**: Zoom in on the current log file.
- **Zoom out**: Zoom out of the current log file.
- **Expand**: Expand all events from the aggregate line that belong to a particular process ID.
- **Collapse**: Minimize any expanded events.
- **interactions**: View the interactions among processes in any given log file.
- **Collapse time**: Condense the information on the screen.
- **Snapshot target**: Place names on processes. This is typically done after the first capture.
- **Event Count Chart**: Open the **Event Class Populations Chart**.
- **Help**: Open the on-line help facility.

Log File Window

The HawkEye log file window (shown in Figure 2-2) is the location of the results from the analyzation operation.

Figure 2-2. Log File Window
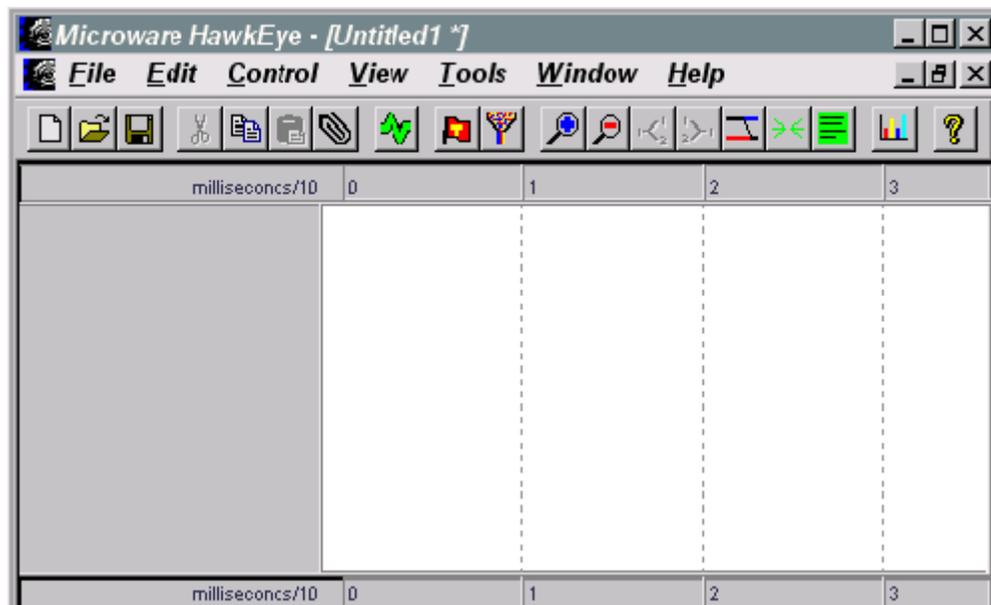
Capture Sessions

"Capture sessions" essentially involve capturing the occurrences between the host and target systems. The results of each capture are visible in the Log File Window.

The following steps can help you get started with capturing events.

Step 1.   In HawkEye, select `Start Capture` from the **Control** menu. This initiates event logging based on the specified trigger and filter criteria. Criteria objects selected for filtering are included during the capture; objects not selected are filtered out.

> For more information on trigger and filter criteria, refer to Chapter 3.

Step 2.   Click on the `Snapshot target` button. This assigns module names to process numbers in the display window.

Step 3.   Once the capture session is initiated, you can cancel by pressing the `Stop` button.

After a capture, HawkEye looks similar to that shown in Figure 2-3.
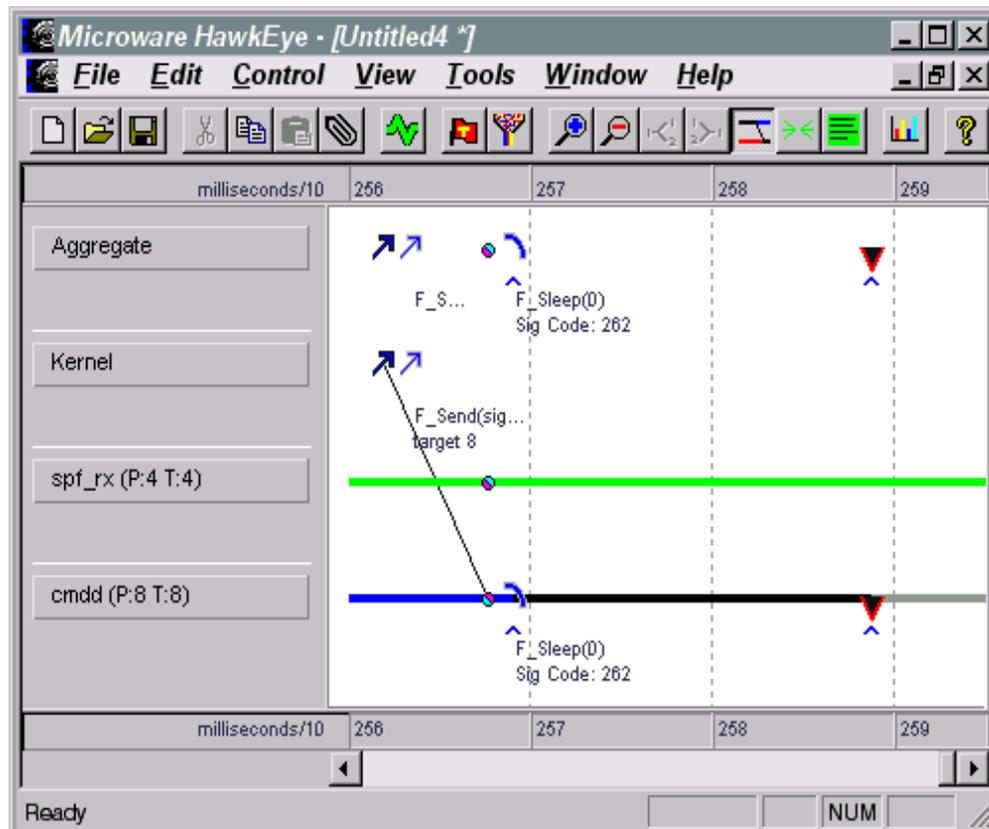
Figure 2-3. Viewed Capture Session
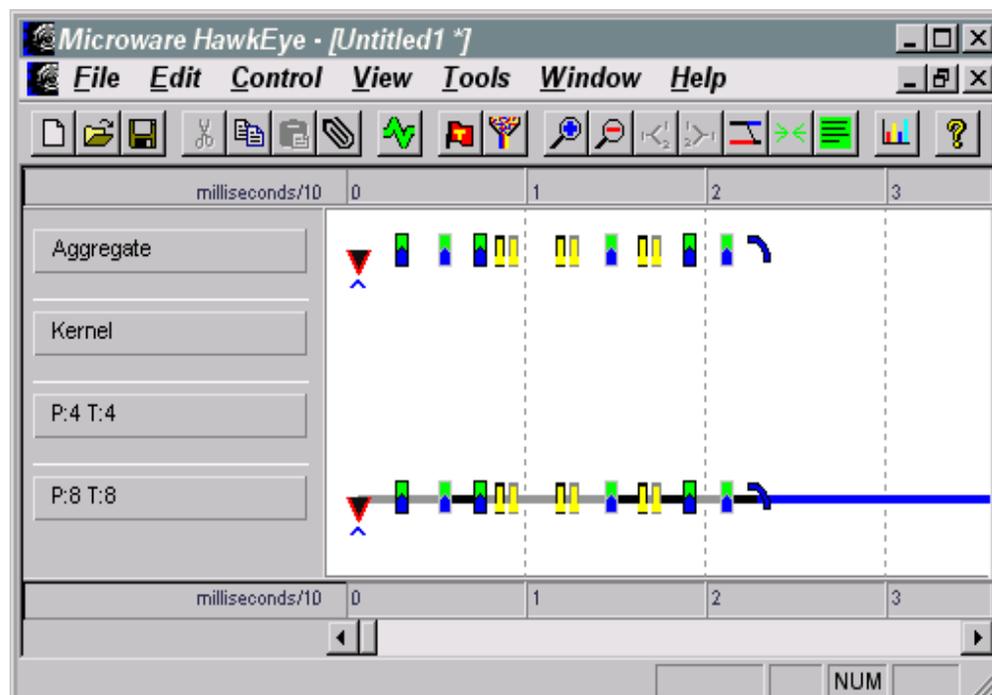
Analyzing the Log File

During a capture session, a cause and effect relationship occurs between processes. The interactions feature of HawkEye maps out the interactions in a log file for you, drawing lines from event to event. This view is shown in Figure 2-4.

Figure 2-4. Log File with Interactions

Each interaction is a tracking of cause and effect events that occur, such as those between signals and processes. To hide interactions, reselect the Interactions option from the View menu.

Aggregate Lines

All captured events are initially displayed in the interface of the log file window on the aggregate line, also known as the "timeline". Thereafter, you can "split out" each process running in the system on separate lines. This is useful if you want to view events from one process on a separate line from those within another process.

In addition, events that are copied from an aggregate lines are always pasted on a new line. Lines in the log file are also colored to inform the user of the state. A list of these lines and their colors is shown below:

Table 2-2. Log File Line Colors

Event Type	Color
Event Wait	light green
Sleep	dark blue
Q	red
Current	black
System State	gray
Process Wait	yellow
Active	dark gray

Expand and Collapse

The Expand menu option expands the log file view for the selected aggregate line event. The Expand feature copies all of the events for each process ID represented in a set of selected events and creates a separate timeline for each process ID in the log file display. Disable Expand by selecting the event timeline you want to remove from the log file window and select Collapse from the View menu.

The Collapse option relieves the log file from an expanded view by removing the expanded process timeline from the log file.

Events Icons

Events displayed in the log file window are displayed as event icons on system lines. Most event types display as two icons: begin process and end process. All available icons are defined in Table 2-3.

Table 2-3. Event Icons

Event	Icon	Possible Details for this Event
Context Switch		Time Stamp, Last PID, Next PID, Process Group, Annotation
Event Signal		Event ID, Time Stamp, State, PID/TID, PC, SP, Annotation

Table 2-3. Event Icons (Continued)

| Event	Icon	Possible Details for this Event
Event Signal Return	↗	Event ID, Return Code, Time Stamp, State, PID/TID, PC, SP, Annotation
Event Wait	↘	Event ID, Time Stamp, State, PID/TID, PC, SP, Annotation
Event Wait Return	↘	Event ID, Return Code, Time Stamp, State PID/TID, PC, SP, Annotation
Exception	⬆	Vector, Level, Time Stamp, Annotation
Exception Return	⬇	Vector, Level, Time Stamp, Annotation
Error Warning	▪	This appears underneath another event to indicate an error warning.
Annotation	⌃	This appears underneath an event to indicate an annotation.
Process Exit	⫾	Exit Status, Time Stamp, State, PID, PC, Annotation
Process Fork	◀	Name, Time Stamp, PID, Annotation
Process Fork Return	◀	Name, Return Code, Child's PID, Child's Module Address, Child's Stack Address, Child's Data Address, Time Stamp, State, PID, Annotation
Interrupt	⬆	Nesting level, Time Stamp, Annotation
Interrupt Return	⬇	Nesting level, Time Stamp, Annotation
Named IO Call	▣	Path Name, Time stamp, State, PID, PC, SP, Annotation
Named IO Return	▣	File Name, Path ID, Return Code, Time Stamp, State, PID, PC, SP, Annotation
Path	▣	Time Stamp, State, PID/TID, PC, SP, Annotation
Path Return	▣	Return Code, Time Stamp, State, PID/TID, PC, SP, Annotation
Resource SVC	◣	Time Stamp, State, PID/TID, PC, SP, Annotation

Table 2-3. Event Icons (Continued)

Event	Icon	Possible Details for this Event
Resource SVC Return		Return Code, Time Stamp, State, PID/TID, PC, SP, Annotation
RTE		Time Stamp, State, PID/TID, PC, SP, Annotation
Semaphore P Call		Time Stamp, State, PID, PC, SP, Annotation
Semaphore P Return		Return Code, Time Stamp, State, PID, PC, SP, Annotation
Semaphore V Call		Time Stamp, State, PID, PC, SP, Annotation
Semaphore V Return		Return Code, Time Stamp, State, PID, PC, SP, Annotation
Signal Call		Signal Code, Target Process, Time Stamp, PID, PC, SP, Annotation
Signal Call Return		Signal Code, Return Code, Time Stamp, PID, PC, SP, Annotation
Sleep		Time Stamp, State, PID/TID, PC, SP, Annotation
Sleep Return		Ticks Remaining, Return Code, Time Stamp, State, PID/TID, PC, SP, Annotation
Generic SVC Return		Return Code, Time Stamp, State, PID/TID, PC, SP, Annotation
Generic SVC		Time Stamp, State, PID/TID, PC, SP, Annotation
Wait Call		Time Stamp, State, PID, PC, SP, Annotation
Wait Call Return		Child PID, Child Exit Status, Return Code, Time Stamp, State, PID, PC, SP, Annotation
User Event		Message, Time Stamp, State, PID, Annotation

Analyzing Details

Each event possesses a set of details that are created after a session has been captured. Double-clicking an event icon opens the **Details** dialog (shown in Figure 2-5) for that event.

Figure 2-5. Details Dialog Box
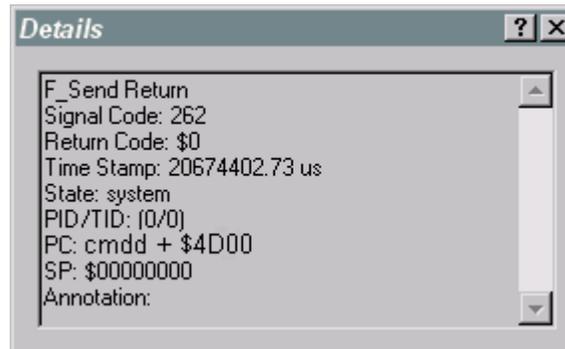
The **Details** dialog that appears after you double-click an event icon provides specific event data, depending on the situation of the event. Analyzing information in the **Details** dialog is vital for determining the results of the capture session.

The following descriptions apply for much of the information that appears in the **Details** dialog:

- **Time Stamp**: Time stamp of the log entry in microseconds from the start of the trace.
- **State**: The ASCII data that Hawkeye provides regarding the state from which the call was executed.
- **PID**: The process ID from which the system call was executed.
- **PC**: The return PC for the system call.
- **SP**: The stack pointer for the system call.
- **Annotation**: The annotation text created via the Annotations feature.
- **Label**: The name of the system call.
- **Signal Code**: A standard OS-9 signal displayed as decimal and ASCII.

- **Ticks Remaining**: A returned value resulting from varied system call returns.
- **Return Code**: The value returned by OS-9 system calls.

> **Hints for reading details**:
>
> - All hex numbers are prefixed with a "$".
> - The time stamp is a real-time stamp.
> - If the system clock does not give submicrosecond resolution, the time stamp will be inaccurate. For instance, if the system clock is based on a 0.01 second system ticker, it will show time in milliseconds with 10 millisecond resolution.
> - Hawkeye handles high-resolution counters that wrap. When a counter rolls over, Hawkeye will notice this and maintain all system information in the proper order. In addition, the time stamp on each event will be modified by adding one clock period for each roll over of the high-resolution times. However, Hawkeye cannot detect more than one rollover.

Comparing Events

HawkEye allows you to compare events different ways:

- Open multiple log file windows simultaneously.
- Create logs with events copied from current or other log files.

Once you have found a preferred way of viewing log data, analyze the data by viewing interactions and details as described earlier.

You can perform event comparison by viewing the log file and details on specific events; you can also manipulate events via clipboard functions.

Displaying Events in Different Formats

An additional feature of HawkEye provides is the ability to view the log file in different ways. The following sections provide ways in which you can view the log file.

Icons Only

With the **Icons Only** label view (shown in Figure 2-6), the events display only as icon graphics. No additional labels or text data are displayed.

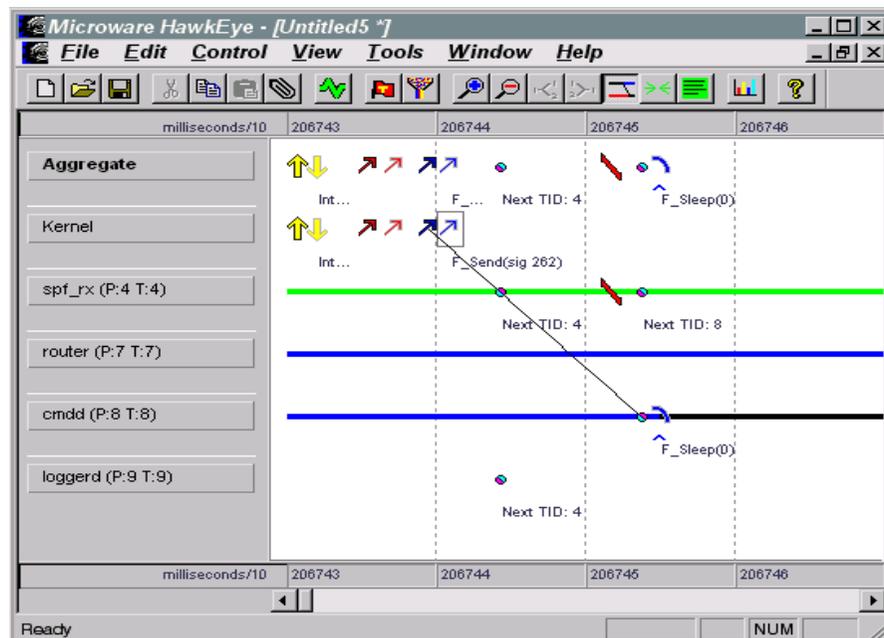Figure 2-6. Icons Only Label Display
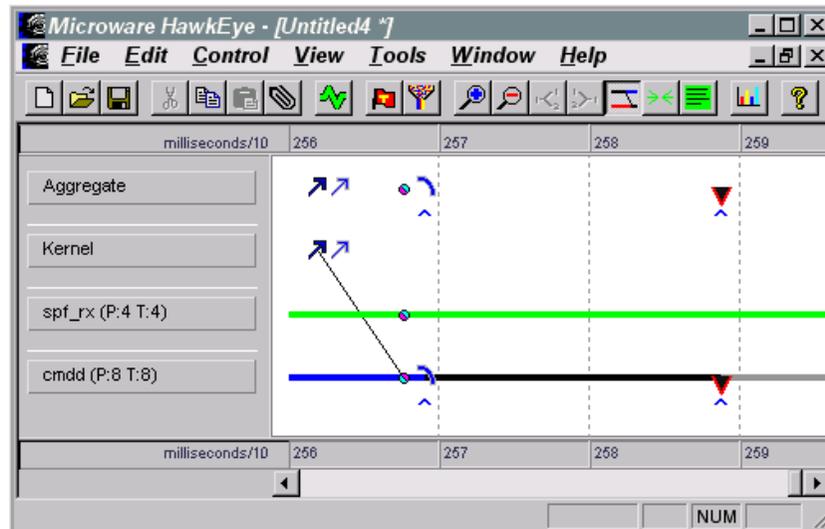
Simple Labels

With the **Simple Labels** label option (shown in Figure 2-7), you can view events in a log file as small, brief labels only.

Figure 2-7. Simple Label display

Using HawkEye

The **Simple Labels** selection displays only one line of descriptive text data below each event in the log file window.

Simple labels are constrained to the horizontal space available. The portion of a label that cannot be displayed is truncated with an ellipses to inform you that additional data exists.

Full Labels

With the **Full Labels** label view (shown in Figure 2-8), the events in a log file display as more descriptive labels.

Figure 2-8. Full Label Display
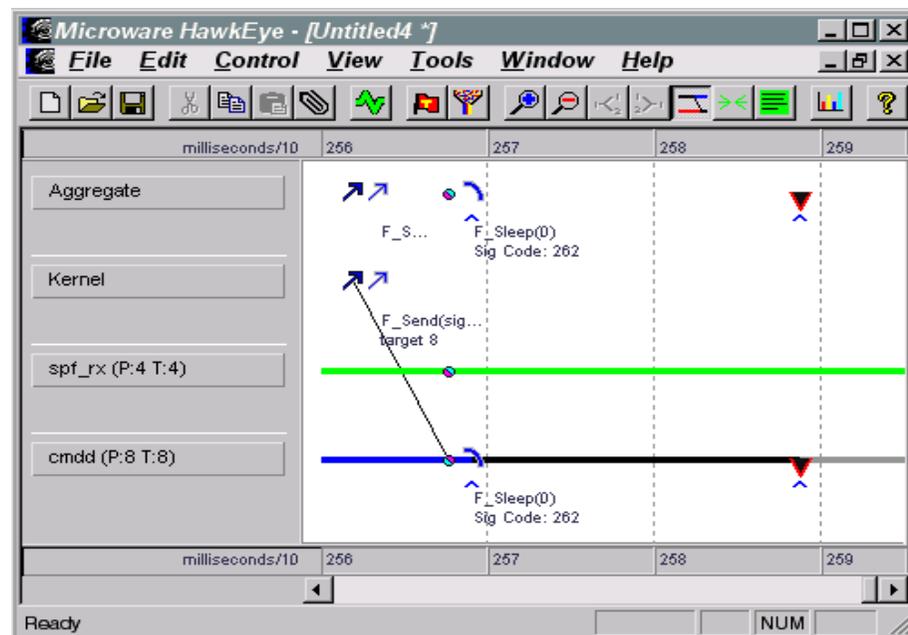
The **Full Labels** selection displays two lines of descriptive data below each event in the log file window.

Full Labels are also constrained to the horizontal space available and are truncated with an ellipsis when the entire label that cannot be displayed.

Automatic Labels

In the **Automatic** label view (shown in Figure 2-9), the events in a log file automatically display.

Figure 2-9. Automatic Label Display
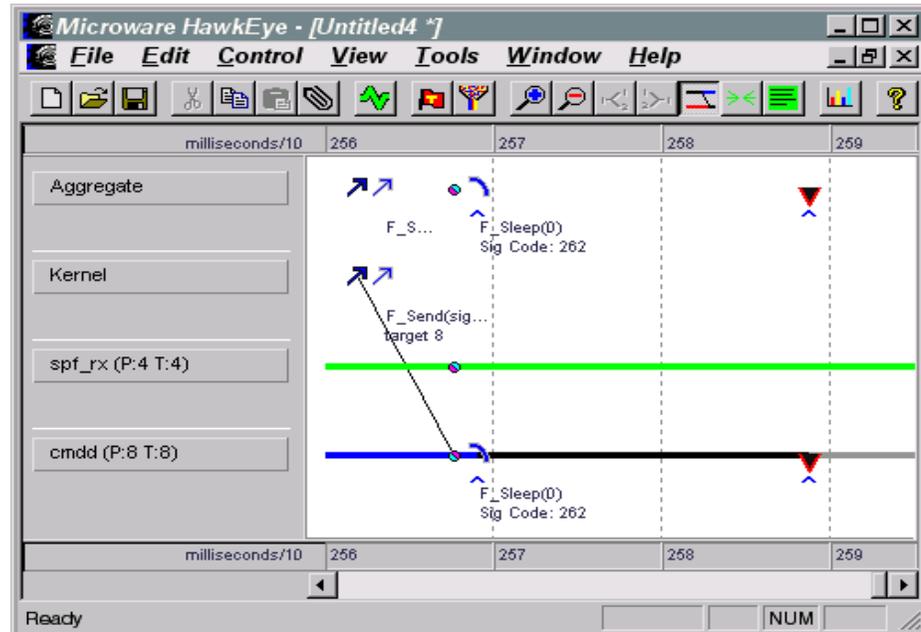
The **Automatic** selection displays zero, one, or two lines of data below each event. The number of lines displayed is selected automatically based on the amount of individual vertical space available. The larger the display view, the greater number of process lines that are visible. Automatic labels are also constrained to the horizontal space available and are truncated with an ellipsis if the entire label cannot be displayed.

Magnified View

The **Magnifying Glass** menu options enable the magnifying glass feature. When turning on the magnifying glass for the first time, HawkEye automatically magnifies the x-axis of the log file to **2x**. Thereafter, the 2x, 4x, 8x and 16x sizes are available.

To turn off the Magnifying Glass, de-select **Magnifying Glass** from the **View** menu.

The Magnify 2x magnifies the log file 200% larger than the original size. You can additionally magnify the log file to Magnify 4x (400%), Magnify 8x (800%) and Magnify 16x (1600%).

3 Using the HawkEye Interface

This chapter details different aspects of working in HawkEye. The following sections are included:

- User Event Logger
- Customizing the View
- Trigger and Filter Criteria
- Log File Charts

User Event Logger

The User Event Logger is a mechanism for programs to insert events into a HawkEye log. Applications need to link against a special HawkEye library and include a special HawkEye header file:

`MWOS/SRC/DEFS/LIB/slmlib.h`

For MIPS3000 Users:

User Event Logging is not available for MIPS3000 processors.

User Event Logging

Using User Event Logging, you can put extra data in the log file that is specific to the program you are on running the target machine.

To use the User Event Logger, HawkEye must be set up to trigger and/or filter specific user events. In addition, you must write a program to the target that makes calls to the User Event Logger library (`slmlib.l`).

Configuring the Host Machine for Event Logger

To configure your host machine for use with the Event Logger, you need to perform two steps:

1. Specify triggers (if any) for specified user events.
2. Specify the user events to be logged (filtered).

Configuring the Target Machine for the Event Logger

To configure your target for use with the Event Logger, two steps must be performed:

1. Write a C program using the User Event Logger library (`slmlib.l`) to log when user events have occurred. Each event will be specific to your program.

2. As your program runs, a system event log is created and user events are logged to the system log. These events are graphically displayed on the host system with the rest of the system events.

> A sample program called `slmtest` is included with this product; it demonstrates the use of `slmlib` and user triggers.
>
> When the user logs are merged with the system log, they are merged based on time stamps. HawkEye's limit of 255 seconds of log applies to the total log, from the first entry in any log to the last entry in any log.

User Log Entries

User log entries annotate each log with events that are normally invisible. User log entries can also be used as triggers. The functions below, found in `slmlib.l`, are included as user log entries:

- `hawk_control_log(event, note)`

  This function logs a text user entry. The event is recorded with the specified user event number and an ASCII string of up to 31 characters. The event number is a 16-bit value (unsigned short). The entire 16-bit value is logged onto the target and displayed on the host. The low-order 8 bits can be used as trigger values and the host can filter the display of user events based on the low-order 8 bits.

- `hawk_control_log2(event, note, ptr1, ptr2, pointer1, pointer2)`

  This function logs a mixed text and numeric user entry. This works similarly to a log entry from `hawk_control_log()`; however, this function accepts four numbers. These can include any combination of pointers and numbers, however, HawkEye attempts to decode `ptr1` and `ptr2` as pointers, and displays `number3` and `number4` as numbers.

hawk_control_log()
Adds User Log Entry to System Activity Log

Syntax

```
#include <slmlib.h>
error_code hawk_control_log(
                           const     int event,
                           char      *const note);
```

Attributes

Operating System: OS-9

State: User and System

Description

`hawk_control_log()` adds a user log entry to the system activity log maintained by the `slm` system module.

> `hawk_control_log()` will return an unknown service error if `slm` is not currently installed in the system.

The event number can be any number in the range 0 to 65535. Larger numbers will be truncated to 16 bits before they are transmitted to the host. (However, the system may record all 32 bits at a future revision.) The low order eight bits of the event number may be used as a "user event" trigger. There are no pre-set numbering conventions for user events, but there is the possibility that triggers might be set on particular event values.

`note`

> an ASCII string up to 31 characters long
>
> This is recorded in the event log and passed without modification to the host.

Errors

`hawk_control_log()`   Return `EOS_UNKSVC` if `slm` is not installed.

`hawk_control_log()`   Return a memory access or protection error if the pointer to `note` is bad.

hawk_control_log2()
Adds User Log Entry to System Activity Log

Syntax

```
#include <slmlib.h>
error_code hawk_control_log2(
                        const   int event,
                        char       *const note,
                        const      u_int32 ptr1,
                        const      u_int32 ptr2,
                        const      u_int32 n1,
                        const      u_int32 n2);
```

Attributes

Operating System: OS-9

State: User and System

Description

`hawk_control_log2()` adds a user log entry to the system activity log maintained by the `slm` system module. It is an extended version of `hawk_control_log2()`.

> `hawk_control_log2()` will return an unknown service error if `slm` is not currently installed in the system.

The event number can be any number in the range 0 to 65535. Larger numbers will be truncated to 16 bits before they are transmitted to the host. (However, the system may record all 32 bits at some future revision.) The low order eight bits of the event number may be used as a user event trigger. There are no pre-set numbering conventions for user events, but there is a possibility that triggers might be set on particular event values.

`note`

> an ASCII string up to 31 characters long
>
> This is recorded in the event log and passed without modification to the host.

`ptr1` and `ptr2`

> passed to the HawkEye host code without modification
>
> The host attempts to decode these as pointers.

`n1` and `n2`

> passed to the HawkEye host code without modification
>
> The host displays these as numbers.

Errors

`hawk_control_log2()`	Return EOS_UNKSVC if slm is not installed.
`hawk_control_log2()`	Return a memory access or protection error if the pointer to note is bad.

Customizing the View

There are any number of ways to configure a view of the HawkEye GUI. The following sections explain options for configuring your view.

Tick Compression

The Tick Compression menu option (detailed in Figure 3-1) reduces empty sections of the logged time to small tick marks on the log file display.

Figure 3-1. Log File with Tick Compression
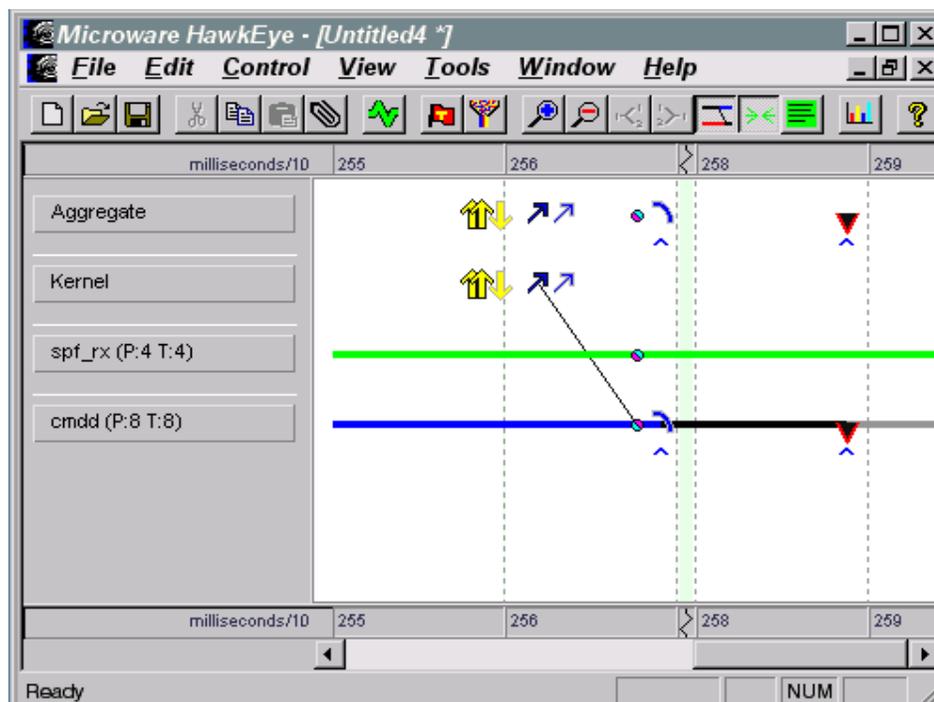
Tick compression eliminates empty space within the log file; this enables you to view more useful event data without having to scroll through lengths of the log file window. In addition, you can click the Collapse time toolbar button to filter out empty ticks.

Annotations

The **Edit** menu allows you to specify annotations for a selected event. This is done by selecting the Annotate menu option. This option brings up the Annotations dialog box (shown in Figure 3-2).

Figure 3-2. **Annotations Dialog Box**
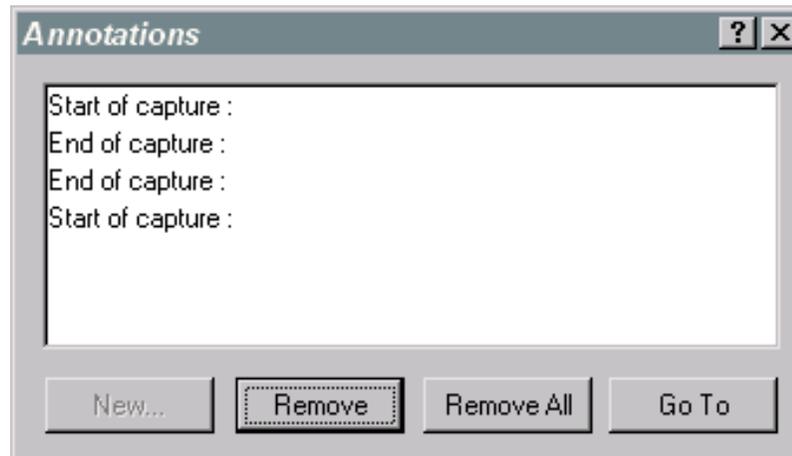
The **Annotations** dialog box enables you to add descriptive text to an event. Annotation options allow you to complete any of the following tasks:

- Select `New` and the **New Annotation** dialog appears, allowing you to create a new annotation for a selected event.

- Remove the selected annotation from the Annotations list box and selected event by selecting the **Remove** button; select `Remove All` to remove all annotations.

- Select the `Go To` button to move to the event of the selected annotation; this event should now be visible in the log file window.

Preferences

Selecting the **Preferences** option from the **File** menu opens the **Preferences** dialog. This dialog is where you specify host and target preferences.
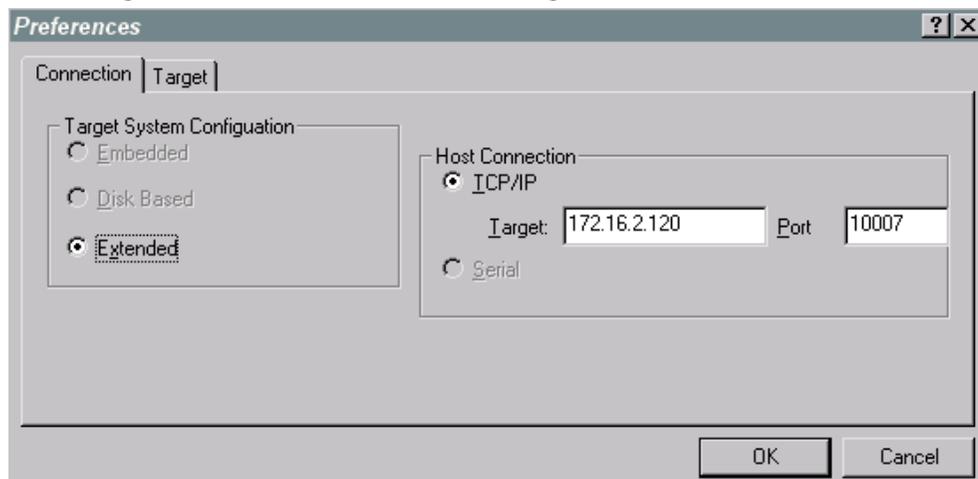
Connection

The **Preferences** dialog opens with the **Connection** tab displayed (shown in Figure 3-3). This tab is the location of the target and host connections. There are three options for configuring the target system, including the embedded, disk-based, and extended options.

> This release of HawkEye only supports the extended OS-9 installation option.

Figure 3-3. Preferences Dialog Box - Connection Tab

The **Host Connection** section of the **Connection** tab specifies the connection type. Host connection options include TCP/IP and Serial. A TCP/IP connection selection dictates that the target machine is accessed via a network connection.

> This release of HawkEye only supports the TCP/IP connection option.

The **Host** field should be modified to contain the IP address of the target; this must be a numerical address. In addition, it is recommended that you use the default value for the **Port** text box.

Target

The second tab in the Preferences dialog is the Target tab (shown in Figure 3-4).

Figure 3-4. Preferences Dialog Box - Target Tab
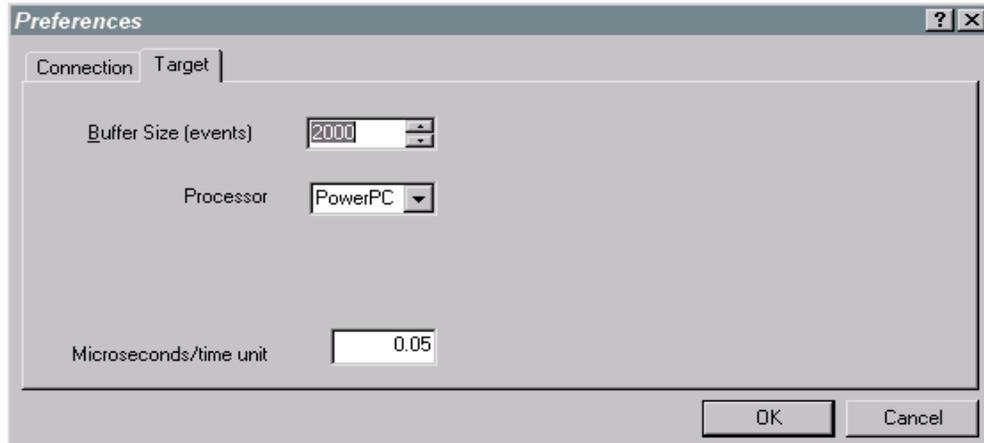
The **Target** tab is the configuration site for the event buffer size. The buffer size can range from zero to 32,767. The buffer size option sets the number of events that the target logs. The buffer size control is manipulated by the either control spinners or user entry.

In addition, the **Target** tab is the location in which you select your processor and the number of microseconds per display time unit (the milliseconds time units displayed in the log file window).

The resolution and range of the target clock is measured by slm and returned with the trace information from the target. This overrides any information placed in the preferences/target/target clock field, but the user can re-override that value after the log has been collected.

Trigger and Filter Criteria

HawkEye allows you to set trigger and filter criteria and analyze certain results of the log file to diagnose potential problems and inconsistencies of the software running on the OS-9 system. In addition, controlling trigger and filter conditions allows you to capture information surrounding specific criteria of interest and to view the data without cluttering the log file with extraneous data.

Trigger Criteria

To set trigger criteria, select `Control -> Trigger` from the main menu. This brings up the **Trigger Criteria** dialog (shown in Figure 3-5).

Figure 3-5. Trigger Criteria Dialog Box
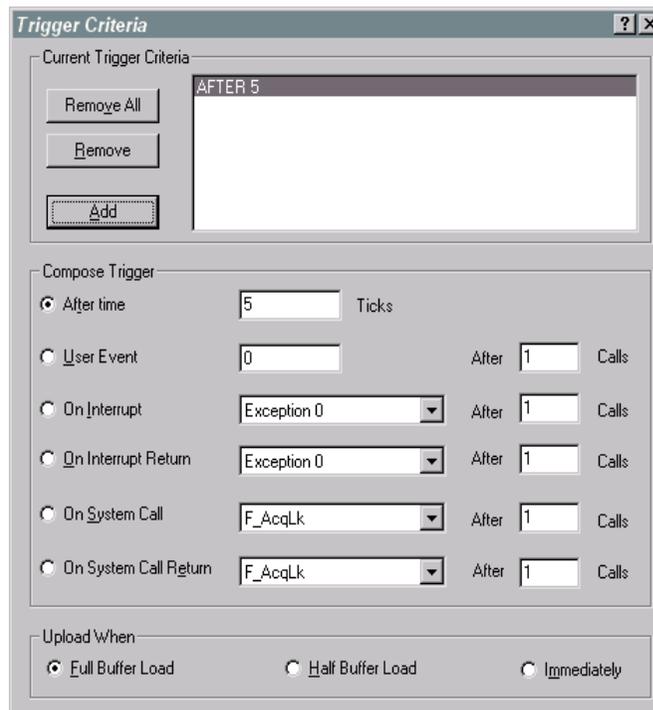
This dialog is the specification site for the actions that trigger the capture session. When one of the following trigger criteria is met, logging begins:

- following a designated number of target system ticks
- following a designated number of user events
- on interrupts or interrupt returns
- on system calls or system call returns

In addition, you can specify the timing of log file uploads:

- Select `Immediately` to upload when the trigger event occurs. HawkEye records system information after the **Start Capture** button is selected, and sends the data when the trigger condition is met.

- Select `Half Buffer Upload` to upload when the trigger event is in the middle of the log. HawkEye records all system information until the trigger condition is met, then records half of a buffer more of data before sending it to the host.

- Select `Full Buffer Upload` to upload when the event is at the beginning of the log. HawkEye only starts to record data when the trigger condition is met. Then it records one buffer's worth of data.

Manual Triggers

You can perform a manual trigger during a capture. The **Trigger** button allows you to send a manual trigger to the target system. This has the same effect as if one of the trigger conditions was met in the **Trigger** window.
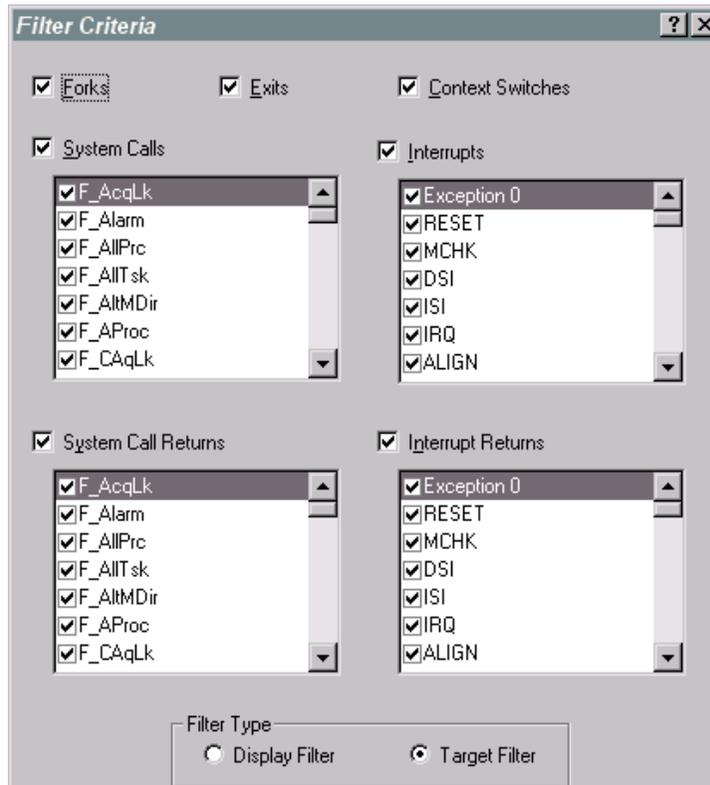
Figure 3-6. Capture Window
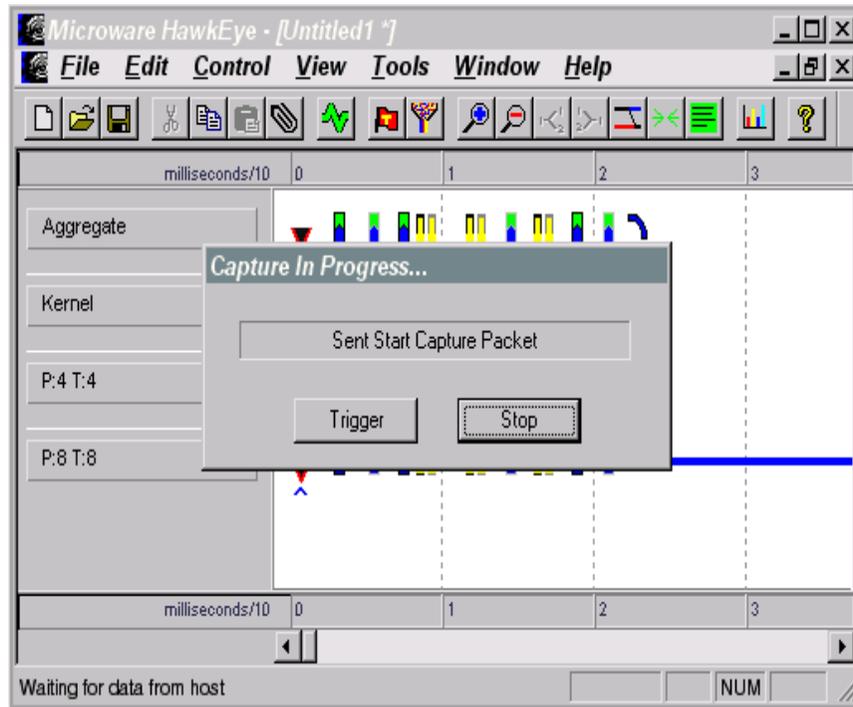
Filter Criteria

To begin specifying filter criteria during event captures, select `Control -> Filter` from the main menu. The **Filter Criteria** dialog is displayed (shown in Figure 3-7).

Figure 3-7. Filter Criteria Dialog Box

The **Filter Criteria** dialog box is the location for filtering actions in the capture session. Filter criteria can include any of the following items:

- forks
- exits
- context switches
- system calls
- interrupts

In addition, you can specify which filter type to use:

- Select `Display Filter` to select filter options that are applied at display time.
- Select `Target Filter` to select filter options that are applied at capture time.

Log File Charts

Chart options are found under the **View** menu. Each chart displays a different view of the events in the current log file. These charts are detailed in the following sections.
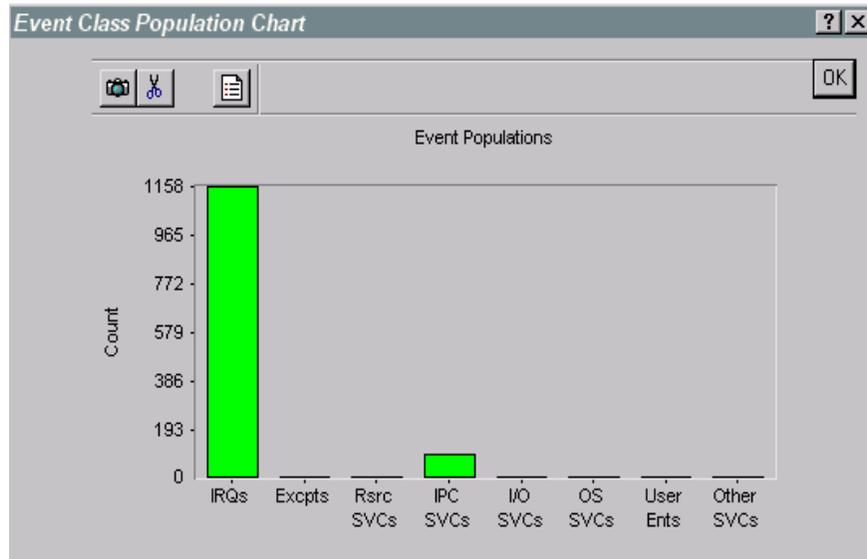
> The charts described in this section work only if `cfx32.ocx` is registered with Windows. The section Running HawkEye describes how to install `cfx32.ocx`.

Basic Population Chart

The **Basic Population Chart** option opens the **Event Class Population Chart** dialog (shown in Figure 3-8) for the current log file.

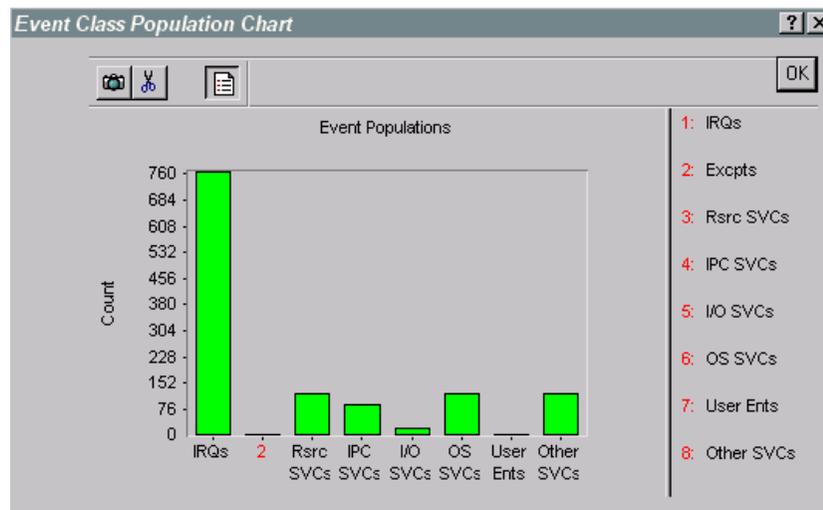**Figure 3-8. Basic Population Chart**

This dialog breaks the events into eight separate classes:

- interrupts
- exceptions
- resource system calls
- inter-process communication calls
- I/O system calls
- OS internal system calls
- user events
- other system calls

The dialog box displays the number of events logged for each class. With this dialog box, you can perform any of the following tasks:

- View the charted log file.
- Copy the chart to the clipboard as a bitmap graphic.
- Copy data to the clipboard as text.
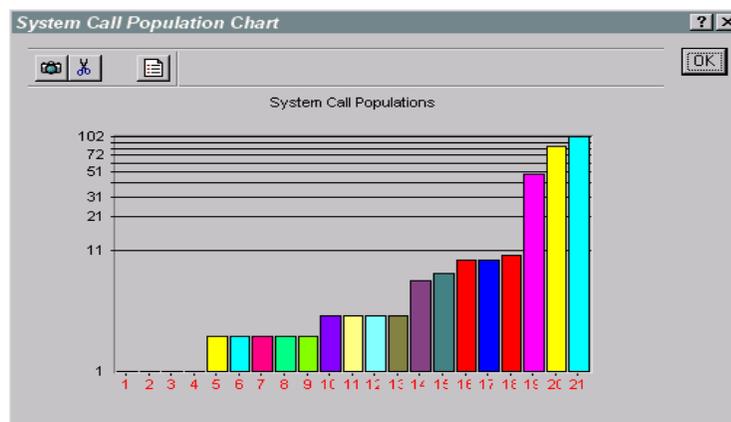- Show or hide the legend (shown in Figure 3-9):

Figure 3-9. Basic Population Chart Legend

System Call Populations

The **System Call Populations Population Chart** (shown in Figure 3-10) charts instances of all system calls in the log file.
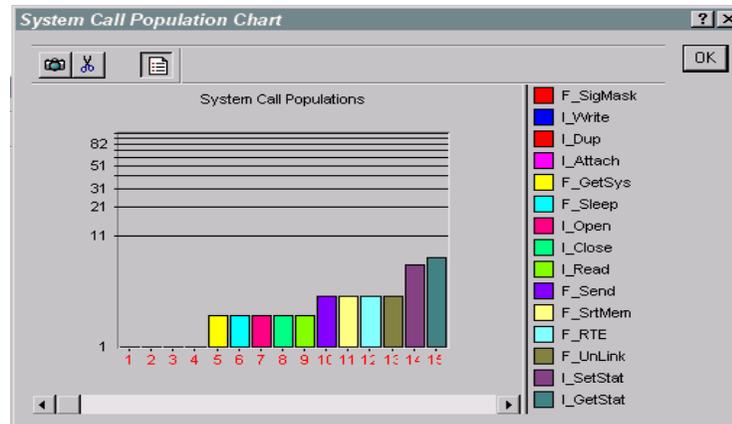
Figure 3-10. System Call Populations

In the **System Call Populations Population Chart** box, you can do any of the following tasks:

- Copy the chart to the clipboard as a bitmap graphic.
- Copy data to the clipboard as text.
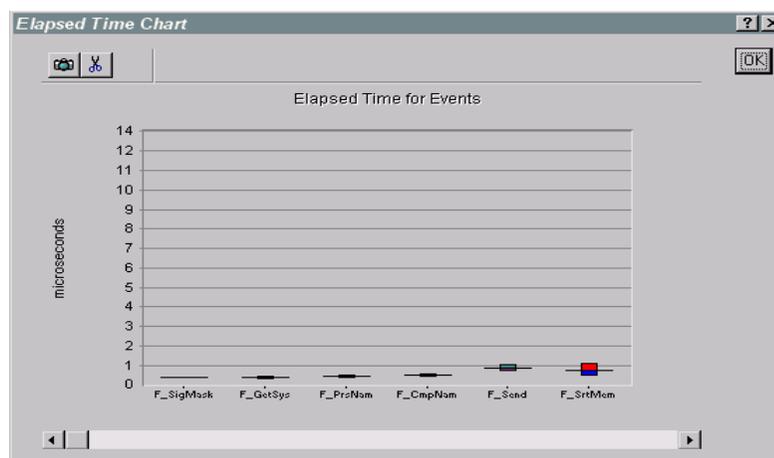- Show or hide the legend (shown in Figure 3-11).

Figure 3-11. System Call Population Chart Legend

Simple Elapsed Times

The **Simple Elapsed Time** option opens the **Elapsed Time Chart** (shown in Figure 3-12). This dialog charts elapsed times for various system events, each system call, and each interrupt entry exit.
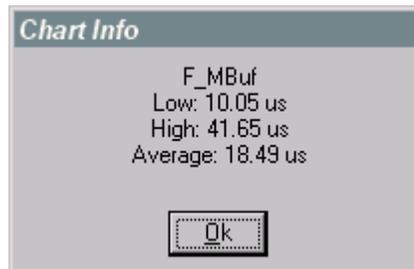
Figure 3-12. Simple Elapsed Time

With the **Elapsed Time Chart** dialog, you can complete the following tasks:

- View the charted log file.
- Copy the chart to the clipboard as a bitmap graphic.
- Copy data to the clipboard as text.

Double-clicking a bar in the chart opens the **Chart Info** dialog box (shown in Figure 3-13).

Figure 3-13. Elapsed Time Chart Info Dialog Box

This dialog box provides a summary of the low, high and average in microseconds of the selected event in the bar graph.

A Porting HawkEye to Custom Hardware

This appendix provides the information needed to port HawkEye to custom-designed hardware. The following section is included:

* High-Resolution Clock Subroutine Module

High-Resolution Clock Subroutine Module

The Hawkeye `slm` module requires access to a high-resolution counter/timer. Hawkeye operates most sucessfully when two events do not contain the same time stamp. In addition, the counter/timer operates most sucessfully with more bits of precision, rather than less; this decreases the likelihood of a complete wrap of the counter between events. In most cases, a one megahertz (or better) clock with 16, 24, or 32-bits fits these requirements.

In some processor architectures, `slm` includes a suitable timer; therefore, no high-resolution clock module is needed. However, in other architectures no timer is included. In such architectures, `slm` links to a module called `hcsub`. The `hcsub` module provides a standard interface to the board-level high-resolution timer.

High-Resolution Timer

The required high-resolution timer is a free-running counter that contains a fixed width. As long as it contains an event inside a roll-over period, Hawkeye will correct the roll-over of the counter. Therefore, it is important that the counter be as wide as possible; this will limit the rate at which the counter rolls over. (Counter widths of 16, 24, and 32 bits are supported.)

In addition, the counter must run at a fixed frequency; frequency cannot change during a capture session. To acheive the best resolution, make the frequency as high as possible.

hcsub Module Overview

The `hcsub` module is a system state subroutine module. You can use global or static variables with `hcsub` functions (described below). You can also use `const` globals if `hcsub` is compiled with `-bepg` to allow code segment `const` variables.

Functions

The `hcsub` module contains three functions (C prototypes included):

```
error_code sub_hc_init(int init_param);
error_code sub_hc_get_ticks(u_int32 *ticks);
error_code sub_hc_get_resolution(u_int32 *clock_res,
u_int32 *timer_bits);
```

The prototypes for these functions are located in `hcpriv.h`.

Table 3-1. hcsub Functions

Function	Description
`sub_hc_init`	Initialize the counter. The initialization function is called before counting begins. The `init_param` prepares the hardware for use. If 1 is passed in, the timer will start. If 2 is passed in, the timer will stop.
`sub_hc_get_ticks`	Return the current count of the timer. This function is calledonce each time Hawkeye logs an event.
`sub_hc_get_resolution`	Return the number of bits in the counter and the number ofcounts per second. This function is called once at the beginning of each Hawkeye capture.