



Using HawkEye

Version 2.1

www.radisys.com

World Headquarters
5445 NE Dawson Creek Drive • Hillsboro, OR
97124 USA
Phone: 503-615-1100 • Fax: 503-615-1121
Toll-Free: 800-950-0044

International Headquarters
Gebouw Flevopoort • Televisieweg 1A
NL-1322 AC • Almere, The Netherlands
Phone: 31 36 5365595 • Fax: 31 36 5365620

RadiSys Microwave Communications Software Division, Inc.
1500 N.W. 118th Street
Des Moines, Iowa 50325
515-223-8000

Revision D
November 2001

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 Microware Communications Software Division, Inc.

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.

November 2001
Copyright ©2001 by RadiSys Corporation.
All rights reserved.

EPC, INtime, iRMX, MultiPro, RadiSys, The Inside Advantage, and ValuPro are registered trademarks of RadiSys Corporation. ASM, Brahma, DAI, DAQ, MultiPro, SAIB, Spirit, and ValuePro are trademarks of RadiSys Corporation.

DAVID, MAUI, OS-9, and OS-9000, are registered trademarks of RadiSys Microware Communications Software Division, Inc. FasTrak, Hawk, SoftStax, and UpLink are trademarks of RadiSys Microware Communications Software Division, Inc.

† All other trademarks, registered trademarks, service marks, and trade names are the property of their respective owners.

Table of Contents

Chapter 1: Getting Started **5**

- 6 Introduction
- 6 How HawkEye Works
- 8 Running HawkEye
- 8 Assumptions
- 8 Requirements
- 8 Host Requirements
- 8 Target Requirements
- 9 Setting Up the Target Machine
- 11 Setting Up the Host Machine
- 12 Verifying the ActiveX Component

Chapter 2: The HawkEye Interface **13**

- 14 The Menu Bar
- 15 Keyboard Mnemonics
- 17 The Toolbar
- 19 Log File Window
- 20 Capture Sessions
- 22 Analyzing the Log File
- 23 Aggregate Lines
- 24 Expand and Collapse
- 24 Events Icons
- 28 Analyzing Details
- 29 Comparing Events
- 29 Displaying Events in Different Formats
- 30 Icons Only
- 31 Simple Labels

32	Full Labels
34	Magnified View

Chapter 3: Working with the HawkEye Interface 35

36	User Event Logger
36	User Event Logging
36	Setting up the Host Machine for Event Logger
37	Setting up the Target Machine for Event Logger
38	User Log Entries
43	Customizing the View
43	Tick Compression
44	Annotations
44	Preferences
45	Connection
46	Target
48	Trigger and Filter Criteria
48	Trigger Criteria
50	Manual Triggers
51	Filter Criteria
53	Log File Charts
54	Basic Population Chart
56	System Call Populations
57	Simple Elapsed Times

Appendix A: Porting HawkEye to Custom Hardware 59

60	High-Resolution Clock Subroutine Module
60	High-Resolution Timer
61	hcsb Module Overview
61	Functions

Chapter 1: Getting Started

This chapter provides an overview of HawkEye and how it works, as well as steps to begin running HawkEye. It includes the following sections:

- **Introduction**
- **Running HawkEye**



MICROWARE SOFTWARE

Introduction

The HawkEye software application is a Graphic User Interface (GUI)-based execution and visualization tool for the OS-9 operating system.

HawkEye is a useful tool for capturing and analyzing logs of events during the execution of a program on an OS-9 target machine. Some possible events may include context switches, interrupts, and system calls, such as process forks, signals, and exits.

In addition, HawkEye 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.

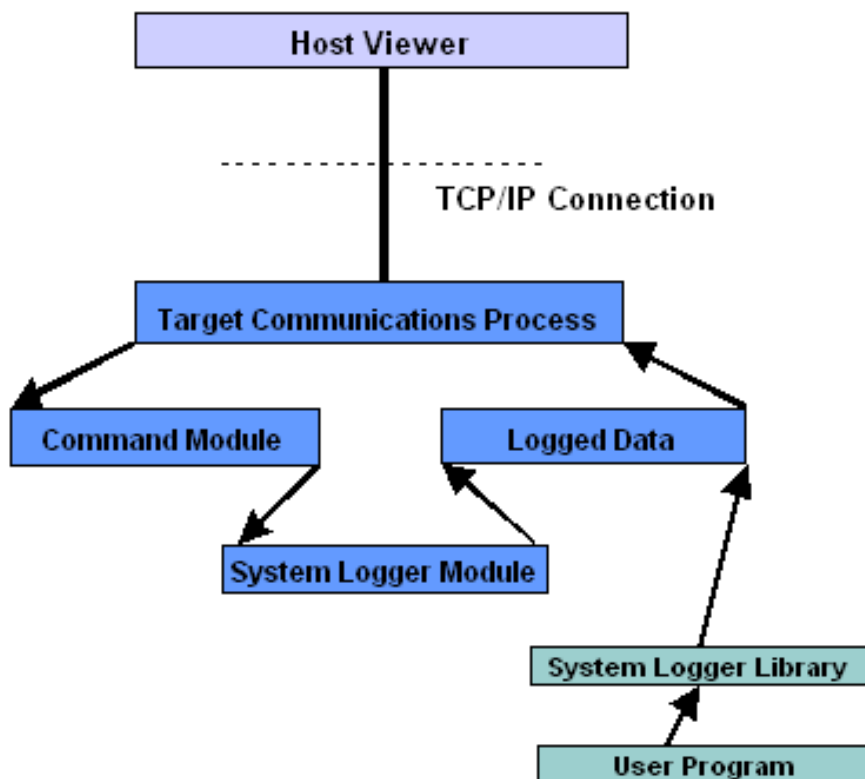
How HawkEye Works

HawkEye is a software application run on your host machine that communicates a specific set of information to the target machine running the OS-9 operating system. This communication between the host and target machine exists via a TCP/IP connection.

The target machine, as depicted in **Figure 1-1**, has the following subsystems installed:

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

Figure 1-1 HawkEye Software Architecture



Each of the above subsystems works in coordination with HawkEye to provide the resulting logged data.

Running HawkEye

This section provides an example session that helps you with running HawkEye.

Assumptions

The sections below assume the following facts:

- 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

The following sections list the host and target requirements for running HawkEye.

Host Requirements

In order to run HawkEye, your host machine must have the following items available:

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

Target Requirements

In order to run HawkEye, your target machine must meet the following requirements:

- There must be a TCP/IP connection configured with OS-9.
- The debugger daemons, `spfindpd` and `spfindpdc`, must be loaded into memory (with `spfindpd` running).

Setting Up the Target Machine

The following sections provide steps on setting up and running HawkEye.



For More Information

Refer to the **Getting Started with Hawk** and **Using Hawk** manuals for more information on setting up Hawk for development.

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

-
- Step 1. Open Hawk and select **Target -> Load** from the Hawk menu.
 - Step 2. Load all of the necessary HawkEye modules onto the OS-9 target machine. The required modules and their pathnames are listed below.

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



Note

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 or custom hardware, proceed to step three for information on loading an additional module.
- If you are using other standard architecture, proceed to step four to start HawkEye.

Step 3. If you are using ARM or custom hardware, you will need to load the high-resolution clock subroutine module, `hcsb`, into memory before HawkEye can run successfully. The process for loading this module varies among these pieces of hardware:

ARM

The `hcsb` module for ARM can be found in the following location:

```
<MWOS>/OS9000/ARMV4/PORTS/<port directory>/  
CMDS/BOOTOBJS/hcsb
```

Custom Hardware



For More Information

If you are porting HawkEye to custom hardware, refer to **[Appendix A:Porting HawkEye to Custom Hardware](#)** for information on loading the `hcsb` module.

When you are finished loading the `hcsb` module, proceed to step four to start HawkEye.

Step 4. 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** to open the **Preferences** dialog box. From here, enter the TCP/IP address of the target in the **Host Connection** field.
 - Step 3. Click on the **Target** tab and select the processor type of your OS-9 target. 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 on **OK** to save the selected settings and dismiss the dialog box.



Note

The target and host machines are configured for HawkEye. You can now click the **Start Capture** button to record two seconds of data.

Verifying the ActiveX Component

As soon as you have started HawkEye, you should verify that the CharFX ActiveX component, `cfx32.ocx`, 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, go to the **Tools** menu in the HawkEye interface and select any one of the three options listed. If none of the options is accessible, `cfx32.ocx` has not been registered with Windows.

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

Step 1. Put `cfx32.ocx` in the `windows/system` folder (Windows 95)

-OR-

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

Step 2. Run `regsvr32 cfx32.ocx` to register it.



Note

`regsvr32` should be included with your Windows 95 or Windows NT system.

Chapter 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**



MICROWARE SOFTWARE

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 More Information

For more information on the **Preferences** option, see **Preferences** in the **Chapter 3: Working with the HawkEye Interface**.

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 More Information

For more information on the trigger and filter criteria, see **Trigger and Filter Criteria** in **Chapter 3: Working with the HawkEye Interface**.

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 Bar 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	
Select All	<Ctrl> + <A>

Table 2-1 HawkEye Keyboard Mnemonics (continued)

Function	Mnemonic
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



- Click the **New** button to open a new log file document.
- Click the **Open** button to open an existing log file document.
- Click the **Save** button to save the current log file document.
- Click the **Cut** button to cut the current selection from the pasted timeline.
- Click the **Copy** button to copy the current selection.
- Click the **Paste** button to paste the clipboard item to the cursor placement area.
- Click the **Annotate** button to open the **Annotations** dialog box and view and create annotations for an event.
- Click the **Start Capture** button to start a log capture.
- Click the **Trigger** button to open the **Trigger Criteria** dialog box and set trigger criteria.
- Click the **Filter** button to open the **Filter Criteria** dialog box and set filter criteria.
- Click the **Zoom in** button to zoom in on the current log file.
- Click the **Zoom out** button to zoom out of the current log file.
- Click the **Expand** button to expand all of events from the aggregate line that belong to a particular process ID.
- Click the **Collapse** button to minimize any expanded events.
- Click the **interactions** button to see the interactions among processes in any given log file.

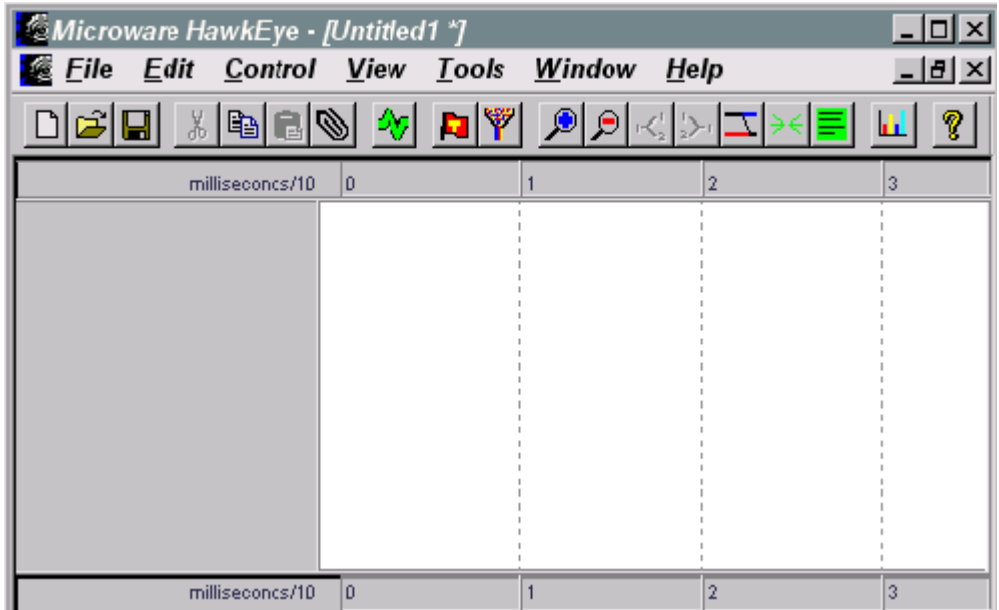
- Click the **Collapse time** button to condense the information on the screen.
- Click the **Snapshot target** to put names on processes. This is typically done after the first capture.
- Click the **Event Count Chart** button to open the **Event Class Populations Chart** dialog box.
- Click the **Help** button to open the on-line help facility.

Using drag and drop functionality, you can move the toolbar from the default position to any area below the title bar. The toolbar can be “docked” vertically along the side of the application window, horizontally along the bottom of the application window, or can remain free-floating over and outside of the application window.

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

The ability to capture data is a major feature of HawkEye. To perform a capture session is to capture the occurrences between the OS-9 system and the software running on the system. Upon capturing a session, the results are visible in the Log File Window.

Below are steps to 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.



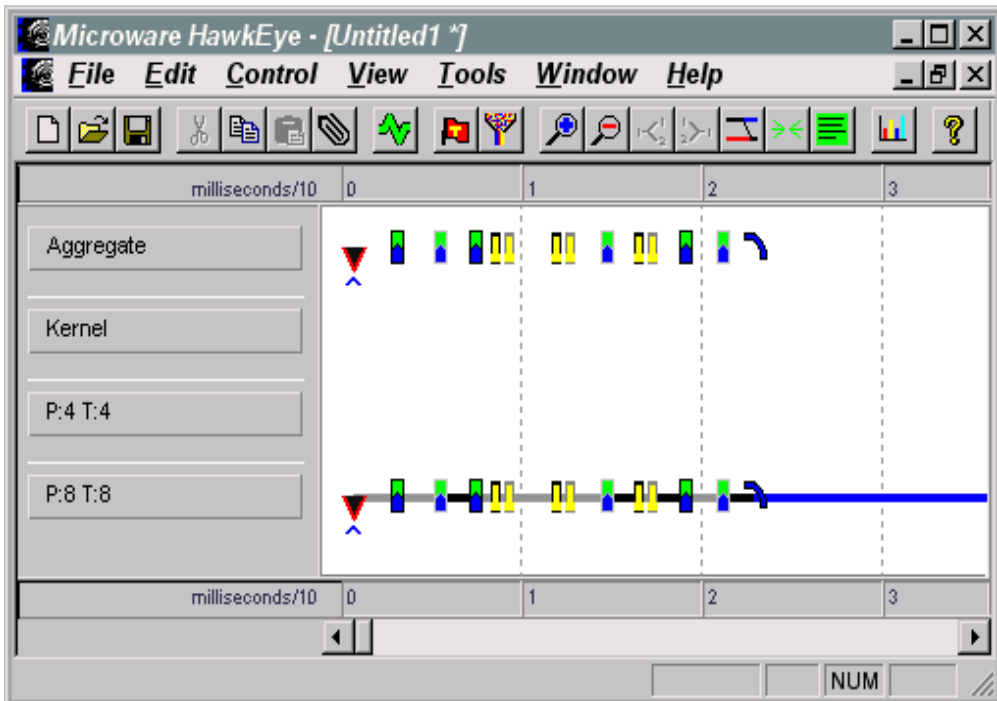
For More Information

For more information on trigger and filter criteria, see **Trigger and Filter Criteria** in **Chapter 3: Working with the HawkEye Interface**.

- 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.
-

Right after a capture, **HawkEye** looks similar to **Figure 2-3**, shown below.

Figure 2-3 Viewed Capture Session



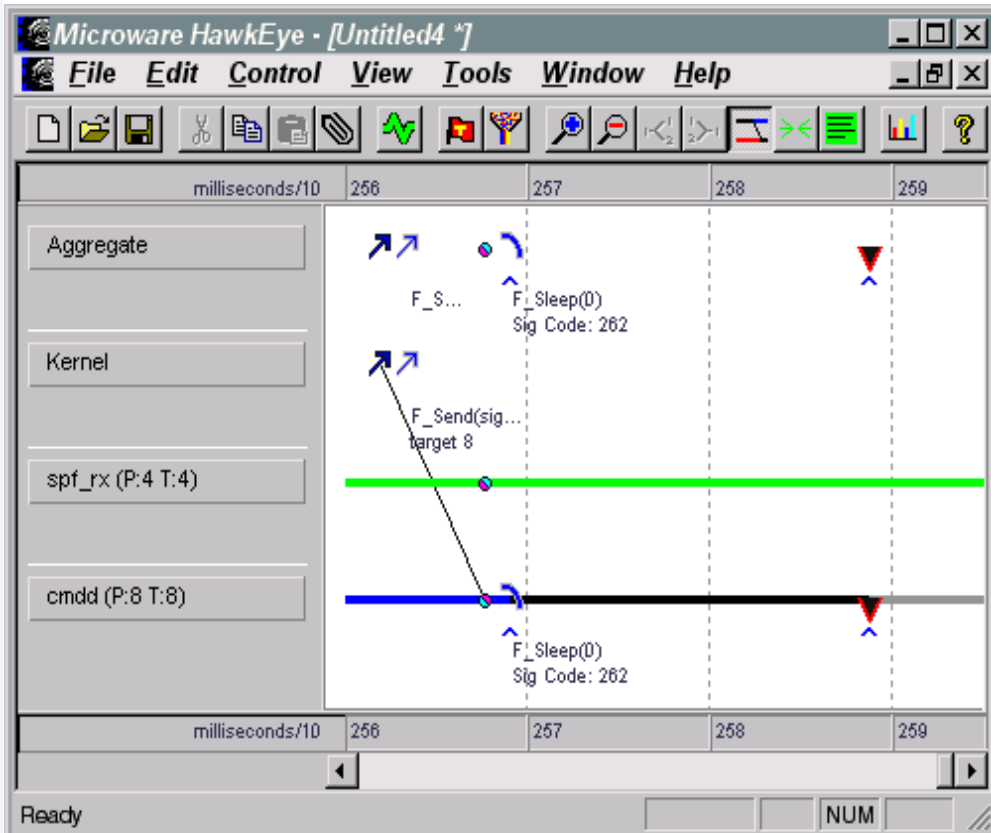
Note

Because events on the OS-9 system may not occur in the same order, each captured session may appear different from the previous 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
Event Signal Return		Event ID, Return Code, Time Stamp, State, PID/TID, PC, SP, Annotation

Table 2-3 Event Icons (continued)












Event	Icon	Possible Details for this Event
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

Table 2-3 Event Icons (continued)

















Event	Icon	Possible Details for this Event
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
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

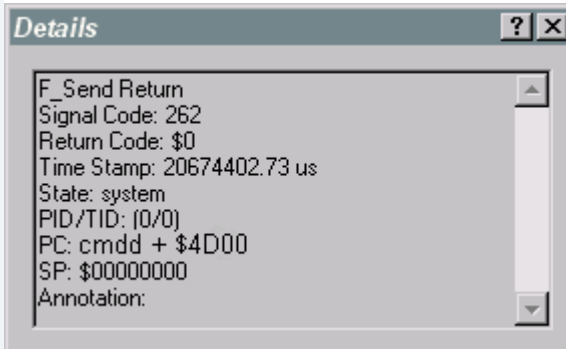
Table 2-3 Event Icons (continued)

Event	Icon	Possible Details for this Event
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 box (shown in **Figure 2-5**) for that event.

Figure 2-5 Details Dialog Box



The **Details** dialog box 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 box 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 box:

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

- **Ticks Remaining** is a returned value resulting from varied system call returns.
- **Return Code** is the value returned by OS-9 system calls.

The following hints may be helpful 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.

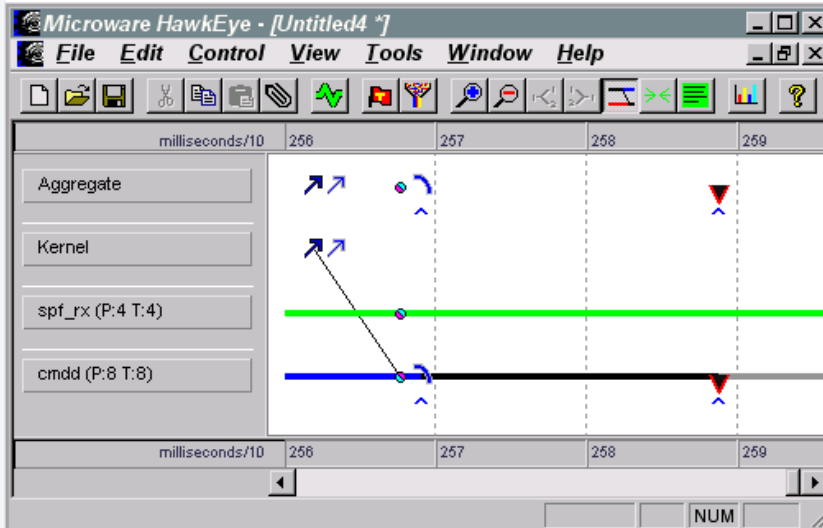
Displaying Events in Different Formats

An additional feature of HawkEye provides is the ability to view the log file in different ways. Below are the 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.

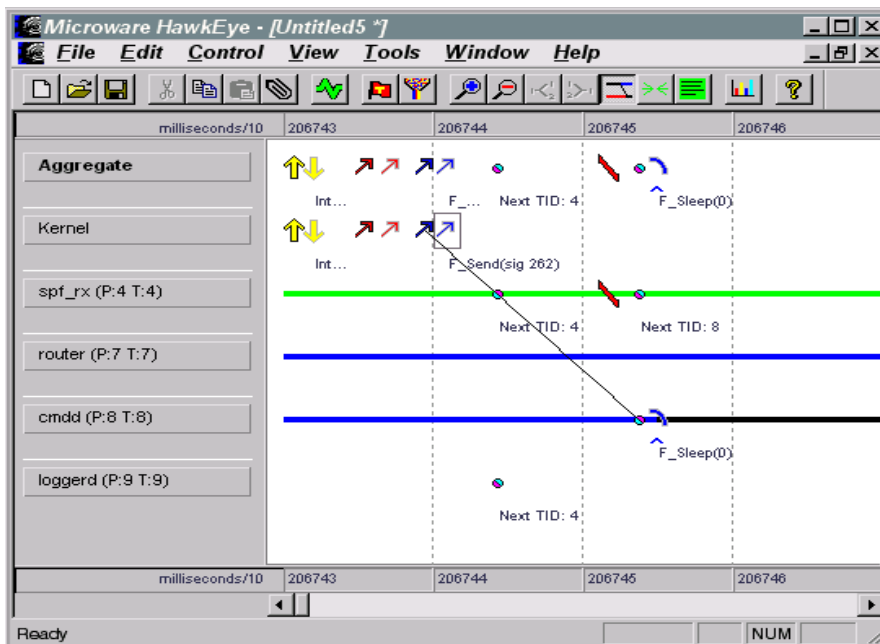
Figure 2-6 Icons Only Label Display



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



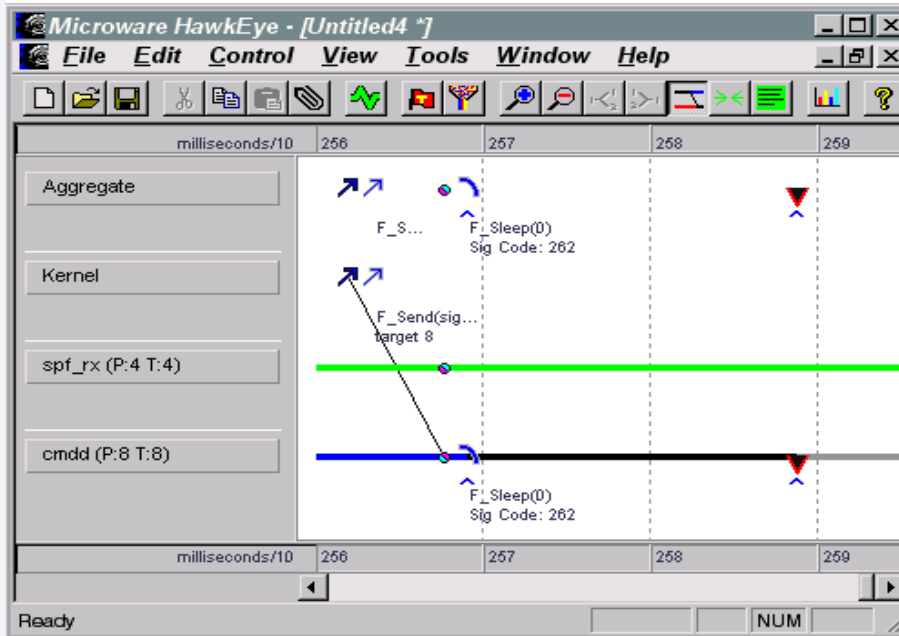
The **Simple Labels** selection displays only one line of descriptive text data below each event in the log file window.

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



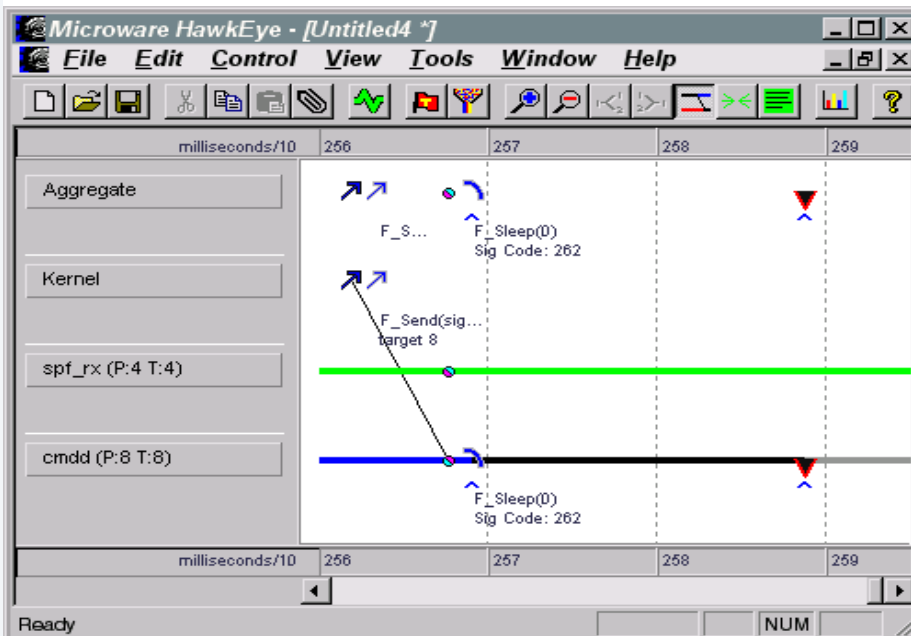
The **Full Labels** selection displays two lines of descriptive data below each event in the log file window.

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

With the **Automatic** label view (shown in [Figure 2-9](#)), the events in a log file automatically display depending on the space available in the log file window.

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 when 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, it automatically magnifies the x-axis of the log file to **2x**. Thereafter, the **2x**, **4x**, **8x** and **16x View** menu options are available for use. The magnification area is movable in order to magnify different portions of the log file window.

To turn off the **Magnifying Glass**, reselect **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%).

Chapter 3: Working with 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**



MICROWARE SOFTWARE

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/slmllib.h



Note

User Event Logging is not available on 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 (`slmllib.l`).

Setting up the Host Machine for Event Logger

The following steps illustrate how to set up your host machine for using the event logger:

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

Setting up the Target Machine for Event Logger

The following steps illustrate how to set up your target machine for using the event logger:

-
- Step 1. Write a C program using the User Event Logger library (`slmlib.l`) to log when user events have occurred. These events are specific to the program written.
- Step 2. When the C program from step 1 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.
-



Note

A sample program called `slmtest` has been provided that demonstrates the use of user triggers and the `slmlib` logger library.



Note

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 have several purposes. One purpose of user log entries is to annotate the log with events that are normally invisible to HawkEye. In addition, user log entries can be used as triggers.

The following functions, found in the `slmlib.1` library, are included as user log entries:

- `hawk_control_log(event, note)` 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 machine and displayed on the host machine. 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 of the event code.
- `hawk_control_log2(event, note, ptr1, ptr2, pointer1, pointer2)` logs a mixed text and numeric user entry. This works much like a log entry from `hawk_control_log()`, but it accepts four numbers. These can be any combination of pointers and numbers, however, HawkEye attempts to decode `ptr1` and `ptr2` as pointers, and displays `number3` and `number4` as numbers.

The above functions are explained in detail in the following section.

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.



Note

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()` will return `EOS_UNKSVC` if `slm` is not installed.

`hawk_control_log()` will return a memory access or protection error if the pointer to `note` is not good.

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_log()`.



Note

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.

<code>note</code>	an ASCII string up to 31 characters long. This is recorded in the event log and passed without modification to the host.
<code>ptr1</code> and <code>ptr2</code>	passed to the HawkEye host code without modification The host attempts to decode these as pointers.
<code>n1</code> and <code>n2</code>	passed to the HawkEye host code without modification The host displays these as numbers.

Errors

`hawk_control_log2()` will return `EOS_UNKSVC` if `slm` is not installed.

`hawk_control_log2()` will return a memory access or protection error if the pointer to `note` is not good.

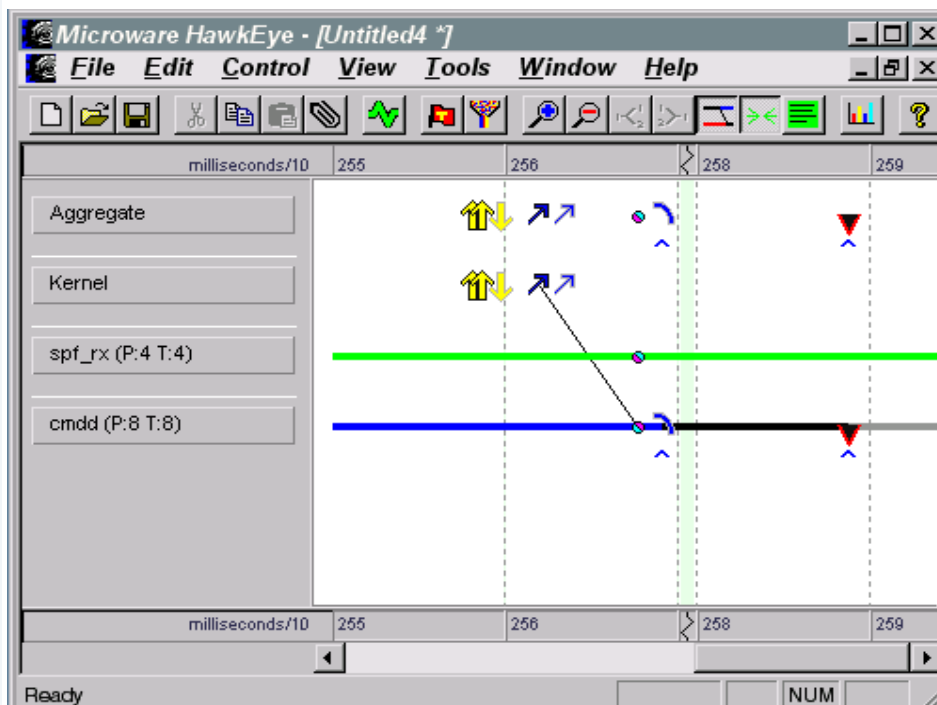
Customizing the View

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

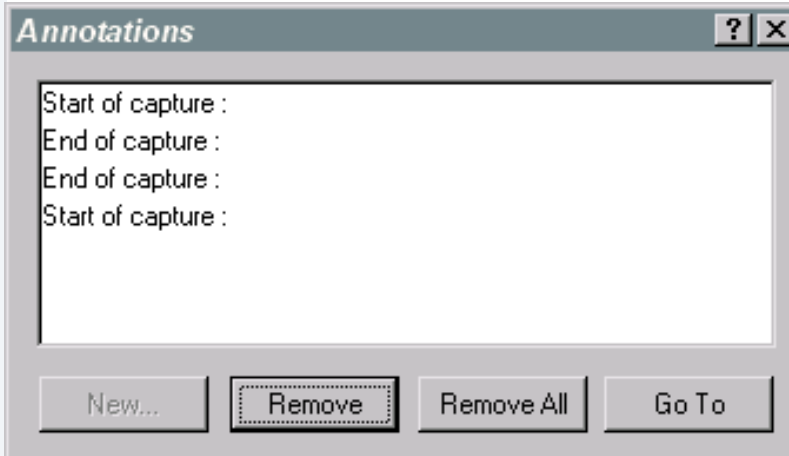


Tick compression eliminates empty space within the log file; this enables you to view more useful event data without having to scroll through lengths of the log file window. In addition, you can click the **Collapse time** toolbar button to filter out empty ticks.

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

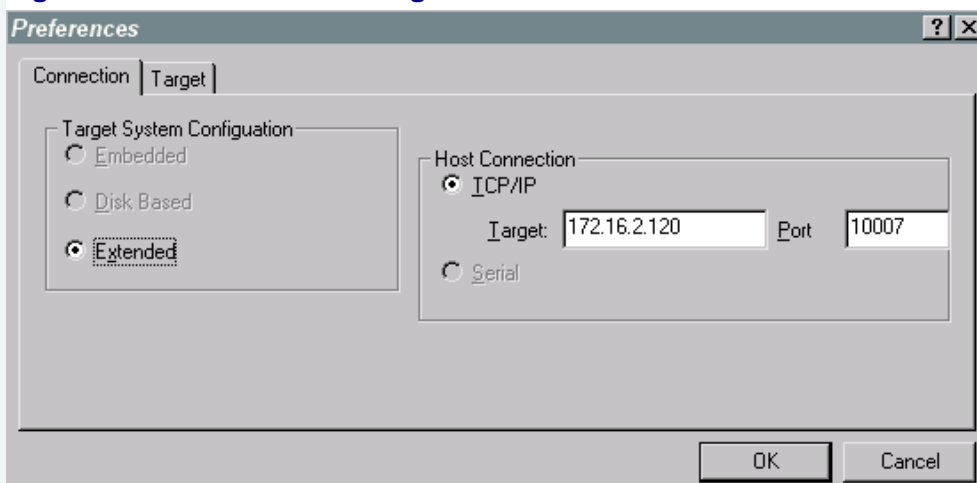
The **Preferences** dialog opens with the **Connection** tab displayed (as shown in [Figure 3-3](#)). This tab is the location of the target and host connections. There are three options for configuring the target system, including the embedded, disk-based, and extended options.



Note

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.



Note

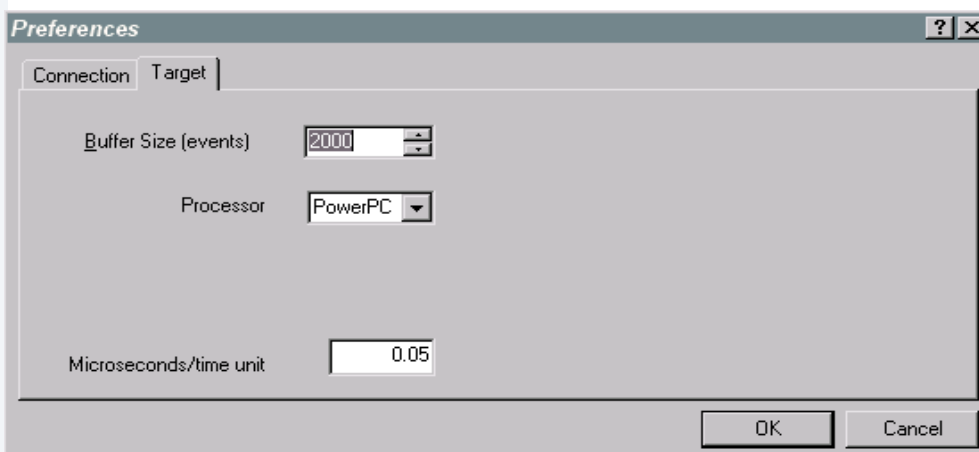
For this release HawkEye only supports the TCP/IP connection option.

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 **Trigger** from the **Control** menu. This brings up the **Trigger** Criteria dialog (shown in [Figure 3-5](#)).

Figure 3-5 Trigger Criteria Dialog Box

Trigger Criteria [?] [X]

Current Trigger Criteria

Remove All

Remove

Add

AFTER 5

Compose Trigger

☒ After time 5 Ticks

☐ User Event 0 After 1 Calls

☐ On Interrupt Exception 0 After 1 Calls

☐ On Interrupt Return Exception 0 After 1 Calls

☐ On System Call F_AcqLk After 1 Calls

☐ On System Call Return F_AcqLk After 1 Calls

Upload When

☒ Full Buffer Load ☐ Half Buffer Load ☐ Immediately

OK Cancel

The **Trigger Criteria** dialog is the specification site for the actions that trigger the capture session. When one of the trigger criteria is met, logging begins. Trigger criteria is relevant at any of the following times:

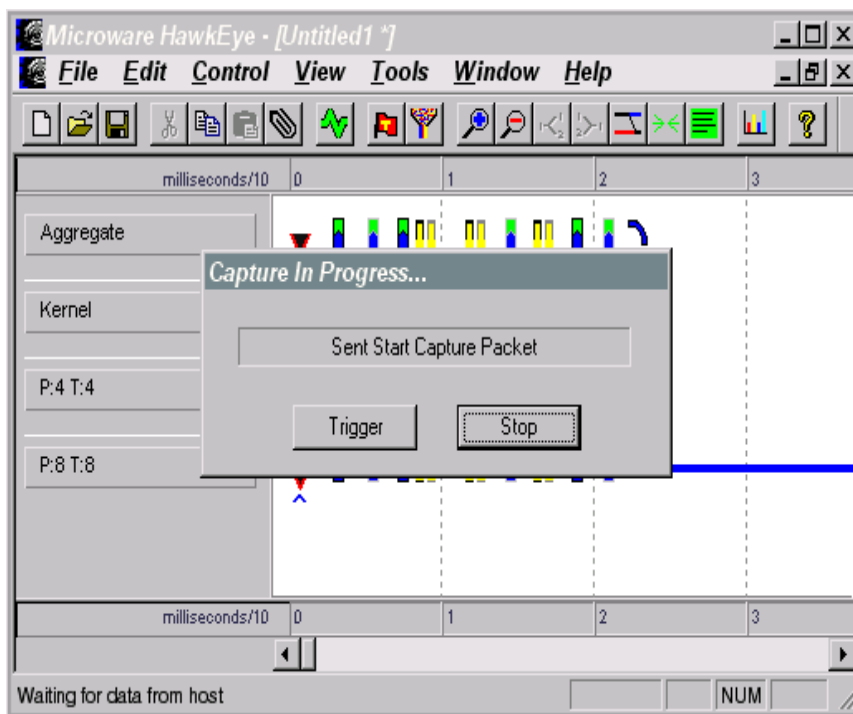
- following a designated number of target system ticks
- following a designated number of user events
- on interrupts
- on interrupt returns
- on system calls
- on system call returns

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

- Click **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.
- Click **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.
- Click **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

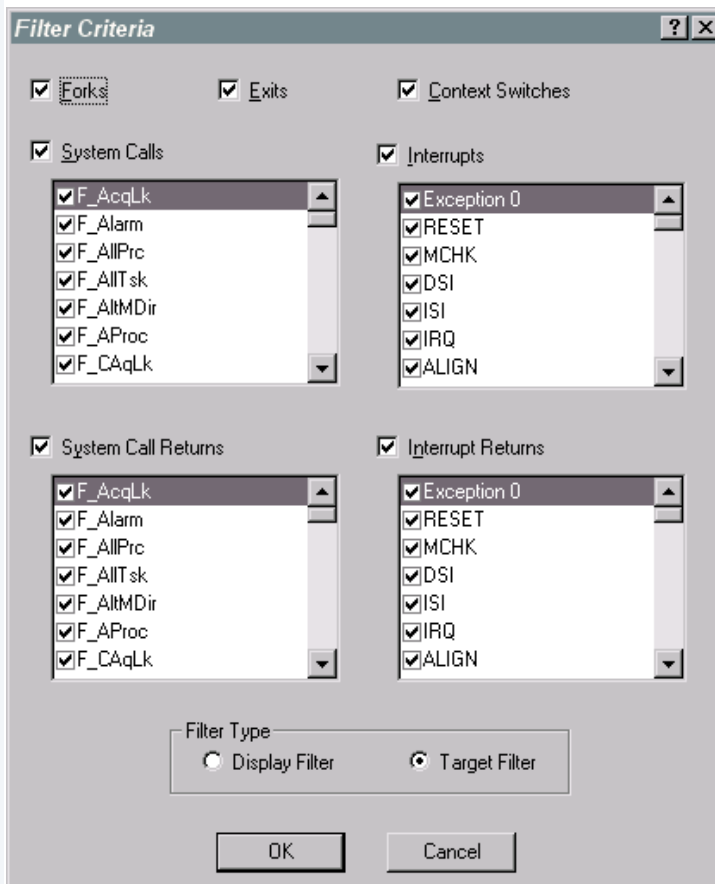
You can perform a manual trigger during a capture. The **Trigger** button allows you to send a manual trigger to the target system. This has the same effect as if one of the trigger conditions was met in the Trigger window.



Filter Criteria

To begin specifying filter criteria during event captures, select **Filter** from the **Control** menu. The **Filter Criteria** dialog is displayed (shown in [Figure 3-6](#)):

Figure 3-6 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. Your options are explained below:

- 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.



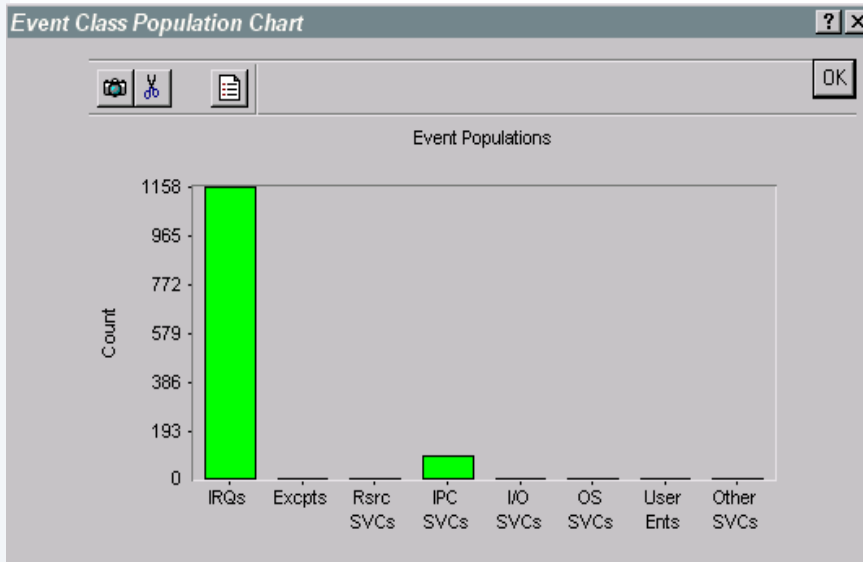
Note

The charts described in this section work only if `cfx32.ocx` is registered with Windows. The section **Running HawkEye** in **Chapter 1: Getting Started** describes how to install `cfx32.ocx`.

Basic Population Chart

The **Basic Population Chart** option opens the **Event Class Population Chart** dialog (shown in [Figure 3-7](#)) for the current log file.

Figure 3-7 Basic Population Chart



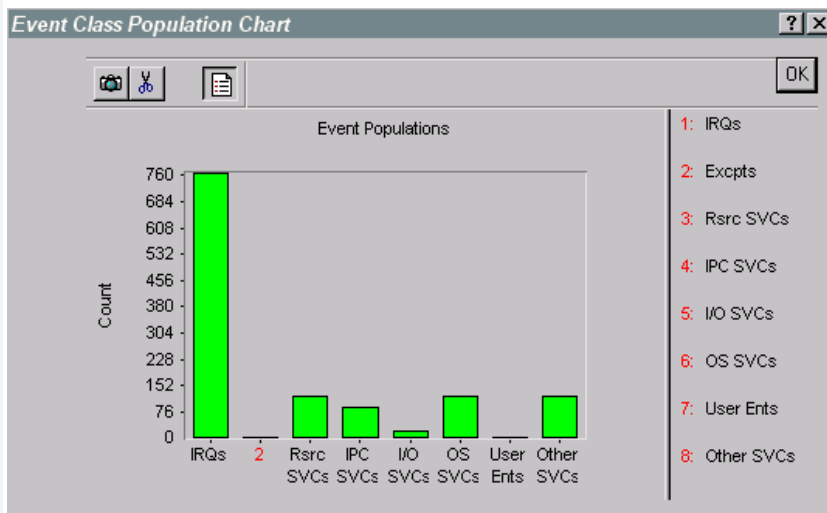
The **Basic Population Chart** dialog box breaks the events into eight separate classes:

- 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 3-8**):

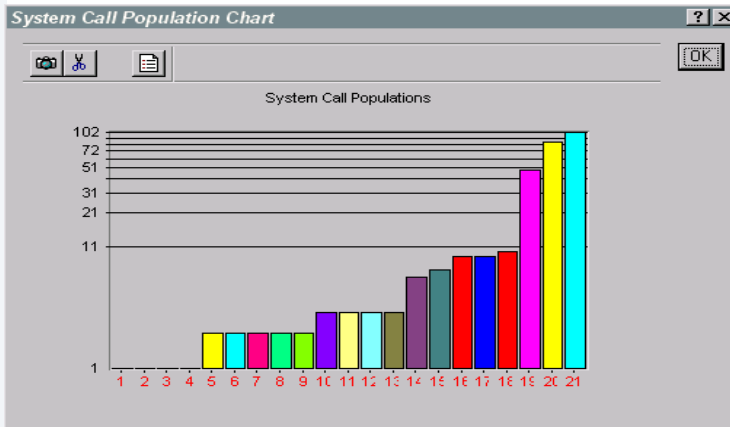
Figure 3-8 Basic Population Chart Legend



System Call Populations

The **System Call Populations Population Chart** (shown in **Figure 3-9**) charts instances of all system calls in the log file.

Figure 3-9 System Call Populations

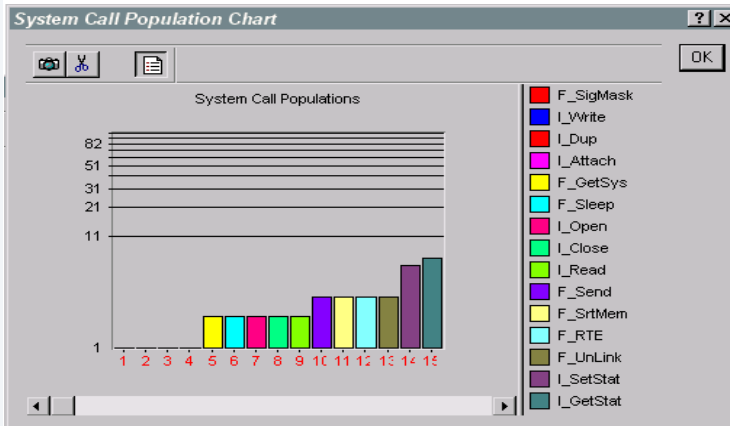


In this box, you can do any of the following tasks:

- 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 3-10](#)).

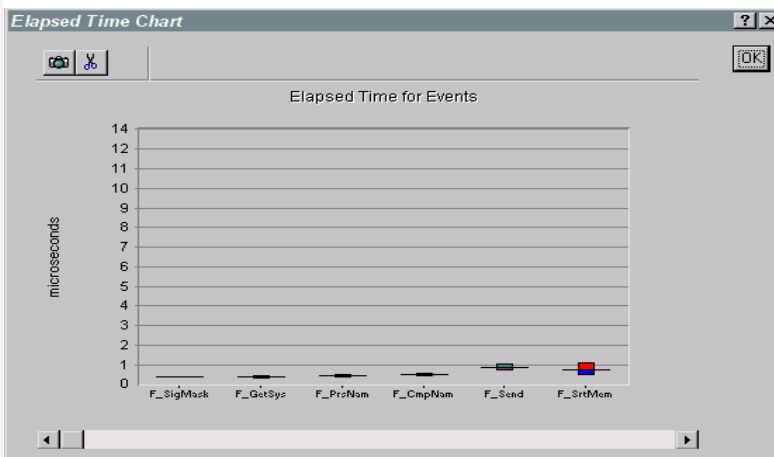
Figure 3-10 System Call Population Chart Legend



Simple Elapsed Times

The **Simple Elapsed Time** option opens the **Elapsed Time Chart** (shown in [Figure 3-11](#)). This dialog charts elapsed times for various system events, each system call, and each interrupt entry exit.

Figure 3-11 Simple Elapsed Time

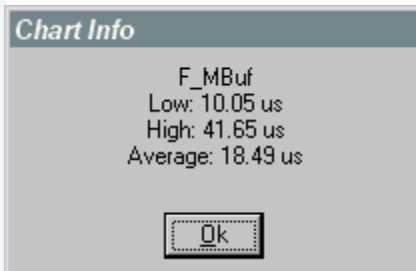


With the **Elapsed Time Chart** dialog, you can complete the following tasks:

- 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 3-12**).

Figure 3-12 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.

Appendix 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**



MICROWARE SOFTWARE

High-Resolution Clock Subroutine Module

The Hawkeye `slm` module requires access to a high-resolution counter/timer. Hawkeye operates most successfully when two events do not contain the same time stamp. In addition, the counter/timer operates most successfully 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 `hcsb`. The `hcsb` 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 achieve the best resolution, make the frequency as high as possible.

hcsb Module Overview

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

Functions

The `hcsb` module contains three functions. These functions are described in [Table 3-1](#) and listed below. (C prototypes are 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 hcsb Functions

Function	Description
<code>sub_hc_init</code>	<p>Initialize the counter.</p> <p>The initialization function is called before counting begins.</p>
<code>sub_hc_get_ticks</code>	<p>Return the current count of the timer.</p> <p>This function is called once each time Hawkeye logs an event.</p>
<code>sub_hc_get_resolution</code>	<p>Return the number of bits in the counter and the number of counts per second.</p> <p>This function is called once at the beginning of each Hawkeye capture.</p>

