

*RedHawk™ Linux®
Frequency-Based Scheduler User's Guide*



0898005-240
March 2006

Copyright 2006 by Concurrent Computer Corporation. All rights reserved. This publication or any part thereof is intended for use with Concurrent products by Concurrent personnel, customers, and end-users. It may not be reproduced in any form without the written permission of the publisher.

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

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

Linux is a registered trademark of Linus Torvalds.

Red Hat is a registered trademark of Red Hat, Inc.

RedHawk and iHawk are trademarks of Concurrent Computer Corporation.

NightSim and NightView are trademarks of Concurrent Computer Corporation.

POSIX is a registered trademark of the Institute of Electrical and Electronics Engineers, Inc.

Other products mentioned in this document are trademarks, registered trademarks, or trade names of the manufacturers or marketers of the product with which the marks or names are associated.

Printed in U. S. A.

Revision History:	Level:	Effective With:
August 2002	000	RedHawk Linux 1.1 & 1.2
September 2003	110	RedHawk Linux 1.4
December 2003	200	RedHawk Linux 2.0
March 2004	210	RedHawk Linux 2.1
July 2004	220	RedHawk Linux 2.2
May 2005	230	RedHawk Linux 2.3
March 2006	240	RedHawk Linux 4.1

Scope of Manual

This manual describes the Frequency-Based Scheduler (FBS) and the Performance Monitor (PM) operating on the RedHawk Linux operating system. These utilities are used for scheduling processes at specified frequencies for the purpose of monitoring performance.

Instructions for using the associated real-time command processor (**rtcp**) and the C and FORTRAN library routines are included in this guide. Not included are instructions for using NightSim™ RT, a graphical user interface for FBS and PM, which is documented in the *NightSim RT User's Guide*.

Structure of Manual

This manual consists of the following sections:

- Chapter 1, *Introduction*, provides an introduction to the Frequency-Based Scheduler and the Performance Monitor as well as software, configuration and access privilege requirements.
- Chapter 2, *Using the Frequency-Based Scheduler*, provides a description of the Frequency-Based Scheduler, its capabilities and user interface summary.
- Chapter 3, *Timing Sources*, explains how to use a real-time clock, an edge-triggered interrupt and a user-supplied device as the timing source for a frequency-based scheduler.
- Chapter 4, *Using the Performance Monitor*, provides a description of the Performance Monitor, its capabilities and user interface summary.
- Chapter 5, *Using rtcp*, explains the procedures for using the real-time command processor, **rtcp**, and provides reference information for each of its commands.
- Chapter 6, *The C Library Interface*, describes the C Library interface to the Frequency-Based Scheduler and the Performance Monitor and provides reference information for each of the routines.
- Chapter 7, *The FORTRAN Library Interface*, describes the FORTRAN Library interface to the Frequency-Based Scheduler and the Performance Monitor and provides reference information for each of the routines.
- Appendix A, *Example rtcp Script*, contains an example **rtcp** script.
- Appendix B, *rtcp Error Messages*, provides explanations of the errors that may be reported by **rtcp**.
- Appendix C, *Example C Interface*, contains an example program that shows how to use the C library interface to the Frequency-Based Scheduler and the Performance Monitor.

The *Glossary* contains definitions of technical terms that are important to understanding the concepts presented in this book.

The *Index* contains an alphabetical reference to key terms and concepts and numbers of pages where they occur in the text.

Syntax Notation

The following notation is used throughout this manual:

<i>italic</i>	Books, reference cards, and items that the user must specify appear in <i>italic</i> type. Special terms may also appear in <i>italic</i> .
list bold	User input appears in list bold type and must be entered exactly as shown. Names of directories, files, commands, options and man page references also appear in list bold type.
list	Operating system and program output such as prompts, messages and listings of files and programs appears in list type.
[]	Brackets enclose command options and arguments that are optional. You do not type the brackets if you choose to specify these options or arguments.
hypertext links	When viewing this document online, clicking on chapter, section, figure, table and page number references will display the corresponding text. Clicking on Internet URLs provided in blue type will launch your web browser and display the web site. Clicking on publication names and numbers in red type will display the corresponding manual PDF, if accessible.

Related Publications

The following Concurrent publications supplied with RedHawk are referenced in this document.

Title	Pub No.
<i>RedHawk Linux Release Notes Version x.x</i>	0898003
<i>RedHawk Linux User's Guide</i>	0898004
<i>Real-Time Clock and Interrupt Module (RCIM) PCI Form Factor User's Guide</i>	0898007
<i>NightSim RT User's Guide</i>	0890480
<i>NightView RT User's Guide</i>	0890395

where *x.x* = release version

Contents

Preface	iii
Chapter 1 Introduction	
Frequency-Based Scheduler Overview	1-1
Performance Monitor Overview	1-2
Software Requirements	1-2
Configuration	1-3
Frequency-Based Scheduler	1-3
Performance Monitor	1-3
Privileges	1-4
Chapter 2 Using the Frequency-Based Scheduler	
What Is the Frequency-Based Scheduler?	2-1
How Is Scheduler Frequency Defined?	2-2
How Are Processes Scheduled?	2-2
Tolerating Frame Overruns	2-5
User Interface	2-6
Debugging Frequency-Based Scheduler Processes	2-6
Chapter 3 Timing Sources	
Overview	3-1
Using a Real-Time Clock	3-1
Understanding the Real-Time Clock Device	3-1
Understanding the User Interface	3-2
General Procedures for Using a Real-Time Clock	3-2
Using an Edge-Triggered Interrupt	3-3
Understanding the Edge-Triggered Interrupt	3-3
Understanding the User Interface	3-4
Chapter 4 Using the Performance Monitor	
What Is the Performance Monitor?	4-1
What Values Are Monitored?	4-1
Monitoring Idle Time	4-3
Monitoring Unscheduled Processes	4-4
User Interface	4-5
Optimizing the Performance of a Simulation	4-5
Chapter 5 Using rtcp	
What Is the Real-Time Command Processor?	5-1
rtcp and the Frequency-Based Scheduler	5-1
rtcp and the Performance Monitor	5-2
Execution Modes	5-2

Using Direct Mode	5-2
Invoking rtcp with a Command Name and Arguments	5-3
Invoking rtcp with Commands Redirected from a Script File	5-3
Invoking a Script that Calls rtcp.	5-3
Using Interactive Mode	5-4
Getting Help	5-5
rtcp Commands	5-7
rtcp Command Summary	5-7
Command Sequence	5-8
Using rtcp Commands	5-10
ats – Attach Timing Source to a Frequency-Based Scheduler	5-11
chs – Change Permissions for a Frequency-Based Scheduler	5-12
cs – Configure a Frequency-Based Scheduler	5-13
dts – Detach Timing Source from a Frequency-Based Scheduler	5-15
rms – Remove a Frequency-Based Scheduler	5-16
svs – Save Scheduler Configuration	5-17
vc – View Minor Cycle/Major Frame Count	5-18
vs – View Scheduler Configuration	5-19
ls – Display All Schedulers on the System.	5-21
rc – Start Real-Time Clock.	5-21
sc – Stop Real-Time Clock.	5-22
stc – Set Real-Time Clock	5-22
gtc – Display Real-Time Clock Settings	5-23
start – Start Scheduling on a Frequency-Based Scheduler.	5-24
resume – Resume Scheduling on a Frequency-Based Scheduler.	5-25
stop – Stop Scheduling on a Frequency-Based Scheduler	5-25
rmp – Remove a Process from a Frequency-Based Scheduler	5-26
rsp – Reschedule a Process	5-28
sp – Schedule a Process on a Frequency-Based Scheduler	5-32
vp – View Processes on a Frequency-Based Scheduler	5-35
cpm – Clear Performance Monitor Values	5-38
pm – Start/Stop Performance Monitoring	5-40
vcm – View/Modify Performance Monitor Timing Mode	5-42
vpm – View Performance Monitor Values	5-43
ex – Exit Real-Time Command Processor	5-47
he – Display Help Information	5-47

Chapter 6 The C Library Interface

Overview	6-1
Compiling and Linking Programs.	6-1
Frequency-Based Scheduler Routines	6-2
Routine Summary	6-2
C Library Call Sequence	6-4
Using Frequency-Based Scheduler Routines	6-5
fbsaccess – Change Permissions for a Frequency-Based Scheduler	6-5
fbsattach – Attach Timing Source to a Frequency-Based Scheduler	6-6
fbsavail – Query if the Frequency-Based Scheduler is Configured	6-7
fbsconfigure – Configure a Frequency-Based Scheduler.	6-8
fbscycle – Return Minor Cycle/Major Frame Count	6-11
fbsdetach – Detach Timing Source from a Frequency-Based Scheduler.	6-12
fbsdir – Return a List of Scheduler Keys	6-13
fbsgetpid – Return Process ID for a Scheduled Process.	6-14

fbsetrtc – Obtain Current Values for a Real-Time Clock	6-15
fbssid – Return the Frequency-Based Scheduler Identifier for a Key	6-16
fbinfo – Return Information for a Frequency-Based Scheduler	6-17
fbintrpt – Start/Stop/Resume Scheduling	6-19
fbremove – Remove a Frequency-Based Scheduler	6-20
fbresume – Resume Scheduling on a Frequency-Based Scheduler	6-21
fbsetrtc – Start/Stop a Real-Time Clock	6-22
fbsschedself – Add a Calling Process to a Frequency-Based Scheduler	6-23
fbsetrtc – Set a Real-Time Clock	6-25
fbstrig – Make a Sleeping Frequency-Based Scheduler Process Runnable	6-26
fbwait – Wait on a Frequency-Based Scheduler	6-27
nametopid, namepid – Return the Process ID for a Specified Process Name	6-28
pgmremove – Remove a Process from a Frequency-Based Scheduler	6-29
pgmtrigger – Trigger a Process on a Frequency-Based Scheduler	6-31
sched_fbsqry – Query Processes on a Frequency-Based Scheduler	6-32
sched_pgm_set_soft_overrun_limit - Set Soft Overrun Limit	6-34
sched_pgm_soft_overrun_query - Query Soft Overrun Processing	6-36
sched_pgmadd – Schedule a Process on a Frequency-Based Scheduler	6-38
sched_pgmqry – Query a Process	6-41
sched_pgmresched – Reschedule a Process	6-44
Performance Monitor Routines	6-47
Routine Summary	6-47
C Library Call Sequence	6-47
Using Performance Monitor Routines	6-48
pmclrpgm – Clear Values for a Process	6-49
pmclrtable – Clear Values for Processor(s)	6-51
pmmonitor – Start/Stop Performance Monitoring on Processor(s)	6-52
pmprogram – Start/Stop Performance Monitoring on a Process	6-53
pmqrycpu – Query Values for Selected Processor(s)	6-55
pmqrylist – Query Values for a List of Processes	6-58
pmqrypgm – Query Values for a Selected Process	6-60
pmqrytimer – Query Performance Monitor Mode	6-63
pmselect – Select Performance Monitor Mode	6-64

Chapter 7 The FORTRAN Library Interface

Overview	7-1
Compiling and Linking Procedures	7-1
Frequency-Based Scheduler Routines	7-2
Routine Summary	7-2
FORTRAN Library Call Sequence	7-4
Using Frequency-Based Scheduler Routines	7-5
fbaccess – Change Permissions for a Frequency-Based Scheduler	7-5
fbattach – Attach Timing Source to a Frequency-Based Scheduler	7-6
fbconfigure – Configure a Frequency-Based Scheduler	7-7
fbcycle – Return Minor Cycle/Major Frame Count	7-9
fbdetach – Detach Timing Source from a Frequency-Based Scheduler	7-10
fbsetrtc – Obtain Current Values for Real-Time Clock	7-11
fbssid – Return the FBS Identifier for a Key	7-12
fbinfo – Return Information for a Frequency-Based Scheduler	7-13
fbintrpt – Start/Stop/Resume Scheduling on a Frequency-Based Scheduler	7-15
fbquery – Query Processes on a Frequency-Based Scheduler	7-16
fbremove – Remove a Frequency-Based Scheduler	7-19
fbresume – Resume Scheduling on a Frequency-Based Scheduler	7-20
fbsetrtc – Start/Stop Real-Time Clock	7-22

fbsschedself – Schedule a Process/Thread on a Frequency-Based Scheduler	7-23
fbsssetrtc – Set Real-Time Clock	7-25
fbswait – Wait on a Frequency-Based Scheduler	7-26
nametopid – Return the Process ID for a Specified Process Name	7-27
pgmquery – Query a Process on a Frequency-Based Scheduler	7-28
pgmremove – Remove a Process from a Frequency-Based Scheduler	7-30
pgmreschedule – Reschedule a Process	7-32
pgmschedule – Schedule a Process on a Frequency-Based Scheduler	7-35
pgmstat – Query State of FBS-Scheduled Process	7-38
pgmtrigger – Trigger Process Waiting on FBS	7-40
rtparm – Return Initiation Parameter	7-41
sched_pgm_set_soft_overrun_limit - Set Soft Overrun Limit	7-42
sched_pgm_soft_overrun_query - Query Soft Overrun Processing	7-43
schedfbsqry – Query Processes on a Frequency-Based Scheduler	7-44
schedpgmadd – Schedule a Process on a Frequency-Based Scheduler	7-47
schedpgmqry – Query a Process on a Frequency-Based Scheduler	7-49
schedpgmresched – Reschedule a Process	7-52
Performance Monitor Routines	7-55
Routine Summary	7-55
FORTRAN Library Call Sequence	7-55
Using Performance Monitor Routines	7-57
pmclrpgm – Clear Values for a Process	7-57
pmclrtable – Clear Values for Processor(s)	7-59
pmmonitor – Start/Stop Performance Monitoring on Processor(s)	7-60
pmprogram – Start/Stop Performance Monitoring on a Process	7-61
pmqrycpu – Query Values for Selected Processor(s)	7-63
pmqrylist – Query Values for a List of Processes	7-65
pmqrypgm – Query Values for a Selected Process	7-67
pmquerytimer – Query Performance Monitor Mode	7-70
pmselect – Select Performance Monitor Mode	7-71
Appendix A Example rtcp Script	A-1
Appendix B rtcp Error Messages	B-1
Appendix C Example C Interface	C-1
Glossary	Glossary-1
Index	Index-1

List of Screens

Screen 5-1. rtcp Command Display	5-5
Screen 5-2. First Screen of rtcp Arguments.	5-6
Screen 5-3. Second Screen of rtcp Arguments	5-7

List of Illustrations

Figure 5-1. rtcp Command Sequence: Frequency-Based Scheduler	5-9
Figure 5-2. rtcp Command Sequence: Performance Monitor	5-10
Figure 6-1. C Library Call Sequence: Frequency-Based Scheduler	6-4
Figure 6-2. C Library Call Sequence: Performance Monitor	6-48
Figure 7-1. FORTRAN Library Call Sequence: Frequency-Based Scheduler	7-4
Figure 7-2. FORTRAN Library Call Sequence: Performance Monitor	7-56

List of Tables

Table 2-1. Process Scheduling Example 1	2-3
Table 2-2. Process Scheduling Example 2	2-4
Table 3-1. rtcp Commands and Routines for using RTCs	3-3
Table 3-2. rtcp Commands and Routines for using ETIs	3-4
Table 5-1. rtcp Commands	5-7
Table 6-1. Frequency-Based Scheduler C Library Routines	6-2
Table 6-2. Frequency-Based Scheduler Permissions	6-5
Table 6-3. Performance Monitor C Library Routines	6-47
Table 7-1. Frequency-Based Scheduler FORTRAN Library Routines	7-2
Table 7-2. Frequency-Based Scheduler Permissions	7-5
Table 7-3. Contents of Array Elements: fbsinfo	7-13
Table 7-4. Contents of Array Elements: fbsquery	7-17
Table 7-5. Contents of Array Elements: fbssetrtc	7-24
Table 7-6. Contents of Array Elements: schedfbsqry	7-45
Table 7-7. Performance Monitor FORTRAN Library Routines	7-55
Table 7-8. Contents of Array Elements: pmqrycpu	7-64
Table 7-9. Contents of Array Elements: pmqrylist	7-66
Table B-1. rtcp Errors	B-1

Introduction

Frequency-Based Scheduler Overview	1-1
Performance Monitor Overview	1-2
Software Requirements	1-2
Configuration	1-3
Frequency-Based Scheduler	1-3
Performance Monitor	1-3
Privileges	1-4

This chapter introduces you to the Frequency-Based Scheduler (FBS) and the Performance Monitor (PM). It also provides software and configuration requirements and discusses the privileges and capabilities needed to use these utilities.

Frequency-Based Scheduler Overview

The Frequency-Based Scheduler (FBS) is a task synchronization mechanism used to initiate processes at specified frequencies. Frequencies can be based on high-resolution clocks such as those provided by the Real-Time Clock and Interrupt Module (RCIM), an external interrupt source, or the completion of a cycle. The processes are then scheduled using a priority-based scheduler. FBS can control the periodic execution of multiple, coordinated processes utilizing major and minor cycles with overrun detection.

When used in conjunction with the Performance Monitor, FBS can be used to determine the best way of allocating processors to various tasks for a particular application.

You can easily configure FBS to meet the needs of specific applications. More than one frequency-based scheduler can be configured, each with different capabilities allowing a variety of scheduling techniques for different purposes. A full description of the Frequency-Based Scheduler is provided in Chapter 2.

Access to the major functions associated with frequency-based scheduling is achieved through the following interfaces:

- **rtcp**, the real-time command processor. **rtcp** is explained in Chapter 5.
- a set of library routines that can be called from application programs written in C. The library routines are explained in Chapter 6.
- a set of library routines that can be called from application programs written in FORTRAN. The library routines are explained in Chapter 7.
- NightSim RT, a Graphical User Interface (GUI) to FBS. NightSim RT is explained in the *NightSim RT User's Guide*.

Performance Monitor Overview

The Performance Monitor (PM) monitors use of the CPU by processes that are scheduled on the Frequency-Based Scheduler. Values obtained assist in determining how to redistribute processes among processors for improved load balancing and processing efficiency. A full description of the Performance Monitor and its capabilities is provided in Chapter 4.

Access to the major functions associated with the Performance Monitor is achieved through the following interfaces:

- **rtcp**, the real-time command processor. **rtcp** is explained in Chapter 5.
- a set of library routines that can be called from application programs written in C. The library routines are explained in Chapter 6.
- a set of library routines that can be called from application programs written in FORTRAN. The library routines are explained in Chapter 7.
- NightSim RT, a Graphical User Interface (GUI) to PM. NightSim RT is explained in the *NightSim RT User's Guide*.

Software Requirements

The Frequency-Based Scheduler (FBS) RPM distributed with the RedHawk Linux operating system provides kernel support for the Frequency-Based Scheduler, the Performance Monitor and **rtcp (1)**. This software must be installed in order to access FBS and PM. Refer to the appropriate version of the *RedHawk Linux Release Notes* for installation procedures.

If you want to use RCIM devices for timing FBS processes, the Real-Time Clock and Interrupt Module (RCIM) must be installed and configured in the system. Refer to the appropriate version of the *RedHawk Linux Release Notes* for installation procedures. Refer to the *Real-Time Clock and Interrupt Module (RCIM) PCI Form Factor User's Guide* for information about configuring the RCIM driver and using the clocks and interrupts.

Configuration

Configuration requirements for the Frequency-Based Scheduler and the Performance Monitor are listed below. For information on how to change kernel parameters, see the “Configuring and Building the Kernel” chapter in the *RedHawk Linux User’s Guide*.

Frequency-Based Scheduler

The following system tunable parameter is accessible through the Frequency-Based Scheduler selection of the Kernel Configuration GUI.

FBSCHED	Configures FBS in the system. By default, FBS is configured as a module in each of the RedHawk Linux pre-built kernels.
---------	-------------------------------------------------------------------------------------------------------------------------

Performance Monitor

In addition to having FBS configured in the kernel, the Performance Monitor requires the following:

FBSCHED_PM	This tunable, accessible through the Frequency-Based Scheduler selection of the Kernel Configuration GUI, is used to configure the Performance Monitor in the system. It is configured by default in each of the RedHawk Linux pre-built kernels.
HRACCT	This parameter, accessible through the General Setup selection of the Kernel Configuration GUI, is required in order to use the Performance Monitor. It enables high resolution process accounting, which provides the means for measuring the execution time of each process for its timing values. This parameter is configured by default in the RedHawk Linux debug and trace kernels.

Privileges

RedHawk Linux supports a Pluggable Authentication Module (PAM) called **pam_capability(8)** that provides a role-based access control scheme. In this scheme, you set up a series of roles in the **capability.conf(5)** file. A role is defined as a set of valid Linux capabilities.

RedHawk systems are configured with the “fbscheduser” role. This role defines the set of capabilities needed to execute the libraries and kernel code that make up the FBS subsystem. The FBS user must be configured to use (at a minimum) the capabilities provided by the “fbscheduser” role by default. This is accomplished by adding a line in **/etc/security/capability.conf** for each FBS user with the “fbscheduser” role defined:

```
user      username      fbscheduser
```

where *username* is the name of the user.

Programs that are scheduled on a frequency-based scheduler might require other capabilities to be able to make the system calls that they utilize. The fbscheduser role may be used as a building block to create a more general purpose role. For example, the following two entries would grant all users in a group named “software” the capabilities to use the FBS subsystem (fbscheduser) as well as the capability to use **mlock(2)** and **mlockall(2)** (cap_ipc_lock) along with other capabilities in some user-defined role (user_defined).

```
role      realtimeuser  fbscheduser cap_ipc_lock user_defined
group     software      realtimeuser
```

During system initialization the RCIM device files, by default, are created writeable by all users (mode=0666). If the administrator wishes to restrict the permissions to only certain users, this must be done in such a way as to allow FBS users permission to write these files. One way to achieve this is to define a group that provides write access to the RCIM device files and assign FBS users to that group. An example of how to do this follows.

To add a group named “realtimer”, do the following:

```
# groupadd -r realtimer
```

Next, give the realtimer group write access to the RCIM device files by editing **/etc/sysconfig/rcim** and setting the following:

```
RCIM_OWN=root
RCIM_GRP=realtimer
RCIM_PERM=0664
```

Finally, add FBS users to the group. The example here adds “joe” to the realtimer group:

```
# adduser -u 50 -g 100 -d /home/joe -G realtimer -s /bin/bash -c "Joe User" joe
```


Using the Frequency-Based Scheduler

What Is the Frequency-Based Scheduler?	2-1
How Is Scheduler Frequency Defined?.....	2-2
How Are Processes Scheduled?	2-2
Tolerating Frame Overruns	2-5
User Interface	2-6
Debugging Frequency-Based Scheduler Processes	2-6

Using the Frequency-Based Scheduler

This chapter contains a description of the Frequency-Based Scheduler and its capabilities. It explains how scheduler frequency is defined, how processes are scheduled and what overruns are and how they are managed. The user interfaces for accessing the scheduler are introduced.

What Is the Frequency-Based Scheduler?

The Frequency-Based Scheduler (FBS) is a task synchronization mechanism that enables you to run processes at specified frequencies. A number of frequency-based schedulers can be configured and used simultaneously. Scheduling frequencies are based on a user-defined number of minor cycles that compose a major frame. Each scheduler can use any of several available timing sources to define scheduling frequency or can schedule at the completion of a cycle. In addition, frame overruns can be monitored.

Specifically, FBS allows you to:

- define frequency in terms of the duration of a minor cycle and the number of minor cycles per major frame
- specify the scheduling parameters with which processes are scheduled
- control all scheduling features from one processor (that is, schedule and query a process on any processor)
- detect frame overruns for all frequency-based scheduled processes
- obtain the status of a single frequency-based scheduled process, all frequency-based scheduled processes on a single processor, or all frequency-based scheduled processes on all processors
- remove one or all frequency-based scheduled processes from a scheduler
- reschedule a frequency-based scheduled process
- start, stop, and resume scheduling on a frequency-based scheduler
- connect a timing source to and disconnect it from a frequency-based scheduler
- control use of the real-time clock device as the timing source for a frequency-based scheduler
- configure up to 100 frequency-based schedulers system-wide in a single processor or multiprocessor environment
- use both frequency-based scheduling and static priority scheduling simultaneously
- set the soft overrun limit for a frequency-based scheduled process
- query the soft overrun limit and the total number of soft overruns incurred by a frequency-based scheduled process.

How Is Scheduler Frequency Defined?

Scheduler frequency is defined, in part, by defining the number of *minor cycles* that compose a *major frame* using either the `fbsconfigure(3)` library routine or `rtcp(1)`.

As the smallest unit of frequency maintained by a frequency-based scheduler, a minor cycle has a duration associated with it. The duration is the time that elapses between interrupts generated by the timing source attached to the scheduler. If, for example, the timing source is a real-time clock, the minor cycle duration is defined by specifying the number of clock counts per minor cycle and the number of microseconds per clock count.

A major frame is one pass through all of the minor cycles with which a frequency-based scheduler is configured. The duration of a major frame is determined by multiplying the duration of a minor cycle by the number of minor cycles per major frame.

If, for example, you configure a scheduler with 100 minor cycles per major frame and you use as the timing source a real-time clock with a clock count of 10,000 and a clock count duration of one microsecond, each minor cycle has a duration of 10,000 microseconds, or 0.01 second, and each frame a duration of one second.

End-of-cycle scheduling is triggered when the last process scheduled during the current minor cycle of the current major frame completes its processing.

How Are Processes Scheduled?

A process is scheduled to run at a certain frequency by specifying the first minor cycle in which the process is to be awakened in each major frame (called the *starting base cycle*) and the frequency with which it is to be awakened (called the *period*).

If, for example, you schedule `Process-1` with a starting base cycle of zero and a period of two, the process will be awakened once every two minor cycles, starting with the first minor cycle in the frame.

If you schedule `Process-2` with a starting base cycle of one and a period of four, that process will be awakened once every four minor cycles, starting with the second minor cycle in the frame.

If you then schedule `Process-3` with a starting base cycle of two and a period of two, that process will be awakened once every two minor cycles, starting with the third minor cycle in the frame.

On a scheduler configured with 100 minor cycles per major frame, these processes are awakened as illustrated in Table 2-1.

Table 2-1. Process Scheduling Example 1

Minor Cycle	Processes Awakened
0	Process-1
1	Process-2
2	Process-1, Process-3
3	
4	Process-1, Process-3
5	Process-2
.	
.	
.	
97	Process-2
98	Process-1, Process-3
99	

The maximum frequency that a process can be scheduled is once per minor cycle (a period of one); the minimum frequency is once per major frame (in the case of the example, a period of 100).

A process runs until it calls an FBS library routine that causes it to sleep. FBS wakes those sleeping processes that are scheduled to be awakened in the current minor cycle of the current major frame and repeats the process for each minor cycle in the current frame. It continues to repeat the entire process on every major frame until the scheduler is disabled.

To extend the previous example, we can use the three processes from the previous example:

- Process-1 - starting base cycle=0, period=2
- Process-2 - starting base=1, period=4,
- Process-3 - starting base=2, period=2

If we also configure a scheduler with 100 minor cycles per major frame, a minor cycle duration of 10,000 microseconds (0.01 second), and a major frame duration of one second, the three processes are scheduled as illustrated in Table 2-2.

Table 2-2. Process Scheduling Example 2

Major Frame	Time (sec.)	Minor Cycle	Processes Awakened
0	0	0	Process-1
	0.01	1	Process-2
	0.02	2	Process-1, Process-3
	...		
	0.97	97	Process-2
	0.98	98	Process-1, Process-3
	0.99	99	
1	1.00	0	Process-1
	1.01	1	Process-2
	1.02	2	Process-1, Process-3
	...		
	1.97	97	Process-2
	1.98	98	Process-1, Process-3
	1.99	99	
<i>n</i>	<i>n</i> .00	0	Process-1
	<i>n</i> .01	1	Process-2
	<i>n</i> .02	2	Process-1, Process-3
	...		
	<i>n</i> .97	97	Process-2
	<i>n</i> .98	98	Process-1, Process-3
	<i>n</i> .99	99	

As illustrated in Table 2-2, when the current major frame is zero and the current minor cycle is zero, the scheduler wakes Process-1. After 0.01 second, it wakes Process-2; after 0.02 second, it wakes Process-1 and Process-3; and so on.

At one second, when the current major frame becomes one, the current minor cycle becomes zero again, and the scheduler wakes Process-1. After 0.01 second, it wakes Process-2; after 0.02 second, it wakes Process-1 and Process-3; and so on. The scheduler continues repeating this process for as long as it is enabled.

Tolerating Frame Overruns

A *frame overrun* occurs when a scheduled process does not finish its processing before it is scheduled to run again. Frame overruns are classified in two categories:

<i>hard overruns</i>	catastrophic failures of the scheduled process
<i>soft overruns</i>	<p>catastrophic failures only if the process reaches its limit on the number of soft overruns tolerated. Each scheduled process has a soft overrun limit, defaulting to 0.</p> <p>Letting a process survive a reasonable number of soft overruns makes the system more flexible and efficient. Some soft overruns result from random, unpredictable, or external events unlikely to recur. Other soft overruns result from only minor frame overruns. Soft overruns give the scheduled process a chance to recover from a frame overrun and return to synchronization.</p>

Note that there are no overruns associated with end-of-cycle scheduling. By definition, the cycle ends when all processes scheduled are blocked in the **fbwait(3)** call.

RedHawk Linux counts both soft and hard overruns for each scheduled process, but only hard overruns for each scheduler. Other processes can get these counts by querying the scheduled process or scheduler.

When scheduling a process, it is possible to specify that the scheduler is to be stopped by the kernel when that process running under it causes a hard overrun. Failing to specify that the scheduler should stop when a process causes a hard overrun allows the scheduler to continue to run, regardless of the number of hard overruns.

When scheduling a process, it is also possible to specify a consecutive soft overrun limit count that the process will tolerate and have processed as soft overruns by the kernel. Note that the default consecutive soft overrun limit is zero. With the limit set to zero, *all* overruns incurred by the process are treated as hard overruns (see below).

When a scheduled process overruns a frame and is not configured to be blocked in **fbwait(3)** when a frame interrupt occurs, the kernel decides whether to treat this overrun as a soft or hard overrun using the following steps:

- The consecutive soft overrun counter for the process is incremented.
- If the count does not reach or exceed the soft overrun limit for the process, the overrun is treated as a soft overrun. The process does not block the next time it calls **fbwait**; instead it returns immediately from the **fbwait** call with a status value of 2.
- Once the soft overrun limit reaches or exceeds the per-process limit, the overrun is treated as a hard overrun. The process blocks the next time it calls **fbwait**, and when the next normally scheduled frequency-based scheduler wakeup for that process occurs, the process returns out of the **fbwait** call returning a status value of 0. Note that a status of 2 is *not* returned in the hard overrun case.

User Interface

FBS is accessed using one of the following:

- the real-time command processor—**rtcp**. This program acts as a command interpreter for FBS, allowing you to perform key FBS operations by entering commands from the keyboard or invoking a script. **rtcp** is explained in Chapter 5, including an illustration showing the sequence in which you might invoke the commands.
- a set of library routines. This interface enables you to perform the entire range of functions associated with the scheduler from application programs written in C or FORTRAN. The C library routines are explained in Chapter 6; the FORTRAN library routines are explained in Chapter 7. Included are illustrations showing the sequence in which you might call the routines.
- NightSim RT. This is a Graphical User Interface (GUI) to the entire range of FBS functions. NightSim RT is explained in the *NightSim RT User's Guide*.

Debugging Frequency-Based Scheduler Processes

Debugging processes that have been scheduled on a frequency-based scheduler can be done using NightView RT a general-purpose, source-level debugger.

To be able to debug a C executable program, you must compile the source program specifying the **-g** option.

The NightView RT commands you can use to debug frequency-based scheduled processes are:

attach attach to a running process. This command allows you to debug a process that is already running

detach detach from an attached process. This command allows you to release an attached process from the control of the debugger.

To use NightView RT to debug a frequency-based scheduled process, you must supply the process ID (PID). You can easily obtain the PID for a frequency-based scheduled process by using the **ps (1)** command. You can obtain the PID for a selected process name by using the C library routine **nametopid (3)**. Use of each of these routines is explained in the corresponding man pages.

For NightView RT to attach to a running process, the debugger's effective user and group ID must match the effective user and group ID of the process controlled by the debugger.

For additional information on the procedures for using the **attach** and **detach** commands, see the *NightView RT User's Guide*.

Timing Sources

Overview	3-1
Using a Real-Time Clock	3-1
Understanding the Real-Time Clock Device	3-1
Understanding the User Interface	3-2
General Procedures for Using a Real-Time Clock	3-2
Using an Edge-Triggered Interrupt	3-3
Understanding the Edge-Triggered Interrupt	3-3
Understanding the User Interface	3-4
Using a User-Supplied Timing Device	3-4

This chapter contains the procedures for utilizing various timing sources for frequency-based scheduling.

Overview

The Real-Time Clock and Interrupt Module (RCIM) provides real-time clocks and edge-triggered interrupts suitable as a timing source for a frequency-based scheduler. Each RCIM provides up to eight real-time clock timers (RTCs) and up to twelve edge-triggered interrupts (ETIs). These can be used locally or distributed across systems connected in an RCIM chain. A user-defined device can also be used as a timing source.

Before running the scheduler, the timing source is selected by “attaching” it to the scheduler. If a real-time clock is used, you can define the duration of a minor cycle by setting the count and the resolution values.

The sections that follow provide information about the available timing sources.

Using a Real-Time Clock

This section provides information for using an RCIM real-time clock as the timing source for a frequency-based scheduler. Discussions include:

- an overview of the real-time clock device
- a description of the user-interface to the device
- general procedures for using the device

Understanding the Real-Time Clock Device

The real-time clock (RTC) device is designed for a variety of timing and frequency control functions. It provides a range of clock count values and a set of resolutions that, taken together, produce many different timing intervals—a feature that makes it particularly appropriate for frequency-based scheduling.

The Real-Time Clock and Interrupt Module (RCIM) provides up to eight real-time clocks (0-7) on each system. When multiple systems are connected using an RCIM chain, all RTCs can be designated to be distributed; that is, interrupts are sent to all connected systems. A distributed RTC may be located on any system within the RCIM chain.

Each RTC is referenced through its own character special device file:

`/dev/rcim/rtcN`

For more information, refer to the `rcim(4)` man page or the *Real-Time Clock and Interrupt (RCIM) PCI Form Factor User's Guide*.

Understanding the User Interface

When using a real-time clock as a timing source for a frequency-based scheduler, any of the interfaces to FBS can be used to attach the clock to or detach the clock from the scheduler, or to set, start and stop the clock. The actions to be performed and the `rtcp` commands and C and FORTRAN library routines that are used are given in the section “General Procedures for Using a Real-Time Clock” below. Using NightSim RT for selecting and operating a real-time clock for frequency-based scheduling is described in the *NightSim RT User's Guide*.

It is recommended that you use `rtcp` or the routines contained in the C library as your interface to the real-time clock. The `rtcp` command is the easiest to use because it isolates you from most of the initialization tasks.

Real-time clocks can be controlled directly by using the system calls `open(2)`, `close(2)`, and `ioctl(2)`. Note that this device does not support `read(2)` and `write(2)` system calls. See the `rcim_rtc(4)` man page for details.

General Procedures for Using a Real-Time Clock

Whether you elect to use `rtcp` or the routines contained in the libraries as your interface to the real-time clock, the general procedures for using a real-time clock for frequency-based scheduling are the same:

- STEP 1: Attach a real-time clock to a frequency-based scheduler.
- STEP 2: Establish the duration of a minor cycle by specifying the clock count value and the resolution per clock count.
- STEP 3: Start the real-time clock counting.
- STEP 4: Start the simulation.
- STEP 5: Stop the simulation.
- STEP 6: Stop the real-time clock counting.
- STEP 7: Detach the real-time clock.

The `rtcp` commands and C and FORTRAN routines that correspond to each step are presented in Table 3-1.

Table 3-1. rtcp Commands and Routines for using RTCs

Step	rtcp Command	C Routine	FORTRAN Routine
1	ats	fbsattach	fbsattach
2	stc	fbssetrtc	fbssetrtc
3	rc	fbsrunrtc	fbsrunrtc
4	start	fbsintrpt	fbsintrpt
5	stop	fbsintrpt	fbsintrpt
6	sc	fbsrunrtc	fbsrunrtc
7	dts	fbsdetch	fbsdetch

Refer to Chapter 5 for descriptions of the **rtcp** commands, Chapter 6 for explanations of the routines included in the C library and Chapter 7 for the FORTRAN routines.

Using an Edge-Triggered Interrupt

This section provides information for using an edge-triggered interrupt as the timing source for a frequency-based scheduler. Discussions include:

- an overview of the edge-triggered interrupt device
- a description of the user-interface to the device

Understanding the Edge-Triggered Interrupt

The edge-triggered interrupt (ETI) device is a software interface to the external interrupt lines provided by the RCIM. External interrupt lines provide a means for the computer system to detect an external interrupt coming into the system from any user device that generates a signal pulse. This signal pulse can be used as the timing source for a frequency-based scheduler.

There are up to twelve edge-triggered interrupts (0-11) per RCIM. When multiple systems are connected via an RCIM chain, all ETIs may be designated to be distributed; that is, interrupts are sent to all connected systems. A distributed ETI may be located on any system within the RCIM chain.

Each ETI is referenced through its own character special device file:

```
/dev/rcim/etiN
```

For more information, refer to the **rcim(4)** man page or the *Real-Time Clock and Interrupt (RCIM) PCI Form Factor User's Guide*.

Understanding the User Interface

When using an edge-triggered interrupt as a timing source for a frequency-based scheduler, any of the interfaces to FBS can be used to attach the interrupt to or detach it from the scheduler. Ensure that it is already generating interrupts when you start the simulation.

The **rtcp** commands and C and FORTRAN library routines that are used to attach and detach the interrupt from the scheduler are given in Table 3-2.

Table 3-2. rtcp Commands and Routines for using ETIs

rtcp Command	C Routine	FORTRAN Routine
ats	fbsattach	fbsattach
dts	fbsdetch	fbsdetch

Refer to Chapter 5 for descriptions of the **rtcp** commands, Chapter 6 for explanations of the routines included in the C library and Chapter 7 for the FORTRAN routines. Using NightSim RT for selecting and operating a real-time clock for frequency-based scheduling is described in the *NightSim RT User's Guide*.

Edge-triggered interrupts can be controlled directly by using the system calls **open(2)**, **close(2)**, and **ioctl(2)**. Note that this device does not support the **read(2)**, **write(2)** and **mmap(2)** system calls. See the **rcim_eti(4)** man page for details.

Using a User-Supplied Timing Device

You may wish to use your own device as the timing source for a frequency-based scheduler. To use your own device, you must ensure the following:

- Your device driver supports the **IOCTLVECNUM ioctl** call
- Your device driver supports the **IOCTLKEEPALIVE ioctl** call
- Your device generates a series of interrupts

FBS makes the **IOCTLVECNUM ioctl** call to provide the timing source driver with an **IRQ** that is associated with the FBS interrupt handler routine. The timing source driver must invoke this interrupt handler by calling **invoke_irq()** from its interrupt handler.

The **ioctl** call requires the following specifications:

```
#include <linux/ioctl/vecnum.h>
```

```
ioctl (fd, IOCTLVECNUM, arg);
```

fd the file descriptor for the device

IOCTLVECNUM the command to pass the **IRQ** to the timing source driver

arg an IRQ whose interrupt handler(s) is to be invoked in addition to the interrupt handler naturally attached to this device. A value of zero (0) disables a previous association.

To prevent the associated device from shutting down when the final close to the device is made, the following is also provided:

```
#include <linux/ioctl/keepalive.h>
```

```
ioctl (fd, IOCTLKEEPALIVE, arg);
```

arg when nonzero, enables the keepalive state enabling the device to continue running, if possible, after final close of the device. When zero (0), disables the keepalive state.

At the bottom of the interrupt handler, the following is done to invoke the handler(s) for the given IRQ:

```
#include <linux/interrupt.h>
```

```
if (irq)
    invoke_irq (irq, (struct pt_regs *)0);
```

Examples of Linux drivers implementing this interface can be found in the following RedHawk kernel source directory:

```
<kernel-sourcedir>/drivers/char/rcim
```


Using the Performance Monitor

What Is the Performance Monitor?	4-1
What Values Are Monitored?	4-1
Monitoring Idle Time	4-3
Monitoring Unscheduled Processes	4-4
User Interface	4-5
Optimizing the Performance of a Simulation	4-5

Using the Performance Monitor

This chapter contains a description of the Performance Monitor and its capabilities. It explains the values that are monitored and how to enable monitoring of system idle time and unscheduled processes. Suggestions for optimizing the performance of a simulation are given and user interface methods are introduced.

What Is the Performance Monitor?

The Performance Monitor (PM) allows you to monitor use of a CPU by processes that are scheduled using a frequency-based scheduler.

The Performance Monitor provides the ability to:

- obtain PM values by process or processor
- control all performance monitoring features from one processor (that is, enable performance monitoring for any processor)
- start and stop performance monitoring by process or processor
- clear PM values by process or processor
- set the timing mode to include or exclude interrupt times

Note that high-resolution process accounting must be configured in the system in order to use the Performance Monitor. See the “Configuration” section in Chapter 1 for more information.

What Values Are Monitored?

The performance monitor keeps track of the time that a process spends running from the time that it is awakened by a frequency-based scheduler until it calls `fbwait(3)`. Time is measured in microseconds.

One instance of a process being awakened by a scheduler is referred to as an *iteration* or a *cycle*. PM values for frequency-based scheduled processes are reported both in terms of cycles or iterations, and in terms of *major frames*. They reflect what has happened since the last time PM values were cleared and performance monitoring was enabled.

When the PM timing mode is set to include interrupt time, the user and system times of a process total the elapsed time that accrues when the process is running; time spent servicing interrupts is added to the system time of the process. When the timing mode is set to exclude interrupt time, a process' user and system times total the time that accrues when the process is the currently running process, excluding time spent servicing interrupts. Whether the timing mode is set to include or exclude interrupt time, context switch time is always included in the new process' system time.

Whether performance monitoring is enabled for a single frequency-based scheduled process or for all frequency-based scheduled processes on a processor, the following types of values are maintained for each process:

Total iterations, cycles	The number of times the process has been awakened by the scheduler
Last time	The amount of time the process has spent running from the last time it has been awakened by the scheduler until it has called <code>fbwait(3)</code>
Total time	The total amount of time the process has spent running in all cycles
Minimum cycle time	The least amount of time the process has spent running in a cycle
Minimum cycle cycle	The number of the minor cycle in which the minimum cycle time has occurred
Minimum cycle frame	The number of the major frame in which the minimum cycle time has occurred
Maximum cycle time	The greatest amount of time the process has spent running in a cycle
Maximum cycle cycle	The number of the minor cycle in which the maximum cycle time has occurred
Maximum cycle frame	The number of the major frame in which the maximum cycle time has occurred
Minimum frame time	The least amount of time the process has spent running during a major frame
Minimum frame frame	The number of the major frame in which the minimum frame time has occurred
Maximum frame time	The greatest amount of time the process has spent running during a major frame
Maximum frame frame	The number of the major frame in which the maximum frame time has occurred
Number of overruns	The number of times the process has caused a frame overrun

Monitoring Idle Time

The Performance Monitor can monitor a processor's *idle time*. Idle time refers to the time the CPU is not busy.

By monitoring a processor's idle time, you can determine the amount of CPU time available for allocation to additional processes.

You monitor a processor's idle time by adding the process **/idle** to a frequency-based scheduler and scheduling it on the desired processor. You can monitor idle time for a number of different processors by adding **/idle** to a selected frequency-based scheduler more than once and scheduling it on a different processor each time. You can also add **/idle** to more than one frequency-based scheduler. It is important to note, however, that you can schedule **/idle** on a particular processor only once.

To add **/idle** to a frequency-based scheduler, do one of the following:

- execute the **rtcp** command **sp**
- make a call to **sched_pgmadd(3)** from a C program
- make a call to **schedpgmadd(3f)** from a FORTRAN program

A description of the **sp** command is provided on page 5-32. A description of the **sched_pgmadd(3)** routine can be found on page 6-38 and on the man page. A description of the **schedpgmadd(3f)** routine can be found on page 7-47 and on the man page.

When you add **/idle** to a frequency-based scheduler, the only parameter you must specify is the CPU. The default scheduling priority for **/idle** is zero. The starting base cycle is zero, and the period is one. The **/idle** process is scheduled every minor cycle, starting with the first minor cycle in each major frame.

NOTE

When using the **sched_pgmadd(3)** subroutine to add **/idle** to a scheduler, only one bit can be set in the bit mask that specifies the processor. To add **/idle** to a scheduler and schedule it on more than one processor, you must call the subroutine repeatedly, specifying a different processor on each call.

After **/idle** is scheduled, a unique frequency-based scheduler identifier is returned. Use this number to identify **/idle** when performing tasks related to FBS or PM.

You obtain scheduling information for **/idle** in the same way you obtain it for other frequency-based scheduled processes, by doing one of the following:

- execute the **rtcp** command **vp**
- make a call to **sched_fbsqry(3)** or **sched_pgmqry(3)** from a C program
- make a call to **schedfbsqry(3f)** or **schedpgmqry(3f)** from a FORTRAN program

A description of the **vp** command is provided on page 5-35. A description of the **sched_fbsqry** routine is provided on page 6-32, **sched_pgmqry** is on page 6-41, **schedfbsqry(3f)** is on page 7-44, **schedpgmqry(3f)** is on page 7-49 and descriptions are on the respective man pages.

If you enable performance monitoring for the processor(s) on which **/idle** is scheduled or for the process itself, you can obtain all PM values.

Monitoring Unscheduled Processes

The Performance Monitor provides the additional capability of monitoring the performance of *unscheduled processes*. Unscheduled processes are those that are not awakened by the scheduler and do not call **fbswait**.

To be able to obtain PM values for such processes, add them to a frequency-based scheduler and specify a starting base cycle of zero and a period of zero. Other scheduling parameters you must specify include the process scheduling priority and the CPU on which it is to be scheduled. Optionally specify an octal value to be passed to a process that is scheduled on a frequency-based scheduler. The halt on overrun flag does not apply to an unscheduled process.

You add unscheduled processes to a frequency-based scheduler by doing one of the following:

- executing the **rtcp** command **sp**
- making a call to **sched_pgmadd(3)** from a C program
- making a call to **schedpgmadd(3f)** from a FORTRAN program

An explanation of the **sp** command is provided on page 5-32. A description of the **sched_pgmadd** routine is located on page 6-38, **schedpgmadd** is on page 7-47 and descriptions are on the respective man pages.

After a process is scheduled, the unique frequency-based scheduler identifier is returned. You can subsequently use this number to identify the process when you are performing tasks related to FBS or PM.

You obtain scheduling information for unscheduled processes in the same way you obtain it for other frequency-based scheduled processes by doing one of the following:

- executing the **rtcp** command **vp**
- making a call to **sched_pgmqry(3)** from a C program
- making a call to **schedpgmqry(3f)** from a FORTRAN program

An explanation of the **vp** command is provided on page 5-35. A description of the **sched_pgmqry** routine can be found on page 6-41, **schedpgmqry** is on page 7-49 and descriptions are on the respective man pages.

PM values maintained for unscheduled processes include:

- last time
- total time
- minimum frame time and the number of the frame in which it occurred
- maximum frame time and the number of the frame in which it occurred

You obtain these values by enabling performance monitoring for the processor(s) on which the processes have been scheduled or on the individual processes.

User Interface

PM is accessed using one of the following:

- the real-time command processor—**rtcp**. This program acts as a command interpreter for PM, allowing you to perform key performance monitoring operations by entering commands from the keyboard or invoking a script. **rtcp** is explained in Chapter 5, including an illustration showing the sequence in which you might invoke the commands.
- a set of library routines. This interface enables you to perform the entire range of functions associated with PM from application programs written in C and FORTRAN. The library routines are explained in Chapter 6 and Chapter 7, respectively, including an illustration showing the sequence in which you might call the routines.
- NightSim RT. This is a Graphical User Interface (GUI) to the entire range of PM functions. NightSim RT is explained in the *NightSim RT User's Guide*.

Optimizing the Performance of a Simulation

One of the benefits of using multiprocessor systems for real-time processing is that you can optimize the performance of a simulation by distributing processes among several processors.

Using FBS to schedule the programs that make up a simulation allows you to use the Performance Monitor to determine the extent to which the processes are using a CPU and to find out whether or not they are running at the specified frequency.

A program is scheduled on a processor when it is added to a frequency-based scheduler. The processor on which it is scheduled is determined by the CPU bias (affinity) specified when it is added to the scheduler. After programs have been scheduled, enable performance monitoring on one or more processes or processors and run the simulation.

By examining the PM values that are maintained for each frequency-based scheduled process, you can determine:

- the processors to which the processes have been assigned
- the amount of time the processes have spent running
- the processes that have not run at their assigned frequency

If you find that a process is not running at its assigned frequency, examine the frequency, the amount of CPU time being used by the other processes, and the CPU biases for all processes.

Note that if the CPU bias of a process identifies more than one processor, you cannot determine how much time the process has spent on a particular CPU specified in the bit mask because of dynamic load balancing. To avoid dynamic load balancing, specify only one processor in the bit mask. By using performance monitoring, you can then tell how much time a process has spent on its assigned CPU. You can redistribute processes as necessary.

You can also enable performance monitoring for a processor's idle time. The procedures for doing so are explained in the section "Monitoring Idle Time" in this chapter. By examining the amount of idle time on each processor, you will be able to identify the processors that have the lightest load and calculate the additional amount of CPU time that can be used for scheduling real-time processes.

You can determine the processor assignments that are optimal for your simulation by analyzing the PM values for frequency-based scheduled processes and for idle time on selected processors. Redistribute your frequency-based scheduled processes as necessary by changing their CPU biases.

It is important to note that in order to do so, you must first remove the process from the scheduler on which it has been scheduled and then add it to a scheduler. For an overview of the Frequency-Based Scheduler and the interfaces that accommodate its use, refer to Chapter 2.

The performance of a task can also be influenced by the way in which the system's CPUs are configured. See the "Real-Time Performance" chapter of the *RedHawk Linux User's Guide* for more about CPU shielding and hyper-threading.

What Is the Real-Time Command Processor?	5-1
rtcp and the Frequency-Based Scheduler	5-1
rtcp and the Performance Monitor	5-2
Execution Modes	5-2
Using Direct Mode	5-2
Invoking rtcp with a Command Name and Arguments	5-3
Invoking rtcp with Commands Redirected from a Script File	5-3
Invoking a Script that Calls rtcp	5-3
Using Interactive Mode	5-4
Getting Help	5-5
rtcp Commands	5-7
rtcp Command Summary	5-7
Command Sequence	5-8
Using rtcp Commands	5-10
ats – Attach Timing Source to a Frequency-Based Scheduler	5-11
chs – Change Permissions for a Frequency-Based Scheduler	5-12
cs – Configure a Frequency-Based Scheduler	5-13
dts – Detach Timing Source from a Frequency-Based Scheduler	5-15
rms – Remove a Frequency-Based Scheduler	5-16
svs – Save Scheduler Configuration	5-17
vc – View Minor Cycle/Major Frame Count	5-18
vs – View Scheduler Configuration	5-19
ls – Display All Schedulers on the System	5-21
rc – Start Real-Time Clock	5-21
sc – Stop Real-Time Clock	5-22
stc – Set Real-Time Clock	5-22
gtc – Display Real-Time Clock Settings	5-23
start – Start Scheduling on a Frequency-Based Scheduler	5-24
resume – Resume Scheduling on a Frequency-Based Scheduler	5-25
stop – Stop Scheduling on a Frequency-Based Scheduler	5-25
rmp – Remove a Process from a Frequency-Based Scheduler	5-26
rsp – Reschedule a Process	5-28
sp – Schedule a Process on a Frequency-Based Scheduler	5-32
vp – View Processes on a Frequency-Based Scheduler	5-35
cpm – Clear Performance Monitor Values	5-38
pm – Start/Stop Performance Monitoring	5-40
vcm – View/Modify Performance Monitor Timing Mode	5-42
vpm – View Performance Monitor Values	5-43
ex – Exit Real-Time Command Processor	5-47
he – Display Help Information	5-47

This chapter describes the real-time command processor, **rtcp**. It describes the execution modes, the sequence of commands to be used for frequency-based scheduling and performance monitoring, how to get help, and a complete description of each command.

What Is the Real-Time Command Processor?

The real-time command processor (**rtcp**) is a program that acts as a command interpreter for the Frequency-Based Scheduler and the Performance Monitor. **rtcp** allows you to perform key operations by entering commands from the keyboard or invoking a script from the shell command line. It reads the commands and interprets them as requests to execute the related services.

There are two modes of execution: direct and interactive. These modes are described in detail on page 5-2.

rtcp also has a help facility that makes it possible for you to obtain online information about commands and arguments. Procedures for using the help facility are explained on page 5-5.

rtcp and the Frequency-Based Scheduler

rtcp commands associated with FBS enable you to perform such key operations as:

- configuring a scheduler
- scheduling programs
- saving a scheduler configuration
- setting up a timing source
- running a simulation
- querying status

See Table 5-1 for a summary of the **rtcp** commands associated with the Frequency-Based Scheduler. An overview of the Frequency-Based Scheduler is provided in Chapter 2. It is recommended that you read this chapter and the chapter on Performance Monitor before using **rtcp**.

rtcp and the Performance Monitor

rtcp commands associated with the Performance Monitor enable you to perform such key operations as:

- clearing performance monitor values
- starting and stopping performance monitoring
- setting the timing mode
- querying values

See Table 5-1 for a summary of the **rtcp** commands associated with the Performance Monitor. An overview of the Performance Monitor is provided in Chapter 4. It is recommended that you read this chapter and the chapter on the Frequency-Based Scheduler before using **rtcp**.

Execution Modes

The real-time command processor provides two modes for executing commands:

direct mode	enables you to invoke real-time command processor commands from the shell command line
interactive mode	enables you to invoke the real-time command processor, itself, from the shell command line and then to enter the desired commands from within the command processor

Using Direct Mode

Use direct mode in one of the following ways. Each method is described below.

- Invoke the real-time command processor with a command name and its arguments at the system command prompt.
- Invoke the real-time command processor at the system command prompt, redirecting the standard input to come from a file instead of the terminal keyboard.
- Invoke a script at the system command prompt.

Invoking *rtcp* with a Command Name and Arguments

Use the following format when invoking *rtcp* with a command and arguments:

```
$ rtcp command [-option [argument]] [-option [argument]] . . .
```

Note that you are allowed to enter only one command on the command line at a time. If you need more than one line to enter a command and its arguments, enter a backslash (\) at the end of the line to continue on the next line.

Invoking *rtcp* with Commands Redirected from a Script File

This method requires that you create a file that contains the *rtcp* commands you wish to execute, and the file is used as standard input to *rtcp*. Create the file using a text editor of your choice or by executing the *svs* (Save Scheduler Configuration) command. The *svs* command is explained on page 5-17.

If you use a text editor to create the file, enter each command on a separate line. You may use either of the following formats:

```
rtcp command [-option [argument]] [-option [argument]] . . .
```

or

```
command [-option [argument]] [-option [argument]] . . .
```

If you need more than one line to enter a command and its arguments, enter a backslash (\) at the end of the line to continue on the next line.

After you have created the file, invoke the real-time command processor, and redirect the standard input to come from the file:

```
$ rtcp < rtcp_input_file
```

Invoking a Script that Calls *rtcp*

This method requires that you create an executable file that contains the *rtcp* commands you wish to execute. The first line in the file can contain a command or the following text:

```
#!program_name
```

where *program_name* specifies the name of the file that contains the shell to be invoked; for example */bin/bash*.

Each command must be entered on a separate line according to the following format:

```
rtcp command [-option [argument]] [-option [argument]] . . .
```

If you need more than one line to enter a command and its arguments, enter a backslash (\) at the end of the line to continue on the next line.

After you have created the file, make it an executable file by using the *chmod* (1) command, then invoke it from the command line as you would any other command:

```
$ chmod 755 rtcp_script
$ rtcp_script
```

For information on use of the **chmod(1)** command, see the corresponding man page. See Appendix A for an example of a real-time command processor script.

Using Interactive Mode

Use the following procedure to interactively invoke the real-time command processor:

1. Specify the **rtcp** command at the system command prompt:

```
$ rtcp
```

The real-time command processor prompt is then displayed:

```
rtcp>
```

2. At the prompt, type real-time processor commands by using the following format:

```
rtcp>command [-option [argument]] [-option [argument]] . . .
```

If you need more than one line to enter a command and its arguments, enter a backslash (\) to cause the line to be continued.

In most instances, if the command is successfully executed, a message is displayed and, where applicable, configuration data, scheduling information, or performance monitor values are displayed.

Messages and data associated with the commands are included in the reference information that begins on page 5-7. If an error occurs, a message indicating the nature of the error is displayed. Error messages are listed and described in Appendix B.

3. To exit the command processor and return to the shell, type the following:

```
rtcp>ex
```

The system command prompt is again displayed.

Getting Help

Two methods of accessing information about **rtcp** online are available:

- **rtcp(1)** man page
- **rtcp** help facility

Access the **rtcp** help facility using the **he** command. The information provided includes:

- a list and brief description of all **rtcp** processor commands
- a description and format of a particular command
- a list and description of all command arguments

To display a list of all commands, enter the **he** command as follows:

```
he
```

Commands are displayed as illustrated in Screen 5-1.

Screen 5-1. rtcp Command Display

```

rtcp commands

ats - attach timing source to FBS
cs  - configure FBS
rms - remove FBS
vc  - view current frame/cycle count
ls  - list FBSs

rc  - run real-time clock
stc - set real-time clock values

start - start FBS
stop  - stop FBS

rmp - remove a process on a FBS
sp  - schedule a process on a FBS

cpm - clear performance monitor tables
vcm - view/modify PM timing mode

he  - help

chs - modify FBS access permissions
dts- detach timing source from FBS
svs- save FBS configuration to a file
vs  - view FBS configuration

sc  - stop real-time clock
gtc- get real-time clock values

resume - resume FBS

rsp- reschedule a process on a FBS
vp  - view scheduled processes on FBS

pm  - start/stop performance monitor
vpm- view performance

ex  - exit rtcp

```

To display a description of a particular command, enter **he** with the name of the command as an argument. For example, to display a description of the **ats** command, specify:

```
he ats
```

A description of the **ats** command and the format for entering the command are displayed as follows:

```
Attach timing source to a FBS
```

```
rtcp ats -s scheduler -d device | -e
```

To display a list of command arguments, enter **he** with **option** as the argument:

```
he option
```

The first screen of **rtcp** arguments is displayed as illustrated in Screen 5-2.

Screen 5-2. First Screen of rtcp Arguments

rtcp parameters

```
-a          remove program from FBS and terminate
-b {F|R|O} scheduling policy
-c cpu_bias cpu bias (* = all CPUs) (default = current CPU)
-d name     devicename or filename
-e          EOC flag
-f frequency number of minor cycles to next wakeup (default = 1)
-i fpid     process fpid number (default = -1)
-m start_cycle 1st minor cycle to wakeup (default = 0)
-n proc_name process name
-o {halt|nohalt} halt FBS on overrun flag (default = nohalt)
-p priority process priority
-s scheduler FBS scheduler key
-t {in|ex}  include or exclude interrupt time in pm monitor
-v parameter process initiation parameter
-x {av|mi|ma|al} performance monitor display option (default = average)
```

```
Enter 'he op2' for more parameters
```


To display the second screen of **rtcp** arguments, enter **he** with **op2** as the argument:

```
he op2
```

The second screen of **rtcp** arguments is displayed as illustrated in Screen 5-3.

Screen 5-3. Second Screen of rtcp Arguments

```

rtcp parameters

-C cycles/frame      number of minor cycles per major frame
-D duration          clock tick duration (default = 10us)
-G gid               effective group ID for FBS (default = current user)
-I permissions       permissions for FBS in octal (default = 0600)
-L soft_limit        soft overrun limit (default = 0)
-M progs/cycle       maximum number of processes per minor cycle
-N progs/fbs         maximum number of processes per FBS
-O clock_ticks       number of clock ticks per minor cycle
-P {ON|OFF}          enable/disable performance monitor (default = OFF)
-R {-1 | 0 | 1}      reset process flag (default = 0)
-S delay             time to delay, in seconds
-U uid               effective user ID for FBS (default = current user)

```

rtcp Commands

rtcp Command Summary

rtcp commands are listed in Table 5-1. Complete information about each command as well as a command sequence flowchart are provided below.

Table 5-1. rtcp Commands

Command	Page	Description
Frequency-Based Scheduler:		
ats	5-11	Attach timing source to a frequency-based scheduler
chs	5-12	Change permissions for a frequency-based scheduler
cs	5-13	Configure a frequency-based scheduler
dts	5-15	Detach timing source from a frequency-based scheduler
gtc	5-23	Get real-time clock values
ls	5-21	List all frequency-based schedulers configured on the system
rc	5-21	Start real-time clock

(continued on next page)

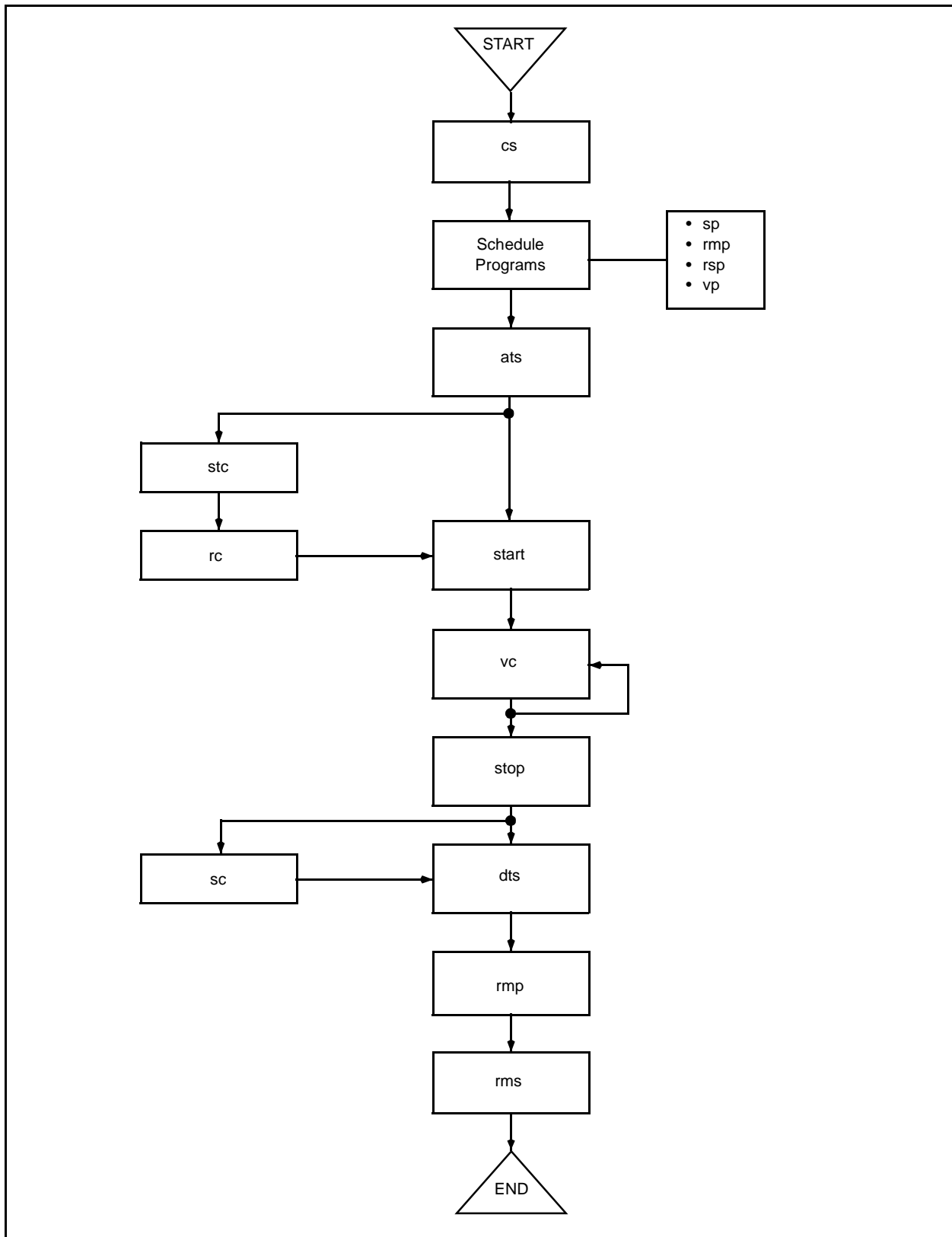
Table 5-1. rtcp Commands (Continued)

Command	Page	Description
Frequency-Based Scheduler (Continued):		
resume	5-25	Resume scheduling on a frequency-based scheduler
rmp	5-26	Remove a process from a frequency-based scheduler
rms	5-16	Remove a frequency-based scheduler
rsp	5-28	Reschedule a process
sc	5-22	Stop real-time clock
sp	5-32	Schedule a process on a frequency-based scheduler
start	5-24	Start scheduling on a frequency-based scheduler
stc	5-22	Set real-time clock values
stop	5-25	Stop scheduling on a frequency-based scheduler
svs	5-17	Save scheduler configuration
vc	5-18	View minor cycle/major frame count
vp	5-35	View processes on a frequency-based scheduler
vs	5-19	View scheduler configuration
Performance Monitor:		
cpm	5-38	Clear performance monitor values
pm	5-40	Start/stop performance monitoring
vcm	5-42	View or modify performance monitor timing mode
vpm	5-43	View performance monitor values
rtcp:		
ex	5-47	Exit real-time command processor
he	5-47	Display help information

Command Sequence

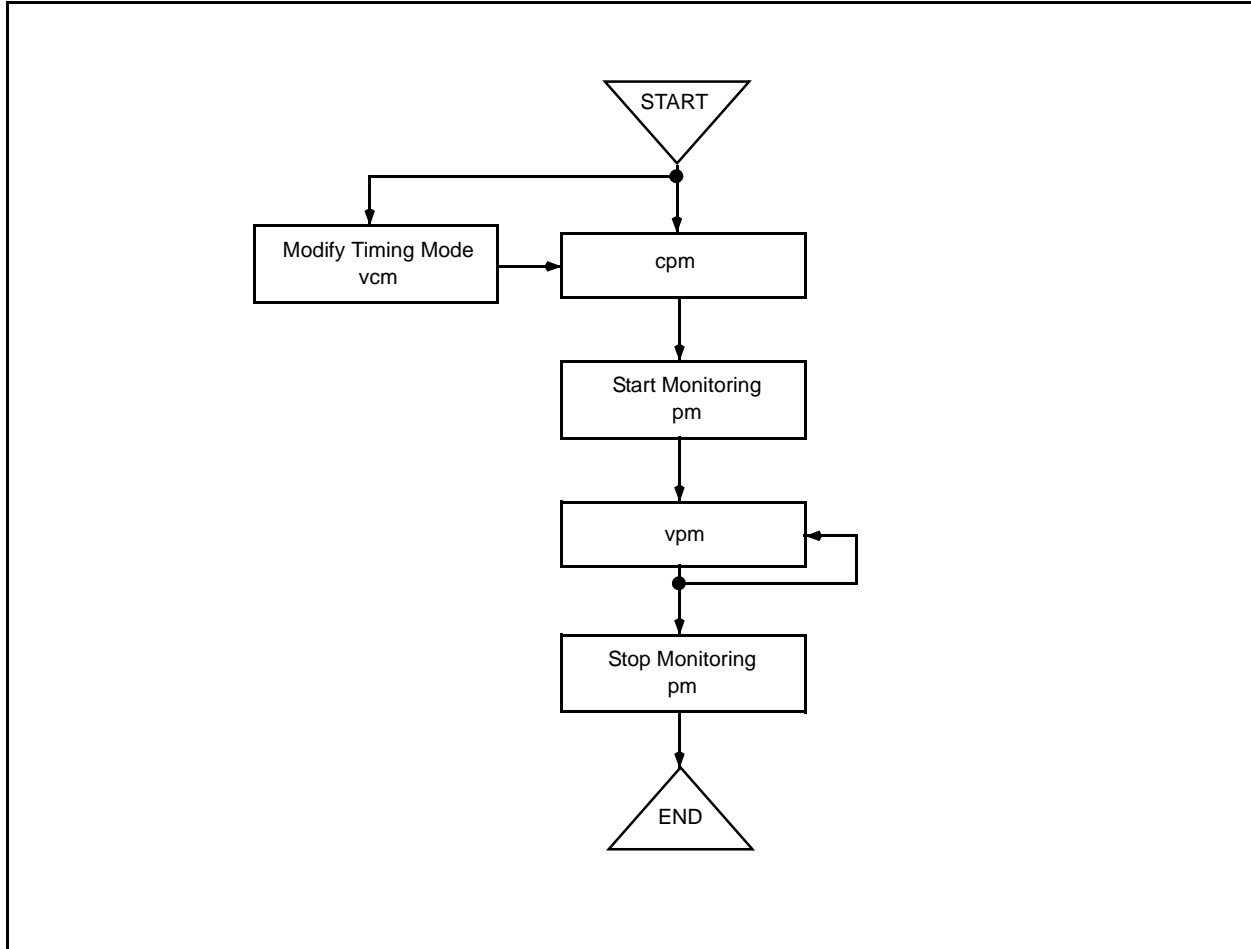
The order in which you might execute the **rtcp** commands associated with the Frequency-Based Scheduler is illustrated in Figure 5-1.

Figure 5-1. rtcp Command Sequence: Frequency-Based Scheduler



The order in which you might execute the commands associated with the Performance Monitor is illustrated in Figure 5-2.

Figure 5-2. rtcp Command Sequence: Performance Monitor



Using rtcp Commands

This section provides reference information for all of the commands supported by the real-time command processor. Commands are presented in the order in which they are described in the help facility. For each command, the following information is provided:

- a description of the command
- the format for entering the command
- detailed descriptions of each argument
- an example of the output from the command

ats – Attach Timing Source to a Frequency-Based Scheduler

The **ats** command attaches a timing source to a frequency-based scheduler or specifies end-of-cycle scheduling. In the latter case, scheduling is triggered when the last process scheduled during the current minor cycle completes its processing.

Synopsis

```
ats -s scheduler {-d device | -e}
```

Arguments

- s** *scheduler* the numeric key associated with the frequency-based scheduler for which the timing source is to be attached or end-of-cycle scheduling specified. The scheduler must previously have been configured. It can be any positive integer value.
- d** *device* the path name of the real-time clock or edge-triggered interrupt being used as the timing source for the specified scheduler. Refer to Chapter 3 for detailed information on the form associated with each type of device.
- e** specifies end-of-cycle scheduling. In this case, execution of the processes in the next minor cycle occurs when the last process scheduled to run in the current minor cycle finishes its processing for the cycle.

Display

The following message is displayed when the specified timing source is successfully attached to the scheduler or if end-of-cycle scheduling is successfully enabled:

```
Scheduler attached
```

chs – Change Permissions for a Frequency-Based Scheduler

The **chs** command changes the permissions assigned for a frequency-based scheduler. In order to do this, you must have an effective user ID equal to that of the creator/owner of the frequency-based scheduler.

Synopsis

```
chs -s scheduler [-I permissions] [-G gid] [-U uid]
```

Arguments

-s *scheduler* the numeric key associated with the frequency-based scheduler for which the permissions are to be changed. The scheduler must previously have been configured. See the section “cs – Configure a Frequency-Based Scheduler” on page 5-13 for information on configuring a frequency-based scheduler.

-I *permissions* the permissions required for operations related to the specified scheduler.

The *permissions* argument specifies three octal digits—the first indicates permissions granted to the owner, the second those granted to the group, and the third those granted to other users.

The octal method for changing permissions associated with a scheduler is the same as that used for specifying *mode* with the **chmod** command (see the **chmod (1)** man page). The default, **600**, grants read and alter (write) permission to the owner only.

-G *gid* the effective group ID of the selected frequency-based scheduler. The default effective group is the current user.

-U *uid* the effective user ID of the selected frequency-based scheduler. The default is the current user.

Display

The following message is displayed when the permissions assigned to the scheduler are successfully changed:

```
Scheduler permissions changed
```

cs – Configure a Frequency-Based Scheduler

The **cs** command creates a frequency-based scheduler.

To create a scheduler, you must specify a key, which is a user-selected numeric identifier with which the scheduler will be associated. You must also define:

- the number of minor cycles that compose a major frame on the scheduler
- the maximum number of tasks that can be scheduled during a minor cycle
- the maximum number of tasks scheduled on a scheduler at one time

A frequency-based scheduler uses two types of permission that control users' ability to perform scheduler operations: read and alter (write). Read permission is required to perform query operations. Alter permission is required to:

- schedule, remove, and reschedule programs
- attach a timing source to and detach it from a scheduler
- start, stop, and resume scheduling

Permissions are assigned when you create the scheduler. They are specified in the same way in which permissions associated with files are assigned. See the **chmod(1)** man page for assistance in specifying permissions.

When you execute the **cs** command, a unique, positive frequency-based scheduler identifier and corresponding data structure are created for the specified key if *both* of the following conditions are met:

- the key is not already associated to a frequency-based scheduler identifier
- the number of frequency-based schedulers already configured is less than the maximum number of schedulers allowed on your system

The newly created frequency-based scheduler identifier is displayed on the terminal.

When you specify a key that is already associated with a frequency-based scheduler, the corresponding frequency-based scheduler identifier is displayed on your terminal screen if *all* of the following conditions are met:

- the number of minor cycles specified by the **-C** *cycles/frame* argument matches the number of minor cycles associated with the existing scheduler
- the maximum specified by the **-M** *progs/cycle* argument is less than or equal to the maximum number of processes per minor cycle associated with the existing scheduler
- the maximum specified by the **-N** *progs/fbs* argument is less than or equal to the maximum number of processes allowed on the existing scheduler

If these conditions are not met, an error message is displayed.

Synopsis

```
cs -s scheduler -C cycles/frame -M progs/cycle -N progs/fbs [-I permissions] \  
[-R reset]
```

Arguments

-s scheduler a key for the frequency-based scheduler that you wish to create. The key is a user-selected numeric identifier with which the scheduler will be associated. The value of *scheduler* can be any positive integer value.

-C cycles/frame the number of minor cycles that compose a frame on the specified frequency-based scheduler.

-M progs/cycle the maximum number of programs that can be scheduled to execute during one minor cycle.

-N progs/fbs the maximum number of programs that can be scheduled on the specified scheduler at one time. This value must be less than or equal to the product obtained by multiplying the values specified for the *cycles/frame* and the *progs/cycle* arguments.

-I permissions permissions required for operations related to the specified scheduler.

The *permissions* argument specifies three octal digits—the first indicates permissions granted to the owner, the second those granted to the group, and the third those granted to other users.

The octal method for changing permissions associated with a scheduler is the same as that used for specifying *mode* with the **chmod** command (see the **chmod (1)** man page). The default, **600**, grants read and alter (write) permission to the owner only.

-R reset the manner in which processes currently scheduled on the specified scheduler are to be handled.

The value of the *reset* argument can be 1, -1, or 0.

Specifying 0 (the default) allows these processes to remain on the scheduler. Specifying 1 removes these processes from the scheduler but allows them to continue executing. Specifying -1 removes these processes from the scheduler and terminates them.

Display

The following message is displayed when the scheduler is successfully configured:

```
Scheduler 10 has FBS ID of 3
```

Scheduler

Indicates the user-specified key for the selected frequency-based scheduler. It is important to note that this value is required by most of the real-time command processor commands.

FBS ID

Indicates the unique positive integer value representing the identifier for the selected frequency-based scheduler.

dts – Detach Timing Source from a Frequency-Based Scheduler

The **dts** command detaches the timing source from a frequency-based scheduler or disables end-of-cycle scheduling. If the timing source is a real-time clock, it is recommended that you stop the clock prior to invoking this routine by making a call to **sc** (page 5-22).

Synopsis

```
dts -s scheduler
```

Arguments

-s scheduler the numeric key associated with the frequency-based scheduler for which the timing source is to be detached or end-of-cycle scheduling disabled. The scheduler must previously have been configured. See the section “**cs – Configure a Frequency-Based Scheduler**” on page 5-13 for information on configuring a frequency-based scheduler.

Display

The following message is displayed when the timing source is successfully detached from the scheduler or end-of-cycle scheduling is successfully disabled:

```
Scheduler detached
```

rms – Remove a Frequency-Based Scheduler

The **rms** command removes a frequency-based scheduler. Prior to executing this command, you must ensure that the timing source for the scheduler has been detached or that end-of-cycle scheduling has been disabled (see “dts – Detach Timing Source from a Frequency-Based Scheduler” on page 5-15 for information on use of the **dts** command).

Note that to remove a scheduler, the calling process must have an effective user ID equal to that of the owner/creator of the frequency-based scheduler.

Synopsis

```
rms -s scheduler [-a]
```

Arguments

- s scheduler** the numeric key associated with the frequency-based scheduler that you wish to remove. The scheduler must previously have been configured. See the section “cs – Configure a Frequency-Based Scheduler” on page 5-13 for information on configuring a frequency-based scheduler.
- a** all processes currently scheduled on the specified scheduler are to be removed from the scheduler and terminated. If this option is not specified, all processes currently scheduled on the specified scheduler are removed but continue executing.

Display

The command displays the following message when the specified scheduler is successfully removed:

```
Scheduler removed
```

svs – Save Scheduler Configuration

The **svs** command stores configuration and scheduling data for a selected frequency-based scheduler in a file.

When you execute this task, the **rtcp** commands entered from the command line are saved to the specified output file. The file can then be used as an **rtcp** input file. The commands saved by **svs** are those used to configure a scheduler (**cs**), schedule programs on it (**sp**), attach a timing source to it (**ats**), and set the real-time clock (**stc**).

Synopsis

```
svs -s scheduler -d output_file_name
```

Arguments

- | | |
|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| -s scheduler | the numeric key associated with the frequency-based scheduler for which you wish to store configuration and scheduling data. The scheduler must previously have been configured. See the section “cs – Configure a Frequency-Based Scheduler” on page 5-13 for information on configuring a frequency-based scheduler. |
| -d output_file_name | a standard path name identifying the file in which you wish configuration data to be stored. It can be a full or relative path name of up to 1024 characters. |

Display

No message is displayed if the scheduler configuration is successfully saved to a file.

vc – View Minor Cycle/Major Frame Count

The **vc** command displays the current minor cycle and major frame count values for a frequency-based scheduler. These values help determine the progress of a simulation.

Synopsis

vc -s scheduler

Arguments

-s scheduler the numeric key associated with the frequency-based scheduler for which you wish to view the current cycle and frame counts. The scheduler must previously have been configured. See the section “cs – Configure a Frequency-Based Scheduler” on page 5-13 for information on configuring a frequency-based scheduler.

Display

When successfully executed, the command displays the following types of information:

```
Major frame = 65 Minor cycle = 25
```

Major frame

Indicates the number of the current major frame for the simulation running on the selected scheduler.

Minor cycle

Indicates the number of the current minor cycle for the simulation running on the selected scheduler.

vs – View Scheduler Configuration

The **vs** command allows you to view information related to a frequency-based scheduler. Viewable information includes:

- the key and identifier associated with the scheduler
- the number of minor cycles per major frame, the maximum number of programs per minor cycle, and the maximum number of programs per scheduler
- the user and group IDs of the owner and creator of the scheduler
- the permissions assigned for the scheduler
- the total number of overruns for all processes on the scheduler
- an indication of whether the scheduler is in the run or the stop state
- the CPUs that are active in the system and the CPUs for which performance monitoring has been enabled
- the path name of the device that has been attached to the scheduler

Synopsis

vs -s scheduler

Arguments

-s scheduler the numeric key associated with the frequency-based scheduler for which you wish to view current information. The scheduler must previously have been configured. See the section “cs – Configure a Frequency-Based Scheduler” on page 5-13 for information on configuring a frequency-based scheduler.

Display

When successfully executed, the command displays configuration and status information:

```
Scheduler 417 has FBS ID of 32768:    Cycles per frame = 101
Max programs per cycle = 10:    Max programs per FBS = 110
owner uid    = 9999:    owner gid    = 101
creator uid = 9999:    creator gid = 101:
total overruns = 0:    access mode = 600:    flags word = 1
active CPU mask = '----xxxx':    active PM CPU mask = '----x--x'
interrupt device name = /dev/rcim/rtc2
FBS is currently running
```

Scheduler

Indicates the user-specified key for the selected frequency-based scheduler.

FBS ID

Indicates the unique, positive integer value representing the identifier for the selected frequency-based scheduler.

Cycles per frame

Indicates the number of minor cycles that compose a major frame on the selected scheduler.

Max programs per cycle

Indicates the maximum number of programs that can be scheduled in a minor cycle on the selected scheduler.

Max programs per FBS

Indicates the maximum number of programs that can be scheduled on the selected scheduler at one time.

owner uid

Indicates the user ID of the scheduler's owner.

owner gid

Indicates the group ID of the scheduler's owner.

creator uid

Indicates the user ID of the scheduler's creator.

creator gid

Indicates the group ID of the scheduler's creator.

total overruns

Indicates the total number of overruns for all processes on the selected scheduler.

access mode

Indicates an octal value representing the permissions assigned to the selected scheduler.

flags word

Indicates if a timing source has been attached to the selected scheduler or if end-of-cycle scheduling has been enabled. If true, this field displays 1; otherwise, it displays 0.

active CPU mask

Contains a mask of the CPUs that are active in the system. The rightmost position corresponds to the first logical CPU. The letter **x** signifies that a CPU is active; the dash (-) signifies that it is not.

active PM CPU mask

Contains a mask of the CPUs for which performance monitoring has been enabled. The rightmost position corresponds to the first logical CPU. The letter **x** signifies that performance monitoring has been enabled on a CPU; the dash (-) signifies that it has not.

interrupt device name

If a timing source has been attached to the selected scheduler, this field contains the full path name of the device. If end-of-cycle scheduling has been enabled, this field contains the following: EOC triggering.

FBS is

Indicates the state of the selected scheduler.

ls – Display All Schedulers on the System

The **ls** command displays a single line of information for each frequency-based scheduler configured on the system.

Synopsis

ls

Display

This command displays the following information for each configured scheduler:

ID	SCHEDULER	PERMISSIONS	OWNER_UID	OWNER_GID	STATE	DEVICE
0	37	--rw-rw-r--	fbsusr1	5309	running	/dev/rcim/rtc0
1	38	--rw-rw-r--	fbsusr2	5309	running	/dev/rcim/rtc1
2	39	--rw-rw-r--	fbsusr1	5309	stopped	/dev/rcim/rtc2
3	40	--rw-rw-r--	fbsusr3	5309	running	/dev/rcim/rtc3

rc – Start Real-Time Clock

The **rc** command starts the real-time clock that has been specified as the timing source for a selected frequency-based scheduler (see “ats – Attach Timing Source to a Frequency-Based Scheduler” on page 5-11 for an explanation of the **ats** command). Note that you must first have set the count and resolution values for the real-time clock by executing the **stc** command (page 5-22).

Synopsis

rc -s scheduler

Arguments

-s scheduler

the numeric key associated with the frequency-based scheduler for which you wish to start the attached real-time clock. The scheduler must previously have been configured. See the section “cs – Configure a Frequency-Based Scheduler” on page 5-13 for information on configuring a frequency-based scheduler.

Display

The following message is displayed when the real-time clock is successfully started:

Clock started

sc – Stop Real-Time Clock

The **sc** command stops the real-time clock that has been specified as the timing source for a selected frequency-based scheduler.

Synopsis

sc -s scheduler

Arguments

-s scheduler the numeric key associated with the frequency-based scheduler for which you wish to stop the attached real-time clock. The scheduler must previously have been configured. See the section “cs – Configure a Frequency-Based Scheduler” on page 5-13 for information on configuring a frequency-based scheduler.

Display

The following message is displayed when the real-time clock is successfully stopped:

```
Clock stopped
```

stc – Set Real-Time Clock

The **stc** command establishes the duration of a minor cycle by setting the count and duration values for a real-time clock that has been specified as the timing source for a selected frequency-based scheduler.

Synopsis

stc -s scheduler [-D clock_duration] -O clock_count

Arguments

-s scheduler the numeric key associated with the frequency-based scheduler to which the real-time clock has been attached. The scheduler must previously have been configured. See the section “cs – Configure a Frequency-Based Scheduler” on page 5-13 for information on configuring a frequency-based scheduler.

-D clock_duration the duration in microseconds of one clock count. The value of *clock_duration* must be one of the following: 1, 10, 100, 1000, 10000. The default value is 10.

-O clock_count the number of clock counts per minor cycle. The value of *clock_count* can range from one to 65535.

Display

The following message is displayed when the real-time clock is successfully set:

```
Clock set
```


gtc – Display Real-Time Clock Settings

The **gtc** command displays the current count and duration values for a real-time clock that has been specified as the timing source for a selected frequency-based scheduler. The clock must previously have been set using the **stc** command (page 5-22).

Synopsis

gtc -s scheduler

Arguments

-s scheduler the numeric key associated with the frequency-based scheduler to which the real-time clock has been attached. The scheduler must previously have been configured. See the section “cs – Configure a Frequency-Based Scheduler” on page 5-13 for information on configuring a frequency-based scheduler.

Display

When the real-time clock is currently set, this command displays the following information:

```
Clock count = 50: duration = 1000
```

start – Start Scheduling on a Frequency-Based Scheduler

The **start** command starts scheduling processes on a frequency-based scheduler. When you execute this command, the minor cycle, major frame, and overrun count values are set to zero.

Prior to executing this command, you must have executed the **ats** command to attach a timing source to the scheduler or to specify end-of-cycle scheduling (see “ats – Attach Timing Source to a Frequency-Based Scheduler” on page 5-11 for an explanation of the **ats** command).

If you have specified a real-time clock as the timing source for the scheduler, scheduling will not start until you have set the clock using the **stc** commands (page 5-22), and started the clock using the **rc** command (page 5-21).

If you have specified an edge-triggered interrupt as the timing source, it must already be generating interrupts in order for scheduling to start.

Synopsis

start -s scheduler

Arguments

-s scheduler the numeric key associated with the frequency-based scheduler on which you wish to start scheduling. The scheduler must previously have been configured. See the section “cs – Configure a Frequency-Based Scheduler” on page 5-13 for information on configuring a frequency-based scheduler.

Display

The following message is displayed when scheduling on the specified scheduler is successfully started:

```
FBS started
```

resume – Resume Scheduling on a Frequency-Based Scheduler

The **resume** command resumes scheduling processes on a selected frequency-based scheduler with the major frame, minor cycle, and overrun count values the same as they were when you stopped the scheduler.

Synopsis

resume -s scheduler

Arguments

-s scheduler the numeric key associated with the frequency-based scheduler for which you wish to resume scheduling. The scheduler must previously have been configured. See the section “cs – Configure a Frequency-Based Scheduler” on page 5-13 for information on configuring a frequency-based scheduler.

Display

The following message is displayed when scheduling on the specified scheduler is successfully resumed:

```
FBS resumed
```

stop – Stop Scheduling on a Frequency-Based Scheduler

The **stop** command stops scheduling processes on a selected frequency-based scheduler.

Synopsis

stop -s scheduler

Arguments

-s scheduler the numeric key associated with the frequency-based scheduler for which you wish to stop scheduling. The scheduler must previously have been configured. See the section “cs – Configure a Frequency-Based Scheduler” on page 5-13 for information on configuring a frequency-based scheduler.

Display

The following message is displayed when scheduling on the specified scheduler is successfully stopped:

```
FBS stopped
```

rmf – Remove a Process from a Frequency-Based Scheduler

The **rmf** command removes a process from a frequency-based scheduler. Identify the process that you wish to remove by specifying one of the following:

- the name of the process and the CPU on which it is scheduled
- the process' frequency-based scheduler process identifier
- the name of the process, the CPU on which it is scheduled, and its frequency-based scheduler process identifier

NOTE

The only method that can be used to identify a process that has been scheduled multiple times on the same CPU is to specify its frequency-based scheduler identifier.

Synopsis

```
rmf -s scheduler { -i fpid | -n proc_name [-c cpu_bias] } [-a]
```

Arguments

- s** *scheduler* the numeric key associated with the frequency-based scheduler on which the process is scheduled. The scheduler must previously have been configured. See the section “cs – Configure a Frequency-Based Scheduler” on page 5-13 for information on configuring a frequency-based scheduler.
- i** *fpid* the frequency-based scheduler process identifier for the process to be removed. This value is displayed when you successfully schedule a program by executing the **sp** command (page 5-32). If you have not identified the process by name, you must specify this argument.
- The default value for *fpid* is -1. If you accept the default value, you must identify the process by name and CPU.
- n** *proc_name* a standard path name identifying the process to be removed from the specified scheduler. This can be a full or relative path name of up to 1024 characters.
- If you do not specify this argument, you must provide the frequency-based scheduler process identifier by specifying the **-i** *fpid* argument.
- c** *cpu_bias* the processor(s) to be used with the value of the **-n** *proc_name* argument to identify the process to be removed from the specified scheduler.
- The value of *cpu_bias* may be a single CPU ID or a list of CPU IDs. CPU IDs range from zero to seven, where the number 0 represents the first logical CPU, 1 represents the second, and so on.

A list of CPU IDs may specify a sequence or a range of numbers—for example, **-c 1,3-5,7**. Note that you must use commas to separate items in the list. Specify the entire range of CPU IDs (0-7) by entering an asterisk (*).

If you do not specify the **-c *cpu_bias*** argument, the default processor is the CPU on which **rtcp** is currently executing.

-a the process is to be removed from the specified scheduler and terminated. If this option is not specified, the process is removed from the scheduler but allowed to continue executing.

Display

If the specified process is successfully removed from the scheduler, the following message is displayed:

```
Process removed
```

rsp – Reschedule a Process

The **rsp** command changes the scheduling parameters for a process that has been scheduled on a frequency-based scheduler. You may wish, for example, to change the process' scheduling policy or priority. You may also wish to change the frequency with which the process is scheduled to run. You cannot, however, change the CPU on which it has been scheduled.

The effective user ID of the calling process must match the effective user ID of the target process (the process for which the scheduling policy and priority are being set) for the following conditions:

- if you execute this command and you wish to change a process' scheduling policy to the first-in-first-out (FIFO) or the round-robin policy
- if you wish to change the priority of a process scheduled under the FIFO or the round-robin policy
- if you wish to raise the priority of a process scheduled under the time-sharing policy above a per-process limit

You can reschedule a process without first having executed the **rmp** command (page 5-26) to remove it from the scheduler or the **stop** command (page 5-25) to stop scheduling.

Use one of the following methods to identify the process that you wish to reschedule:

- specify the name of the process and the CPU on which it is scheduled
- specify the process' frequency-based scheduler process identifier
- specify the name of the process, the CPU on which it is scheduled, and its frequency-based scheduler process identifier.

NOTE

The only method that can be used to identify a process that has been scheduled multiple times on the same CPU is to specify its frequency-based scheduler identifier.

Synopsis

```
rsp -s scheduler { -i fpid | -n proc_name [-c cpu_bias] } [-f frequency] \
[-m start_cycle] [-b policy] [-p priority] [-o halt_flag] [-L soft_limit]
```

Arguments

- s scheduler** the numeric key associated with the frequency-based scheduler on which the process is scheduled. The scheduler must previously have been configured. See the section “cs – Configure a Frequency-Based Scheduler” on page 5-13 for information on configuring a frequency-based scheduler.
- i fpid** the frequency-based scheduler process identifier for the process to be rescheduled. This value is displayed when you execute the **sp** command (page 5-32). If you have not identified the process by name, you must specify this argument. The default value for

fpid is -1. If you use the default value, you must identify the process by name and CPU.

-n *proc_name* a standard path name identifying the process to be rescheduled. This can be a full or relative path name of up to 1024 characters. If you do not specify this argument, you must provide the frequency-based scheduler process identifier by specifying the **-i** *fpid* argument.

-c *cpu_bias* the processor(s) to be used with the value of the **-n** *proc_name* argument to identify the process to be rescheduled.

The value of *cpu_bias* may be a single CPU ID or a list of CPU IDs. CPU IDs range from 0 to 7, where the number 0 represents the first logical CPU, 1 represents the second, and so on.

A list of CPU IDs may specify a sequence or a range of numbers—for example, **-c 1,3-5,7**. Note that you must use commas to separate items in the list. Specify the entire range of CPU IDs (0-7) by entering an asterisk (*).

If you do not specify this argument, the default processor is the CPU on which the real-time command processor is currently executing.

-f *frequency* the frequency with which the specified process is to be awakened in each major frame.

A frequency of 1 indicates that the specified process is to be awakened every minor cycle; a frequency of 2 indicates that it is to be awakened once every two minor cycles and so on.

Specify the number of minor cycles representing the frequency with which you wish the process to be awakened. The value of *frequency* can range from 1 to the number of minor cycles that compose a frame on the scheduler. The default is 1.

The total number of minor cycles per frame is defined by executing the **cs** command (page 5-13).

-m *start_cycle* the first minor cycle in which the specified process is to be awakened in each frame. The value of *start_cycle* can range from 0 to the total number of minor cycles per frame minus 1. The default is 0. The total number of minor cycles per frame is defined by executing the **cs** command (see page 5-13).

-b *policy* the POSIX scheduling policy for the specified program: **F** to select the first-in-first-out (FIFO) scheduling policy, **R** to select the round-robin scheduling policy or **O** to select the time-sharing scheduling policy.

If you do not specify the **-b** *policy* argument, the default policy is the time-sharing scheduling policy.

Note: It is recommended that you specify this argument.

-p *priority* the scheduling priority for the specified process. The default value is 0.

The range of priority values that you can enter is governed by the value of the policy argument. Determine the allowable range of priorities associated with each policy (**F**, **R**, or **O**) by invoking the **run (1)** command from the shell and not specifying any options or arguments. Higher numerical values correspond to *more* favorable scheduling priorities.

-o *halt_flag* indicates whether or not the scheduler should be stopped in the event that the specified process causes a frame overrun. The value of *halt_flag* must be either **halt** or **nohalt**. The default is **nohalt**.

-L *soft_limit* the soft overrun limit for the process. The default is 0. If you reschedule a process that already has a non-zero soft overrun limit set and you do not specify a soft overrun limit, the process' soft overrun limit is set to 0.

Display

When the specified process is successfully rescheduled, the following is displayed:

CPU	fpid	Priority	Frequency	Start	HaltOnOR	SoftLimit	Process Name
1	198	53	4	1	N	2	/usr_1/guest/task03

CPU

The identifier for the CPU on which the specified process is scheduled.

fpid

The unique frequency-based scheduler process's identifier for the specified process. This identifier is displayed by the real-time command processor when you schedule a program on the scheduler using the **sp** command (page 5-32).

Priority

The scheduling priority of the specified process.

Frequency

The frequency with which the specified process is scheduled to be awakened in each major frame.

Start

The first minor cycle in which the specified process is scheduled to be awakened in each major frame.

HaltOnOR

Indicates whether or not the “halt on overrun” flag has been set for the specified process; Y (yes) or N (no).

SoftLimit

The soft overrun limit of the process.

Process Name

The full path name of the process that has been scheduled on the selected frequency-based scheduler.

sp – Schedule a Process on a Frequency-Based Scheduler

The **sp** command creates a process and schedules it on a frequency-based scheduler.

The effective user ID of the calling process must match the effective user ID of the target process (the process for which the scheduling policy and priority are being set) for the following conditions:

- if you execute this command and you wish to change a process' scheduling policy to the first-in-first-out (FIFO) or the round-robin policy
- if you wish to change the priority of a process scheduled under the FIFO or the round-robin policy
- if you wish to raise the priority of a process scheduled under the time-sharing policy above a per-process limit.

If you wish to modify the CPU bias of the process when you invoke this command, the real or effective user ID of the calling process must match the real or saved user ID of the process for which the CPU assignment is being changed.

When you execute this command, the real-time command processor returns a unique frequency-based scheduler process identifier. You can subsequently use this identifier to specify the process when you are executing other commands.

Synopsis

```
sp -s scheduler -n proc_name [-c cpu_bias] [-f frequency] [-m start_cycle] [-b policy] \  
[-p priority] [-v parameter] [-o halt_flag] [-L soft_limit]
```

Arguments

- | | |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| -s scheduler | the numeric key associated with the frequency-based scheduler on which the process is to be scheduled. The scheduler must previously have been configured. See the section “cs – Configure a Frequency-Based Scheduler” on page 5-13 for information on configuring a frequency-based scheduler. |
| -n proc_name | a standard path name identifying the program that you wish to schedule. This can be a full or relative path name of up to 1024 characters. |
| -c cpu_bias | the CPU bias for the specified program. The CPU bias determines the processor or processors on which the program can be scheduled. |

The value of *cpu_bias* may be a single CPU ID or a list of CPU IDs. CPU IDs range from 0 to 7, where the number 0 represents the first logical CPU, 1 represents the second, and so on.

A list of CPU IDs may specify a sequence or a range of numbers—for example, **-c 1,3–5,7**. Note that you must use commas to separate items in the list. Specify the entire range of CPU IDs (0–7) by entering an asterisk (*).

If you do not specify this argument, the default processor is the

- CPU on which the real-time command processor is currently executing.
- f** *frequency* the frequency with which the specified process is to be awakened in each major frame.
- A frequency of 1 indicates that the specified process is to be awakened every minor cycle; a frequency of 2 indicates that it is to be awakened once every two minor cycles, and so on.
- Specify the number of minor cycles representing the frequency with which you wish the process to be awakened. The value of *frequency* can range from 1 to the number of minor cycles that compose a frame on the scheduler. The default value is 1. The total number of minor cycles per frame is defined by executing the **cs** command (see page 5-13).
- m** *start_cycle* the first minor cycle in which the specified process is scheduled to be awakened in each frame. The value of *start_cycle* can range from 0 to the total number of minor cycles per frame minus one. The default value is 0. The total number of minor cycles per frame is defined by executing the **cs** command (see page 5-13).
- b** *policy* the POSIX scheduling policy for the specified process: **F** to select the first-in-first-out (FIFO) scheduling policy, **R** to select the round-robin scheduling policy or **O** to select the time-sharing scheduling policy.
- If you do not specify the **-b** *policy* argument, the default policy is the time-sharing scheduling policy.
- Note: It is recommended that you specify this argument.
- p** *priority* the scheduling priority for the specified process. The default value is 0.
- The range of priority values is governed by the value of the policy argument. Determine the allowable range of priorities associated with each policy (**F**, **R**, or **O**) by invoking the **run (1)** command from the shell and not specifying any options or arguments.
- Higher numerical values correspond to more favorable scheduling priorities.
- v** *parameter* an integer value to be passed to a process that is scheduled on a frequency-based scheduler.
- The value of *parameter* must be a 32-bit decimal number.
- o** *halt_flag* indicates whether or not the scheduler should be stopped in the event that the specified program causes a frame overrun.
- The value of *halt_flag* must be either **halt** or **nohalt**. The default is **nohalt**.
- L** *soft_limit* the soft overrun limit for the process. The default is 0.

Display

The following information is displayed when the specified process is successfully scheduled on the frequency-based scheduler:

```
fpid 199 assigned to process task02
```

`fpid`

The unique frequency-based scheduler process identifier assigned to the specified process.

`process`

The full path name of the process that has been scheduled on the selected frequency-based scheduler.

vp – View Processes on a Frequency-Based Scheduler

The **vp** command displays information about a particular frequency-based scheduled process or all frequency-based scheduled processes on one or more processors on a selected frequency-based scheduler.

You can display information about all frequency-based scheduled processes on a specified processor or all processors by specifying the scheduler and the CPU(s) on which the processes are scheduled.

If you wish to display information for a particular frequency-based scheduled process, you can identify the process by specifying:

- the name of the process and the CPU on which it is scheduled
- the process' frequency-based scheduler process identifier
- the name of the process, the CPU on which it is scheduled, and its frequency-based scheduler process identifier

NOTE

The only method that can be used to identify a process that has been scheduled multiple times on the same CPU is to specify its frequency-based scheduler identifier.

Information displayed for each process includes:

- the CPU on which the process is executing
- the frequency-based scheduler process identifier
- the process' scheduling priority
- the frequency (the number of minor cycles indicating the frequency with which the process is awakened in each major frame)
- the starting base cycle (the first minor cycle in which the process is scheduled to be awakened in each major frame)
- the value of the "halt on overrun" flag
- the soft overrun limit of the process
- the path name of the process.

Synopsis

```
vp -s scheduler [-n proc_name] [-i fpid] [-c cpu_bias]
```

Arguments

-s *scheduler* the numeric key associated with the frequency-based scheduler for which scheduling information is to be displayed. The scheduler must previously have been configured. See the "cs – Configure a Frequency-Based Scheduler" section on page 5-13 for information on configuring a frequency-based scheduler.

-n *proc_name* a standard path name identifying a particular process for which scheduling information is to be displayed. This can be a full or relative path name of up to 1024 characters.

-i *fpid* the frequency-based scheduler process identifier for a particular process for which scheduling information is to be displayed. This value is displayed when you successfully schedule a program by executing the **sp** command (page 5-32).

The default value for *fpid* is -1. If you accept the default value, you must identify the process by name and CPU.

-c *cpu_bias* the processor(s) for which scheduling information is to be displayed.

The value of *cpu_bias* may be a single CPU ID or a list of CPU IDs. CPU IDs range from 0 to 7, where the number 0 represents the first logical CPU, 1 represents the second, and so on.

A list of CPU IDs can specify a sequence or a range of numbers—for example, **-c 1,3-5,7**. Note that you must use commas to separate items in the list. Specify the entire range of CPU IDs (0-7) by entering an asterisk (*).

The default processor is the CPU on which the real-time command processor is currently executing.

Display

The following information is displayed when the command is successfully executed:

CPU	fpid	Priority	Frequency	Start	HaltOnOR	SoftLimit	Process Name
1	199	55	1	0	Y	0	/usr_1/guest/task02
1	198	53	4	1	N	3	/usr_1/guest/task03
2	197	53	2	0	Y	1	/usr_1/guest/task04

CPU

The identifiers for the CPUs on which the respective processes are scheduled.

fpid

The frequency-based scheduler process identifiers for the respective processes. Identifiers are displayed by the real-time command processor when you schedule a program on the scheduler using the **sp** command (page 5-32).

Priority

The scheduling priorities of the respective processes.

Frequency

The frequency with which the respective processes are scheduled to be awakened in each major frame.

Start

The first minor cycle in which the respective processes are scheduled to be awakened in each major frame.

HaltOnOR

Indicates whether or not the “halt on overrun” flag has been set for the respective processes; Y (yes) or N (no).

SoftLimit

The soft overrun limit of the process.

Process Name

The full path names of the processes that have been scheduled on the selected frequency-based scheduler.

cpm – Clear Performance Monitor Values

The **cpm** command clears performance monitor values for a particular process or all processes on one or more processors for a selected frequency-based scheduler.

You can clear values for all frequency-based scheduled processes on a specified processor or all processors by specifying the scheduler and the CPU(s) on which the processes are scheduled.

If you wish to clear performance monitor values for a particular frequency-based scheduled process, you can identify the process by specifying:

- the name of the process and the CPU on which it is scheduled
- the process' frequency-based scheduler process identifier
- the name of the process, the CPU on which it is scheduled, and its frequency-based scheduler process identifier

Synopsis

```
cpm -s scheduler [-i fpid] [-n proc_name] [-c cpu_bias]
```

Arguments

- | | |
|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| -s scheduler | the numeric key associated with the frequency-based scheduler on which the process or processes are scheduled. The scheduler must previously have been configured. See the section “cs – Configure a Frequency-Based Scheduler” section on page 5-13 for information on configuring a frequency-based scheduler. |
| -i fpid | the unique frequency-based scheduler process identifier for a particular process for which values are to be cleared. This value is displayed when you execute the sp command (page 5-32).

The default value for <i>fpid</i> is -1. If you accept the default value, you must identify the process by name and CPU. |
| -n proc_name | a standard path name identifying a particular process for which values are to be cleared. This can be a full or relative path name of up to 1024 characters. |
| -c cpu_bias | the processor(s) for which performance monitor values are to be cleared. |

The value of *cpu_bias* may be a single CPU ID or a list of CPU IDs. CPU IDs range from 0 to 7, where the number 0 represents the first logical CPU, 1 represents the second, and so on.

A list of CPU IDs may specify a sequence or a range of numbers—for example, **-c 1,3–5,7**. Note that you must use commas to separate items in the list. Specify the entire range of CPU IDs (**0–7**) by entering an asterisk (*).

The default processor is the CPU on which the real-time command processor is currently executing.

Display

The following message is displayed when the performance monitor values are successfully cleared:

```
Performance monitor values cleared
```

NOTE

This command clears the soft overrun count for all processes specified by the user.

pm – Start/Stop Performance Monitoring

The **pm** command starts or stops performance monitoring for a particular process or all processes on one or more processors for a selected frequency-based scheduler.

You can start or stop performance monitoring for all frequency-based scheduled processes on a specified processor or all processors by specifying the scheduler and the CPU(s) on which the processes are scheduled.

If you wish to start or stop performance monitoring for a particular frequency-based scheduled process, you can identify the process by specifying:

- the name of the process and the CPU on which it is scheduled
- the process' frequency-based scheduler process identifier
- the name of the process, the CPU on which it is scheduled, and its frequency-based scheduler process identifier

Synopsis

```
pm -s scheduler [-i fpid] [-n proc_name] [-c cpu_bias] [-P pm_flag]
```

Arguments

-s scheduler the numeric key associated with the frequency-based scheduler on which the process or processes are scheduled. The scheduler must previously have been configured. See the section “cs – Configure a Frequency-Based Scheduler” on page 5-13 for information on configuring a frequency-based scheduler.

-i fpid the unique frequency-based scheduler process identifier for a particular process for which performance monitoring is to be started or stopped. This value is displayed when you execute the **sp** command (page 5-32).

The default value for *fpid* is -1. If you accept the default value, you must identify the process by name and CPU.

-n proc_name a standard path name identifying a particular process for which performance monitoring is to be started or stopped. This can be a full or relative path name of up to 1024 characters.

-c cpu_bias the processor(s) for which performance monitoring is to be started or stopped.

The value of *cpu_bias* may be a single CPU ID or a list of CPU IDs. CPU IDs range from 0 to 7, where the number 0 represents the first logical CPU, 1 represents the second, and so on.

A list of CPU IDs may specify a sequence or a range of numbers—for example, **-c 1,3-5,7**. Note that you must use commas to separate items in the list. Specify the entire range of CPU IDs (**0-7**) by entering an asterisk (*).

The default processor is the CPU on which the real-time command processor is currently executing.

-P *pm_flag* indicates whether performance monitoring is to be started or stopped. The value of *pm_flag* must be either **ON** or **OFF**. The default is **OFF**.

Display

The following message is displayed when performance monitoring is successfully started:

```
Performance monitoring enabled
```

The following message is displayed when performance monitoring is successfully stopped:

```
Performance monitoring disabled
```

vcm – View/Modify Performance Monitor Timing Mode

The **vcm** command views or modifies the performance monitor timing mode. The timing mode can be set to include or exclude time spent servicing interrupts from the performance monitor timing values.

When interrupt time is included, a process' user and system times will total the elapsed time which accrues when the process is the currently running process, including all time spent servicing interrupts. The time spent servicing interrupts is added to the process' system time.

When excluding interrupt time, a process' user and system times will total the time which accrues when the process is the currently running process, excluding all time spent servicing interrupts.

CAUTION

The timing mode is the mode in which the High-Resolution Process Timing Facility operates system-wide. It affects all processes running on all CPUs.

Synopsis

vcm [-t *pm_mode*]

Arguments

-t *pm_mode* specifies whether interrupt time is to be included in or excluded from performance monitor timing values. The value of *pm_mode* must be either **in** (included) or **ex** (excluded).

Display

One of the following messages is displayed when you successfully execute the **vcm** command to view the performance monitor timing mode:

```
PM timing mode includes interrupt times.
```

or

```
PM timing mode excludes interrupt times.
```

vpm – View Performance Monitor Values

The **vpm** command displays performance monitor values for one process or all processes on one or more processors for a selected frequency-based scheduler.

Display values for all frequency-based scheduled processes on a processor or all processors by specifying the scheduler and the CPU(s) on which the processes are scheduled.

If you wish to display performance monitor values for a particular frequency-based scheduled process, you can identify the process by specifying:

- the name of the process and the CPU on which it is scheduled
- the process' frequency-based scheduler process identifier
- the name of the process, the CPU on which it is scheduled, and its frequency-based scheduler process identifier

Synopsis

```
vpm -s scheduler [-i fpid] [-n proc_name] [-c cpu_bias] [-x pm_output]
```

Arguments

-s scheduler the numeric key associated with the frequency-based scheduler on which the process or processes are scheduled. The scheduler must previously have been configured. See the section “cs – Configure a Frequency-Based Scheduler” on page 5-13 for information on configuring a frequency-based scheduler.

-i fpid the unique frequency-based scheduler process identifier for a particular process for which performance monitor values are to be displayed. This value is displayed when you execute the **sp** command (see page 5-32).

The default value for *fpid* is -1. If you accept the default value, you must identify the process by name and CPU.

-n proc_name a standard path name identifying a particular process for which performance monitor values are to be displayed. This can be a full or relative path name of up to 1024 characters.

-c cpu_bias the processor(s) for which performance monitor values are to be displayed.

The value of *cpu_bias* may be a single CPU ID or a list of CPU IDs. CPU IDs range from 0 to 7, where the number 0 represents the first logical CPU, 1 represents the second, and so on.

A list of CPU IDs may specify a sequence or a range of numbers—for example, **-c 1,3–5,7**. Note that you must use commas to separate items in the list. Specify the entire range of CPU IDs (**0–7**) by entering an asterisk (*).

The default processor is the CPU on which the real-time command processor is currently executing.

-x pm_output

indicates the type of values to display. Four types of values can be specified: average, minimum, maximum, or all.

Specify **av** (average) to display: the number of iterations, or cycles; the last time; the total time; the average time; and the number of overruns.

Specify **mi** (minimum) to display: the number of iterations; the minimum cycle time and the number of the minor cycle and the major frame in which it has occurred; and the minimum frame time and the number of the major frame in which it has occurred.

Specify **ma** (maximum) to display: the number of iterations; the maximum cycle time and the number of the minor cycle and major frame in which it has occurred; and the maximum frame time and the number of the major frame in which it has occurred.

Specify **al** (all) to display average, minimum, and maximum values.

Times are reported in microseconds. The default value is **av** (average).

Display

The following information is displayed when you select **average** values and the command is successfully executed:

fpid	Iterations	TimeLast (us)	TotalTime (us)	Average (us)	OverRuns	
					Hard	Soft
199	480	1023	499200	1040	0	0
198	120	2013	240960	2008	1	2
197	240	1521	378960	1579	0	3

fpid

The unique frequency-based scheduler process identifiers for the processes. This identifier is displayed by the real-time command processor when you schedule a program on the scheduler using the **sp** command (see page 5-32).

Iterations

The number of times that the processes have been awakened by the frequency-based scheduler since the last time that performance monitor values have been cleared and performance monitoring has been enabled.

TimeLast (us)

The amount of time in microseconds that the processes have spent running from the last time that they were awakened by the frequency-based scheduler until they called **fbwait**.

TotalTime (us)

The cumulative times in microseconds that the processes have spent running in all cycles, or iterations.

Average (us)

The average amount of time in microseconds that the respective processes have spent running in all cycles, or iterations. These values are obtained by dividing the values reported in the Total Time column by the values reported in the Iterations column.

Overruns

Hard The number of times that the respective processes have caused a catastrophic frame overrun.

Soft The number of times that the respective processes have caused a non-catastrophic frame overrun.

The following information is displayed when you select **minimum** values and the command is successfully executed:

fpid	Iterations	Minimum Cycle		Minimum Frame	
		time(us)	Frame/Cycle	time(us)	Frame
199	30	1002	6/7	8013	17
198	30	1943	2/1	3995	22
197	30	1312	1/2	5314	11

fpid

The unique frequency-based scheduler process identifiers for the respective processes. This identifier is displayed by the real-time command processor when you schedule a program on the scheduler using the **sp** command (see page 5-32).

Iterations

The number of times that the respective processes have been awakened by the frequency-based scheduler since the last time that performance monitor values have been cleared and performance monitoring has been enabled.

Minimum Cycle
time(us)

The least amount of time in microseconds that the processes spent running in a cycle, or iteration.

Minimum Cycle
Frame/Cycle

The number of the major frame and the minor cycle in which the minimum cycle time has occurred.

Minimum Frame
time(us)

The least amount of time in microseconds that the processes spent running during a major frame.

Minimum Frame
Frame

The number of the major frame in which the minimum frame time has occurred.

The following information is displayed when you select **maximum** values and the command is successfully executed:

fpid	Iterations	Maximum Cycle		Maximum Frame	
		time(us)	Frame/Cycle	time(us)	Frame
199	30	1303	2/4	9502	12
198	30	2201	7/5	4391	17
197	30	1917	9/6	7431	24

fpid

The unique frequency-based scheduler process identifiers for the respective processes. This identifier is displayed by the real-time command processor when you schedule a program on the scheduler using the **sp** command (see page 5-32).

Iterations

The number of times that the respective processes have been awakened by the frequency-based scheduler since the last time that performance monitor values were cleared and performance monitoring was enabled.

Maximum Cycle
time(us)

The greatest amount of time in microseconds that the processes spent running in a cycle, or iteration.

Maximum Cycle
Frame/Cycle

The number of the major frame and the minor cycle in which the maximum cycle time has occurred.

Maximum Frame
time (us)

The greatest amount of time in microseconds that the processes have spent running during a major frame.

Maximum Frame
Frame

The number of the major frame in which the maximum frame time has occurred.

ex – Exit Real-Time Command Processor

The **ex** command is used only when you are using the real-time command processor in interactive mode. It exits the command processor and returns to the shell.

Synopsis

ex

The system command prompt is displayed when the command is successfully executed.

he – Display Help Information

The **he** command displays the following types of help information for the real-time command processor:

- a list and brief descriptions of all commands
- a description and format of a particular command
- a list and description of all command arguments

Synopsis

he [*command* | **option** | **op2**]

Arguments

<i>command</i>	the command for which you wish to obtain information
option	the first screen of command arguments is to be displayed
op2	the second screen of command arguments is to be displayed

Display

If you specify the **he** command without an argument, a list of all commands with a brief description is displayed. See Screen 5-1 on page 5-5.

If you specify the **he** command with the *option* argument, the first screen of **rtcp** command arguments is displayed. See Screen 5-2 on page 5-6.

If you specify the **he** command with the *op2* argument, the second screen of **rtcp** command arguments is displayed. See Screen 5-3 on page 5-7.

If you specify the **he** command with the *command* argument, a description of the specified command and the format for entering the command is displayed. For example, **he chs** displays the following:

```
Change FBS permissions
```

```
rtcp chs -s scheduler -I permissions [-G gid] [-U uid]
```

The C Library Interface

Overview	6-1
Compiling and Linking Programs	6-1
Frequency-Based Scheduler Routines	6-2
Routine Summary	6-2
C Library Call Sequence	6-4
Using Frequency-Based Scheduler Routines	6-5
fbsaccess – Change Permissions for a Frequency-Based Scheduler	6-5
fbsattach – Attach Timing Source to a Frequency-Based Scheduler	6-6
fbsavail – Query if the Frequency-Based Scheduler is Configured	6-7
fbsconfigure – Configure a Frequency-Based Scheduler	6-8
fbscycle – Return Minor Cycle/Major Frame Count	6-11
fbsdetach – Detach Timing Source from a Frequency-Based Scheduler	6-12
fbsdir – Return a List of Scheduler Keys	6-13
fbsgetpid – Return Process ID for a Scheduled Process	6-14
fbsgetrtc – Obtain Current Values for a Real-Time Clock	6-15
fbsid – Return the Frequency-Based Scheduler Identifier for a Key	6-16
fbsinfo – Return Information for a Frequency-Based Scheduler	6-17
fbsintrpt – Start/Stop/Resume Scheduling	6-19
fbsremove – Remove a Frequency-Based Scheduler	6-20
fbsresume – Resume Scheduling on a Frequency-Based Scheduler	6-21
fbsrunrtc – Start/Stop a Real-Time Clock	6-22
fbsschedself – Add a Calling Process to a Frequency-Based Scheduler	6-23
fbssetrtc – Set a Real-Time Clock	6-25
fbstrig – Make a Sleeping Frequency-Based Scheduler Process Runnable	6-26
fbswait – Wait on a Frequency-Based Scheduler	6-27
nametopid, namepid – Return the Process ID for a Specified Process Name	6-28
pgmremove – Remove a Process from a Frequency-Based Scheduler	6-29
pgmtrigger – Trigger a Process on a Frequency-Based Scheduler	6-31
sched_fbsqry – Query Processes on a Frequency-Based Scheduler	6-32
sched_pgm_set_soft_overrun_limit – Set Soft Overrun Limit	6-34
sched_pgm_soft_overrun_query – Query Soft Overrun Processing	6-36
sched_pgmadd – Schedule a Process on a Frequency-Based Scheduler	6-38
sched_pgmqry – Query a Process	6-41
sched_pgmresched – Reschedule a Process	6-44
Performance Monitor Routines	6-47
Routine Summary	6-47
C Library Call Sequence	6-47
Using Performance Monitor Routines	6-48
pmclrpgm – Clear Values for a Process	6-49
pmclrtable – Clear Values for Processor(s)	6-51
pmmonitor – Start/Stop Performance Monitoring on Processor(s)	6-52
pmprogram – Start/Stop Performance Monitoring on a Process	6-53
pmqrycpu – Query Values for Selected Processor(s)	6-55
pmqrylist – Query Values for a List of Processes	6-58
pmqrypgm – Query Values for a Selected Process	6-60
pmqrytimer – Query Performance Monitor Mode	6-63
pmselect – Select Performance Monitor Mode	6-64

The C Library Interface

This chapter describes the C library interface to the Frequency-Based Scheduler and the Performance Monitor. Library information, call sequences for using the scheduler and the Performance Monitor, and details for each of the library routines are provided.

Overview

Access to the functions associated with the Frequency-Based Scheduler and the Performance Monitor is provided through libraries of routines that can be called from application programs written in C.

The following information is provided in this chapter for each routine:

- a description of the routine
- the synopsis of the routine
- detailed descriptions of each parameter
- the return value

Information about the individual routines, including an illustration of the sequence in which you would call the routines during frequency-based scheduling, begins on page 6-2. The same information is provided for performance monitoring beginning on page 6-47.

An example program that illustrates use of the C library interface to the Frequency-Based Scheduler and the Performance Monitor is provided in Appendix C.

Compiling and Linking Programs

When statically linking a C program, the following libraries are required:

```
/usr/lib/libccur_fbsched.a  
/usr/lib/libccur_rt.a
```

When dynamically linking a C program, the following libraries are used:

```
/usr/lib/libccur_fbsched.so  
/usr/lib/libccur_rt.so
```

To compile and link a C program, the command line instruction is as follows:

```
gcc [options...] source_file.c -lccur_fbsched -lccur_rt
```

For additional information on compiling and linking procedures, refer to the `gcc(1)` and `ld(1)` man pages.

Frequency-Based Scheduler Routines

Frequency-Based Scheduler routines provide access to the key features of the scheduler. They enable you to perform such basic operations as:

- configuring a scheduler
- scheduling programs on a scheduler
- setting up and connecting a timing source to a scheduler
- starting, stopping and resuming scheduling on a scheduler
- getting information about scheduled processes
- rescheduling and removing scheduled processes
- disconnecting a timing source
- removing a scheduler

Routine Summary

Frequency-Based Scheduler routines are summarized in Table 6-1. Complete information about each routine is provided under the section “Using Frequency-Based Scheduler Routines.”

Table 6-1. Frequency-Based Scheduler C Library Routines

Routine	Page	Description
fbsaccess	6-5	Change permissions for a frequency-based scheduler
fbsattach	6-6	Attach timing source to a frequency-based scheduler
fbsavail	6-7	Query if FBS is configured in the system
fbsconfigure	6-8	Configure a frequency-based scheduler
fbscycle	6-11	Return minor cycle/major frame count
fbsdetach	6-12	Detach timing source from a frequency-based scheduler
fbsdir	6-13	Return list of scheduler keys
fbsgetpid	6-14	Return process ID of scheduled process
fbsgetrtc	6-15	Get real-time clock values
fbsid	6-16	Return the frequency-based scheduler identifier for a key
fbsinfo	6-17	Return information for a frequency-based scheduler
fbsintrpt	6-19	Start/stop/resume scheduling
fbsremove	6-20	Remove a frequency-based scheduler

(continued on next page)

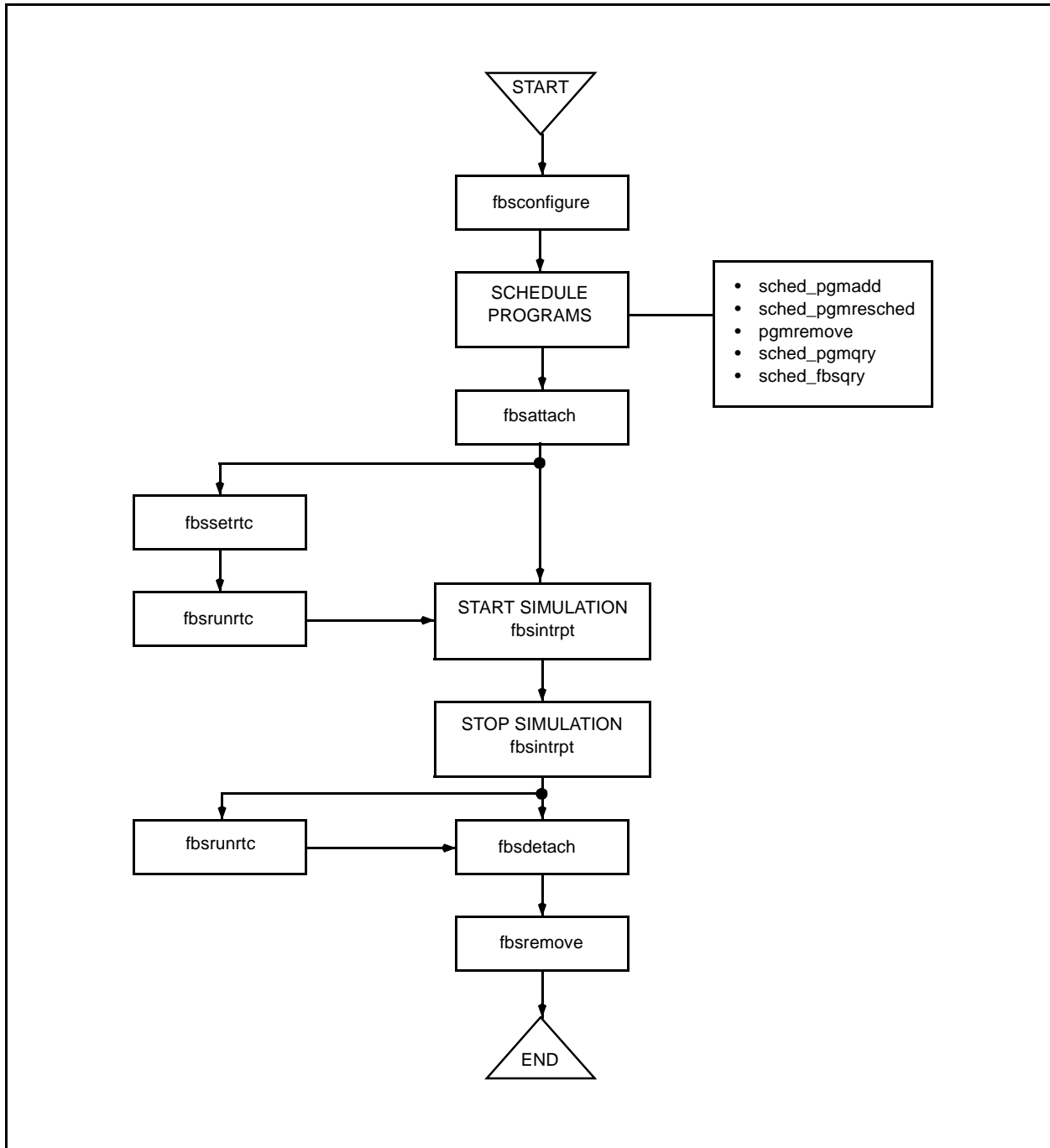
Table 6-1. Frequency-Based Scheduler C Library Routines (Continued)

Routine	Page	Description
fbsresume	6-21	Resume scheduling on a frequency-based scheduler
fbsrunrtc	6-22	Start/stop real-time clock
fbsschedself	6-23	Add a calling process to a frequency-based scheduler
fbssetrtc	6-25	Set real-time clock
fbstrig	6-26	Make a sleeping frequency-based scheduled process runnable
fbswait	6-27	Wait on a frequency-based scheduler
namepid	6-28	Return the process ID for a specified process name
nametopid	6-28	Return the process ID for a specified process name on a specified scheduler
pgmremove	6-29	Remove a process from a frequency-based scheduler
pgmtrigger	6-31	Trigger a process on a frequency-based scheduler
sched_fbsqry	6-32	Query processes on a frequency-based scheduler
sched_pgm_set_soft_ouerrun_limit	6-34	Set soft overrun limit
sched_pgm_soft_ouerrun_query	6-36	Query soft overrun processing
sched_pgmadd	6-38	Schedule a process on a frequency-based scheduler
sched_pgmqry	6-41	Query a process
sched_pgmresched	6-44	Reschedule a process

C Library Call Sequence

The approximate order in which you might call the routines from an application program is illustrated in Figure 6-1.

Figure 6-1. C Library Call Sequence: Frequency-Based Scheduler



Using Frequency-Based Scheduler Routines

In the sections that follow, the Frequency-Based Scheduler routines contained in the `libccur_fbsched` library are presented in alphabetical order.

fbsaccess – Change Permissions for a Frequency-Based Scheduler

This routine changes the permissions assigned for a selected frequency-based scheduler. It is important to note that the permissions can be changed only by a process that has an effective user ID that is equal to that of the owner/creator of the frequency-based scheduler.

Synopsis

```
#include <fbsched.h>
```

```
int fbsaccess(fbs_id, uid, gid, permissions)
int fbs_id, uid, gid, permissions;
```

Parameters

<i>fbs_id</i>	a unique positive integer value representing the identifier for a frequency-based scheduler. This value can be obtained using fbsid(3) (page 6-16) or fbsconfigure(3) (page 6-8). Enter MY_FBS to reference the frequency-based scheduler on which the calling process is scheduled without knowing the identifier.
<i>uid</i>	an integer value representing the effective user ID of the specified frequency-based scheduler.
<i>gid</i>	an integer value representing the effective group ID of the specified frequency-based scheduler.
<i>permissions</i>	a bit pattern used to set the permissions associated with the specified frequency-based scheduler. Bit patterns and corresponding permissions are presented in Table 6-2.

Table 6-2. Frequency-Based Scheduler Permissions

Bit Pattern	Permissions
400	Read by user
200	Alter by user
060	Read, alter by group
006	Read, alter by others

Return Value

A return value of 0 indicates that the call has been successful. A return value of -1 indicates that an error has occurred; `errno` is set to indicate the error. Refer to the **fbsaccess(3)** man page for a listing of the types of errors that may occur.

fbsattach – Attach Timing Source to a Frequency-Based Scheduler

This routine attaches a timing source to a frequency-based scheduler or specifies end-of-cycle scheduling. The timing source can be a real-time clock or an edge-triggered interrupt device.

Synopsis

```
#include <fbsched.h>

int fbsattach(fbs_id, devname)
int fbs_id;
char *devname;
```

Parameters

fbs_id a unique positive integer value representing the identifier for a frequency-based scheduler. This value can be obtained using **fbsid(3)** (page 6-16) or **fbsconfigure(3)** (page 6-8).

Enter MY_FBS to reference the frequency-based scheduler on which the calling process is scheduled without knowing the identifier.

devname a null string or the path name of the device that is to be used as the timing source for the specified scheduler.

If *devname* contains a null string, end-of-cycle scheduling is specified; that is, execution of the processes in the next minor cycle will occur when the last process scheduled to execute in the current minor cycle finishes its execution for that cycle.

If *devname* contains a path name, it may refer to a real-time clock or an edge-triggered interrupt. See Chapter 3 for information about timing source device files.

Return Value

A return value of 0 indicates that the call has been successful. A return value of -1 indicates that an error has occurred; `errno` is set to indicate the error. Refer to the **fbsattach(3)** man page for a listing of the types of errors that may occur.

fbsavail – Query if the Frequency-Based Scheduler is Configured

This routine queries the currently running kernel to determine if the Frequency-Based Scheduler is configured.

Synopsis

```
#include <fbsched.h>
```

```
int fbsavail()
```

Return Value

A return value of 1 indicates that FBS is configured. A return value of 0 indicates that FBS is not configured.

fbsconfigure – Configure a Frequency-Based Scheduler

This routine configures a frequency-based scheduler or obtains configuration details for a scheduler that has already been configured.

Indicate the function to be performed using the value assigned to the *cycles* field of the `fbsconfig_ds` structure as follows:

- If the value specified in *cycles* is NOT equal to 0, and if the number of existing frequency-based schedulers is less than the system imposed limit (default = 10) a frequency-based scheduler is created and an identifier is returned in the *fbs_id* field.

The value assigned to the *reset* field of the `fbsconfig_ds` structure can be used to further qualify the action to be taken when *cycles* is not equal to 0 and the scheduler already exists.

- If the value of *cycles* is equal to 0, the current configuration for the frequency-based scheduler is returned in the `fbsconfig_ds` structure. The user-supplied *key* field can be used to identify a frequency-based scheduler.

Synopsis

```
#include <fbsched.h>

int fbsconfigure(fbs_buf)
struct fbsconfig_ds {
    int key;
    int cycles;
    int progs;
    int max;
    int reset;
    int configflg;
    int fbs_id;
} *fbs_buf;
```

Parameters

To create a frequency-based scheduler, you must specify the following parameters as described.

<i>fbs_buf</i>	An <code>fbsconfig_ds</code> structure that contains the information with which you wish to configure a frequency-based scheduler. The type of information specified in each component is presented below.
<i>key</i>	an integer value that may be used later to identify the frequency-based scheduler to be created. Zero is not a valid key value but can be assigned to indicate that a key is not to be used. This is an optional identifier since the <i>fbs_id</i> value returned here can always be used to reference a frequency-based scheduler.
<i>cycles</i>	an integer value indicating the number of minor cycles that compose a frame on the specified scheduler.

<i>progs</i>	an integer value indicating the maximum number of programs that can be scheduled to execute during one minor cycle.
<i>max</i>	an integer value indicating the maximum number of programs that can be scheduled on the specified scheduler at one time. This value must be less than or equal to the product that is obtained by multiplying the values specified for the <i>cycles</i> and <i>progs</i> parameters.
<i>reset</i>	an integer value indicating whether or not processes currently scheduled on the specified scheduler are to be killed before the scheduler is reconfigured. Acceptable values and corresponding results are as follows: <ul style="list-style-type: none"> <0 Kill and remove all processes currently scheduled on the specified scheduler 0 Ignore all processes currently scheduled on the specified scheduler >0 Remove all processes currently scheduled on the specified scheduler
<i>configflg</i>	an integer value indicating the permissions to be assigned to the specified scheduler.
<i>fbs_id</i>	a unique positive integer value that is returned by fbsconfigure and represents the identifier for the specified frequency-based scheduler. It is important to note that this identifier is required by most of the library routines for the Frequency-Based Scheduler and the Performance Monitor.

To obtain information for an existing frequency-based scheduler, you must specify the following parameters as described.

<i>fbs_buf</i>	An <code>fbsconfig_ds</code> structure to which information is returned for an existing frequency-based scheduler. The types of information specified or returned in each component is presented below.
<i>key</i>	the user-supplied value specified when the scheduler was created. This value can be used to identify the frequency-based scheduler for which configuration information is to be returned.
<i>cycles</i>	the integer value 0 indicating that current configuration information for the specified scheduler is to be returned. An integer value indicating the number of minor cycles that compose a frame on the specified scheduler is returned to this component.
<i>progs</i>	the maximum number of programs that can be scheduled to run during one minor cycle on the specified scheduler.
<i>max</i>	the maximum number of programs that can be scheduled on the specified scheduler at one time.

<i>configfg</i>	the permissions assigned to the specified scheduler.
<i>fbs_id</i>	a unique positive integer value representing the identifier for the specified frequency-based scheduler.

Return Value

A return value of 0 indicates that the call has been successful. A return value of -1 indicates that an error has occurred; `errno` is set to indicate the error. Refer to the **fbsconfigure(3)** man page for a listing of the types of errors that may occur.

fbscopy – Return Minor Cycle/Major Frame Count

This routine obtains the current minor cycle and major frame count values for a frequency-based scheduler. These values enable you to determine the progress of a simulation.

Synopsis

```
#include <fbsched.h>

int fbscopy(fbs_id, cycle_buf)
int fbs_id;
struct fbscopy_ds {
    int ccycle;
    int cframe;
} *cycle_buf;
```

Parameters

fbs_id a unique positive integer value representing the identifier for a frequency-based scheduler. This value can be obtained using **fbsid(3)** (page 6-16) or **fbsconfigure(3)** (page 6-8).

Enter MY_FBS to reference the frequency-based scheduler on which the calling process is scheduled without knowing the identifier.

cycle_buf refers to an fbscopy_ds structure to which **fbscopy** will return integer values indicating the current minor cycle and major frame for the specified scheduler. The *ccycle* component will contain the number of the cycle. The *cframe* component will contain the number of the frame.

Return Value

A return value of 0 indicates that the call has been successful. A return value of -1 indicates that an error has occurred; `errno` is set to indicate the error. Refer to the **fbscopy(3)** man page for a listing of the types of errors that may occur.

fbsdetach – Detach Timing Source from a Frequency-Based Scheduler

This routine detaches the currently attached timing source from a frequency-based scheduler or disables end-of-cycle scheduling. If the timing source is a real-time clock, it is recommended that you stop the clock prior to invoking this routine. You can do so by making a call to **fbsrunrtc** (see page 6-22 for an explanation of this routine).

Synopsis

```
#include <fbsched.h>
```

```
int fbsdetach(fbs_id)  
int fbs_id;
```

Parameters

fbs_id a unique positive integer value representing the identifier for a frequency-based scheduler. This value can be obtained using **fbsid(3)** (page 6-16) or **fbsconfigure(3)** (page 6-8).

Enter MY_FBS to reference the frequency-based scheduler on which the calling process is scheduled without knowing the identifier.

Return Value

A return value of 0 indicates that the call has been successful. A return value of -1 indicates that an error has occurred; `errno` is set to indicate the error. Refer to the **fbsdetach(3)** man page for a listing of the types of errors that may occur.

fbsdir – Return a List of Scheduler Keys

This routine returns the list of keys associated with the frequency-based schedulers currently configured in the system.

Synopsis

```
#include <fbsched.h>
```

```
fbsdir(key_t *keylist, size_t *keycnt)
```

Parameters

keylist a pointer to a memory reservation to which the keys of frequency-based schedulers configured on the system is returned. The area must large enough to store the number of `key_t` type elements specified by *keycnt*.

keycnt the number of frequency-based scheduler keys configured on the system. If this number is less than the number of schedulers configured on the system, an error is returned.

Return Value

A return value of 0 indicates that the call has been successful. A return value of -1 indicates that an error has occurred; `errno` is set to indicate the error. Refer to the **fbsdir(3)** man page for a listing of the types of errors that may occur.

fbsgetpid – Return Process ID for a Scheduled Process

This routine returns the process ID (pid) of a process currently scheduled on a frequency-based scheduler.

Synopsis

```
#include <fbsched.h>

int fbsgetpid(fbs_id, *name_ptr, fpid)
int fbs_id, fpid;
char *name_ptr;
```

Parameters

fbs_id a unique positive integer value representing the identifier for a frequency-based scheduler. This value can be obtained using **fbsid(3)** (page 6-16) or **fbsconfigure(3)** (page 6-8).

Enter MY_FBS to reference the frequency-based scheduler on which the calling process is scheduled without knowing the identifier.

name_ptr a pointer to the path name that identifies the process. This argument is currently ignored and should be NULL. Use **namepid(3)** or **nametopid(3)** to obtain a process ID by specifying the process' name.

fpid an integer value providing the unique frequency-based scheduler process identifier for the process. This value is obtained when you make a call to **sched_pgmadd** (see page 6-38 for an explanation of this routine).

Return Value

Upon successful completion, **fbsgetpid** returns the process ID. A return value of -1 indicates that an error has occurred; **errno** is set to indicate the error. Refer to the **fbsgetpid(3)** man page for a listing of the types of errors that may occur.

fbsetrtc – Obtain Current Values for a Real-Time Clock

This routine obtains the current count and resolution values for the real-time clock attached to a specified frequency-based scheduler.

Synopsis

```
#include <fbsched.h>

int fbsetrtc(fbs_id, count, resolution)
int fbs_id, *count, *resolution;
```

Parameters

fbs_id a unique positive integer value representing the identifier for a frequency-based scheduler. This value can be obtained using **fbsid(3)** (page 6-16) or **fbsconfigure(3)** (page 6-8).

Enter MY_FBS to reference the frequency-based scheduler on which the calling process is scheduled without knowing the identifier.

count the current number of clock counts per minor cycle. This value can range from 1 to 65535.

resolution an integer value indicating the current duration in microseconds of one clock count. This value is one of the following: 1, 10, 100, 1000, or 10000.

Return Value

A return value of 0 indicates that the call has been successful. A return value of -1 indicates that an error has occurred; `errno` is set to indicate the error. Refer to the **fbsetrtc(3)** man page for a listing of the types of errors that may occur.

fbsid – Return the Frequency-Based Scheduler Identifier for a Key

This routine obtains the frequency-based scheduler identifier associated with a particular user-specified key. The key must match the key that was specified when the scheduler was created via a call to **fbsconfigure(3)**.

Synopsis

```
#include <fbsched.h>
```

```
int fbsid(fbs_key)  
int fbs_key;
```

Parameters

fbs_key an integer value identifying the frequency-based scheduler. This value is the same value specified for *key* when the scheduler was created using **fbsconfigure** (see page 6-8 for an explanation of this routine).

Return Value

Upon successful completion, **fbsid** returns an integer value representing the unique frequency-based scheduler identifier associated with the key. A return value of -1 indicates that an error has occurred; `errno` is set to indicate the error. Refer to the **fbsid(3)** man page for a listing of the types of errors that may occur.

fbsinfo – Return Information for a Frequency-Based Scheduler

This routine obtains information related to a selected frequency-based scheduler that cannot be obtained by invoking other routines (for example, `sched_fbsqry`, `sched_pgmqry`).

Information includes:

- user and group IDs of the owner and the creator of the scheduler
- permissions assigned for the scheduler
- key associated with the scheduler's identifier
- total number of overruns for all processes on the scheduler
- CPUs that are active in the system
- CPUs on which performance monitoring has been enabled
- FBS enabled flag
- path name of the device that has been attached to the scheduler

Synopsis

```
#include <fbsched.h>

int fbsinfo(fbs_id, info_buf, devname)
int fbs_id;
struct fbsinfo_ds {
    int uid;
    int gid;
    int cuid;
    int cgid;
    int mode
    int key;
    int flags;
    int devid;
    int overruns;
    int cpuactive;
    int pm_cpuactive;
    int enabled;
    int filler[29]
} *info_buf;
char *devname;
```

Parameters

fbs_id a unique positive integer value representing the identifier for a frequency-based scheduler. This value can be obtained using `fbsid(3)` (page 6-16) or `fbsconfigure(3)` (page 6-8).

Enter MY_FBS to reference the frequency-based scheduler on which the calling process is scheduled without knowing the identifier.

info_buf An `fbsinfo_ds` structure to which `fbsinfo` returns information about the specified scheduler. The information returned in each component of the structure is presented below.

uid owner's user ID

<i>gid</i>	owner's group ID
<i>cuid</i>	creator's user ID
<i>cgid</i>	creator's group ID
<i>mode</i>	access modes
<i>key</i>	key
<i>flags</i>	flags word
<i>devid</i>	reserved for future use
<i>overruns</i>	total number of hard overruns for all processes on the scheduler
<i>cpuactive</i>	mask of CPUs active in the system
<i>pm_cpuactive</i>	mask of CPUs on which performance monitoring has been enabled
<i>enabled</i>	FBS enabled flag
<i>filler</i>	reserved for future use
<i>devname</i>	a variable to which fbsinfo returns the path name of the device being used as the timing source for the specified frequency-based scheduler. If end-of-cycle scheduling is specified, <i>devname</i> contains a null string.

Return Value

A return value of 0 indicates that the call has been successful. A return value of -1 indicates that an error has occurred; `errno` is set to indicate the error. Refer to the **fbsinfo(3)** man page for a listing of the types of errors that may occur.

fbsintrpt – Start/Stop/Resume Scheduling

This routine starts, stops or resumes scheduling on a frequency-based scheduler. If you invoke this routine to start scheduling, the minor cycle, major frame, and overrun count values are reset. If you invoke it to resume scheduling, these values are not reset.

Prior to invoking **fbsintrpt**, you must have invoked **fbsattach** to specify end-of-cycle scheduling or attach a timing source to the frequency-based scheduler on which you are starting scheduling (see page 6-6 for an explanation of **fbsattach**).

If you have specified a real-time clock as the timing source, scheduling will not begin until you have set and started the clock (see pages 6-25 and 6-22 for explanations of **fbssetrtc** and **fbsrunrtc**, respectively).

Synopsis

```
#include <fbsched.h>

int fbsintrpt(fbs_id, intrflag)
int fbs_id, intrflag;
```

Parameters

<i>fbs_id</i>	a unique positive integer value representing the identifier for a frequency-based scheduler. This value can be obtained using fbsid(3) (page 6-16) or fbsconfigure(3) (page 6-8).
	Enter MY_FBS to reference the frequency-based scheduler on which the calling process is scheduled without knowing the identifier.
<i>intrflag</i>	an integer value indicating whether scheduling of processes on the specified scheduler is to be started, stopped or resumed. Acceptable values and corresponding results are:
<0	Start scheduling (waking up) processes with the initial frame, cycle, and overrun count values set to zero
0	Stop scheduling processes and save the count values for the current frame and cycle
>0	Resume scheduling processes with the frame, cycle, and overrun count values set to the values that were saved when the scheduler was last stopped

Return Value

A return value of 0 indicates that the call has been successful. A return value of -1 indicates that an error has occurred; `errno` is set to indicate the error. Refer to the **fbsintrpt(3)** man page for a listing of the types of errors that may occur.

fbsremove – Remove a Frequency-Based Scheduler

This routine removes a frequency-based scheduler and frees the data structure associated with it. Prior to invoking **fbsremove**, you must ensure that the timing source is detached from the scheduler (see page 6-12 for information on the use of **fbsdetach**).

It is important to note that **fbsremove** removes all processes scheduled on the specified scheduler. It is recommended, however, that you remove all scheduled processes prior to invoking **fbsremove**. You can do so by making a call to **pgmremove** (see page 6-29 for information on the use of this routine).

Note that to remove a frequency-based scheduler, the calling process must have an effective user ID that is equal to that of the owner/creator of the scheduler.

Synopsis

```
#include <fbsched.h>

int fbsremove(fbs_id, ab)
int fbs_id, ab;
```

Parameters

fbs_id a unique positive integer value representing the identifier for a frequency-based scheduler. This value can be obtained using **fbsid(3)** (page 6-16) or **fbsconfigure(3)** (page 6-8).

Enter MY_FBS to reference the frequency-based scheduler on which the calling process is scheduled without knowing the identifier.

ab an integer value indicating the manner in which processes scheduled on the scheduler are to be handled. Acceptable values and corresponding results are:

<0	Kill and remove all processes currently scheduled on the specified scheduler
≥0	Remove all processes currently scheduled on the specified scheduler

Return Value

A return value of 0 indicates that the call has been successful. A return value of -1 indicates that an error has occurred; **errno** is set to indicate the error. Refer to the **fbsremove(3)** man page for a listing of the types of errors that may occur.

fbsresume – Resume Scheduling on a Frequency-Based Scheduler

The **fbsresume** library routine resumes scheduling of processes on a frequency-based scheduler at the specified minor cycle, major frame, and overrun count.

Note that to resume scheduling of processes on a frequency-based scheduler, the calling process must have alter permission for the scheduler.

If you wish to resume scheduling of processes on a frequency-based scheduler without altering the scheduler's current frame, cycle, and overrun values, it is recommended that you use the **fbsintrpt(3)** routine (see page 6-19 for an explanation of this routine).

CAUTION

The **fbsresume** routine clears Performance Monitor values for all processes scheduled on the specified scheduler. Changing the frame and cycle count for the scheduler causes the values that are being maintained by the Performance Monitor to be inaccurate.

Synopsis

```
#include <fbsched.h>

int fbsresume(fbs_id, frame, cycle, overruns)
int fbs_id, frame, cycle, overruns;
```

Parameters

fbs_id a unique positive integer value representing the identifier for a frequency-based scheduler. This value can be obtained using **fbsid(3)** (page 6-16) or **fbsconfigure(3)** (page 6-8).

Enter MY_FBS to reference the frequency-based scheduler on which the calling process is scheduled without knowing the identifier.

frame an integer value indicating the major frame in which you wish scheduling of processes to be resumed on the specified scheduler.

cycle an integer value indicating the minor cycle in which you wish scheduling of processes to be resumed on the specified scheduler.

This value can range from 0 to the total number of minor cycles per frame minus 1. The total number of minor cycles per frame was specified when the scheduler was created by making a call to **fbsconfigure** (see page 6-8).

overruns an integer value indicating the value to which you wish the overrun count to be set when scheduling resumes on the specified scheduler.

Specify the value -1 if you do not wish to change the overrun count.

Return Value

A return value of 0 indicates that the call has been successful. A return value of -1 indicates that an error has occurred; `errno` is set to indicate the error. Refer to the **fbsresume(3)** man page for a listing of the types of errors that may occur.

fbsrunrtc – Start/Stop a Real-Time Clock

This routine starts or stops the counting of a real-time clock that has been attached to a frequency-based scheduler.

Synopsis

```
#include <fbsched.h>

int fbsrunrtc(fbs_id, runflag)
int fbs_id, runflag;
```

Parameters

fbs_id a unique positive integer value representing the identifier for a frequency-based scheduler. This value can be obtained using **fbsid(3)** (page 6-16) or **fbsconfigure(3)** (page 6-8).

Enter MY_FBS to reference the frequency-based scheduler on which the calling process is scheduled without knowing the identifier.

runflag an integer value indicating whether the real-time clock is to be started or stopped. A nonzero value indicates that the clock is to be started. A zero value indicates that the clock is to be stopped.

Return Value

A return value of 0 indicates that the call has been successful. A return value of -1 indicates that an error has occurred; `errno` is set to indicate the error. Refer to the **fbsrunrtc(3)** man page for a listing of the types of errors that may occur.

fbsschedself – Add a Calling Process to a Frequency-Based Scheduler

This routine schedules the calling process on a frequency-based scheduler.

It is important to note that **fbsschedself** does not allow a process to set its scheduling policy and priority or its CPU bias. These tasks must be performed prior to invoking **fbsschedself**.

Note that you cannot use this routine to add **/idle** to a frequency-based scheduler.

To schedule the calling process on a frequency-based scheduler, the calling process must have alter permission for the scheduler.

To change the scheduling policy or priority of a frequency-based scheduled process, use the **sched_pgmresched** routine (see page 6-44 for an explanation of this routine).

Synopsis

```
#include <fbsched.h>

int fbsschedself(fbs_id, name, sched_buf)
int fbs_id;
char *name;
struct fbssched_buf {
    int version;
    int param;
    int period;
    int cycle;
    int ab;
    int fpid;
} *sched_buf;
```

Parameters

<i>fbs_id</i>	a unique positive integer value representing the identifier for a frequency-based scheduler. This value can be obtained using fbssid(3) (page 6-16) or fbconfigure(3) (page 6-8).
<i>name</i>	a standard path name or arbitrary content identifying the program associated with the calling process. A full or relative path name of up to 1023 characters can be specified.
<i>sched_buf</i>	a fbssched_buf structure that contains the scheduling information for the process that is to be scheduled. The type of information specified in each component is presented below.
<i>version</i>	an integer value indicating the version of <i>sched_buf</i> that is being passed to fbsschedself . Specify the symbolic constant FBSSCHED_BUF_V1 , which is defined in <fbsched.h> for this purpose.
<i>param</i>	an integer value to be passed to a process that is scheduled on a frequency-based scheduler. This value can be retrieved by the frequency-based scheduled process through a call to sched_pgmqry (see page 6-41 for an explanation of this routine).

<i>period</i>	<p>an integer value indicating the frequency with which the calling process is to be awakened in each major frame. A period of 1 indicates that the calling process is to be awakened every minor cycle; a period of 2 indicates that it is to be awakened once every two minor cycles and so on.</p> <p>This value can range from 1 to the number of minor cycles that compose a frame on the specified scheduler as defined in a call to fbconfigure (see page 6-8 for an explanation of this routine).</p>
<i>cycle</i>	<p>an integer value indicating the first minor cycle in which the calling process is scheduled to be awakened in each frame. This value can range from 0 to the total number of minor cycles per frame minus 1. The total number of minor cycles per frame is specified in a call to fbconfigure (see page 6-8 for an explanation of this routine).</p>
<i>ab</i>	<p>an integer value indicating whether or not the scheduler should be stopped in the event that the calling process causes a frame overrun. A nonzero value indicates that the scheduler will be stopped.</p>
<i>fpid</i>	<p>an integer value returned by fbsschedself that is the unique frequency-based scheduler process identifier for the scheduled process.</p>

Return Value

A return value of 0 indicates that the call has been successful. A return value of -1 indicates that an error has occurred; `errno` is set to indicate the error. Refer to the **fbsschedself(3)** man page for a listing of the types of errors that may occur.

fbsetrtc – Set a Real-Time Clock

This routine establishes the duration of a minor cycle by setting the count and the resolution values for a real-time clock to be used as an FBS timing device.

Synopsis

```
#include <fbsched.h>

int fbsetrtc(fbs_id, count, resolution)
int fbs_id, count, resolution;
```

Parameters

fbs_id a unique positive integer value representing the identifier for a frequency-based scheduler. This value can be obtained using **fbid(3)** (page 6-16) or **fbconfigure(3)** (page 6-8).
Enter MY_FBS to reference the frequency-based scheduler on which the calling process is scheduled without knowing the identifier.

count an integer value indicating the number of clock counts per minor cycle. This value can range from 1 to 65535.

resolution an integer value indicating the duration in microseconds of one clock count. This value must be one of the following: 1, 10, 100, 1000, or 10000.

Return Value

A return value of 0 indicates that the call has been successful. A return value of -1 indicates that an error has occurred; `errno` is set to indicate the error. Refer to the **fbsetrtc(3)** man page for a listing of the types of errors that may occur.

fbstrig – Make a Sleeping Frequency-Based Scheduler Process Runnable

This routine makes a process scheduled on a frequency-based scheduler runnable if it is in the **fbwait(3)** sleep state. In addition, a context switch can be forced on the processor on which the process is executing.

Synopsis

```
#include <fbsched.h>

int fbstrig (int fpid, int tgrflg);
int fpid, tgrflg;
```

Parameters

fpid an integer value providing the unique frequency-based scheduler process identifier for the sleeping process. This value is obtained when you make a call to **sched_pgmadd** (see page 6-38 for an explanation of this routine).

tgrflg an integer value indicating whether or not a context switch is to be forced on the processor on which the awakened process is executing. A nonzero value indicates that a context switch is to be forced.

Return Value

A return value of 0 indicates that the call has been successful. A return value of -1 indicates that an error has occurred; `errno` is set to indicate the error. Refer to the **fbstrig(3)** man page for a listing of the types of errors that may occur.

fbwait – Wait on a Frequency-Based Scheduler

This routine enables a process that is scheduled on a frequency-based scheduler to sleep until its next scheduled minor cycle.

When the scheduled process does not call this service by its next scheduled minor cycle, either a soft overrun or hard overrun is incurred.

A soft overrun occurs if the per-process count of consecutively missed scheduled minor cycles does not reach or exceed the per-process soft overrun limit. When a soft overrun occurs, the process returns immediately from the **fbwait** call instead of blocking to wait for the next scheduled minor cycle.

When the count of consecutively missed scheduled minor cycles reaches or exceeds the per-process overrun limit, a hard overrun occurs. In this case, the process is blocked in **fbwait** until the next scheduled minor cycle.

A process' consecutive soft overrun limit may be changed from the default value of 0 using **sched_pgm_set_soft_overrun_limit(3)**. The hard overrun count, which can be read using **pmqryp(3)**, and the soft overrun count, which can be read using **sched_pgm_soft_overrun_query(3)**, indicate if the process is actually running at its assigned frequency.

Synopsis

```
#include <fbsched.h>
int fbwait(void)
```

Return Value

A return value of 0 indicates that the process has been awakened by the frequency-based scheduler. A return value of 1 indicates that the process has been awakened by **fbstrig(3)**. A return value of 2 indicates that the process did not sleep because the kernel detected a soft overrun and is allowing the process to attempt to recover from it. A return value of -1 indicates that an error has occurred; **errno** is set to indicate the error. Refer to the **fbwait(3)** man page for a listing of the types of errors that can occur.

nametopid, namepid – Return the Process ID for a Specified Process Name

The **namepid** routine returns the process ID (pid) of the specified process. The **nametopid** routine performs the same function but allows the search to be restricted to processes scheduled on a particular frequency-based scheduler.

Synopsis

```
#include <fbsched.h>

int nametopid(name, fbs_key, cpu)
int namepid(name, cpu)
int fbs_key, cpu;
char *name;
```

Parameters

<i>name</i>	the path name that identifies the process
<i>fbs_key</i>	an integer value identifying the frequency-based scheduler. This value is the same value specified for <i>key</i> when the scheduler was created using fbsconfigure (see page 6-8 for an explanation of this routine).
<i>cpu</i>	an integer value indicating the processor(s) to be used in conjunction with the value of the <i>name</i> parameter to identify the process. Acceptable values and corresponding results are: 0 the first process named by <i>name</i> currently running on the processor from which the call is made is requested -1 the first process named by <i>name</i> currently running on any processor is requested bit mask those processors with (<i>cpu</i> & (1<< <i>i</i>)) set (where <i>i</i> is an integer ranging from 0 to 15 and representing a CPU) are requested

Return Value

On success, **nametopid** returns the process ID. On failure, **nametopid** returns -1 and **errno** is set to indicate the error. Refer to the **nametopid(3)** man page for a listing of the types of errors that can occur

pgmremove – Remove a Process from a Frequency-Based Scheduler

This routine removes a process from a frequency-based scheduler. Identify the process to remove by using one of the following methods:

- Specify the name of the process and the CPU on which it is scheduled.
- Specify the process' frequency-based scheduler process identifier.
- Specify the name of the process, the CPU on which it is scheduled, and its frequency-based scheduler process identifier.

NOTE

The only method that can be used to identify a process that has been scheduled multiple times on the same CPU is to specify its frequency-based scheduler process identifier.

Synopsis

```
#include <fbsched.h>

int pgmremove(fbs_id, name, cpu, fpid, ab)
int fbs_id, cpu, fpid, ab;
char *name;
```

Parameters

fbs_id a unique positive integer value representing the identifier for a frequency-based scheduler. This value can be obtained using **fbsid(3)** (page 6-16) or **fbsconfigure(3)** (page 6-8).

Enter MY_FBS to reference the frequency-based scheduler on which the calling process is scheduled without knowing the identifier.

name a standard path name identifying the process to be removed from the specified scheduler. A full or relative path name of up to 1024 characters can be specified. If this variable contains the null string, you must provide the frequency-based scheduler process identifier in the *fpid* parameter.

cpu an integer value indicating the processor(s) to be used in conjunction with the value of the *name* parameter to identify the process to be removed from the specified scheduler. Acceptable values and corresponding results are:

0	the first process named by <i>name</i> currently running on the processor from which the call is made is removed
-1	the first process named by <i>name</i> currently running on any processor is removed
bit mask	if (<i>cpu</i> & (1<< <i>i</i>)) is set (where <i>i</i> is an integer ranging from 0 to 15 and representing a CPU) and it is the only

bit set, the first process named by *name* that is running on CPU *i* is removed

If (*cpu* & (1<<*i*)) is set and it is not the only bit set, the first process named by *name* currently running on any of the selected CPUs is removed.

fpid an integer value providing the unique frequency-based scheduler process identifier for the process to be removed. This value is obtained when you make a call to **sched_pgmadd** (see page 6-38 for an explanation of this routine). This value must be -1 if you choose to identify the program to be removed only by specifying *name* and *cpu*.

ab a flag that contains an integer value indicating the manner in which the specified process is to be removed from the specified scheduler. A positive value indicates that the process is to be removed from the scheduler but allowed to continue executing. A negative value indicates that the process is to be removed from the scheduler and terminated.

Return Value

A return value of 0 indicates that the call has been successful. A return value of -1 indicates that an error has occurred; `errno` is set to indicate the error. Refer to the **pgmremove (3)** man page for a listing of the types of errors that can occur.

pgmtrigger – Trigger a Process on a Frequency-Based Scheduler

This routine enables a process to wake a process that is in the **fbwait** sleep state. It is important to note that the calling process does not have to be scheduled on a frequency-based scheduler; the target process must be.

Synopsis

```
#include <fbsched.h>

int pgmtrigger(fpid, tgrflg)
int fpid, tgrflg;
```

Parameters

fpid an integer value providing the unique frequency-based scheduler process identifier for the sleeping process. This value is obtained when you make a call to **sched_pgmadd** (see page 6-38 for an explanation of this routine).

tgrflg an integer value indicating whether or not a context switch is to be forced on the processor on which the awakened process is executing. A nonzero value indicates that a context switch is to be forced.

Return Value

A return value of 0 indicates that the call has been successful. A return value of -1 indicates that an error has occurred; **errno** is set to indicate the error. Refer to the **pgmtrigger(3)** man page for a listing of the types of errors that may occur.

sched_fbsqry – Query Processes on a Frequency-Based Scheduler

This routine obtains information about processes scheduled on a frequency-based scheduler. Information is returned for all processes scheduled on the user-specified processor(s). Information provided for each process includes the following:

- a mask of the CPU(s) on which the process can execute
- the frequency-based scheduler process identifier
- the policy under which the process has been scheduled
- the scheduling priority
- the period (the number of minor cycles indicating the frequency with which the process is awakened in each major frame)
- the starting base cycle (the first minor cycle in which the process is scheduled to be awakened in each major frame)
- the value of the “halt on overrun” flag
- the current state of the process

Synopsis

```
#include <fbsched.h>

int sched_fbsqry(fbs_id, cpu, fbs_buf, buf_cnt)
int fbs_id, cpu, buf_cnt;
struct pgm2_ds {
    char *name_ptr;
    int cpu;
    int fpid;
    int cid;
    int prior;
    int param;
    int period;
    int cycle;
    int halt;
    int status;
} *fbs_buf;
```

Parameters

fbs_id a unique positive integer value representing the identifier for a frequency-based scheduler. This value can be obtained using **fbsid(3)** (page 6-16) or **fbsconfigure(3)** (page 6-8).

Enter MY_FBS to reference the frequency-based scheduler on which the calling process is scheduled without knowing the identifier.

cpu an integer value indicating the processor(s) for which scheduling information is to be obtained. The acceptable values and corresponding results are:

0	scheduling information for processes executing on the processor from which the call is made is returned
---	---------------------------------------------------------------------------------------------------------

-1	scheduling information for all processes on the scheduler is returned
bit mask	if $(cpu \& (1 \ll i))$ is set (where i is an integer ranging from 0 to 15 and representing a CPU), scheduling information for processes executing on CPU i is returned
<i>fbs_buf</i>	a pointer to an array of <code>pgm2_ds</code> structures to which <code>sched_fbsqry</code> returns scheduling information for each process on the processor(s) specified with the <i>cpu</i> parameter. The type of information returned in each component of the structure for a single process is presented below.
<i>name_ptr</i>	a pointer to a variable that contains a path name identifying the process for which information is returned. Note that the application must deallocate the memory associated with <i>name_ptr</i> for all valid entries using <code>free</code> (see <code>malloc(3)</code>).
<i>cpu</i>	a bit mask indicating the processor(s) on which the process can execute
<i>fpid</i>	the process' frequency-based scheduler process identifier
<i>cid</i>	the process' scheduling policy: <code>SCHED_FIFO</code> , <code>SCHED_RR</code> or <code>SCHED_OTHER</code>
<i>prior</i>	an integer value indicating the specified process' scheduling priority
<i>param</i>	the process' initiation parameter (optional)
<i>period</i>	the number of minor cycles indicating the frequency with which the process is to be awakened in each major frame
<i>cycle</i>	the first minor cycle in which the process is scheduled to be awakened in each major frame (starting base cycle)
<i>halt</i>	the value of the "halt on overrun" flag. A nonzero value indicates that the flag is set. A value of zero indicates that the flag is not set.
<i>status</i>	the current state of the process as defined in <code><fbsched.h></code> .
<i>buf_cnt</i>	an integer value indicating the number of structures in the array to which <i>fbs_buf</i> points.

Return Value

A return value of 0 indicates that the call has been successful. A return value of -1 indicates that an error has occurred; `errno` is set to indicate the error. Refer to the `sched_fbsqry(3)` man page for a listing of the types of errors that may occur.

sched_pgm_set_soft_ouerrun_limit – Set Soft Overrun Limit

This routine sets the consecutive soft overrun limit for a currently scheduled process on the frequency-based scheduler. To set the consecutive soft overrun limit, the calling process must have alter permission for the scheduler.

Identify the process by using one of the following methods:

- Specify the name of the process and the CPU on which it is scheduled.
- Specify the process' frequency-based scheduler process identifier.
- Specify the name of the process, the CPU on which it is scheduled, and its frequency-based scheduler process identifier.

Synopsis

```
#include <fbsched.h>

int sched_pgm_set_soft_ouerrun_limit (fbs_id, soft_ouerrun_buf)
int fbs_id;
struct soft_ouerrun_ds{
    char *name_ptr;
    int cpu;
    int fpid;
    int soft_limit;
} *soft_ouerrun_buf;
```

Parameters

fbs_id a unique positive integer value representing the identifier for a frequency-based scheduler. This value can be obtained using **fbsid(3)** (page 6-16) or **fbsconfigure(3)** (page 6-8).

Enter MY_FBS to reference the frequency-based scheduler on which the calling process is scheduled without knowing the identifier.

soft_ouerrun_buf a `soft_ouerrun_ds` structure that contains the soft overrun status for the scheduled process. The structure definition is:

name_ptr a pointer to the path name that identifies the process. If the path name is the null string, the *fpid* field must be given.

cpu an integer value indicating the processors on which the specified program can be scheduled to run. Acceptable values and corresponding results are:

0 the program pointed to by *name_ptr* can be scheduled on the processor from which the call is made.

-1 the program pointed to by *name_ptr* can be scheduled on any processor.

bit mask if $(cpu \& (1 \ll i))$ is set (where *i* is an integer

ranging from 0 to 15 and representing a CPU), the program pointed to by *name_ptr* can be scheduled on CPU *i*.

fpid an integer value providing the unique frequency-based scheduler process identifier. If this value is -1, *name_ptr* must be supplied.

soft_limit the number of consecutive soft overruns allowed to occur before failure. This value must be non-negative and less than INT_MAX (see `/usr/include/limits.h`). By default, this value is 0.

Return Value

A return value of 0 indicates that the call has been successful. A return value of -1 indicates that an error has occurred; `errno` is set to indicate the error. Refer to the `sched_pgm_set_soft_overrun_limit(3)` man page for a listing of the types of errors that may occur.

sched_pgm_soft_ouerrun_query – Query Soft Overrun Processing

This routine obtains the status of soft overrun processing for a currently scheduled process on a frequency-based scheduler.

Synopsis

```
#include <fbsched.h>
int sched_pgm_soft_ouerrun_query(fbs_id, soft_ouerrun_info_buf)
int fbs_id;
struct soft_ouerrun_info_ds{
    char *name_ptr;
    int cpu;
    int fpid;
    int soft_limit;
    int soft_total;
} *soft_ouerrun_info_buf;
```

Parameters

fbs_id a unique positive integer value representing the identifier for a frequency-based scheduler. This value can be obtained using **fbsid(3)** (page 6-16) or **fbsconfigure(3)** (page 6-8).

soft_ouerrun_info_buf a `soft_ouerrun_info_ds` structure that contains the soft overrun status for the scheduled process. The structure definition is:

name_ptr a pointer to the path name that identifies the process. If the path name is the null string, the *fpid* field must be given.

cpu an integer value indicating the processors on which the specified program can be scheduled to run. Acceptable values and corresponding results are:

0 the program pointed to by *name_ptr* can be scheduled on the processor from which the call is made.

-1 the program pointed to by *name_ptr* can be scheduled on any processor.

bit mask If (*cpu* & (1<<*i*)) is set (where *i* is an integer ranging from 0 to 15 and representing a CPU), the program pointed to by *name_ptr* can be scheduled on CPU *i*.

fpid an integer value providing the unique frequency-based scheduler process identifier. If this value is -1, *name_ptr* must be supplied.

soft_limit the number of consecutive soft overruns allowed to occur before failure. This value must be non-negative and less than INT_MAX (see `/usr/include/limits.h`). By default, this value is 0.

soft_total the total number of soft overruns incurred by the process

Return Value

A return value of 0 indicates that the call has been successful. A return value of -1 indicates that an error has occurred; `errno` is set to indicate the error. Refer to the `sched_pgm_soft_overrun_query(3)` man page for a listing of the types of errors that may occur.

sched_pgmadd – Schedule a Process on a Frequency-Based Scheduler

The `sched_pgmadd` routine creates a new process and schedules it on a frequency-based scheduler.

It is important to note that to use this routine to (1) change a process' scheduling policy to the SCHED_FIFO or the SCHED_RR policy or (2) change the priority of a process scheduled under SCHED_FIFO or SCHED_RR, or (3) raise the priority of a process scheduled under SCHED_OTHER above a per-process limit, the effective user ID of the calling process must match the effective user ID of the target process (the process for which the scheduling policy and priority are being set).

To modify the process' CPU bias when you invoke this routine, the real or effective user ID of the calling process must match the real or saved user ID of the process for which the CPU assignment is being changed.

Synopsis

```
#include <fbsched.h>

int sched_pgmadd(fbs_id, sched_buf)
int fbs_id;
struct pgm2_ds {
    char *name_ptr;
    int cpu;
    int fpid;
    int cid;
    int prior;
    int param;
    int period;
    int cycle;
    int halt;
    int status;
} *sched_buf;
```

Parameters

fbs_id a unique positive integer value representing the identifier for a frequency-based scheduler. This value can be obtained using **fbsid(3)** (page 6-16) or **fbsconfigure(3)** (page 6-8).

Enter MY_FBS to reference the frequency-based scheduler on which the calling process is scheduled without knowing the identifier.

sched_buf a `pgm2_ds` structure that contains the scheduling parameters with which you wish to schedule the process. The type of information specified in each component is presented below. Note that the *status* component is ignored on this call.

name_ptr a pointer to a variable that contains a path name identifying the program to be scheduled. A full or relative path name of up to 1024 characters can be specified.

<i>cpu</i>	<p>an integer value indicating the processors on which the specified program can be scheduled to run. Acceptable values and corresponding results are:</p> <p>0 the program pointed to by <i>name_ptr</i> can be scheduled on the processor from which the call is made.</p> <p>-1 the program pointed to by <i>name_ptr</i> can be scheduled on any processor.</p> <p>bit mask if $(cpu \& (1 \ll i))$ is set (where <i>i</i> is an integer ranging from 0 to 15 and representing a CPU), the program pointed to by <i>name_ptr</i> can be scheduled on CPU <i>i</i>.</p>
<i>fpid</i>	an integer value returned by sched_pgmadd that is the unique frequency-based scheduler process identifier for the scheduled process
<i>cid</i>	an integer value indicating the POSIX scheduling policy under which the specified program is to be scheduled. Scheduling policies are defined in the file <sched.h> : SCHED_FIFO, SCHED_RR or SCHED_OTHER.
<i>prior</i>	an integer value indicating the scheduling priority of the specified process. The range of acceptable priority values is governed by the scheduling policy specified.
<i>param</i>	an integer value to be passed to a process scheduled on a frequency-based scheduler. This value can be retrieved by the frequency-based scheduled process through a call to sched_pgmqry (see page 6-38 for an explanation of this routine).
<i>period</i>	<p>an integer value indicating the frequency with which the specified program is to be awakened in each major frame.</p> <p>A period of 1 indicates that the specified program is to be awakened every minor cycle; a period of 2 indicates that it is to be awakened once every two minor cycles, and so on.</p> <p>This value can range from 1 to the number of minor cycles that compose a frame on the specified scheduler as defined in a call to fbsconfigure (see page 6-8).</p>
<i>cycle</i>	an integer value indicating the first minor cycle in which the specified program is scheduled to be awakened in each frame. This value can range from 0 to the total number of minor cycles per frame minus 1. The total number of minor cycles per frame is specified in a call to fbsconfigure (see page 6-8).
<i>halt</i>	an integer value indicating whether or not the scheduler should be stopped in the event that the specified

program causes a frame overrun. A nonzero value indicates that the scheduler will be stopped.

Return Value

A return value of 0 indicates that the call has been successful. A return value of -1 indicates that an error has occurred; `errno` is set to indicate the error. Refer to the `sched_pgmadd(3)` man page for a listing of the types of errors that may occur.

sched_pgmqry – Query a Process

The `sched_pgmqry` routine obtains information for a particular process that has been scheduled on a frequency-based scheduler. You can identify the process by using one of the following methods:

- Specify the name of the process and the CPU(s) on which it is scheduled.
- Specify the process' frequency-based scheduler process identifier.
- Specify the name of the process, the CPU on which it is scheduled, and its frequency-based scheduler process identifier.

NOTE

The only method that can be used to identify a process that has been scheduled multiple times on the same CPU is to specify its frequency-based scheduler process identifier.

Information returned includes the following:

- the process' path name
- the CPU on which the process can execute
- the frequency-based scheduler process identifier
- the scheduling policy under which the process has been scheduled
- the scheduling priority
- the period (the number of minor cycles indicating the frequency with which the process is awakened in each major frame)
- the starting base cycle (the first minor cycle in which the process is scheduled to be awakened in each major frame)
- the value of the "halt on overrun" flag

Synopsis

```
#include <fbsched.h>

int sched_pgmqry(fbs_id, qry_buf)
int fbs_id;
struct pgm2_ds {
    char *name_ptr;
    int cpu;
    int fpid;
    int cid;
    int prior;
    int param;
    int period;
    int cycle;
    int halt;
    int status;
} *qry_buf;
```

Parameters

fbs_id a unique positive integer value representing the identifier for a frequency-based scheduler. This value can be obtained using **fbsid(3)** (page 6-16) or **fbsconfigure(3)** (page 6-8).

Enter MY_FBS to reference the frequency-based scheduler on which the calling process is scheduled without knowing the identifier.

qry_buf a pointer to a `pgm2_ds` structure that contains scheduling information for the process. **sched_pgmqry** will return to this structure the scheduling information for a specified process. The information contained in each component of the structure to which *qry_buf* points is presented below.

name_ptr a pointer to the path name identifying the process for which information is to be returned. A full or relative path name of up to 1024 characters can be specified. If the pointer points to a null string, you must provide the frequency-based scheduler process identifier in the *fpid* component.

cpu an integer value indicating the processor(s) to be used with the value of *name_ptr* to identify the program for which information is to be obtained. Acceptable values and corresponding results follow:

0 the first process whose name matches the name pointed to by *name_ptr* currently running on the processor from which the call is made is requested.

-1 the first process whose name matches the name pointed to by *name_ptr* currently running on any processor is requested.

bit mask if $(cpu \& (1 \ll i))$ is set (where *i* is an integer ranging from 0 to 15 and representing a CPU) and it is the only bit set, the first process whose name matches the name pointed to by *name_ptr* running on CPU *i* is requested.

If $(cpu \& (1 \ll i))$ is set and it is not the only bit set, the first process whose name matches the name pointed to by *name_ptr* currently running on any of the selected CPUs is requested.

<i>fpid</i>	an integer value providing the frequency-based scheduler process identifier for the process for which information is to be returned. This value is obtained when you make a call to sched_pgmadd (see page 6-38). This value must be -1 if you wish to identify the program to be queried only by specifying <i>name_ptr</i> and <i>cpu</i> .
<i>cid</i>	an integer value indicating the specified process' scheduling policy.
<i>prior</i>	an integer value indicating the process' scheduling priority.
<i>param</i>	an integer value indicating the value passed to the process via a call to sched_pgmadd or sched_pgmresched .
<i>period</i>	an integer value indicating the frequency with which the specified program is to be awakened in each major frame.
<i>cycle</i>	an integer value indicating the first minor cycle in which the specified process is scheduled to be awakened in each frame.
<i>halt</i>	an integer value indicating the value of the "halt on overrun" flag. A nonzero value indicates that the flag is set. A value of zero indicates that the flag is not set.
<i>status</i>	an integer value indicating the current state of the specified process as defined in <fbsched.h>

Return Value

A return value of 0 indicates that the call has been successful. A return value of -1 indicates that an error has occurred; `errno` is set to indicate the error. Refer to the **sched_pgmqry (3)** man page for a listing of the types of errors that may occur.

sched_pgmresched – Reschedule a Process

The **sched_pgmresched** routine changes the scheduling parameters for a process that is scheduled on a frequency-based scheduler. You may wish, for example, to change a program's policy or priority or the frequency with which it is scheduled to run. You cannot, however, change the CPU on which it has been scheduled.

It is important to note that to use this routine to (1) change a process' scheduling policy to the SCHED_FIFO or the SCHED_RR policy or (2) change the priority of a process scheduled under SCHED_FIFO or SCHED_RR, or (3) raise the priority of a process scheduled under SCHED_OTHER above a per-process limit, the effective user ID of the calling process must match the effective user ID of the target process (the process for which the scheduling policy and priority are being set).

Identify the process to reschedule using one of the following methods:

- Specify the name of the process and the CPU on which it is scheduled.
- Specify the process' frequency-based scheduler process identifier.
- Specify the name of the process, the CPU on which it is scheduled, and its frequency-based scheduler process identifier.

NOTE

The only method that can be used to identify a process that has been scheduled multiple times on the same CPU is to specify its frequency-based scheduler process identifier.

Synopsis

```
#include <fbsched.h>

int sched_pgmresched(fbs_id, rsch_buf)
int fbs_id;
struct pgm2_ds {
    char *name_ptr;
    int cpu;
    int fpid;
    int cid;
    int prior;
    int param;
    int period;
    int cycle;
    int halt;
    int status;
} *rsch_buf;
```

Parameters

fbs_id a unique positive integer value representing the identifier for a frequency-based scheduler. This value can be obtained using **fbsid(3)** (page 6-16) or **fbsconfigure(3)** (page 6-8).

Enter MY_FBS to reference the frequency-based scheduler on which the calling process is scheduled without knowing the identifier.

rsch_buf

A `pgm2_ds` structure that contains the scheduling parameters with which to reschedule the process. The type of information specified in each component is presented below. Note that the *status* component is ignored on this call.

name_ptr a pointer to a variable that contains a path name identifying the process for which information is to be rescheduled. A full or relative path name of up to 1024 characters can be specified. If the pointer points to a null string, you must provide the frequency-based scheduler process identifier in the *fpid* component.

cpu an integer value indicating the processor(s) to be used with the value of *name_ptr* to identify the process to be rescheduled. Acceptable values and corresponding results follow:

0 the first process whose name matches the name pointed to by *name_ptr* currently running on the processor from which the call is made is rescheduled.

-1 the first process whose name matches the name pointed to by *name_ptr* currently running on any processor is rescheduled.

bit mask if $(cpu \& (1 \ll i))$ is set (where *i* is an integer ranging from 0 to 15 and representing a CPU) and it is the only bit set, the first process whose name matches the name pointed to by *name_ptr* running on CPU *i* is rescheduled.

If $(cpu \& (1 \ll i))$ is set and it is not the only bit set, the first process whose name matches the name pointed to by *name_ptr* currently running on any of the selected CPUs is rescheduled.

fpid an integer value providing the frequency-based scheduler process identifier for the process to be rescheduled. This value is obtained when you make a call to `sched_pgmadd` (see page 6-38). This value must be -1 if you wish to identify the program to be queried only by specifying *name_ptr* and *cpu*.

cid an integer value indicating the specified process' scheduling policy: SCHED_FIFO, SCHED_RR or SCHED_OTHER.

prior an integer value indicating the process' scheduling priority. The range of acceptable priority values is governed by the scheduling policy specified.

<i>param</i>	an integer value indicating the value passed to the process scheduled on a frequency-based scheduler.
<i>period</i>	<p>an integer value indicating the frequency with which the specified program is to be awakened in each major frame.</p> <p>A period of 1 indicates that the specified program is to be awakened every minor cycle; a period of 2 indicates that it is to be awakened once every two minor cycles, and so on.</p> <p>This value can range from 1 to the number of minor cycles that compose a frame on the specified scheduler as defined in a call to fbsconfigure (see page 6-8)</p>
<i>cycle</i>	<p>an integer value indicating the first minor cycle in which the specified process is scheduled to be awakened in each frame.</p> <p>This value can range from 0 to the total number of minor cycles per frame minus 1. The total number of minor cycles per frame is specified in a call to fbsconfigure (see page 6-8).</p>
<i>halt</i>	an integer value indicating whether or not the scheduler should be stopped in the event that the specified process causes a frame overrun. A nonzero value indicates that the scheduler will be stopped.

Return Value

A return value of 0 indicates that the call has been successful. A return value of -1 indicates that an error has occurred; `errno` is set to indicate the error. Refer to the **sched_pgmresched(3)** man page for a listing of the types of errors that may occur.

Performance Monitor Routines

The Performance Monitor routines provide access to the key features of the Performance Monitor. They enable you to perform such basic operations as:

- clearing performance monitor values for a process or processor
- starting and stopping performance monitoring for a process or processor
- obtaining performance monitor values for a process or processor

Routine Summary

Performance Monitor routines are summarized in Table 6-3. Complete information about each routine is provided under the section “Using Performance Monitor Routines.”

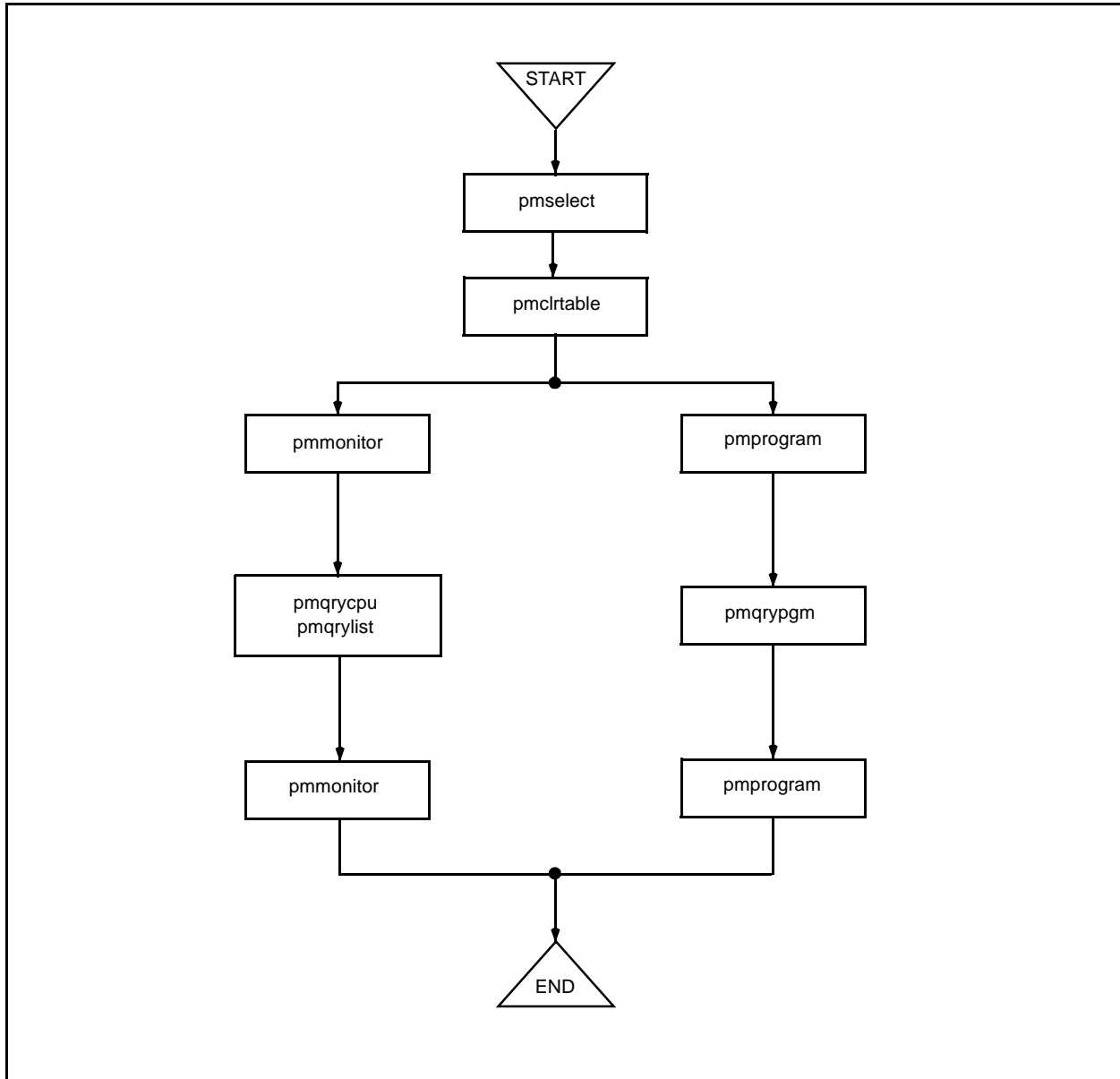
Table 6-3. Performance Monitor C Library Routines

Routine	Page	Description
pmclrpqm	6-49	Clear values for a process
pmclrtable	6-51	Clear values for processor(s)
pmmonitor	6-52	Start/stop performance monitoring on processor(s)
pmprogram	6-53	Start/stop performance monitoring on a process
pmqrycpu	6-55	Query values for selected processor(s)
pmqrylist	6-58	Query values for a list of processes
pmqrypqm	6-60	Query values for a selected process
pmqrytimer	6-63	Query Performance Monitor mode
pmselect	6-64	Select Performance Monitor mode

C Library Call Sequence

The approximate order in which you might call the Performance Monitor routines from an application program is illustrated in Figure 6-2..

Figure 6-2. C Library Call Sequence: Performance Monitor



Using Performance Monitor Routines

In the sections that follow, the Performance Monitor routines contained in the `libccur_fbsched` library are presented in alphabetical order.

pmclrpqm – Clear Values for a Process

This routine clears performance monitor values for a particular process that has been scheduled on a frequency-based scheduler. You can identify the process using one of the following methods:

- Specify the name of the process and the CPU(s) on which it is scheduled.
- Specify the process' frequency-based scheduler process identifier.
- Specify the name of the process, the CPU on which it is scheduled, and its frequency-based scheduler process identifier.

NOTE

This routine clears the process' total soft overrun count.

Synopsis

```
#include <fbsched.h>

int pmclrpqm(fbs_id, name, cpu, fpid)
int fbs_id, cpu, fpid;
char *name;
```

Parameters

fbs_id a unique positive integer value representing the identifier for a frequency-based scheduler. This value can be obtained using **fbsid(3)** (page 6-16) or **fbsconfigure(3)** (page 6-8).

Enter MY_FBS to reference the frequency-based scheduler on which the calling process is scheduled without knowing the identifier.

name a path name identifying the process for which values are to be cleared. A full or relative path name of up to 1024 characters can be specified. If this variable is the null string, you must provide the frequency-based scheduler process identifier in the *fpid* parameter.

cpu an integer value indicating the processor(s) to be used in conjunction with the value of the *name* parameter to identify the process for which values are to be cleared. Acceptable values and corresponding results are presented below.

0 the first process named by *name* currently running on the processor from which the call is made is specified

-1 the first process named by *name* currently running on any processor is specified

bit mask if $(cpu \& (1 \ll i))$ is set (where *i* is an integer ranging from 0 to 15 and representing a CPU) and it is the only bit set, the first process named by *name* currently running on CPU *i* is specified

If $(cpu \& (1 \ll i))$ is set and it is not the only bit set, the first process named by *name* currently running on any of the selected CPUs is specified.

fpid an integer value providing the unique frequency-based scheduler process identifier for the process for which values are to be cleared. This value is obtained when you make a call to **sched_pgmadd** (see page 6-38). This value must be -1 to identify the process only by specifying *name* and *cpu*.

Return Value

A return value of 0 indicates that the call has been successful. A return value of -1 indicates that an error has occurred; `errno` is set to indicate the error. Refer to the **pmc1rpgm(3)** man page for a listing of the types of errors that may occur.

pmclrtable – Clear Values for Processor(s)

This routine clears performance monitor values for frequency-based scheduled processes on one or more specified processors on a selected scheduler.

NOTE

This routine clears the total soft overrun count for all related processes.

Synopsis

```
#include <fbsched.h>

int pmclrtable(fbs_id, cpu)
int fbs_id, cpu;
```

Parameters

<i>fbs_id</i>	a unique positive integer value representing the identifier for a frequency-based scheduler. This value can be obtained using fbsid(3) (page 6-16) or fbsconfigure(3) (page 6-8). Enter MY_FBS to reference the frequency-based scheduler on which the calling process is scheduled without knowing the identifier.
<i>cpu</i>	an integer value indicating the processor or processors for which performance monitor values are to be cleared. Acceptable values and corresponding results are presented below.
0	performance monitor values for frequency-based scheduled processes executing on the processor from which the call is made are cleared
-1	performance monitor values for all processes on the scheduler are cleared
bit mask	if (<i>cpu</i> & (1<< <i>i</i>)) is set (where <i>i</i> is an integer ranging from 0 to 15 and representing a CPU), performance monitor values for processes executing on CPU <i>i</i> are cleared

Return Value

A return value of 0 indicates that the call has been successful. A return value of -1 indicates that an error has occurred; `errno` is set to indicate the error. Refer to the **pmclrtable(3)** man page for a listing of the types of errors that may occur.

pmmonitor – Start/Stop Performance Monitoring on Processor(s)

This routine starts or stops performance monitoring for frequency-based scheduled processes on one or more processors on a selected scheduler.

Synopsis

```
#include <fbsched.h>

int pmmmonitor(fbs_id, pmflag, cpu)
int fbs_id, pmflag, cpu;
```

Parameters

<i>fbs_id</i>	a unique positive integer value representing the identifier for a frequency-based scheduler. This value can be obtained using fbsid(3) (page 6-16) or fbsconfigure(3) (page 6-8). Enter MY_FBS to reference the frequency-based scheduler on which the calling process is scheduled without knowing the identifier.
<i>pmflag</i>	an integer value indicating whether performance monitoring is to be started or stopped. A nonzero value indicates that performance monitoring is to be started. A zero value indicates that performance monitoring is to be stopped.
<i>cpu</i>	an integer that indicates the processor or processors for which performance monitoring is to be started or stopped. Acceptable values and corresponding results are presented below.
0	performance monitoring for frequency-based scheduled processes executing on the processor from which the call is made is started or stopped.
-1	performance monitoring for all processes on the scheduler is started or stopped.
bit mask	if (<i>cpu</i> & (1<< <i>i</i>)) is set (where <i>i</i> is an integer ranging from 0 to 15 and representing a CPU), performance monitoring for processes executing on CPU <i>i</i> is started or stopped.

Return Value

A return value of 0 indicates that the call has been successful. A return value of -1 indicates that an error has occurred; `errno` is set to indicate the error. Refer to the **pmmmonitor(3)** man page for a listing of the types of errors that may occur.

pmprogram – Start/Stop Performance Monitoring on a Process

This routine starts or stops performance monitoring for a particular process that has been scheduled on a frequency-based scheduler. Identify the process by using one of the following methods:

- Specify the name of the process and the CPU(s) on which it is scheduled.
- Specify the process' frequency-based scheduler process identifier.
- Specify the name of the process, the CPU(s) on which it is scheduled, and its frequency-based scheduler process identifier.

Synopsis

```
#include <fbsched.h>

int pmprogram(fbs_id, name, cpu, fpid, pmflag)
int fbs_id, cpu, fpid, pmflag;
char *name;
```

Parameters

fbs_id a unique positive integer value representing the identifier for a frequency-based scheduler. This value can be obtained using **fbsid(3)** (page 6-16) or **fbsconfigure(3)** (page 6-8).

Enter MY_FBS to reference the frequency-based scheduler on which the calling process is scheduled without knowing the identifier.

name a path name identifying the process for which performance monitoring is to be started or stopped. A full or relative path name of up to 1024 characters can be specified. If this variable is the null string, you must provide the frequency-based scheduler process identifier in the *fpid* parameter.

cpu an integer value indicating the processor(s) to be used in conjunction with the value of the *name* parameter to identify the process for which performance monitoring is to be started or stopped. Valid values are presented below:

0 the first process named by *name* currently running on the processor from which the call is made is specified

-1 the first process named by *name* currently running on any processor is specified

bit mask if $(cpu \& (1 \ll i))$ is set (where *i* is an integer ranging from 0 to 15 and representing a CPU) and it is the only bit set, the first process named by *name* currently running on CPU *i* is specified.

If $(cpu \& (1 \ll i))$ is set and it is not the only bit set, the first process named by *name* currently running on any of the selected CPUs is specified.

fpid an integer value providing the unique frequency-based scheduler process identifier for the process for which performance monitoring is to be started or stopped.

Get this value by making a call to `sched_pgmadd` (see page 6-38). This value must be -1 if you wish to identify the process only by specifying *name* and *cpu*.

pmflag an integer value indicating whether performance monitoring is to be started or stopped. A nonzero value indicates that performance monitoring is to be started. A zero value indicates that performance monitoring is to be stopped.

Return Value

A return value of 0 indicates that the call has been successful. A return value of -1 indicates that an error has occurred; `errno` is set to indicate the error. Refer to the `pmprogram(3)` man page for a listing of the types of errors that may occur.

pmqrycpu – Query Values for Selected Processor(s)

This routine obtains performance monitor values for frequency-based scheduled processes on one or more specified processors on a selected scheduler.

Note that all time related units are in microseconds (usecs).

Synopsis

```
#include <fbsched.h>

int pmqrycpu(fbs_id, cpu, pm_buf, buf_cnt)
int fbs_id, cpu, buf_cnt;
struct pmqry_ds {
    int fpid;
    struct timespec lastcyc_tm;
    int tot_cycles;
    struct timespec tot_cycles_tm;
    int overruns;
    struct timespec mincyc_tm;
    int mincyc_cycle;
    int mincyc_frame;
    struct timespec maxcyc_tm;
    int maxcyc_cycle;
    int maxcyc_frame;
    struct timespec minframe_tm;
    int minframe;
    struct timespec maxframe_tm;
    int maxframe;
} *pm_buf;
```

Parameters

fbs_id a unique positive integer value representing the identifier for a frequency-based scheduler. This value can be obtained using **fbsid(3)** (page 6-16) or **fbsconfigure(3)** (page 6-8).

Enter MY_FBS to reference the frequency-based scheduler on which the calling process is scheduled without knowing the identifier.

cpu an integer value indicating the processor(s) for which performance monitor values are to be obtained. Acceptable values and corresponding results are presented below.

0 performance monitor values for frequency-based scheduled processes executing on the processor from which the call is made are returned.

-1 performance monitor values for all processes on the scheduler are returned.

bit mask if (*cpu* & (1<<*i*)) is set (where *i* is an integer ranging from 0 to 15 and representing a CPU), performance monitor values for processes executing on CPU *i* are returned.

pm_buf a pointer to an array of `pmqry_ds` structures to which `pmqrycpu` returns the performance monitor values for each frequency-based scheduled process on the processor(s) specified with the *cpu* parameter.

The number of processes for which these values are returned is bound by the value of the *buf_cnt* parameter. The type of information returned in each component of the structure for a single process is presented below.

<i>fpid</i>	the process' frequency-based scheduler process identifier.
<i>lastcyc_tm</i>	the amount of time that the process has spent running from the last time that it was awakened by the scheduler until it called <code>fbwait</code> .
<i>tot_cycles</i>	the number of times that the process has been awakened by the scheduler (total iterations, or cycles).
<i>tot_cycles_tm</i>	the number of seconds that the process has spent running in all cycles.
<i>overruns</i>	the number of hard frame overruns caused by the process.
<i>mincyc_tm</i>	the least amount of time that the process has spent running in a cycle (minimum cycle time).
<i>mincyc_cycle</i>	the number of the minor cycle in which the minimum cycle time has occurred (minimum cycle cycle).
<i>mincyc_frame</i>	the number of the major frame in which the minimum cycle time has occurred (minimum cycle frame).
<i>maxcyc_tm</i>	the greatest amount of time that the process has spent running in a cycle (maximum cycle time).
<i>maxcyc_cycle</i>	the number of the minor cycle in which the maximum cycle time has occurred (maximum cycle cycle).
<i>maxcyc_frame</i>	the number of the major frame in which the maximum cycle time has occurred (maximum cycle frame).
<i>minframe_tm</i>	the least amount of time that the process has spent running during a major frame (minimum frame time).
<i>minframe</i>	the number of the major frame in which the minimum frame time has occurred (minimum frame frame).
<i>maxframe_tm</i>	the greatest amount of time that the process has spent running during a major frame (maximum frame time).
<i>maxframe</i>	the number of the major frame in which the maximum frame time has occurred (maximum frame frame).

buf_cnt an integer value indicating the number of structures in the array to which *pm_buf* points.

Return Value

A return value of 0 indicates that the call has been successful. A return value of -1 indicates that an error has occurred; `errno` is set to indicate the error. Refer to the **pmqrycpu (3)** man page for a listing of the types of errors that may occur.

pmqrylist – Query Values for a List of Processes

This routine gets performance monitor values for a list of processes scheduled on a frequency-based scheduler.

Note that all time related units are in microseconds (usecs).

Synopsis

```
#include <fbsched.h>

int pmqrylist(fbs_id, pm_buf, buf_cnt)
int fbs_id, buf_cnt;
struct pmqry_ds {
    int fpid;
    struct timespec lastcyc_tm;
    int tot_cycles;
    struct timespec tot_cycles_tm;
    int overruns;
    struct timespec mincyc_tm;
    int mincyc_cycle;
    int mincyc_frame;
    struct timespec maxcyc_tm;
    int maxcyc_cycle;
    int maxcyc_frame;
    struct timespec minframe_tm;
    int minframe;
    struct timespec maxframe_tm;
    int maxframe;
} *pm_buf;
```

Parameters

fbs_id a unique positive integer value representing the identifier for a frequency-based scheduler. This value can be obtained using **fbsid(3)** (page 6-16) or **fbsconfigure(3)** (page 6-8).

Enter MY_FBS to reference the frequency-based scheduler on which the calling process is scheduled without knowing the identifier.

pm_buf a pointer to an array of pmqry_ds structures to which **pmqrylist** returns the performance monitor values for a list of frequency-based scheduled processes.

The list of processes for which values are returned is created by placing the frequency-based scheduler identifier in the *fpid* component of each structure in the array. The type of information contained in each component of the structure for a single process is presented below.

fpid the process' frequency-based scheduler process identifier.

lastcyc_tm the amount of time that the process has spent running from the last time that it was awakened by the scheduler until it called **fbswait**.

<i>tot_cycles</i>	the number of times that the process has been awakened by the scheduler (total iterations, or cycles).
<i>tot_cycles_tm</i>	the number of seconds that the process has spent running in all cycles.
<i>overruns</i>	the number of hard frame overruns caused by the process.
<i>mincyc_tm</i>	the least amount of time that the process has spent running in a cycle (minimum cycle time).
<i>mincyc_cycle</i>	the number of the minor cycle in which the minimum cycle time has occurred (minimum cycle cycle).
<i>mincyc_frame</i>	the number of the major frame in which the minimum cycle time has occurred (minimum cycle frame).
<i>maxcyc_tm</i>	the greatest amount of time that the process has spent running in a cycle (maximum cycle time).
<i>maxcyc_cycle</i>	the number of the minor cycle in which the maximum cycle time has occurred (maximum cycle cycle).
<i>maxcyc_frame</i>	the number of the major frame in which the maximum cycle time has occurred (maximum cycle frame).
<i>minframe_tm</i>	the least amount of time that the process has spent running during a major frame (minimum frame time).
<i>minframe</i>	the number of the major frame in which the minimum frame time has occurred (minimum frame frame).
<i>maxframe_tm</i>	the greatest amount of time that the process has spent running during a major frame (maximum frame time).
<i>maxframe</i>	the number of the major frame in which the maximum frame time has occurred (maximum frame frame).
<i>buf_cnt</i>	an integer value indicating the number of structures in the array to which <i>pm_buf</i> points.

Return Value

A return value of 0 indicates that the call has been successful. A return value of -1 indicates that an error has occurred; `errno` is set to indicate the error. Refer to the `pmqrylist(3)` man page for a listing of the types of errors that may occur.

pmqrypgm – Query Values for a Selected Process

This routine gets performance monitor values for a particular process scheduled on a frequency-based scheduler. Identify the process by using one of the following methods:

- Specify the name of the process and the CPU(s) on which it is scheduled.
- Specify the process' frequency-based scheduler process identifier.
- Specify the name of the process, the CPU(s) on which it is scheduled, and its frequency-based scheduler process identifier.

Synopsis

```
#include <fbsched.h>

int pmqrypgm(fbs_id, name, cpu, pm_buf)
int fbs_id, cpu;
char *name;
struct pmqry_ds {
    int fpid;
    struct timespec lastcyc_tm;
    int tot_cycles;
    struct timespec tot_cycles_tm;
    int overruns;
    struct timespec mincyc_tm;
    int mincyc_cycle;
    int mincyc_frame;
    struct timespec maxcyc_tm;
    int maxcyc_cycle;
    int maxcyc_frame;
    struct timespec minframe_tm;
    int minframe;
    struct timespec maxframe_tm;
    int maxframe;
} *pm_buf;
```

Parameters

<i>fbs_id</i>	a unique positive integer value representing the identifier for a frequency-based scheduler. This value can be obtained using fbsid(3) (page 6-16) or fbsconfigure(3) (page 6-8). Enter MY_FBS to reference the frequency-based scheduler on which the calling process is scheduled without knowing the identifier.
<i>name</i>	a pointer to a variable that contains a path name identifying the process for which performance monitoring values are to be returned. A full or relative path name of up to 1024 characters can be specified. If this variable is the null string, you must provide the frequency-based scheduler process identifier in the <i>fpid</i> component of the structure to which <i>pm_buf</i> points.
<i>cpu</i>	an integer value indicating the processor(s) to be used in conjunction with the value of the <i>name</i> parameter to identify the process for which

performance monitoring values are to be returned. Acceptable values and corresponding results are presented below.

0 the first process named by *name* currently running on the processor from which the call is made is specified.

-1 the first process named by *name* currently running on any processor is specified.

bit mask if (*cpu* & (1<<*i*)) is set (where *i* is an integer ranging from 0 to 15 and representing a CPU) and it is the only bit set, the first process named by *name* currently running on CPU *i* is specified.

If (*cpu* & (1<<*i*)) is set and it is not the only bit set, the first process named by *name* currently running on any of the selected CPUs is specified.

pm_buf a pointer to a `pmqry_ds` structure to which `pmqrypgm` returns the performance monitor values for the frequency-based scheduled process pointed to by the *name* parameter.

The type of information contained in each component of the structure is presented below:

fpid an integer value providing the unique frequency-based scheduler process identifier for which performance monitor values are to be returned.

Get this value by calling `sched_pgmadd` (see page 6-38). This value must be -1 if you wish to identify the process only by specifying *name* and *cpu*.

lastcyc_tm the amount of time (in usecs) that the process has spent running from the last time that it was awakened by the scheduler until it called `fbwait`.

tot_cycles the number of times that the process has been awakened by the scheduler since the last time that performance monitor values have been cleared and performance monitoring has been enabled (total iterations, or cycles).

tot_cycles_tm the number of seconds that the process has spent running in all cycles.

overruns the number of hard frame overruns caused by the process.

mincyc_tm the least amount of time (in usecs) that the process has spent running in a cycle (minimum cycle time).

mincyc_cycle the number of the minor cycle in which the minimum cycle time has occurred (minimum cycle cycle).

mincyc_frame the number of the major frame in which the minimum cycle time has occurred (minimum cycle frame).

<i>maxcyc_tm</i>	the greatest amount of time (in usecs) that the process has spent running in a cycle (maximum cycle time).
<i>maxcyc_cycle</i>	the number of the minor cycle in which the maximum cycle time has occurred (maximum cycle cycle).
<i>maxcyc_frame</i>	the number of the major frame in which the maximum cycle time has occurred (maximum cycle frame).
<i>minframe_tm</i>	the least amount of time (in usecs) that the process has spent running during a major frame (minimum frame time).
<i>minframe</i>	the number of the major frame in which the minimum frame time has occurred (minimum frame frame).
<i>maxframe_tm</i>	the greatest amount of time (in usecs) that the process has spent running during a major frame (maximum frame time).
<i>maxframe</i>	the number of the major frame in which the maximum frame time has occurred (maximum frame frame).

Return Value

A return value of 0 indicates that the call has been successful. A return value of -1 indicates that an error has occurred; `errno` is set to indicate the error. Refer to the **pmqrypgm(3)** man page for a listing of the types of errors that may occur.

pmqrytimer – Query Performance Monitor Mode

This routine determines whether performance monitor timing values include or exclude time spent servicing interrupts.

Synopsis

```
#include <fbsched.h>
```

```
int pmqrytimer()
```

Return Value

A return value of 0 indicates that interrupt time is excluded from performance monitor timing values. A return value of 1 indicates that interrupt time is included in timing values. A return value of -1 indicates that an error has occurred; `errno` is set to indicate the error. Refer to the `pmqrytimer(3)` man page for a listing of the types of errors that may occur.

pmselect – Select Performance Monitor Mode

This routine selects the timing mode under which the Performance Monitor is to run. The timing mode can be set to include or exclude time spent servicing interrupts.

When interrupt time is included, a process' user and system times will total the elapsed time which accrues when the process is the currently running process, including all time spent servicing interrupts. The time spent servicing interrupts is added to the process' system time.

When excluding interrupt time, a process' user and system times will total the time which accrues when the process is the currently running process, excluding all time spent servicing interrupts.

Synopsis

```
#include <fbsched.h>
```

```
int pmselect(mode)  
int mode;
```

Parameters

mode an integer value indicating whether time spent servicing interrupts is to be included in or excluded from performance monitor timing values.

A nonzero value indicates that interrupt time is to be included. A value of zero indicates that interrupt time is to be excluded.

Return Value

A return value of 0 indicates that the call has been successful. A return value of -1 indicates that an error has occurred; `errno` is set to indicate the error. Refer to the **pmselect(3)** man page for a listing of the types of errors that may occur.

The FORTRAN Library Interface

Overview	7-1
Compiling and Linking Procedures	7-1
Frequency-Based Scheduler Routines	7-2
Routine Summary	7-2
FORTRAN Library Call Sequence	7-4
Using Frequency-Based Scheduler Routines	7-5
fbsaccess – Change Permissions for a Frequency-Based Scheduler	7-5
fbsattach – Attach Timing Source to a Frequency-Based Scheduler	7-6
fbsconfigure – Configure a Frequency-Based Scheduler	7-7
fbscycle – Return Minor Cycle/Major Frame Count	7-9
fbsdetach – Detach Timing Source from a Frequency-Based Scheduler	7-10
fbsgetrtc – Obtain Current Values for Real-Time Clock	7-11
fbsid – Return the FBS Identifier for a Key	7-12
fbsinfo – Return Information for a Frequency-Based Scheduler	7-13
fbsintrpt – Start/Stop/Resume Scheduling on a Frequency-Based Scheduler	7-15
fbsquery – Query Processes on a Frequency-Based Scheduler	7-16
fbsremove – Remove a Frequency-Based Scheduler	7-19
fbsresume – Resume Scheduling on a Frequency-Based Scheduler	7-20
fbsrunrtc – Start/Stop Real-Time Clock	7-22
fbsschedself – Schedule a Process/Thread on a Frequency-Based Scheduler	7-23
fbssetrtc – Set Real-Time Clock	7-25
fbswait – Wait on a Frequency-Based Scheduler	7-26
nametopid – Return the Process ID for a Specified Process Name	7-27
pgmquery – Query a Process on a Frequency-Based Scheduler	7-28
pgmremove – Remove a Process from a Frequency-Based Scheduler	7-30
pgmreschedule – Reschedule a Process	7-32
pgmschedule – Schedule a Process on a Frequency-Based Scheduler	7-35
pgmstat – Query State of FBS-Scheduled Process	7-38
pgmtrigger – Trigger Process Waiting on FBS	7-40
rtparm – Return Initiation Parameter	7-41
sched_pgm_set_soft_overrun_limit – Set Soft Overrun Limit	7-42
sched_pgm_soft_overrun_query – Query Soft Overrun Processing	7-43
schedfbsqry – Query Processes on a Frequency-Based Scheduler	7-44
schedpgmadd – Schedule a Process on a Frequency-Based Scheduler	7-47
schedpgmqry – Query a Process on a Frequency-Based Scheduler	7-49
schedpgmresched – Reschedule a Process	7-52
Performance Monitor Routines	7-55
Routine Summary	7-55
FORTRAN Library Call Sequence	7-55
Using Performance Monitor Routines	7-57
pmclprgm – Clear Values for a Process	7-57
pmclrtable – Clear Values for Processor(s)	7-59
pmmonitor – Start/Stop Performance Monitoring on Processor(s)	7-60
pmprogram – Start/Stop Performance Monitoring on a Process	7-61
pmqrycpu – Query Values for Selected Processor(s)	7-63
pmqrylist – Query Values for a List of Processes	7-65
pmqrypgm – Query Values for a Selected Process	7-67
pmquerytimer – Query Performance Monitor Mode	7-70
pmselect – Select Performance Monitor Mode	7-71

The FORTRAN Library Interface

This chapter describes the FORTRAN library interface to the Frequency-Based Scheduler and the Performance Monitor. Library information, call sequences for using the scheduler and the Performance Monitor, and details for each of the library routines are provided.

Overview

Access to the functions associated with the Frequency-Based Scheduler and the Performance Monitor is provided through libraries of routines that can be called from application programs written in FORTRAN.

The following information is provided in this chapter for each routine:

- a description of the routine
- the FORTRAN variable declarations and call statement needed to reference the routine in an application program
- detailed descriptions of each parameter

Information about the individual routines, including an illustration of the sequence in which you would call the routines during frequency-based scheduling, begins on page 7-2. The same information is provided for performance monitoring beginning on page 7-55.

For compatibility with the PowerMAX operating system, **usermap (3f)** is provided to map a target process' memory into address space. Refer to the man page for details.

Compiling and Linking Procedures

When statically linking a FORTRAN program, the following libraries are required:

```
/usr/lib/libccur_fbsched.a  
/usr/lib/libccur_rt.a  
/usr/lib/libF77rt.a
```

When dynamically linking a C program, the following libraries are used:

```
/usr/lib/libccur_fbsched.so  
/usr/lib/libccur_rt.so  
/usr/lib/libF77rt.so
```

To compile and link a FORTRAN program, the command line instruction is as follows:

```
cf77 [options ...] file.f -lF77rt -lccur_fbsched -lccur_rt
```

Frequency-Based Scheduler Routines

Frequency-Based Scheduler routines provide access to the key features of the scheduler. They enable you to perform such basic operations as:

- configuring a scheduler
- scheduling programs on a scheduler
- setting up and connecting a timing source to a scheduler
- starting, stopping and resuming scheduling on a scheduler
- getting information about scheduled processes
- rescheduling and removing scheduled processes
- disconnecting a timing source
- removing a scheduler

Routine Summary

Frequency-Based Scheduler routines are summarized in Table 7-1. Complete information about each routine is provided under the section “Using Frequency-Based Scheduler Routines.”

Table 7-1. Frequency-Based Scheduler FORTRAN Library Routines

Routine	Page	Description
fbsaccess	7-5	Change permissions for a frequency-based scheduler
fbsattach	7-6	Attach timing source to a frequency-based scheduler
fbsconfigure	7-7	Configure a frequency-based scheduler
fbscycle	7-9	Return minor cycle/major frame count
fbsdetach	7-10	Detach timing source from a frequency-based scheduler
fbsgetrtc	7-11	Get real-time clock values
fbsid	7-12	Return the frequency-based scheduler identifier for a key
fbsinfo	7-13	Return information for a frequency-based scheduler
fbsintrpt	7-15	Start/stop/resume scheduling
fbsquery	7-16	Query processes on a frequency-based scheduler
fbsremove	7-19	Remove a frequency-based scheduler
<i>(continued on next page)</i>		

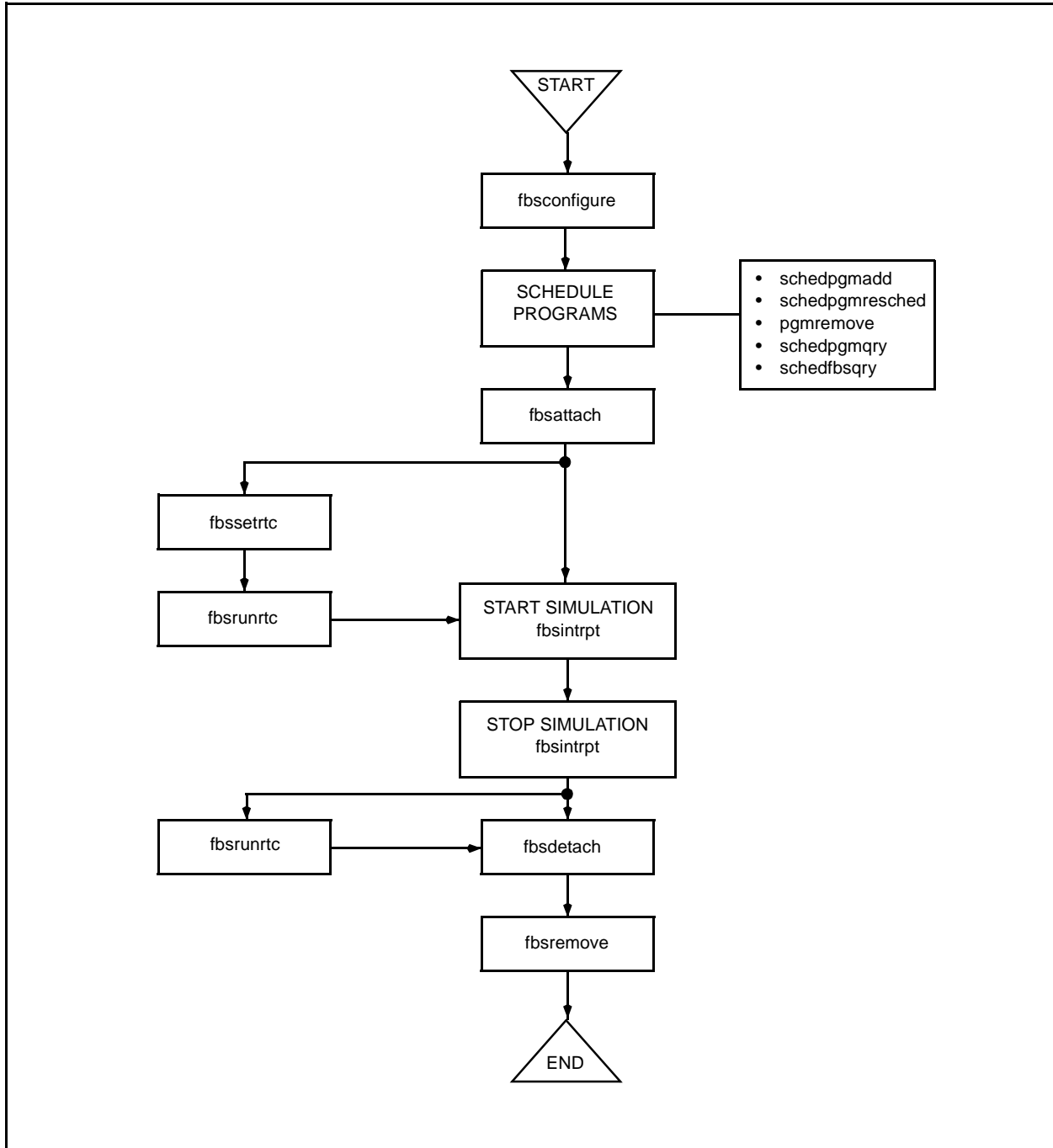
Table 7-1. Frequency-Based Scheduler FORTRAN Library Routines (Cont.)

Routine	Page	Description
fbsresume	7-20	Resume scheduling on a frequency-based scheduler
fbsrunrtc	7-22	Start/stop real-time clock
fbsschedself	7-23	Add a calling process to a frequency-based scheduler
fbssetrtc	7-25	Set real-time clock
fbswait	7-26	Wait on a frequency-based scheduler
nametopid	7-27	Return the process ID
pgmquery	7-28	Query processes on a frequency-based scheduler
pgmremove	7-30	Remove a process from a frequency-based scheduler
pgmreschedule	7-32	Reschedule a process
pgmschedule	7-35	Schedule a process on a frequency-based scheduler
pgmstat	7-38	Query the state of an FBS-scheduled process
pgmtrigger	7-40	Trigger a process on a frequency-based scheduler
rtparm	7-41	Return initiation parameter
sched_pgm_set_soft_ouerrun_limit	7-42	Set soft overrun limit
sched_pgm_soft_ouerrun_query	7-43	Query soft overrun processing
schedfbsqry	7-44	Query processes on a frequency-based scheduler
schedpgmadd	7-47	Schedule a process on a frequency-based scheduler
schedpgmqry	7-49	Query a process
schedpgmresched	7-52	Reschedule a process

FORTRAN Library Call Sequence

The approximate order in which you might call the routines from an application program is illustrated in Figure 7-1.

Figure 7-1. FORTRAN Library Call Sequence: Frequency-Based Scheduler



Using Frequency-Based Scheduler Routines

In the sections that follow, the Frequency-Based Scheduler routines contained in the `libF77rt` library are presented in alphabetical order.

fbsaccess – Change Permissions for a Frequency-Based Scheduler

This subroutine changes the permissions assigned for a selected frequency-based scheduler. Permissions can be changed only by a process that has an effective user ID that is equal to that of the owner/creator of the frequency-based scheduler.

Call Statement

```
call fbsaccess (schdle, uid, gid, permissions, istat)
integer schdle, uid, gid, permissions, istat
```

Parameters

<i>schdle</i>	a unique, positive integer value representing the identifier for a frequency-based scheduler. You can obtain this value by making a call to fbsconfigure(3f) (see page 7-7 for an explanation of this subroutine) or fbsid(3f) (see page 7-12). To reference the frequency-based scheduler on which the calling process is scheduled without knowing the identifier, you can specify the value <code>-1</code> .
<i>uid</i>	an integer value representing the effective user ID of the specified frequency-based scheduler.
<i>gid</i>	an integer value representing the effective group ID of the specified frequency-based scheduler.
<i>permissions</i>	a bit pattern used to set the permissions associated with the specified frequency-based scheduler. Bit patterns and corresponding permissions are presented in Table 7-2.

Table 7-2. Frequency-Based Scheduler Permissions

Bit Pattern	Permissions
400	Read by user
200	Alter by user
060	Read, alter by group
006	Read, alter by others

<i>istat</i>	an integer value indicating whether or not an error has occurred. A value of zero indicates that no error has occurred. A nonzero value indicates that an error of a specific type has occurred. Refer to the fbsaccess(3f) man page for a listing of the nonzero values that may be returned and the types of errors that they represent.
--------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

fbsattach – Attach Timing Source to a Frequency-Based Scheduler

This subroutine is invoked to attach a timing source to a frequency-based scheduler or to specify end-of-cycle scheduling. The timing source can be a real-time clock or an edge-triggered interrupt device.

Call Statement

```
call fbsattach(schdle, cpu, devname, istat)  
integer schdle, cpu, istat  
character* (*) devname
```

Parameters

schdle a unique, positive integer value representing the identifier for the frequency-based scheduler for which the timing source is to be attached or end-of-cycle scheduling specified. You can obtain this value by making a call to **fbsconfigure(3f)** (see page 7-7 for an explanation of this subroutine) or **fbsid(3f)** (see page 7-12). If you wish to reference the frequency-based scheduler on which the calling process is scheduled without knowing its identifier, you can specify a value of **-1**.

cpu a variable that must contain the value **0**.

devname a null string or the path name of the device that is to be used as the timing source for the specified scheduler. If *devname* contains a null string, end-of-cycle scheduling is specified; that is, execution of the processes in the next minor cycle will occur when the last process scheduled to execute in the current minor cycle finishes its execution for that cycle.

If *devname* contains a path name, it may refer to a real-time clock or an edge-triggered interrupt. See Chapter 3 for information about timing source device files.

istat an integer value indicating whether or not an error has occurred. A value of zero indicates that no error has occurred. A nonzero value indicates that an error of a specific type has occurred. Refer to the **fbsattach(3f)** man page for a listing of the nonzero values that may be returned and the types of errors that they represent.

fbsconfigure – Configure a Frequency-Based Scheduler

This subroutine is invoked to configure a frequency-based scheduler or to obtain configuration details for a frequency-based scheduler that has already been configured. Note that to configure a scheduler, the calling process must have the capabilities associated with the *fbscheduler* role (for additional information on privileges, refer to Chapter 1).

If you wish to configure a scheduler, you must specify a *key*, which is a user-chosen numeric identifier for a frequency-based scheduler. The value for *cycles* indicates the function to be performed:

- If the value specified in *cycles* is NOT equal to 0, and if the number of existing frequency-based schedulers is less than the system imposed limit (default = 10) a frequency-based scheduler is created and an identifier is returned.
- If the value of *cycles* is equal to 0, the current configuration for the frequency-based scheduler is returned.

Call Statement

```
call fbsconfigure (key, cycles, progs, max, reset, configflg, schdle, istat)
integer key, cycles, progs, max, reset, configflg, schdle, istat
```

Parameters

To create a frequency-based scheduler, you must specify the following parameters as described.

<i>key</i>	an integer value identifying the frequency-based scheduler that is to be created.						
<i>cycles</i>	an integer value indicating the number of minor cycles that compose a frame on the specified scheduler.						
<i>progs</i>	an integer value indicating the maximum number of programs that can be scheduled to execute during one minor cycle.						
<i>max</i>	an integer value indicating the maximum number of programs that can be scheduled on the specified scheduler at one time. This value must be less than or equal to the <u>product</u> that is obtained by multiplying the values specified for the <i>cycles</i> and <i>progs</i> parameters.						
<i>reset</i>	an integer value indicating whether or not processes currently scheduled on the specified scheduler are to be killed before the scheduler is reconfigured. Acceptable values and corresponding results are as follows: <table style="margin-left: 2em;"> <tbody> <tr> <td><0</td> <td>Kill and remove all processes currently scheduled on the specified scheduler</td> </tr> <tr> <td>0</td> <td>Ignore all processes currently scheduled on the specified scheduler</td> </tr> <tr> <td>>0</td> <td>Remove all processes currently scheduled on the specified scheduler</td> </tr> </tbody> </table>	<0	Kill and remove all processes currently scheduled on the specified scheduler	0	Ignore all processes currently scheduled on the specified scheduler	>0	Remove all processes currently scheduled on the specified scheduler
<0	Kill and remove all processes currently scheduled on the specified scheduler						
0	Ignore all processes currently scheduled on the specified scheduler						
>0	Remove all processes currently scheduled on the specified scheduler						

<i>configflg</i>	an integer value indicating the permissions assigned to the specified scheduler.
<i>schdle</i>	a unique, positive integer value representing the identifier for the specified frequency-based scheduler. It is important to note that this identifier is required by most of the library subroutines for the FBS and the performance monitor.
<i>istat</i>	an integer value indicating whether or not an error has occurred. A value of zero indicates that no error has occurred. A nonzero value indicates that an error of a specific type has occurred. Refer to the fbconfigure(3f) man page for a listing of the nonzero values that may be returned and the types of errors that they represent.

To obtain information for an existing frequency-based scheduler, you must specify the following parameters as described.

<i>key</i>	an integer value identifying the frequency-based scheduler for which configuration information is to be returned. If this value is zero, the frequency-based scheduler identifier associated with this scheduler must also be provided by using the <i>schdle</i> parameter.
<i>cycles</i>	the integer value zero, indicating that current configuration information for the specified scheduler is to be returned. Fbsconfigure will <u>return</u> to this variable an integer value indicating the number of minor cycles that compose a frame on the specified scheduler.
<i>progs</i>	the maximum number of programs that can be scheduled to run during one minor cycle on the specified scheduler.
<i>max</i>	the maximum number of programs that can be scheduled on the specified scheduler at one time.
<i>configflg</i>	the permissions assigned to the specified scheduler.
<i>schdle</i>	a unique, positive integer value representing the identifier for the specified frequency-based scheduler. If you specify a key of 0, this variable must contain the related frequency-based scheduler identifier.

fbcycle – Return Minor Cycle/Major Frame Count

This subroutine is invoked to obtain the current minor cycle and major frame count values for a frequency-based scheduler. These values enable you to determine the progress of a simulation.

Call Statement

```
call fbcycle(schdle, count, istat)
integer schdle, count(2), istat
```

Parameters

<i>schdle</i>	a unique, positive integer value representing the identifier for the frequency-based scheduler for which you wish to obtain the current cycle and frame counts. You can obtain this value by making a call to fbsconfigure (see page 7-7 for an explanation of this subroutine) or fbsid(3f) (see page 7-12). If you wish to reference the frequency-based scheduler on which the calling process is scheduled without knowing its identifier, you can specify a value of -1.
<i>count</i>	an array containing integer values indicating the current minor cycle and major frame for the specified scheduler. <i>Count(1)</i> will contain the current minor cycle. <i>Count(2)</i> will contain the major frame count.
<i>istat</i>	an integer value indicating whether or not an error has occurred. A value of zero indicates that no error has occurred. A nonzero value indicates that an error of a specific type has occurred. Refer to the fbcycle(3f) man page for a listing of the nonzero values that may be returned and the types of errors that they represent.

fbsdetach – Detach Timing Source from a Frequency-Based Scheduler

This subroutine is invoked to detach the currently attached timing source from a frequency-based scheduler or to disable end-of-cycle scheduling. If the timing source is a real-time clock, it is recommended that you stop the clock prior to invoking this subroutine. You can do so by making a call to **fbsrunrtc** (see page 7-22 for an explanation of this subroutine).

Call Statement

```
call fbsdetach(schdle, istat)  
integer schdle, istat
```

Parameters

<i>schdle</i>	a unique, positive integer value representing the identifier for the frequency-based scheduler from which you wish to detach the currently attached timing source or for which you wish to disable end-of-cycle scheduling. You can obtain this value by making a call to fbsconfigure(3f) (see page 7-7 for an explanation of this subroutine) or fbsid(3f) (see page 7-12). If you wish to reference the frequency-based scheduler on which the calling process is scheduled without knowing its identifier, you can specify a value of -1 .
<i>istat</i>	an integer value indicating whether or not an error has occurred. A value of zero indicates that no error has occurred. A nonzero value indicates that an error of a specific type has occurred. Refer to the fbsdetach(3f) man page for a listing of the nonzero values that may be returned and the types of errors that they represent.

fbgetrtc – Obtain Current Values for Real-Time Clock

This subroutine is invoked to obtain the current count and resolution values for the real-time clock that is attached to a specified frequency-based scheduler.

Call Statement

```
call fbgetrtc(schdle, count, resolution, istat1, istat2)
integer schdle, count, resolution, istat1, istat2
```

Parameters

Parameters must be specified in the order indicated. They are described as follows.

<i>schdle</i>	a unique, positive integer value representing the identifier for the frequency-based scheduler to which the real-time clock is attached. You can obtain this value by making a call to fbconfigure(3f) (see page 7-7 for an explanation of this subroutine) or fbid(3f) (see page 7-12). If you wish to reference the frequency-based scheduler on which the calling process is scheduled without knowing its identifier, you can specify a value of -1 .
<i>count</i>	an integer value indicating the current number of clock counts per minor cycle. This value can range from one to 65535.
<i>resolution</i>	an integer value indicating the current duration in microseconds of one clock count. This value will be one of the following: 1, 10, 100, 1000, or 10000.
<i>istat1</i>	an integer value indicating whether or not an error has occurred. A value of zero indicates that no error has occurred. A nonzero value indicates that an error of a specific type has occurred. Refer to the fbgetrtc(3f) man page for a listing of the nonzero values that may be returned and the types of errors that they represent. If <i>istat1</i> contains a value indicating that an error has occurred on an open or ioctl call, the error status of that call is returned in <i>istat2</i> .
<i>istat2</i>	the error status of an open or ioctl call. See the include file <errno.h> for a description of the errors.

fbsid – Return the FBS Identifier for a Key

This subroutine is invoked to obtain the frequency-based scheduler identifier associated with a particular user-specified key. The key must match the key that was specified when the scheduler was created by making a call to **fbsconfigure(3f)**.

Call Statement

```
call fbsid(key, schdle, istat)  
integer key, schdle, istat
```

Parameters

Parameters must be specified in the order indicated. They are described as follows.

<i>key</i>	an integer value identifying a frequency-based scheduler; this value must be the same value that was specified for <i>key</i> when the scheduler was created by making a call to fbsconfigure (see page 7-7 for an explanation of this subroutine).
<i>schdle</i>	an integer value representing the unique frequency-based scheduler identifier associated with the key.
<i>istat</i>	an integer value indicating whether or not an error has occurred. A value of zero indicates that no error has occurred. A nonzero value indicates that an error of a specific type has occurred. Refer to the fbsid(3f) man page for a listing of the nonzero values that may be returned and the types of errors that they represent.

fbsinfo – Return Information for a Frequency-Based Scheduler

This subroutine is invoked to obtain information that is related to a selected frequency-based scheduler but cannot be obtained by invoking other subroutines (for example, `schedfbsqry`, `schedpgmqry`). Such information includes the following:

- The user and group IDs of the owner and the creator of the scheduler
- The permissions assigned for the scheduler
- The key associated with the scheduler's identifier
- The total number of overruns for all processes on the scheduler
- The CPUs that are active in the system
- The CPUs on which performance monitoring has been enabled
- The FBS-enabled flag
- The path name of the device that has been attached to the scheduler

Call Statement

```
call fbsinfo(schdle, buf, devname, istat)
integer schdle, buf(41), istat
character* (*) devname
```

Parameters

Parameters are described as follows.

- schdle* a unique, positive integer value representing the identifier for a frequency-based scheduler. You can obtain this value by making a call to `fbsconfigure(3f)` (see page 7-7 for an explanation of this subroutine) or `fbsid(3f)` (see page 7-12). If you wish to reference the frequency-based scheduler on which the calling process is scheduled without knowing its identifier, you can specify a value of `-1`.
- buf* an array containing information about the specified scheduler. The information returned in each element of the array is presented in Table 7-3.

Table 7-3. Contents of Array Elements: fbsinfo

Element	Contents
buf(1)	owner's user ID
buf(2)	owner's group ID
buf(3)	creator's user ID
buf(4)	creator's group ID
buf(5)	access modes
buf(6)	key
buf(7)	flags word
buf(8)	reserved for future use

Table 7-3. Contents of Array Elements: fbsinfo (Cont.)

Element	Contents
buf(9)	total number of hard overruns for all processes on the scheduler
buf(10)	mask of CPUs active in the system
buf(11)	mask of CPUs on which performance monitoring has been enabled
buf(12)	FBS-enabled flag
buf(13)–(41)	reserved for future use

devname the path name of the device that is being used as the timing source for the specified frequency-based scheduler. If end-of-cycle scheduling has been specified, *devname* will contain a null string.

istat an integer value indicating whether or not an error has occurred. A value of zero indicates that no error has occurred. A nonzero value indicates that an error of a specific type has occurred. Refer to the **fbsinfo(3f)** man page for a listing of the nonzero values that may be returned and the types of errors that they represent.

fbsintrpt – Start/Stop/Resume Scheduling on a Frequency-Based Scheduler

This subroutine is invoked to start, stop, or resume scheduling on a frequency-based scheduler. If you invoke this subroutine to start scheduling, the minor cycle, major frame, and overrun count values are reset. If you invoke it to resume scheduling, these values are not reset.

Prior to invoking **fbsintrpt**, you must have invoked **fbsattach** to specify end-of-cycle scheduling or attach a timing source to the frequency-based scheduler on which you are starting scheduling (see page 7-6 for an explanation of **fbsattach**). If you have specified a real-time clock as the timing source, scheduling will not begin until you have set and started the clock (see pages 7-25 and 7-22 for explanations of **fbssetrtc** and **fbsrunrtc**, respectively). If you have specified an edge-triggered interrupt device as the timing source, it must already be generating interrupts in order for scheduling to start.

Call Statement

```
CALL fbsintrpt(schdle, intrflag, istat)
integer schdle, intrflag, istat
```

Parameters

<i>schdle</i>	a unique, positive integer value representing the identifier for the frequency-based scheduler on which you wish to start, stop, or resume scheduling of processes. You can obtain this value by making a call to fbsconfigure(3f) (see page 7-7 for an explanation of this subroutine) or fbsid(3f) (see page 7-12). If you wish to reference the frequency-based scheduler on which the calling process is scheduled without knowing its identifier, you can specify a value of -1 .
<i>intrflag</i>	an integer value indicating whether scheduling of processes on the specified scheduler is to be started, stopped, or resumed. Acceptable values and results are as follows: <ul style="list-style-type: none"> <0 Start scheduling of processes with the initial frame, cycle, and overrun count values set to zero 0 Stop scheduling of processes, and save the count values for the current frame and cycle >0 Resume scheduling of processes with the frame, cycle, and overrun count values set to the values that were saved when the scheduler was last stopped
<i>istat</i>	an integer value indicating whether or not an error has occurred. A value of zero indicates that no error has occurred. A nonzero value indicates that an error of a specific type has occurred. Refer to the fbsintrpt(3f) man page for a listing of the nonzero values that may be returned and the types of errors that they represent.

fbsquery – Query Processes on a Frequency-Based Scheduler

CAUTION

This interface is obsolete. It is maintained for compatibility with CX/UX, but it returns processes' scheduling priorities without any indication of the scheduling policies with which they are associated. If you have an existing application that uses this interface, it is recommended that you change your application to use **schdfbsqry (3f)** (see p. 7-44).

This subroutine is invoked to obtain information about processes that have been scheduled on a frequency-based scheduler. Information is returned for all processes scheduled on the user-specified processor(s). Information provided for each process includes the following:

- A mask of the CPU(s) on which the process can execute
- The frequency-based scheduler process identifier
- The scheduling priority
- The period (the number of minor cycles indicating the frequency with which the process is wakened in each major frame)
- The starting base cycle (the first minor cycle in which the process is scheduled to be wakened in each major frame)
- The value of the "halt on overrun" flag

Call Statement

```
CALL fbsquery (schdle, cpu, buf1size, buf1, maxsize, buf2size, buf2, istat)
integer schdle, cpu, buf1size, buf1(buf1size), maxsize, buf2size, istat
character* (*) buf2
```

Parameters

<i>schdle</i>	a unique, positive integer value representing the identifier for the frequency-based scheduler for which you wish to obtain scheduling information. You can obtain this value by making a call to fbsconfigure (3f) (see page 7-7 for an explanation of this subroutine) or fbsid (3f) (see page 7-12). If you wish to reference the frequency-based scheduler on which the calling process is scheduled without knowing its identifier, you can specify a value of -1 .
<i>cpu</i>	an integer value indicating the processor(s) for which scheduling information is to be obtained. Acceptable values and corresponding results are as follows: 0 Scheduling information for processes executing on the processor from which the call is made is returned -1 Scheduling information for all processes on the scheduler is returned

Bit mask If $(cpu \& (1 \ll i))$ is set (where i is an integer ranging from zero to 15 and representing a CPU), scheduling information for processes executing on CPU i is returned

buf1size an integer value indicating the size in 32-bit words of the array represented by *buf1*. Because 10 words of information are returned for each process, it is recommended that this value be a multiple of 10.

buf1 an array containing a series of 10 integer values for each process on the processor(s) specified with the *cpu* parameter. The number of processes for which these values are returned is bound by the value of the *buf1size* parameter. If, for example, the value of *buf1size* is 145, values for 14 processes will be returned. These values represent the scheduling information for the process(es). The type of information returned in each array element for a single process is presented in Table 7-4.

Table 7-4. Contents of Array Elements: fbsquery

Element	Contents
1	Byte offset of the process' path name in <i>buf2</i>
2	Length in bytes of the process' path name
3	Zero
4	Zero
5	Mask of the CPU(s) on which the process can execute
6	The process' frequency-based scheduler process identifier
7	The process' scheduling priority
8	The number of minor cycles indicating the frequency with which the process is to be wakened in each major frame (period)
9	The first minor cycle in which the process is scheduled to be wakened in each major frame (starting base cycle)
10	The value of the "halt on overrun" flag. A nonzero value indicates that the flag is set. A value of zero indicates that the flag is not set.

maxsize an integer value indicating the maximum length of a path name to be returned in *buf2*

buf2size an integer value indicating the size in bytes of the character string represented by *buf2*. To ensure that *buf2* is large enough to accommodate the names of all processes that you wish to query, you may find it helpful to compute the number of bytes needed by multiplying the maximum number of processes allowed on the scheduler (see the information on **fbsconfigure** presented on page 7-7) by 32.

<i>buf2</i>	the path names for each process on the processor(s) specified with the <i>cpu</i> parameter. Path names are returned as a series of strings. The length of each string is less than or equal to the value of <i>maxsize</i> . Where <i>maxsize</i> is not large enough to accommodate a full path name, the concluding component names are returned. The number of path names returned is bound by the value of the <i>buf2size</i> parameter.
<i>istat</i>	an integer value indicating whether or not an error has occurred. A value of zero indicates that no error has occurred. A nonzero value indicates that an error of a specific type has occurred. Refer to the fbsquery(3f) man page for a listing of the nonzero values that may be returned and the types of errors that they represent.

fbsremove – Remove a Frequency-Based Scheduler

This subroutine is invoked to remove a frequency-based scheduler and to free the data structure associated with it. It is important to note that prior to invoking **fbsremove**, you must ensure that the timing source is detached from the scheduler or that end-of-cycle scheduling is disabled (see page 7-10 for information on the use of **fbsdetch**). It is important to note that **fbsremove** will remove all processes scheduled on the specified scheduler. It is recommended, however, that you remove all scheduled processes prior to invoking **fbsremove**. You can do so by making a call to **pgmremove** (see page 7-30 for information on the use of this subroutine).

Note that to remove a frequency-based scheduler, the calling process must have an effective user ID that is equal to that of the owner/creator of the scheduler.

Call Statement

```
call fbsremove (schdle, ab, istat)
integer schdle, ab, istat
```

Parameters

<i>schdle</i>	a unique, positive integer value representing the identifier for the frequency-based scheduler that you wish to remove. You can obtain this value by making a call to fbsconfigure(3f) (see page 7-7 for an explanation of this subroutine) or fbsid(3f) (see page 7-12). If you wish to reference the frequency-based scheduler on which the calling process is scheduled without knowing its identifier, you can specify a value of -1 .
<i>ab</i>	an integer value indicating the manner in which processes scheduled on the scheduler are to be handled. Acceptable values and corresponding results are as follows: <ul style="list-style-type: none"> <0 Kill and remove all processes currently scheduled on the specified scheduler ≥0 Remove all processes currently scheduled on the specified scheduler
<i>istat</i>	an integer value indicating whether or not an error has occurred. A value of zero indicates that no error has occurred. A nonzero value indicates that an error of a specific type has occurred. Refer to the fbsremove(3f) man page for a listing of the nonzero values that may be returned and the types of errors that they represent.

fbsresume – Resume Scheduling on a Frequency-Based Scheduler

The **fbsresume** subroutine is invoked to resume scheduling of processes on a frequency-based scheduler at the specified minor cycle, major frame, and overrun count.

Note that to resume scheduling of processes on a frequency-based scheduler, the calling process must have alter permission for the scheduler.

If you wish to resume scheduling of processes on a frequency-based scheduler without altering the scheduler's current frame, cycle, and overrun values, it is recommended that you use the **fbsintrpt (3f)** subroutine (see page 7-15 for an explanation of this subroutine).

CAUTION

The **fbsresume** subroutine clears performance monitor values for all processes scheduled on the specified scheduler. Changing the frame and cycle count for the scheduler causes the values that are being maintained by the performance monitor to be inaccurate.

Call Statement

```
call fbsresume (schdle, frame, cycle, overruns, istat)  
integer schdle, frame, cycle, overruns, istat
```

Parameters

<i>schdle</i>	a unique, positive integer value representing the identifier for the frequency-based scheduler on which you wish to resume scheduling of processes. You can obtain this value by making a call to fbsconfigure (3f) (see page 7-7 for an explanation of this subroutine) or fbsid (3f) (see page 7-12). If you wish to reference the frequency-based scheduler on which the calling process is scheduled without knowing the identifier, you can specify the value -1.
<i>frame</i>	an integer value indicating the major frame in which you wish scheduling of processes to be resumed on the specified scheduler
<i>cycle</i>	an integer value indicating the minor cycle in which you wish scheduling of processes to be resumed on the specified scheduler. This value can range from zero to the total number of minor cycles per frame minus one. The total number of minor cycles per frame was specified when the scheduler was created by making a call to fbsconfigure (see page 7-7 for an explanation of this subroutine).
<i>overruns</i>	an integer value indicating the value to which you wish the overrun count to be set when scheduling resumes on the specified scheduler. If you do not wish to change the overrun count, you can specify the value -1.

istat

an integer value indicating whether or not an error has occurred. A value of zero indicates that no error has occurred. A nonzero value indicates that an error of a specific type has occurred. Refer to the **fbsresume(3f)** man page for a listing of the nonzero values that may be returned and the types of errors that they represent.

fbsrunrtc – Start/Stop Real-Time Clock

This subroutine is invoked to start or stop the counting of a real-time clock that has been attached to a frequency-based scheduler.

Call Statement

```
call fbsrunrtc(schdle, runflag, istat1, istat2)  
integer schdle, runflag, istat1, istat2
```

Parameters

<i>schdle</i>	a unique, positive integer value representing the identifier for the frequency-based scheduler for which you wish to start or stop the attached real-time clock. You can obtain this value by making a call to fbsconfigure(3f) (see page 7-7 for an explanation of this subroutine) or fbsid(3f) (see page 7-12). If you wish to reference the frequency-based scheduler on which the calling process is scheduled without knowing its identifier, you can specify a value of -1.
<i>runflag</i>	an integer value indicating whether the real-time clock is to be started or stopped. A nonzero value indicates that the clock is to be started. A zero value indicates that the clock is to be stopped.
<i>istat1</i>	an integer value indicating whether or not an error has occurred. A value of zero indicates that no error has occurred. A nonzero value indicates that an error of a specific type has occurred. Refer to the fbsrunrtc(3f) man page for a listing of the nonzero values that may be returned and the types of errors that they represent. If <i>istat1</i> contains a value indicating that an error has occurred on an open or ioctl call, the error status of that call is returned in <i>istat2</i> .
<i>istat2</i>	the error status of an open or ioctl call. See the include file <errno.h> for a description of the error.

fbsschedself – Schedule a Process/Thread on a Frequency-Based Scheduler

The **fbsschedself** subroutine is invoked to schedule the calling process or thread on a frequency-based scheduler.

It is important to note that **fbsschedself** does not allow a process to set its scheduling policy and priority or its CPU bias. These tasks must be performed prior to invoking **fbsschedself**.

A process can set its scheduling policy and priority by using the **sched_setscheduler(2)** library routine; it can set its CPU bias by using the **mpadvise(3)** library routine or **run(1)** command. Procedures for using these functions are explained in the *RedHawk Linux User's Guide*.

Note that you cannot use this subroutine to add **/idle** to a frequency-based scheduler.

To schedule the calling process on a frequency-based scheduler, it must have alter permission for the scheduler.

You must not change the scheduling policy or priority of a process while it is scheduled on a scheduler by using **sched_setscheduler** or other program interfaces that allow you to change scheduling policy and priority. The frequency-based scheduler is not aware of changes in scheduling policy and priority that are made by using these interfaces.

If you need to change the scheduling policy or priority of a single-threaded FBS-scheduled process, you may do so by using **schedpgramresched** to reschedule it (see page 7-52 for an explanation of this routine).

Call Statement

```
call fbsschedself (schdle, name, sched_buf, istat)
integer schdle, istat
character* (*) name
integer (*) sched_buf
```

Parameters

<i>schdle</i>	a unique, positive integer value representing the identifier for a frequency-based scheduler. You can obtain this value by making a call to fbconfigure(3f) (see page 7-7 for an explanation of this subroutine) or fbid(3f) (see page 7-12). If you wish to reference the frequency-based scheduler on which the calling process is scheduled without knowing the identifier, you can specify the value -1.
<i>name</i>	a standard path name or arbitrary content identifying the program associated with the calling process. A full or relative path name of up to 1023 characters can be specified.
<i>sched_buf</i>	an integer array that contains the scheduling parameters with which you wish to schedule the process. The information that is specified in this array is presented in Table 7-5.
<i>istat</i>	an integer value indicating whether or not an error has occurred. A value of zero indicates that no error has occurred. A nonzero value indicates that an error of a specific type has occurred. Refer to the fbsschedself(3f) man page for a listing of the nonzero values that may be returned and the types of errors that they represent.

Table 7-5. Contents of Array Elements: `fbsssetrtc`

Element	Contents
<code>sched_buf(1)</code>	an integer value indicating the version of <code>sched_buf</code> that is being passed to <code>fbsschedself</code> . Specify the symbolic constant <code>FBSSCHED_BUF_V1</code> , which is defined in <code><fbssched.h></code> for this purpose.
<code>sched_buf(2)</code>	an integer value to be passed to a process that is scheduled on a frequency-based scheduler. This value can be retrieved by the FBS-scheduled process through a call to <code>rtparm</code> (see page 7-41 for an explanation of this subroutine).
<code>sched_buf(3)</code>	<p>an integer value indicating the frequency with which the calling process is to be wakened in each major frame. A period of one indicates that the calling process is to be wakened every minor cycle; a period of two indicates that it is to be wakened once every two minor cycles, a period of three once every three minor cycles, and so on.</p> <p>This value can range from one to the number of minor cycles that compose a frame on the specified scheduler as defined in a call to <code>fbssconfigure</code> (see page 7-7 for an explanation of this subroutine).</p>
<code>sched_buf(4)</code>	an integer value indicating the first minor cycle in which the calling process is scheduled to be wakened in each frame. This value can range from zero to the total number of minor cycles per frame minus one. The total number of minor cycles per frame is specified in a call to <code>fbssconfigure</code> (see page 7-7 for an explanation of this routine).
<code>sched_buf(5)</code>	an integer value indicating whether or not the scheduler should be stopped in the event that the calling process causes a frame overrun. A nonzero value indicates that the scheduler will be stopped.
<code>sched_buf(6)</code>	an integer value that is returned by <code>fbsschedself</code> and is the unique frequency-based scheduler process identifier for the scheduled process

fbsetrtc – Set Real–Time Clock

This subroutine is invoked to establish the duration of a minor cycle by setting the count and the resolution values for a real–time clock.

Call Statement

```
call fbsetrtc(schdle, count, resolution, istat1, istat2)
integer schdle, count, resolution, istat1, istat2
```

Parameters

Parameters must be specified in the order indicated. They are described as follows.

<i>schdle</i>	a unique, positive integer value representing the identifier for the frequency–based scheduler to which a real–time clock has been attached. You can obtain this value by making a call to fbconfigure(3f) (see page 7-7 for an explanation of this subroutine) or fbid(3f) (see page 7-12). If you wish to reference the frequency–based scheduler on which the calling process is scheduled without knowing its identifier, you can specify a value of -1 .
<i>count</i>	an integer value indicating the number of clock counts per minor cycle. This value can range from one to 65535.
<i>resolution</i>	an integer value indicating the duration in microseconds of one clock count. This value must be one of the following: 1, 10, 100, 1000, or 10000.
<i>istat1</i>	an integer value indicating whether or not an error has occurred. A value of zero indicates that no error has occurred. A nonzero value indicates that an error of a specific type has occurred. Refer to the fbsetrtc(3f) man page for a listing of the nonzero values that may be returned and the types of errors that they represent. If <i>istat1</i> contains a value indicating that an error has occurred on an open or ioctl call, the error status of that call is returned in <i>istat2</i> .
<i>istat2</i>	the error status of an open or ioctl call. See the include file <errno.h> for a description of the error.

fbswait – Wait on a Frequency-Based Scheduler

This subroutine enables a process that is scheduled on a frequency-based scheduler to sleep until its next scheduled minor cycle.

If the scheduled process does not call this library routine by its next scheduled minor cycle, the process may or may not get scheduled to run in that minor cycle depending upon whether or not the consecutive soft overrun limit or the consecutive missed **fbswait** call limit was exceeded. If either limit was not exceeded, this library routine will return immediately without putting the calling process to sleep and the process will run in the minor cycle that it overran in an attempt to recover from the overrun. If either limit was exceeded, the process' hard overrun count is incremented and the process does not get scheduled to run in that minor cycle. The hard overrun count, which can be read via **pmqrypgm(3f)**, and the soft overrun count, which can be read via **sched_pgm_soft_overrun_query(3f)**, indicate whether the process is actually running at its assigned frequency.

Call Statement

CALL **fbswait** (*istat*)

integer *istat*

Parameter

<i>istat</i>	an integer value indicating whether or not an error has occurred and whether the process has been wakened by the scheduler or by an pgmtrigger(3f) call from another process. Values that may be returned are as follows:
0	The process has been wakened normally
1	The process has been wakened as the result of a pgmtrigger(3f) call
2	The process did not sleep because the kernel detected a soft overrun and is allowing the process to attempt to recover from it
Other nonzero value	An error of a specific type has occurred. Refer to the fbswait(3f) man page for a listing of the nonzero values that may be returned and the types of errors that they represent.

nametopid – Return the Process ID for a Specified Process Name

The `nametopid` routine returns the process ID (pid) of the specified process.

Call Statement

```
call nametopid(name, key, cpu, pid, istat)
integer key, cpu, pid, istat
character* (*) name
```

Parameters

<i>name</i>	a standard path name identifying the process.
<i>key</i>	an integer value identifying a frequency-based scheduler; this value must be the same value that was specified for <i>key</i> when the scheduler was created by making a call to <code>fbconfigure</code> (see page 7-7 for an explanation of this subroutine). It may also be set to -1 to indicate that the process is not sheduled on a frequency-based scheduler.
<i>cpu</i>	an integer value indicating the processor(s) to be used in conjunction with the value of the <i>name</i> parameter to identify the program for which information is to be returned. Acceptable values and corresponding results are as follows: <ul style="list-style-type: none"> 0 The first process named by <i>name</i> that is currently running on the processor from which the call is made is specified -1 The first process named by <i>name</i> that is currently running on any processor is specified Bit mask If ($cpu \& (1 \ll i)$) is set (where <i>i</i> is an integer ranging from zero to 15 and representing a CPU) and it is the only bit set, the first process named by <i>name</i> that is running on CPU <i>i</i> is specified If ($cpu \& (1 \ll i)$) is set and it is not the only bit set, the first process named by <i>name</i> that is currently running on any of the selected CPUs is rescheduled
<i>pid</i>	the process ID of the process specified in <i>name</i>
<i>istat</i>	an integer value indicating whether or not an error has occurred. A value of zero indicates that no error has occurred. A nonzero value indicates that an error of a specific type has occurred. Refer to the <code>nametopid(3f)</code> man page for a listing of the nonzero values that may be returned and the types of errors that they represent.

pgmquery – Query a Process on a Frequency-Based Scheduler

CAUTION

This interface is obsolete. It is recommended that you use **schedpgmqry (3f)** (see p. 7-49).

Information that is returned by this subroutine includes the following:

- The process' path name
- The CPU on which the process can execute
- The frequency-based scheduler process identifier
- The scheduling priority
- The period (the number of minor cycles indicating the frequency with which the process is wakened in each major frame)
- The starting base cycle (the first minor cycle in which the process is scheduled to be wakened in each major frame)
- The value of the “halt on overrun” flag

Call Statement

```
call pgmquery (schdle, name, cpu, slot, prior, period, cycle, ab, istat)
integer schdle, cpu, slot, prior, period, cycle, ab, istat
character* (*) name
```

Parameters

<i>schdle</i>	a unique, positive integer value representing the identifier for the frequency-based scheduler on which the process for which you wish to obtain scheduling information has been scheduled. You can obtain this value by making a call to fbconfigure(3f) (see page 7-7 for an explanation of this subroutine) or fbid(3f) (see page 7-12). If you wish to reference the frequency-based scheduler on which the calling process is scheduled without knowing its identifier, you can specify a value of -1 .
<i>name</i>	a standard path name identifying the process for which information is to be returned. A full or relative path name of up to 1024 characters can be specified. If this variable contains blanks, you must provide the frequency-based scheduler process identifier in the <i>slot</i> parameter.
<i>cpu</i>	an integer value indicating the processor(s) to be used in conjunction with the value of the <i>name</i> parameter to identify the program for which information is to be returned. Acceptable values and corresponding results are as follows: 0 The first process named by <i>name</i> that is currently running on the processor from which the call is made is specified -1 The first process named by <i>name</i> that is currently running on any processor is specified

Bit mask If $(cpu \& (1 \ll i))$ is set (where i is an integer ranging from zero to 15 and representing a CPU) and it is the only bit set, the first process named by *name* that is running on CPU i is specified

If $(cpu \& (1 \ll i))$ is set and it is not the only bit set, the first process named by *name* that is currently running on any of the selected CPUs is rescheduled

<i>slot</i>	an integer value providing the unique frequency-based scheduler process identifier for the process for which information is to be returned. This value is obtained when you make a call to pgmschedule (see page 7-35 for an explanation of this subroutine). This value must be -1 if you wish to identify the program to be queried only by specifying <i>name</i> and <i>cpu</i> .
<i>prior</i>	an integer value indicating the specified process' scheduling priority.
<i>period</i>	an integer value indicating the frequency with which the specified program is to be wakened in each major frame.
<i>cycle</i>	an integer value indicating the first minor cycle in which the specified process is scheduled to be wakened in each frame
<i>ab</i>	an integer value indicating the value of the "halt on overrun" flag. A nonzero value indicates that the flag is set. A value of zero indicates that the flag is not set.
<i>istat</i>	an integer value indicating whether or not an error has occurred. A value of zero indicates that no error has occurred. A nonzero value indicates that an error of a specific type has occurred. Refer to the pgmquery(3f) man page for a listing of the nonzero values that may be returned and the types of errors that they represent.

pgmremove – Remove a Process from a Frequency-Based Scheduler

This subroutine is invoked to remove a process from a frequency-based scheduler. You can identify the process that you wish to remove by using one of the following methods:

- Specify the name of the process and the CPU on which it is scheduled.
- Specify the process' frequency-based scheduler process identifier (slot number).
- Specify the name of the process, the CPU on which it is scheduled, and its frequency-based scheduler process identifier.

NOTE

The only method that can be used to identify a process that has been scheduled multiple times on the same CPU is to specify its frequency-based scheduler process identifier.

Call Statement

```
call pgmremove (schdle, name, cpu, slot, ab, istat)
integer schdle, cpu, slot, ab, istat
character* (*) name
```

Parameters

<i>schdle</i>	a unique, positive integer value representing the identifier for the frequency-based scheduler on which the process is scheduled. You can obtain this value by making a call to fbsconfigure (3f) (see page 7-7 for an explanation of this subroutine) or fbsid (3f) (see page 7-12). If you wish to reference the frequency-based scheduler on which the calling process is scheduled without knowing its identifier, you can specify a value of -1 .
<i>name</i>	a standard path name identifying the process to be removed from the specified scheduler. A full or relative path name of up to 1024 characters can be specified. If this variable contains blanks, you must provide the frequency-based scheduler process identifier in the <i>slot</i> parameter.
<i>cpu</i>	an integer value indicating the processor(s) to be used in conjunction with the value of the <i>name</i> parameter to identify the process to be removed from the specified scheduler. Acceptable values and corresponding results are as follows:
0	The first process named by <i>name</i> that is currently running on the processor from which the call is made is removed
-1	The first process named by <i>name</i> that is currently running on any processor is removed
Bit mask	If (<i>cpu</i> & (1<< <i>i</i>)) is set (where <i>i</i> is an integer ranging from zero to 15 and representing a

CPU) and it is the only bit set, the first process named by *name* that is running on CPU *i* is specified

If (*cpu* & (1<<*i*)) is set and it is not the only bit set, the first process named by *name* that is currently running on any of the selected CPUs is specified

- slot* an integer value providing the unique frequency-based scheduler process identifier for the process to be removed from the specified scheduler. This value is obtained when you make a call to **schedpgmadd** (see page 7-47 for an explanation of this subroutine). This value must be **- 1** if you choose to identify the program to be removed only by specifying *name* and *cpu*.
- ab* an integer value indicating the manner in which the specified process is to be removed from the specified scheduler. A positive value indicates that the process is to be removed from the scheduler but allowed to continue executing. A negative value indicates that the process is to be removed from the scheduler and terminated.
- istat* an integer value indicating whether or not an error has occurred. A value of zero indicates that no error has occurred. A nonzero value indicates that an error of a specific type has occurred. Refer to the **pgmremove (3f)** man page for a listing of the nonzero values that may be returned and the types of errors that they represent.

pgmreschedule – Reschedule a Process

CAUTION

This interface is obsolete. It is maintained for compatibility with CX/UX, but its behavior with respect to specification of a process' scheduling priority has changed. If you have an existing application that uses this interface, it is recommended that you change your application to use `schedpgmresched(3f)` (see p. 7-52).

This subroutine is invoked to change the scheduling parameters for a process that is scheduled on a frequency-based scheduler. You may wish, for example, to change a program's priority or the frequency with which it is scheduled to run. You cannot, however, change the CPU on which it has been scheduled.

To change a process' priority, the following conditions must be met:

- The calling process must have the capabilities associated with the `fbscheduser` role (see Chapter 1).
- The effective user ID of the calling process must match the effective user ID of the target process (the process for which the scheduling policy and priority are being set), or the calling process must have the `P_OWNER` privilege.

You can call `pgmreschedule` to change the parameters without having called `pgmremove` to remove the process from the scheduler (see page 7-30) or `fbsintrpt` to stop the simulation (see page 7-15).

You can identify the process that you wish to reschedule by using one of the following methods:

- Specify the name of the process and the CPU on which it is scheduled.
- Specify the process' frequency-based scheduler process identifier (slot number).
- Specify the name of the process, the CPU on which it is scheduled, and its frequency-based scheduler process identifier.

NOTE

The only method that can be used to identify a process that has been scheduled multiple times on the same CPU is to specify its frequency-based scheduler process identifier.

Call Statement

```
call pgmreschedule (schdle, name, cpu, slot, prior, param, period, cycle, ab, istat)  
integer schdle, cpu, slot, prior, param, period, cycle, ab, istat  
character* (*) name
```

Parameters

<i>schdle</i>	a unique, positive integer value representing the identifier for the frequency-based scheduler on which the process is scheduled. You can obtain this value by making a call to fbconfigure(3f) (see page 7-7 for an explanation of this subroutine) or fbid(3f) (see page 7-12). If you wish to reference the frequency-based scheduler on which the calling process is scheduled without knowing its identifier, you can specify a value of -1 .
<i>name</i>	a standard path name identifying the process to be rescheduled. A full or relative path name of up to 1024 characters can be specified. If this variable contains blanks, you must provide the frequency-based scheduler process identifier in the <i>slot</i> parameter.
<i>cpu</i>	an integer value indicating the processor(s) to be used in conjunction with the value of the <i>name</i> parameter to identify the process to be rescheduled. Acceptable values and corresponding results are as follows: <ul style="list-style-type: none"> 0 The first process named by <i>name</i> that is currently running on the processor from which the call is made is rescheduled -1 The first process named by <i>name</i> that is currently running on any processor is rescheduled Bit mask If $(cpu \& (1 \ll i))$ is set (where <i>i</i> is an integer ranging from zero to 15 and representing a CPU) and it is the only bit set, the first process named by <i>name</i> that is running on CPU <i>i</i> is rescheduled <p style="margin-left: 40px;">If $(cpu \& (1 \ll i))$ is set and it is not the only bit set, the first process named by <i>name</i> that is currently running on any of the selected CPUs is rescheduled</p>
<i>slot</i>	an integer value providing the unique frequency-based scheduler process identifier for the process to be rescheduled. This value is obtained when you make a call to pgmschedule (see page 7-35 for an explanation of this subroutine). This value must be -1 if you wish to identify the program to be rescheduled only by specifying <i>name</i> and <i>cpu</i> .
<i>prior</i>	an integer value indicating the specified process' scheduling priority. A process that has been scheduled using pgmschedule (see p. 7-35 for an explanation of this subroutine) is scheduled under the POSIX SCHED_RR scheduling policy. The value specified must lie in the range of priorities associated with this policy. You can obtain the allowable range of priorities by invoking the run(1) command from the shell and not specifying any options or arguments (see the corresponding man page for an explanation of this command). Higher numerical values correspond to more

	favorable scheduling priorities. For complete information on scheduling policies and priorities, refer to the <i>RedHawk Linux User's Guide</i> .
<i>param</i>	an integer value to be passed to a process that is scheduled on a frequency-based scheduler.
<i>period</i>	an integer value indicating the frequency with which the specified program is to be wakened in each major frame. A period of one indicates that the specified program is to be wakened every minor cycle; two indicates that it is to be wakened once every two minor cycles, three once every three minor cycles, and so on. This value can range from one to the number of minor cycles that compose a frame on the specified scheduler as defined in a call to fbsconfigure (see page 7-7).
<i>cycle</i>	an integer value indicating the first minor cycle in which the specified process is scheduled to be wakened in each frame. This value can range from zero to the total number of minor cycles per frame minus one. The total number of minor cycles per frame is specified in a call to fbsconfigure (see page 7-7 for an explanation of this subroutine).
<i>ab</i>	an integer value indicating whether or not the scheduler should be stopped in the event that the specified process causes a frame overrun. A nonzero value indicates that the scheduler will be stopped.
<i>istat</i>	an integer value indicating whether or not an error has occurred. Zero indicates that no error has occurred. A nonzero value indicates that an error of a specific type has occurred. Refer to the pgmreschedule(3f) man page for a listing of the nonzero values that may be returned and the types of errors that they represent.

pgmschedule – Schedule a Process on a Frequency-Based Scheduler

CAUTION

This interface is obsolete. It is maintained for compatibility with CX/UX, but its behavior with respect to specification of a process' scheduling priority has changed. If you have an existing application that uses this interface, it is recommended that you change your application to use `schedpgramadd(3f)` (see p. 7-47).

This subroutine is invoked to create a new process and schedule it on a frequency-based scheduler. When a process is scheduled using this subroutine, it is scheduled under the POSIX `SCHED_RR` scheduling policy (for complete information on scheduling policies and priorities, refer to the *RedHawk Linux User's Guide*).

If you wish to set the process' scheduling priority, the following conditions must be met:

- The calling process must have the capabilities associated with the `fbscheduser` role (see Chapter 1).
- The effective user ID of the calling process must match the effective user ID of the target process (the process for which the scheduling policy and priority are being set), or the calling process must have the `P_OWNER` privilege.

If you wish to modify the process' CPU bias when you invoke this subroutine, the real or effective user ID of the calling process must match the real or saved user ID of the process for which the CPU assignment is being changed.

Call Statement

```
call pgmschedule(schdle, name, prior, param, period, cycle, ab, cpu, slot, istat)
integer schdle, prior, param, period, cycle, ab, cpu, slot, istat
character* (*) name
```

Parameters

<i>schdle</i>	a unique, positive integer value representing the identifier for a frequency-based scheduler. You can obtain this value by making a call to <code>fbsconfigure(3f)</code> (see page 7-7 for an explanation of this subroutine) or <code>fbsid(3f)</code> (see page 7-12). If you wish to reference the frequency-based scheduler on which the calling process is scheduled without knowing the identifier, you can specify the value <code>-1</code> .
<i>name</i>	a standard path name identifying the program to be scheduled on the scheduler. A full or relative path name of up to 1024 characters can be specified.
<i>prior</i>	an integer value indicating the specified process' scheduling priority. A process that is scheduled using <code>pgmschedule</code> is scheduled under the POSIX <code>SCHED_RR</code> scheduling policy. The value specified must lie in the range of priorities associated with this policy. You can obtain the allowable range of priorities by invoking the <code>run(1)</code> command from the shell

and not specifying any options or arguments (see the corresponding man page for an explanation of this command). Higher numerical values correspond to more favorable scheduling priorities.

For complete information on scheduling policies and priorities, refer to the *RedHawk Linux User's Guide*.

<i>param</i>	an integer value to be passed to a process that is scheduled on a frequency-based scheduler. This value can be retrieved by the FBS-scheduled process through a call to rtparm (see page 7-41 for an explanation of this subroutine).
<i>period</i>	an integer value indicating the frequency with which the specified program is to be wakened in each major frame. A period of one indicates that the specified program is to be wakened every minor cycle; a period of two indicates that it is to be wakened once every two minor cycles, a period of three once every three minor cycles, and so on. This value can range from one to the number of minor cycles that compose a frame on the specified scheduler as defined in a call to fbsconfigure (see page 7-7).
<i>cycle</i>	an integer value indicating the first minor cycle in which the specified program is scheduled to be wakened in each frame. This value can range from zero to the total number of minor cycles per frame minus one. (The total number of minor cycles per frame is specified in a call to fbsconfigure . See page 7-7 for an explanation of this subroutine.)
<i>ab</i>	refers to a flag that contains an integer value indicating whether or not the scheduler should be stopped in the event that the specified program causes a frame overrun. A nonzero value indicates that the scheduler will be stopped.
<i>cpu</i>	refers to a mask that identifies the processors on which the specified program can be scheduled to run. Acceptable values and corresponding results are as follows: <ul style="list-style-type: none"> 0 The program specified by <i>name</i> can be scheduled on the processor from which the call is made -1 The program specified by <i>name</i> can be scheduled on any processor Bit mask If (<i>cpu</i> & (1<<<i>i</i>)) is set (where <i>i</i> is an integer ranging from zero to 15 and representing a CPU) the program specified by <i>name</i> can be scheduled on CPU <i>i</i>
<i>slot</i>	refers to a variable to which pgmschedule will return an integer value that is the unique frequency-based scheduler process identifier for the scheduled process.

istat

refers to a variable to which **pgmschedule** will return an integer value indicating whether or not an error has occurred. A value of zero indicates that no error has occurred. A nonzero value indicates that an error of a specific type has occurred. Refer to the **pgmschedule (3f)** man page for a listing of the nonzero values that may be returned and the types of errors that they represent.

pgmstat – Query State of FBS–Scheduled Process

This subroutine is invoked to obtain information about the state of a particular process that has been scheduled on a frequency–based scheduler. The state of the process indicates whether it is in the **fbwait** sleep state or is in another state.

You can identify the process by using one of the following methods:

- Specify the name of the process and the CPU(s) on which it is scheduled.
- Specify the process' frequency–based scheduler process identifier (slot number).
- Specify the name of the process, the CPU on which it is scheduled, and its frequency–based scheduler process identifier.

NOTE

The only method that can be used to identify a process that has been scheduled multiple times on the same CPU is to specify its frequency–based scheduler process identifier.

Information that is returned includes the following:

- The process' path name
- A mask of the CPU(s) on which the process can run
- The frequency–based scheduler process identifier
- The current state of the process

Call Statement

```
call pgmstat (schdle, name, cpu, slot, state, istat)  
integer schdle, cpu, slot, state, istat  
character* (*) name
```

Parameters

schdle a unique, positive integer value representing the identifier for the frequency–based scheduler on which the process for which you wish to obtain state information has been scheduled. You can obtain this value by making a call to **fbconfigure(3f)** (see page 7-7 for an explanation of this subroutine) or **fbid(3f)** (see page 7-12). If you wish to reference the frequency–based scheduler on which the calling process is scheduled without knowing its identifier, you can specify a value of **-1**.

name a standard path name identifying the process for which state information is to be returned. A full or relative path name of up to 1024 characters can be specified. If this variable contains blanks, you must provide the frequency–based scheduler process identifier in the *slot* parameter. **Pgmstat** will return to this variable the path name of the specified FBS–scheduled process.

<i>cpu</i>	<p>an integer value indicating the processor(s) to be used in conjunction with the value of the <i>name</i> parameter to identify the program for which state information is to be returned. Acceptable values and corresponding results are as follows:</p> <p>0 The first process named by <i>name</i> that is currently running on the processor from which the call is made is specified</p> <p>-1 The first process named by <i>name</i> that is currently running on any processor is specified</p> <p>Bit mask If (<i>cpu</i> & (1<<<i>i</i>)) is set (where <i>i</i> is an integer ranging from zero to 15 and representing a CPU) and it is the only bit set, the first process named by <i>name</i> that is running on CPU <i>i</i> is specified</p> <p style="padding-left: 40px;">If (<i>cpu</i> & (1<<<i>i</i>)) is set and it is not the only bit set, the first process named by <i>name</i> that is currently running on any of the selected CPUs is specified</p> <p>Pgmstat will <u>return</u> to this variable the mask of the CPUs on which the specified process can run.</p>
<i>slot</i>	<p>an integer value providing the unique frequency-based scheduler process identifier for the process for which status information is to be returned. This value is obtained when you make a call to schedpgmadd (see page 7-47 for an explanation of this subroutine). This value must be - 1 if you wish to identify the program to be queried only by specifying <i>name</i> and <i>cpu</i>. Pgmstat will <u>return</u> to this variable the frequency-based scheduler process identifier for the specified process.</p>
<i>state</i>	<p>an integer value indicating the current state of the specified process as defined in <fbsched.h></p>
<i>istat</i>	<p>an integer value indicating whether or not an error has occurred. A value of zero indicates that no error has occurred. A nonzero value indicates that an error of a specific type has occurred. Refer to the pgmstat (3f) man page for a listing of the nonzero values that may be returned and the types of errors that they represent.</p>

pgmtrigger – Trigger Process Waiting on FBS

This subroutine enables a process to wake a process that is in the **fbwait** sleep state. It is important to note that the calling process does not have to be scheduled on a frequency-based scheduler; the target process must be.

Call Statement

```
call pgmtrigger(schdle, slot, tgrflg, istat)  
integer schdle, slot, tgrflg, istat
```

Parameters

<i>schdle</i>	a unique, positive integer value representing the identifier for a frequency-based scheduler on which the sleeping process is scheduled.
<i>slot</i>	an integer value providing the unique frequency-based scheduler process identifier for the sleeping process. This value is obtained when you make a call to schedpgmadd (see page 7-47 for an explanation of this subroutine).
<i>tgrflg</i>	an integer value indicating whether or not a context switch is to be forced on the processor on which the wakened process is executing. A nonzero value indicates that a context switch is to be forced.
<i>istat</i>	an integer value indicating whether or not an error has occurred. A value of zero indicates that the process is runnable. A nonzero value indicates that an error of a specific type has occurred. Refer to the pgmtrigger(3f) man page for a listing of the nonzero values that may be returned and the types of errors that they represent.

rtparm – Return Initiation Parameter

This subroutine enables a process that is scheduled on a frequency-based scheduler to obtain the value of a process initiation parameter that has been passed to it via a call to **schedpgmadd** (see page 7-47) or **schedpgmresched** (see 7-52).

Call Statement

```
call rtparm(param)  
integer param
```

Parameter

param the integer value passed to the process via a call to **schedpgmadd** or **schedpgmresched**.

sched_pgm_set_soft_ouerrun_limit – Set Soft Overrun Limit

This subroutine sets the consecutive soft overrun limit for a currently scheduled process or thread on a frequency-based scheduler.

To set the consecutive soft overrun limit, the calling process must have alter permission for the scheduler.

The process can be identified in one of the following ways:

- A slot only (if name is blank).
- A path name and processor id pair only (if slot is -1).
- Both a slot and the path name and processor id pair.

Call Statement

```
call sched_pgm_set_soft_ouerrun_limit (schdle, name, cpu, slot, soft_limit,
    istat)
integer schdle,cpu, slot, soft_limit, istat
character* (*) name
```

Parameters

<i>schdle</i>	obtained from an fbconfigure(3f) or fbid(3f) library routine call or set to -1. -1 enables an FBS-scheduled process to reference the frequency-based scheduler on which it is scheduled without knowing the scheduler identifier.
<i>name</i>	path name that identifies the process. If the name is all blanks, then the slot field (frequency-based scheduler process identifier) must be given.
<i>cpu</i>	either a bit mask or set to 0 or -1. If a bit mask is specified, then those processors with (<i>cpu</i> & (1 << <i>i</i>)) set are requested. If <i>cpu</i> is 0, then the processor on which the call is made is requested. If <i>cpu</i> is -1, then all processors are requested. The first process named that is currently running on one of the requested processors has its soft overrun limit set.
<i>slot</i>	frequency-based scheduler process identifier for the process. If the slot number equals -1, then a name and processor ID must be given.
<i>soft_limit</i>	number of consecutive soft overruns allowed to occur before failure. <i>soft_limit</i> must be non-negative and must be less than INT_MAX. By default, this value is zero; i.e., if the process never sets a consecutive soft overrun limit, then it is zero.
<i>istat</i>	an integer value indicating whether or not an error has occurred. A value of zero indicates that no error has occurred. A nonzero value indicates that an error of a specific type has occurred. Refer to the <code>man</code> page for a listing of the nonzero values that may be returned and the types of errors that they represent.

sched_pgm_soft_ouerrun_query – Query Soft Overrun Processing

This subroutine queries the status of soft overrun processing for a currently scheduled process or thread on a frequency-based scheduler. The process can be identified in one of the following ways:

- A slot only (if name is blank).
- A path name and processor id pair only (if slot is -1).
- Both a slot and the path name and processor id pair.

Call Statement

```
call sched_pgm_soft_ouerrun_query (schdle, name, cpu, slot, soft_limit,
    soft_total, istat)
integer schdle, cpu, slot, soft_limit, soft_total, istat
character* (*) name
```

Parameters

<i>schdle</i>	obtained from an fbconfigure(3f) or fbsid(3f) library routine call or set to -1. -1 enables an FBS-scheduled process to reference the frequency-based scheduler on which it is scheduled without knowing the scheduler identifier.
<i>name</i>	path name that identifies the process. If the name is all blanks, then the slot field (frequency-based scheduler process identifier) must be given.
<i>cpu</i>	either a bit mask or set to 0 or -1. If a bit mask is specified, then those processors with ($cpu \& (1 \ll i)$) set are requested. If <i>cpu</i> is 0, then the processor on which the call is made is requested. If <i>cpu</i> is -1, then all processors are requested. The first process named <i>name</i> that is currently running on one of the requested processors is returned.
<i>slot</i>	frequency-based scheduler process identifier for the process. If the slot number equals -1, then a name and processor id must be given.
<i>soft_limit</i>	number of consecutive soft overruns set by calling sched_pgm_set_soft_ouerrun_limit(3f) .
<i>soft_total</i>	total number of soft overruns incurred by the process.
<i>istat</i>	an integer value indicating whether or not an error has occurred. A value of zero indicates that no error has occurred. A nonzero value indicates that an error of a specific type has occurred. Refer to the <code>man</code> page for a listing of the nonzero values that may be returned and the types of errors that they represent.

schedfbsqry – Query Processes on a Frequency-Based Scheduler

The **schedfbsqry** subroutine is invoked to obtain information about processes that have been scheduled on a frequency-based scheduler. Information is returned for all processes scheduled on the user-specified processor(s). Information provided for each process includes the following:

- A mask of the CPU(s) on which the process can execute
- The frequency-based scheduler process identifier
- The scheduling policy under which the process has been scheduled
- The scheduling priority
- The period (the number of minor cycles indicating the frequency with which the process is wakened in each major frame)
- The starting base cycle (the first minor cycle in which the process is scheduled to be wakened in each major frame)
- The value of the “halt on overrun” flag

Call Statement

```
CALL schedfbsqry (schdle, cpu, buf1size, buf1, maxsize, buf2size, buf2, istat)
integer schdle, cpu, buf1size, buf1(buf1size), maxsize, buf2size, istat
character* (*) buf2
```

Parameters

<i>schdle</i>	a unique, positive integer value representing the identifier for the frequency-based scheduler for which you wish to obtain scheduling information. You can obtain this value by making a call to fbsconfigure (3f) (see page 7-7 for an explanation of this subroutine) or fbsid (3f) (see page 7-12). If you wish to reference the frequency-based scheduler on which the calling process is scheduled without knowing its identifier, you can specify a value of -1 .						
<i>cpu</i>	an integer value indicating the processor(s) for which scheduling information is to be obtained. Acceptable values and corresponding results are as follows: <table style="margin-left: 2em;"> <tr> <td>0</td> <td>Scheduling information for processes executing on the processor from which the call is made is returned</td> </tr> <tr> <td>-1</td> <td>Scheduling information for all processes on the scheduler is returned</td> </tr> <tr> <td>Bit mask</td> <td>If $(cpu \& (1 \ll i))$ is set (where <i>i</i> is an integer ranging from zero to 15 and representing a CPU), scheduling information for processes executing on CPU <i>i</i> is returned</td> </tr> </table>	0	Scheduling information for processes executing on the processor from which the call is made is returned	-1	Scheduling information for all processes on the scheduler is returned	Bit mask	If $(cpu \& (1 \ll i))$ is set (where <i>i</i> is an integer ranging from zero to 15 and representing a CPU), scheduling information for processes executing on CPU <i>i</i> is returned
0	Scheduling information for processes executing on the processor from which the call is made is returned						
-1	Scheduling information for all processes on the scheduler is returned						
Bit mask	If $(cpu \& (1 \ll i))$ is set (where <i>i</i> is an integer ranging from zero to 15 and representing a CPU), scheduling information for processes executing on CPU <i>i</i> is returned						
<i>buf1size</i>	an integer value indicating the size in 32-bit words of the array represented by <i>buf1</i> . Because 9 words of information are returned for each process, it is recommended that this value be a multiple of 9.						

buf1 an array containing a series of 11 integer values for each process on the processor(s) specified with the *cpu* parameter. The number of processes for which these values are returned is bound by the value of the *buf1size* parameter. If, for example, the value of *buf1size* is 145, values for 16 processes will be returned. These values represent the scheduling information for the process(es). The type of information returned in each array element for a single process is presented in Table 7-6.

Table 7-6. Contents of Array Elements: schedfbsqry

Element	Contents
1	Byte offset of the process' path name in <i>buf2</i>
2	Length in bytes of the process' path name
3	Mask of the CPU(s) on which the process can execute
4	The process' frequency-based scheduler process identifier
5	The process' scheduling policy
6	The process' scheduling priority
7	The number of minor cycles indicating the frequency with which the process is to be wakened in each major frame (period)
8	The first minor cycle in which the process is scheduled to be wakened in each major frame (starting base cycle)
9	The value of the "halt on overrun" flag. A nonzero value indicates that the flag is set. A value of zero indicates that the flag is not set.

maxsize an integer value indicating the maximum length of a path name to be returned in *buf2*

buf2size an integer value indicating the size in bytes of the character string represented by *buf2*. To ensure that *buf2* is large enough to accommodate the names of all processes that you wish to query, you may find it helpful to compute the number of bytes needed by multiplying the maximum number of processes allowed on the scheduler (see the information on **fbsconfigure** presented on page 7-7) by 32.

buf2 refers to a variable to which **schedfbsqry** will return the path names for each process on the processor(s) specified with the *cpu* parameter. Path names are returned as a series of strings. The length of each string is less than or equal to the value of *maxsize*. Where *maxsize* is not large enough to accommodate a full path name, the concluding component names are returned. The number of path names returned is bound by the value of the *buf2size* parameter.

istat

refers to a variable to which **schedfbsqry** will return an integer value indicating whether or not an error has occurred. A value of zero indicates that no error has occurred. A nonzero value indicates that an error of a specific type has occurred. Refer to the **schedfbsqry(3f)** man page for a listing of the nonzero values that may be returned and the types of errors that they represent.

schedpgramadd – Schedule a Process on a Frequency-Based Scheduler

The `schedpgramadd` subroutine is invoked to create a new process and schedule it on a frequency-based scheduler. It is important to note that to use this subroutine, the calling process must have the capabilities associated with the `fbscheduler` role (see Chapter 1).

Call Statement

```
call schedpgramadd (schdle, name, cid, prior, param, period, cycle, ab, cpu, slot, istat)
integer schdle, cid, prior, param, period, cycle, ab, cpu, slot, istat
character* (*) name
```

Parameters

<i>schdle</i>	a unique, positive integer value representing the identifier for a frequency-based scheduler. You can obtain this value by making a call to <code>fbsconfigure(3f)</code> (see page 7-7 for an explanation of this subroutine) or <code>fbsid(3f)</code> (see page 7-12). If you wish to reference the frequency-based scheduler on which the calling process is scheduled without knowing the identifier, you can specify the value <code>-1</code> .
<i>name</i>	a standard path name identifying the program to be scheduled on the scheduler. A full or relative path name of up to 1024 characters can be specified.
<i>cid</i>	an integer value indicating the POSIX scheduling policy under which the specified process is to be scheduled: <code>SCHED_FIFO</code> , <code>SCHED_RR</code> or <code>SCHED_OTHER</code> . Scheduling policies are defined in the file <code><sched.h></code> .
<i>prior</i>	an integer value indicating the scheduling priority of the specified program. The range of acceptable priority values is governed by the scheduling policy specified. You can determine the allowable range of priorities associated with each policy (<code>SCHED_FIFO</code> , <code>SCHED_RR</code> , or <code>SCHED_OTHER</code>) by invoking the <code>run(1)</code> command from the shell and not specifying any options or arguments (see the corresponding man page for an explanation of this command). Higher numerical values correspond to more favorable priorities. For complete information on scheduling policies and priorities, refer to the <i>RedHawk Linux User's Guide</i> .
<i>param</i>	an integer value to be passed to a process that is scheduled on a frequency-based scheduler. This value can be retrieved by the FBS-scheduled process through a call to <code>rtparam</code> (see page 7-41 for an explanation of this subroutine).
<i>period</i>	an integer value indicating the frequency with which the specified program is to be wakened in each major frame. A period of one indicates that the specified program is to be wakened every minor cycle; a period of two indicates that it is to be wakened once every two minor cycles, a period of three once every three minor cycles, and so on. This value can range

	from one to the number of minor cycles that compose a frame on the specified scheduler as defined in a call to fbsconfigure (see page 7-7).
<i>cycle</i>	an integer value indicating the first minor cycle in which the specified program is scheduled to be wakened in each frame. This value can range from zero to the total number of minor cycles per frame minus one. (The total number of minor cycles per frame is specified in a call to fbsconfigure . See page 7-7 for an explanation of this subroutine).
<i>ab</i>	a flag that contains an integer value indicating whether or not the scheduler should be stopped in the event that the specified program causes a frame overrun. A nonzero value indicates that the scheduler will be stopped.
<i>cpu</i>	a mask that identifies the processors on which the specified program can be scheduled to run. Acceptable values and corresponding results are as follows: 0 The program specified by <i>name</i> can be scheduled on the processor from which the call is made -1 The program specified by <i>name</i> can be scheduled on any processor Bit mask If $(cpu \& (1 \ll i))$ is set (where <i>i</i> is an integer ranging from zero to 15 and representing a CPU) the program specified by <i>name</i> can be scheduled on CPU <i>i</i>
<i>slot</i>	an integer value that is the unique frequency-based scheduler process identifier for the scheduled process.
<i>istat</i>	an integer value indicating whether or not an error has occurred. A value of zero indicates that no error has occurred. A nonzero value indicates that an error of a specific type has occurred. Refer to the schedpgmadd(3f) man page for a listing of the nonzero values that may be returned and the types of errors that they represent.

schedpqry – Query a Process on a Frequency-Based Scheduler

The `schedpqry` subroutine is invoked to obtain information for a particular process that has been scheduled on a frequency-based scheduler. You can identify the process by using one of the following methods:

- Specify the name of the process and the CPU(s) on which it is scheduled.
- Specify the process' frequency-based scheduler process identifier (slot number).
- Specify the name of the process, the CPU on which it is scheduled, and its frequency-based scheduler process identifier.

NOTE

The only method that can be used to identify a process that has been scheduled multiple times on the same CPU is to specify its frequency-based scheduler process identifier.

Information that is returned includes the following:

- The process' path name
- The CPU on which the process can execute
- The frequency-based scheduler process identifier
- The scheduling policy
- The scheduling priority
- The period (the number of minor cycles indicating the frequency with which the process is wakened in each major frame)
- The starting base cycle (the first minor cycle in which the process is scheduled to be wakened in each major frame)
- The value of the "halt on overrun" flag

Call Statement

```
call schedpqry (schdle, name, cpu, slot, cid, prior, period, cycle, ab, istat)
integer schdle, cpu, slot, cid, prior, period, cycle, ab, istat
character* (*) name
```

Parameters

<i>schdle</i>	a unique, positive integer value representing the identifier for the frequency-based scheduler on which the process for which you wish to obtain scheduling information has been scheduled. You can obtain this value by making a call to fbconfigure(3f) (see page 7-7 for an explanation of this subroutine) or fbid(3f) (see page 7-12). If you wish to reference the frequency-based scheduler on which the calling process is scheduled without knowing its identifier, you can specify a value of -1 .
<i>name</i>	a standard path name identifying the process for which information is to be returned. A full or relative path name of up to 1024 characters can be specified. If this variable contains blanks, you must provide the frequency-based scheduler process identifier in the <i>slot</i> parameter.
<i>cpu</i>	an integer value indicating the processor(s) to be used in conjunction with the value of the <i>name</i> parameter to identify the program for which information is to be returned. Acceptable values and corresponding results are as follows: <ul style="list-style-type: none"> 0 The first process named by <i>name</i> that is currently running on the processor from which the call is made is specified -1 The first process named by <i>name</i> that is currently running on any processor is specified Bit mask If $(cpu \& (1 \ll i))$ is set (where <i>i</i> is an integer ranging from zero to 15 and representing a CPU) and it is the only bit set, the first process named by <i>name</i> that is running on CPU <i>i</i> is specified If $(cpu \& (1 \ll i))$ is set and it is not the only bit set, the first process named by <i>name</i> that is currently running on any of the selected CPUs is specified
<i>slot</i>	an integer value providing the unique frequency-based scheduler process identifier for the process for which information is to be returned. This value is obtained when you make a call to schedpgmadd (see page 7-47 for an explanation of this subroutine). This value must be -1 if you wish to identify the program to be queried only by specifying <i>name</i> and <i>cpu</i> .
<i>cid</i>	an integer value indicating the scheduling policy under which the specified process has been scheduled
<i>prior</i>	an integer value indicating the specified process' scheduling priority
<i>period</i>	an integer value indicating the frequency with which the specified program is to be wakened in each major frame.

<i>cycle</i>	an integer value indicating the first minor cycle in which the specified process is scheduled to be wakened in each frame
<i>ab</i>	an integer value indicating the value of the “halt on overrun” flag. A nonzero value indicates that the flag is set. A value of zero indicates that the flag is not set.
<i>istat</i>	an integer value indicating whether or not an error has occurred. A value of zero indicates that no error has occurred. A nonzero value indicates that an error of a specific type has occurred. Refer to the schedpgrmry (3f) man page for a listing of the nonzero values that may be returned and the types of errors that they represent.

schedpgramresched – Reschedule a Process

The **schedpgramresched** subroutine is invoked to change the scheduling parameters for a process that is scheduled on a frequency-based scheduler. You may wish, for example, to change a program's scheduling policy or priority or the frequency with which it is scheduled to run. You cannot, however, change the CPU on which it has been scheduled.

If you wish to (1) change a process' scheduling policy to the SCHED_FIFO or the SCHED_RR policy, (2) change the priority of a process scheduled under the SCHED_FIFO or the SCHED_RR policy or (3) raise the priority of a process scheduled under the SCHED_OTHER policy above a per-process or process limit, the following conditions must be met:

- The calling process must have the capabilities associated with the *fbscheduser* role (see Chapter 1).
- The effective user ID of the calling process must match the effective user ID of the target process (the process for which the scheduling policy and priority are being set).

You can identify the process that you wish to reschedule by using one of the following methods:

- Specify the name of the process and the CPU on which it is scheduled.
- Specify the process' frequency-based scheduler process identifier (slot number).
- Specify the name of the process, the CPU on which it is scheduled, and its frequency-based scheduler process identifier.

NOTE

The only method that can be used to identify a process that has been scheduled multiple times on the same CPU is to specify its frequency-based scheduler process identifier.

Call Statement

```
call schedpgramresched (schdle, name, cpu, slot, cid, prior, param, period, cycle,
                        ab, istat)
integer schdle, cpu, slot, cid, prior, param, period, cycle, ab, istat
character* (*) name
```

Parameters

<i>schdle</i>	a unique, positive integer value representing the identifier for the frequency-based scheduler on which the process is scheduled. You can obtain this value by making a call to fbsconfigure(3f) (see page 7-7 for an explanation of this subroutine) or fbsid(3f) (see page 7-12). If you wish to reference the frequency-based scheduler on which the calling process is scheduled without knowing its identifier, you can specify a value of -1 .
<i>name</i>	a standard path name identifying the process to be rescheduled. A full or relative path name of up to 1024 characters can be specified. If this variable contains blanks,

you must provide the frequency-based scheduler process identifier in the *slot* parameter.

<i>cpu</i>	<p>an integer value indicating the processor(s) to be used in conjunction with the value of the <i>name</i> parameter to identify the process to be rescheduled. Acceptable values and corresponding results are as follows:</p> <p>0 The first process named by <i>name</i> that is currently running on the processor from which the call is made is rescheduled</p> <p>-1 The first process named by <i>name</i> that is currently running on any processor is rescheduled</p> <p>Bit mask If (<i>cpu</i> & (1<<<i>i</i>)) is set (where <i>i</i> is an integer ranging from zero to 15 and representing a CPU) and it is the only bit set, the first process named by <i>name</i> that is running on CPU <i>i</i> is rescheduled</p> <p style="padding-left: 40px;">If (<i>cpu</i> & (1<<<i>i</i>)) is set and it is not the only bit set, the first process named by <i>name</i> that is currently running on any of the selected CPUs is rescheduled</p>
<i>slot</i>	<p>an integer value providing the unique frequency-based scheduler process identifier for the process to be rescheduled. This value is obtained when you make a call to schedpgramadd (see page 7-47 for an explanation of this subroutine). This value must be -1 if you wish to identify the program to be rescheduled only by specifying <i>name</i> and <i>cpu</i>.</p>
<i>cid</i>	<p>an integer value indicating the scheduling policy under which the specified program is to be scheduled: SCHED_FIFO, SCHED_RR or SCHED_OTHER. Scheduling policies are defined in the file <sched.h>.</p>
<i>prior</i>	<p>an integer value indicating the scheduling priority of the specified program. The range of acceptable priority values is governed by the scheduling policy specified.</p> <p>You can determine the allowable range of priorities associated with each policy (SCHED_FIFO, SCHED_RR, or SCHED_OTHER) by invoking the run (1) command from the shell and not specifying any options or arguments (see the corresponding man page for an explanation of this command). Higher numerical values correspond to more favorable priorities. For complete information on scheduling policies and priorities, refer to the <i>RedHawk Linux User's Guide</i>.</p>
<i>param</i>	<p>an integer value to be passed to a process that is scheduled on a frequency-based scheduler.</p>
<i>period</i>	<p>an integer value indicating the frequency with which the specified program is to be wakened in each major frame. A period of one indicates that the specified program is to be</p>

- wakened every minor cycle; a period of two indicates that it is to be wakened once every two minor cycles, a period of three once every three minor cycles, and so on. This value can range from one to the number of minor cycles that compose a frame on the specified scheduler as defined in a call to **fbsconfigure** (see page 7-7).
- cycle* an integer value indicating the first minor cycle in which the specified process is scheduled to be wakened in each frame. This value can range from zero to the total number of minor cycles per frame minus one. The total number of minor cycles per frame is specified in a call to **fbsconfigure** (see page 7-7 for an explanation of this subroutine).
- ab* a flag that contains an integer value indicating whether or not the scheduler should be stopped in the event that the specified process causes a frame overrun. A nonzero value indicates that the scheduler will be stopped.
- istat* an integer value indicating whether or not an error has occurred. A value of zero indicates that no error has occurred. A nonzero value indicates that an error of a specific type has occurred. Refer to the **schedpgramresched(3f)** man page for a listing of the nonzero values that may be returned and the types of errors that they represent.

Performance Monitor Routines

The Performance Monitor routines provide access to the key features of the Performance Monitor. They enable you to perform such basic operations as:

- clearing performance monitor values for a process or processor
- starting and stopping performance monitoring for a process or processor
- obtaining performance monitor values for a process or processor

Routine Summary

Performance Monitor routines are summarized in Table 7-7. Complete information about each routine is provided under the section “Using Performance Monitor Routines.”

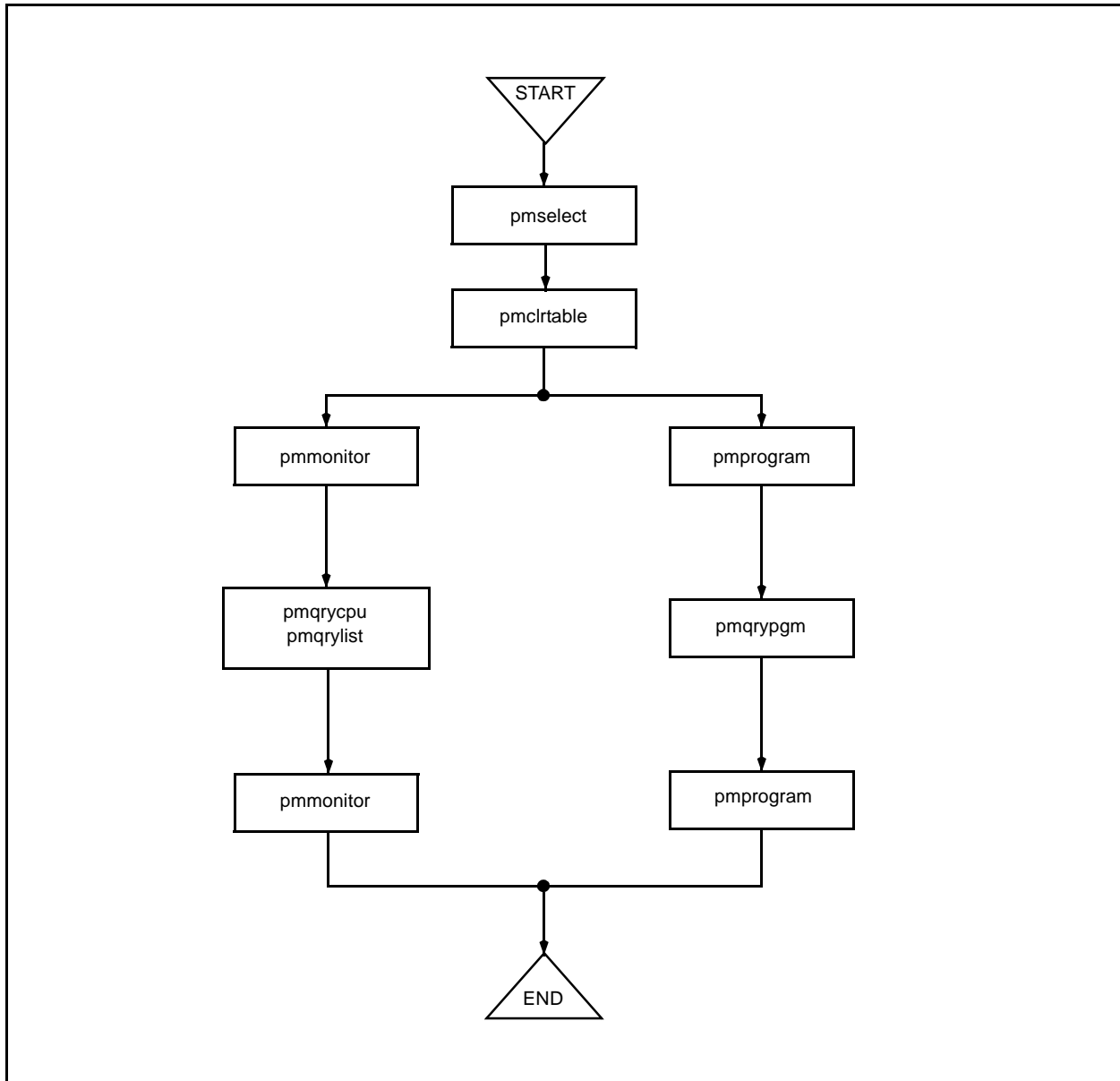
Table 7-7. Performance Monitor FORTRAN Library Routines

Routine	Page	Description
pmclrpgm	7-57	Clear values for a process
pmclrtable	7-59	Clear values for processor(s)
pmmonitor	7-60	Start/stop performance monitoring on processor(s)
pmprogram	7-61	Start/stop performance monitoring on a process
pmqrycpu	7-63	Query values for selected processor(s)
pmqrylist	7-65	Query values for a list of processes
pmqrypgm	7-67	Query values for a selected process
pmquerytimer	7-70	Query Performance Monitor mode
pmselect	7-71	Select Performance Monitor mode

FORTRAN Library Call Sequence

The approximate order in which you might call the Performance Monitor routines from an application program is illustrated in Figure 7-2.

Figure 7-2. FORTRAN Library Call Sequence: Performance Monitor



Using Performance Monitor Routines

In the sections that follow, the Performance Monitor routines contained in the `libF77rt` library are presented in alphabetical order.

pmclrpgm – Clear Values for a Process

This subroutine is invoked to clear performance monitor values for a particular process that has been scheduled on a frequency-based scheduler. You can identify the process by using one of the following methods:

- Specify the name of the process and the CPU(s) on which it is scheduled.
- Specify the process' frequency-based scheduler process identifier (slot number).
- Specify the name of the process, the CPU on which it is scheduled, and its frequency-based scheduler process identifier.

NOTE

This subroutine will clear the process' total soft overrun count.

Call Statement

```
call pmclrpgm(schdle, name, cpu, slot, istat)
integer schdle, cpu, slot, istat
character* (*) name
```

Parameters

<i>schdle</i>	a unique, positive integer value representing the identifier for the frequency-based scheduler on which the process is scheduled. You can obtain this value by making a call to fbsconfigure(3f) (see page 7-7 for an explanation of this subroutine) or fbsid(3f) (see page 7-12). If you wish to reference the frequency-based scheduler on which the calling process is scheduled without knowing its identifier, you can specify a value of -1 .
<i>name</i>	a standard path name identifying the process for which values are to be cleared. A full or relative path name of up to 1024 characters can be specified. If this variable is filled with blanks, you must provide the frequency-based scheduler process identifier in the <i>slot</i> parameter.
<i>cpu</i>	an integer value indicating the processor(s) to be used in conjunction with the value of the <i>name</i> parameter to identify the process for which values are to be cleared. Acceptable values and corresponding results are as follows:
0	The first process named by <i>name</i> that is currently running on the processor from which the call is made is specified
-1	The first process named by <i>name</i> that is currently running on any processor is specified

Bit mask If $(cpu \& (1 \ll i))$ is set (where i is an integer ranging from zero to 15 and representing a CPU) and it is the only bit set, the first process named by *name* that is running on CPU i is specified

If $(cpu \& (1 \ll i))$ is set and it is not the only bit set, the first process named by *name* that is currently running on any of the selected CPUs is specified

slot an integer value providing the unique frequency-based scheduler process identifier for the process for which values are to be cleared. This value is obtained when you make a call to **schedpgmadd** (see page 7-47 for an explanation of this subroutine). This value must be **-1** if you wish to identify the process only by specifying *name* and *cpu*.

istat an integer value indicating whether or not an error has occurred. A value of zero indicates that no error has occurred. A nonzero value indicates that an error of a specific type has occurred. Refer to the **pmc1rpgm(3f)** man page for a listing of the nonzero values that may be returned and the types of errors that they represent.

pmclrtable – Clear Values for Processor(s)

This subroutine is invoked to clear performance monitor values for FBS-scheduled processes on one or more specified processors on a selected scheduler.

NOTE

This subroutine will clear the total soft overrun count for all related processes.

Call Statement

```
call pmclrtable (schdle, cpucount, cpulist, istat)
integer schdle, cpucount, cpulist(cpucount), istat
```

Parameters

<i>schdle</i>	a unique, positive integer value representing the identifier for the frequency-based scheduler on which the processes are scheduled. You can obtain this value by making a call to fbconfigure(3f) (see page 7-7 for an explanation of this subroutine) or fbid(3f) (see page 7-12). If you wish to reference the frequency-based scheduler on which the calling process is scheduled without knowing its identifier, you can specify a value of -1 .
<i>cpucount</i>	an integer value indicating the number of elements contained in the array represented by <i>cpulist</i> .
<i>cpulist</i>	an array that consists of the number of elements specified by the <i>cpucount</i> parameter and contains one or more integer values indicating the processor or processors for which performance monitor values are to be cleared. Acceptable values and corresponding results are as follows: <ul style="list-style-type: none"> 0 Performance monitor values for FBS-scheduled processes executing on the processor from which the call is made are cleared -1 Performance monitor values for all processes on the scheduler Bit mask If (<i>cpu</i> & (1<<<i>i</i>)) is set (where <i>i</i> is an integer ranging from zero to 15 and representing a CPU), performance monitor values for processes executing on CPU <i>i</i> are cleared
<i>istat</i>	an integer value indicating whether or not an error has occurred. A value of zero indicates that no error has occurred. A nonzero value indicates that an error of a specific type has occurred. Refer to the pmclrtable(3f) man page for a listing of the nonzero values that may be returned and the types of errors that they represent.

pmmonitor – Start/Stop Performance Monitoring on Processor(s)

This subroutine is invoked to start or stop performance monitoring for FBS-scheduled processes on one or more specified processors on a selected scheduler.

Call Statement

```
call pmmonitor(schdle, pmflag, cpucount, cpulist, istat)
integer schdle, pmflag, cpucount, cpulist(cpucount), istat
```

Parameters

<i>schdle</i>	a unique, positive integer value representing the identifier for the frequency-based scheduler on which the processes are scheduled. You can obtain this value by making a call to fbconfigure(3f) (see page 7-7 for an explanation of this subroutine) or fbid(3f) (see page 7-12). If you wish to reference the frequency-based scheduler on which the calling process is scheduled without knowing its identifier, you can specify a value of -1 .
<i>pmflag</i>	an integer value indicating whether performance monitoring is to be started or stopped. A nonzero value indicates that performance monitoring is to be started. A zero value indicates that performance monitoring is to be stopped.
<i>cpucount</i>	an integer value indicating the number of elements in the array represented by <i>cpulist</i> .
<i>cpulist</i>	an array that consists of the number of elements specified by the <i>cpucount</i> parameter and contains one or more integer values indicating the processor or processors for which performance monitoring is to be started or stopped. Acceptable values and corresponding results are as follows: <ul style="list-style-type: none"> 0 Performance monitoring for FBS-scheduled processes executing on the processor from which the call is made is started or stopped -1 Performance monitoring for all processes on the scheduler is started or stopped Bit mask If $(cpu \& (1 \ll i))$ is set (where <i>i</i> is an integer ranging from zero to 15 and representing a CPU), performance monitoring for processes executing on CPU <i>i</i> is started or stopped
<i>istat</i>	an integer value indicating whether or not an error has occurred. A value of zero indicates that no error has occurred. A nonzero value indicates that an error of a specific type has occurred. Refer to the pmmonitor(3f) man page for a listing of the nonzero values that may be returned and the types of errors that they represent.

pmprogram – Start/Stop Performance Monitoring on a Process

This subroutine is invoked to start or stop performance monitoring for a particular process that has been scheduled on a frequency-based scheduler. You can identify the process by using one of the following methods:

- Specify the name of the process and the CPU(s) on which it is scheduled.
- Specify the process' frequency-based scheduler process identifier (slot number).
- Specify the name of the process, the CPU(s) on which it is scheduled, and its frequency-based scheduler process identifier.

Call Statement

```
call pmprogram(schdle, name, cpu, slot, pmflag, istat)
integer schdle, cpu, slot, pmflag, istat
character* (*) name
```

Parameters

<i>schdle</i>	a unique, positive integer value representing the identifier for the frequency-based scheduler on which the process is scheduled. You can obtain this value by making a call to fbsconfigure(3f) (see page 7-7 for an explanation of this subroutine) or fbsid(3f) (see page 7-12). If you wish to reference the frequency-based scheduler on which the calling process is scheduled without knowing its identifier, you can specify a value of -1 .
<i>name</i>	a standard path name identifying the process for which performance monitoring is to be started or stopped. A full or relative path name of up to 1024 characters can be specified. If this variable is filled with blanks, you must provide the frequency-based scheduler process identifier in the <i>slot</i> parameter.
<i>cpu</i>	an integer value indicating the processor(s) to be used in conjunction with the value of the <i>name</i> parameter to identify the process for which performance monitoring is to be started or stopped. Acceptable values and corresponding results are as follows: <ul style="list-style-type: none"> 0 The first process named by name that is currently running on the processor from which the call is made is specified -1 The first process named by name that is currently running on any processor is specified Bit mask If $(cpu \& (1 \ll i))$ is set (where <i>i</i> is an integer ranging from zero to 15 and representing a CPU) and it is the only bit set, the first process named by name that is currently running on CPU <i>i</i> is specified <p>If $(cpu \& (1 \ll i))$ is set and it is not the only bit set, the first process named by name that is currently running on any of the selected CPUs is specified</p>

<i>slot</i>	an integer value providing the unique frequency-based scheduler process identifier for the process for which performance monitoring is to be started or stopped. This value is obtained when you make a call to schedpgramadd (see page 7-47 for an explanation of this subroutine). This value must be - 1 if you wish to identify the process only by specifying <i>name</i> and <i>cpu</i> .
<i>pmflag</i>	an integer value indicating whether performance monitoring is to be started or stopped. A nonzero value indicates that performance monitoring is to be started. A zero value indicates that performance monitoring is to be stopped.
<i>istat</i>	an integer value indicating whether or not an error has occurred. A value of zero indicates that no error has occurred. A nonzero value indicates that an error of a specific type has occurred. Refer to the pmprogram(3f) man page for a listing of the nonzero values that may be returned and the types of errors that they represent.

pmqrycpu – Query Values for Selected Processor(s)

This subroutine is invoked to obtain performance monitor values for FBS-scheduled processes on one or more specified processors on a selected scheduler.

Call Statement

```
call pmqrycpu (schdle, cpu, bufsiz, buf, istat)
integer schdle, cpu, bufsiz, buf(bufsiz), istat
```

Parameters

<i>schdle</i>	a unique, positive integer value representing the identifier for the frequency-based scheduler on which the processes are scheduled. You can obtain this value by making a call to fbsconfigure(3f) (see page 7-7 for an explanation of this subroutine) or fbsid(3f) (see page 7-12). If you wish to reference the frequency-based scheduler on which the calling process is scheduled without knowing its identifier, you can specify a value of -1 .
<i>cpu</i>	an integer value indicating the processor(s) for which performance monitor values are to be obtained. Acceptable values and corresponding results are as follows: <ul style="list-style-type: none"> 0 Performance monitor values for FBS-scheduled processes executing on the processor from which the call is made are returned -1 Performance monitor values for all processes on the scheduler are returned Bit mask If $(cpu \& (1 \ll i))$ is set (where <i>i</i> is an integer ranging from zero to 15 and representing a CPU), performance monitor values for processes executing on CPU <i>i</i> are returned
<i>bufsiz</i>	an integer value indicating the size in 32-bit words of the array represented by <i>buf</i> . Because 16 words of information are returned for each process, it is recommended that this value be a multiple of 16.
<i>buf</i>	an array containing a series of 16 integer values for each FBS-scheduled process on the processor(s) specified with the <i>cpu</i> parameter. The number of processes for which these values are returned is bound by the value of the <i>bufsiz</i> parameter. If, for example, the value of <i>bufsiz</i> is 165, values for 10 processes will be returned. These values represent the performance monitoring information for the process(es). The type of information returned in each array element for a single process is presented in Table 7-8.

Table 7-8. Contents of Array Elements: pmqrycpu

Element	Contents
1	The process' frequency-based scheduler process identifier (slot number)
2	The amount of time that the process has spent running from the last time that it has been wakened by the scheduler until it has called fbwait (last time)
3	The number of times that the process has been wakened by the scheduler (total iterations, or cycles)
4	The number of seconds that the process has spent running in all cycles (total seconds). The total amount of time that the process has spent running is equal to the value of Element 4 plus the value of Element 5.
5	The additional number of microseconds that the process has spent running in all cycles (total microseconds). The total amount of time that the process has spent running is equal to the value of Element 4 plus the value of Element 5.
6	The number of hard frame overruns caused by the process
7	The least amount of time that the process has spent running in a cycle (minimum cycle time)
8	The number of the minor cycle in which the minimum cycle time has occurred (minimum cycle cycle)
9	The number of the major frame in which the minimum cycle time has occurred (minimum cycle frame)
10	The greatest amount of time that the process has spent running in a cycle (maximum cycle time)
11	The number of the minor cycle in which the maximum cycle time has occurred (maximum cycle cycle)
12	The number of the major frame in which the maximum cycle time has occurred (maximum cycle frame)
13	The least amount of time that the process has spent running during a major frame (minimum frame time)
14	The number of the major frame in which the minimum frame time has occurred (minimum frame frame)
15	The greatest amount of time that the process has spent running during a major frame (maximum frame time)
16	The number of the major frame in which the maximum frame time has occurred (maximum frame frame)

istat an integer value indicating whether or not an error has occurred. A value of zero indicates that no error has occurred. A nonzero value indicates that an error of a specific type has occurred. Refer to the **pmqrycpu(3f)** man page for a listing of the nonzero values that may be returned and the types of errors that they represent.

pmqrylist – Query Values for a List of Processes

This subroutine is invoked to obtain performance monitor values for a list of processes scheduled on a frequency-based scheduler.

Call Statement

```
call pmqrylist (schdle, slotcount, slotlist, bufsiz, buf, istat)
integer schdle, slotcount, slotlist(slotcount), bufsiz, buf(bufsiz), istat
```

Parameters

<i>schdle</i>	a unique, positive integer value representing the identifier for the frequency-based scheduler for which performance monitor values are requested. You can obtain this value by making a call to fbsconfigure(3f) (see page 7-7 for an explanation of this subroutine) or fbsid(3f) (see page 7-12). If you wish to reference the frequency-based scheduler on which the calling process is scheduled without knowing its identifier, you can specify a value of -1 .
<i>slotcount</i>	an integer value indicating the number of frequency-based scheduler process identifiers contained in the array represented by <i>slotlist</i> .
<i>slotlist</i>	an array that consists of the number of elements specified by the <i>slotcount</i> parameter and contains one or more integer values indicating the frequency-based scheduler process identifiers for which performance monitor values are to be returned.
<i>bufsiz</i>	an integer value indicating the size in 32-bit words of the array represented by <i>buf</i> . Because 15 words of information are returned for each process, it is recommended that this value be a multiple of 15.
<i>buf</i>	an array containing a series of 15 integer values for each FBS-scheduled process. The number of processes for which these values are returned is bound by the value of the <i>bufsiz</i> parameter. If, for example, the value of <i>bufsiz</i> is 155, values for 10 processes will be returned. These values represent the performance monitoring information for the processes. The type of information returned in each array element for a single process is presented in Table 7-9.
<i>istat</i>	an integer value indicating whether or not an error has occurred. A value of zero indicates that no error has occurred. A nonzero value indicates that an error of a specific type has occurred. Refer to the pmqrylist(3f) man page for a listing of the nonzero values that may be returned and the types of errors that they represent.

Table 7-9. Contents of Array Elements: pmqrylist

Element	Contents
1	The amount of time that the process has spent running from the last time that it has been wakened by the scheduler until it has called fbswait (last time)
2	The number of times that the process has been wakened by the scheduler (total iterations, or cycles)
3	The number of seconds that the process has spent running in all cycles (total seconds). The total amount of time that the process has spent running is equal to the value of Element 3 plus the value of Element 4.
4	The additional number of microseconds that the process has spent running in all cycles (total microseconds). The total amount of time that the process has spent running is equal to the value of Element 3 plus the value of Element 4.
5	The number of hard frame overruns caused by the process
6	The least amount of time that the process has spent running in a cycle (minimum cycle time)
7	The number of the minor cycle in which the minimum cycle time has occurred (minimum cycle cycle)
8	The number of the major frame in which the minimum cycle time has occurred (minimum cycle frame)
9	The greatest amount of time that the process has spent running in a cycle (maximum cycle time)
10	The number of the minor cycle in which the maximum cycle time has occurred (maximum cycle cycle)
11	The number of the major frame in which the maximum cycle time has occurred (maximum cycle frame)
12	The least amount of time that the process has spent running during a major frame (minimum frame time)
13	The number of the major frame in which the minimum frame time has occurred (minimum frame frame)
14	The greatest amount of time that the process has spent running during a major frame (maximum frame time)
15	The number of the major frame in which the maximum frame time has occurred (maximum frame frame)

pmqrypnm – Query Values for a Selected Process

This subroutine is invoked to obtain performance monitor values for a particular process scheduled on a frequency-based scheduler.

Call Statement

```
call pmqrypnm (schdle, name, cpu, slot, last, iter, totsec, totusec, over, minc,
               minctc, minctf, maxc, maxctc, maxctf, minf, minftf, maxf, maxftf, istat)
integer schdle, cpu, slot, last, iter, totsec, totusec, over, minc, minctc, minctf, maxc,
        maxctc, maxctf, minf, minftf, maxf, maxftf, istat
character* (*) name
```

Parameters

<i>schdle</i>	a unique, positive integer value representing the identifier for the frequency-based scheduler for which performance monitor values are requested. You can obtain this value by making a call to fbsconfigure (3f) (see page 7-7 for an explanation of this subroutine) or fbsid (3f) (see page 7-12). If you wish to reference the frequency-based scheduler on which the calling process is scheduled without knowing its identifier, you can specify a value of -1 .
<i>name</i>	a standard path name identifying the process for which performance monitoring values are to be returned. A full or relative path name of up to 1024 characters can be specified. If this variable is filled with blanks, you must provide the frequency-based scheduler process identifier in the <i>slot</i> parameter.
<i>cpu</i>	<p>an integer value indicating the processor(s) to be used in conjunction with the value of the <i>name</i> parameter to identify the process for which performance monitoring values are to be returned. Acceptable values and corresponding results are as follows:</p> <p>0 The first process named by <i>name</i> that is currently running on the processor from which the call is made is specified</p> <p>-1 The first process named by <i>name</i> that is currently running on any processor is specified</p> <p>Bit mask If (<i>cpu</i> & (1<<<i>i</i>)) is set (where <i>i</i> is an integer ranging from zero to 15 and representing a CPU) and it is the only bit set, the first process named by <i>name</i> that is currently running on CPU <i>i</i> is specified</p> <p> If (<i>cpu</i> & (1<<<i>i</i>)) is set and it is not the only bit set, the first process named by <i>name</i> that is currently running on any of the selected CPUs is specified</p>
<i>slot</i>	an integer value providing the unique frequency-based scheduler process identifier for the process for which performance monitoring values are to be returned. This value is obtained when you make a call to schdpnmadd (see page 7-47 for an explanation of this subroutine). This value must be

	– 1 if you wish to identify the process only by specifying <i>name</i> and <i>cpu</i> .
<i>last</i>	an integer value indicating the amount of time that the process has spent running from the last time that it has been wakened by the scheduler until it has called fbwait (last time).
<i>iter</i>	an integer value indicating the number of times that the process has been wakened by the frequency-based scheduler since the last time that performance monitor values have been cleared and performance monitoring has been enabled (total iterations, or cycles).
<i>totsec</i>	an integer value indicating the number of seconds that the process has spent running in all cycles (total time in seconds). The total amount of time that the process has spent running is equal to the value of <i>totsec</i> plus <i>totusec</i> .
<i>totusec</i>	an integer value indicating the additional number of microseconds that the process has spent running in all cycles (total time in microseconds). The total amount of time that the process has spent running is equal to the value of <i>totsec</i> plus <i>totusec</i> .
<i>over</i>	an integer value indicating the number of times that the process has caused a hard frame overrun.
<i>minc</i>	an integer value indicating the least amount of time that the process has spent running in a cycle (minimum cycle time).
<i>minctc</i>	an integer value indicating the number of the minor cycle in which the minimum cycle time has occurred (minimum cycle cycle).
<i>minctf</i>	an integer value indicating the number of the major frame in which the minimum cycle time has occurred (minimum cycle frame).
<i>maxc</i>	an integer value indicating the greatest amount of time that the process has spent running in a cycle (maximum cycle time).
<i>maxctc</i>	an integer value indicating the number of the minor cycle in which the maximum cycle time has occurred (maximum cycle cycle).
<i>maxctf</i>	an integer value indicating the number of the major frame in which the maximum cycle time has occurred (maximum cycle frame).
<i>minf</i>	an integer value indicating least amount of time that the process has spent running in a major frame (minimum frame time).
<i>minfff</i>	an integer value indicating the number of the major frame in which the minimum frame time has occurred (minimum frame frame).

<i>maxf</i>	an integer value indicating the greatest amount of time that the process has spent running in a major frame (maximum frame time).
<i>maxftf</i>	an integer value indicating the number of the major frame in which the maximum frame time has occurred (maximum frame frame).
<i>istat</i>	an integer value indicating whether or not an error has occurred. A value of zero indicates that no error has occurred. A nonzero value indicates that an error of a specific type has occurred. Refer to the pmqrypgm(3f) man page for a listing of the nonzero values that may be returned and the types of errors that they represent.

pmquerytimer – Query Performance Monitor Mode

This subroutine is invoked to determine whether performance monitor timing values include or exclude time spent servicing interrupts.

Call Statement

```
call pmquerytimer(mode)  
integer mode
```

Parameter

<i>mode</i>	a value indicating whether performance monitor timing values include or exclude time spent servicing interrupts. A value of one indicates that interrupt time is included. A value of zero indicates that interrupt time is excluded. A value of one or zero is returned if the call is successful; a negative value is returned to indicate that an error of a specific type has occurred. Refer to the pmquerytimer(3f) man page for a listing of the values that may be returned and the types of errors that they represent.
-------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

pmselect – Select Performance Monitor Mode

This subroutine is invoked to select the timing mode under which the performance monitor is to run. The timing mode can be set to include or exclude time spent servicing interrupts.

When interrupt time is included, a process' user and system times will total the elapsed time which accrues when the process is the currently running process, including all time spent servicing interrupts. The time spent servicing interrupts is added to the process' system time.

When excluding interrupt time, a process' user and system times will total the time which accrues when the process is the currently running process, excluding all time spent servicing interrupts.

Note that to set the timing mode, the calling process must have the capabilities associated with the *fbScheduler* role (see Chapter 1).

Call Statement

```
call pmselect(mode, istat)
integer mode, istat
```

Parameters

<i>mode</i>	an integer value indicating whether time spent servicing interrupts is to be included in or excluded from performance monitor timing values. A nonzero value indicates that interrupt time is to be included. A value of zero indicates that interrupt time is to be excluded.
<i>istat</i>	an integer value indicating whether or not an error has occurred. A value of zero indicates that no error has occurred. A nonzero value indicates that an error of a specific type has occurred. Refer to the pmselect(3f) man page for a listing of the nonzero values that may be returned and the types of errors that they represent.

A

Example rtcp Script

This appendix contains an example of a script that can be invoked at the system command prompt to execute **rtcp** commands.

This script illustrates using commands to:

- configure a scheduler and schedule programs on it (**cs, sp**)
- view information about the frequency-based scheduled processes (**vp**)
- view the scheduler configuration (**vs**)
- attach, set, and start a real-time clock (**ats, stc, rc**)
- clear performance monitor values (**cpm**)
- start performance monitoring and frequency-based scheduling (**pm, start**)
- view the minor cycle and major frame count (**vc**)
- stop performance monitoring; stop the real-time clock and the frequency-based scheduler (**pm, sc, stop**)
- detach the real-time clock (**dts**)
- remove the scheduler and all scheduled processes (**rms**)

```
rtcp cs -s37 -C10 -I664 -M5 -N10
rtcp sp -s37 -n ./program1 -c0 -bF -p19 -f4 -m0 -ohalt
rtcp sp -s37 -n ./program2 -c1 -bF -p19 -f2 -m1 -ohalt
rtcp vp -s37 -c*
rtcp vs -s37
rtcp ats -s37 -d/dev/rcim/rtc0
rtcp stc -s37 -O10000 -D1
rtcp rc -s37
rtcp cpm -s37 -c*
rtcp pm -s37 -c* -PON
rtcp start -s37
rtcp vc -s37
rtcp pm -s37 -c* -POFF
rtcp sc -s37
rtcp stop -s37
rtcp dts -s37
rtcp rms -s37 -a
```


B

rtcp Error Messages

This appendix contains descriptions of the errors that may be reported by the real-time command processor, **rtcp**.

Table B-1. rtcp Errors

Error	Description
-2	Interrupt device not specified
-3	Both EOC and interrupt device specified
-4	Process not specified
-5	Invalid rtcp command specified
-6	Invalid help command specified
-7	Invalid cpu (-c) parameter
-8	Invalid frequency (-f) parameter
-9	Invalid halt flag (-h) parameter
-10	Invalid start cycle (-m) parameter
-11	Invalid priority (-p) parameter
-12	Invalid cycle count (-C) parameter
-13	Invalid clock tick duration (-D) parameter
-14	Invalid process per cycle (-M) parameter
-15	Invalid process per frequency-based scheduler (-N) parameter
-16	Invalid clock ticks per cycle (-O) parameter
-17	Invalid PM flag (-P) parameter
-18	Invalid parameter specified
-22	rj file not specified
-23	Invalid rj file specified
-24	Invalid pm viewing mode specified
-25	Invalid pm timing mode specified
-26	Unable to change timing mode to exclude interrupt time
-27	Invalid scheduling policy specified
-28	Exit rtcp
-29	Invalid soft overrun limit (-L) parameter
-30	Number of hosts (-H) exceeds limit
-31	Hostname(s) (-H) not specified

C

Example C Interface

This appendix contains an example program that illustrates use of the C library interface to the Frequency-Based Scheduler and the Performance Monitor.

The program performs the following tasks:

- configures a scheduler
- schedules programs on the scheduler
- attaches, sets, and starts a real-time clock
- starts performance monitoring for each frequency-based scheduled process
- starts frequency-based scheduling
- obtains performance monitor values for the frequency-based scheduled processes
- monitors a processor's idle time

The example program begins on the next page.

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sched.h>
#include <sys/stat.h>
#include <fbsched.h>
#include <errno.h>

#define NUM_PROCS 4
#define START 1
#define STOP 0
#define PROGRAM1 "./prog1"
#define PROGRAM2 "./prog2"
#define IDLE0 "idle0"
#define IDLE1 "idle1"
#define CLOCK1 "/dev/rcim/rtc0"

void
cleanup(int fbs_id)
{
    fbsrunrtc(fbs_id, 0);          /* stop timing device */
    fbsdetach(fbs_id);           /* detach timing device */
    fbsremove(fbs_id, -1);       /* remove frequency-based scheduler */
}

int main()
{
    struct fbsconfig_ds    fbs_buf;
    struct pgm2_ds         sched_buf;
    struct pmqry_ds        pm_buf[NUM_PROCS];
    struct fbsinfo_ds     info_buf;
    struct fbcycle_ds     cycle_buf;
    int idle0_fpid;        /* fpid for idle on cpu 0 */
    int idle1_fpid;        /* fpid for idle on cpu 1 */
    int pgm1_fpid;         /* fpid for testprogram 1 */
    int pgm2_fpid;         /* fpid for testprogram 2 */
    int cpu;
    int istat;
    int pmflg;
    int intrflg;
    int i;
    int count;
    int resolution;
    char name[1024];       /* program's full or orelative path name */
    char *pgmname[4];     /* program's name */
    FILE *fp;

    /* Open file to store performance information */
    fp= fopen("pmresults", "w+");

    if (fp == NULL)
        printf("open failed errno = %d\n", errno);

    /*
     * CONFIGURE SCHEDULER

```

```

*/
fbs_buf.key = 37;          /* scheduler key */
fbs_buf.cycles = 10;     /* number of cycles per frame */
fbs_buf.progs = 5;      /* max. number of programs per cycle */
fbs_buf.max = 10;      /* max. number of programs allowed on the fbs */
fbs_buf.reset = -1;     /* kill/remove processes currently scheduled */

/* owner/group read/write */
fbs_buf.configflg = 0664;

istat = fbsconfigure(&fbs_buf);

if (istat != 0) {
    printf("could not configure scheduler: errno = %d\n" ,errno);
    exit(1);
}

/*
 * SCHEDULE test program1 PROGRAM1
 */
sched_buf.name_ptr = PROGRAM1;
sched_buf.cid = SCHED_FIFO; /* first-in-first out (FIFO) policy */
sched_buf.prior = 19;
sched_buf.param = 0;        /* optional initiation parameter */
sched_buf.period = 4;      /* time between wakeups */
sched_buf.cycle = 0;       /* starting base cycle */
sched_buf.halt = 0;        /* halt on overrun */

/* Set cpu mask to schedule testprogram1 on cpu 0 */
sched_buf.cpu = 1;

istat = sched_pgmadd(fbs_buf.fbs_id, &sched_buf);
printf("fbsid = %d\n", fbs_buf.fbs_id);

if (istat != 0) {
    printf("could not schedule %s on cpu %d : errno = %d\n",
           sched_buf.name_ptr, sched_buf.cpu>>1, errno);
    cleanup(fbs_buf.fbs_id);
    return(1);
}

pgm1_fpid = sched_buf.fpid;
printf("pgm1 fpid = %d\n", pgm1_fpid);

/*
 * SCHEDULE test program2 PROGRAM2
 */
sched_buf.name_ptr = PROGRAM2;
sched_buf.prior = 19;
sched_buf.cid = SCHED_FIFO; /* first-in-first out (FIFO) policy */
sched_buf.param = 0;        /* optional initiation parameter */
sched_buf.period = 2;      /* time between wakeups */
sched_buf.cycle = 1;       /* starting base cycle */
sched_buf.halt = 0;        /* halt on overrun */

/* Set cpu mask to schedule testprogram2 on cpu 0 */
sched_buf.cpu = 1;

```

```
istat = sched_pgmadd(fbs_buf.fbs_id, &sched_buf);
printf("fbsid = %d\n", fbs_buf.fbs_id);

if (istat != 0) {
    printf("could not schedule %s on cpu %d : errno = %d\n",
        sched_buf.name_ptr, sched_buf.cpu>>1, errno);
    cleanup(fbs_buf.fbs_id);
    return(1);
}

pgm2_fpid = sched_buf.fpid;
printf("pgm2 fpid = %d\n", pgm2_fpid);

/*
 * SCHEDULE IDLE ON CPU 0
 *
 * The only parameter required for /idle is the CPU.
 */

sched_buf.name_ptr = "/idle";
sched_buf.cid = SCHED_FIFO; /* first-in-first out (FIFO) policy */
sched_buf.prior = 20;
sched_buf.param = 0; /* optional initiation parameter */
sched_buf.period = 1; /* time between wakeups */
sched_buf.cycle = 0; /* starting base cycle */
sched_buf.halt = 0; /* halt on overrun */

/* Set cpu mask to schedule idle on cpu 0 */
sched_buf.cpu = 1; /* mask = 1<<cpu */

istat = sched_pgmadd(fbs_buf.fbs_id, &sched_buf);
printf("fbsid = %d\n", fbs_buf.fbs_id);

if (istat != 0) {
    printf("could not schedule %s on cpu %d : errno = %d\n",
        sched_buf.name_ptr, sched_buf.cpu, errno);
    cleanup(fbs_buf.fbs_id);
    return(1);
}

idle0_fpid = sched_buf.fpid;
printf("idle0 fpid = %d\n", idle0_fpid);

/*
 * SCHEDULE IDLE ON CPU 1
 *
 * The only parameter required for /idle is the CPU.
 */

sched_buf.name_ptr = "/idle";
sched_buf.prior = 20;
sched_buf.cid = SCHED_FIFO; /* first-in-first out (FIFO) policy */
sched_buf.param = 0; /* optional initiation parameter */
sched_buf.period = 1; /* time between wakeups */
sched_buf.cycle = 0; /* starting base cycle */
sched_buf.halt = 0; /* halt on overrun */

/* Set cpu mask to schedule idle on cpu 1 */
sched_buf.cpu = 2; /* mask = 1<<cpu */
```

```

istat = sched_pgmadd(fbs_buf.fbs_id, &sched_buf);

if (istat != 0) {
    printf("could not schedule %s on cpu %d : errno = %d\n",
        sched_buf.name_ptr, sched_buf.cpu, errno);
    cleanup(fbs_buf.fbs_id);
    return(1);
}

idle1_fpid = sched_buf.fpid;
printf("idle1 fpid = %d\n", idle1_fpid);

/*
 * ATTACH/SET REAL-TIME CLOCK
 * Set the clock to interrupt every 10,000 usecs.
 */
count = 10000;
resolution = 1;
istat = fbsattach(fbs_buf.fbs_id, CLOCK1);

if (istat != 0) {
    printf("could not attach timing source: errno = %d\n", errno);
    cleanup(fbs_buf.fbs_id);
    return(1);
}

istat = fbssetrtc(fbs_buf.fbs_id, count, resolution);

if (istat != 0) {
    printf("could not set rtc: errno = %d\n", errno);
    cleanup(fbs_buf.fbs_id);
    return(1);
}

istat = fbsrunrtc(fbs_buf.fbs_id, START);

if (istat != 0) {
    printf("could not start rtc: errno = %d\n", errno);
    cleanup(fbs_buf.fbs_id);
    return(1);
}

/*
 * START PERFORMANCE MONITORING
 */
pmflg = 1;

/* zero out the "name" variable (fpid must be specified).
 * Ultimately, the "name" variable can be used to store
 * the full or relative path name of a test program.
 */
bzero(name, sizeof(name));
cpu = 0; /* not used if fpid is being used */

/* start performance monitoring for testprogram1 */

```

```
istat = pmprogram(fbs_buf.fbs_id, name, cpu, pgm1_fpid, pmflg);

if (istat != 0) {
    printf("could not start pm for testprogram1 : errno = %d\n", errno);
    cleanup(fbs_buf.fbs_id);
    return(1);
}

/* start performance monitoring for testprogram2 */
istat = pmprogram(fbs_buf.fbs_id, name, cpu, pgm2_fpid, pmflg);

if (istat != 0) {
    printf("could not start pm for testprogram2 : errno = %d\n", errno);
    cleanup(fbs_buf.fbs_id);
    return(1);
}

/* start performance monitoring for idle on cpu 0 */
istat = pmprogram(fbs_buf.fbs_id, name, cpu, idle0_fpid, pmflg);

if (istat != 0) {
    printf("could not start pm for idle0 : errno = %d\n", errno);
    cleanup(fbs_buf.fbs_id);
    return(1);
}

/* start performance monitoring for idle on cpu 1 */
istat = pmprogram(fbs_buf.fbs_id, name, cpu, idle1_fpid, pmflg);

if (istat != 0) {
    printf("could not start pm for idle1 : errno = %d\n", errno);
    cleanup(fbs_buf.fbs_id);
    return(1);
}

/*
 * START SCHEDULING
 */
intrflg = 1;
istat = fbsintrpt(fbs_buf.fbs_id, intrflg);

if (istat != 0) {
    printf("could not start scheduler : errno = %d\n", errno);
    cleanup(fbs_buf.fbs_id);
    return(1);
}

/*
 * QUERY PERFORMANCE MONITOR VALUES
 * 1 second = 100 cycles  1 minute = 600 frames
 * Query once per second for 1 minute
 */
pm_buf[0].fpid = pgm1_fpid;  pgmname[0] = PROGRAM1;
pm_buf[1].fpid = pgm2_fpid;  pgmname[1] = PROGRAM2;
pm_buf[2].fpid = idle0_fpid; pgmname[2] = IDLE0;
pm_buf[3].fpid = idle1_fpid; pgmname[3] = IDLE1;
```



```

sleep(1); /* sleep for a while */

printf("Please wait, performance information being gathered. \n");

cycle_buf.cframe = 0;
while ((cycle_buf.cframe < 600)) {
    istat = fbscopycycle(fbs_buf.fbs_id, &cycle_buf);
    if (istat != 0) {
        printf("could not query fbscopycycle: errno = %d\n", errno);
        cleanup(fbs_buf.fbs_id);
        return(1);
    }
    istat = pmqrylist(fbs_buf.fbs_id, pm_buf, NUM_PROCS);
    if (istat != 0) {
        printf("could not query process: errno = %d\n", errno);
        cleanup(fbs_buf.fbs_id);
        return(1);
    }
    fprintf(fp, "*****\n");
    fprintf(fp, "\tFRAME=%d, CYCLE=%d\n", cycle_buf.cframe,
            cycle_buf.ccycle);
    fprintf(fp, "*****\n");

    /* Write performance information for each
     * process into a file
     */
    for (i = 0; i < NUM_PROCS; i++) {
        fprintf(fp, "pgm=%s, fpid=%d\n", pgmname[i],
                pm_buf[i].fpid);
        fprintf(fp, "-----\n");
        fprintf(fp, "last cycle secs:nsecs = %d:%d\n",
                pm_buf[i].lastcyc_tm.tv_sec,
                pm_buf[i].lastcyc_tm.tv_nsec);
        fprintf(fp, "total cycles = %d\n", pm_buf[i].tot_cycles);
        fprintf(fp, "total cycles secs:nsecs = %d:%d\n",
                pm_buf[i].tot_cycles_tm.tv_sec,
                pm_buf[i].tot_cycles_tm.tv_nsec);
        fprintf(fp, "overruns = %d\n", pm_buf[i].overruns);
        fprintf(fp, "mincyc secs:nsecs = %d:%d\n",
                pm_buf[i].mincyc_tm.tv_sec,
                pm_buf[i].mincyc_tm.tv_nsec);
        fprintf(fp, "mincyc_cycle = %d\n",
                pm_buf[i].mincyc_cycle);
        fprintf(fp, "mincyc_frame = %d\n",
                pm_buf[i].mincyc_frame);
        fprintf(fp, "maxcyc secs:nsecs = %d:%d\n",
                pm_buf[i].maxcyc_tm.tv_sec,
                pm_buf[i].maxcyc_tm.tv_nsec);
        fprintf(fp, "maxcyc_cycle = %d\n",
                pm_buf[i].maxcyc_cycle);
        fprintf(fp, "maxcyc_frame = %d\n",
                pm_buf[i].maxcyc_frame);
        fprintf(fp, "minframe secs:nsecs = %d:%d\n",
                pm_buf[i].minframe_tm.tv_sec,
                pm_buf[i].minframe_tm.tv_nsec);
        fprintf(fp, "minframe = %d\n", pm_buf[i].minframe);
        fprintf(fp, "maxframe secs:nsecs = %d:%d\n",
                pm_buf[i].maxframe_tm.tv_sec,
                pm_buf[i].maxframe_tm.tv_nsec);
    }
}

```

```
                fprintf(fp, "maxframe = %d\n", pm_buf[i].maxframe);
                fprintf(fp, "\n");
            }
            sleep(1);          /* sleep for a while */
        }

        printf("Performance data has been gathered. \n");

        if (istat != 0)
            printf("istat = %d\n", istat);

        /* Stop PM on CPUs 0 and 1 */
        pmmonitor(fbs_buf.fbs_id, 0, 3);

        /* Stop the clock */
        istat = fbsrunrtc(fbs_buf.fbs_id, STOP);

        if (istat != 0) {
            printf("could not stop rtc: errno = %d\n", errno);
            cleanup(fbs_buf.fbs_id);
            return(1);
        }

        /* Detach from the fbs */
        istat = fbsdetach(fbs_buf.fbs_id);

        if (istat != 0) {
            printf("could not dettach timing source: errno = %d\n", errno);
            cleanup(fbs_buf.fbs_id);
            return(1);
        }

        /* Remove the fbs */
        istat = fbsremove(fbs_buf.fbs_id, -1);

        if (istat != 0) {
            printf("could not remove timing source: errno = %d\n", errno);
            cleanup(fbs_buf.fbs_id);
            return(1);
        }
        cleanup(fbs_buf.fbs_id);
        return(0);
    }
}
```

Glossary

end-of-cycle scheduling

A form of frequency-based scheduling in which scheduling is triggered when the last process scheduled to execute in the current minor cycle of the current major frame completes its processing.

frame overrun

The condition that occurs when a frequency-based scheduled process does not finish its processing before it is scheduled to run again. This applies to interrupt scheduling only, not end-of-cycle scheduling.

frequency

When applied to the frequency-based scheduler, indicates a time interval. This interval can be based on high-resolution clocks, an external interrupt source or completion of a cycle.

frequency-based scheduler

A high resolution task synchronization mechanism that enables processes to run at user-specified frequencies.

hard overrun

A type of frame overrun that is a catastrophic failure of the scheduled process.

idle time

Time during which the CPU is not busy.

iteration

One instance of a process being awakened by a frequency-based scheduler.

last time

A value returned by the performance monitor indicating the amount of time that a frequency-based scheduled process has spent running from the last time that it has been awakened by the scheduler until it has called `fbwait(3)`.

major frame

One pass through all of the minor cycles with which a frequency-based scheduler is configured. A major frame has associated with it a duration, which is obtained by multiplying the duration of a minor cycle by the number of minor cycles per major frame.

maximum cycle cycle

A value returned by the performance monitor indicating the number of the minor cycle in which the maximum cycle time has occurred.

maximum cycle frame

A value returned by the performance monitor indicating the number of the major frame in which the maximum cycle time has occurred.

maximum cycle time

A value returned by the performance monitor indicating the greatest amount of time in microseconds that a frequency-based scheduled process has spent running in a cycle.

maximum frame frame

A value returned by the performance monitor indicating the number of the major frame in which the maximum frame time has occurred.

maximum frame time

A value returned by the performance monitor indicating the greatest amount of time in microseconds that a frequency-based scheduled process has spent running during a major frame.

minimum cycle cycle

A value returned by the performance monitor indicating the number of the minor cycle in which the minimum cycle time has occurred.

minimum cycle frame

A value returned by the performance monitor indicating the number of the major frame in which the minimum cycle time has occurred.

minimum cycle time

A value returned by the performance monitor indicating the least amount of time in microseconds that a frequency-based scheduled process has spent running in a cycle.

minimum frame frame

A value returned by the performance monitor indicating the number of the major frame in which the minimum frame time has occurred.

minimum frame time

A value returned by the performance monitor indicating the least amount of time in microseconds that a frequency-based scheduled process has spent running during a major frame.

minor cycle

The smallest unit of frequency maintained by a frequency-based scheduler. A minor cycle has associated with it a duration, which is the time that elapses between interrupts generated by the timing source attached to the scheduler. If the timing source is a real-time clock, the minor cycle duration is defined by specifying the number of clock counts per minor cycle and the number of microseconds per clock count.

number of overruns

A value returned by the performance monitor indicating the number of times that a frequency-based scheduled process has caused a frame overrun.

performance monitor

A mechanism that makes it possible to monitor use of the CPU by processes that are scheduled on a frequency-based scheduler.

period

A frequency-based scheduler parameter that specifies how often a specified program is to be awakened in each major frame. A period of one indicates that the program is to be awakened every minor cycle; a period of two indicates that it is to be awakened once every two minor cycles; and so on.

privilege

A mechanism through which processes are allowed to perform sensitive operations or override system restrictions.

process dispatch latency

The time that elapses from the occurrence of an external event, which is signified by an interrupt, until the process waiting for that external event executes its first instruction in user mode.

scheduler key

A user-supplied numeric identifier for a frequency-based scheduler.

shielded processor

A CPU that is responsible for running high-priority tasks that are protected from the unpredictable processing associated with interrupts and system daemons.

soft overrun

A type of frame overrun that is a catastrophic failure only if the process has reached its limit on the number of soft overruns tolerated. Each scheduled process has a soft overrun limit, defaulting to 0.

Some soft overruns result from random, unpredictable, or external events unlikely to recur. Other soft overruns result from only minor frame overruns. Soft overruns give the scheduled process a chance to recover from a frame overrun and return to synchronization.

Soft overruns apply to interrupt scheduling only, not end-of-cycle scheduling.

spare time

Processor time that is composed of the following: (1) idle time, (2) CPU time of processes that are not scheduled on a frequency-based scheduler, and (3) CPU time of frequency-based scheduled processes for which performance monitoring has not been enabled.

starting base cycle

A frequency-based scheduler scheduling parameter that specifies the first minor cycle in which a frequency-based scheduled process is to be awakened in each major frame.

timing mode

The mode under which the performance monitor runs. It specifies whether time spent servicing interrupts is to be included in or excluded from performance monitor timing values.

total iterations

A value returned by the performance monitor indicating the number of times that a frequency-based scheduled process has been awakened by the scheduler.

total time

A value returned by the performance monitor indicating the total amount of time that a frequency-based scheduled process has spent running in all cycles.

unscheduled process

A process that is not awakened by the frequency-based scheduler and does not call **fbswait**; it is not scheduled to run at a certain frequency.

A

ats command 5-11
attach a timing source 5-11, 6-6, 7-6

C

C library call sequence
 Frequency-Based Scheduler 6-4
 Performance Monitor 6-48

C library routines

 fbsaccess 6-5
 fbsattach 6-6
 fbsavail 6-7
 fbsconfigure 6-8
 fbscycle 6-11
 fbsdetach 6-12
 fbsdir 6-13
 fbsgetpid 6-14
 fbsgetrtc 6-15
 fbsid 6-16
 fbsinfo 6-17
 fbsintrpt 6-19
 fbsremove 6-20
 fbsresume 6-21
 fbsrunrtc 6-22
 fbsschedself 6-23
 fbssetrtc 6-25
 fbstrig 6-26
 fbswait 2-5, 6-27
 namepid 6-28
 nametopid 6-28
 pgmremove 6-29
 pgmtrigger 6-31
 pmclrpgm 6-49
 pmclrtable 6-51
 pmmonitor 6-52
 pmprogram 6-53
 pmqrycpu 6-55
 pmqrylist 6-58
 pmqrypgm 6-60
 pmqrytimer 6-63
 pmselect 6-64
 sched_fbsqry 6-32

 sched_pgm_set_soft_overrun_limit 6-34
 sched_pgm_soft_overrun_query 6-36
 sched_pgmadd 6-38
 sched_pgmqry 6-41
 sched_pgmresched 6-44

C program example C-1

change

 CPU bias 5-32, 6-38, 7-47
 process priority 5-32, 6-38, 6-44, 7-47
 scheduler permissions 5-12, 6-5, 7-5
 scheduling policy 5-32, 6-38, 6-44, 7-47
 timing mode 5-42, 6-64, 7-71

chs command 5-12

clear performance monitor values 5-38, 6-49, 6-51,
 7-57, 7-59

configuration parameters 1-3

configure a scheduler 5-13, 6-8, 7-7

cpm command 5-38

CPU bias 4-5, 6-38, 7-47

cs command 5-13

D

debug a process 2-6

detach a timing source 5-15, 6-12, 7-10

direct mode 5-2, 5-3

disable end-of-cycle scheduling 5-15, 6-12, 7-10

distributed interrupt 3-3

dts command 5-15

E

edge-triggered interrupt

 attach 5-11, 6-6, 7-6
 detach 5-15, 6-12, 7-10
 device special files 3-3
 overview 3-3
 user interface 3-4

end-of-cycle scheduling

 definition 2-2
 disable 5-15, 6-12, 7-10
 enable 5-11, 6-6, 7-6

error messages B-1

- ex command 5-47
- examples
 - C program C-1
 - process scheduling 2-3, 2-4
 - rtcp script A-1
 - simulation optimization 4-5
- execution modes 5-2–5-4
- exit rtcp 5-47

F

- fbsaccess 6-5, 7-5
- fbsattach 6-6, 7-6
- fbsavail 6-7
- fbscheduser 1-4
- fbsconfigure 6-8, 7-7
- fbcycle 6-11, 7-9
- fbsdetach 6-12, 7-10
- fbsdir 6-13
- fbsgetpid 6-14
- fbsgetrtc 6-15, 7-11
- fbsid 6-16, 7-12
- fbsinfo 6-17, 7-13
- fbsintrpt 6-19, 7-15
- fbsquery 7-16
- fbsremove 6-20, 7-19
- fbsresume 6-21, 7-20
- fbsrunrtc 6-22, 7-22
- fbsschedself 6-23, 7-23
- fbssetrtc 6-25, 7-25
- fbstrig 6-26
- fbswait 2-5, 6-27, 7-26
- FORTRAN library call sequence
 - Frequency-Based Scheduler 7-4
 - Performance Monitor 7-56
- FORTRAN library routines
 - fbsaccess 7-5
 - fbsattach 7-6
 - fbsconfigure 7-7
 - fbcycle 7-9
 - fbsdetach 7-10
 - fbsgetrtc 7-11
 - fbsid 7-12
 - fbsinfo 7-13
 - fbsintrpt 7-15
 - fbsquery 7-16
 - fbsremove 7-19
 - fbsresume 7-20
 - fbsrunrtc 7-22
 - fbsschedself 7-23
 - fbssetrtc 7-25
 - fbswait 7-26
 - nametopid 7-27

- pgmquery 7-28
- pgmremove 7-30
- pgmreschedule 7-32
- pgmschedule 7-35
- pgmstat 7-38
- pgmtrigger 7-40
- pmclrpqm 7-57
- pmclrtable 7-59
- pmmonitor 7-60
- pmprogram 7-61
- pmqrycpu 7-63
- pmqrylist 7-65
- pmqrypgm 7-67
- pmquerytimer 7-70
- pmselect 7-71
- rtparm 7-41
- sched_pgm_set_soft_overrun_limit 7-42
- sched_pgm_soft_overrun_query 7-43
- schedfbsqry 7-44
- schedpgmadd 7-47
- schedpgmqry 7-49
- schedpgmresched 7-52
- frame overruns, see overruns
- frequency 2-2
- Frequency-Based Scheduler
 - C library call sequence 6-4
 - C library routines 6-2, 6-5
 - configuration 1-3
 - FORTRAN library call sequence 7-4
 - FORTRAN library routines 7-5
 - overview 1-1, 2-1
 - privileges 1-4
 - rtcp command sequence 5-9
 - rtcp commands 5-7, 5-8, 5-10
 - timing sources, see timing source
 - user interface 2-6

G

- glossary Glossary-1
- gtc command 5-23

H

- hard overrun, see overruns
- he command 5-47
- high-resolution process accounting 1-3

I

- idle time 4-3, 4-6, 6-23

interactive mode 5-4
 interrupt time include/exclude 4-1, 5-42, 6-63, 6-64,
 7-70, 7-71
 iteration 4-1, 4-2

L

last time 4-2
 list all schedulers on system 5-21
 load balancing 4-5
 ls command 5-21

M

major frame 2-2, 4-1, 5-18, 6-11, 7-9
 manual structure iii
 maximum
 cycle cycle 4-2, 5-43, 6-55, 6-58, 7-63, 7-65
 cycle frame 4-2, 5-43, 6-55, 6-58, 7-63, 7-65
 cycle time 4-2, 5-43, 6-55, 6-58, 7-63, 7-65
 frame frame 4-2, 5-43, 6-55, 6-58, 7-63, 7-65
 frame time 4-2, 5-43, 6-55, 6-58, 7-63, 7-65
 minimum
 cycle cycle 4-2, 5-43, 6-55, 6-58, 7-63, 7-65
 cycle frame 4-2, 5-43, 6-55, 6-58, 7-63, 7-65
 cycle time 4-2, 5-43, 6-55, 6-58, 7-63, 7-65
 frame frame 4-2, 5-43, 6-55, 6-58, 7-63, 7-65
 frame time 4-2, 5-43, 6-55, 6-58, 7-63, 7-65
 minor cycle 2-2, 4-2, 5-18, 6-11, 7-9
 modify
 CPU bias 5-32, 6-38, 7-47
 process priority 5-32, 6-38, 6-44, 7-47
 scheduler permissions 5-12, 6-5, 7-5
 scheduling policy 5-32, 6-38, 6-44, 7-47
 timing mode 5-42, 6-64, 7-71

N

namepid 6-28
 nametopid 2-6, 6-28, 7-27
 NightSim RT iii, 1-1
 NightView RT 2-6

O

obtain
 minor cycle/major frame count 5-18, 6-11, 7-9
 performance monitor values 5-43, 6-55, 6-58, 6-60,
 7-63, 7-65, 7-67
 process ID 6-14, 6-28, 7-27

process information 5-35, 6-32, 6-41, 7-16, 7-28,
 7-38, 7-44, 7-49
 process initiation parameter 7-41
 real-time clock settings 5-23, 6-15, 7-11
 scheduler identifier for a key 6-16, 7-12
 scheduler information 5-19, 6-8, 6-16, 6-17, 7-7,
 7-12, 7-13
 scheduler key list 6-13
 soft overrun status 5-35, 6-36, 7-43
 timing mode 5-42, 6-63, 7-70
 overruns
 definition 2-5, 4-2, 6-27, 7-26
 set soft limit 5-32, 6-34, 7-42
 soft status 5-35, 6-36, 7-43

P

PAM 1-4
 pam_capability 1-4
 Performance Monitor
 C library call sequence 6-48
 C library routines 6-47, 6-48
 configuration 1-3
 FORTRAN library call sequence 7-56
 FORTRAN library routines 7-55, 7-57
 overview 1-2, 4-1
 rtcp command sequence 5-10
 rtcp commands 5-8
 simulation example 4-5
 start/stop 5-40, 6-52, 6-53, 7-60, 7-61
 timing mode 4-1, 6-63, 6-64, 7-70, 7-71
 user interface 4-5
 values
 clear 5-38, 6-49, 6-51, 7-57, 7-59
 definition 4-1, 4-2
 obtain 5-43, 6-55, 6-58, 6-60, 7-63, 7-65, 7-67
 period 2-2
 permissions
 FBS 1-4
 scheduler 5-12, 5-13, 6-5, 7-5
 pgmquery 7-28
 pgmremove 6-29, 7-30
 pgmreschedule 7-32
 pgmschedule 7-35
 pgmstat 7-38
 pgmtrigger 6-31, 7-40
 Pluggable Authentication Module (PAM) 1-4
 pm command 5-40
 pmclrpqm 6-49, 7-57
 pmclrtable 6-51, 7-59
 pmmonitor 6-52, 7-60
 pmprogram 6-53, 7-61
 pmqrycpu 6-55, 7-63

pmqrylist 6-58, 7-65
pmqrypgm 6-60, 7-67
pmqrytimer 6-63
pmquerytimer 7-70
pmselect 6-64, 7-71
privileges, see permissions
process rescheduling 5-28, 6-44, 7-32, 7-52
process scheduling 2-2, 5-32, 6-23, 6-38, 7-23, 7-35,
7-47
process sleep 6-27, 7-26

Q

query
FBS kernel configuration 6-7
performance monitor values 5-43, 6-55, 6-58, 6-60,
7-63, 7-65, 7-67
process information 5-35, 6-32, 6-41, 7-16, 7-28,
7-38, 7-44, 7-49
real-time clock 5-23, 6-15, 7-11
scheduled processes 5-35, 6-32, 6-41, 7-16, 7-38,
7-44, 7-49
scheduler configuration 5-19, 6-8, 7-7
soft overrun processing 5-35, 6-36, 7-43
timing mode 5-42, 6-63, 7-70

R

rc command 5-21
RCIM
device files 1-4
timing devices 1-2, 3-3
real-time clock
attach 5-11, 6-6, 7-6
C routine summary 3-3
detach 5-15, 6-12, 7-10
device special files 3-2
obtain settings 5-23, 6-15, 7-11
overview 3-1
procedures 3-2
rtcp command summary 3-3
set 5-22, 6-25, 7-25
start 5-21, 6-22, 7-22
stop 5-22, 6-22, 7-22
user interface 3-2
related publications iv
remove
a process from a scheduler 5-26, 6-29, 7-30
a scheduler 5-16, 6-20, 7-19
reschedule a process 5-28, 6-44, 7-32, 7-52
resume command 5-25
resume frequency-based scheduling 5-25, 6-19, 6-21,

7-15, 7-20
rmp command 5-26
rms command 5-16
rsp command 5-28
rtcp
command script 5-3, A-1
command sequence
FBS 5-9
PM 5-10
command summary 5-7
commands
ats 5-11
chs 5-12
cpm 5-38
cs 5-13
dts 5-15
ex 5-47
gtc 5-23
he 5-47
ls 5-21
pm 5-40
rc 5-21
resume 5-25
rmp 5-26
rms 5-16
rsp 5-28
sc 5-22
sp 5-32
start 5-24
stc 5-22
stop 5-25
svs 5-17
vc 5-18
vcm 5-42
vp 5-35
vpm 5-43
vs 5-19
errors B-1
example script A-1
execution modes 5-2-5-4
exit 5-47
help 5-5, 5-47
overview 5-1
rtparm 7-41

S

save scheduler configuration 5-3, 5-17
sc command 5-22
sched_fbsqry 6-32
sched_pgm_set_soft_overrun_limit 6-34, 7-42
sched_pgm_soft_overrun_query 6-36, 7-43
sched_pgmadd 6-38

sched_pgmqry 6-41
 sched_pgmresched 6-44
 schedfbsqry 7-44
 schedpgmadd 7-47
 schedpgmqry 7-49
 schedpgmresched 7-52
 schedule a process 2-2, 5-32, 6-23, 6-38, 7-23, 7-35,
 7-47
 scheduler configuration 5-13, 6-8, 7-7
 scheduler frequency 2-2
 set a real-time clock 5-22, 6-25, 7-25
 set soft overrun limit 5-32, 6-34, 7-42
 soft overrun, see overruns
 software requirements 1-2
 sp command 5-32
 spare time 4-3, 4-6
 start
 frequency-based scheduling 5-24, 6-19, 7-15
 performance monitoring 5-40, 6-52, 6-53, 7-60,
 7-61
 real-time clock 5-21, 6-22, 7-22
 start command 5-24
 starting base cycle 2-2
 stc command 5-22
 stop
 frequency-based scheduling 5-25, 6-19, 7-15
 performance monitoring 5-40, 6-52, 6-53, 7-60,
 7-61
 real-time clock 5-22, 6-22, 7-22
 stop command 5-25
 svcs command 5-3
 syntax notation iv

T

timing mode 4-1, 5-42, 6-63, 6-64, 7-70, 7-71
 timing source
 attach 5-11, 6-6, 7-6
 detach 5-15, 6-12, 7-10
 overview 3-1
 privileges 1-4
 total iterations 4-2
 total time 4-2
 trigger a process 6-26, 6-31, 7-40

U

unscheduled processes 4-4

V

vc command 5-18
 vcm command 5-42
 vp command 5-35
 vpm command 5-43
 vs command 5-19

W

wait on a frequency-based scheduler 6-27, 7-26
 wake a sleeping process 6-26, 6-31, 7-40

Spine for 1.5" Binder

**Product Name: 0.5" from
top of spine, Helvetica,
36 pt, Bold**

**Volume Number (if any):
Helvetica, 24 pt, Bold**

**Volume Name (if any):
Helvetica, 18 pt, Bold**

**Manual Title(s):
Helvetica, 10 pt, Bold,
centered vertically
within space above bar,
double space between
each title**

**Bar: 1" x 1/8" beginning
1/4" in from either side**

**Part Number: Helvetica,
6 pt, centered, 1/8" up**

RedHwak Linux OS

Progr

**Frequency-Based
Scheduler**

0898005

